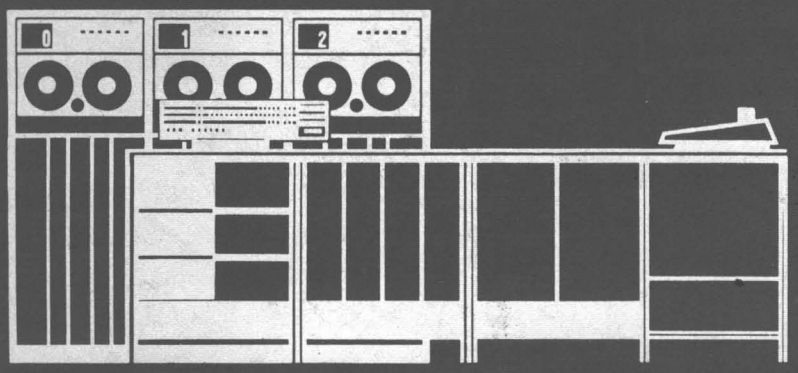
R. CRANE
2330 SUMPTER DR.
COQUITLAM, BC.
936-4700

TRANSITION TO EASYCODER

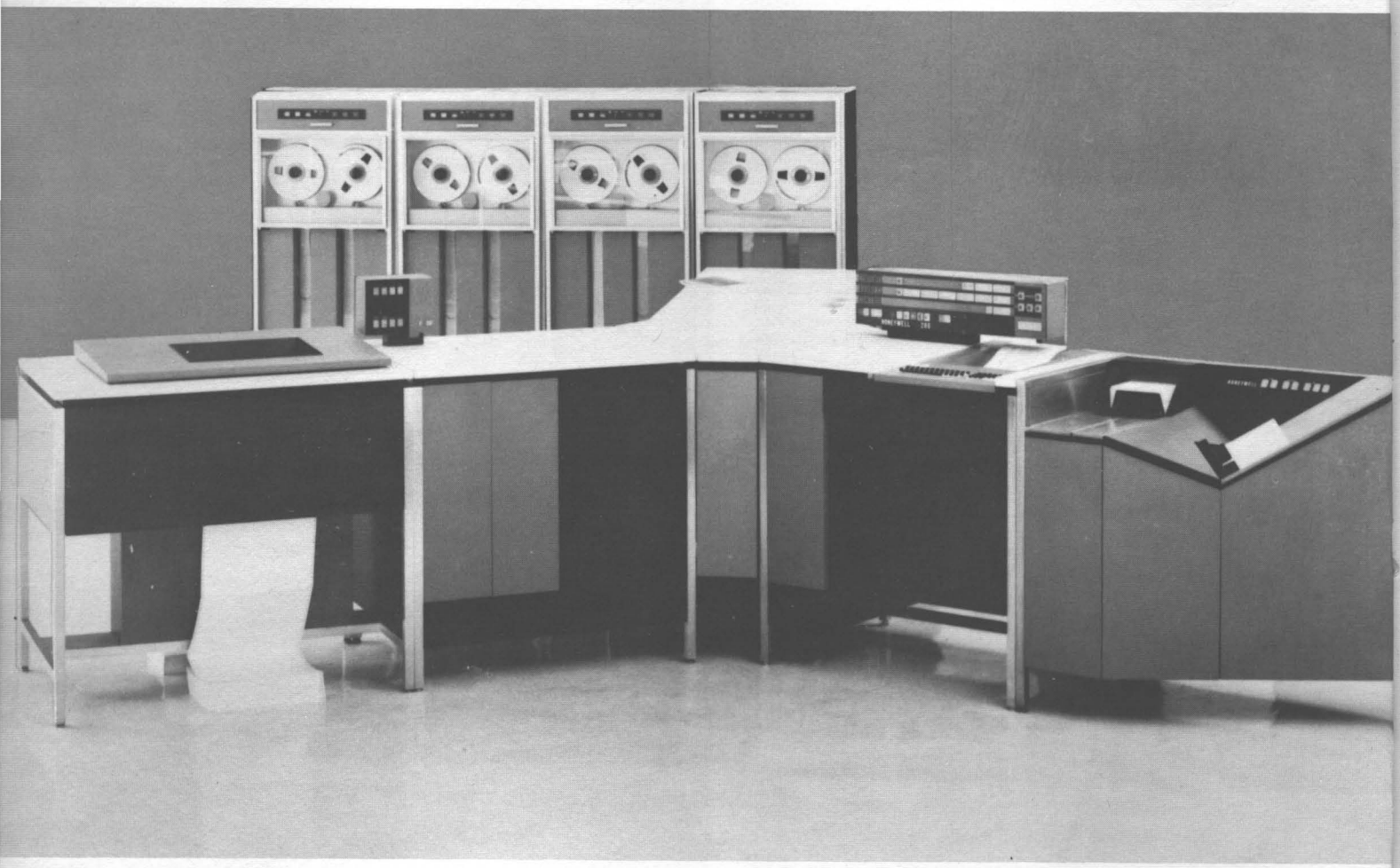
A Programmed Text



Honeywell

ELECTRONIC DATA PROCESSING

Thoth, symbolic Egyptian god of wisdom and learning

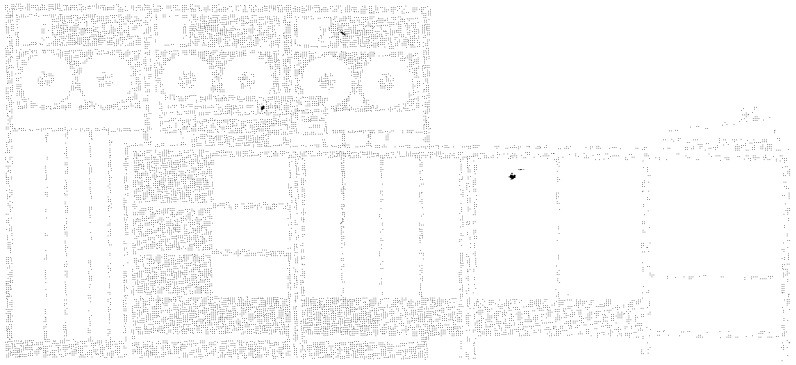


Printed in U.S.A.
NP-2939

DSI-319
3964

TRANSITION TO EASYCODER

A Programmed Text

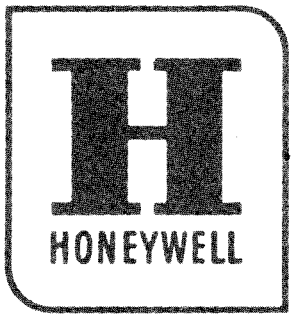


By
John E. Harrah
and
Harris J. Hulburt
Programmed Instruction Development

Honeywell
ELECTRONIC DATA PROCESSING

PRICE $4.50

Questions and comments regarding this manual should be addressed to:
Honeywell Electronic Data Processing
Programmed Instruction Development
60 Walnut Street
Wellesley Hills, Massachusetts 02181



UNREVISED FIRST EDITION
First Printing, September, 1964

FOREWORD

This manual is specifically written for the reader whose prior programming experience included a 1401 system. The intent of this manual is to introduce Easycoder language, provide familiarization with Honeywell 200 computer capabilities, describe programming procedures, and define Honeywell terminology.

This manual is designed to be used as a general introduction and/or a classroom text. The basic organization of lessons is outlined below:

Lesson I:	Introduction to Easycoder
Lessons II and III:	H-200 Hardware
Lesson IV:	Numbering Systems and Honeywell Alphanumeric Code
Lesson V:	Storage, Retrieval, and Execution
Lessons VI, VII, and VIII:	Easycoder Programming

NOTE: Lesson IV is presented in two parts. Part I - Numbering Systems and Part II - Honeywell Alphanumeric Code. Selective utilization of portions or all of Part I may be made at the discretion of the reader as determined by subject matter familiarity.

Lessons VI, VII, and VIII concern assembly control statements, data formatting statements and data processing statements. Descriptions and reference tables for these statements are also included in the Honeywell H-200 Programmers' Reference Manual (DSI-214).

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

LIST OF ILLUSTRATIONS

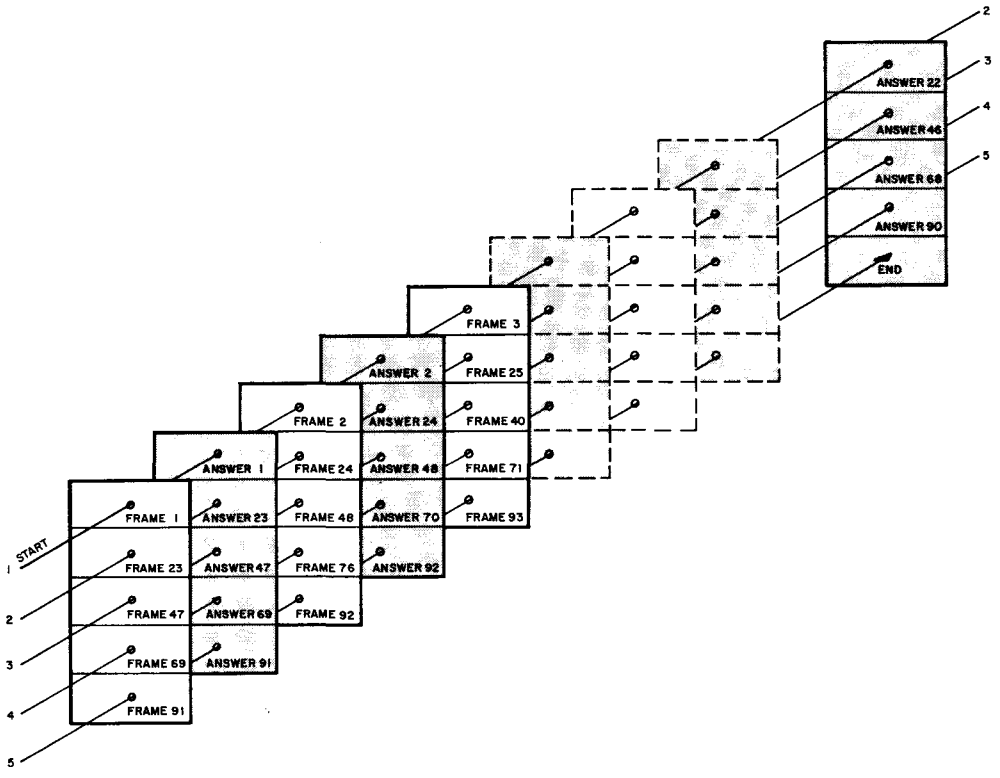
INTRODUCTION

As a programmer, you will soon be working with a new but not entirely unfamiliar computer system. Due to similarities with your previous system, the Honeywell 200's basic orientation provides an initial foundation for understanding. The purpose of this programmed text is to assist you in gaining insight to the extended scope and advanced performance capabilities of the H-200 system.

A programmed text is designed to encourage your active interaction and participation with the information being presented. In the remainder of this book, you will be given questions to answer and statements to complete concerning the reading material. While you should feel free to make any desired notes, it is important to the success of this teaching method that you:

1. Follow instructions.
2. Write responses as required.
3. Check answers.
4. Correctly re-write any wrong responses.
5. Take your time.

With this book you will be in the interesting position of being both the teacher and the student. The diagram below illustrates page format and how you are to proceed from page to page <u>rather than</u> down a page. Exceptions to this format will be stated.



LESSON I
INTRODUCTION TO EASYCODER LANGUAGE

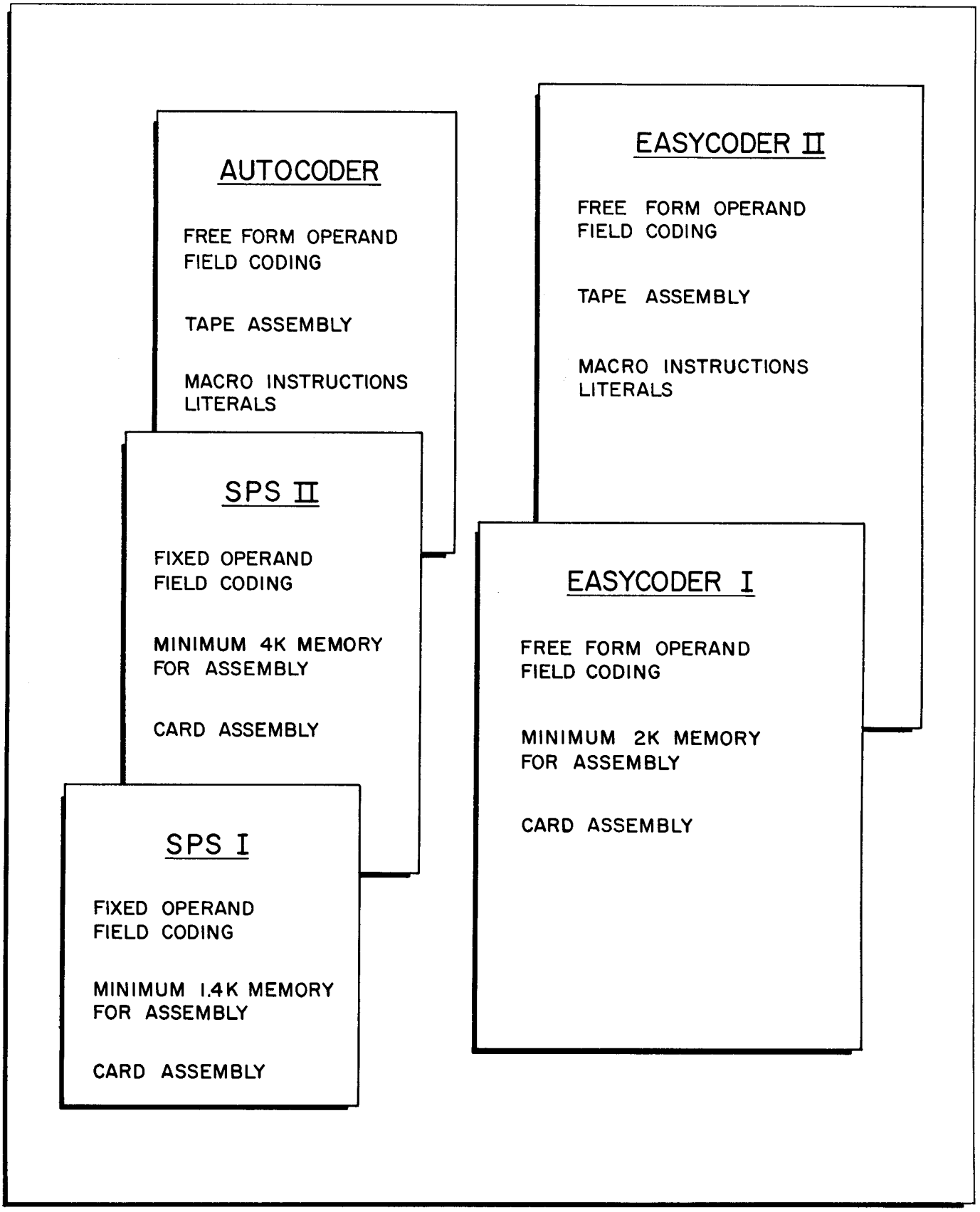


Figure 1. General Language Comparison

1. TRANSITION TO EASYCODER - PROGRAMMED TEXT

Being synonymous with "book," the word TEXT in the title above should not require further explanation.

CHECK THE WORD YOU WROTE IN THE BLANK BY TURNING THE PAGE.

1. TEXT

This first "frame" demonstrates how you are to use this book. Write your responses in the blanks, then check them by turning the page to see the answer.

Continue to frame 2 on the next page.

2. While "text" is easily understood to mean "book," a programmed text is a special kind of book. As demonstrated by the first frame, a blank in a sentence allows you to write a RESPONSE. These responses are then checked by TURNING the page.

---

8. The use of Basic or Extended Easycoder depends on whether the assembly program is on punched cards or magnetic tape.

The assembly program for BASIC Easycoder is on PUNCHED CARD, the assembly program for EXTENDED Easycoder is on MAGNETIC TAPE.

---

14. ASSEMBLY CONTROL STATEMENTS are listed below in mnemonic form. Copy them on the notepaper just titled.

PROG	EQU
ORG	CEQU
MORG	HSM
ADMODE	CLEAR
EX	END

On the notepaper, write the complete word beside each mnemonic you recognize from your previous SPS or AUTOCODER experience.

---

20. The H-200 has two outstanding arithmetic capabilities not found in your previous equipment.

These are: Binary Addition, whose mnemonic is BA.
Binary Subtraction, whose mnemonic is BS.

By listing the mnemonics above with the arithmetic mnemonics used with SPS or AUTOCODER, your notes will show the full Easycoder arithmetic instructions.

---

26. Three of the five types of Data Processing Statements have been introduced. They are:

(1). ARITHMETIC INSTRUCTIONS
(2). LOGIC INSTRUCTIONS
(3). INPUT / OUTPUT INSTRUCTIONS

The remaining two types of Easycoder Data Processing Statements deal with:

(4). Editing
(5). Control (Setting WORD MARKS etc.)

2.

RESPONSE (ANSWER)
TURNING

Several equivalent responses may be possible. Occasionally, alternate responses will be given in parentheses following the preferred response. Use reasonable judgement in deciding whether your response agrees with the printed answer. If it does not agree, return and correct your response.

CONTINUE TO FRAME 3

8.

BASIC - PUNCHED CARDS
EXTENDED - MAGNETIC TAPE

---

14. PROGRAM-Operand field entry titles program listing.

*ORIGIN-Tells assembly program beginning assignment of sequential addresses.

MODULAR ORIGIN-Similar to above. Multiple of assigned address.

ADDRESS MODE-Addresses to be assembled as 2 or 3 characters.

*EXECUTE-Partial program execution during loading.

EQUALS-Tag for specified address.

CONTROL EQUALS-Tag for specified characters.

HIGH SPEED MEMORY-Obtains printed listing of memory.

CLEAR-Removes punctuation.

*END-Shows end of source program.

With SPS or AUTOCODER experience, you probably recognized those mnemonics marked with an *. Complete any remaining Assembly Control Statements. Utilization of these Basic Easycoder and additional Extended Easycoder statements are subjects of a later lesson.

20.

BA - BINARY ADDITION
BS - BINARY SUBTRACTION
A - ADDITION
S - SUBTRACTION
ZA - ZERO AND ADD
ZS - ZERO AND SUBTRACT
• M - MULTIPLY
• D - DIVIDE

• Pertains to an optional instruction.

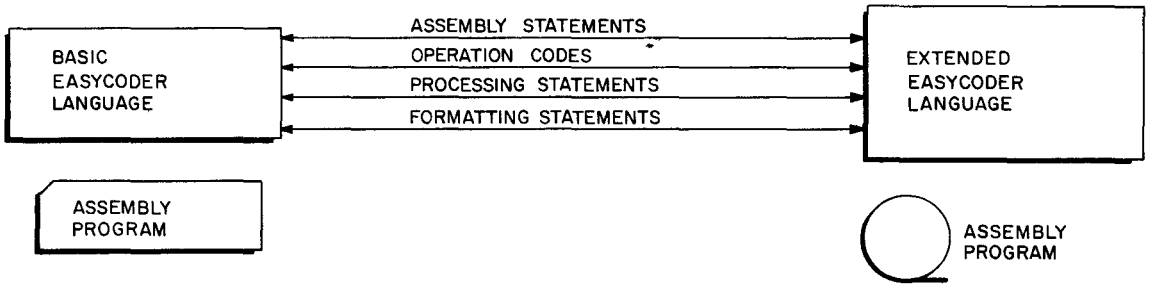
26.

ARITHMETIC
LOGIC
INPUT/OUTPUT

3. A programmed text is NOT designed to be a test that causes you to write answers in "frames." A programmed text DOES present information in easily understood steps called FRAMES that require written RESPONSES to help you remember the information presented.

---

9. The illustration below shows that many elements of BASIC Easycoder language are also found in EXTENDED EASYCODER LANGUAGE.



---

15. In addition to Assembly Control Statements, Easycoder also uses DATA FORMATTING STATEMENTS and DATA PROCESSING STATEMENTS.

Reserving a work area in memory or storing a constant are examples of DATA FORMATTING STATEMENTS.

---

21. The second group of instructions under Data Processing Statements pertain to logic functions such as branching and comparing.

Properly title this section of your notes, copy the mnemonics below and write the full word for each that you recognize.

EXT	B
HA	BCC
C	BCT
SST	BCE

---

27. As can be seen on the notes, space is provided for one Editing Instruction mnemonic. Since editing is a dual process of Move Characters and Edit, the mnemonic is MCE.

This instruction is used to insert identifying symbols, punctuation, and to suppress unwanted zeros in a data field.

3.

FRAMES
RESPONSE

As you know from previous experience, something you write is easier to remember than something you have simply read. In addition, a programmed text lets you check responses immediately. If you ever happen to write a wrong response, you should correct it immediately.

9.

BASIC
EXTENDED EASYCODER LANGUAGE

15.

DATA FORMATTING STATEMENTS

21.

EXTRACT
HALF ADD
COMPARE
SUBSTITUTE

BRANCH
BRANCH ON CHARACTER CONDITION
BRANCH ON CONDITION TEST
• BRANCH IF CHARACTER EQUAL

• Pertains to an optional instruction.

27.

MCE

4. One more frame about a programmed text before discussing the title TRANSITION TO EASYCODER. You should be able to write correct responses because you already know them, or because the words are presented in the same frame, or because the words have been presented in a PREVIOUS FRAME.

---

10. The general difference then between Basic and Extended Easycoder language is whether the assembly program is on PUNCHED CARD or MAGNETIC TAPE. Consequently many of the additional instructions simply provide extended control of the ASSEMBLY program due to its more versatile storage media.

---

16. For instance a programmer can reserve an 80 character card input area and assign it a symbolic address (such as CARDIN) without knowing the actual address of the field. The Easycoder mnemonic RESV is the DATA FORMATTING statement to accomplish this example. The full word for RESV of course is RESERVE.

---

22. As you know from SPS or AUTOCODER, a "d character" modifies and extends basic instructions. Example: (BASIC INSTRUCTION) B xxx blank
(MODIFIED WITH "d character" Z) as B xxx Z
becomes a BAV-Branch on Arithmetic Overflow.

Easycoder considers modifications of this sort as providing a VARIANT of a basic instruction. Consequently, Easycoder modifying characters are referred to as VARIANT characters.

---

28. The fifth entry under Data Processing Statements on the notes refers to instructions which control the H-200. As such, they should be titled CONTROL INSTRUCTIONS.

4.

PREVIOUS FRAME
(PRECEDING)
(PRIOR, ETC.)

You will decide how rapidly to progress through the frames. If a blank appears difficult to fill in, perhaps you need to pause and consider what you have learned, or reread the frame, or possibly review a few previous frames. In any case, the pace of proceeding through the text is up to you.

10.

PUNCHED CARDS
MAGNETIC TAPE
ASSEMBLY

16.

DATA FORMATTING
RESERVE

22.

VARIANT

28.

CONTROL

5. Now, about TRANSITION TO EASYCODER:

Your previous computer system consisted of hardware and its software in either SPS or AUTOCODER symbolic language. Since you are now progressing to Honeywell 200 hardware, it is necessary to learn about its SOFTWARE written in EASYCODER symbolic LANGUAGE.

---

11. EASYCODER language is classified into three categories:

1. ASSEMBLY CONTROL STATEMENTS
2. DATA FORMATTING STATEMENTS
3. DATA PROCESSING STATEMENTS

In your previous systems terms, "Processor Control Operations" correspond with EASYCODER ASSEMBLY CONTROL statements. Similarly,

(SPS) Area Definition or (AUTO) Declarative Operations } are like DATA FORMATTING. statements.

(SPS) Instructions or (AUTO) Imperative Operations } are like DATA PROCESSING statements.

---

17. The Easycoder mnemonics below are to be added to your notepaper under the second title DATA FORMATTING STATEMENTS.

DCW
DC
RESV
DSA
DA

Due to your SPS or AUTOCODER background you should probably be able to write the full word for each mnemonic on your notes.

---

23. An advantage of the Easycoder variant character is that one or more can modify and further specify the operation to be performed. In this manner, a single Easycoder instruction may have none, one, or as many VARIANT CHARACTERS as required.

---

29. Many of the control mnemonics refer to H-200 features not found in your previous equipment. For this reason, the complete words for several of the mnemonics have been entered on the notes. Write the complete words for the remaining mnemonics you recognize.

5.

SOFTWARE
EASYCODER
LANGUAGE

11.

SPS or AUTOCODER	EASYCODER
Processor Control Operations	ASSEMBLY CONTROL STATEMENTS
(SPS) Area Definition or (AUTO) Declarative Operations	DATA FORMATTING STATEMENTS
(SPS) Instructions or (AUTO) Imperative Operations	DATA PROCESSING STATEMENTS

17.

DATA FORMATTING

DCW-DEFINE CONSTANT WITH WORD MARK
DC-DEFINE CONSTANT WITHOUT WORD MARK
RESV-RESERVE
DSA-DEFINE SYMBOL ADDRESS
• DA-DEFINE AREA

• =Extended Easycoder

23.

VARIANT CHARACTERS

29.

SW-SET WORD MARK
CW-CLEAR WORD MARK
H-HALT
NOP-NO OPERATION
MCW-MOVE CHARACTERS TO WORD MARK
LCA-LOAD CHARACTERS TO A FIELD WORD MARK

While much remains to be presented concerning how the statements on your notes are used, the following two frames show what has been taught in this section of the programmed text.

6. The term EASYCODER was chosen for H-200 symbolic programming language for two reasons:

   1. The H-200 uses an E fficient A ssembly SY stem.
   2. The mnemonic code used by the programmer is not difficult, therefore it is an EASY CODE to learn and use.

---

12. Each of the three classifications of Easycoder is discussed in following frames.

    Those Easycoder statements which control the assembly program are known as ASSEMBLY CONTROL STATEMENTS.

---

18. As stated earlier, Easycoder language is classified as three kinds of statements:

    1. assembly control statements.
    2. data formatting statements.

    And those statements for processing data, simply called data processing statements.

---

24. Easycoder's use of as many variant characters as required in a single instruction greatly reduces the number of basic INPUT/OUTPUT INSTRUCTIONS.

    Where more than ten SPS System Control Instructions
    or
    more than fifty AUTOCODER I/O Commands are used,

    Easycoder only needs two INPUT / OUTPUT INSTRUCTIONS and their appropriate variants.

---

30. Sometimes the terms EASYCODER I and EASYCODER II may be used for brevity. EASYCODER I refers to BASIC EASY CODER and Easycoder II refers to EXTENDED EASY CODER. The word EASYCODER by itself usually implies both BASIC and EXTENDED EASYCODER language.

    The assembly program for EASYCODER I is on PUNCHED CARDS.
    The assembly program for EASYCODER II is on MAGNETIC TAPE.

6.

EASY CODE

12.

ASSEMBLY CONTROL STATEMENTS

18.

ASSEMBLY CONTROL
DATA FORMATTING
DATA PROCESSING

24.

INPUT/OUTPUT INSTRUCTIONS

30.

BASIC EASYCODER
EXTENDED EASYCODER
BASIC (I)
EXTENDED (II)
PUNCHED CARDS
MAGNETIC TAPE

7. H-200 symbolic programming language is of two types. A basic computer system (assembly program on punched cards) uses BASIC EASYCODER symbolic programming language. Similarly, an extended computer system (assembly program on magnetic tape) employs EXTENDED EASYCODER symbolic programming language.

---

13. Assembly Program Control Statements can be compared to "PROCESSOR CONTROL OPERATIONS" used with your previous system. Examples are mnemonics such as:

ORG ORIGIN
END END
EX EXECUTE

Write the complete word for each mnemonic above.

---

19. Complete the third title on your notepaper.

Rather than adding all the Easycoder mnemonics under this last title, it is better at this time to separate them into five groups according to function.

Since the first group deals with arithmetic, the first entry in your notes should be simply ARITHMETIC INSTRUCTIONS.

---

25. Only two INPUT/OUTPUT mnemonics are required with Easycoder.

Peripheral Data Transfer

and

Peripheral Control and Branch

The mnemonics are: PDT and PCB.

Add them to the notes.

---

31. EASYCODER consists of three kinds of statements, they are:

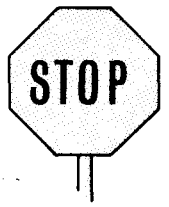
1. ASSEMBLY CONTROL STATEMENTS.
2. DATA FORMATTING STATEMENTS.
3. DATA PROCESSING STATEMENTS.

The five types of instructions are:
a. [illegible]
b. LOGIC
c. INPUT/OUTPUT
d. EDITING
e. CONTROL

In EASYCODER, a VARIANT CHARACTER corresponds to a "d character" except that MORE THAN ONE VARIANT CHARACTER CAN BE USED TO MODIFY OR EXTEND FUNCTION OF THE INSTRUCTION

7.

EXTENDED EASYCODER



(Return to page 5, frame 8.)

13.

ORIGIN
END
EXECUTE

At this time you will begin a set of notes to construct an overview of Easycoder language. Remove the perforated sheet of paper at the right and complete the first title (statements that control the assembly program).

(Return to page 5, frame 14.)

19.

ARITHMETIC

(Return to page 5, frame 20.)

25.

PDT
PCB

(Return to page 5, frame 26.)

31.

1. ASSEMBLY CONTROL STATEMENTS
2. DATA FORMATTING STATEMENTS
3. DATA PROCESSING STATEMENTS

(INSTRUCTIONS)
a. ARITHMETIC
b. LOGIC
c. INPUT/OUTPUT
d. EDITING
e. CONTROL

VARIANT CHARACTER

MORE THAN ONE VARIANT CHARACTER MAY BE USED TO MODIFY OR FURTHER SPECIFY AN INSTRUCTION. (Or equivalent answer.)

(Continue to page 17.) ——————→

1. (PROGRAM) ASSEMBLY CONTROL STATEMENTS

PROG - OPERAND FIELD ENTRY TITLES PROG LISTIN!
ORG - (ORIGIN) TELLS ASSEMBLY PROGRAM BEGINNING ASSIGNMENT OF SEQUENTIAL ADDRESSES
MORG - (MODULAR ORIGIN) MULTIPLE OF ASSIGNED ADDRESS
ADMODE - (ADDRESS MODE) ADDRESSES TO BE ASSEMBLED AS 2 OR 3 CHARACTERS
PARTIAL PROGRAM EX - EXECUTION DURING LOADING (EXECUTE)
EQU - (EQUALS) ADDRESS
TAG FOR SPECIFIED CONTROL CHARACTERS
CEQU - (CONTROL EQUATE) TAG FOR SPECIFIED CHARACTERS
HSM - OBTAINS PRINTED LISTING OF (HIGH SPEED MEMORY) MEMORY!
CLEAR - REMOVES PUNCTUATION
END - END OF SOURSE PROG

2. DATA FORMATTING STATEMENTS

DCW - Define constant with word mark
DC. ✓ ✓ without ✓ ✓
RESV - Reserve
DSA - define symbolic address
DA. define area

3. DATA PROCESSING STATEMENTS

(1) ARITHMETIC INSTRUCTIONS

BA	ZA
BS	ZS
A	M
S	D

(2) LOGIC INSTRUCTIONS

EXT EXTRACT	B - BRANCH
HA - HALF ADD	BCC - BRANCH ON CHARACTER CONDITION
C - COMPARE	BCT - BRANCH ON CONDITION TEST
SST - SUBSTITUTE	BCE - BRANCH IF CHARACTER EQUAL

(3) INPUT / OUTPUT INSTRUCTIONS

PDT - PERIPHERAL DATA TRANSFER
PCB - PERIPHERAL CONTROL & BRANCH

(4) EDITING INSTRUCTION

MCE

(5) CONTROL INSTRUCTIONS

SW - SET WORD MARK
SI - SET ITEM MARK
CW - CLEAR WORD MARK
CI - CLEAR ITEM MARK
H - HALT
NOP - NO OPERATION
• CSM - CHANGE SEQUENCING MODE
• CAM - CHANGE ADDRESS MODE
• RNM - RESUME NORMAL MODE
MCW - MOVE CHARACTER TO WORD MARK
• EXM - EXTENDED MOVE
• MAT - MOVE AND TRANSLATE
LCA - LOAD CHARACTERS TO A FIELD WORD MARK
SCR - STORE CONTROL REGISTERS
• LCR - LOAD CONTROL REGISTERS

• = ADVANCED PROGRAMMING OPTION

This page is intended to provide only an introduction or overview of the elements in Easycoder language. Detailed discussion of those statements that appear unfamiliar or different will be found in later lessons. Retain this page for future reference to the material above as well as for information on the reverse side.

NUMERIC ONLY GROUP "0"			12 ZONE & NUMERIC GROUP "1"			11 ZONE & NUMERIC GROUP "2"			0 ZONE & NUMERIC GROUP "3"		
	B A	8 4 2 1		B A	8 4 2 1		B A	8 4 2 1		B A	8 4 2 1
0 =	0 0	0 0 0 0	A =	0 1	0 0 0 1	J =	1 0	0 0 0 1	S =	1 1	0 0 1 0
1 =	0 0	0 0 0 1	B =	0 1	0 0 1 0	K =	1 0	0 0 1 0	T =	1 1	0 0 1 1
2 =	0 0	0 0 1 0	C =	0 1	0 0 1 1	L =	1 0	0 0 1 1	U =	1 1	0 1 0 0
3 =	0 0	0 0 1 1	D =	0 1	0 1 0 0	M =	1 0	0 1 0 0	V =	1 1	0 1 0 1
4 =	0 0	0 1 0 0	E =	0 1	0 1 0 1	N =	1 0	0 1 0 1	W =	1 1	0 1 0 1
5 =	0 0	0 1 0 1	F =	0 1	0 1 1 0	O =	1 0	0 1 1 0	X =	1 1	0 1 1 1
6 =	0 0	0 1 1 0	G =	0 1	0 1 1 1	P =	1 0	0 1 1 1	Y =	1 1	1 0 0 0
7 =	0 0	0 1 1 1	H =	0 1	1 0 0 0	Q =	1 0	1 0 0 0	Z =	1 1	1 0 0 1
8 =	0 0	1 0 0 0	I =	0 1	1 0 0 1	R =	1 0	1 0 0 1			
9 =	0 0	1 0 0 1									

HONEYWELL ALPHANUMERIC CODE

OCTAL-BINARY CROSS REFERENCE

EXAMPLE: Decode BINARY 101 011

or OCTAL 5 3

Locate first three digits in the left vertical column.
Locate second three digits in top horizontal column.

SECOND THREE BITS

Octal → / Binary ↓		0	1	2	3	4	5	6	7
	Binary	000	001	010	011	100	101	110	111
0	000	0	1	2	3	4	5	6	7
1	001	8	9	'	=	:	blank	>	&
2	010	+	A	B	C	D	E	F	G
3	011	H	I	;	.	)	%	■	?
4	100	-	J	K	L	M	N	O	P
5	101	Q	R	#	$	*	"	≠	:
6	110	<	/	S	T	U	V	W	X
7	111	Y	Z	@	,	(	CR	■	¢

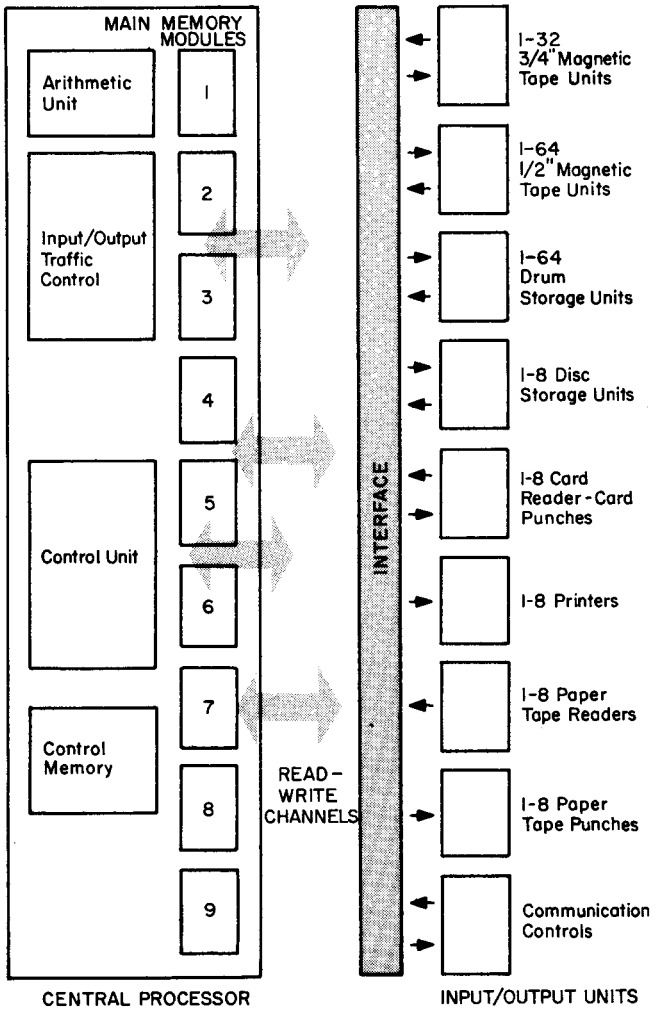
FIRST THREE BITS

LESSON II
H-200 HARDWARE

H-200 Hardware

Section I provided an overview of Easycoder language and served to introduce some of the "differences" encountered when programming for the H-200. Certain statements, recognized as "new" or unfamiliar, actually reflect features not found in your previous equipment. Examples are: BA, EXT, HA, ADMODE, CAM, CSM, SI, etc.

While it is realized that your primary interest is with the language of the computer, knowledge of the computer itself serves as a foundation for programming skills. Section II of this programmed text is devoted to comparing the H-200 to your previous equipment. Machine features that provide greater flexibility of programming and simultaneous performance of peripheral operations are introduced in this section.



CENTRAL PROCESSOR--The minimum 2048 character positions can be expanded as needed in modular increments. The first added increment is a 2048-character module. Additional modules of 4096 characters each can be added for a total of 32,768 character positions. (Double the storage capacity of the 1401.)

CONTROL MEMORY--Unlike most small computers, the Honeywell 200 contains a control memory which complements the main memory. There are up to 16 storage registers available, each capable of storing one main memory character address.

INPUT/OUTPUT TRAFFIC CONTROL--This central processor element directs simultaneous computing and multiple peripheral operations. The Honeywell 200 is equipped with three read-write channels which feed data to, and accept data from, input-output trunks connected to the peripheral equipment. With these three channels and the minimum eight bi-directional trunks, three peripheral operations can be performed simultaneously with central processor computation. Optional: eight more input-output trunks plus an auxiliary read-write channel.

(Refer to the chart below to complete and check statements on page 21.)

SYSTEM SIMILARITIES

FEATURE	HONEYWELL 200	IBM 1401
BASIC ORGANIZATION	Character oriented	Character Oriented
DATA	Variable Length Fields	Variable Length Fields
INSTRUCTIONS	Variable Length	Variable Length
FIELDS	Word Mark Defines Field	Word Mark Defines Field
RECORDS	Record Mark Defines Records	Record Mark Defines Records
INFORMATION UNITS	Character: 6 Data, 1 Parity, and 2 Punctuation Bits.	Character: 6 Data, 1 Parity, and 1 Punctuation Bit.
FIELD LIMIT	Word Mark	Word Mark
ITEM LIMIT	Item Mark	---(Record Mark?)
RECORD LIMIT	Record Mark	---(Group Mark?)
INSTRUCTION FORMAT		
OPERATION CODE	One Character	One Character
A-ADDRESS	2 or 3 Characters	3 Characters
B-ADDRESS	2 or 3 Characters	3 Characters
VARIANT	1 or More Characters	1 "d" Character

NOTE: In this lesson (LESSON II H-200 Hardware) complete an entire page before proceeding. Check answers by referring to charts or illustrations.

(Refer to the chart on page 20 to complete and check these statements.)

1. The first entry shows that both machines are "character oriented." In simplest terms, this means that a single memory location can be accessed and one memory location stores one six bit CHARACTER.

2. The next two entries show that both machines store data and instructions of VARIABLE LENGTH. The number of memory locations require to store data or an instruction therefore equals the number of characters it contains.

3. Another similarity between the two machines is that the limit of a field in memory is defined with a WORD MARK.

4. An important difference exists between the two machines in regard to defining a "Record" with a RECORD MARK. The 1401 uses an additional character whereas the H-200 generates this punctuation as a part of the memory location storing a data character. Therefore the H-200 uses one less memory location than the 1401 each time a "Record" is defined.

5. The entry concerning units of information points out the difference mentioned above. How many bits (cores) are contained in a 1401 memory location? 8. How many are there in an H-200 memory location? 9.

6. In designating the limit of a field, both machines use a WORD MARK. However, the H-200 has one more PUNCTUATION bit per memory location than the 1401, so further grouping of data as an ITEM is possible. It should be noted that the Honeywell Item Marks and Record Marks do not have a direct correspondence with 1401 Record and Group Marks.

7. Concerning instruction formats, the H-200 utilizes an efficient addressing system which permits designation of an address up to #4095 in only two characters (two memory locations). Determine the number of memory locations required for the H-200 and the 1401 to store the instruction shown below. 1401= 7 H-200= 5

OP. CODE	"A" ADDRESS	"B" ADDRESS
A	2411	1124

8. The final point of comparison in the chart at the left was pointed out earlier as it pertained to peripheral instructions. The H-200 only requires two peripheral instructions (PCB and PDT) because they may be further specified as required by appending one or more VARIANT CHARACTER.

Your EASYCODER notes and the preceding chart have shown several programmer oriented H-200 features that are similar to your previous system. Concurrently, a few H-200 features were introduced which enable more efficient and more effective operation. What has not been indicated is the extent of H-200 superiority when both design and performance of the two systems are compared. The table below makes this comparison and it should also suggest some areas that invite your further study to take full advantage of H-200 capability.

	H-200	1401
INTERNAL SPEED		
Cycle Time (microsec.)	2	11.5
A & B→B (5 char.)	44	230.0
Compare A:B (5 char.)	34	207.0
Instructions/Second	25,000	4,600
CONTROL MEMORY		
Access Time	250 nanoseconds (billionths)	
MAIN MEMORY		
Minimum	2000 characters	1400
Maximum	32,000 characters +	16,000
Expandable	YES	NO
Addressing	Binary	Decimal
Indirect Addressing	YES	NO
Arithmetic	Decimal and Binary	Decimal
Sequence Counter	3: Sequence, Cosequence, Interrupt Counter	1
External Interrupt	YES	NO
Index Registers	6	3
PERIPHERAL SIMULTANEITY	Multiple Read-Write-Compute. Up to four peripheral transfer operations together with computing.	Essentially serial. Either Read or Write or Compute.
I/O DEVICES	Up to 16 input or output controls together with their devices. May be attached in any combination e.g. up to 64 magnetic tapes etc.	Maximum: One card reader, one card punch, one printer, six magnetic tapes.

Page 23 illustrates many possible configurations in which the H-200 may operate. The following pages show a simplified H-200 Environments illustration on which you will draw lines as connecting wiring and also refer to the illustration to answer questions.

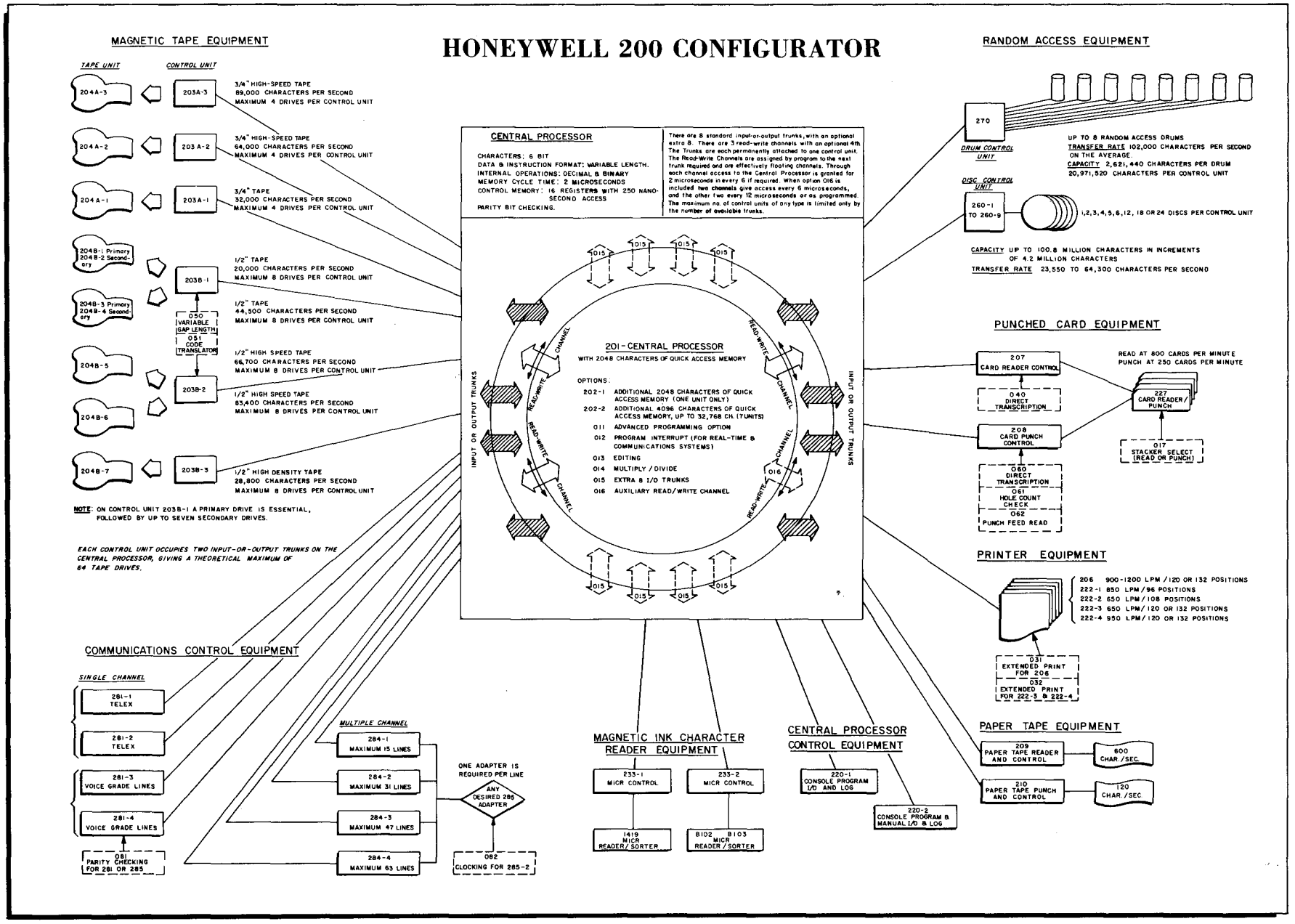


Figure 2. H-200 Configurator

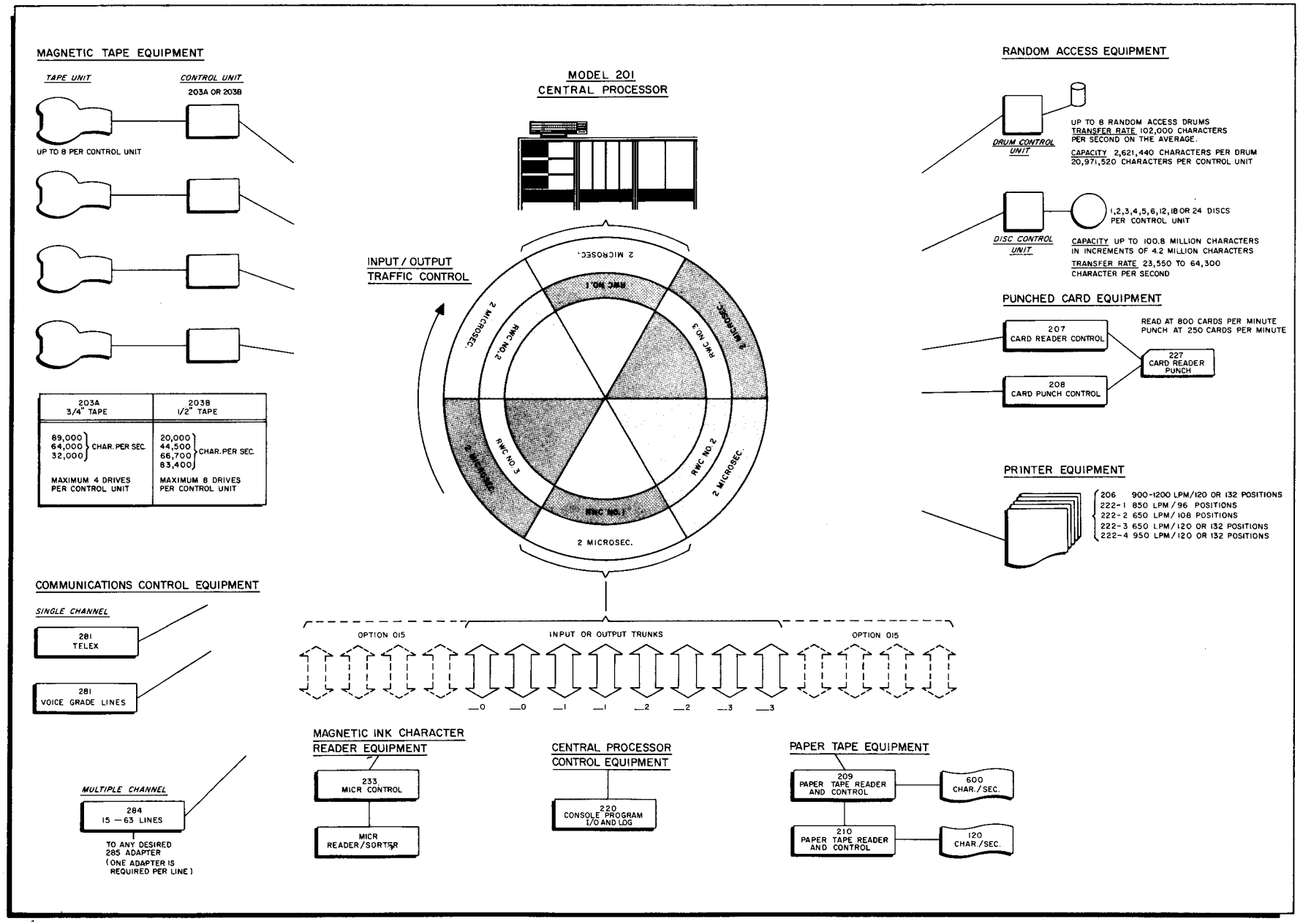


Figure 3. H-200 Environments

(Refer to Figure 3. to answer the following questions.)

1. An important H-200 feature provides simultaneous peripheral operations together with computing. (Shown between the central processor and input/output trunks.) This feature is known as INPUT / OUTPUT TRAFFIC CONTROL.

2. The times illustrated denote that each Read/Write Channel (RWC) is granted 2 microseconds access to the central processor. Since there are three RWC's, each will have access to the central processor once out of every 6 microseconds.

   If an input or output device is not sending or receiving information during a two microsecond RWC period, the time is allotted to the central processor. For example, the mechanical operations of card reading, card punching, and printing a line require: 75 milliseconds (Reading a card at the rate of 800 CPM)
   240 milliseconds (Punching a card at the rate of 250 CPM)
   67 milliseconds (Printing a line at the rate of 900 LPM)

   However, for these three operations, transfer of information either to or from the central processor and devices only requires a total of 19 milliseconds. Because of RWC Traffic Control computations are performed by the central processor during 73% of the time, even when maintaining full rated speeds of: 800 CPM Reading
   250 CPM Punching
   900 LPM Printing

3. Note that peripheral devices may be connected to either INPUT or OUTPUT trunks. Rather than having devices permanently connected to the central processor, they are alternately attached by a Read/Write Channel.

4. While eight optional (015) input or output trunks are available, your present concern will be with the basic eight trunks numbered -0 -0 -1 -1 -2 -2 -3 -3 in the figure above.

5. The number of a trunk should contain two digits. (The second digit identifies the trunks from 0 to 3 as in the figure.) The first digit denotes whether the trunk is being used for INPUT or OUTPUT. Whether a trunk is input from a device or output of the central processor to a device depends upon the type equipment attached. Assigning these first digits to denote input or output for various devices is explained on the next page.

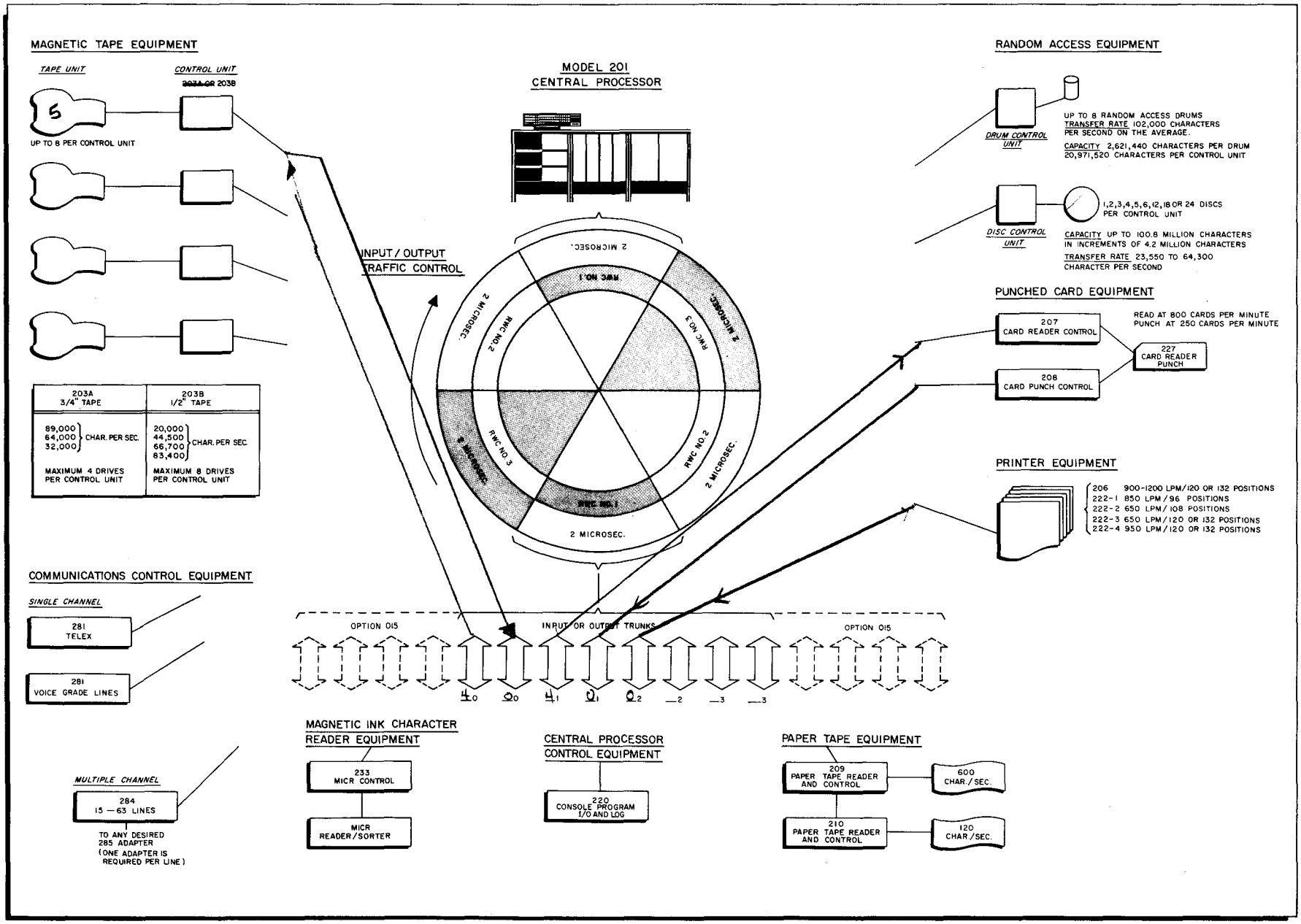


Figure 4. H-200 Environments

1. If the peripheral equipment provides input to the C. P., such as a card reader, the first digit of the trunk designation should be a "4." If the peripheral equipment receives output from the central processor, such as a printer, the first digit should be a "0." An input output device, such as a magnetic tape control unit, must be attached to two trunks; one with a first digit of 4 denoting input and one with a first digit of 0 denoting output.

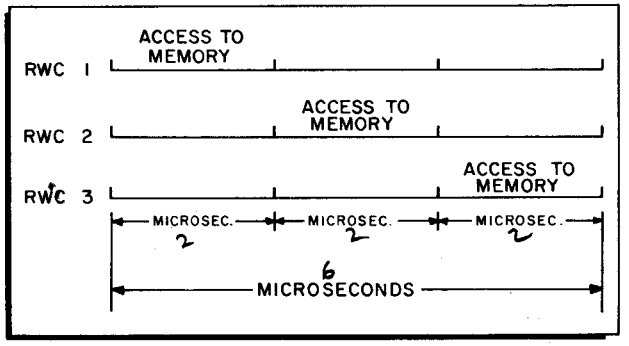
2. The following trunk assignment may be followed for standardization, if desired:

Magnetic Tape (Output)	00
(Input)	40
Card or Paper Tape Reader	41
Card or Paper Tape Punch	01
High Speed Printer	02

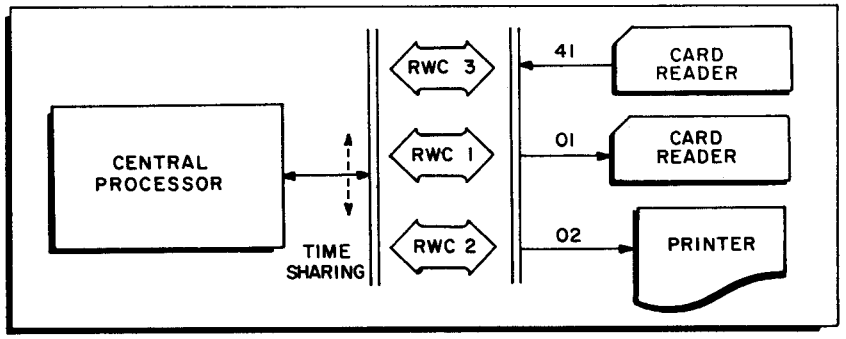
If you know the configuration of your H-200 system, draw lines (as connecting wiring) from the appropriate control units to selected input or output trunks in Figure 4. above. Properly designate the first trunk digit to denote input or output.

If you are not aware of the H-200 configuration with which you will be working, simply assign the eight trunks to selected units, then draw lines and designate appropriate first digits as described in #1 above.

3. The lines drawn in #2 above represent electrical wiring and as such are an installation concern. The programmer is involved with specifying an input or output trunk (00, 40, etc.) and assigning a READ WRITE CHANNEL. These trunk designations and RWC assignments are accomplished when peripheral program instructions are written. Time sharing of main memory by read/write channels is on a demand basis. If one or more RWC does not require access to memory, the unrequired portions of the time sharing cycle are used by the central processor. Complete the times in this illustration.



4.



In this sample configuration, a programmer would specify (with PDT VARIANTS) the following RWC's and I/O trunks for a:

Card Read Instruction: RWC # 3 I/O TRUNK 41

Card Punch Instruction: RWC # 1 I/O TRUNK 01

Printer Instruction: RWC # 2 I/O TRUNK 02

The programmer can change RWC assignment to other devices whenever desired by simply using another VARIANT control character in the peripheral instruction.

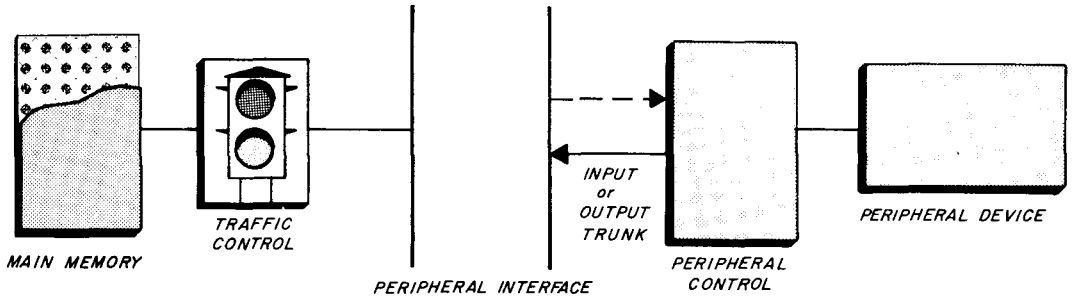
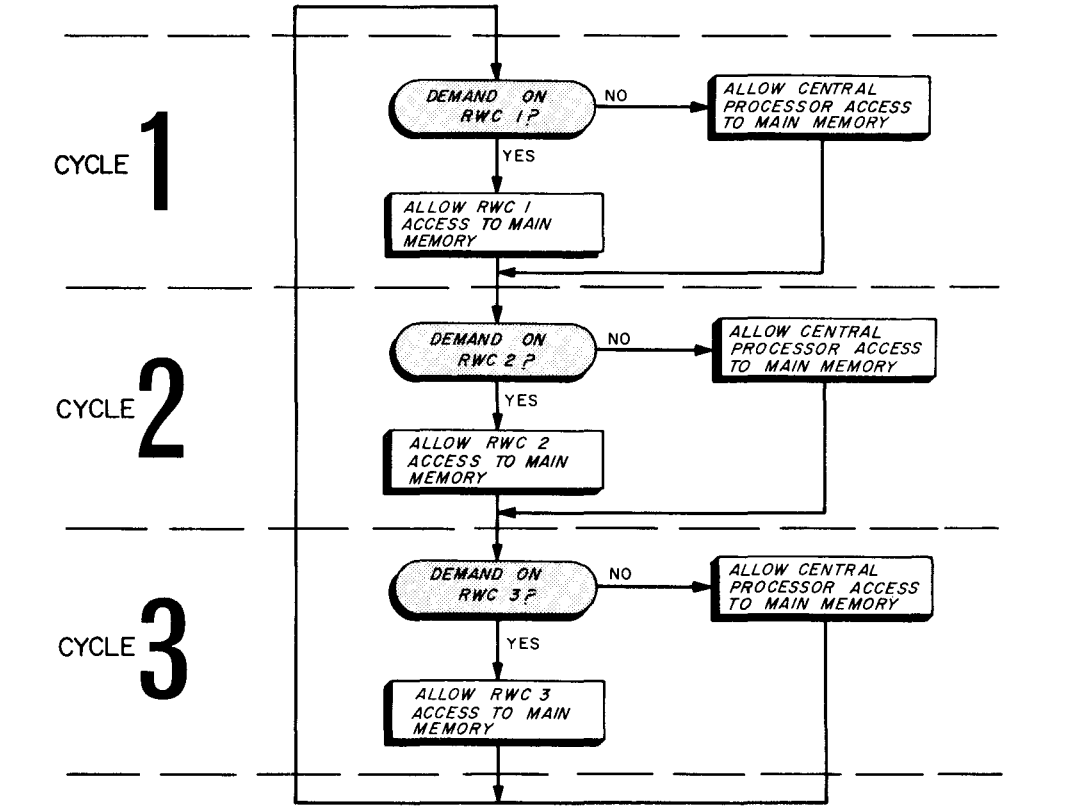


Figure 5. Basic Input/Output Data Path



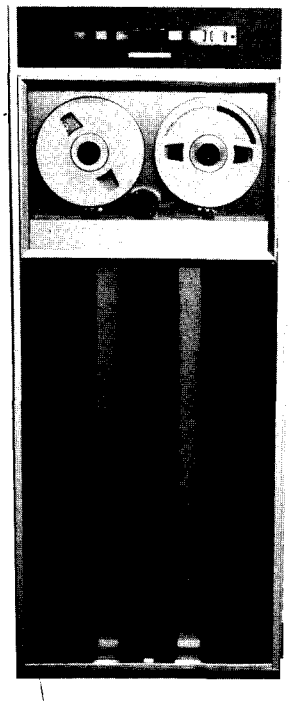
Each cycle is two microseconds in duration; therefore, a device connected to the main memory via an RWC is guaranteed access to the memory once every six microseconds.

Figure 6. Symbolic Representation of Input/Output Traffic Control

AUXILIARY READ/WRITE CHANNEL

An auxiliary read/write channel (RWC 1') is available as an optional feature. In systems equipped with this option, up to four peripheral data transfer operations can be performed simultaneously with computing. It is called an auxiliary channel because of the manner in which it is granted access to the main memory by the traffic control. RWC 1' and RWC 1 are connected to an alternator. Every six microseconds the alternator switches to allow one of these two channels access to the main memory. By alternating between the two channels, each is allowed access to the memory once every 12 microseconds. Note that RWC 2 and RWC 3 are still guaranteed access to the memory once every six microseconds.

MAGNETIC TAPE



Two complete series of magnetic tape equipment are offered for use with the Honeywell 200: the 204B series units process one-half inch tape, while the 204A series units process three-quarter inch tape. Both 203B controls for one-half inch tape units and 203A controls for three-quarter inch tape units can be included in the same system. The characteristics of the two series of tape equipment are summarized below.

MAGNETIC TAPE SPECIFICATIONS

Tape Unit	3/4 Inch 1.	3/4 Inch 2.	3/4 Inch 3.	1/2 Inch 1.	1/2 Inch 2.	1/2 Inch 3.	1/2 Inch 4.
READ-WRITE SPEED Inches per Second	60	120	120	36	80	120	150
RECORDING DENSITY Characters per Inch	533	533	740	200 556 800	200 556	200 556	200 556
TRANSFER RATE Character per Second	32,000	64,000	88,800	7,200 20,000 28,800	16,000 44,400	24,000 66,700	30,000 83,300
REWIND SPEED Inches per Second	180	360	360	108	240	360	360
INTER-RECORD GAP	.67"	.67"	.67"	.45"	.6"	.7"	.75"
DATA FORMAT	Honeywell 48-bit word			variable			
CHECKING FEATURES	frame and channel parity checks and Orthotronic Control			frame and channel parity checks for read; and read after write			

The Honeywell 200 uses two basic peripheral instructions for all input-output operations on all devices. Using these instructions, the programmer may instruct the tape unit to read forward, write, backspace and rewind. In addition, tape units may be read backward, a feature not available in most other small computer systems. The utilization of 3/4-inch and 1/2-inch magnetic tape makes the Honeywell 200 compatible with a wide range of computers.

The ability to perform tape operations simultaneously is enhanced by the fact that the central processor is involved in a tape read or write operation during only two microseconds per character transferred. The proportion of available central processor time during a data transfer interval shared with a tape read or write operation ranges from 82.2% to 98.6%, depending upon the data transfer rate of the tape unit being used. A typical tape processing interval is shown in the illustration below.

PROCESSING INTERVAL SHARED WITH TAPE OPERATION

0 ms	7.5 ms	132.5 ms
Begin crossing gap	Data transfer begins	Data transfer ends

DEVICE: Model 204B-3 magnetic tape unit.

OPERATION: Read or write a 2000-character record at a density of 200 characters per inch.

CENTRAL PROCESSOR TIME REQUIRED: 4 milliseconds (3.0% of entire processing interval).

CHARACTERISTICS	MODEL 204B-1, 2 TAPE UNITS	MODEL 204B-3, 4 TAPE UNTIS	MODEL 204B-5 TAPE UNITS	MODEL 204B-6 TAPE UNITS
CONTROL	MODEL 203B-1 TAPE CONTROL		MODEL 203B-2 TAPE CONTROL	
TAPE	Reels of approx. 2400 fit. of 1/2-in. Mylar[1]-base, oxide-coated tape.			
DATA FORMAT	Variable-lenght records separated by short or 3/4-inch gap. Records consisting of 6-bit characters spaced at 556 or 200 per inch can be read. Normally writes at 556 char/in., but can write at 200 char/in.			
PROGRAMMED OPERATIONS	Read forward, write forward, backspace one record, rewind, rewind and release, and erase, optional read backward and capability to translate between card images in IBM even-parity tape code and H-200 machine code.			
TRANSPORT	Pneumatic capstans and tape brakes.			
CROSS GAP TIME Short gap 3/4 inch gap	0.45 in. - 12.5 ms 20.8 ms	0.60 in. - 7.5 ms 9.4 ms	0.70 in. - 5.8 ms 6.3 ms	n/a 5.0 ms
READ/WRITE SPEED	36"/sec.	80"/sec.	120"/sec.	150"/sec.
DATA TRANSFER RATE (NOMINAL) 556 char/in. 200 char/in.	 20,000 char/sec. 7,200 char/sec.	 44,400 char/sec. 16,000 char/sec.	 66,700 char/sec. 24,000 char/sec.	 83,300 char/sec. 30,000 char/sec.
REWIND SPEED	108"/sec.	240"/sec.	360"/sec.	360"/sec.
SIMULTANEITY	Simultaneously compute and perform three tape operations: read or backspace-write-rewind-compute. Reading or writing engages central processor for only 2 microseconds per character transferred. Central processor is available for other operations during 83.3 to 98.6% of transfer interval shared with tape unit, depending upon data transfer rate.			
INPUT/OUTPUT AREA	Any main memory area.			
DATA PROTECTION	Write/protect ring and manual protect switch prevent destruction by unintentional write. While writing, TCU generates even or odd frame parity and even channel parity. Checks: Writing -- Immediate read back and check of information written. Reading -- Frame and channel parity checks. Failure of any check automatically sets a program-accessible indicator.			
TRUNKS	A tape control requires one input trunk and one output trunk.			
MAX. NO. OF UNITS PER SYSTEM	8 tape units per tape control; 8 tape controls per system.			

[1]Registered trademark of E. I. du Pont de Nemours and Company (Inc.)

Figure 7. Half-Inch Magnetic Tape Unit Characteristics

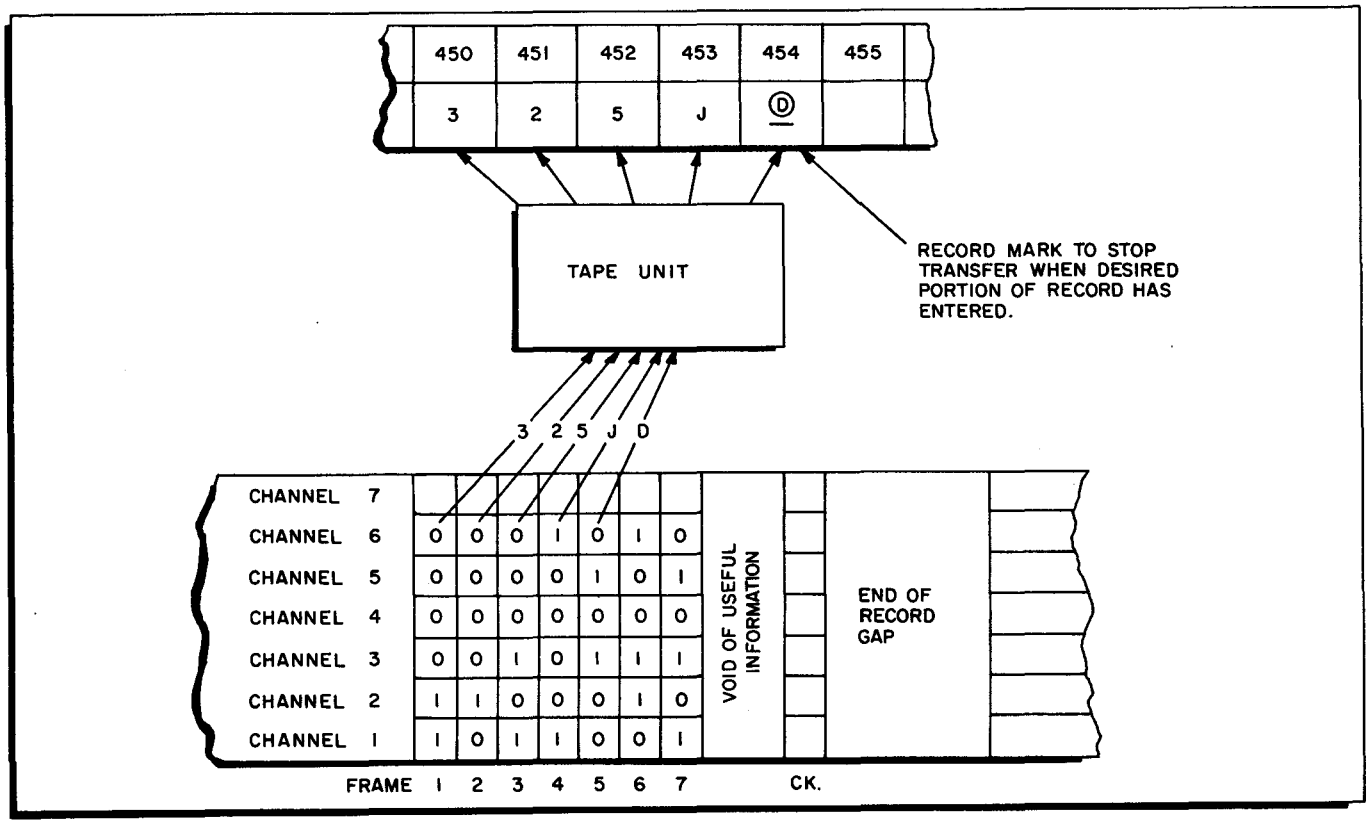


Figure 8. Data Transfer to Half-Inch Tape Segment

Answer and check these questions by referring to the chart and illustration on page 30.

1. An H-200-1401 "difference" is shown in the chart concerning "Cross Gap Time." H-200 magnetic tape can conserve space by using a short end of record gap of .45 in., .60 in., or .70 in. depending on the type of tape unit. A switch on the tape control unit is engaged if compatability with the .75 inch interrecord gap of your previous system is desired.

2. A VARIANT character, selected by the programmer and written as part of a peripheral instruction, specifies whether frame parity (across the tape width) is to be odd or even. As shown in the illustration, the desired parity bits are to be appended by the tape unit in CHANNEL # 7 .

3. Totaling the bits in each channel with the CHECK frame bits at the end of the record produces longitudinal channel parity. As listed in the chart, channel parity is stated as being ODD-PARITY

4. One tape frame will contain a six bit character and the parity bit from the tape unit. It should be apparent from the number of channels shown, that RECORD MARK (PUNCTUATION) is not transferred from memory to tape.

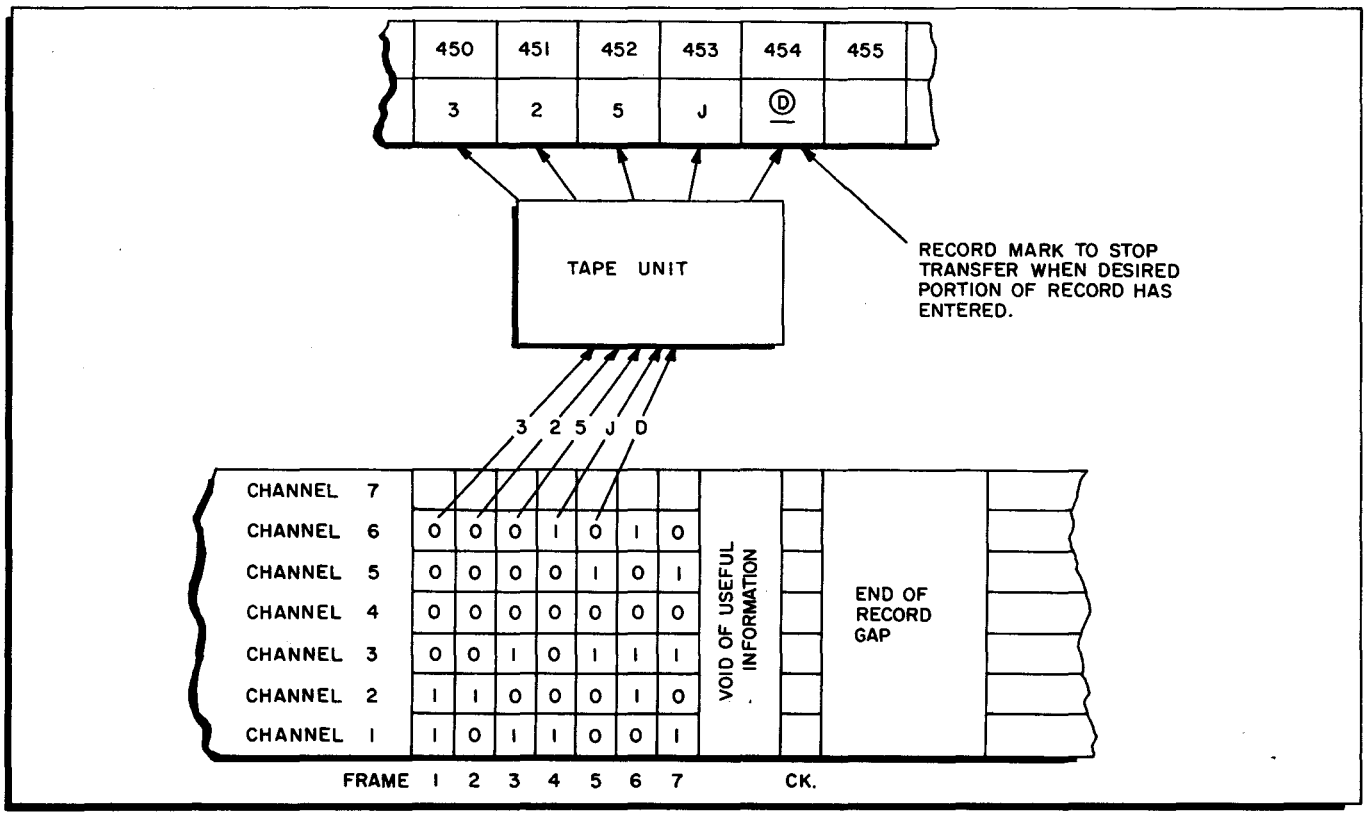
5. The only manner in which punctuation could be considered as being transferred to tape is that a RECORD mark in memory signals the tape unit to produce; a small void, then a check frame, and then the END of RECORD GAP.

Your previous system employed a GROUP mark on tape to facilitate transfer of only part of a tape record to memory. Similarly, an H-200 programmer may place a record mark in a predetermined memory location. This record mark stops transfer from tape to memory when the desired portion of a record has been read in. Check the answer to the following question by continuing to page 32.

6. Assume that the characters shown on tape in Figure 8. are to be read into memory starting at location address #450. If a record mark is placed in #454, what characters will be transferred from the tape?

450	451	452	453	454	455	456	457
3	2	5	J	Ⓓ			

(Answer to question #6 on the preceding page.)

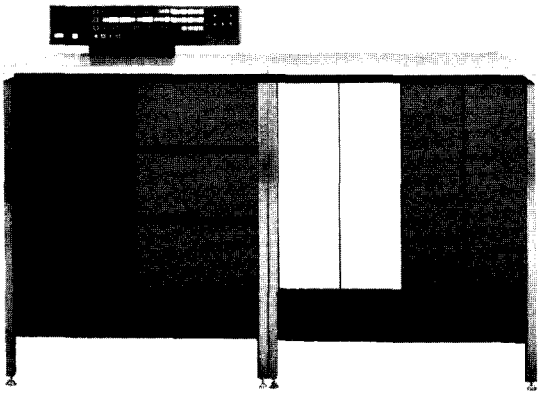


NOTE: When transferring from memory to tape, the character in the memory location with a record mark is NOT written on the tape.

When transferring from tape to memory, a character WILL be sent into the memory location containing record mark punctuation. This is shown in the illustration above.

LESSON III
H-200 CENTRAL PROCESSOR

The H-200 Central Processor



The Model 201 Central Processor is the computing and control center of the H-200 system. It houses the circuitry for arithmetic and logical operations, the high-speed magnetic core memory, the operator's control panel, and several special-purpose control elements such as read/write controls, etc. Functionally, the central processor is divided into three units: arithmetic, control, and storage. The arithmetic unit performs such operations as addition, subtraction, comparison, etc. The control unit directs the operation of the entire system: it controls the flow of information within the central processor; it controls the flow of information between the central processor and all input/output devices; it monitors the time sharing of the system to insure maximum operating efficiency; it selects, interprets, and controls the execution of all instructions; and it governs address selection within the high-speed memory. The storage unit provides magnetic core storage for the instructions and operands which the central processor uses in processing a particular program segment. It also provides storage for the new data which results from the operations performed by the central processor.

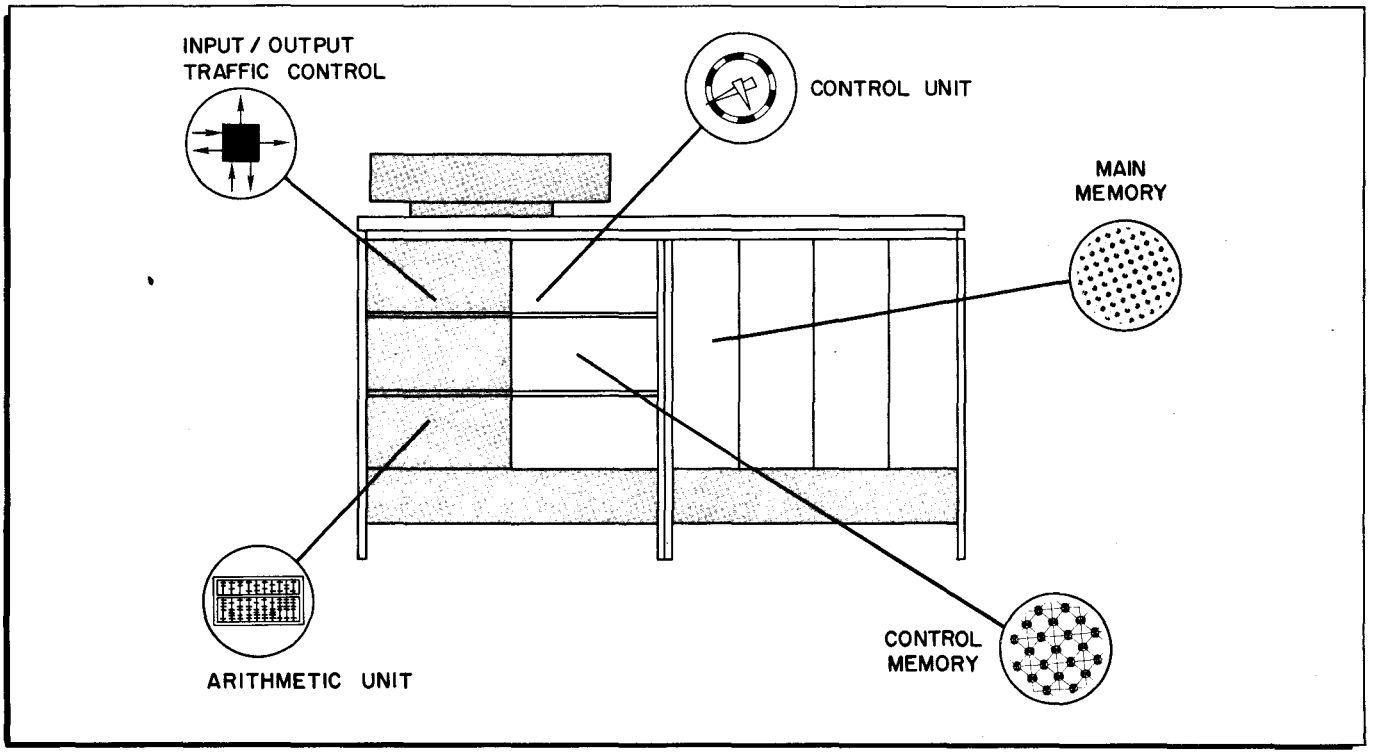


Figure 9. Logical Division of the Central Processor

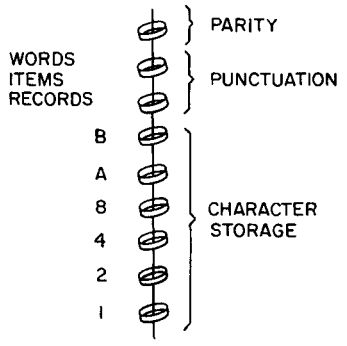
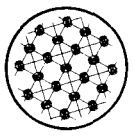
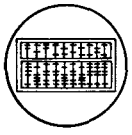
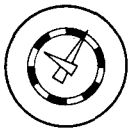
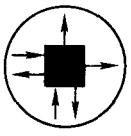
 A MAIN MEMORY LOCATION	BASIC MEMORY — 2,048 character locations. ADDITIONAL MEMORY — One 2,048-character module and additional 4,096-character modules. PROCESSING UNIT — Six-bit character. Variable-length groups of consecutive characters form instruction and data fields. INSTRUCTION FORMAT — Variable. Typical configuration: operation code, two addresses, and variant character. ADDRESSING MODES — Direct, indirect, indexed. INDEX REGISTERS — Six, each capable of storing three six-bit characters. MEMORY CYCLE — Two microseconds to read and restore one character.
 CONTROL MEMORY	MEMORY CAPACITY — 16 control registers, each capable of storing the address of a character position in the main memory. CONTROL REGISTERS — Basic configuration: two operand-address registers, two instruction address registers, and up to eight read/write channel counters. ACCESS TIME — 0.25 microseconds. MEMORY CYCLE — 0.5 microseconds.
 ARITHMETIC UNIT	OPERATIONS — Decimal arithmetic, binary arithmetic, logical operations. TYPICAL OPERATING SPEEDS — 5-digit decimal add (A + B⟶B) 44 microseconds. 5-digit compare (A:B) 34 microseconds.
 CONTROL UNIT	PARITY CHECKING — One parity bit with each character stored in memory. PROGRAM CONTROL — Sequential selection, interpretation and execution of all stored program instructions. CONTROL PANEL — Control and display functions.
 INPUT/OUTPUT TRAFFIC CONTROL	READ/WRITE CHANNELS — Three channels standard; auxiliary channel optional. I/O instructions designate channel connections. INPUT/OUTPUT TRUNKS — Basic configuration of eight input or output trunks; expandable by eight input or output trunks. PERIPHERAL SIMULTANEITY — Up to four peripheral transfer operations simultaneous with computing.

Figure 10. Summary of Central Processor Characteristics

1. Figure 10. states that basic H-200 memory contains 2,048 characters (memory locations). This number of memory locations can be expanded by a first module of 2048 characters then additional modules of 4096 characters.

TURN THE PAGE TO CHECK YOUR ANSWERS.

5. Besides the six cores required to store a CHARACTER, three additional cores are incorporated in each H-200 memory location. You are already familiar with the single core used for PARITY checking. In H-200 terminology, the other two cores are referred to as PUCTUATION cores and are used for the separation of RECORD, ITEM, WORDS.

9. The H-200 assures accuracy of storage by checking for ODD parity each time a character is read from memory. An error would be indicated if "bad" parity (an EVEN total of character "1" bits plus parity bit) should even occur.

Does the memory location below contain good or bad parity?

PARITY	IM	WM	CHARACTER
0	0	1	1 0 0 1 1 0

13. It is often convenient to transfer words pertaining to the same subject into adjacent memory locations. They may then be treated as an ITEM. An ITEM is defined as one or more related WORD stored in ADJACENT memory LOCATION. It is represented in illustrations by UNDERLINING the high order or low order character as desired.

17. This mark, O , is a combination of word and item mark symbols and is known as a RECORD mark. This punctuation is formed by using both punctuation cores in the first memory location following the RIGHTMOST character to be transferred to a peripheral device. Character by character transfer proceeds from LEFT to RIGHT until the RECORD mark is sensed.

1.

BASIC - 2048 MEMORY LOCATIONS
FIRST MODULE - 2048 MEMORY LOCATIONS
ADDITIONAL MODULES-4096 MEMORY LOCATIONS

5.

CHARACTER
PARITY
PUNCTUATION
WORDS, ITEMS, RECORDS

9.

EVEN

THE MEMORY LOCATION CONTAINED "GOOD" PARITY BECAUSE THE TOTAL OF CHARACTER "1" BITS PLUS THE PARITY BIT WAS ODD. PUNCTUATION BITS ARE NOT TOTALED IN A PARITY CHECK.

13.

ITEM
WORDS
ADJACENT
LOCATIONS
UNDERLINING

17.

O RECORD
LEFT (HIGH ORDER) - RIGHT (LOW ORDER)
RECORD

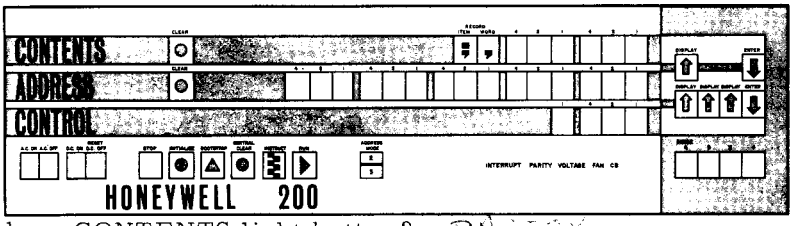
2. Core memory units are composed of planes of cores stacked in sufficient number to accommodate the 6 bit character format plus 2 word separation bits and 1 parity bit. The basic H-200 has 9 planes of 32 x 64 cores. This configuration provides $\frac{2048}{(32 \times 64)}$ memory locations 9 cores in depth.

---

6. The first six cores of a memory location are used for storage of any alphanumeric C-PRECTER. The next two cores are available as PUNCTUATION bits to designate a word, item, or record. The ninth core represents the PARITY bit used to check accuracy of bit storage.

---

10. A programmer or operator can check the contents of a memory location by observing the CONTENTS lights buttons on the central processor control panel. An illuminated button represents a "1" bit.



Which bit is not shown by a CONTENTS light button? PARITY

---

14. A word mark is used with the H-200 in the same manner as in the 1401. It is placed in the high order (leftmost) memory location of an instruction or data word where it:

1. Indicates the beginning of an instruction.
2. Defines length of a data word.
3. Stops instruction execution.

(This frame does not require a written answer.)

---

18. A RECORD MARK O is placed in the memory location following the RIGHTMOST character to be transferred. Record transfer to a peripheral device terminates when a [illegible] is sensed.

The following frame asks you to properly draw the punctuation above and also to draw circles for WORD marks and underlines for ITEM marks.

2.

2048

6.

CHARACTER
PUNCTUATION
PARITY

10.

PARITY

14.

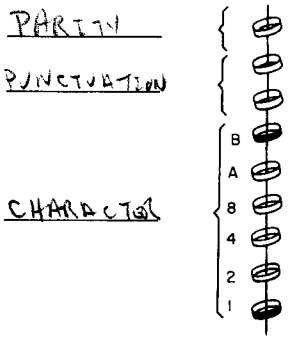
NO ANSWER REQUIRED

18.

⊗ - RECORD MARK following RIGHTMOST character
⊗ - WORD MARK
X - ITEM MARK

3. The basic H-200 core martrix provides 2048 memory locations. To be capable of storing one character plus two word separation bits and one parity bit, each location must be 9 cores deep.

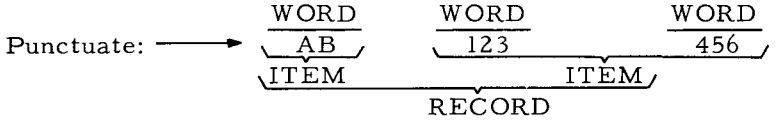
7. State the name or purpose of each core or group of cores in an H-200 memory location.



11. H-200 punctuation is also different in the rather obvious respects that the 1401 cannot designate items and a 1401 "record" requires a special character in an additional memory location. In the H-200, setting a word mark makes the word mark core a "1". To set an item mark, the ITEM MARK core is made a "1". Making both cores "1's" produces an H-200 RECORD MARK.

15. Item marks are most commonly set in the low order (rightmost) memory location of a data word. Consequently, items are usually retrieved or transferred character by character from HIGH order to LOW order until the ITEM marked character has been retrieved.

19.



ADDRESS	141	142	143	144	145	146	147	148	149	150
CONTENTS	Z	Ⓐ	B	Ⓛ	2	3	④	5	6	Ⓧ

What memory location would be addressed to:

1. Retrieve word 456? 149
2. Transfer item 123 456? 144
3. Transfer the record to peripheral device? 142

3.

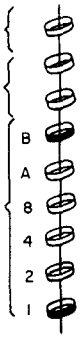
2048
9

7.

PARITY (ACCURACY CHECKING)

PUNCTUATION (SEPARATION OF WORDS, ITEMS, RECORDS)

CHARACTER (STORAGE)



11.

ITEM MARK
RECORD MARK

15.

HIGH order to LOW order
ITEM

NOTE: THE DIRECTION OF RETRIEVAL WOULD OF COURSE BE REVERSED IF THE ITEM MARK WERE IN THE HIGH ORDER POSITION.

19.

141	142	143	144	145	146	147	148	149	150
Z	Ⓐ	B	①	2	3	④	5	6	ⓧ

1. Address 149 to retrieve word 456.
2. Address 144 to transfer item 123 456.
3. Address 142 to transfer the entire record.

4. H-200 magnetic core memory provides high speed-one millionth of a second-random access to a memory location. Your previous system gained access to a memory location five times slower than the H-200. Additionally, a 1401 memory location only stores 8 binary digits because it contains 8 cores. The H-200 can store 9 BINARY DIGITS because it has 9 CORES per memory location.

---

8. A 1401-H200 "difference" should be noted at this point concerning parity checking and punctuation cores. The H-200 does NOT include punctuation bits in its parity check.

"Good" parity is shown if the total of character "1" bits and the parity bit equals an ODD number. When a character is written into memory, the parity core is magnetized as a "1" or "0" to produce an ODD total with the character "1" bits.

---

12. With your previous sytem, a word mark was shown in illustrations by underlining the proper character. H-200 illustrations use a circle around a character to represent a word mark. An underlined H-200 character represents the punctuation unique to the H-200 and therefore signifies an ITEM MARK.

---

16. A word or item mark core is used when the character is at the limit of a word or item. As shown below, _____ mark cores are used in addresses ____ and ____. The _____ mark core is used in address _____.

94	95	96	97	98	99	100
B	C	D	E	F	G	H

---

20. The preceding frames can be summarized by completing the blanks below and by drawing punctuation symbols for the X's.

FORMAT	SYMBOL	LOCATION	RETRIEVAL ADDRESS
WORD	(X)	HIGH ORDER	LOW ORDER
ITEM	X	HIGH ORDER or LOW ORDER	LOW ORDER or HIGH ORDER
RECORD	(X)	LOW ORDER FOLLOWING LAST CHARACTER TRANSFERRED	HIGH ORDER _____ _____

4.

9 BINARY DIGITS
9 CORES

(Return to page 37, frame 5.)

8.

ODD

(Return to page 37, frame 9.)

12.

ITEM MARK

(Return to page 37, frame 13.)

16.

WORD
94 97
ITEM
100

(Return to page 37, frame 13.)

20

FORMAT	SYMBOL	LOCATION	RETRIEVAL ADDRESS
WORD	⊗	HIGH ORDER	LOW ORDER
ITEM	X	HIGH ORDER or LOW ORDER	LOW ORDER or HIGH ORDER
RECORD	⊗	FOLLOWING LAST CHARACTER TRANSFERRED	HIGH ORDER

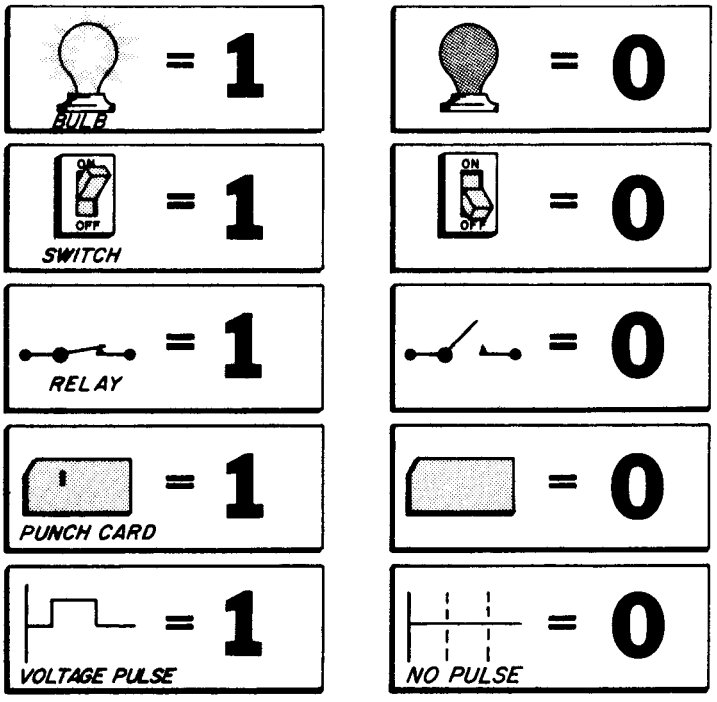
(Continue to page 45.)

LESSON IV

PART I. NUMBERING SYSTEMS

AND

PART II. HONEYWELL ALPHANUMERIC CODE



BINARY, OCTAL, AND DECIMAL EQUIVALENTS

BIN.	OCT.	DEC.	BIN.	OCT.	DEC.
0	0	0	10000	20	16
1	1	1	10001	21	17
10	2	2	10010	22	18
11	3	3	10011	23	19
100	4	4	10100	24	20
101	5	5	10101	25	21
110	6	6	10110	26	22
111	7	7	10111	27	23
1000	10	8	11000	30	24
1001	11	9	11001	31	25
1010	12	10	11010	32	26
1011	13	11	11011	33	27
1100	14	12	11100	34	28
1101	15	13	11101	35	29
1110	16	14	11110	36	30
1111	17	15	11111	37	31

POWERS OF 2

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1 024
11	2 048
12	4 096
13	8 192
14	16 384
15	32 768

Figure 11. Binary Representation

1. Just as different languages can express the same meaning, different numbering systems have the capability of expressing the same quantities. To aid in understanding and using numbering systems adaptable to electronic computers, it is beneficial to first review the familiar decimal system.

---

13. If we represent the off state of the light bulb by the binary zero (0), it is readily apparent the on state can be represented by the binary 1. A string of lights in a systematic on and off configuration could represent any BINARY number.

---

25. The most commonly used method of decimal to binary conversion is the remainder method. The decimal number is divided by two and that quotient and all succeeding quotients are in turn divided by two. The remainders of each division must be 1 or 0 and these make up the bits of the binary number with the final remainder the most significant digit. Using the decimal 13, the remainder method is illustrated below.

$(2/\overline{13}\ \overset{6}{}\ R = 1)\ (2/\overline{6}\ \overset{3}{}\ R = 0)\ (2/\overline{3}\ \overset{1}{}\ R = 1)\ (2/\overline{1}\ \overset{0}{}\ R = 1) = 1101$ binary

---

37. Complement the subtrahend of the binary subtraction problems listed below.

101111 — 100101 ans. _____

1110111 — 0100010 ans. _____

---

49. Using the powers of the base 8, convert the following octal numbers to their decimal equivalent.

$6540_8 =$ $235_8 =$

$11_8 =$ $77_8 =$

1.

NO ANSWER REQUIRED

13.

1

BINARY

25.

NO ANSWER REQUIRED

37.

011010 1011101

49.

$6 \times 8^3 = 3072$
$5 \times 8^2 = 320$
$4 \times 8^1 = 32$
$0 \times 8^0 = 0$
3424_{10}

$1 \times 8^1 = 8$
$1 \times 8^0 = 1$
9_{10}

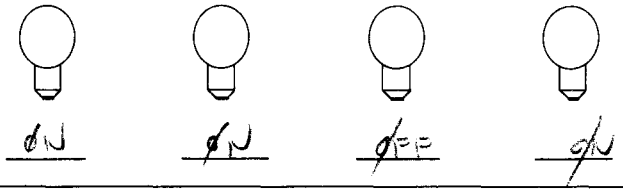
$2 \times 8^2 = 128$
$3 \times 8^1 = 24$
$5 \times 8^0 = 5$
157_{10}

$7 \times 8^1 = 56$
$7 \times 8^0 = 7$
63_{10}

2. Peculiar to each numbering system is the base (or radix) and the number of digits used in that system. The base or radix of the system indicates the number of digits used. The decimal system, with a base of ten, uses 10 different digits.

---

14. A binary number is represented by a series of 1's and 0's called "bits" (a contraction of binary digits). Using light bulbs to represent the binary number 1101, which bulbs would be on and which ones would be off?



---

26. In the previous example, decimal 13 was converted to binary 1101. To prove this answer, convert binary 1101 to decimal by using powers of two.

8 + 4 + 0 + 1 = 13

---

38. After complementing the subtrahend, the next step is adding the complemented number to the minuend. Complement and add in the following problems.

101010 = 101010 — 010101 ⟶ _____

111011 = 111011 — 100011 ⟶ _____

---

50. For decimal to octal conversion, the easiest approach is the remainder method explained in decimal to binary conversion. A division of 8 is used instead of 2. Remember, the last remainder is the most significant digit of the total number.

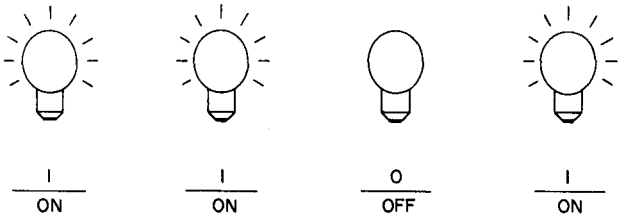
Convert 77_{10} to its octal equivalent.

ANS. _____

2.

10

14.



26.

$$(\underline{1} \times 2^3) + (\underline{1} \times 2^2) = (\underline{0} \times 2^1) + (\underline{1} \times 2^0) = \underline{13}$$

$$(1 \times 8) \quad (1 \times 4) \quad (0 \times 2) \quad (1 \times 1)$$

$$8 \qquad 4 \qquad 0 \qquad 1$$

38.

c c c	ccc
101010	111011
101010	011100
1010100	1010111

---

50. 115_8

$$\begin{array}{r} 0 \\ 8/\overline{1} \quad R=1 \\ 8/\overline{9} \quad R=1 \\ 8/\overline{77} \quad R=5 \end{array}$$

3. The numbers of the decimal system are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. The highest number represented by a single digit in a system will be one less than the BASE OR RADIX

---

15. The computer, of course, does not employ light bulbs, but does operate on the same on-off principle. Transistors are either conducting or non-conducting; magnetic tape, discs, and drums are magnetized or not magnetized, and cores are magnetized in one of two polarities. Punch cards are either PUNCHED or NOT PUNCHED.

---

27. Remember, the last remainder is the most significant digit in the binary number when converting decimal to binary by the remainder method. To ease in applying this rule, division can be solved thus:

0		
2 / 1	R = 1	
2 / 3	R = 1	
2 / 7	R = 1	
2 / 15	R = 1	
2 / 30	R = 0	What is the binary number? 11110

---

39. The previous problems were:

101010	101010	111011	111011
-010101	+101010	-100011	+011100
	1010100		1010111

It is apparent these answers are not correct. In both cases, the answer contains more digits than the minuend. One additional step is required.

---

51. Convert the following decimal numbers to their octal equivalent.

8_{10} = 786_{10} =

88_{10} = 888_{10} =

3.

BASE (RADIX)

15.

PUNCHED

NOT PUNCHED

27.

11110

39.

NO ANSWER REQUIRED

51.

$8_{10} = 10_8$ $\qquad$ $786_{10} = 1422_8$

$88_{10} = 130_8$ $\qquad$ $888_{10} = 1570_8$

4. A peculiarity of a positional number system is the manner in which we record the digits. The number 10 is quite different in value than 01 although the same digits are used. The difference in value is determined by the POSITION of the digits in the whole number.

---

16. The bistable or two state devices listed in the previous statement are adaptable to the BINARY numbering system. Since the position of the digits 0 and 1 determine their value, this system, like the decimal system is also a POSITIONAL numbering system.

---

28. Convert the following decimal numbers to binary numbers using the remainder method.

11 = _____

51 = _____

358 = _____

---

40. The last step is called "end around carry."

101010	101010	111011	111011
-010101	101010	-100011	011100
	① 010100		① 010111
	↘1		↘1
	10101		11000

Rule for end around carry: THE 1 IN THE HIGH ORDER POSITION (MOST SIGNIFICANT DIGIT) IS ADDED TO THE 2^0 POSITION (LEAST SIGNIFICANT DIGIT).

CONVERT THE PROBLEMS TO DECIMALS AND CHECK THESE ANSWERS.

---

52. Explain briefly in your own words why 2 is not a valid binary number and 9 is not a valid octal number. _____

_____

_____

_____

4.

POSITION

16.

BINARY

POSITIONAL

28.

11 = 1011

```
   0
2/1  R = 1
2/2  R = 0
2/5  R = 1
2/11 R = 1
```

51 = 110011

```
   0
2/1  R = 1
2/3  R = 1
2/6  R = 0
2/12 R = 0
2/25 R = 1
2/51 R = 1
```

358 = 101100110

```
   0
2/1   R = 1
2/2   R = 0
2/5   R = 1
2/11  R = 1
2/22  R = 0
2/44  R = 0
2/89  R = 1
2/179 R = 1
2/358 R = 0
```

40.

```
101010 =  42
-010101 = -21
 10101 =  21
```

```
111011 =  59
-100011 = -35
 11000 =  24
```

52. The base or radix of a numbering system indicates the number of digits used in that system, also the highest number represented by a single digit in any system is one less than the base. Binary has a base of 2 and the highest single digit is 1. Two does not exist in this system. Octal has a base of 8 and the highest single digit is 7. Nine does not exist in the Octal system.

5. Characteristic of a positional numbering system is that the value of each position in a multidigit number represents a specific power of the base. In the decimal system, the positions to the left or right of the decimal point increase or decrease by powers of 10.

---

17. Since binary is a numbering system with a base of two, positional value of digits increase or decrease by powers of TWO.

---

29. Elementary to converting binary to decimal would be the construction of a simple graph of the powers of two.

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1
1	0	1	1	0	1	1	0	0	1

With the binary number 1 0 1 1 0 1 1 0 0 1 entered in the chart, we need only to add the powers of two to arrive at the decimal equivalent. What is the decimal equivalent? 1. What is the largest decimal number that could be represented in this chart? 1023.

---

41. Using the complementation and end around carry method solve this problem.

SUBTRACT B FROM A.

A. 101011

B. 011101

STEP #1 COMPLEMENT B

A. 101011

B. _____

STEP #2 ADD A, AND B COMPLEMENT

A. 101011

B. _____

ANS. _____ (STEP #3 NEXT FRAME)

---

53. Define base or radix and positional numbering systems. _____

5.

10

17.

TWO

29.

729

1023

THE LARGEST QUANTITY REPRESENTED BY A SERIES OF 1 BITS WILL ALWAYS BE ONE LESS THAN THE NEXT HIGHER POWER OF 2.

41.

STEP #1

```
  101011
  100010
```

STEP #2

```
  101011
  100010
 1001101
```

---

53. BASE OR RADIX INDICATES THE NUMBER OF DIGITS IN THE SYSTEM. THE POWER OF THE BASE IS DENOTED BY THE POSITION OF THE DIGIT.

6. The powers of ten are: $10^0=1$, $10^1=10$, $10^2=100$, $10^3=1000$, etc. We commonly call these positions of the decimal system units, tens, hundreds, thousands, etc. Expressed as a power of ten, 10,000 would be 10^4.

---

18. Powers of ten are 10^0, 10^1, 10^2, 10^3, etc., and represent values 1, 10, 100, 1000. Using the powers of two, what are the decimal values of 2^0, 2^1, 2^2, 2^3, 2^4?

$2^0=1$ $2^1=2$ $2^2=4$ $2^3=8$ $2^4=16$

---

30. The basic operation performed in the arithmetic unit of the central processor is CALCULATION. Consequently, any numbering system compatible with electronic data processing must have the quality to permit calculation.

---

42. Continuing with the previous problem 101011 minus 011101.

STEP #1 COMPLEMENTING		STEP #2 ADD		STEP #3 PERFORM END AROUND CARRY AND ADD.
A. 101011	A. 101011	A. 101011		(1) 001101
B. -011101	B. 100010	B. 100010		1
		1001101	Answer	

Convert the original problem to decimal and check your answer.

---

54. It was mentioned that octal provides a shorthand method for dealing with binary numbers. To illustrate, first represent each of the 8 octal numbers as three bit binary numbers. If necessary, add zeroes to the left to make three bit binary numbers.

0. = ____ 4. = ____

1. = ____ 5. = ____

2. = ____ 6. = ____

3. = ____ 7. = ____

6.

10^4

18.

$2^0 = 1$ $2^1 = 2$ $2^2 = 4$ $2^3 = 8$ $2^4 = 16$

30.

CALCULATION

42.

A. 101011 = 43

B. 011101 = 29

14 = 1110

54.

0. = 000	4. = 100
1. = 001	5. = 101
2. = 010	6. = 110
3. = 011	7. = 111

7. For clarity and comparison, a simple graph illustrating positional value in powers of the base ten and the literal description may be useful.

thousands	hundreds	tens	units
10^3 or 1000	10^2 or 100	10^1 or 10	10^0 or 1
3	0	0	7

Record the decimal numbers 5347 and 3000 in the above graph, each digit in its proper value position.

---

19. Illustrated in a simple graph as used with the decimal system, the powers of two and positional values are easily determined.

Decimal Value	Sixteen	Eight	Four	Two	Units
Power	2^4	2^3	2^2	2^1	2^0
	1	0	0	0	1

Record the binary numbers 1101 and 10001 in the graph and determine the decimal equivalent. 1101 = 13 10001 = 17

---

31. Binary arithmetic follows the same general rules as decimal arithmetic except that base two tables are used instead of base ten tables. The following are the four basic rules of binary addition; 0+0=0, 0+1=1, 1+0=1, 1+1=0 plus a carry of 1.

EXAMPLE: Add 1011 + 1010

c c ("c" indicates a carry)

1011 =
1010 =
10101 =

Convert the binary numbers to decimal, add and check the result.

---

43. For practice and understanding, solve the following subtraction using complementation and end around carry.

1101011	10110
-1011110	-01001
ANS. _____	ANS. _____

---

55. Any binary number may be converted to octal by dividing it into groups of three bits starting at the right-most bit and then converting each group into its octal equivalent

EXAMPLE: $100/111 = 47_8$

To prove; convert the binary number 100111 and the octal number 47 to their decimal equivalents

100111_2= _____ 47_8 = _____

7.

thousands	hundreds	tens	units
10^3 or 1000	10^2 or 100	10^1 or 10	10^0 or 1
5	3	4	7
3	0	0	0

19.

SIXTEEN	EIGHT	FOUR	TWO	UNITS
2^4	2^3	2^2	2^1	2^0
	1	1	0	1
1	0	0	0	1

1101 = 8 + 4 + 0 + 1 = 13
10001 = 16 + 0 + 0 + 0 + 1 = 17

31.

1011 = 11
1010 = 10
10101 = 21

43.

1101011
0100001
(1)0001100
1
1101

10110
10110
(1)01100
1
1101

55.

32 16 8 4 2 1
1 0 0 1 1 1 = 32
4
2
1
39_{10}

100111 = 39_{10}

64 8 1
4 7 = 32
7
39_{10}

$47_8 = 39_{10}$

8. As with any positional numbering system, each digit of a multidigit number can be expressed as that number times its power of the base. Example: 4,968 is $(4 \times 10^3) + (9 \times 10^2) + (6 \times 10^1) + (8 \times 10^0)$. The sum of these numbers is the original multidigit number. Write the decimal number 6521 using powers of the base.

(6×10^3) + (5×10^2) + (2×10^2) + (1×10^0).

---

20. Each digit of a multidigit binary number can be expressed as that number times its power of the base 2. Example: 1011 is $(1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$. The sum of the individual digits is the decimal equivalent. Write the binary number 1111 using powers of the base and determine the decimal value.

(1×2^3) + (1×2^2) + (1×2^1) + (1×2^1) = 15

---

32. As practice and to check accuracy, solve the following binary additions then convert to decimal and verify results.

1111	1101	1011
1111	1110	11
11110	11011	1110

---

44. The complementing and end around carry steps work just as effectively with any numbering system. Using the 9s complement, the decimal subtraction problems below illustrate this fact. Complete the end around carry and add.

7632		7632	9678		9678
-5246	COMPLEMENT →	4753	-8422	COMPLEMENT →	1577
		12385			11255

---

56. Convert the following binary numbers to octal numbers and the resultant octal numbers to their decimal equivalent.

	OCTAL	DECIMAL
101110 =		
1001101 =		
1111111111 =		

8.

$$(6 \times 10^3) + (5 \times 10^2) + (2 \times 10^1) + (1 \times 10^0)$$

20.

$$(1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) = 15$$

32.

1111 = 15	1101 = 13	1011 = 11
1111 = 15	1110 = 14	11 = 3
11110 = 30	11011 = 27	1110 = 14

44.

7632	9678
4753	1577
(1) 2385	(1) 1255
1	1
2386	1256

56.

$$101/110 = 56_8 = 46_{10}$$
$$1/001/101 = 115_8 = 77_{10}$$
$$1/111/111/111 = 1777_8 = 1023_{10}$$

9. One rule to be remembered which is applicable to <u>any</u> positional numbering system; any base (or radix) to the zero power equals one (1).

$$2^0 = 1,\ 8^0 = 1,\ 10^0 = 1,\ 16^0 = 1$$

---

21. Binary 0 is equal to decimal 0 and binary one by itself is equal to decimal 1. Since binary is a system using a base of two and only digits 0 and 1, any quantity over one (1) requires a multidigit binary number. Decimal 2 written in binary is 10 .
$(1 \times 2^1) + (0 \times 2^0)$

---

33. Whenever a column generates more than one carry, a "c" is inserted in the next column for each carry. Each "c" is treated as a 1 in its column.

	c c			
	ccccc			
	10111=23		10110	1011
Example	10011=19	Solve:	1010	111
	11010=26		111011	1011
	1000100=68		(0)1)1	1101

---

45. Often a binary number may contain so many bits it becomes unwieldy and extremely difficult to communicate other than in the computer. Another positional numbering system is used to permit communication of binary numbers without resorting to a series of 1s and 0s. This "shorthand" system is the octal numbering system using the eight digits 0, 1, 2, 3, 4, 5, 6, and 7.

---

57. Convert the following decimal numbers to binary using the remainder method and then convert the binary results to octal numbers using the shorthand 3s method.

	BINARY	OCTAL
511 =	_____	_____
426 =	_____	_____
112 =	_____	_____

9.

NO ANSWER REQUIRED

21.

10

33.

		c
	ccccc	cccc
	10110	1011
	1010	111
	111011	1011
	1011011	11101

45.

NO ANSWER REQUIRED

57.

$511 = 111111111 = 777_8$

$426 = 110101010 = 652_8$

$112 = 1110000 = 160_8$

10. Complexity of electronic circuitry necessary for utilizing the decimal system has resulted in a simpler two digit system for computer use. This system, having a base of two, must use the digits 0 and 1.

22. Binary numbers may be converted to decimal numbers quite easily by the positional notation method. Each position is assigned its value and the values are then added together. $1101 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 13$. 111011 is equal to:

$1 \times 2^0 =$ 1
$1 \times 2^1 =$ 2
$0 \times 2^2 =$ 0
$1 \times 2^3 =$ 8
$1 \times 2^4 =$ 16
$1 \times 2^5 =$ 59 Decimal equivalent

34. To facilitate computer subtraction, a method involving COMPLEMENTATION and END AROUND CARRY is used.

PROBLEM: Minuend 1101011 = _____ 1101011 Minuend
Subtrahend - 1011110 = _____ +0100001 (Complemented Subtrahend)
①0001100
1
0001101 = _____

Convert the problem to decimal, perform the subtraction and compare your answer to the complementation and end around carry answer.

46. Each position within an octal number represents a specific power of the radix 8. What are the specific values (decimal) of the following powers of 8?

$8^0 =$ 1 $8^2 =$ 64
$8^1 =$ 8 $8^3 =$ 512

58. The shorthand octal method of converting binary numbers is not by accident or coincidence. In the graph below you can readily see the interrelationship of each higher power of 8 to each third higher power of 2.

8^4			8^3			8^2			8^1			8^0
2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
4096			512			64			8			1

10.

0 and 1

22.

$1 \times 2^0 = 1$
$1 \times 2^1 = 2$
$0 \times 2^2 = 0$
$1 \times 2^3 = 8$
$1 \times 2^4 = 16$
$1 \times 2^5 = 32$
59 = Decimal equivalent

34.

$$\begin{array}{rl} 1101011 & = 107 \\ -\ 1011110 & = \ 94 \\ \hline & 13 = 1101 \end{array}$$

46.

$8^0 = 1$ $\qquad$ $8^2 = 64$

$8^1 = 8$ $\qquad$ $8^3 = 512$

58.

NO ANSWER REQUIRED

11. Just as the decimal system can express any quantity with ten digits, the binary system can express any quantity with the two digits 0 and 1.

---

23. Convert the following binary numbers to their decimal equivalents.

10101 = 21

110011 = 51

101100110 = 358

---

35. The first rule in solving subtraction by addition -- THE COMPLEMENT OF A DIGIT IS EQUAL TO ONE LESS THAN THE RADIX, MINUS THAT DIGIT. Following this rule, the decimal system uses the 9s complement and binary uses the 1s complement. Both 9 and 1 are one less than the base 10 and 2 respectively.

Example: The 9s complement of 6 is: 9 minus 6, or 3.

The 1s complement of 0 is: 1 minus 0, or 1.

What is the 9s complement of 632? (Complement each digit) 367.

What is the 1s complement of 1100? 0011.

---

47. Dealing with more than one numbering system can lead to confusion unless care is exercised. As examples, 236 could be either a decimal or octal number and 101 could be decimal, octal, or binary. If there is any room for doubt, a subscript must be appended to the number.

236_8 is an OCTAL number. 236_{10} is a DECIMAL number.

101_2 is an BINARY number.

---

59. As shown in your Easycoder notes, two arithmetic capabilities of the H-200 are:

(BA) BINARY FIXED

(BS) BINARY FLOATING

When these operations are explained, you will see that the preceding 58 frames have provided necessary background information about numbering systems.

11.

0 and 1

23.

10101 = 21

110011 = 51

101100110 = 358

35.

367

0011

47.

236_8 OCTAL

101_2 BINARY

236_{10} DECIMAL

59.

BINARY ADDITION
BINARY SUBTRACTION

A numbering systems background aids understanding of several areas besides arithmetic operations. Examples are: Deciphering control panel lights displaying binary address and memory location contents. Decoding octal portions of printed listings. Writing binary literals or constants on coding forms. Specifying six bit VARIANT characters with two digit octal.

12. This two value system using 0 and 1 and called the binary numbering system, lends itself to computer circuitry. A common example used in explaining this two value concept is the light bulb. The light bulb can only be in one of two states, ON or OFF.

---

24. As further practice in binary to decimal conversion, list the decimal equivalents of the following:

1010 = _____	0100 = _____
1001 = _____	0001 = _____
0101 = _____	0011 = _____
0010 = _____	0110 = _____
0111 = _____	1000 = _____

---

36. Complementing the subtrahend of a binary subtraction problem merely involves changing all ones to zeros and all zeros to ones. The ones complement of 0011101 is 1100010.

Complement the following:

10010 = _____

00100 = _____

11111 = _____ It is that simple.

---

48. In octal to decimal conversion, as with binary to decimal, each position is assigned its value of the power of the base and the values are added together. Thus, 356_8 is equal to:

3×8^2 =

5×8^1 =

6×8^0 =

TOTAL = 238

---

60. Quite often addresses are changed by the programmer from binary to decimal or from decimal to binary. These changes are most easily accomplished in the following sequences:

(BINARY TO DECIMAL) Convert Binary to Octal, then Octal to Decimal

111111_2 77_8, 77_8 63_{10}

For DECIMAL TO BINARY, convert DECIMAL to _____ then _____ to BINARY. Example, 63_{10} = _____, _____ = _____.

12.

ON or OFF

(Return to page 47, frame 13.)

24.

1010 = 10	0100 = 4
1001 = 9	0001 = 1
0101 = 5	0011 = 3
0010 = 2	0110 = 6
0111 = 7	1000 = 8

(Return to page 47, frame 25.)

36.

01101

11011

00000

(Return to page 47, frame 37.)

48.

$3 \times 8^2 = 192$

$5 \times 8^1 = 40$

$6 \times 8^0 = 6$

TOTAL = 238_{10}

(Return to page 47, frame 49.)

60.

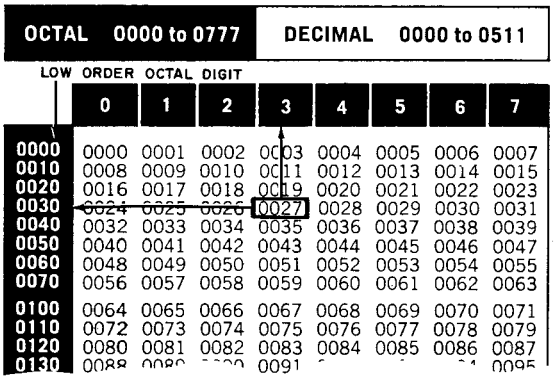
DECIMAL (to) OCTAL, (then) OCTAL (to) BINARY

$63_{10}=77_8$, $77_8=1111111_2$

Binary to octal and octal to binary can be accomplished without much difficulty. Decimal to octal and octal to decimal is simplified through use of the conversion tables on the following pages.

OCTAL - DECIMAL CONVERSION

Notice in the table at the right, that OCTAL numbers are shown as white digits on a black background. DECIMAL numbers compose the majority of the table as four digits and increase in seven columns from left to right.



OCTAL 0000 to 0777 DECIMAL 0000 to 0511

LOW ORDER OCTAL DIGIT

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091				0095

DECIMAL TO OCTAL CONVERSION:

Locate decimal number 27 in the table (0027).

Read to the left for the octal number (0030).

Read up from the decimal to determine the low order octal digit (3). Answer: $27_{10}=33_8$

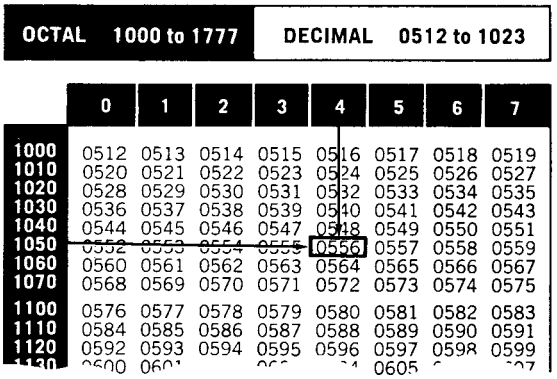
Leading zeros may be omitted.

$33_8=011011_2$

OCTAL TO DECIMAL CONVERSION:

Locate octal high and low order digits.

Example: octal 1054 (1050, 4) = 556_{10}



OCTAL 1000 to 1777 DECIMAL 0512 to 1023

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601				0605		

The conversion tables on the following few pages may be removed for future reference. As practice in their use, convert the following:

(A) BINARY 101 011
OCTAL ___ ___
DECIMAL _____

(B) DECIMAL 4 0 9 5 (On the fourth table.)
OCTAL _ _ _ _
BINARY _____

(C) OCTAL 6573
DECIMAL ____

(D) DECIMAL 2048
OCTAL _____
BINARY _____

Answers:

(D) OCTAL 4000
BINARY 100000000000

(C) DECIMAL 3451

(B) OCTAL 7 7 7 7
BINARY 111 111 111 111

(A) OCTAL 53
DECIMAL 43

OCTAL 0000 to 0777	DECIMAL 0000 to 0511

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

OCTAL 1000 to 1777	DECIMAL 0512 to 1023

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

Octal-Decimal Conversion Table

OCTAL 2000 to 2777	DECIMAL 1024 to 1535

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

OCTAL 3000 to 3777	DECIMAL 1536 to 2047

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

Octal-Decimal Conversion Table (cont)

OCTAL 4000 to 4777									DECIMAL 2048 to 2559								
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055	5000	2560	2561	2562	2563	2564	2565	2566	2567
4010	2056	2057	2058	2059	2060	2061	2062	2063	5010	2568	2569	2570	2571	2572	2573	2574	2575
4020	2064	2065	2066	2067	2068	2069	2070	2071	5020	2576	2577	2578	2579	2580	2581	2582	2583
4030	2072	2073	2074	2075	2076	2077	2078	2079	5030	2584	2585	2586	2587	2588	2589	2590	2591
4040	2080	2081	2082	2083	2084	2085	2086	2087	5040	2592	2593	2594	2595	2596	2597	2598	2599
4050	2088	2089	2090	2091	2092	2093	2094	2095	5050	2600	2601	2602	2603	2604	2605	2606	2607
4060	2096	2097	2098	2099	2100	2101	2102	2103	5060	2608	2609	2610	2611	2612	2613	2614	2615
4070	2104	2105	2106	2107	2108	2109	2110	2111	5070	2616	2617	2618	2619	2620	2621	2622	2623
4100	2112	2113	2114	2115	2116	2117	2118	2119	5100	2624	2625	2626	2627	2628	2629	2630	2631
4110	2120	2121	2122	2123	2124	2125	2126	2127	5110	2632	2633	2634	2635	2636	2637	2638	2639
4120	2128	2129	2130	2131	2132	2133	2134	2135	5120	2640	2641	2642	2643	2644	2645	2646	2647
4130	2136	2137	2138	2139	2140	2141	2142	2143	5130	2648	2649	2650	2651	2652	2653	2654	2655
4140	2144	2145	2146	2147	2148	2149	2150	2151	5140	2656	2657	2658	2659	2660	2661	2662	2663
4150	2152	2153	2154	2155	2156	2157	2158	2159	5150	2664	2665	2666	2667	2668	2669	2670	2671
4160	2160	2161	2162	2163	2164	2165	2166	2167	5160	2672	2673	2674	2675	2676	2677	2678	2679
4170	2168	2169	2170	2171	2172	2173	2174	2175	5170	2680	2681	2682	2683	2684	2685	2686	2687
4200	2176	2177	2178	2179	2180	2181	2182	2183	5200	2688	2689	2690	2691	2692	2693	2694	2695
4210	2184	2185	2186	2187	2188	2189	2190	2191	5210	2696	2697	2698	2699	2700	2701	2702	2703
4220	2192	2193	2194	2195	2196	2197	2198	2199	5220	2704	2705	2706	2707	2708	2709	2710	2711
4230	2200	2201	2202	2203	2204	2205	2206	2207	5230	2712	2713	2714	2715	2716	2717	2718	2719
4240	2208	2209	2210	2211	2212	2213	2214	2215	5240	2720	2721	2722	2723	2724	2725	2726	2727
4250	2216	2217	2218	2219	2220	2221	2222	2223	5250	2728	2729	2730	2731	2732	2733	2734	2735
4260	2224	2225	2226	2227	2228	2229	2230	2231	5260	2736	2737	2738	2739	2740	2741	2742	2743
4270	2232	2233	2234	2235	2236	2237	2238	2239	5270	2744	2745	2746	2747	2748	2749	2750	2751
4300	2240	2241	2242	2243	2244	2245	2246	2247	5300	2752	2753	2754	2755	2756	2757	2758	2759
4310	2248	2249	2250	2251	2252	2253	2254	2255	5310	2760	2761	2762	2763	2764	2765	2766	2767
4320	2256	2257	2258	2259	2260	2261	2262	2263	5320	2768	2769	2770	2771	2772	2773	2774	2775
4330	2264	2265	2266	2267	2268	2269	2270	2271	5330	2776	2777	2778	2779	2780	2781	2782	2783
4340	2272	2273	2274	2275	2276	2277	2278	2279	5340	2784	2785	2786	2787	2788	2789	2790	2791
4350	2280	2281	2282	2283	2284	2285	2286	2287	5350	2792	2793	2794	2795	2796	2797	2798	2799
4360	2288	2289	2290	2291	2292	2293	2294	2295	5360	2800	2801	2802	2803	2804	2805	2806	2807
4370	2296	2297	2298	2299	2300	2301	2302	2303	5370	2808	2809	2810	2811	2812	2813	2814	2815
4400	2304	2305	2306	2307	2308	2309	2310	2311	5400	2816	2817	2818	2819	2820	2821	2822	2823
4410	2312	2313	2314	2315	2316	2317	2318	2319	5410	2824	2825	2826	2827	2828	2829	2830	2831
4420	2320	2321	2322	2323	2324	2325	2326	2327	5420	2832	2833	2834	2835	2836	2837	2838	2839
4430	2328	2329	2330	2331	2332	2333	2334	2335	5430	2840	2841	2842	2843	2844	2845	2846	2847
4440	2336	2337	2338	2339	2340	2341	2342	2343	5440	2848	2849	2850	2851	2852	2853	2854	2855
4450	2344	2345	2346	2347	2348	2349	2350	2351	5450	2856	2857	2858	2859	2860	2861	2862	2863
4460	2352	2353	2354	2355	2356	2357	2358	2359	5460	2864	2865	2866	2867	2868	2869	2870	2871
4470	2360	2361	2362	2363	2364	2365	2366	2367	5470	2872	2873	2874	2875	2876	2877	2878	2879
4500	2368	2369	2370	2371	2372	2373	2374	2375	5500	2880	2881	2882	2883	2884	2885	2886	2887
4510	2376	2377	2378	2379	2380	2381	2382	2383	5510	2888	2889	2890	2891	2892	2893	2894	2895
4520	2384	2385	2386	2387	2388	2389	2390	2391	5520	2896	2897	2898	2899	2900	2901	2902	2903
4530	2392	2393	2394	2395	2396	2397	2398	2399	5530	2904	2905	2906	2907	2908	2909	2910	2911
4540	2400	2401	2402	2403	2404	2405	2406	2407	5540	2912	2913	2914	2915	2916	2917	2918	2919
4550	2408	2409	2410	2411	2412	2413	2414	2415	5550	2920	2921	2922	2923	2924	2925	2926	2927
4560	2416	2417	2418	2419	2420	2421	2422	2423	5560	2928	2929	2930	2931	2932	2933	2934	2935
4570	2424	2425	2426	2427	2428	2429	2430	2431	5570	2936	2937	2938	2939	2940	2941	2942	2943
4600	2432	2433	2434	2435	2436	2437	2438	2439	5600	2944	2945	2946	2947	2948	2949	2950	2951
4610	2440	2441	2442	2443	2444	2445	2446	2447	5610	2952	2953	2954	2955	2956	2957	2958	2959
4620	2448	2449	2450	2451	2452	2453	2454	2455	5620	2960	2961	2962	2963	2964	2965	2966	2967
4630	2456	2457	2458	2459	2460	2461	2462	2463	5630	2968	2969	2970	2971	2972	2973	2974	2975
4640	2464	2465	2466	2467	2468	2469	2470	2471	5640	2976	2977	2978	2979	2980	2981	2982	2983
4650	2472	2473	2474	2475	2476	2477	2478	2479	5650	2984	2985	2986	2987	2988	2989	2990	2991
4660	2480	2481	2482	2483	2484	2485	2486	2487	5660	2992	2993	2994	2995	2996	2997	2998	2999
4670	2488	2489	2490	2491	2492	2493	2494	2495	5670	3000	3001	3002	3003	3004	3005	3006	3007
4700	2496	2497	2498	2499	2500	2501	2502	2503	5700	3008	3009	3010	3011	3012	3013	3014	3015
4710	2504	2505	2506	2507	2508	2509	2510	2511	5710	3016	3017	3018	3019	3020	3021	3022	3023
4720	2512	2513	2514	2515	2516	2517	2518	2519	5720	3024	3025	3026	3027	3028	3029	3030	3031
4730	2520	2521	2522	2523	2524	2525	2526	2527	5730	3032	3033	3034	3035	3036	3037	3038	3039
4740	2528	2529	2530	2531	2532	2533	2534	2535	5740	3040	3041	3042	3043	3044	3045	3046	3047
4750	2536	2537	2538	2539	2540	2541	2542	2543	5750	3048	3049	3050	3051	3052	3053	3054	3055
4760	2544	2545	2546	2547	2548	2549	2550	2551	5760	3056	3057	3058	3059	3060	3061	3062	3063
4770	2552	2553	2554	2555	2556	2557	2558	2559	5770	3064	3065	3066	3067	3068	3069	3070	3071

Octal-Decimal Conversion Table (cont)

OCTAL 6000 to 6777	DECIMAL 3072 to 3583

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

OCTAL 7000 to 7777	DECIMAL 3584 to 4095

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

Octal-Decimal Conversion Table (cont)

LESSON IV

PART II. HONEYWELL ALPHANUMERIC CODE

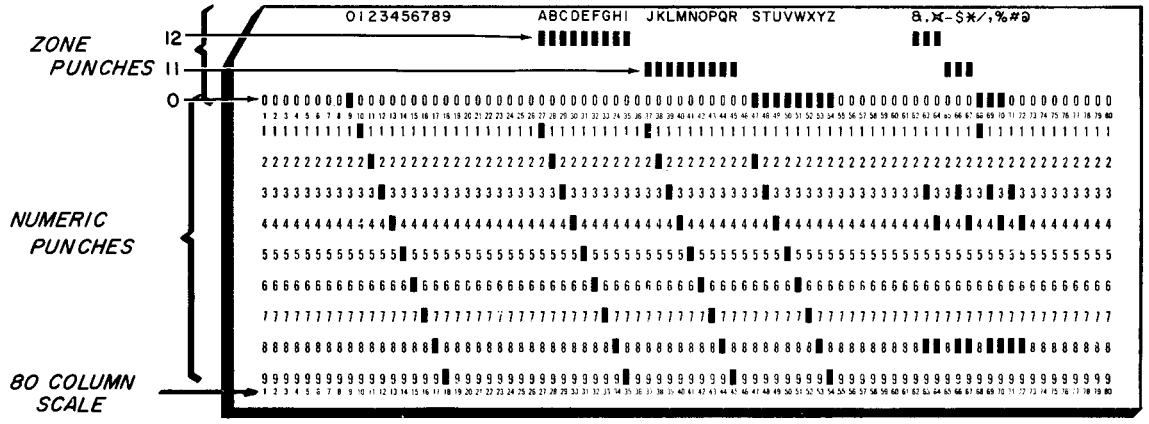


Figure 12. Hollerith Punched Card Code

ALPHABETIC CHARACTERS		
A 01 0001	J 10 0001	S 11 0010
B 01 0010	K 10 0010	T 11 0011
C 01 0011	L 10 0011	U 11 0100
D 01 0100	M 10 0100	V 11 0101
E 01 0101	N 10 0101	W 11 0110
F 01 0110	O 10 0110	X 11 0111
G 01 0111	P 10 0111	Y 11 1000
H 01 1000	Q 10 1000	Z 11 1001
I 01 1001	R 10 1001	

DECIMAL DIGITS	
0 00 0000	5 00 0101
1 00 0001	6 00 0110
2 00 0010	7 00 0111
3 00 0011	8 00 1000
4 00 0100	9 00 1001

SPECIAL CHARACTERS		
' 00 1010	) 01 1100	■ 10 1111
= 00 1011	% 01 1101	■ 11 0000
: 00 1100	□ 01 1110	/ 11 0001
Blank 00 1101	■ 01 1111	@ 11 1010
■ 00 1110	- 10 0000	, 11 1011
& 00 1111	# 10 1010	(11 1100
+ 01 0000	$ 10 1011	C_R 11 1101
; 01 1010	* 10 1100	■ 11 1110
. 01 1011	" 10 1101	■ 11 1111
	■ 10 1110	

■ = Non standard symbol. Printed blank by standard printer.

Figure 13. Alphanumeric Representation

61. Punched card code is shown in Figure 12. (NOTE: This code should properly be referred to by the name of its originator, Dr. Herman Hollerith. You may have been accustomed to improperly calling it by a company name in your previous programming.)

To assure yourself that _____ code is the same punched card code with which you are familiar, notice the designation of digits and letters according to: no zone punch, 12 zone punch, 11 zone punch, 0 zone punch.

---

64. The relationship between Hollerith groups, zone punches and the BA cores is shown below:

BA	GROUP	CONTAINS	ZONE PUNCHED
0 0	0	0-9	NONE
0 1	1	A-1	12
1 0	2	J-R	11
1 1	3	S-Z	0

Write the binary digits to designate the group containing:

4 0 0, E 0 1, L 1 0, W 1 1

---

67. Check the correctness of the chart constructed in frame 66 by referring to the chart printed on the reverse side of the EASYCODER NOTES PAGE (from LESSON I).

B A

Each letter in GROUP "3" (1 1) is numerically designated by the 8421 cores as being

ONE MORE than the position it occupies.

---

70. The octal numbering system is simply a shorthand method of expressing six bits by writing only two digits. It is called octal because it uses powers of 8 and is compatible with the binary base 2 because the THIRD power of two (2^3)= 8.

---

73. Use the cross reference chart to locate the following characters, then write both the octal and binary designations.

H O N E Y W E L L

OCTAL __ __ __ __ __ __ __ __ __

BINARY _____ _____ _____ _____ _____ _____ _____ _____ _____

61.

HOLLERITH

64.

B A
4 = 0 0
E = 0 1
L = 1 0
W = 1 1

	12 Punch AND	11 Punch AND	0 Punch AND
If only a numeric punch is in any column it represents whatever number is punched out	1-A 2-B 3-C 4-D 5-E 6-F 7-G 8-H 9-I	1-J 2-K 3-L 4-M 5-N 6-O 7-P 8-Q 9-R	2-S 3-T 4-U 5-V 6-W 7-X 8-Y 9-Z
Group "0" BA = 00	Group "1" BA = 01	Group "2" BA = 10	Group "3" BA = 11

67.

ONE GREATER (MORE)

70.

THIRD power of two
$2^3 = 8$

73.

H	O	N	E	Y	W	E	L	L
30	46	45	25	70	66	25	44	44
011000	100110	100101	010101	111000	110110	010101	100100	100100

62. Hollerith code is divided into four groups referred to as Group "0" containing 0-9, "1" with A-I, "2" with J-R, and designated by the absence or presence of 12, 11, 0, Zone punches. Complete this chart.

GROUP	CONTAINS	ZONE PUNCHED
0	0-9	
1	A-I	
2	J-R	
3	S-Z	

65. The 8, 4, 2, 1 cores specify the numeric punch. This designates the position of the character within the group identified by the BA cores. Example: BA 8421 / 01 0101 is GROUP "1", FIFTH CHARACTER, which is E. Refer to the chart at the left as needed to decode:

12 punch & 8,	BA 8421, 00 0010,	BA 8421, 00 0000,	BA 8421, 00 0000,				
12 punch & 3,	10 0110,	10 0100,	10 0111,	11 0100,	11 0011,	01 0101,	10 1001

68. Also on the back of the Basic EASYCODER NOTES page is a reference chart for all the Honeywell alphanumeric letters, digits, and special symbols. The example shows how to decode binary 101011. Similarly, you can encode by locating a character, reading the column to the left to locate the FIRST THREE BITS and reading the column at the top for the SECOND THREE BITS.

71. The relationship between base 2 and base 8 is shown by writing the decimal values in this chart.

BINARY	2^6	2^3	2^0
OCTAL	8^2	8^1	8^0
DECIMAL	64	8	1

74. Octal designation of six bit binary numbers is a convenience that will become familiar through practice. Simply remember that each octal digit is formed by a combination of the 4, 2, and 1 bits. Encode:

101 010 000 111 011 110 110 111

62.

GROUP	CONTAINS	ZONE PUNCHED
0	0-9	NONE
1	A-I	12
2	J-R	11
3	S-Z	0

65.

H-200
COMPUTER

68.

FIRST THREE BITS
SECOND THREE BITS

71.

BINARY	2^6	2^3	2^0
OCTAL	8^2	8^1	8^0
DECIMAL	64	8	1

74.

52 07 36 67

63. Hollerith punched card code is the basis for Honeywell's alphanumeric code. The character storage portion of a memory location (BA 8421 cores) designates both the Hollerith group and the numeric punch as binary numbers. Letting the B and A cores represent binary 1's or 0's, write the two bit binary number for each Hollerith group.

	B	A
GROUP "0" =	0	0
GROUP "1" =	0	1
GROUP "2" =	1	0
GROUP "3" =	1	1

66.

When a chart or table is available, Hollerith or Honeywell codes may be decoded or encoded easily. However, if you were without a reference, it would not be too difficult to construct your own chart. As an example, complete the chart on the reverse side of this frame.

69. Notice on the chart example that two methods are shown for expressing digits. One method is BINARY, the other method is called OCTAL and expresses the first three bits with one digit and the second 3 bits with ONE digit.

72. For practice in using octal to denote six bit binary, write the following binary numbers as their octal equivalent.

BINARY	0 1 0 1 0 1	0 1 0 0 0 1	1 1 0 0 1 0	1 1 1 0 0 0
OCTAL	25	21	62	70

Now, use each group of two octal digits to locate the appropriate characters on the cross reference chart that is on the BASIC EASYCODER NOTES page. _ _ _ _ ✗

75. One specific difference between 1401 alphameric and Honeywell alphanumeric code concerns zero and blank.

In 1401 code, 000 000 equals blank and 001 010 equals zero.

However in Honeywell code, zero is logically 000 000 and blank is a special ✗ symbol coded 001 101 .

63.

	B	A
GROUP "0" =	0	0
GROUP "1" =	0	1
GROUP "2" =	1	0
GROUP "3" =	1	1

(Return to page 18, frame 64.)

---

66. CHART: Hollerith Zone and Numeric or Honeywell Alphanumeric

NUMERIC ONLY	12 ZONE & NUMERIC	11 ZONE & NUMERIC	0 ZONE & NUMERIC
GROUP "0"	GROUP "1"	GROUP "2"	GROUP "3"
B A 8421	B A 8421	B A 8421	B A 8421
0 = 0 0 0000	A =	J =	S = 1 1 0010
1 =	=	=	=
2 =	=	=	=
3 =	=	=	=
4 =	=	=	=
5 =	=	=	=
6 =	=	=	=
7 =	=	=	Z =
8 =	I =	R =	
9 =			

(Return to page 18 frame 67.)

69.

OCTAL
THREE BITS with ONE DIGIT

(Return to page 18 frame 70.)

72.

BINARY	0 1 0 1 0 1	0 1 0 0 0 1	1 1 0 0 1 0	1 1 1 0 0 0
OCTAL	2 5	2 1	6 2	7 0
CHARACTERS:	E	A	S	Y

(Return to page 18 frame 73.)

75.

HONEYWELL ZERO = 000 000
HONEYWELL BLANK = 001 101

(Continue to page 85.)

LESSON V

STORAGE, RETRIEVAL AND EXECUTION

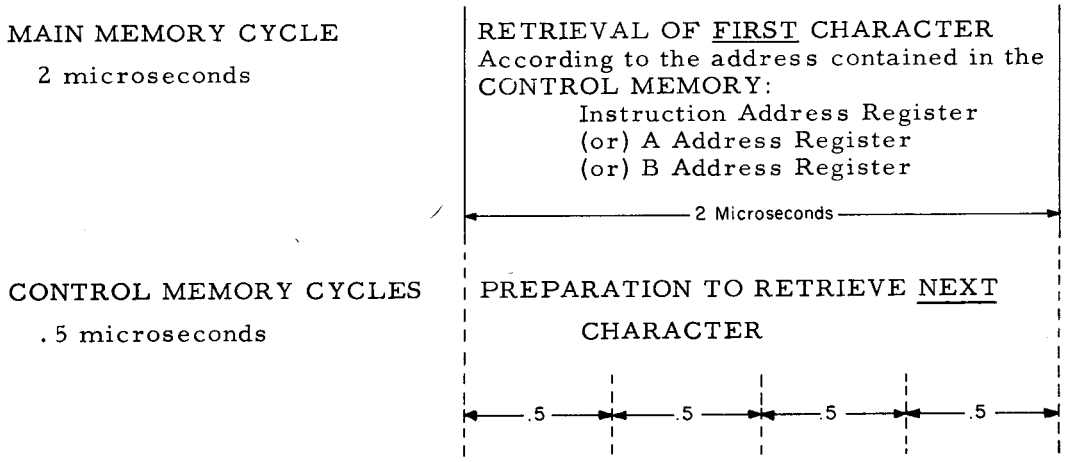
STORAGE, RETRIEVAL AND EXECUTION

From a programmers' standpoint, there are two obvious H-200 superiorities apparent when contrasted with 1401 operation:

	H-200		1401
MEMORY CYCLE	2 microseconds	vs.	11.5 microseconds
SIMULTANEITY	Multiple Operations	vs.	Serial Operations

Simultaneity and the ability to take advantage of fast memory cycle time are made possible by the H-200's CONTROL MEMORY. Registers in control memory provide simultaneity of peripheral operations with computation and also contribute to memory cycle of less than one fifth that of the 1401. Control memory cycle of the 16 available registers is 500 billionths of a second. Consequently, control memory has four complete cycles in which operations may be accomplished during a 2 microsecond main memory cycle.

Some of the control memory operations performed are to assist retrieval from main memory. (Selecting addresses, interpreting addresses, directing retrieval, directing arithmetic functions, etc.) The illustration below shows how control memory operations overlap a main memory cycle enabling memory locations to be accessed and retrieved in only 2 microseconds.



Four complete control memory cycles occur during a main memory cycle. The control unit selects the appropriate register, interprets an address, etc. This prepares for retrieval or execution of the NEXT character while main memory is retrieving the previous character. A 1401 requires 230 microseconds to select, interpret, retrieve and execute typical Add instruction. The fast memory cycle of the H-200, aided by its control memory, can accomplish the same instruction in only 44 microseconds.

It should be remembered that while H-200 central processor operations are much faster than the 1401, they are also SIMULTANEOUS WITH PERIPHERAL OPERATIONS. For example, the H-200 can simultaneously: Read or write 4360 tape records of 500 characters each, punch 250 cards, read 800 cards, print 900 lines of 120 characters each, and execute 1,000,000 instructions in one minute.

1. Prior to this lesson, reference has mostly been relative to main memory, i.e., the 2048 memory locations of a basic H-200. The control unit uses a small memory bank for computer control descriptively named CONTROL MEMORY.

---

9. The rule for punctuating an instruction is simpler than the several rules for punctuating data. Instructions are stored in consecutive memory locations with a WORD MARK in the leftmost (high order) memory location of each instruction. Therefore the OP CODE of each instruction will contain a WORD MARK

---

17. After retrieval of the B address portion of the instruction, control memory will contain the A address in the A ADDRESS REGISTER and the B address in the B ADDRESS REGISTER. It should be remembered that these addresses are stored as 12 or 18 bits.

---

25. The further specification or modification of an OP. CODE is the purpose of a VARIANT CHARACTER. One or more of these "modifiers" may be included as the rightmost memory location of the three instruction formats illustrated in frame 23.

EXAMPLES:

OP. CODE	VARIANT

OP. CODE	A ADDRESS	VARIANT	VARIANT

---

33. Six registers in control memory operate as counters and are assigned to the three read/write channels. All three pairs of read/write channel counters function identically. Therefore, only those associated with Read/Write Channel 1 will be introduced.

1.

CONTROL MEMORY

9.

OP.
WORD MARK

17.

A ADDRESS REGISTER

B ADDRESS REGISTER

25.

VARIANT CHARACTER

33.

NO ANSWER REQUIRED

2. Control memory is a matrix of cores providing 16 memory locations 18 cores in length. The 18 cores of each control memory location will store up to 3 six bit CHARACTERS.

---

10. The control unit starts retrieving an instruction at the leftmost memory location, the OP. CODE. The computer is designed to ignore the first WORD MARK sensed during Instruction Retrieval. Retrieval continues from left to right until the WORD mark in the OP CODE of the next instruction is sensed.

---

18. "Two character addressing mode" (2 memory locations - 12 continuous bits) is sufficient to address any H-200 memory location up to #4096.

Example: 000000001101_2 = Address $\#13_{10}$

111111111111_2 = Address $\#4095_{10}$

The decimal address 757 can be stated as _ _ _ _ _ _ _ _ _ _ _ _.

Note: A two character address must contain 12 bits.

---

26. An OP. CODE register is part of the control unit and the purpose of a VARIANT character is to modify or further specify an operation. Consequently, in addition to the nine registers of control memory, the CONTROL unit must contain a register for both the OP CODE and VARIANT characters.

---

34. Read/write channel counters are control memory registers which store the starting location and current location addresses of data being transferred. Descriptively named, they are the STARTING LOCATION counter and the CURRENT LOCATION counter.

2.

CHARACTERS

10.

WORD

OP. CODE

18.

$757_{10} = 1011110101_2$

TO CONTAIN 12 BITS WRITTEN AS 001011110101_2

26.

CONTROL
OP. CODE
VARIANT

34.

STARTING LOCATION

CURRENT LOCATION

3. The 16 memory locations in CONTROL MEMORY are called control registers. These control registers store the main memory addresses of data and instructions to be used by the control unit. The control unit sequentially performs the functions of selection, interpretation, and execution of instructions.

---

11. As you remember, a data word is retrieved from right to left and terminates with the leftmost memory location which contains a WORD MARK. Instruction retrieval is opposite to that of data; that is, retrieval is from LEFT to RIGHT and effectively terminates when the WORD MARK of the next instruction OP. CODE is sensed.

---

19. The A and B address registers will each contain at least 2 six bit characters (12 bits 1's and 0's) indicating the main memory address of the operands. Using the example S, 126, 141, the A address register would contain the binary equivalent of 126 and the B address register would contain the binary equivalent of 141.

A address register contents = _____

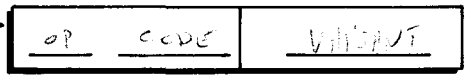
B address register contents = _____

Note: Add binary zeros to the left to make complete 12 bit addresses.

---

27. Identify each part of the instruction formats described below.

Modified Single Character Instruction



Modified Single Operand Instruction

Modified Two Operand Instruction

---

35. As each successive character is transferred, the current location counter is incremented by one. Therefore, when transfer ceases, the address of the character position immediately following the last character transferred will be found in the CURRENT LOCATION COUNTER.

3.

CONTROL MEMORY

11.

LEFT

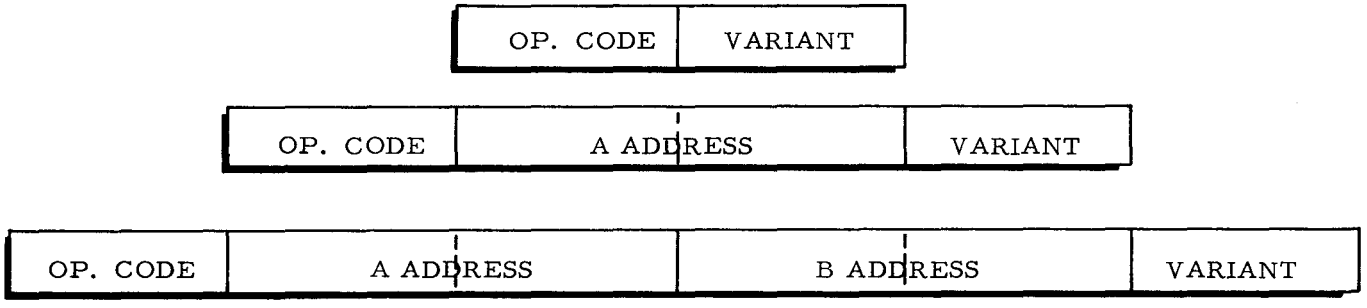
RIGHT

19.

126_{10}=000001111110

141_{10}=000010001101

27.



35.

CURRENT LOCATION COUNTER

4. Control memory contains 16 memory locations called CONTROL REGISTERS.
A basic H-200 uses the 9 listed below:
(1) Instruction Address Register
(1) A Address Register
(1) B Address Register
(6) Two registers for each of the 3 Read/Write Channels.
The remaining 7 registers are available for use with features such as Advanced Programming etc.

---

12. The first character retrieved according to the address in the Instruction Address Register is the OP. CODE. This single character is placed in a control unit register (NOT one of the control memory registers). Named for the character it contains, it is simply called an OP CODE register.

---

20. Memory beyond 4096 locations requires a change of addressing mode to "Three character address." This will involve THREE memory locations - 18 bits - for an operand address.

---

28. What is the least number of memory locations for the shortest instruction? ONE. How many memory locations are required for an instruction in "2 character addressing mode" containing an A operand and two variant characters? 5.

---

36. The current location counter will contain the memory address of the next character to be transferred. The main memory address from which or to which transfer began is stored in the STARTING LOCATION COUNTER.

4.

CONTROL REGISTERS

12.

OP. CODE

20.

THREE

28.

ONE

FIVE

36.

STARTING LOCATION COUNTER

5. Basic control memory uses nine CONTROL registers. Since a computer must rely on instructions and the location in memory of these instructions is a prerequisite to their use, the first register to be considered is the INSTRUCTION address register.

---

13. The OP. CODE character is interpreted by the control unit. Retrieval of the remaining instruction characters then follows. A common instruction format such as S, 126, 141, contains first the OP CODE, next the A ADDRESS and finally the B ADDRESS. This is the most common but only one of six possible formats.

---

21. As the operation code character was retrieved, the instruction address register incremented by one. Since the operation code is only one character, the incremented instruction address register would then contain the address of the first character of the A ADDRESS.

---

29. In summary of instruction retrieval: Retrieval of instruction characters is directed by the INSTRUCTION ADDRESS register in control memory, which is incremented by 1 as each character is retrieved. The OP. CODE is stored in the OP CODE register of the control unit. Operand addresses are stored in the A and B ADDRESS REGISTERS of control memory. Variant characters (if present) are stored in the VARIANT register of the control unit. Instruction execution commences when the WORD mark of the next instruction OP. CODE is sensed.

---

37. The computer operator or programmer can determine where data transfer begins and where it ends by referring to the STARTING LOCATION counter and the CURRENT LOCATION counter associated with each READ / WRITE channel. Besides resumption of data transfer at the proper location, these counters can determine length of records, etc.

5.

CONTROL
INSTRUCTION

13.

OP. CODE
ADDRESS
ADDRESS

21.

A ADDRESS

29.

INSTRUCTION ADDRESS
ONE
OP. CODE
A
B ADDRESS REGISTERS
VARIANT
WORD

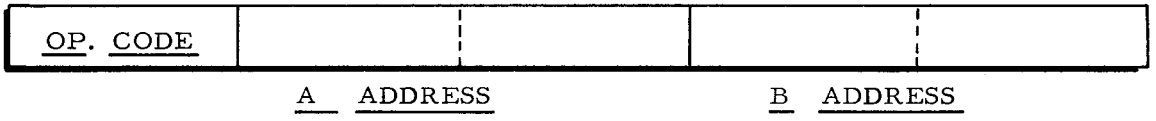
37.

CURRENT LOCATION
PRESENT LOCATION
READ/WRITE

6. The programmer specifies the main memory address of the first instruction to be selected, interpreted and executed by the control unit. The control unit is directed to the proper main memory location by this ADDRESS placed in the INSTRUCTION ADDRESS REGISTER.

---

14. An "A" operand is retrieved from main memory by the control unit according to its "A" ADDRESS in the instruction. Consequently a basic H-200 operation involving both "A" and "B" operands must contain A and B ADDRESSES.



---

22. Each time the control unit selects or retrieves an instruction character, the instruction address register increments by one. As the last character of the B address is retrieved, the instruction address register will contain the address of the OP CODE of the next instruction.

---

30. To this point in the lesson, only three of the nine control registers of a basic H-200 control memory have been introduced. They are the INSTRUCTION ADDRESS register, A ADDRESS register, and B ADDRESS register. Read/Write Channel Time Sharing uses six registers, two for each channel.

---

38. List the nine control registers of the basic H-200.

1. INSTRUCTION ADDRESS REGISTER
2. A ADDRESS REGISTER
3. B ADDRESS REGISTER
4. RWC#1 START TAG LOCATION COUNTER
5. RWC#1 [illegible]
6. RWCH1 SLC
7. RWCH1 CLC
8. RWC#2 SLC
9. RWC#2 CLC

6.

ADDRESS
INSTRUCTION ADDRESS REGISTER

14.

A and B ADDRESSES

22.

OP. CODE

30.

INSTRUCTION ADDRESS
A ADDRESS
B ADDRESS

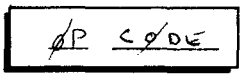
38.

INSTRUCTION ADDRESS REGISTER (IAR)
A ADDRESS REGISTER (AAR)
B ADDRESS REGISTER (BAR)

RWC#1 { STARTING LOCATION COUNTER
CURRENT LOCATION COUNTER

RWC#2 { STARTING LOCATION COUNTER
CURRENT LOCATION COUNTER

RWC#3 { STARTING LOCATION COUNTER
CURRENT LOCATION COUNTER

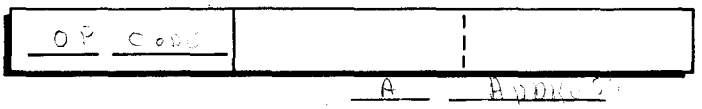
7. An OP. CODE is always in the first (leftmost) memory location of an instruction. This single character code may sometimes be a complete instruction. As such, it is simply illustrated as one memory location block identified as an 

.

---

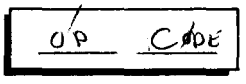
15. An instruction will contain the 12 or 18 bit address of the "A" operand. Since 12 bits are required to express any address up to #4096, storage of an "A" address up to #4096 will require TWO memory locations. Identify parts of the instruction shown below.

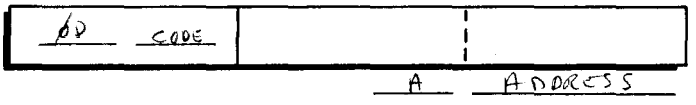
MEMORY LOCATIONS

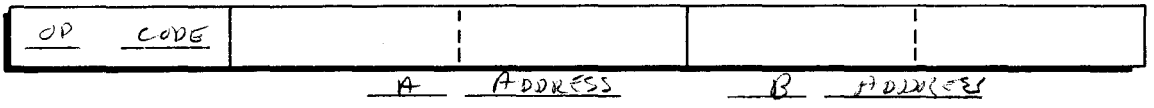


---

23. Three instruction formats have been discussed. Identify each part of these formats according to the description given.

Single Character Instruction 

Single Operand Instruction 

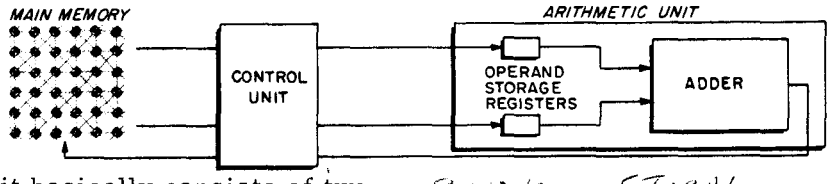
Two Operand Instruction 

---

31. Time sharing permits a second peripheral device to use the central processor during mechanical operations of the first device. This second input or output operation can in turn share access to main memory with a third. Any of these data transfer operations are communicated through the READ / WRITE channels.

---

39. During the execution phase, retrieval of data from memory and its transfer to the arithmetic unit is illustrated in this diagram.



The arithmetic unit basically consists of two OPERAND STORAGE REGISTERS and an ADDER.

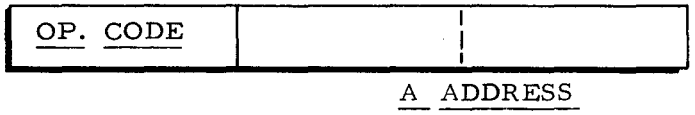
7.

OP. CODE

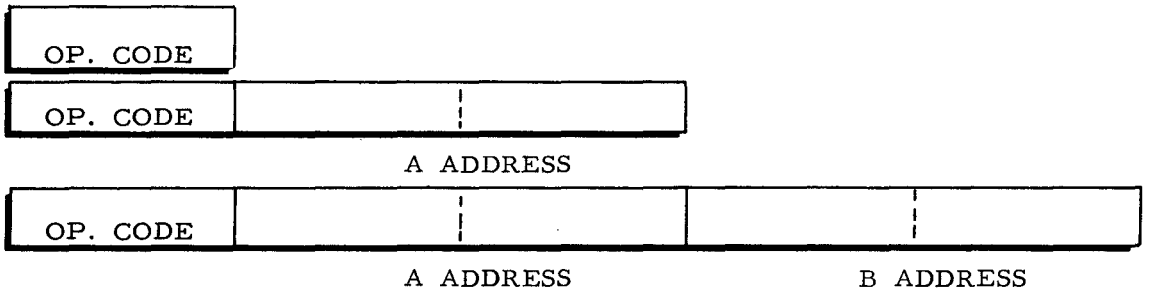
15.

TWO (2)

MEMORY LOCATIONS



23.



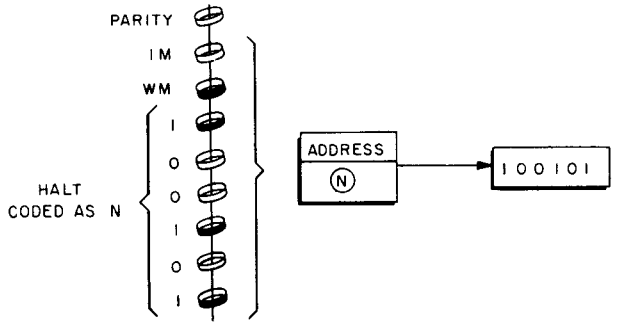
31.

READ/WRITE

39.

OPERAND STORAGE REGISTERS
ADDER

8. The operation code (OP. CODE), a six bit character, is interpreted (decoded) by the control unit. For example, HALT is coded as an alphanumeric N and stored in the first (leftmost) memory location of an instruction.



This block illustrates the first memory location of an instruction and therefore contains an OP CODE.

16. Just as the instruction address was stored in the instruction address register, the A address is stored in the A ADDRESS REGISTER. This address indicates the main memory location of the A operand.

24. Many of the H-200 OP. CODES may be further specified by including one or more VARIANT characters at the end of the instruction. For example, "Change Addressing Mode" is an OP. CODE telling the computer basically what to do.

To further specify the change as 2 or 3 character addressing mode requires a VARIANT character at the END of the INSTRUCTION.

32. Data transfer between peripheral devices and the central processor is provided by the READ / WRITE CHANNELS. Assignment of a channel is determined by a programmed instruction. After an operation is completed, the RWC can be reassigned to another peripheral device.

40. Following retrieval from memory, operands are transferred to the OPERAND STORAGE registers one character at a time. Each pair of characters in the registers (one character from each register) is then combined by the ADDER.

8.

OP. CODE

(Return to page 87, frame 9.)

16.

A ADDRESS REGISTER

(Return to page 87, frame 17.)

24.

VARIANT END INSTRUCTION

(Return to page 87, frame 25.)

32.

READ/WRITE CHANNELS

(Return to page 87, frame 33.)

40.

OPERAND STORAGE
ADDER

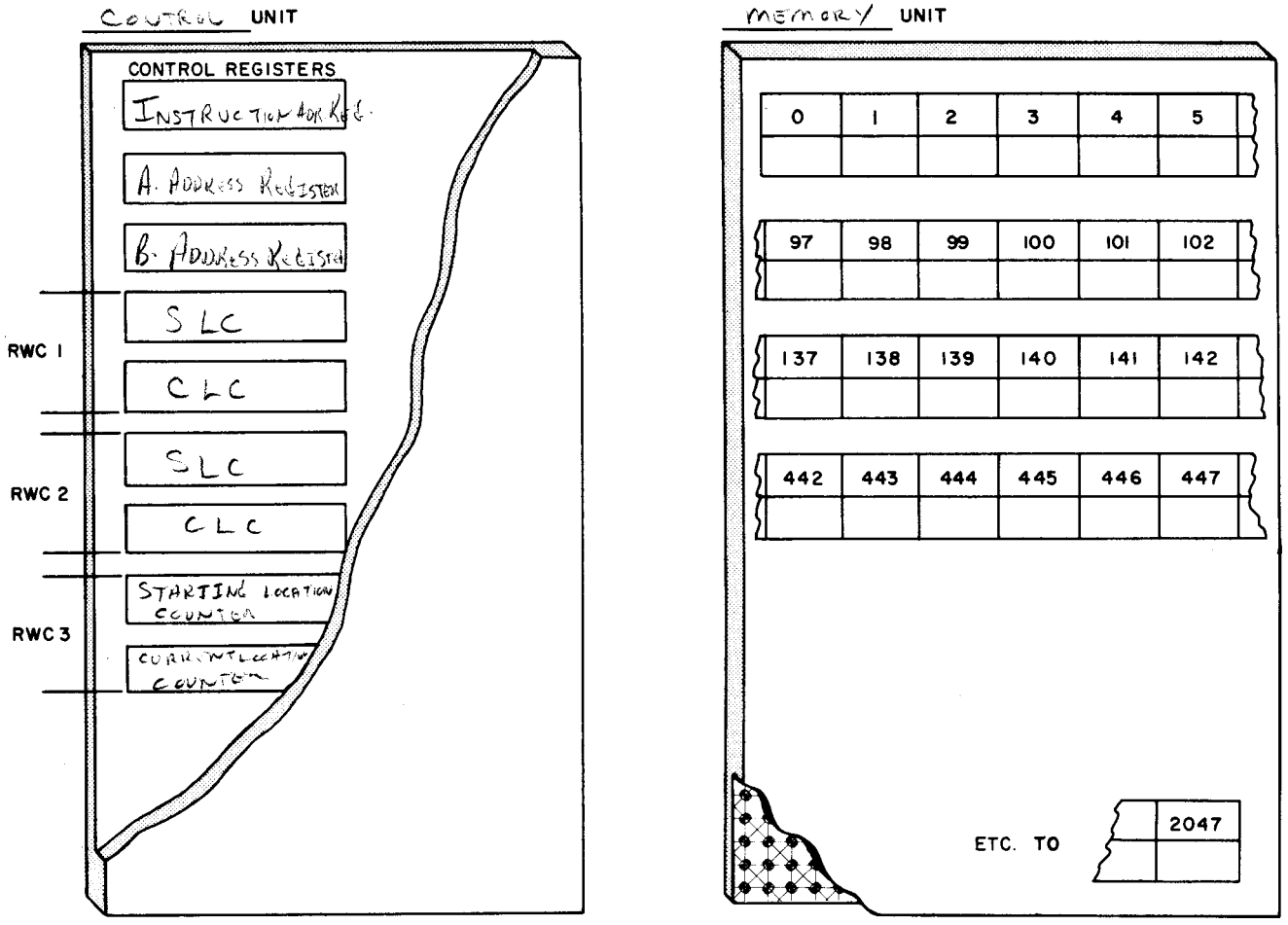
(Continue to page 103.)

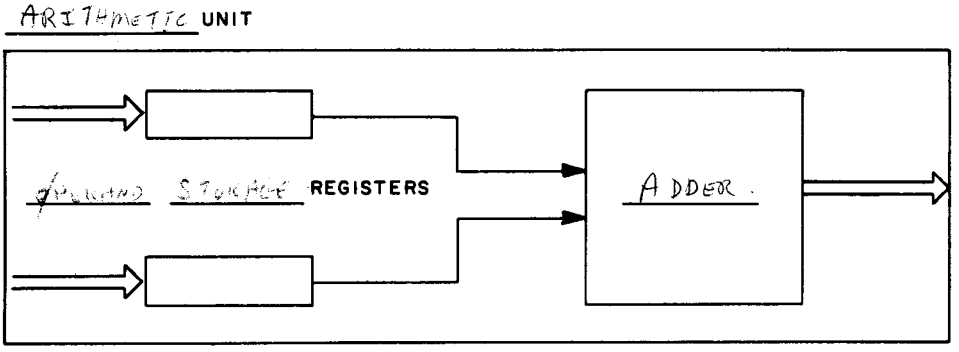
41. Three units of the Central Processor are symbolized below.

Identify each unit by writing its name in the blank at the top of the block.

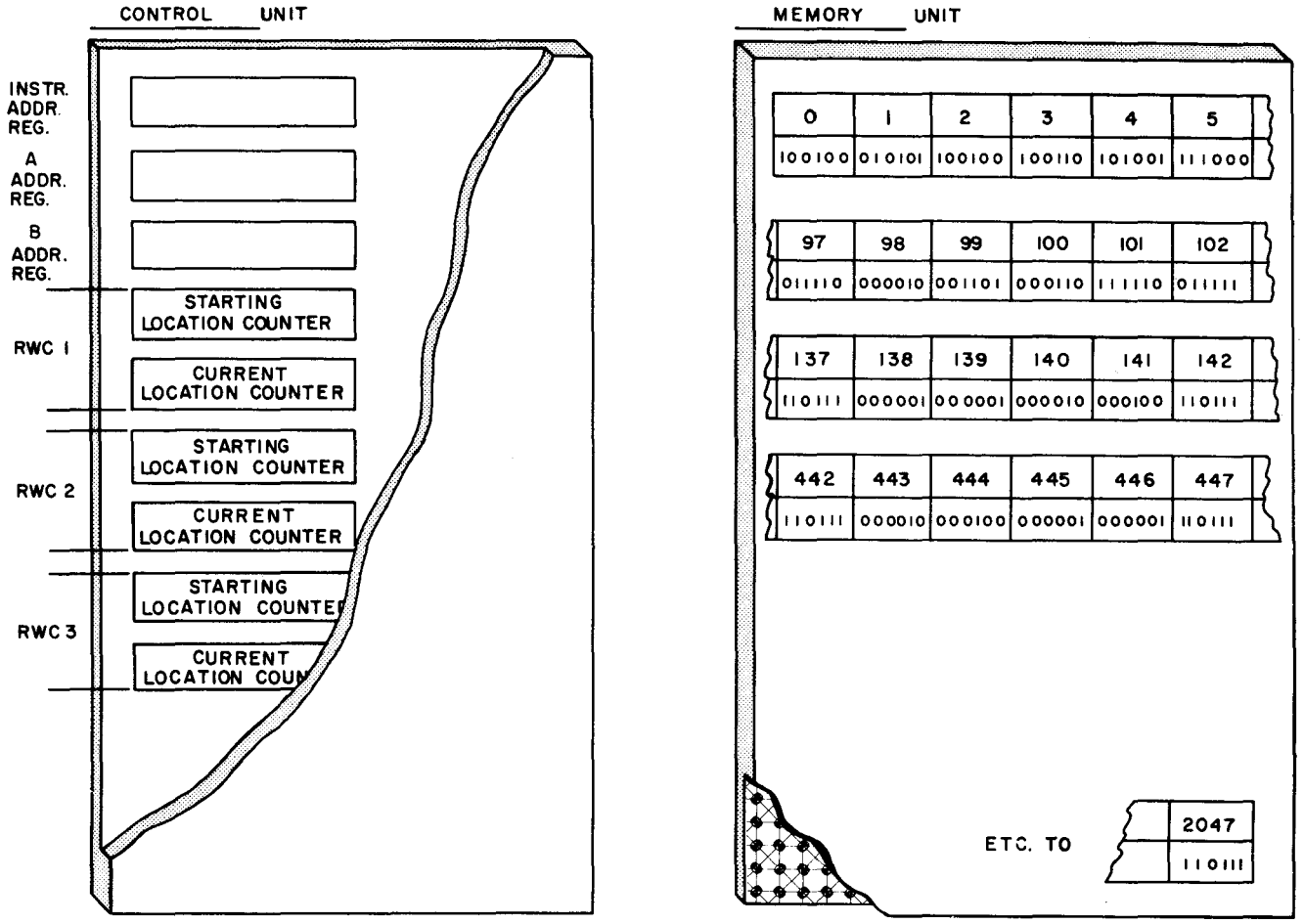
Name each of the nine control registers.

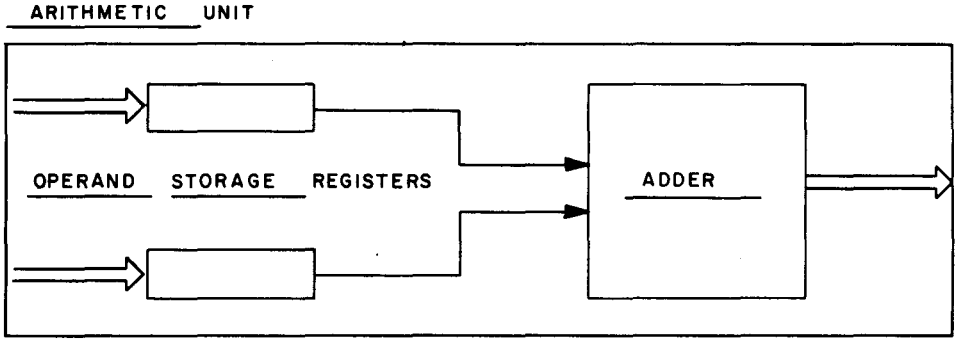
Name the components in the unit at the bottom of the page.





41. The program of sequential instructions and the operands (data to be processed) are stored in memory locations within the memory unit. With the exception of operand addresses in an instruction, the block diagrams on the following pages simplify memory location contents by expressing them as alphabetic or decimal characters.

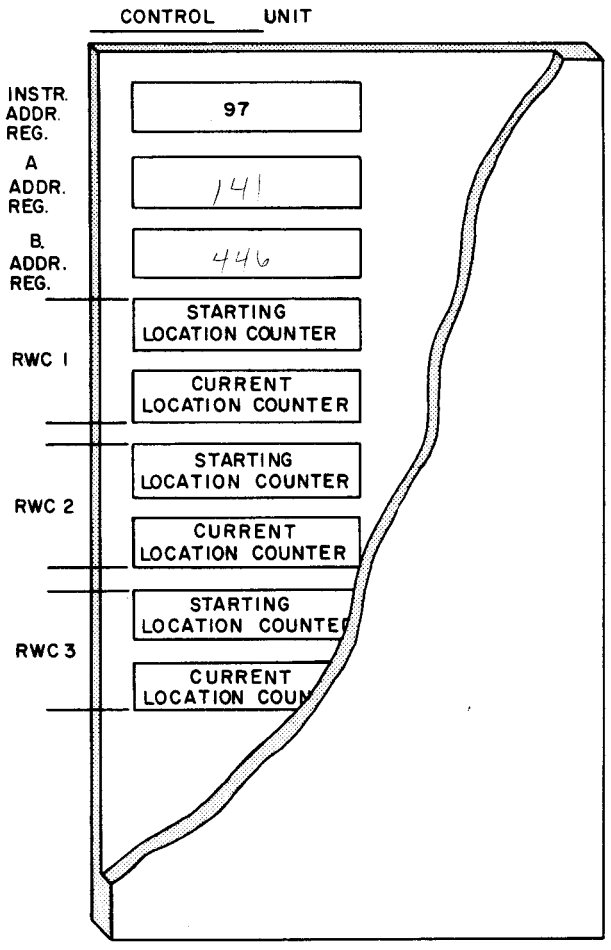


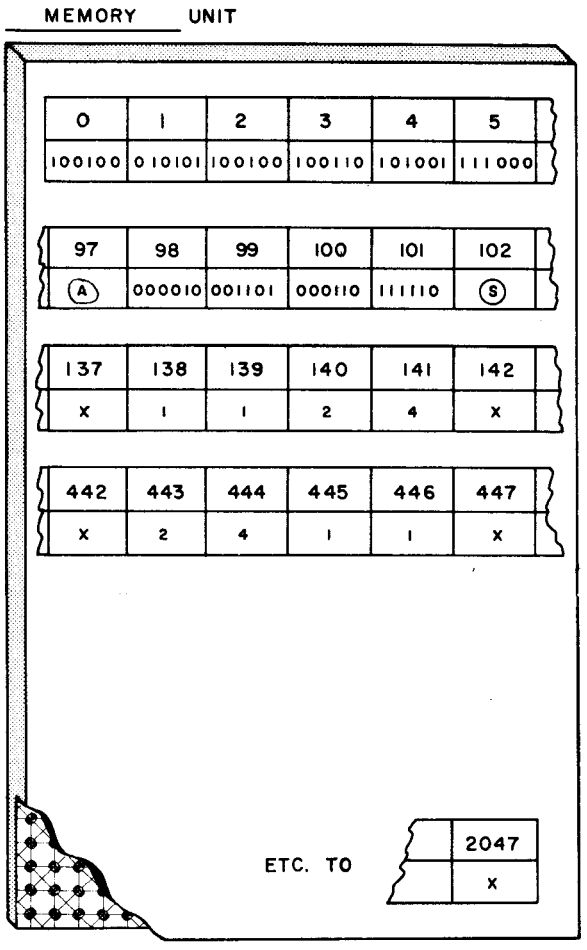


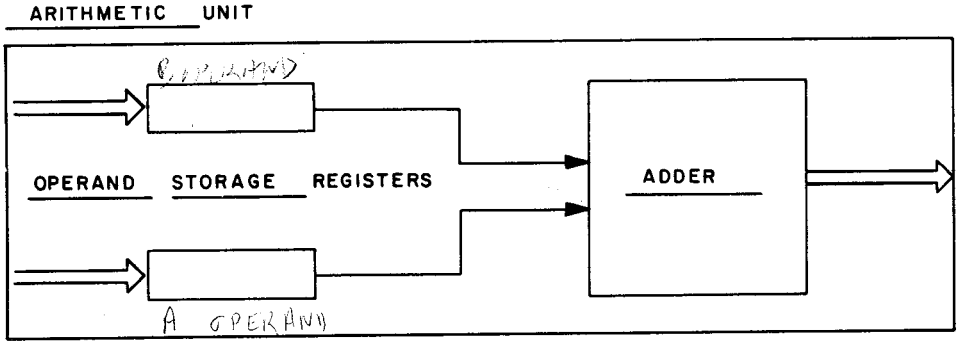
42. Control unit selection of an instruction is directed by the address in the Instruction Address Register (IAR).

OP. CODE	A ADDRESS	B ADDRESS

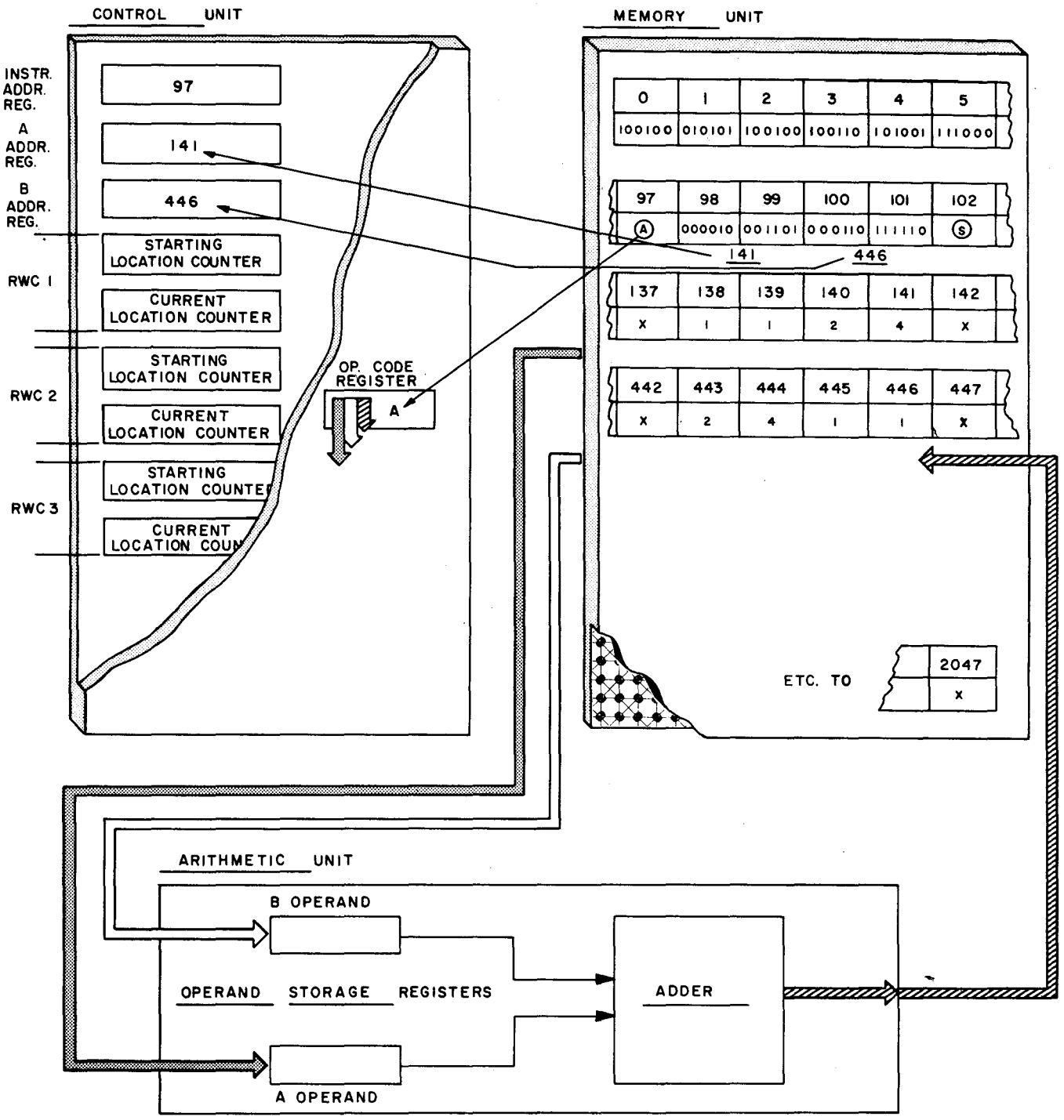
1. Locate and punctuate the instruction.
2. Change the 12 bit A operand address to decimal, write it in the proper register.
3. Change the 12 bit B operand address to decimal, write it in the proper register.







42. The OP. CODE is shown being retrieved for interpretation in the Control Unit Op. Code Register. This register prepares the Arithmetic Unit to receive operands. A and B operand addresses are stored in their respective registers to control the character transfer to the Arithmetic Unit.

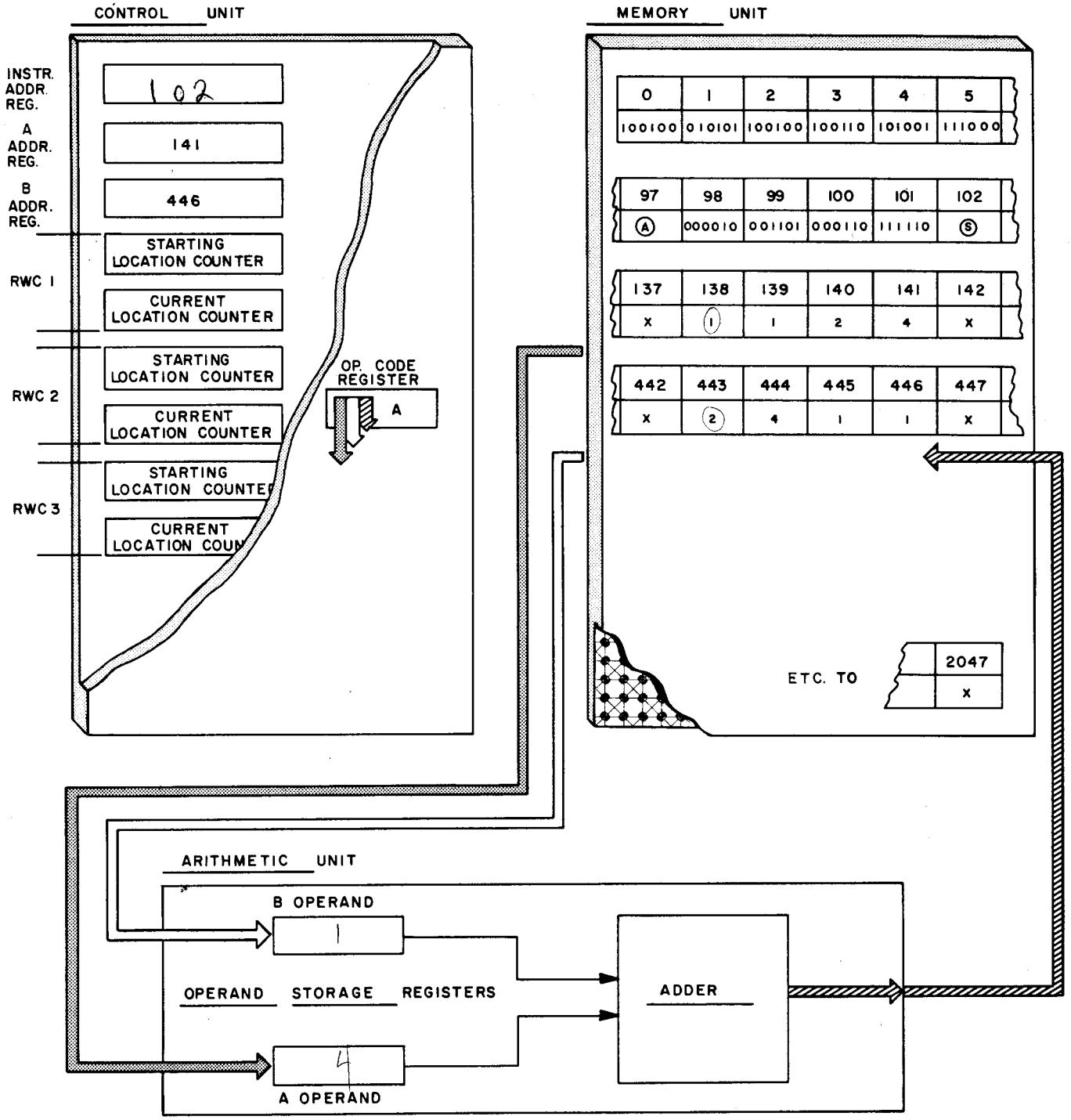


43. The IAR is incremented by one each time an instruction character is accessed.

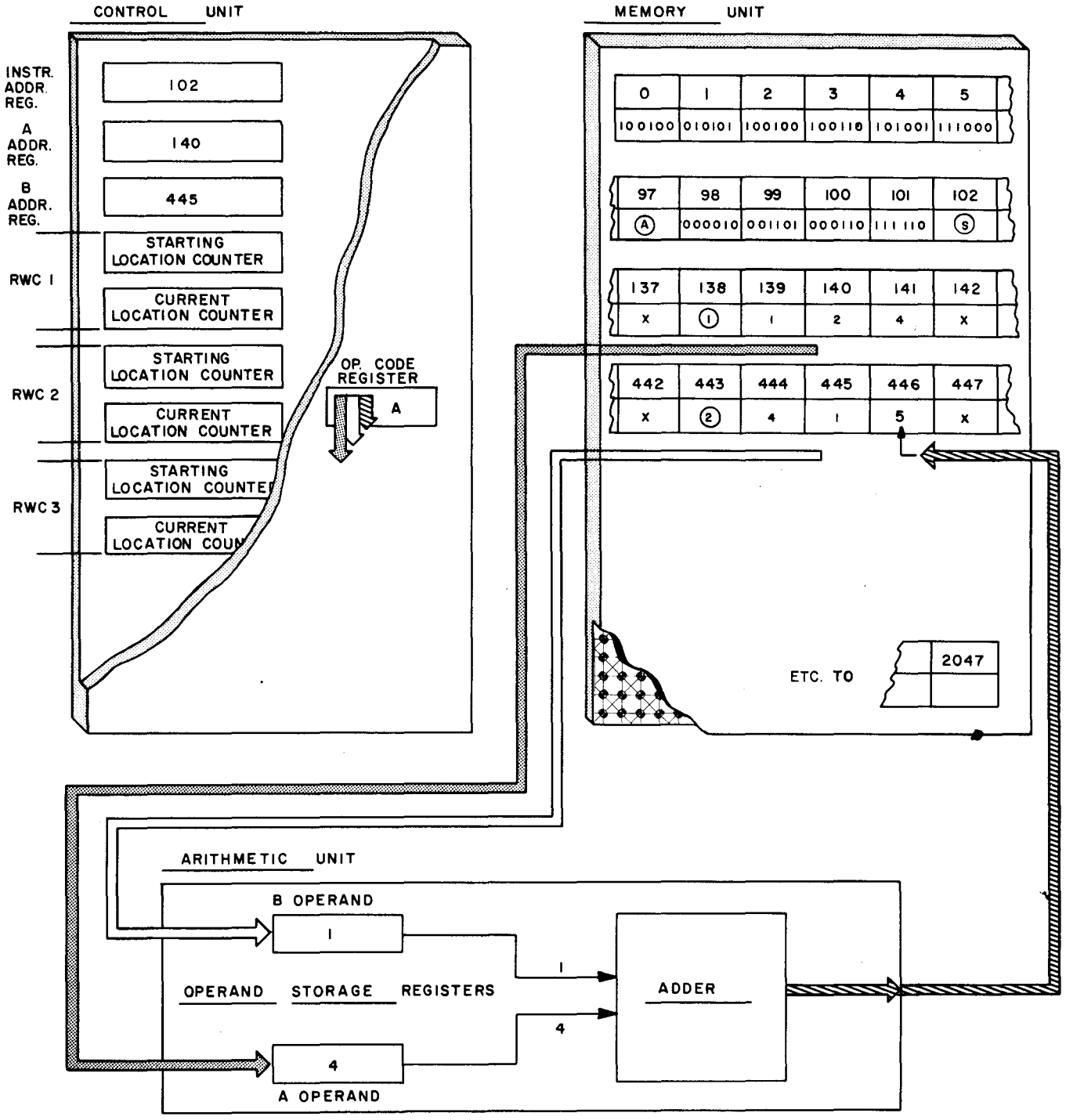
Retrieval continues until the Word Mark of the next instruction Op. Code is sensed.

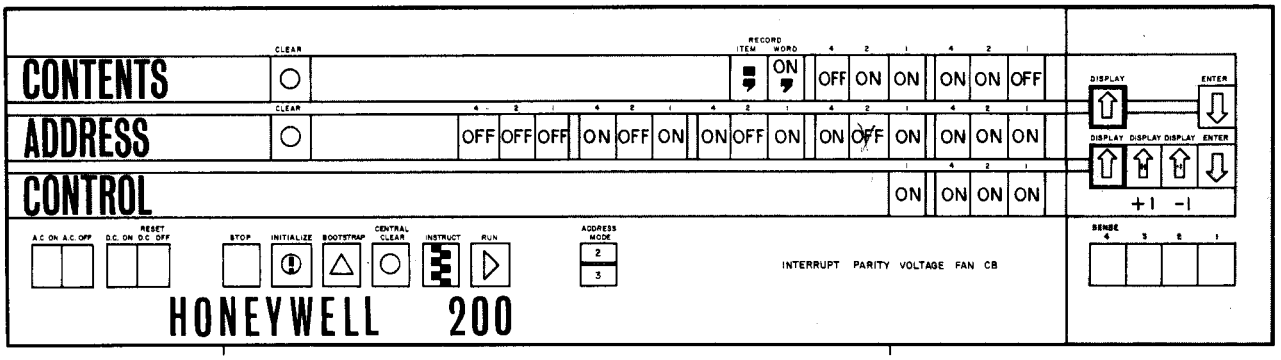
Assume that the instruction A, 141, 446 has been retrieved.

1. Appropriately increment the IAR.
2. Punctuate the operand words 1124 and 2411.
3. Complete the data flow lines to show transfer of the first character of each operand to the Arithmetic Unit.
4. In the Operand Storage Registers, write the first character transferred from each operand.



43. A pair of characters is combined in the Adder and the result is sent back to the B address. As shown below, 1 and 4 are combined and written back into address #446 as 5. This procedure continues to the left until all characters have been added and written into memory. Upon completion of this operation, the Control Unit refers to the incremented IAR to begin retrieving the next instruction. The A Addr. Reg. and B Addr. Reg. decrement by one as each operand character is retrieved. They will contain 137 and 442 at the completion of this operation.





OPERATORS CONTROL PANEL

CONTROL MEMORY REGISTERS	OCTAL ADDRESS
RWC #1 CURRENT LOCATION	01
RWC #2 CURRENT LOCATION	02
RWC #3 CURRENT LOCATION	03
IAR' (CO-SEQUENCE)	04
RWC #1'	05
INTERRUPT REGISTER	06
WORK REGISTER	07
"B" ADDRESS REGISTER	10
RWC #1 STARTING LOCATION	11
RWC #2 STARTING LOCATION	12
RWC #3 STARTING LOCATION	13
"A" ADDRESS REGISTER	14
RWC #1'	15
WORK REGISTER	16
IAR (SEQUENCE)	17
UNASSIGNED	00

A desired CONTROL register is displayed by illuminated and darkened light buttons. (ON=1, OFF=0)

Example: The operator depresses the four CONTROL light buttons. (BINARY 1111, OCTAL 17.) The operator then depresses the DISPLAY button and IAR contents are shown by the ADDRESS light buttons. The ADDRESS illustrated on the control panel above is:

BINARY	000	101	101	111	111
OCTAL	0	5	5	7	7
DECIMAL	0	2	9	4	3

To view CONTENTS of memory location number 2943, the operator presses the upper DISPLAY button. In the illustration, CONTENTS are shown as a WORD MARK and binary digits of an Add op. code. If the operator wishes to see the following or preceding location, he presses the DISPLAY + 1 or DISPLAY - 1 button. CONTENTS or ADDRESS bits may be altered by depressing the desired light buttons and then the appropriate ENTER button.

The table above lists all sixteen CONTROL MEMORY REGISTERS and their OCTAL ADDRESSES. Octal addresses of the nine registers discussed to this point are shown below. Write the name and state the purpose of each of these nine registers. (Answers on page 110.)

OCTAL ADDRESS	NAME	PURPOSE
01	RWC#1 current location counter	
02	RWC#2 ✓ ✓	✓
03	RWC#3 ✓ ✓	✓
10	"B" address Register	
11	RWC #1 Starting location counter	
12	RWC #2	✓ ✓
13	RWC #3	✓ ✓
14	A. ADDR=3	
17	IAR (sequence)	

NOTE: The remaining seven registers will be discussed in following frames.

(Equivalent answers are acceptable.)

01 - RWC #1-Current Location Counter
02 - RWC #2-Current Location Counter
03 - RWC #3-Current Location Counter

Used in conjunction with other counters for simultaneity through read/write channel time sharing. Provides the current address at which transfer is to begin either to or from a peripheral device during allotted 2 microsecond period.

10 - BAR-"B" Address Register - Provides main memory address of B operand character.

11 - RWC #1-Starting Location Counter
12 - RWC #2-Starting Location Counter
13 - RWC #3-Starting Location Counter

Used with other counters. Contain the address at which transfer began either to or from a peripheral device. The numerical difference between the address in SLC and the address in CLC after transfer is complete shows the number of characters transferred.

14 - AAR-"A" Address Register - Provides main memory address of A operand character.
17 - IAR-Instruction Address Register-Provides address of next sequential instruction character to be retrieved. (Sometimes called "sequence register.")

Registers to be discussed in the following frames:

04-IAR' - Instruction Address Register' (Sometimes called "co-sequence register.")
05-RWC #1' - Current Location Counter' (Optional fourth read/write channel)
06-Interrupt Register
07-Work Register
15-RWC #1' - Starting Location Counter' (Optional fourth read/write channel)
16-Work Register
00-Unassigned Register

44. Seven remaining control memory registers are available for special or optional use. These registers are, the: Instruction Address Register' (IAR')
Interrupt Register
Starting Location Counter (RWC #1')
Current Location Counter (RWC #1')
Work Register
Work Register
Unassigned Register

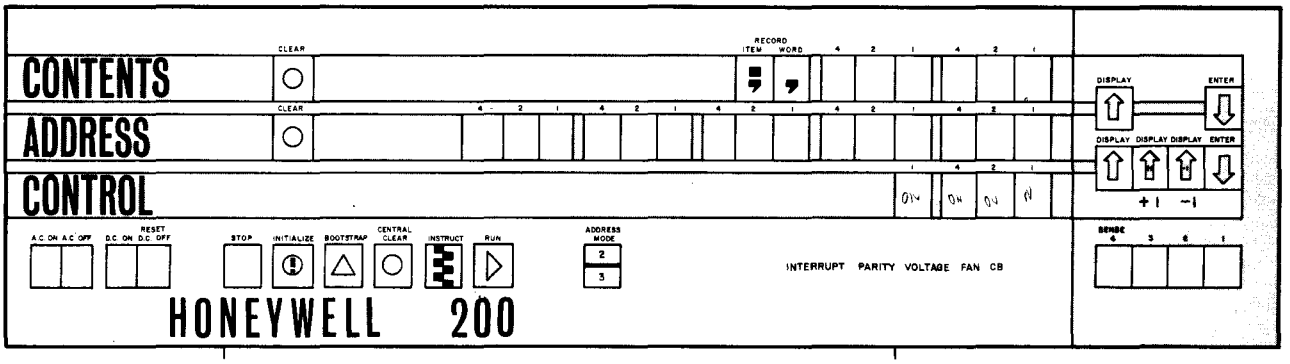
Counting the registers previously discussed, control memory has a total of 16 registers available.

53. The Interrupt register contains the address of the routine's first instruction for the external devices. This register's function is similar to that of the co-sequence register. Show the contents of the registers below after an interrupt signal is received.

	IAR	INTERRUPT REGISTER
	PROGRAM SEQUENCE ADDRESS	ROUTINE ADDRESS
INTERRUPT SIGNAL	ROUTINE ADDRESS	PROGRAM SEQUENCE ADDRESS

62. Use of the control panel will be reviewed before discussing substituting of the Unassigned register. Assume that CONTENTS of a series of memory locations are to be checked starting with the ADDRESS in the CONTROL memory IAR. Mark the appropriate buttons ON (Binary 1) to select the IAR (Octal 17).

AFTER 2nd CSM is the ad- shows the



71. Instead of using the letters IAR, it is simpler to designate the Instruction Address Register with the single letter I. Similarly, the letter and prime mark - I' - represent the alternate Instruction address register. This register is often called the CP - SEQUENCE register.

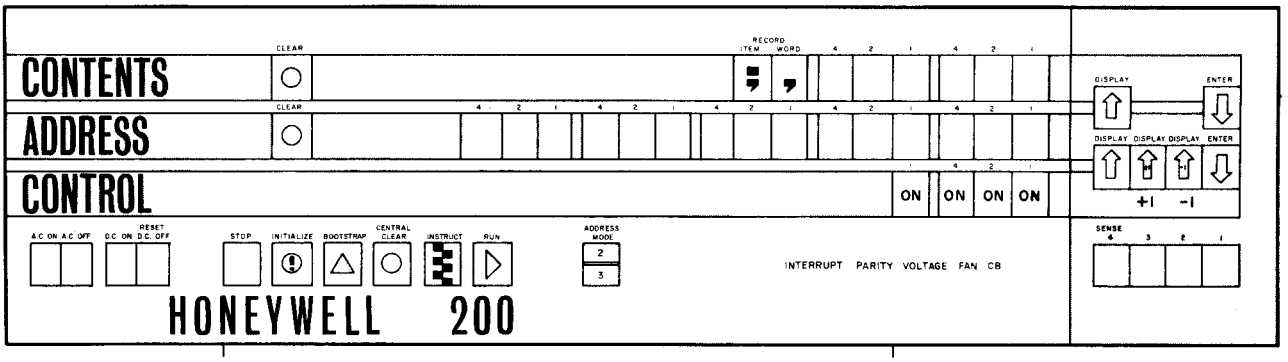
44.

16

53.

	IAR	INTERRUPT REGISTER
INTERRUPT SIGNAL	ROUTINE ADDRESS	PROGRAM SEQUENCE ADDRESS

62.



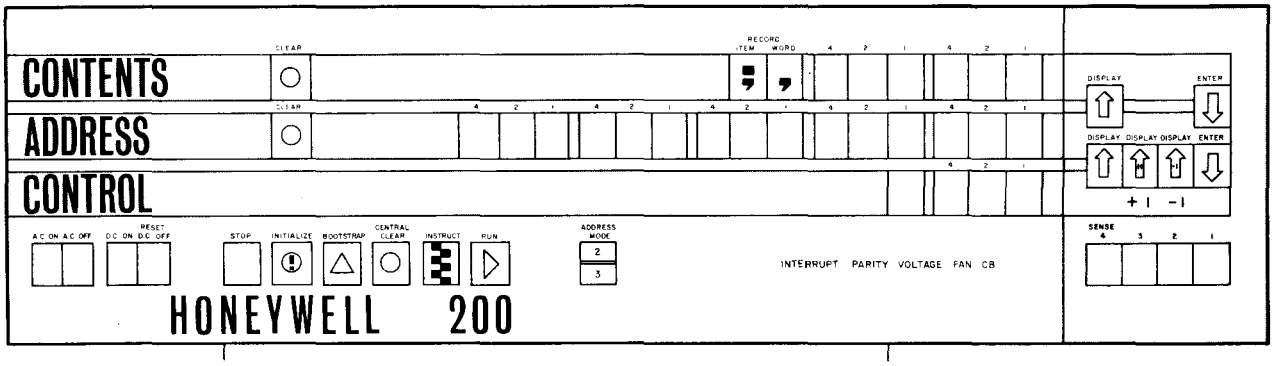
71.

INSTRUCTION ADDRESS REGISTER
CO-SEQUENCE

45. IAR'-sometimes called the "co-sequence" register - is an alternate instruction address register. The purpose of this register is to provide an ALTERNATE instruction address for a frequently required routine.

54. Register contents are exchanged when an interrupt signal is received. This allows the routine to be performed because its first instruction address is put into the INSTRUCTION ADDRESS register. The address of the program sequence is preserved by transferring it to the INTERRUPT register.

63. With the IAR selected by pressing CONTROL buttons for octal 17, the next step is to display the address it contains. To accomplish this, the lower DISPLAY button is depressed. Then, the binary number contained within the IAR appears as illuminated ADDRESS lights.



72. The letter A or the letters ac, are abbreviations for the A Address Register. Logically then, the letter B or the letters bc, are abbreviations for the B ADDRESS REGISTER.

45.

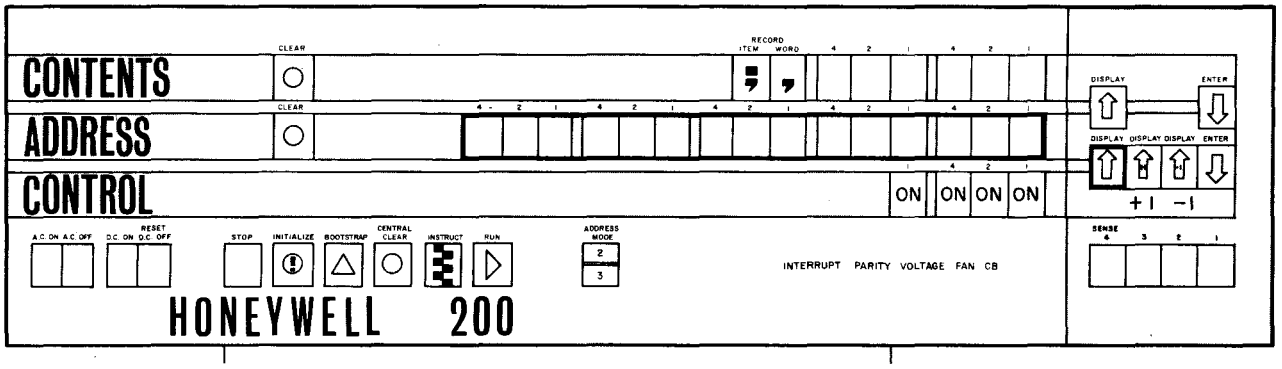
ALTERNATE

54.

INSTRUCTION ADDRESS
INTERRUPT

63.

DISPLAY
ADDRESS



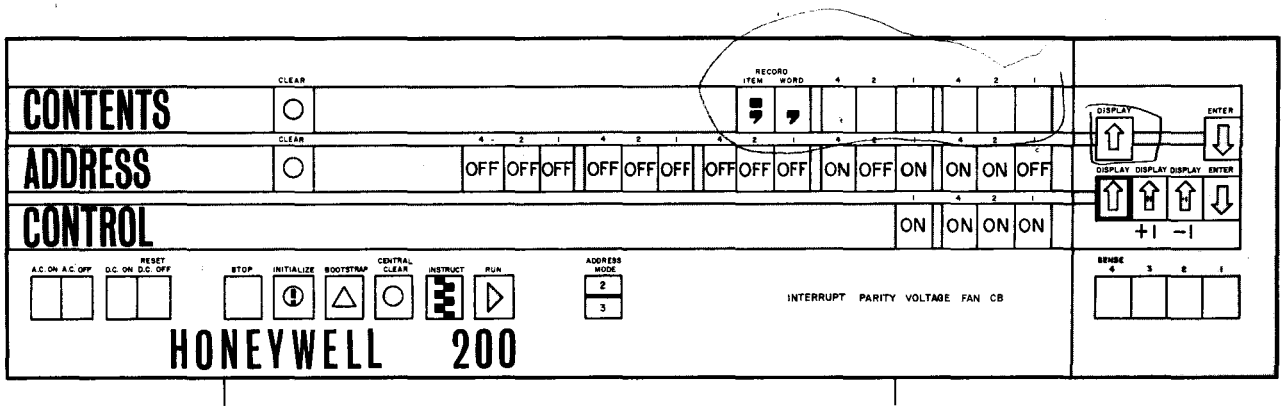
72.

B ADDRESS REGISTER

46. When an alternate instruction routine (co-sequence of instructions) is required, the IAR contents are exchanged with the address from the co-sequence register. This exchange of addresses between registers accomplishes two purposes. The address of the program sequence is preserved and the Co - SEQUENCE address is placed in the IAR.

55. Upon completion of the interrupt routine, the registers are exchanged with a RESUME NORMAL MODE (RNM) instruction. This returns the address to the IAR so that the program can RESUME NORMAL program sequence.

64. The ADDRESS contained in the IAR is shown below as binary 000 000 000 101 110, which is octal 56 or decimal 46.



Mark the button that must be pressed to display the contents of memory location #46.

73. I stands for the Instruction address register and I' for the alternate instruction address register called the Co - sequence register. The A Address Register may be represented by either A or ac. The letter B or bc designates the B ADDRESS REGISTER.

46.

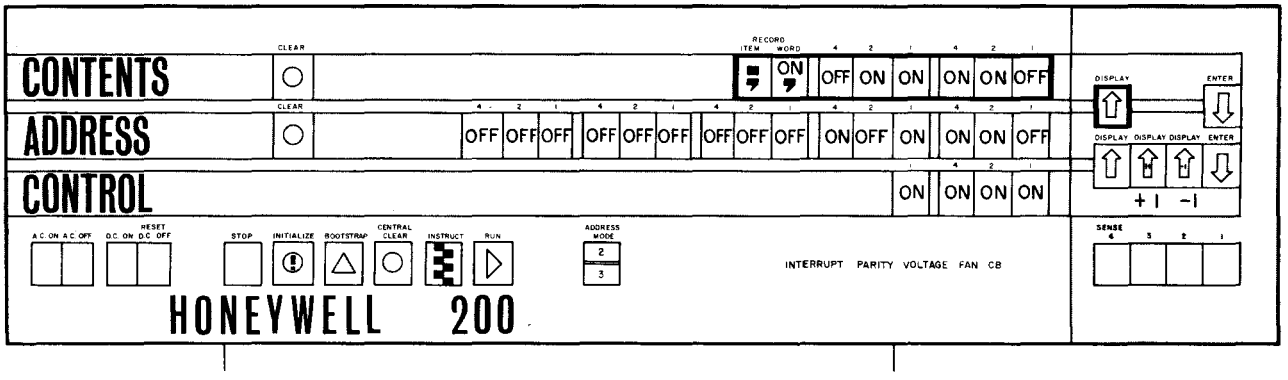
CO-SEQUENCE

55.

RESUME NORMAL

64.

OCTAL 56= DECIMAL 46
(CONTENTS IS THE OP. CODE A)



73.

INSTRUCTION ADDRESS REGISTER
CO-SEQUENCE REGISTER
A - ac
B ADDRESS REGISTER

47. With the address of the first instruction of the co-sequence in the IAR, retrieval and execution of the desired instruction routine occurs.

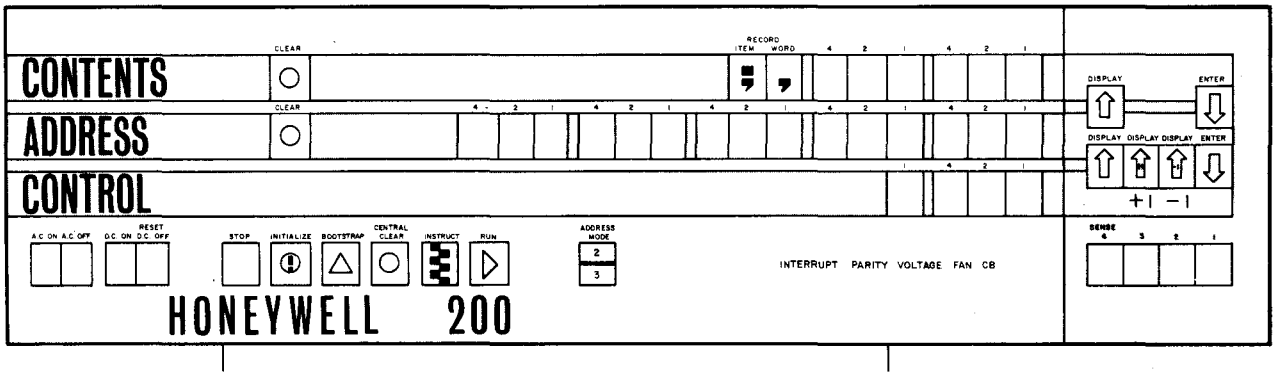
Of course, in order to retrieve this desired instruction ROUTINE, its address needs to have been in the CO-SEQUENCE register before the exchange with the INSTRUCTION ADDRESS register was accomplished.

---

56. As previously explained, the basic H-200 has three read/write channels. Two control memory registers (counters) are associated with each read/write channel. They are called the STARTING LOCATION counter and the CURRENT LOCATION counter.

---

65. To view the following memory location (#47), the DISPLAY +1 button is depressed causing the ADDRESS in the IAR to increment by one. With the address changed from #46 to #47, pressing the upper DISPLAY button will show CONTENTS of memory location #47. In order to view a preceding memory location, the DISPLAY −1 button is depressed to decrement the register.



---

74. The two registers in the control unit, but not part of control memory, are the OP. CODE and VARIANT registers. Obviously, the abbreviation V is for the VARIANT register. Designating the Function to be performed, the letter F stands for the OP CODE register.

47.

ROUTINE
CO-SEQUENCE
INSTRUCTION ADDRESS

56.

STARTING LOCATION
CURRENT LOCATION

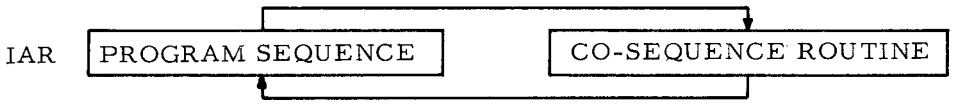
65.

DISPLAY -1

74.

VARIANT
OP CODE

48. A "Change Sequence Mode" (CSM) instruction initiates the exchange of IAR and IAR' addresses.



Thus, the address of the first co-sequence instruction directs the performance of the routine and address of the program sequence is preserved. At the completion of the routine, another CHANGE SEQUENCE MODE instruction re-exchanges registers. The address of the PROGRAM SEQUENCE is returned to the IAR.

57. The inclusion of an optional fourth read/write channel requires two more control memory registers. One of these registers serves as a STARTING LOCATION COUNTER the other as a CURRENT LOCATION COUNTER.

66. A programmer/operator may want to "step through" a portion of a program with DISPLAY + 1 or DISPLAY -1. However, this will increment or decrement a register. If the computer were started after viewing several memory locations, the register would no longer contain the first address. In a situation such as this, the octal 00 register may be substituted for another register because it is UNASSIGNED to a specific machine function.

75. The instruction format F/A/B/V means that the instruction contains an OP CODE, A ADDRESS, B ADDRESS, and a VARIANT. The registers involved are the I register for retrieval, then the F A B V registers for storage during interpretation.

48.

CHANGE SEQUENCE MODE
PROGRAM SEQUENCE
IAR (INSTRUCTION ADDRESS REGISTER)

57.

STARTING LOCATION COUNTER
CURRENT LOCATION COUNTER

66.

UNASSIGNED

This control memory register can be used by the programmer/operator to simulate any of the other fifteen registers. For example, an address from the IAR can be duplicated in the Unassigned register. Then, the programmer/operator can manipulate instructions with the control panel, through the Unassigned register, without actually changing IAR.

75.

OP. CODE
A ADDRESS
B ADDRESS
VARIANT
F, A, B, V.

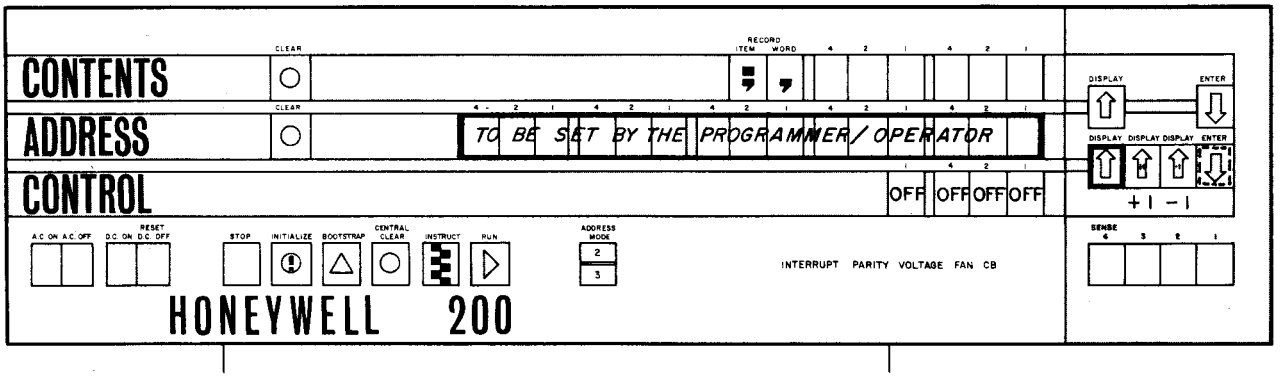
49. Name the registers below and write the name of the address each contains.

	IAR	IAR'
BEFORE FIRST CSM	PROGRAM SEQUENCE ADDRESS	CO-SEQUENCE ADDRESS
CSM EXECUTED	CO- SEQUENCE ADDRESS	PROGRAM SEQUENCE ADDRESS
DURING ROUTINE	CO - SEQUENCE ADDRESS	PRG SEQ ADDRESS
AFTER 2nd CSM	PRG SEQ ADDRESS	CO - SEQ ADDRESS

58. These counters keep track of where a read/write channel operation began and at which memory location it halted when the 2 microsecond time sharing cycle moved to the next read/write channel.

An H-200 with an optional read/write channel will use a total of 8 control memory registers as starting location and current location counters.

67. To select the 00 Unassigned register, all that is required is to set all four CONTROL buttons to zero (OFF) and then press the display button. To load the Unassigned register with the desired address, the correct ADDRESS buttons are depressed and the ENTER button is engaged.



76. Some abbreviations used in timing formulas are shown below. Complete the entries in the MEANING column.

ABBREVIATION	MEANING
N_i	The number of characters in the instruction.
N_a	The number of characters in the A-field.
N_b	" " " " " " B-field.
Nw	The number of characters in the smaller field.
NXT	The address of NEXT sequential instruction.

49.

	IAR	IAR'
CSM EXECUTED	CO-SEQUENCE ADDRESS	PROGRAM SEQUENCE ADDRESS
DURING ROUTINE	CO-SEQUENCE ADDRESS	PROGRAM SEQUENCE ADDRESS
AFTER 2nd CSM	PROGRAM SEQUENCE ADDRESS	CO-SEQUENCE ADDRESS

58.

EIGHT

67.

CONTROL
ADDRESS

76.

N_b - THE NUMBER OF CHARACTERS IN THE B FIELD.
NXT-NEXT

50. The co-sequence routine's first instruction address enters the IAR to start retrieval. While in the IAR, this address is incremented as each instruction character is retrieved. Therefore, at the completion of the routine, the IAR no longer contains the address of the first co-sequence instruction. Regeneration of the first co-sequence address is accomplished with a "Branch" instruction at the end of the routine.

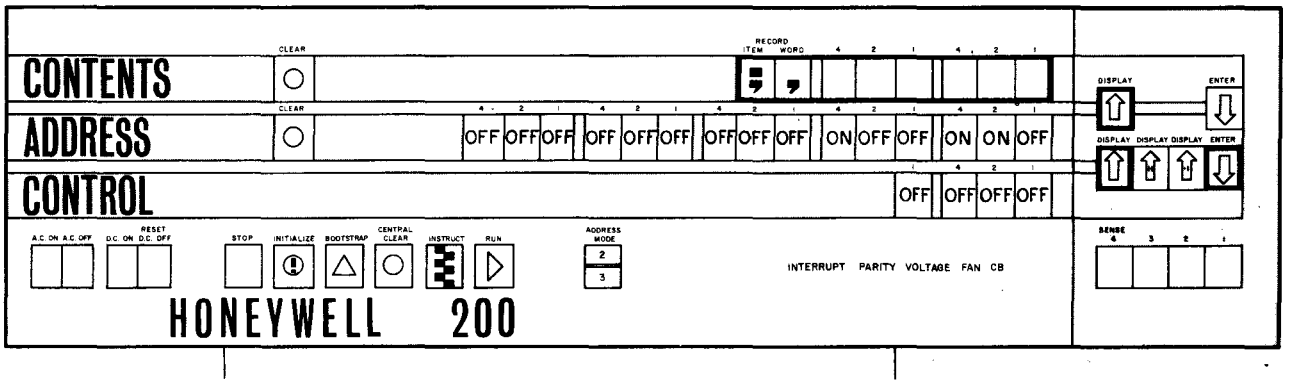
Example: Assume that the co-sequence routine starts at address #300.

	IAR	IAR'
The programmer writes the first CSM instruction to exchange registers	2547	300
This places address #300 into the IAR.	300	2547

Continue to the back of this frame.

59. Two registers are available for internal functions of control memory. For certain instructions, control memory can store a register's contents and transfer another address into the emptied register. Because control memory uses these two registers while it is accomplishing work, they are know as WORK registers.

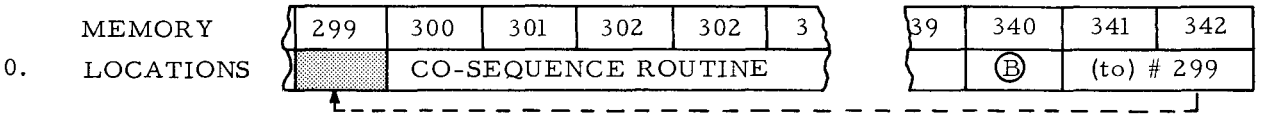
68. After engaging the ENTER button, the number set up with the ADDRESS lights is the address contained in the Unassigned register. Pressing the upper DISPLAY button shows the CONTENTS of the memory location specified by the ADDRESS now in the Unassigned register.



77. The remaining timing formula abbreviations are shown below. Complete the entries in the MEANING column.

ABBREVIATION	MEANING
JI	Address of the next instruction if a branch occurs.
A_p	Previous A Address register setting.
B_p	✓ B ✓ ✓ ✓
A	A Address
B	B ✓

50.



Co-sequence routine retrieval commences at address #300. IAR increments as each character is retrieved. The BRANCH instruction puts address #299 into the IAR. Memory location #299 precedes the first co-sequence address and contains a CSM. When this second CSM instruction is retrieved to re-exchange registers, the IAR will increment to #300. Therefore, when the registers are re-exchange, IAR' will contain the proper address.

299		IAR INCREMENTS TO	IAR'
CSM	CSM RETRIEVED	300	2547
	EXECUTED	2547	300

59.

WORK

68.

CONTENTS

77.

B_P-PREVIOUS B ADDRESS REGISTER SETTING.
B ADDRESS

51. This frame reviews CSM operation in six steps. General names are used for addresses instead of specific numbers.

		IAR	IAR'
STEP 1.	PROGRAMMER'S CSM EXECUTED	PROGRAM SEQUENCE ADDRESS	CO-SEQUENCE ADDRESS
STEP 2.	CO-SEQUENCE RETRIEVAL STARTS	CO-SEQUENCE ADDRESS	(PRESERVED) PROGRAM SEQUENCE ADDRESS
STEP 3.	INCREMENTATION DURING RETRIEVAL	INCREMENTED ADDRESS	PROGRAM SEQUENCE ADDRESS
STEP 4.	BRANCH TO ADDRESS PRECEDING 1st CO-SEQUENCE ADDRESS	CO-SEQUENCE ADDRESS - 1	PROGRAM SEQUENCE ADDRESS
	Show register contents during steps 5 and 6.		
STEP 5.	RETRIEVE 2nd CSM INCREMENTING IAR	CO-SEQUENCE ADDRESS	PROGRAM SEQUENCE ADDRESS
STEP 6.	EXECUTE 2nd CSM	PROGRAM SEQ ADDRESS	CO-SEQUENCE ADDRESS

60. Control memory automatically uses its two work registers to preserve addresses in much the same manner that a person "jots down" something to be remembered. In .5 microseconds, control memory can empty a register to receive another address, while preserving the original address by transferring it to a WORK REGISTER.

69. Once the Unassigned register is selected and supplied with the desired starting address, DISPLAY +1 or DISPLAY -1 may be used to "step through" the program. This will increment or decrement the UNASSIGNED register but will not change any of the other registers. Effectively, this register can substitute for the other registers.

78. Give the meaning of the following timing formula abbreviations.

N_i No. characters in the instruction JI address of next instruction if a branch

N_a ✓ ✓ ✓ ✓ A FIELD A_p Previous A address register setting

N_b ✓ ✓ ✓ ✓ B FIELD B_p ✓ B ✓

Nw ✓ ✓ ✓ smaller ✓ (A or B) A A address

NXT address of NEXT SEQUENTIAL INSTRUCTION B B address

51.

	IAR	IAR'
STEP 5.	CO-SEQUENCE ADDRESS	
STEP 6.	PROGRAM SEQUENCE ADDRESS	CO-SEQUENCE ADDRESS

60.

WORK REGISTER

69.

UNASSIGNED

78.

ABBREVIATION	MEANING
N_i	The number of characters in the instruction
N_a	The number of characters in the A-field
N_w	The number of characters in the A- or B-field, whichever is smaller
N_b	The number of characters in the B-field
NXT	Address of the next sequential instruction
JI	Address of next instruction if a branch occurs
A_p	The previous setting of the A-address register
B_p	The previous setting of the B-address register
A	A address
B	B address

52. External devices such as communication equipment, send the central processor a demand signal to indicate when a specialized routine needs to be performed. Interruption of the program for a routine involves a register similar to the co-sequence register. Named for its response to an interrupt, this control memory register is called the INTERRUPT register.

---

61. The sixteenth control memory register has an octal address of 00 and is "unassigned" to any specific machine function.

Consequently, the 00 register is called the UNASSIGNED register. It is available for use by a programmer or operator through buttons on the control panel.

---

70. Name and briefly state purposes of each of the seven additional control memory registers.

IAR - To supply a co-sequence routine address for CSM

INTERRUPT - To ~ ~N EXTERNAL DEVICE ~~SIGNAL~~ ROUTINE IN REPLY TO AN INTERRUPT SIGNAL

UNASSIGNED - To substitute for another register when incrementing or decrementing with DISPLAY +1 or -1

RWC' - SLC

RWC - CLC

2 WORK Registers - available for automatic control memory preservation of register address

---

79. Determine the number of microseconds used for the H-200 to execute an Add Instruction.

Formula: $2(N_i + 2 + N_w + 2\,N_b)$ Format F/A/B (5)

Operands: Five characters each. (Two Characters per address.)

$2(5+2+5+10)$

= 44 microseconds

52.

INTERRUPT

(Return to page 111, frame 53.)

61.

UNASSIGNED

This control memory register can be used by the programmer/operator to simulate any of the other fifteen registers. For example, an address from the IAR can be duplicated in the Unassigned register. Then, the programmer/operator can manipulate instructions with the control panel, through the Unassigned register, without actually changing IAR.

(Return to page 111, frame 62.)

70.

IAR' - To Supply a co-sequence routine address for CSM.

INTERRUPT - To supply an external device routine in response to an interrupt.

RWC #1' } Starting and current location counters for the optional fourth read/write channel.
RWC #1' }

WORK REGISTER Available for automatic control memory preservation of register
WORK REGISTER addresses.

UNASSIGNED REGISTER - To substitute for other registers when incrementing or decrementing with control panel DISPLAY + or -1.

(Equivalent answers are acceptable.)

(Return to page 111, frame 71.)

79.

44 microseconds.

F/A/B=5 characters. N_w=5. N_b=5

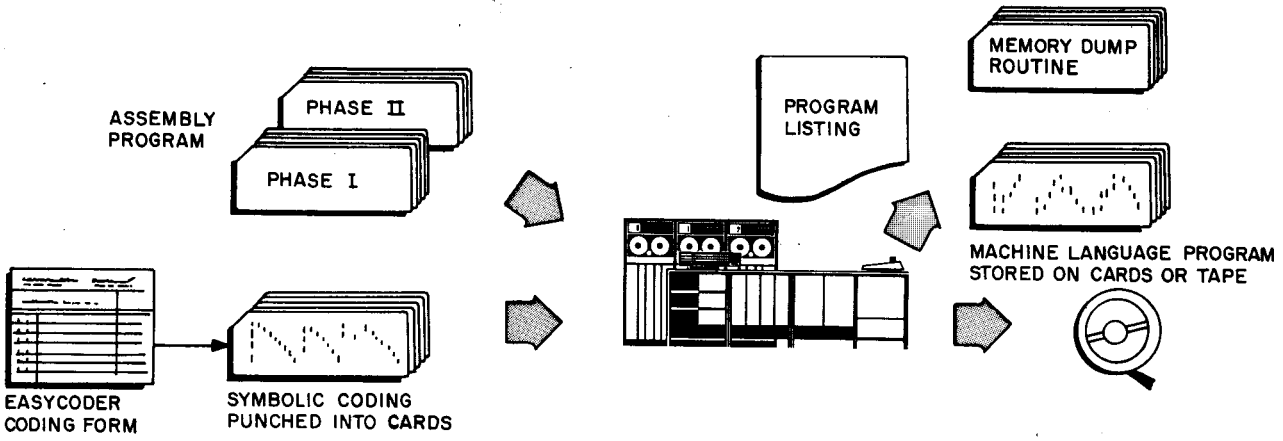
$2 (N_i + 2 + N_w + 2N_b) = 2(5 + 2 + 5 + 10) = 44$

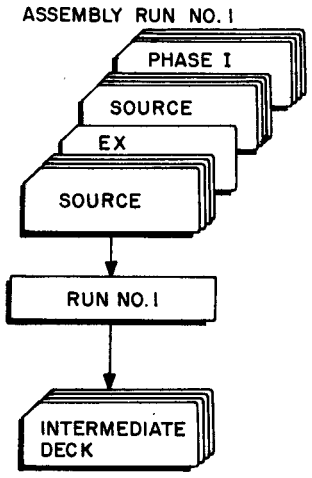
(Continue to page 129.)

LESSON VI
EASYCODER PROGRAMMING

EASYCODER ASSEMBLY

Basic Easycoder's assembly system uses two Honeywell - supplied card decks and one card deck punched according to a programmer's entries on coding sheets. Assembly of these card decks in two "runs" (Phase I and II) produces: a printed listing of the source program, a card deck for a "memory dump routine" - complete listing of memory contents -, and the object program on punched cards or tape.

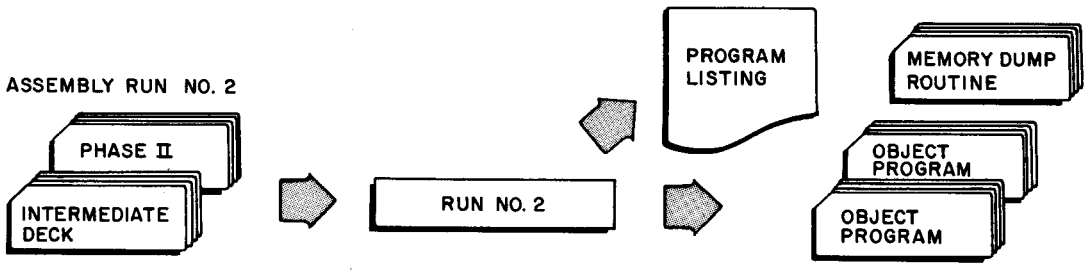




In the first assembly run, the Phase I and Source program decks are fed into the machine for partial conversion of the source program into an object program. The following steps are accomplished during this run:

1. Mnemonic Op. Codes are translated.
2. A tag table is generated.
3. Sizes of operand fields are defined.
4. Assembly control statements are processed.
5. Errors are detected and flagged.
6. An intermediate deck is punched.

Notice that the source program deck is segmented by an EX card. This permits some processing before the remainder of the source program is entered.



The following operations are accomplished during the second assembly run: Addresses of operands and constants are assigned from the tag table generated by Phase I. The memory dump routine - a separate self-loading deck - is punched. The object program deck and its loading routine are punched. A printed listing of the source to object program is produced.

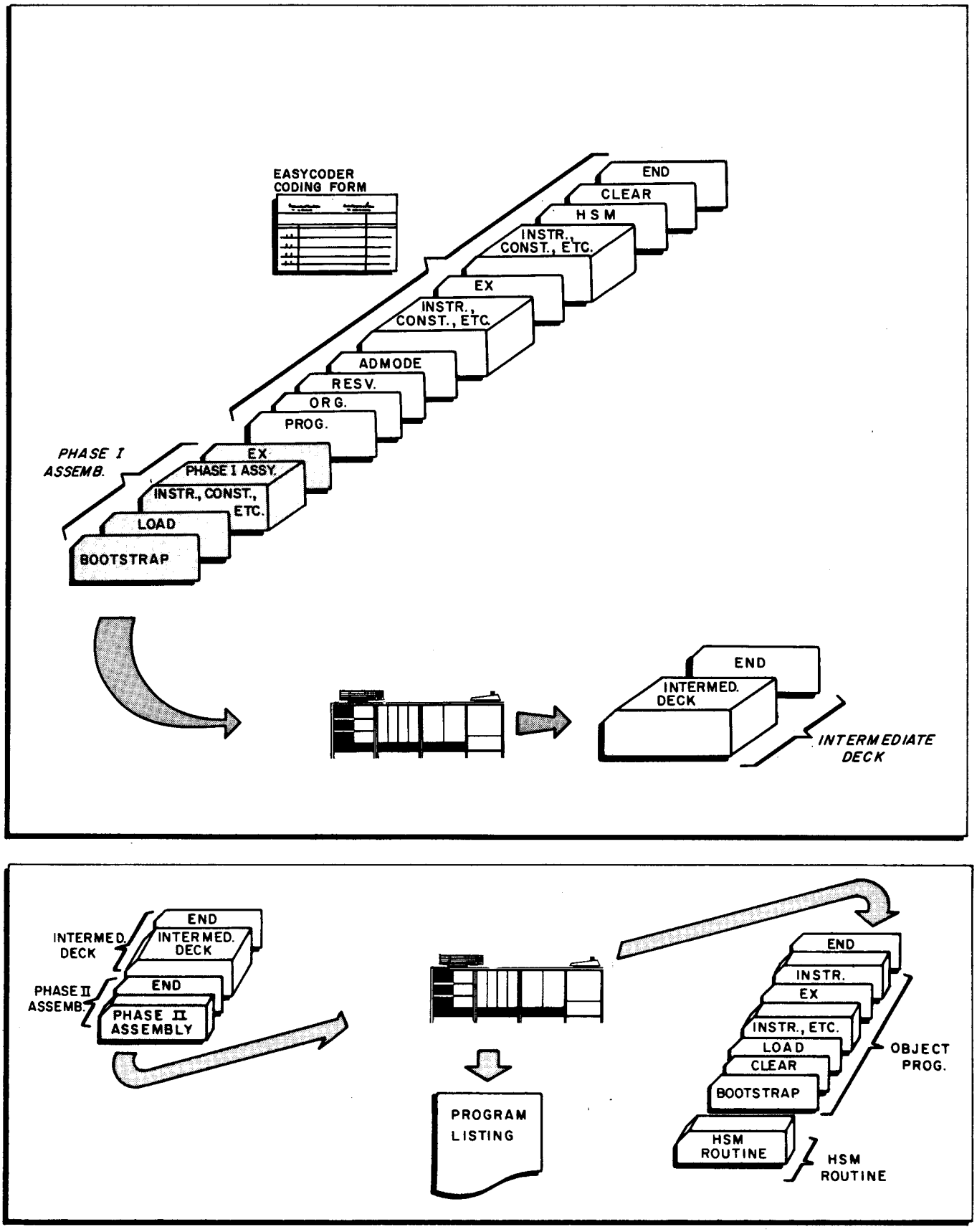
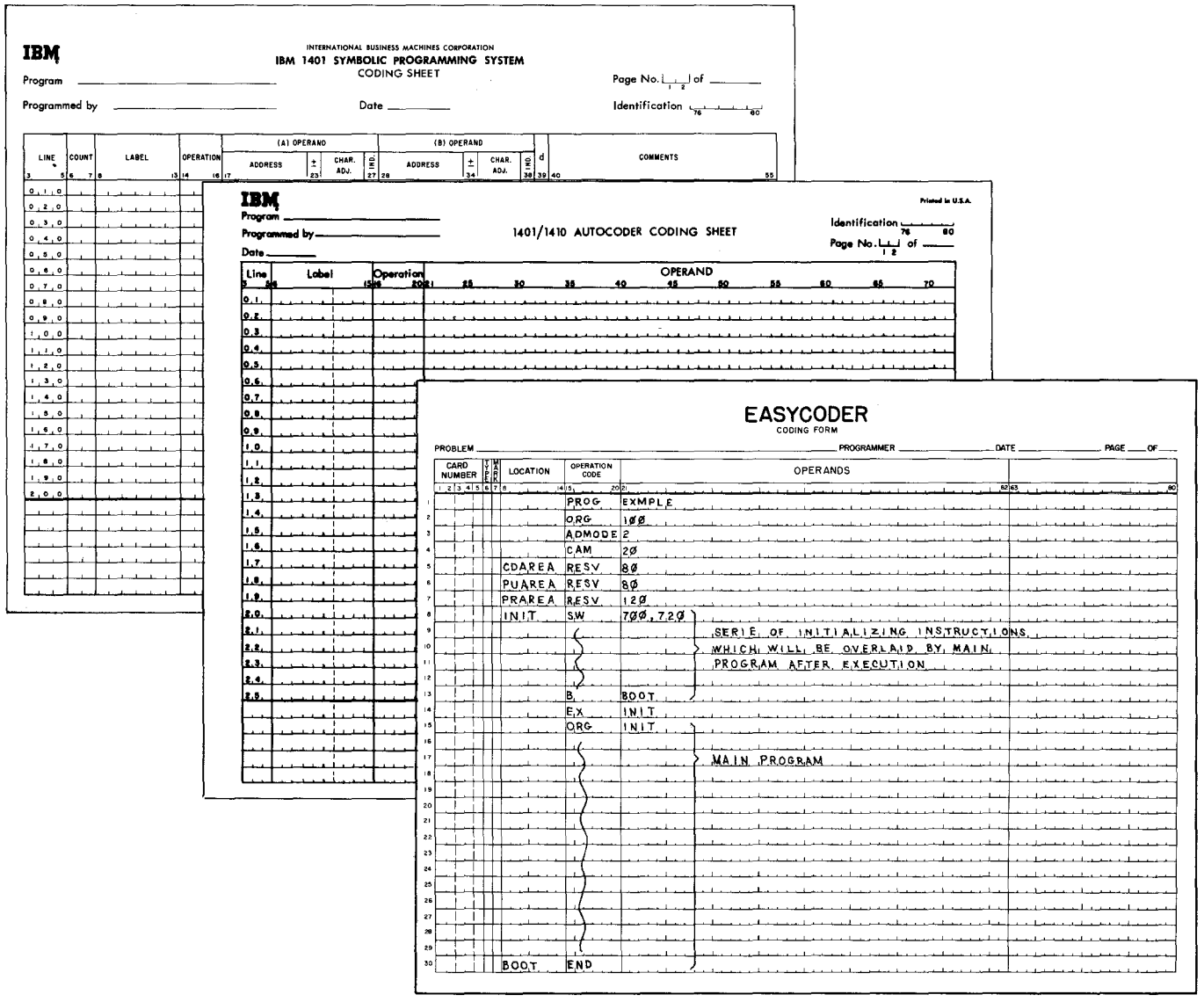


Figure 14. Easycoder Assembly



Coding Form Entries

Col. 1-2: Contain page number.
Col. 3-4: Contain line number.
Col. 5: Contains a number if statement is to be inserted between two lines.
Col. 6: Contains an asterisk (*) if strictly Remarks statement.
Col. 7: Contains an L if an item mark is desired in the leftmost character position of the statement. Contains an R if an item mark is desired in the rightmost character position.
Col. 8-14: If a tag, it must be no more than six characters long. First character of tag must be alphabetic.
Col. 15-20: Mnemonic op code must begin in col. 15. (Octal machine language-columns 19 and 20.)
Col. 21-62: Operands must begin in col. 21. A comma must follow all operands except the last operand in the line. Comments and remarks must be separated from operands by blank space.
Col. 63-80: Comments and remarks, not included in the object program coding.

Figure 15. Coding Forms

EASYCODER CODING FORM DIFFERENCES

(Answer the following questions by reference to Figure 15.)

1. A single line on the Easycoder coding form contains a total of 80 columns. Therefore, the complete contents of a punched card can be recorded on one line.

2. Logically, the first Easycoder column on a line is column number 1, and the last column on a line is number 80.

   The first column of an SPS or Autocoder line is number 3. The last column of an SPS line is number 55, because further card entries cause incorrect processing. Similarly, the last column in an Autocoder line is number 72, because card columns 73 - 75 are required by the 1401 processor.

3. Punched card columns 1-5 are used for the same purposes in the 1401 and H-200 systems. However, greater flexibility is afforded by an Easycoder coding form than with the other forms. Line numbers are not pre-printed (but numbers are supplied for reference) on an Easycoder form. Deletion of an SPS or Autocoder line by scratching it out causes a line number to be "missing" in the final deck of cards.

   Insertions may conveniently be written on <u>any</u> of the 30 Easycoder coding lines by noticing the page and line number that the desired insertion is to follow. This page number is then entered in columns 1 and 2, followed by the line number in columns 3 and 4, and the insertion number in column 5.

   As you recall, lines 26-30 on an SPS or Autocoder form were to be used for insertions.

4. Additional convenience is provided by columns 6 and 7 of the Easycoder form. Column 6 contains an asterisk if strictly a REMARKS statement is to be entered. Column 7 places the unique H-200 punctuation - the ITEM mark - in either the high or low order position. (<u>Extended</u> use of column 7 is available in EXTENDED Easycoder.)

   NOTE: The "COUNT" columns required in SPS are not needed by Easycoder.

5. In apparent purpose, the LOCATION columns on the Easycoder form are similar to the LABEL columns of SPS or Autocoder forms. Easycoder refinement in this area will be pointed out in following frames.

6. The size of an SPS operand field is restricted and usually is referred to as "fixed form". Operand sizes are not specified, hence they are "free form" on the AUTO CODER coding sheet and EASYCODER coding form.

1. Entries in columns 1 - 5 may be accomplished in several fashions, depending upon programmer preference or established key punch procedure. For purposes of this book, indicate page 1, line 1, zero insertion, on the coding form segment below:

EASYCODER
CODING FORM

PROBLEM PAYROLL PROCEDURE PROGRAMMER J.E.H. DATE 15/6/64 PAGE 1 OF 4

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62 63 ... 80
0 1 0 1 0					

15. An Op. Code is always found as the leftmost character of an instruction. When a tag begins in the leftmost Location column (# 8) and refers to an instruction, the ADDRESS assigned by assembly is that of the memory location containing the OP CODE, which is the LEFTMOST character of the instruction.

29. The H-200 has larger memory than the 1401; consequently larger decimal numbers may be used as addresses. However, your familiarity with what were called actual addresses will let you to use those which Honeywell refers to as ABSOLUTE addresses without further discussion. Similarly, what you already know about symbolics is consistent with Honeywell SYMBOLIC addresses.

43. If direct absolute addressing had been used instead of indexed or indirect addressing, the coding form would have looked like this:

OPERATION CODE	
15 ... 20	21 ...
A	3120,415

Refer to frames 41 and 42, then show how the coding form would be completed to specify:

Indirect Addressing of the A Operand

OPERATION CODE	
15 ... 20	21 ...
A	(1027),415

Indexed Addressing of the A Operand using index register X2.

OPERATION CODE	
15 ... 20	21 ...
A	1027+X2,415

57. The PROG assembly control statement was discussed at the start of this lesson. Briefly explain its purpose and indicate when it is written on the coding form. First entry in the program, takes the operand with 6 characters as the program name.

1. NOTE: Programmer's should complete at least the first five columns on the first line of each coding form.

EASYCODER
CODING FORM

PROBLEM PAYROLL PROCEDURE PROGRAMMER J. E. H. DATE 15/6/64 PAGE 1 OF 4

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
Ø1Ø1Ø						

15. #8
ADDRESS
OP. CODE
LEFTMOST

29. ABSOLUTE
SYMBOLIC

43.

INDIRECT

MARK	LOCATION	OPERATION CODE	
7 8	14	15 20	21
		A	(1Ø27),415

INDEXED

MARK	LOCATION	OPERATION CODE	
7 8	14	15 20	21
		A	1Ø27+X2,415

57. PROG IS THE FIRST ENTRY IN THE PROGRAM. PROG CAUSES ASSEMBLY TO TAKE UP TO SIX CHARACTERS WRITTEN IN THE OPERANDS FIELD AS THE PROGRAM NAME.

(Or equivalent answer)

2. If prior agreement has been made with the key punch operator for duplication of the first entries in columns 1, 2, and 5, a programmer may complete only the line number columns for subsequent entries. The programmer's entries in columns 3 and 4 identify the LINE NO. of the coding form.

Of course, a programmer would be correct if he decided to complete all five columns for each of the 30 coding form lines.

---

16. Constants or characters in reserved areas are usually retrieved from the rightmost to the leftmost character. Appropriately then, a tag which begins in column 8 and refers to a constant or reserved area, will have an assembly assigned address of the RIGHTMOST character.

---

30. What was called "address adjustment" in your previous system is termed "relative addressing" in Easycoder. These examples:

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
Ø2Ø1Ø				A	DATA+SIZE-1,12+ITEM-3	

show that RELATIVE addressing may be used with either SYMBOLIC or ABSOLUTE addresses.

---

44. Indexing and indirect addressing require three bits for identification by the address type indicators. In two character addressing mode, all 12 bits are required to express addresses up to # 4095. ($111111111111_2 = 4095_{10}$) Since all twelve bits are used in two character addressing, no bits are available for address type indicators. Consequently, when in two character addressing mode, neither INDEXING nor INDIRECT addressing are available.

---

58. In addition to the name written in the operands field, a PROG card will contain certain other information to direct the assembly process. This additional punched information replaces the requirement for the "control" card that was needed with a 1401.

You are already familiar with the mnemonic written to originate addresses. It causes assembly to assign subsequent addresses starting at other than location 0, and is the mnemonic, ORG

2.

LINE NUMBERS

16.

RIGHTMOST

30.

RELATIVE
SYMBOLIC
ABSOLUTE

44.

INDEXING
INDIRECT

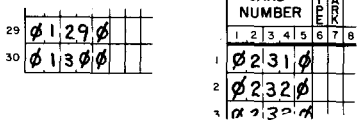
NOTE: Indexing and indirect addressing require an address mode greater than two characters. The instruction to specify the ADMODE as two, three, or four characters and the instruction to Change Addressing Mode - CAM - are explained later in this lesson.

58.

ORG

3. The manner of line numbering (columns 3 and 4) is left to the programmer's discretion. Lines may be numbered sequentially and continue from one sheet to the next.

Example:



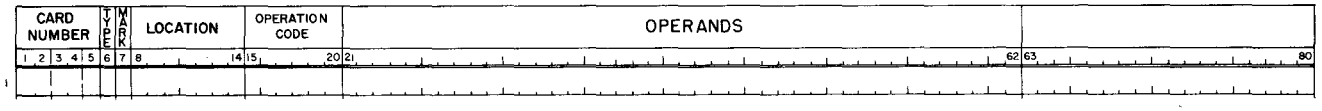
Alternatively, line numbers may begin again on each page. In this case, line numbers could go from number 1 to number 30 on each page.

17. A tag beginning in column 8 for an instruction is assigned the address of the OP CODE which is always the LEFTMOST character.

A tag beginning in column 8 for a constant or reserved area is assigned the address of the RIGHTMOST CHARACTER.

31. An * may be used in the operands field to signify self reference. Assembly interprets the * as the memory location of the Op. Code for the instruction in which the * appears. Since the * is a symbol, its address is considered to be a SYMBOLIC address. Notice that where an * signified the rightmost memory location in your 1401 programming, the H-200 uses an * to refer to the LEFTMOST most memory location of the instruction.

45. To this point, seventy-nine of the eighty coding form columns have been mentioned in various examples. What is the name and number of the column that has not been discussed so far? INFO Column # 7



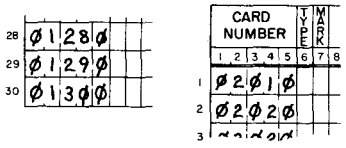
59. ORG statements may be written at any point in the program causing assembly to assign subsequent addresses starting with the location specified in the operands field.

Assembly will start assigning addresses with location O unless an ØRG statement is written immediately following the PROG statement.

3.

1 - 30

EXAMPLE:



17.

OP. CODE
LEFTMOST
RIGHTMOST CHARACTER

31.

SYMBOLIC
LEFT

45.

MARK
COLUMN # 7

59.

ORG
PROG

4. An assembly control mnemonic Op. Code is always the first coding form entry. This Op. Code causes assembly to name the PROGram, using up to six characters from the operands field. Abbreviate the problem title below and make the proper entries on the coding form.

EASYCODER
CODING FORM

PROBLEM PAYROLL PROCEDURE PROGRAMMER J.E.H. DATE 15/6/64 PAGE 1 OF 4

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62
ØlØlØ				PRØG	PAYRØL

18. Easycoder provides a versatility of tag address assignment that was not available in your previous system. The left or rightmost address assignments discussed in the previous frame may be reversed simply by starting a tag in column 9.

A tag beginning in column 9 for an instruction is assigned the address of the RIGHTMOST CHARACTER.

A tag beginning in column 9 for a constant or reserved area is assigned the address of the LEFTMOST CHARACTER.

32. The utilization of an * address and relative addressing is illustrated in the MCW instruction below:

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
	1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62
1					MCW	*+9,WORK
2					S	TAX,PAY

The function of MCW instructions is to move the field specified by the A address to that specified by the B address. The notation *+9 refers to the rightmost character of the instruction stored immediately to the right of the MCW instruction (assuming that two-character address assembly has been specified). The instruction following the MCW instruction will be moved to the field tagged WORK when the MCW instruction is executed.

46. This Mark Column (#7) brings up a point that should be noted. To this point in the lesson, no distinction has been made between EASYCODER and EXTENDED EASYCODER. The subjects discussed were applicable to both systems. Certain entries in column #7 apply equally to both systems, but Extended Easycoder - as its name implies - makes additional or EXTENDED use of the Mark Column.

60. Tags may be used with ORG statements but a tag must begin in Column 8 of the location field if it is to be used as a symbolic address. A tag should be defined prior to being used as a symbolic address with an ORG statement. This may be accomplished by writing the tag beginning in Column 8 of the location columns either along with the ORG instruction or as a preceding entry.

At which address will assembly start assigning addresses if an ORG statement is not written ? 0

4. NOTE: Any name of up to six characters may be used in the operands field.

EASYCODER
CODING FORM

PROBLEM PAYROLL PROCEDURE PROGRAMMER J.E.H. DATE 15/6/64 PAGE 1 OF 4

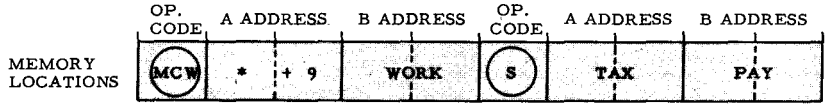
CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62
Ø1Ø1Ø				PROG	PAYROL

18.

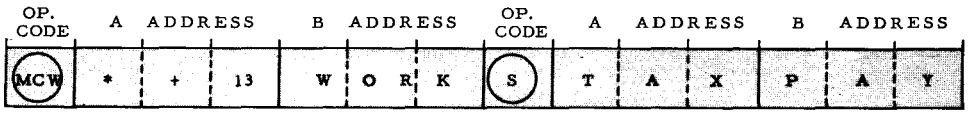
RIGHTMOST CHARACTER
LEFTMOST CHARACTER

32. The examples below indicate that a programmer must take the addressing mode into consideration when writing an * address.

TWO CHARACTER ADDRESSING



THREE CHARACTER ADDRESSING



Four Character Addressing to access memory up to 65,000 memory locations will be discussed later in this lesson.

46.

EXTENDED

60.

Assembly will start assigning addresses at memory location Ø unless directed otherwise by a ORG statement.

5. The first mnemonic Op. Code will always be the assembly control statement, PROG. This causes assembly to name the program. Up to SIX characters can be entered in the OPERANDS field to express the program name.

---

19. The address assignments for tags are summarized below:
A tag beginning in Col. 8 of an instruction refers to the LEFTMOST CHARACTER (OP CODE)
A tag beginning in Col. 9 of an instruction refers to the RIGHTMOST ✓.
A tag beginning in Col. 8 of a constant or reserved area refers to the RIGHTMOST ✓.
A tag beginning in Col. 9 of a constant or reserved area refers to the LEFTMOST ✓.

---

33. Make the following comparisons between an H-200* and a 1401*:

1. The H-200* references the LEFT MOST memory location of the instruction in which it appears.

2. A programmer needs to take into account the number of characters being used to express an ADDRESS for the H-200.

---

47. In this and subsequent frames, subjects applicable to both Easycoder and Extended Easycoder will be presented. Any topics pertaining exclusively to Extended Easycoder will be indentified with the titles, "EXTENDED EASYCODER".

One of the purposes of Column 7 is to provide a convenient method of setting ITEM marks without writing a SET ITEM MARK instruction. Obviously, if this column remains blank, NO ITEM MARK is set.

---

61. The mnemonic MORG (for Modular Origin) is written when it is desired to have assembly start assigning addresses at the first multiple of an address written in the operands field.

For example, if the last address assigned was location number 100, the MORG statement below would cause assembly to start assigning subsequent addresses at location number 128.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
		MORG	64

5.

PROG
SIX
OPERANDS

19.

OP. CODE
RIGHTMOST CHARACTER
RIGHTMOST CHARACTER
LEFTMOST CHARACTER

33.

LEFT MOST
ADDRESS

47.

NO ITEM MARK

61.

128

NOTE: The operands field entry for a MORG statement must be a power of 2. Examples: 2, 4, 8, 16, 32, 64, ... etc.

6. When more information needs to be conveyed than is afforded by the six character name, subsequent lines may provide remarks (comments). An * is used to specify this type of entry and as illustrated below, the * is placed in the TYPE column.

EASYCODER
CODING FORM

PROBLEM PAYROLL PROCEDURE PROGRAMMER J.E.H. DATE 15/6/64 PAGE 1 OF 4

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1 2 3 4 5	6	7	8 14	15 20	21 62 63 80
1 ØIØIØ				PROG	PAYROL
2 ØIØ2Ø	*				PAYROLL EXAMPLE PREPARED FOR EDUCATION RESEARCH TEXTBOOK 200

20. As previously stated, tags can contain up to six characters, but the first character must be alphabetic.

If desired, absolute decimal addresses may be written in Location columns in place of tags. Briefly state in your own words how assembly tells the difference between a tag or an absolute address in the Location column. TAG - FIRST CHARACTER ~~IS~~ ALPHABETIC.
ABSOLUTE - FIRST CHARACTER NUMERIC

34. Your previous system was limited to the use of only three index registers denoted as +X1, +X2, and +X3. The H-200 makes six index registers available and they are specified on the coding form in the manner to which you are accustomed. Write the designations for each of the H-200 index registers: +X1, +X2, +X3, +X4, +X5, +X6.

48. If an L (for Left) is written in column seven, an ITEM MARK will be placed in the leftmost memory location of the field (or instruction).

An R in column 7 is the converse of the above. Briefly state the effect of an R in column 7.
An item mark is placed in the rightmost memory location of the field (or instruction).

62. If several programmers are each writing portions of a program, different symbolic tags may inadvertantly be used for the same program element. Easycoder contains an assembly control statement to correct this situation. It is named for the operation of making different tags equal to one address, hence it is called an EQUAL statement.

6.

TYPE

---

20. TAGS BEGIN WITH AN ALPHABETIC CHARACTER. ABSOLUTE ADDRESSES BEGIN WITH A DIGIT.

34.

+X1
+X2
+X3
+X4
+X5
+X6

48.

AN R IN COLUMN 7 PLACES AN ITEM MARK IN THE RIGHTMOST MEMORY LOCATION OF THE FIELD OR INSTRUCTION.

62.

EQUAL
(Mnemonic: EQU)

7. A remarks line may be written at any point in a program. Name clarification is simply one example of the use of a REMARKS line. When this type of line is required, an * is written in the TYPE column (column # 6).

21. In the examples below, indicate whether the tags refer to the right or leftmost memory location.

	LOCATION	OPERATION CODE	
	8 ... 14	15 ... 20	21 ...
1.	EXEMPT	DCW	@VETTAX@
2.	NOV,65	DC	@11TH, MO@
3.	1652	DCW	@PART A@
4.	GROSS	A	TAX,NET
5.	2122	A	100,200
6.	STOKOPS	S	STOK,NET

1. L most memory location.
2. R most memory location.
3. L most memory location.
4. L most memory location.
5. L most memory location.
6. R most memory location.

35. Index designators must all begin with a plus sign. However, it is not proper to introduce an operand with either a + or - sign. When the contents of an index register are used in the entire address, it should be preceded by a 0 on the coding form. In the ADD instruction below, write the A address as the address stored in index register six and the B address as a location tagged WORK.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	... 14	15 ... 20	21 ... 62
		A	0,X6,WORK

49. Item Marks set through the use of column 7 conveniently replace the necessity of writing a SET ITEM instruction. This convenience may be utilized whenever Item Marks are desired in either the Rightmost or Leftmost memory location of an entry.

The SET ITEM instruction (to be discussed later) is still required if the punctuation is to be placed in locations other than the extremes.

63. EQU may be used in various situations other than the single example previously cited. However, the basic purpose of EQU is to cause a symbolic tag to be EQUAL to the ADDRESS written in the OPERANDS field.

LOCATION	OPERATION CODE	
8 ... 14	15 ... 20	21 ...
WTHL	EQU	2048

7.

REMARKS

*

TYPE

6

21.

LEFT

RIGHT

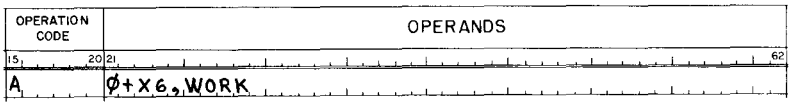
LEFT

LEFT

LEFT

RIGHT

35.



49.

LEFTMOST

RIGHTMOST

63.

EQUAL

ADDRESS

OPERANDS

8. It should be noted that the operands field begins with column # 21 and ends with column # 62.

The portion of a remark that continues into columns # 63 to # 68 will not appear in the assembled object program printed listing. Write the following remark as it will appear in an assembled object program printed listing.

EASYCODER
CODING FORM

PROBLEM PAYROLL PROCEDURE PROGRAMMER J.E.H. DATE 15/6/64 PAGE 1 OF 4

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1 Ø1Ø1Ø				PROG	PAYROL
2 Ø1Ø2Ø	*				PAYROLL EXAMPLE PREPARED FOR EDUCATION RESEARCH TEXTBOOK 200

22. There are two conditions in which a blank OPERAND field is valid:

1. The instruction does not require an OPERAND. (Such as H, NOP, etc.)
2. Operands are implicitly addressed as in chaining, where the address of the A operand is supplied by the contents of the A ADDRESS register, etc.

---

36. Instead of specifying the location of a data field directly, it is sometimes useful to designate other memory locations, which in turn contain the address of the desired data. Addressing accomplished through memory locations which contain the address of desired data is not direct addressing. Consequently, this type of addressing is called INDIRECT addressing.

---

50. WORD MARKS are automatically placed with instructions. Example: The Op. Code of any instruction is automatically word marked.

What would be the resultant punctuation when an L is written in Column 7 of an instruction? RECORD MARK.

---

64. In order to assign the tags, X1, X2, X3, X4, X5, X6, to the actual addresses of the index registers, ADDRESSES: 4, 8, 16, 20, 24, (the absolute addresses of the registers) must be written in the operands field. Fill in the coding form to make the tag "X3" equal to the index register occupying memory locations #10, 11, and 12.

LOCATION	OPERATION CODE	OPERANDS
X3	EQU	12

8.

21

62

63 - 80

PAYROLL EXAMPLE PREPARED FOR EDUCATION RES

22.

OPERAND

A ADDRESS

36.

INDIRECT

50.

RECORD MARK

NOTE: A record mark is a combination of word and item mark. It may also be set by writing SW and SI instructions.

64.

LOCATION	OPERATION CODE	OPERANDS
8 ... 14	15 ... 20	21 ... 62
X3	EQU	12

Refer to the following chart for frame #65.

INDEX REGISTER	ADDRESS TYPE INDICATOR	STORAGE FIELD	ADDRESS
X 1	001	2-4	4
X 2	010	6-8	8
X 3	011	10-12	12
X 4	100	14-16	16
X 5	101	18-20	20
X 6	110	22-24	24

9. If a printed listing of all 80 card columns is desired, tabulating equipment (an accounting machine) or a source card print routine may be used. An * is placed in Column #6 when only a remark is written. As with the 1401, remarks may also be entered following the last entry in the operand field. Easycoder requires one space between the last operand and the first remark.

---

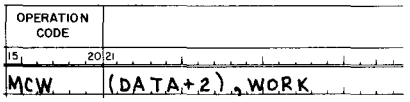
23. List the two conditions for which a blank operand field is valid.

1. NO OPERAND REQUIRED - H, NOP
2. CHANGING ADDRESS IN A ADDRESS REGISTER (IMPLICIT ADDRESSING)

In all other situations, the operands field will contain addresses (symbolic, absolute, indexed, indirect) octal variants or entries as remarks and constants.

---

37. Indirect addressing is an H-200 capability not found in your previous system. A programmer encloses the indirect address in parentheses, and the program then refers to that address for the desired data address. Indirect addressing can be compared to additional indexing in excess of the six available registers. Since an indirect address can specify another indirect address, etc., through any desired number of levels, the capability of multilevel indirect addressing is provided. Indirect addressing requires only that 1.) the indirect address be enclosed by parentheses, and 2.) the program is in three or four character addressing. Example:



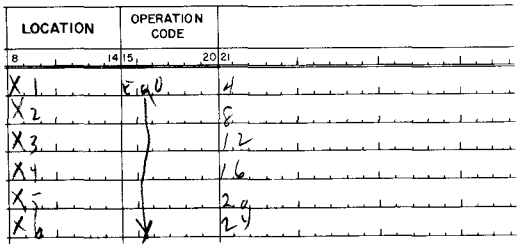
---

51. The following minimum hardware configurations are required for H-200 systems using:

	EASYCODER	EXTENDED EASYCODER
CENTRAL PROCESSOR	2048-character core storage	8192-character core storage
PERIPHERAL EQUIPMENT	Card reader/ card punch	Card reader/ card punch
	Printer	Printer
		3 Magnetic tape units

---

65. Write the statements making tags XI through X6 refer to their correct address.



9.

NO ANSWER REQUIRED

23.

1. OPERANDS NOT REQUIRED. EXAMPLES: H, NOP, ETC.
2. IMPLICIT ADDRESSING (CHAINING).

37.

NO ANSWER REQUIRED

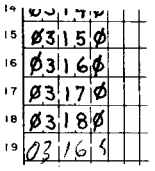
51. EXTENDED EASYCODER

In addition to blank, L, and R, there is another set of punctuation indicators available to Extended Easycoder. If any of the letters A through T (excluding L and R, O and Q) are written in column 7, word marking is not automatically placed by instructions. Any punctuation indicator from this second set, controls the complete punctuation.

65.

LOCATION	OPERATION CODE	
8 14	15 20	21
X1	EQU	4
X2	EQU	8
X3	EQU	12
X4	EQU	16
X5	EQU	20
X6	EQU	24

10. SPS or AUTOCODER uses lines 26-30 for insertions; EASYCODER permits insertions to be written for any line, on any line. Indicate that a line is to be inserted between lines 16 &17.



---

24. In certain cases, either or both operand addresses are written as zeros on the coding form. Actual addresses will then be supplied by another instruction. For example, SCR - which is similar to SAR or SBR of the 1401 - supplies operand addresses to a Resume Normal Mode instruction as part of an interrupt routine. The coding of this portion of an interrupt routine is illustrated on the answer side of this frame.

---

38. In preceding lessons, it was shown that a "two character" address - 12 binary digits - can express any address from 0 to 4095. "Two character" refers to the fact that the six character bits from two adjacent memory locations form a continuous 12 bit address. When "three character" addressing is required for addresses above 4095 or for INDEXED or INDIRECT addressing, a total of THREE adjacent memory locations form a continuous 18 bit address.

---

52. EXTENDED EASYCODER

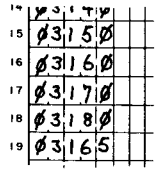
Specifically, the punctuation indicators A through T (excluding L and R, O and Q) set whatever punctuation is required. Consequently, this second set of punctuation indicators controls WORD marks, ITEM marks, and RECORD marks, in any combination of leftmost or rightmost memory locations (extremes).

66.

The EQU statement is often used to make other tags equal to index registers. In the previous discussion of indexing, you saw that the value stored as the contents of an index register could be used to modify an address. For example, DATA +X1, instructs the computer to add the value stored in X1 to the address of the symbolic tag DATA. (Continue to the answer side of this frame.)

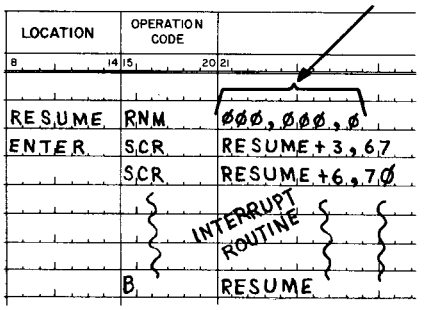
10.

NOTE: Any digit in the insertion column is correct. However, it is a common practice to number insertions with a central digit between 0 and 9. In this manner, insertions could then be made between the original line and the first insertion.



24.

Notice these zeros.



Interrupt Routine Begins at this point →

Stores "A" Address Register (67).

Stores "B" Address Register (70).

38.

THREE (3)

18

52.

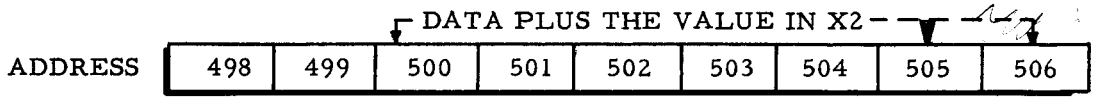
WORD

ITEM

RECORD

66.

Suppose that the tag DATA has been assigned to memory location #500 and that X2 contains a value of 5. Retrieval of the desired operand DATA +X2 is shown below:



The computer begins retrieving the operand at memory location #505. Because of indexing, the effective operand address, DATA +X2, has been modified without actually changing the original address of DATA.

11. A programmer should complete at least Columns 1-5 on the first line of each coding form. Columns 1 & 2 show PAGE NO., columns 3 & 4 show LINE NO., column 5 shows INSERTION NO.

The first Op. Code of a program is PROG. This causes assembly to take up to 6 characters written in the OPERATION field as the NAME of the PROGRAM.

An * in the TYPE column indicates a line of REMARKS. Any extension of this line beyond column # 72 will not appear in an assembled object program listing.

---

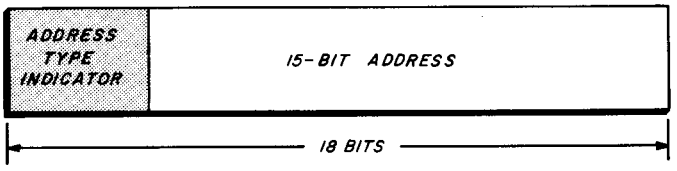
25. The SCR instructions move three character addresses from each register in the frame 24 example. A correct number of memory locations must previously have been allocated during assembly for storage of these addresses.

Briefly, then, what do the ØØØ, ØØØ, of the RNM instruction indicate to assembly?

address to be supplied by SCR instruction
(ALLOCATE SIX MEMORY LOCATIONS 999, 000 to receive A & B addresses from other instructions)

---

39. The high order three bits of the 18 bits available in three character addressing are used to indicate whether addressing is to be accomplished directly, indirectly, or by indexing. These high order three bits are illustrated below. They are called the ADDRESS TYPE INDICATOR.



---

53. EXTENDED EASYCODER

The punctuation indicators A, B, C, place a WM, IM, RM, respectively in the left most memory location.

The indicators D, E, F, place the same respective punctuation at the other extreme, that is,

the letter D sets a WM in the RIGHT most location,
the letter E sets an IM in the [illegible] most location,
the letter F sets a RM in the [illegible] most location.

---

67. The contents of an index register could also be the address of an operand. In this case, it is written as Ø+X1, Ø+X2, etc. It is important to remember that index designators such as +X1 or Ø+X1 specify that the CONTENTS of a certain register is to be used to locate another address in memory.

11.

Columns 1 & 2 show PAGE NUMBER
Columns 3 & 4 show LINE NUMBER
Column 5 shows INSERTION NUMBER

PROG

SIX

OPERANDS

NAME

PROGRAM

* - TYPE - REMARKS - 62

25.

ALLOCATE SIX MEMORY LOCATIONS (ØØØ, ØØØ) TO RECEIVE A AND B ADDRESSES FROM OTHER INSTRUCTIONS.

NOTE: When a variant character is to be stored, the operands field entry should be ØØØ, ØØØ, Ø.

39.

ADDRESS TYPE INDICATOR

53.

D sets a WM in the RIGHT most location.

E sets an IM in the RIGHT most location.

F sets a RM in the RIGHT most location.

67.

CONTENTS

12. A symbolic tag (label) is composed of from one to six characters, the first of which must be alphabetic. Tags are written in the LOCATION columns # 8 through # 14.

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1 Ø1Ø1Ø				PROG	PAYROL
13 Ø1113Ø			START	SW	1ØØ,2ØØ

26. The operands field may be blank if no operands are involved or chaining is being performed. However, if addresses are to be supplied by another instruction, the correct number of (ZEROS) CHARACTERS should be written in the operands field. Assembly will then allocate the CORRECT NUMBER of storage memory locations.

40. In two character addressing, the computer is not involved with any address type indicator. In three character addressing, the high order three bits indicate either direct, indirect, or indexed addressing. 000=DIRECT, 111=INDIRECT. Write the address type indicators as they appear in binary, indicating the index registers 1 through 6.

001 =X1 010 =X2 011 =X3

100 =X4 101 =X5 110 =X6

54. EXTENDED EASYCODER

The remaining punctuation indicators (G through T) place combinations of punctuation at both extremes.

COLUMN 7	LEFTMOST LOCATION	RIGHTMOST LOCATION
G	IM	IM
H	IM	WM
I	IM	RM
J	WM	IM
K	WM	WM
M	WM	RM
N	Δ	Δ
P	RM	WM
S	RM	IM
T	RM	RM

In addition to L and R, which two letters have been omitted? O, Q. Which letter places no punctuation at either extreme? N.

68. EQU is used to assign a tag to index register designators in the example below:

This ADD instruction refers to the index register X3, which is the address of the data to be added to NET.

LOCATION	OPERATION CODE	OPERANDS
TAX	EQU	Ø+X3
	A	TAX,NET

12.

LOCATION
#8 - #14

26.

ZEROS
CORRECT NUMBER

40. Notice that the index registers occupy memory locations 2 through 24. This chart will be presented again in this lesson and reference will be made to the addresses (4, 8, 12, 16, 20, 24) of the index registers.

INDEX REGISTER	ADDRESS TYPE INDICATOR	STORAGE FIELD	ADDRESS
x 1	001	2-4	4
x 2	010	6-8	8
x 3	011	10-12	12
x 4	100	14-16	16
x 5	101	18-20	20
x 6	110	22-24	24

54. EXTENDED EASYCODER

O, Q.
N

NOTE: The first set of punctuation indicators (blank, L, R,) is usually sufficient. The second set of punctuation indicators may be used at the programmer's discretion.

68. CONTENTS

NOTE: A tag that has been made equal to an index register designator of the type Ø+X3, may only be used to specify the contents of the register and not the address of the register itself.

13. An Easycoder tag may begin in either the first Location column (#8) or the second Location column (#9). The address assigned to a tag by assembly is determined by two variables:

1. Whether the tag begins in LOCATION column # 8 or # 9.
2. Whether the tag refers to an instruction or to constants and reserved areas.

---

27. Previous experience has made you familiar with the several types of addresses which may be entered in the operands field.

For example, any unsigned decimal number from 0 up to the limit of memory constitutes an absolute address. While you referred to this as an "actual" address in your previous system, Easycoder terminology calls it an ABSOLUTE address.

---

41. On the back of this frame (#41) and in frame #42 you will be shown how the computer retrieves indirect or indexed addresses. You are to compare frame 41 with frame 42 and decide which frame illustrates indirect addressing of the A address and which frame illustrates indexing of the A address.

---

55. You will recall that assembly language was divided into three types of statements at the start of this text. These are:

1. ASSEMBLY Control Statements
2. DATA Formatting Statements
3. Data PROCESSING Statements.

The third type was separated further into five kinds of instructions.

---

69. Now, fill in this coding form to make tag FICA refer to the contents of index register X6.

LOCATION	OPERATION CODE	OPERANDS
8 ... 14	15 ... 20	21 ... 62
X6	EQU	24
FICA	EQU	0+X6

13.

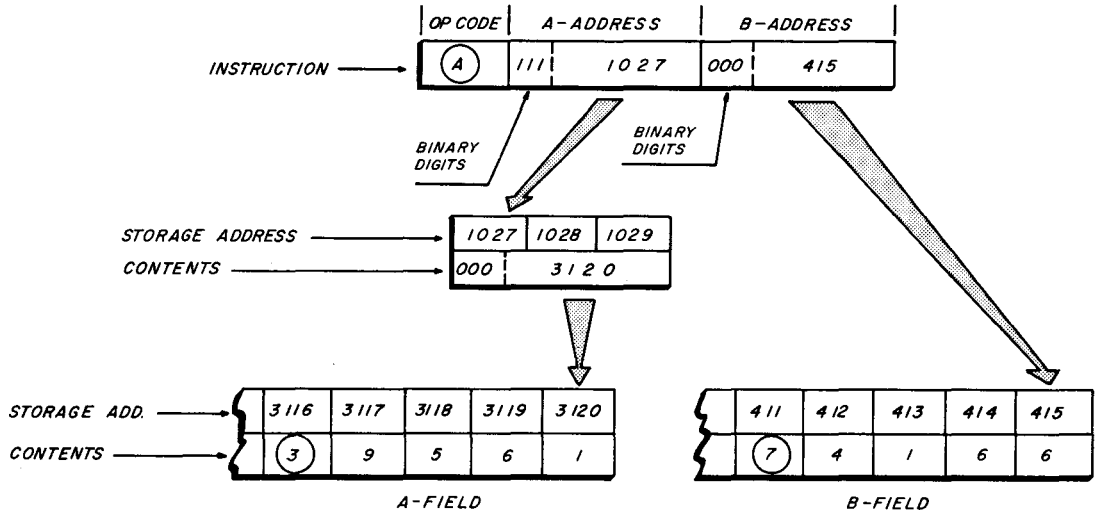
LOCATION

#8 or #9

27.

ABSOLUTE

41.



This frame is an example of _____ addressing of the A address.

55.

ASSEMBLY

DATA

PROCESSING

69.

LOCATION	OPERATION CODE	OPERANDS
8 ... 14	15 ... 20	21 ... 62
X6	EQU	24
FICA	EQU	Ø+X6

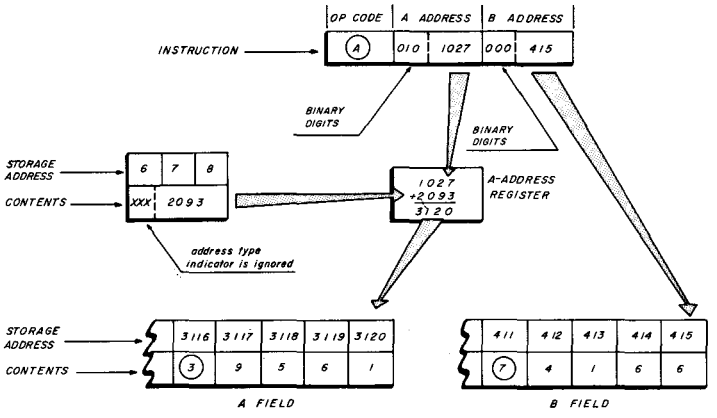
14. Consider first a tag that starts in column #8 and refers to an instruction. Due to the direction of instruction retrieval, the address assigned to the tag by assembly will be the memory location containing an OP CODE.

28. Identify the types of addresses in the example below:

The first line shows ABSOLUTE addresses.
The second line shows SYMBOLIC addresses.

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	
	1 2 3 4 5	6	7 8	14	15 20	21
1	Ø2Ø1Ø				A	375,450
2					S	TAX,PAY

42.



This frame is an example of INDEX addressing of the A address.

56. List those of the ten Easycoder Assembly Control mnemonics you remember: CEQU

PROG	ADMODE	END
EY	HSM	CLEAR
ORG	MORG	EQU

70. Compare the two examples below, then briefly describe the different effects of the ADD Instructions.

Example #1

LOCATION	OPERATION CODE	OPERANDS
8 14	15 20	21 62
X6	EQU	24
FICA	EQU	Ø+X6
	A	FICA,X6

Example #2

LOCATION	OPERATION CODE	OPERANDS
8 14	15 20	21 62
X6	EQU	24
FICA	EQU	Ø+X6
	A	FICA,FICA

14.

OP. CODE

(RETURN TO FRAME 15, PAGE 135.)

28.

ABSOLUTE
SYMBOLIC

(RETURN TO FRAME 29, PAGE 135)

42.

FRAME 41 IS AN EXAMPLE OF INDIRECT ADDRESSING
FRAME 42 IS AN EXAMPLE OF INDEXED ADDRESSING

(RETURN TO FRAME 43, PAGE 135.)

56.

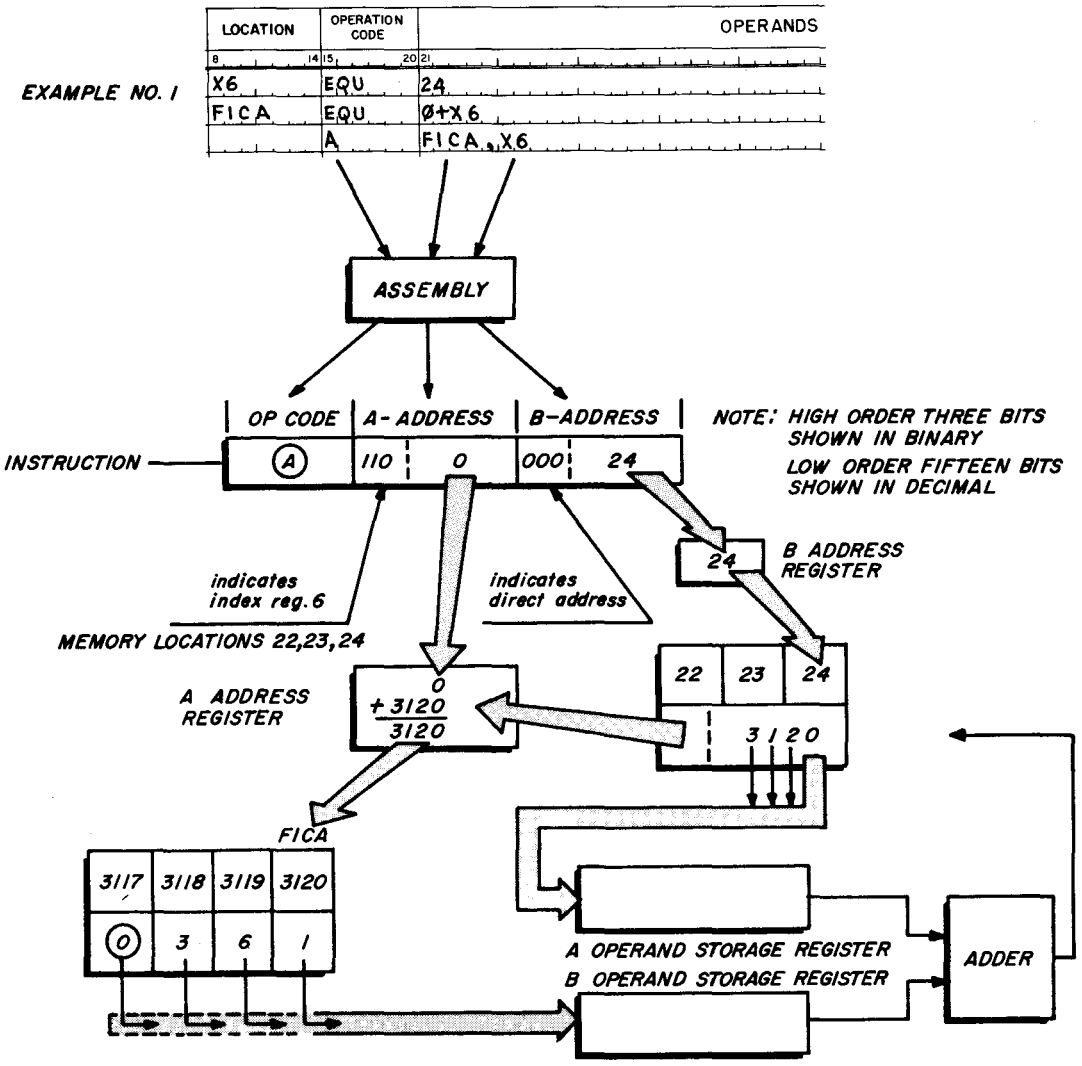
PROG	MORG	EX	EQU	HSM
ORG	ADMODE	CLEAR	CEQU	END

(RETURN TO FRAME 57, PAGE 135.)

70.

EXAMPLE #1. THE DATA STORED AT THE MEMORY ADDRESS INDICATED BY THE CONTENTS OF X6 WILL BE ADDED TO MEMORY LOCATION #24. SINCE THIS IS THE ADDRESS OF X6, THE CONTENTS OF X6 WILL BE CHANGED.
EXAMPLE #2. THE DATA STORED AT THE MEMORY ADDRESS INDICATED BY THE CONTENTS OF X6 WILL BE ADDED TO ITSELF. THE MEMORY ADDRESS - FICA - REMAINS IN X6 AND IS NOT CHANGED.
(The diagram on page 163 illustrates how the computer executes Example #1)

71.



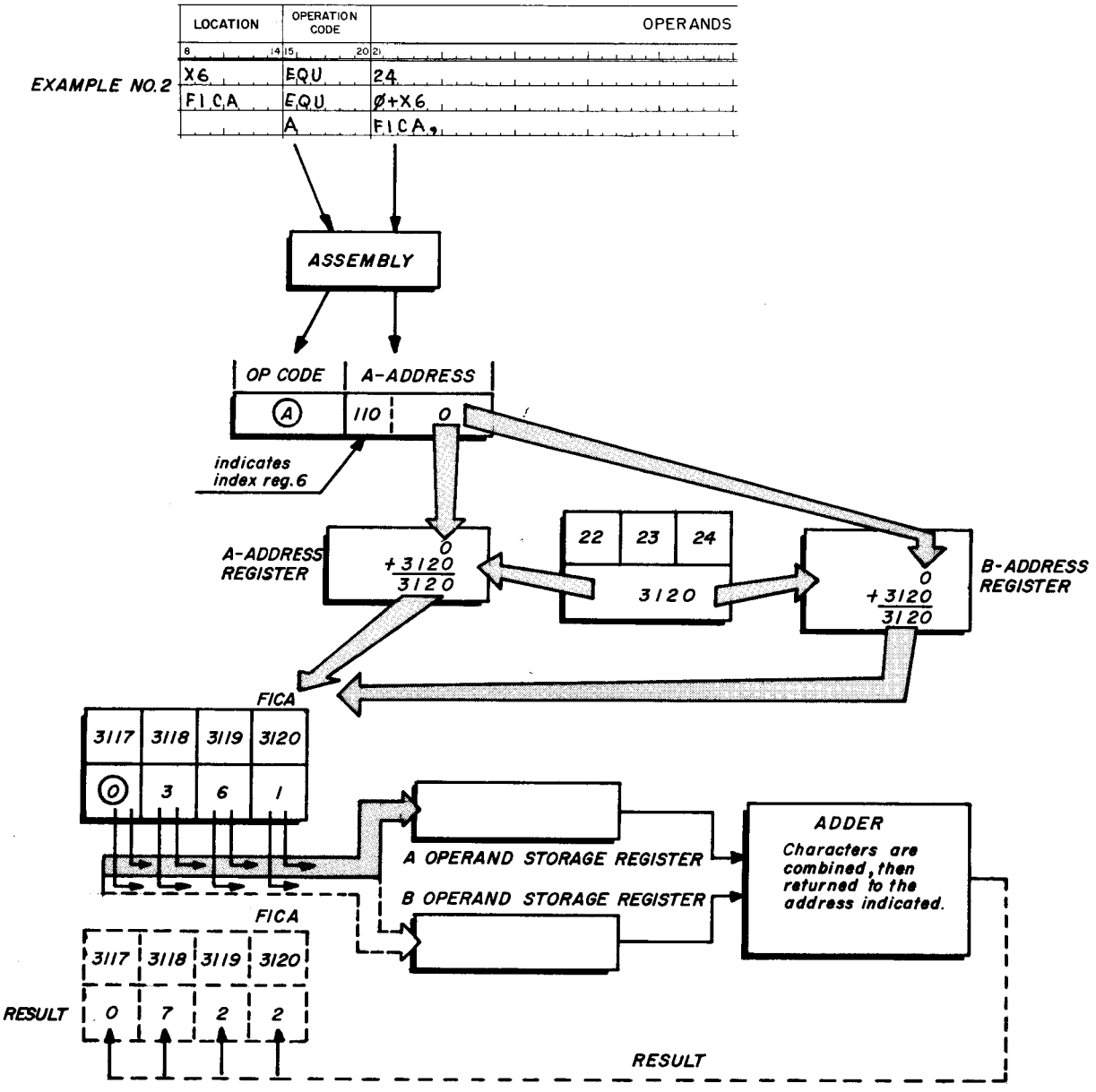
The A address is supplied to the A address register from index register X6. The B address is supplied to the B address register from the address in the instruction. A character from each operand is sent to its respective operand storage register and is then combined in the adder. This process continues and the result is sent back to the address indicated by the B address register until the character with a word mark has been processed.

What will the total be at the completion of the ADD operation and where will it be stored?

3481 22,23,24.

71. The total stored in memory locations 22, 23, 24, will be 3481. Since memory locations 22, 23, 24, constitute index register 6, subsequent use of this register will involve the value 3481. The example explained was a special case where it was desirable to change the contents of X6.

The coding below illustrates the second example that was written A, FICA, FICA. Note, however, that the appropriate format is written more efficiently as Op. Code, A Address. This format simply duplicates the A Address in an Add instruction.



If you are interested in another example of indexing, you may review frame #42 on page 161 before continuing to frame 72, page 165.

72. The examples just presented combined a review of indexing with the use of an EQU assembly control statement. This does not mean that EQU is used only with index tags. EQU was simply demonstrated in conjunction with indexing.

The purpose of EQU is to make a tag written in the location columns EQUAL to the ADDRESS in the operands field. This address may be direct, indirect, or indexed.

---

83. EX (for EXecute) is similar to the EX statement with which you are familiar. Briefly explain the purpose of an EX statement.

To execute a portion of the program before the entire program loaded.

---

94. The first direct address specifies the lowest memory location to be cleared of punctuation and data bits. It is separated from the second direct address by a comma. If this "highest" (second) address is not also followed by a comma - the data bits in the area specified are cleared to zeros. Indicate that the area from WORK to WORK + 10 is to be cleared to zeros.

LOCATION	OPERATION CODE	OPERANDS
	CLEAR	WORK, WORK+10
	END	

---

105. Alternatively, an alphanumeric constant may be written as follows:

LOCATION	OPERATION CODE	OPERANDS
		#12AUNIT#6@$1.20

The notation #12 A is interpreted by assembly as meaning the number (#) of memory locations (12) to store the alphanumeric (A) constant. Then, the constant to be stored is specified as UNIT#6@$1.20. Using this type of notation, indicate that TYPE 3 is to be stored as a constant without a word mark.

LOCATION	OPERATION CODE	OPERANDS
	DC	#5A3TYPE3

72.

EQUAL

ADDRESS

---

83. THE PURPOSE OF THE EX STATEMENT IS TO EXECUTE A PORTION OF A PROGRAM BEFORE THE ENTIRE PROGRAM HAS BEEN LOADED BY A LOADING ROUTINE.

A programmer writes mnemonic EX in the op code field. He then writes a previously defined address in the operands field. This address is that which appears in the location field of the first instruction of the segment to be executed.

94.

LOCATION	OPERATION CODE	OPERANDS
	CLEAR	WORK,WORK+10
	END	

105.

LOCATION	OPERATION CODE	OPERANDS
	DC	#5ATYPE3

73. The EQU statement is not required as part of a specific sequence of assembly control statements. As you remember, PROG is written first, and ORG follows PROG. The next two statements that need to be written are ADMODE and CAM. ADMODE specifies the mode of ADDRESS (2, 3, or 4 characters) in which to start assembly. CAM actually changes the ADDRESS MODE to 2, 3, or 4 characters.

---

84. Since the purpose of an EX statement is to execute portions of the program before the remainder is loaded, more coding follows an EX statement.

At the END of program coding, the assembly control statement END is written in the Op. Code field of the final coding line.

---

95. A comma written following the second address indicates that the area is to be cleared with the character written after the comma. Punctuation bits will always be cleared. Indicate that ten memory locations starting at address #150 are to be cleared with X characters.

LOCATION	OPERATION CODE	OPERANDS
	CLEAR	150,150+10,X
	END	

---

106. Decimal constants (signed or unsigned) are simply written beginning in column 21 of the DC or DCW operands field. If a sign is specified, it will be denoted in memory by the zone bits (B and A) of the rightmost character. Example: DCW -212 produces 10 0010 (Octal 42), while DCW +212 produces 01 0010 (Octal 22) as the rightmost character in memory.

The examples above demonstrate that a + sign is stored in the rightmost memory location with the BA cores respectively 0 and 1. A minus sign causes the BA cores of the rightmost memory location to be 1 and 0.

73.

ADDRESS

ADDRESS MODE

84.

END

The programmer:

1. Writes END in the op. code field.
2. May write a previously defined address (either absolute or symbolic) in the location field, to indicate the location of the 80 - character object program loading area. If the location field is left blank, an 80-character leading area is automatically reserved by the assembly program immediately following the last assembled instruction.

95.

OPERATION CODE	OPERANDS
CLEAR	150,159,X
END	

106.

BA

\+ = 01 in the rightmost memory location

\- = 10

NOTE: Each digit (0 - 9) in the preceding examples will be stored in memory as a separate character, with the sign of the group shown by the rightmost character.

\+ 212 in memory as 00 0010 00 0001 01 0010

\- 212 in memory as 00 0010 00 0001 10 0010

74. ADMODE is an assembly control mnemonic op. code. It indicates the mode of addressing for assembly when either a 2, 3, or 4 is written in the operands field. Specify three character addressing on the coding form below.

OPERATION CODE	OPERANDS
PROG	PAY ROL
ORG	100
ADMODE	3

85. A programmer writes an EX statement to execute a portion of a program. He also needs to have written a branch instruction as the last entry in the segment to be executed. This branch instruction refers to the address in the location field (columns 8 - 14) of the END statement.

At the completion of the segment being executed, the program will BRANCH to the address written in the location field of the END statement.

96. HSM (High Speed Memory) assembly control statement is used with EASYCODER but is not required by EXTENDED EASYCODER. HSM is written to cause a card deck of memory contents to be punched. This "memory dump" deck can then be used to print a listing of complete memory contents when desired. If an HSM statement is written, it must immediately precede CLEAR and END statements. A total of no more than 10 HSM, CLEAR and END statements may be written for an EASYCODER system. HSM and a memory dump printed listing are illustrated after this lesson.

107. DC or DCW constants may also be written in decimal by the programmer, but they can be specified to be interpreted by assembly as a binary value. For example, when a two character (two memory locations) binary constant with a decimal value of 212 is desired, it will be written as follows:

DC	#2B212

The notation #2B means that the number (#) of memory locations to be used is 2 and the constant is to be stored as a Binary value equal to 212 in decimal.

74.

OPERATION CODE	OPERANDS
15 20	21 62
PROG	PAYROL
ORG	100
ADMODE	3

85.

BRANCH

END

The programmer must write a branch instruction to the address in the location field of END as the last instruction of the segment to be executed. Since the location field of End contains the address of the object program loading area, branch returns control to the loading routine.

96.

NO ANSWER REQUIRED

Data formatting statements are discussed beginning in frame 97.

107.

2

B

212

NOTE: As a result of the statement DC #2B212 this binary number with a value of 212_{10} will occupy two memory locations as:

000011 | 010100

because, $2^7 + 2^6 + 2^4 + 2^2 = 128 + 64 + 16 + 4 = 212_{10}$

75. ADMODE is an assembly control statement, and it should always be followed by a CAM (Change Addressing Mode) instruction.

Whereas ADMODE simply directs the assembly program, CAM is required to actually CHANGE the ADDRESSING MODE of the computer.

---

86. The location field of the END statement provides the address of the object program loading area. Consequently, after execution of a portion of a program, the branch instruction refers to the location field of the END statement. This provides the address of object program LOADING area. Loading then continues for the portion of the program that follows EX.

---

97. Constants and reserved areas are defined in Easycoder with one of four data formatting statements. (RESV, DC, DCW, and DSA).

Obviously, the statement that causes assembly to set aside a specified number of memory locations is RESV. A tag beginning in location column #8 of this type of statement refers to the RIGHT most memory location.

---

108. The preceding example assumed that the programmer knew the value to be 212 in decimal, but wanted it stored in "two characters" as the 12 bit number:

000011010100

However, if the situation were such that the programmer knew a binary number and wanted to store it as a binary number, it would be more convenient to convert the binary to octal. Then, octal notation would be preceded by #2C in the DC or DCW statement. In this case, the #2 signifies two memory locations and the C specifies that the following digits are octal. This is illustrated on the answer side of this frame.

75.

CHANGE ADDRESSING MODE

Mnemonic: CAM

86.

LOADING

Refer to the illustration below to complete the sentences in frame 87.

LOCATION	OPERATION CODE	OPERANDS
LOAD	EQU	381
	ORG	381
START	SW	405,500
	MCW	BL, LOC
	≷	
	B	LOAD
	NOP	
	EX	START
	SW	TAX, PAY
	≷	
LOAD	END	

97.

RESV

RIGHT

The programmer:

1. Writes the mnemonic code RESV in the op code field.
2. Writes the number of characters to be reserved in the operands field. This may be written as a decimal or symbolic entry. If a symbolic tag is written, it must be defined previously in the source program.
3. May write an actual or symbolic address in the location field. The programmer can refer to the reserved location via this tag.

108.

Suppose the programmer knows the binary number. He could first convert it to octal and then write octal constant as follows:

$$\underbrace{000}_{0_8}\ \underbrace{011}_{3_8}\ \underbrace{010}_{2_8}\ \underbrace{100}_{4_8}{}_2$$

Since 2 octal digits constitute two Characters, (two memory locations) the DC or DCW is written:

DC	#2C0324

76. CAM specifies whether the change should be to 2, 3, or 4 character addressing mode. The desired mode is indicated by the VARIANT character written in the operands field.

Variant characters are written in octal so as to represent six binary digits. Therefore, the operands field entry of a CAM instruction will contain a total of 2 octal digits and is called a VARIANT character.

---

87. The first OP. CODE in frame 86 equates address #301 to tag LOAD. Since this tag is also the entry in columns 8 - 14 of END, the 80 character area beginning at address #301 will be used for object program LOADING. Assembly begins assigning sequential addresses at # 301 due to the ORG statement. Assembly continues assigning addresses until the E statement is encountered. (NOP does not affect assembly. NOP provides a word mark terminating retrieval of B.)

List the sequence of events from when assembly encounters EX until END is assembled.

EXECUTION OF PROGRAM FROM LABEL 'START' UNTIL THE INSTRUCTION BRANCH IS ~~ENCOUNTERED~~ EXECUTED. BRANCH REFERS TO LOCATION FIELD OF THE "END" INSTRUCTION. THIS FIELD TAGGED "LOAD" IS EQU TO ADDRESS #301. THE LOADING ROUTINE FOR ASSEMBLY THEN CONTINUES WITH THE INSTRUCTIONS FOLLOWING 'EX'

98. Assign the tag "DATA" to refer to the LEFTMOST of 80 reserved memory locations.

LOCATION	OPERATION CODE	OPERANDS
8 ... 14	15 ... 20	21 ... 62
DATA	RES.V	80.

---

109. In Lesson VIII you will write DC or DCW statements in octal for use as "MASKS" in conjunction with EXTRACT instructions. Write the statement defining a word marked constant in octal to occupy three characters (memory locations) such that all bits are 1's. Tag this statement as MASK 3, referring to the rightmost memory location.

LOCATION	OPERATION CODE	OPERANDS
8 ... 14	15 ... 20	21 ... 62
MASK3	DC.W	#3C777777

76.

TWO, THREE, or FOUR Character Addressing
TWO Octal Digits
VARIANT Character.

87.

LOAD
LOADING
381
ORG
EX

When EX is encountered, execution begins with the portion of the program tagged "START". Execution continues until the "BRANCH" instruction is executed. Branch refers to the location field of the END instruction. This field tagged "LOAD" is EQU to address #301. The loading routine for assembly then continues with the instructions following EX.

98.

LOCATION	OPERATION CODE	OPERANDS
DATA	RESV	8Ø

NOTE: A tag beginning in column 9 of a constant or reserved area refers to the leftmost memory location.

109.

$$\underbrace{111}_{7}\ \underbrace{111}_{7}\ \underbrace{111}_{7}\ \underbrace{111}_{7}\ \underbrace{111}_{7}\ \underbrace{111}_{7}$$

LOCATION	OPERATION CODE	OPERANDS
MASK3	DCW	#3C7777777

NOTE: A tag beginning in column 8 of a constant or reserved area refers to the rightmost location.
A tag beginning in column 9 of a constant or reserved area refers to the leftmost memory location.

77. The CAM variants to specify two, three, or four character addressing are octal: 20, 00, 60, respectively.

Write the ADMODE assembly control statement for four character addressing, then the CAM instruction and its appropriate octal variant.

LOCATION	OPERATION CODE	OPERANDS
	ADMODE	4
	CAM	60

88. It may be desirable to overlay the portion of a program that has been assembled and executed, thereby, utilizing memory more efficiently. The executed portion will be overlaid by subsequent instructions when an appropriate ORG statement is written following the EX statement. The example on the answer side of this frame illustrates an ORG statement causing the preceding executed portion to be overlaid.

99. The DSA (Define Symbol Address) data formatting statement is written to store one, or two addresses as a constant. If desired, variant characters may also be written and stored. The assembled length of each address written in the operands field is determined by the current address mode.

Write the statement to store the A and B address ITEM - 5, PAY +X6, as a constant.

LOCATION	OPERATION CODE	OPERANDS
	DSA	ITEM-5, PAY+X6

110. Write statements to accomplish the following:

1. Reserve 80 memory locations, tag the rightmost as CARDIN.
2. Store the addresses ITEM - 5, PAY +X6 as a constant.
3. Define 20 blank memory locations as a word marked constant.
4. Define TAX DEDUCTABLE as a constant without a word mark.
5. Define UNIT #6@$1.20 as a word marked constant.

LOCATION	OPERATION CODE	OPERANDS
CARDIN	RESV	80
	DSA	ITEM-5, PAY+X6
	DCW	#20
	DC	@TAX DEDUCTABLE@
	DCW	UNIT@$1.20

77.

LOCATION	OPERATION CODE	OPERANDS
	ADMODE	4
	CAM	6Ø

88. Notice which coding will be overlaid by the remainder of the program, then continue to frame 89.

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1	Ø2Ø1Ø			LOAD	EQU	3Ø1
2	Ø2Ø2Ø				ORG	381
3	Ø3			START	SW	4Ø5,5ØØ
4	Ø4				MCW	BL,LOC
5	Ø5				S	
6	Ø6				S	
7	Ø7				B	LOAD
8	Ø8				NOP	
9	Ø9				EX	START
10	1Ø				ORG	381
11	11			START2	SW	TAX,PAY
12	12	*			S	PROGRAM CODING CONTINUES
13	13	*			S	UP TO LINE 30
30	3Ø			LOAD	END	START2

99.

LOCATION	OPERATION CODE	OPERANDS
	DSA	ITEM-5,PAY+X6

NOTE: A word mark will be automatically placed at the leftmost character of the field. If column #7 contains an R, an item mark will be set at the rightmost character. If column #7 contains an L, a record mark will result at the leftmost character. (Word mark and item mark = record mark.)

110.

LOCATION	OPERATION CODE	OPERANDS
CARDIN	RESV	8Ø
	DSA	ITEM-5,PAY+X6
	DCW	#2Ø
	DC	@TAX DEDUCTABLE@
	DCW	=UNIT#6@$1.20=

NOTE: Numbers 4 and 5 could be written:

DC # 14 A TAX DEDUCTABLE

DCW # 12 A UNIT # 6@or, number 5

could be written surrounded by any character except +, -, #, digits Ø - 9 or a character in the constant.

78. An octal CAM variant of 20 allows memory locations to be addressed up to #4095 ($111111111111_2 = 4095_{10}$).

A CAM variant of octal ØØ provides three character addressing, in which memory locations up to #32767 may be addressed. Indexing and Indirect addressing are available when addressing is in at least three characters. Identify the name or purpose of the high order three bits shown

One Character		One Character	One Character
101	111	111111	111111

ADDRESS TYPE INDEX-WORD

$= 32767_{10}$

89. If the operands field of an END statement remains blank, the machine will halt after completing the loading. A programmer may write an address (symbolic or absolute) in the operands field of the END statement. When this address is written in the operands field, it designates the point at which execution is to start at the completion of loading.

Refer to the overlaid example illustrated in frame 88, then write the appropriate address designating the point where execution is to start after loading has been completed.

100. Using the self reference * as the B address (indicating the leftmost character of the DSA) and NET as the A address, write the statement storing them as a constant. Indicate that an item mark is to be placed at the rightmost character.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8 ... 14		15 ... 20	21 ... 62
¢		DSA	NET,*

111. Write statements to accomplish the following:

1. Define decimal - 26 as a constant without word mark.
2. Define the decimal number 26 to be stored as a single Binary memory location with a word mark.
3. Define binary 1111110110000000000 as three characters in memory tagged MASK 2 without a word mark.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8 ... 14		15 ... 20	21 ... 62
		DC	-26
		DCW	#626
	MASK2	DC	#3C77300

78.

ADDRESS TYPE INDICATOR

INDICATES DIRECT, INDIRECT, OR INDEXED ADDRESSING.

---

89. START 2 in the END statements operand field causes execution to begin with the instruction on line #11.

	09				EX	START
10	10				ORG	381
11	11		START2		SW	TAX, PAY
12	12	*			C	PROGRAM CODING CONTINUES
13	13	*			C	UP TO LINE 30
30	30		LOAD		END	START2

The rules regarding END and EX statements are reviewed in frame 90 and its answer space on the following page.

100.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 14	15 20	21 62
R		DSA	NET, *

111.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 14	15 20	21 62
		DC	-26
		DCW	#1826
	MASK2	DC	#3C7730ØØ

79. Indexing and indirect addressing are also available in any system with sufficient memory to make use of four character addressing. Four character addressing provides 24 bits, of which the high order 3 bits serve as address type indicators. Only the low order 16 bits are needed to address the locations up to #65535. $1111111111111111_2 = 65535_{10}$

---

90. For an END statement, the programmer:

1. Writes END in the op code field.
2. May write a previously defined address (either absolute or symbolic) in the location field, which specifies the location of the 80-character object program loading area. If the location field is left blank, an 80-character loading area is automatically reserved by the assembly program immediately following the last assembled instruction.
3. Writes an address in the operands field if it is desired to execute the object program immediately after loading. This address designates the location of the first object program instruction to be executed. The address may be either absolute or symbolic. If the operands field is left blank, the machine will halt after the loading routine has been completed.

---

101. Use of the data formatting statements DC (Define Constant without word mark) and DCW (Define Constant with word mark) should be familiar from your previous experience. As a convenience for the programmer, constants may be written in DC and DCW operands fields specifying either alphanumeric, decimal, binary, octal, or the number of memory locations to be set to blanks.

---

112. The answer side of this frame through frames and answer sides 115 illustrate Easycoder card formats.

Page 187, Figure 16 shows two and three character addressing.

Page 188 may be used for future reference concerning the CAM instruction and its variants.

You will not be required to answer any questions until page 189.

79. The capability of increasing memory size to 65000 + demonstrates the H-200's expansibility and versatility of binary addressing.

The appropriate octal variants for CAM instructions are : TWO CHARACTER 2Ø, THREE CHARACTER ØØ, FOUR CHARACTER 6Ø.

An example of efficient utilization of two and three character addressing is illustrated in Figure 16, page 187.

---

90. For an EX statement, the programmer:

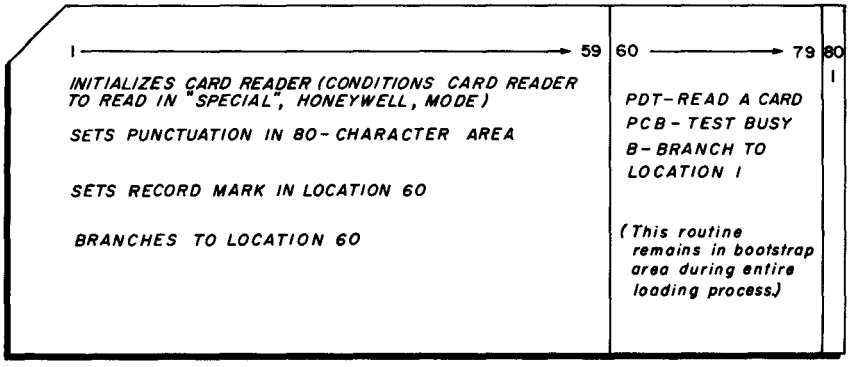
1. Writes the mnemonic code EX in the op code field.
2. Writes a previously defined address in the operands field. This address is that which appears in the location field of the first instruction of the segment to be executed.
3. Must have written a Branch instruction to the address specified in the location field of the End card as the last instruction of the segment to be executed. Since the location field of the End card contains the address of the object program loading area, this Branch instruction returns control to the loading routine.

101.

DEFINE CONSTANT with WORD MARK

112.

Bootstrap Card



The Bootstrap Card is the first card in the object program. The Bootstrap Card sets punctuation in the 80 - character area that will allow subsequent cards to be read. A record mark protects the routine which the Bootstrap Card sets into the area beginning at location 60. A read routine remains in this area during the entire loading process.

80. The assembly control statements discussed so far are: PROG, ORG, ADMODE, MORG, and EQU. The remaining assembly control statements are CEQU (Control Equal), EX (Execute) HSM (High Speed Memory - printed listing of memory), CLEAR, and END.

CEQU is similar to EQU in that it is used to assign a symbolic TAG to the entry in the OPERANDS field.

---

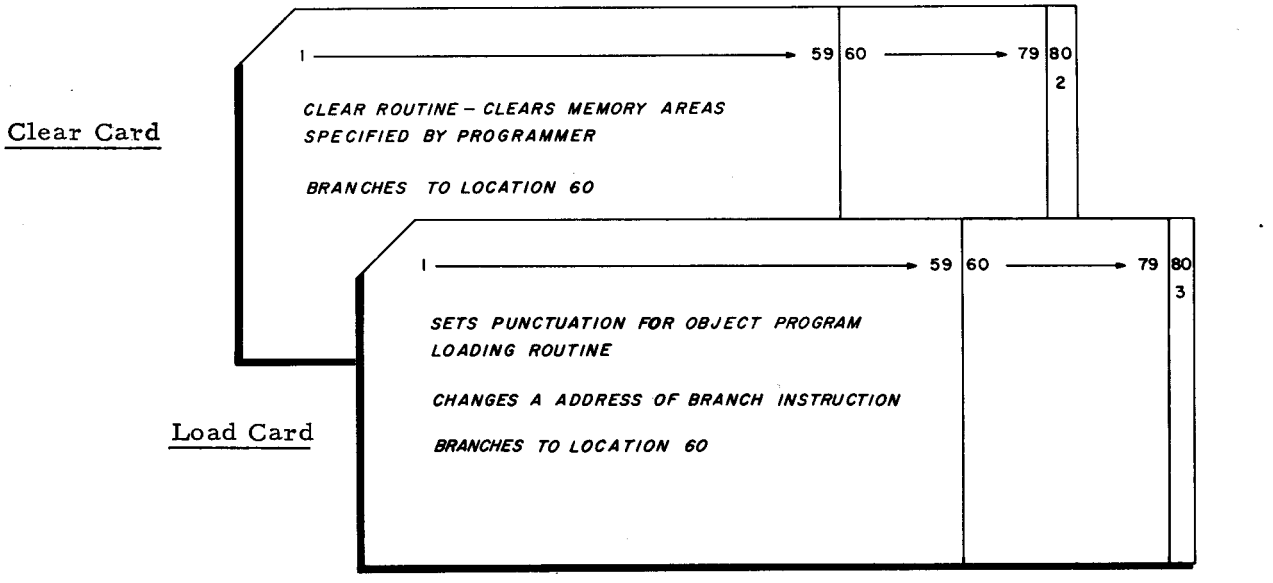
91. The END statement is always the last entry in a program. Immediately preceding the END statement, CLEAR statements may be written. As implied by the name of this op. code, its purpose is to CLEAR the memory area designated in its operand field.

---

102. Constants are limited to a maximum of forty memory locations. When DC or DCW is used to define a constant as blanks, a number sign, #, is written in column 21 of the operands field. # is followed by the number of blank memory locations desired. Indicate that fifteen blank memory locations are to be treated as a constant without a word mark and that twenty blank memory locations are to be a constant with a word mark.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14 15	20 21	62
		DC	#15
		DCW	#20

113.



80.

TAG

OPERANDS

91.

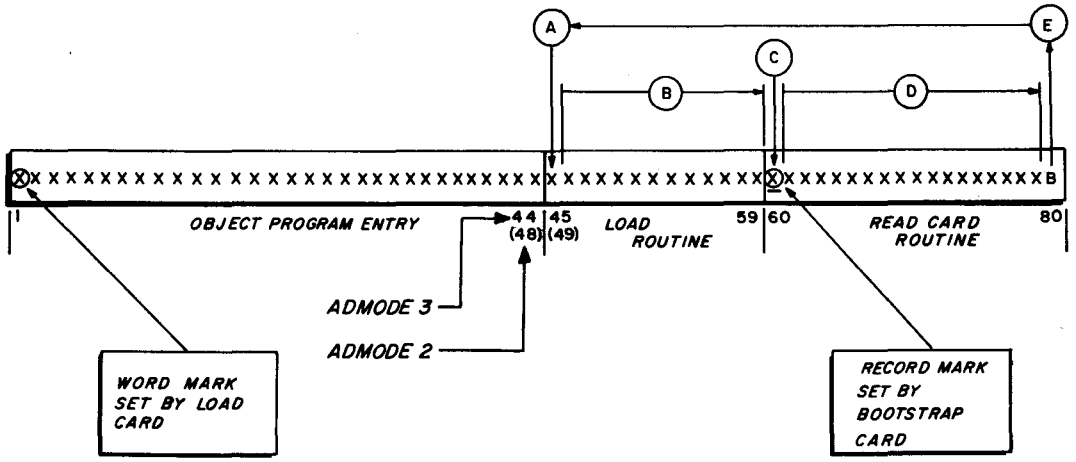
CLEAR

102.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
		DC	#15
		DCW	#20

NOTE: Tags and L or R in column #7 may be used with DC or DCW statements as desired.

113.



Bootstrap Area After Load Instruction

The object program card immediately following the Load Card is read into the first area, and a word mark is assigned to the op code. The first data to be read on the present card is the self-load routine (A). Execution (B) of this routine loads the entry into the memory area specified in this particular routine. Location 60 (C) contains a PDT instruction which allows the read routine (i.e., PDT, PCB, and B) beginning at this location to be executed (D). The following card is then read (E). Subsequent cards are self-loaded in this manner, until either an End card or an EX card is encountered by the machine.

81. CEQU is used to assign a tag to an octal value written in the operands field. You recently saw an octal value being appended to a CAM instruction where it specified the mode of addressing. Tags are often assigned to octal values that are used as VARIANT characters. Since the purpose of this type of character is control, it is appropriate to use a CEQU statement when assigning a tag.

---

92. CLEAR is used to specify an area of memory to be cleared of punctuation and data bits before loading of the program. Limits of the area to be cleared are specified in the operands field as TWO direct (not indexed nor indirect) addresses. This first direct address specifies the lowest memory location to be cleared. Consequently, the SECOND DIRECT ADDRESS SPECIFIES the HIGHEST MEMORY LOCATION to be CLEARED.

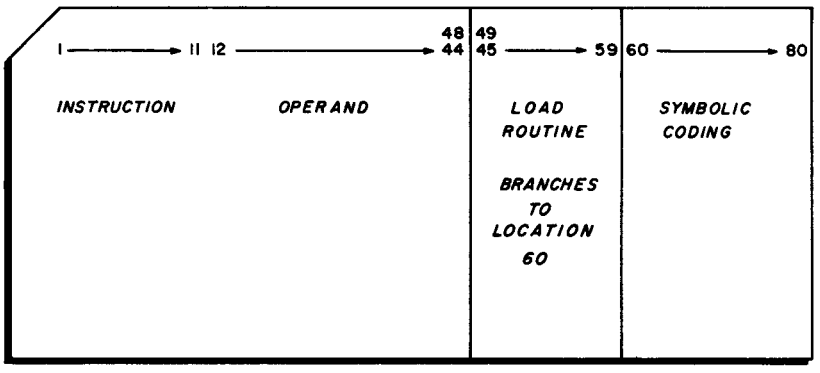
---

103. Constants may also be specified as either alphanumeric, decimal, binary, or octal. Alphanumeric constants may be written surrounded by @ symbols. A constant written in this manner can contain any symbol (including space) except the @ symbol. After the example below, write a DCW with UNIT #6 as a constant.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14	15 20	21 62
		DC	@TAX DEDUCTABLE@

---

114 Instruction card



81.

VARIANT
CEQU

NOTE: Instructions may use variant characters sometimes synonymously referred to as "control characters."

92.

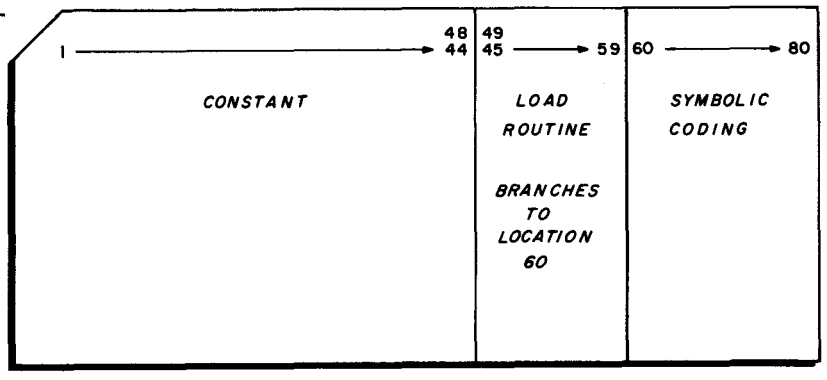
SECOND	DIRECT	ADDRESS	SPECIFIES
HIGHEST	MEMORY	LOCATION	CLEARED

103.

LOCATION	OPERATION CODE	OPERANDS
8 14	15 20	21 62
	DC	@TAX DEDUCTABLE@
	DCW	@UNIT#6@

---

114. Constants



In a Define Constant without Word Mark statement, a Clear Word Mark instruction (CW) is placed on the card by Assembly. This instruction clears the word mark set into the area by the Load Card, since the DC statement specifies that this word mark is not desired.

82. Instructions which use a control field may require one variant character or a group of control characters. As you saw previously, a single variant character is written as 2 octal digits. Consequently an instruction with a control field of three characters will require a total of 6 octal digits. This is the maximum number of octal digits that may be written in the operands field of a CEQU instruction. Refer to the illustration on the answer side of this frame for a CEQU example.

---

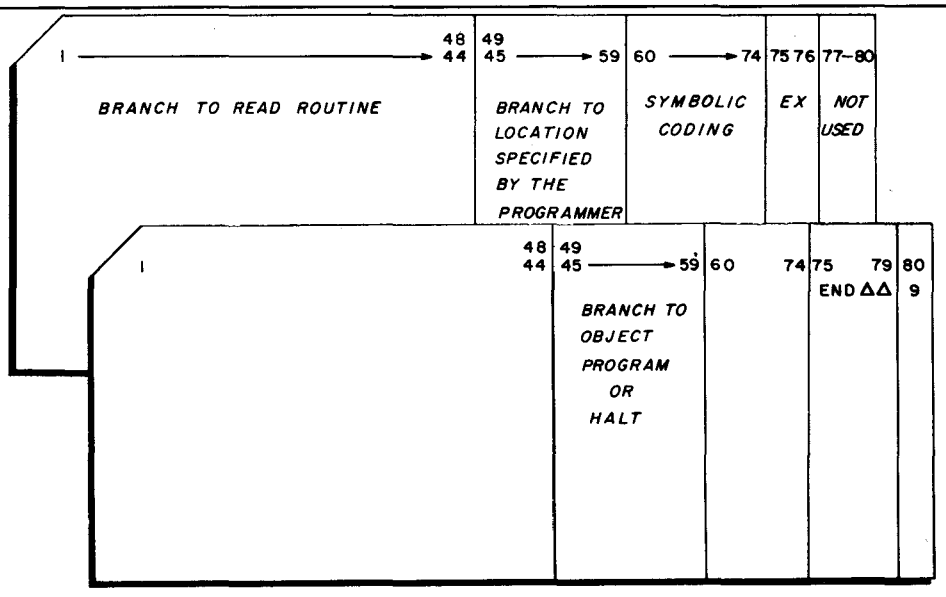
93. Addresses in the operands field of a CLEAR instruction should not be indexed or INDIRECT. However, they may be written as either absolute or symbolic addresses because these are considered to be DIRECT addresses.

---

104. If the @ symbol is desired within the constant, for example UNIT # 6@$1.20, another character not in the constant may be chosen to surround the constant. That is, any character except blank -, +, #, or the digits 0 - 9. Define UNIT # 6@$1.20 as a word marked constant.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8 ... 14	15 ... 20	21 ... 62	
		DCN	A#6@$1.20A

115.



82.

TWO (2)

SIX (6)

MARK	LOCATION	OPERATION CODE	OPERANDS
	OFLOW	CEQU	#1C05
		B	SUB2,OFLOW

The coding above illustrates a symbolic tag used in place of a variant character. CEQU directs assembly to equate the tag OFLOW of octal 05. The second line of coding contains a branch instruction. This specifies that the program should branch to location SUB2 if the condition indicated by the variant character (OFLOW) is present. Variant character 05 specifies that an arithmetic overflow condition should be tested. The coding (as an octal constant) will be explained when constants are discussed.

(RETURN TO FRAME 83, PAGE 165.)

93.

INDIRECT

DIRECT

(RETURN TO FRAME 94, PAGE 165.)

104.

NOTE: Any character except blank, -, +, #, or the digits 0-9, and not appearing in the constant, could have been chosen to surround the constant.

MARK	LOCATION	OPERATION CODE	OPERANDS
		DCW	=UNIT#6@$1.20=

(RETURN TO FRAME 105, PAGE 165.)

115.

Page 187 illustrates an efficient utilization of two and three character addressing.

Page 188 is provided for future reference regarding CAM and its variants.

Continue to Page 189.

EXAMPLE:

The following illustration shows the coding which provides entry to and exit from a subroutine to be executed in the two-character addressing mode. Both an ADMODE statement and a CAM instruction must be coded at the beginning and end of the subroutine. However, only the CAM instructions are stored in the main memory. Since CAM instructions have no address portions, the manner in which they are stored is not affected by an ADMODE statement.

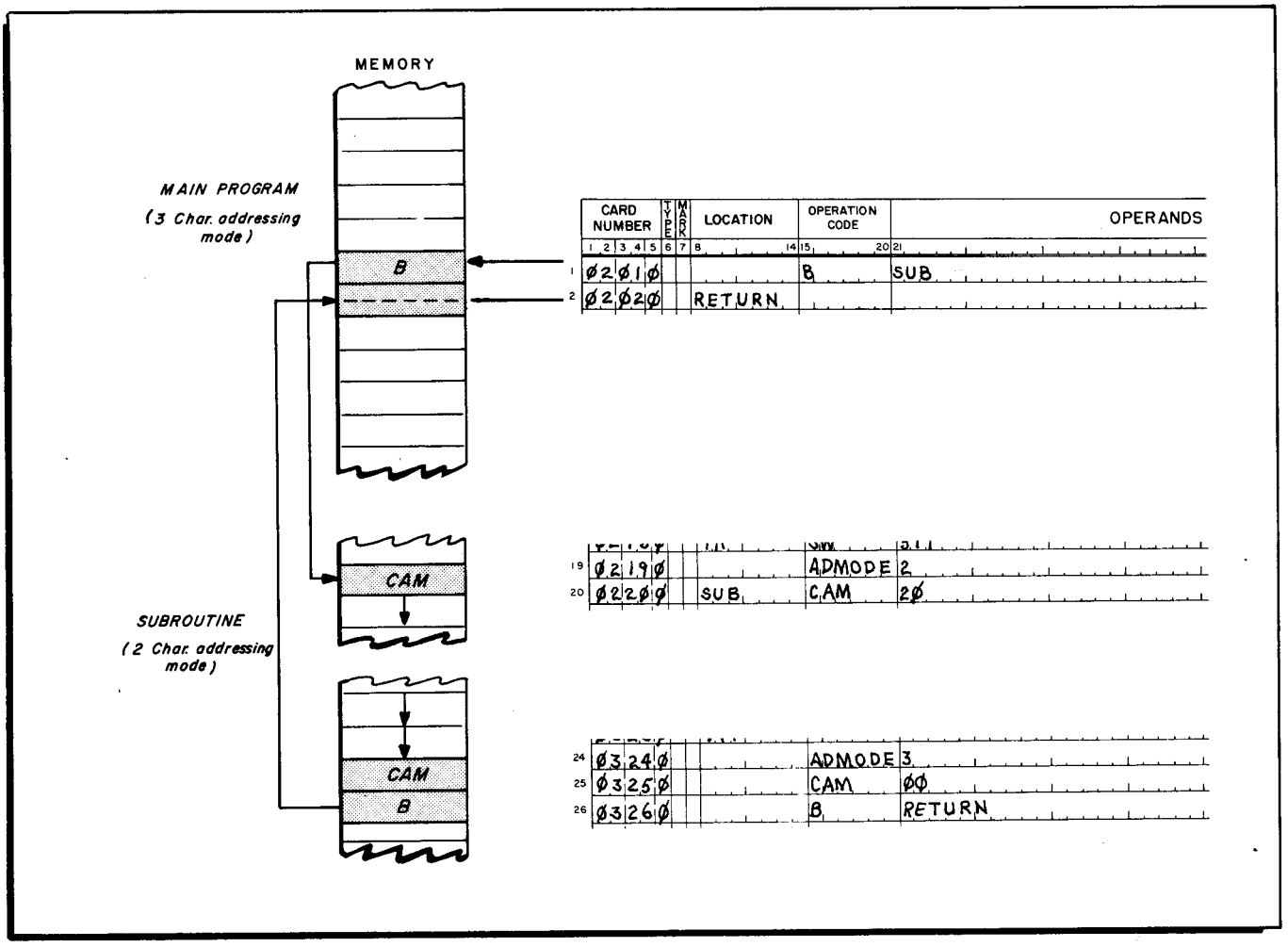


Figure 16. Two and Three Character Addressing

CAM — CHANGE ADDRESSING MODE

FORMAT

OP CODE	A-ADDRESS	B-ADDRESS	VARIANT
[redacted]			[redacted]

FUNCTION

The Change Addressing Mode instruction is used in conjunction with the ADMODE assembly control statement.

The CAM instruction directs the machine to interpret the address portions of all subsequent object program instructions as either two, three, or four-character addresses. The addressing mode is specified in the variant character of this instruction:

v = 20 for two-character addressing,
v = 00 for three-character addressing,
v = 60 for four-character addressing.

The ADMODE statement directs the Assembly Program to assemble the address portions of all subsequent source program instructions as either two-character addresses or three-character addresses.

WORD MARK: Word marks are not affected by this instruction.

TIMING: 8 microseconds.

ADDRESS REGISTERS AFTER OPERATION

I - Add. Reg.	A - Add. Reg.	B - Add. Reg.
NXT	A_P	B_P

NOTE:

1. The CAM instruction is included in the instruction repertoire of H-200 systems with a memory capacity greater than 4,096 characters or as part of an Advanced Programming option. Programs written for such systems must be coded so that the first instruction executed in the object program is a CAM instruction. As a general rule, the number of CAM instructions and ADMODE assembly directives in a program will be equal.

ASSURE THAT FRAMES 1 - 115 HAVE BEEN COMPLETED BEFORE

CONTINUING TO PAGE 189.

If the H-200 system with which you will be working does not utilize EXTENDED EASY-CODER, continue to page 190.

EXTENDED EASYCODER

Information from preceding pages applies to both Easycoder and Extended Easycoder. The capabilities of Extended Easycoder are available with larger system configurations, thereby providing utilization of literals, an additional data formatting statement, and six more assembly control instructions.

DATA FORMATTING - DEFINE AREA - DA

A specialized area within the main memory can be defined and reserved by the DA statement. The DA statement can define fields and subfields within the reserved area, and may also define two or most contiguous areas if these areas are identical in format. The programmer uses a DA statement to provide: (1) The size and name of the reserved area, (2) The number of identical areas (if more than one) which should be reserved, (3) The names, lengths, and relative positions of the fields and subfields within the reserved area(s).

ASSEMBLY CONTROL STATEMENTS

Six additional assembly control statements are available with Extended Easycoder, Some Easycoder statements have been expanded for Extended Easycoder. For example, PROG as well as the SEG statement, can identify a segment within the program; an EX statement terminates a program segment.

Segment Header - SEG - This statement defines the beginning of a portion of a program loaded into memory and executed as a unit. If a programmer does not provide segment identification, Extended Easycoder Assembly Program automatically generates SEG statements at the beginning of the program and immediately following each EX statement.

Literal Origin - LITORG - Similar to the ORG statement, the LITORG statement directs assembly to assign sequential locations to previously defined literals.

Skip - SKIP - This statement controls vertical spacing of the assembly printed program listing.

Suffix - SFX - This statement is used by the programmer principally to identify all tags in a given program segment by appending a unique single character suffix to each tag in the coding that follows.

Repeat - REP - This statement is used in conjuction with the constants DC and DCW, and it directs the Assembly Program to repeat the following constant the number of times specified in the operands field.

Generate - GEN - This statement directs assembly to repeat the following instruction a specified number of times, incrementing or decrementing operands as specified by the operands field of the GEN statement.

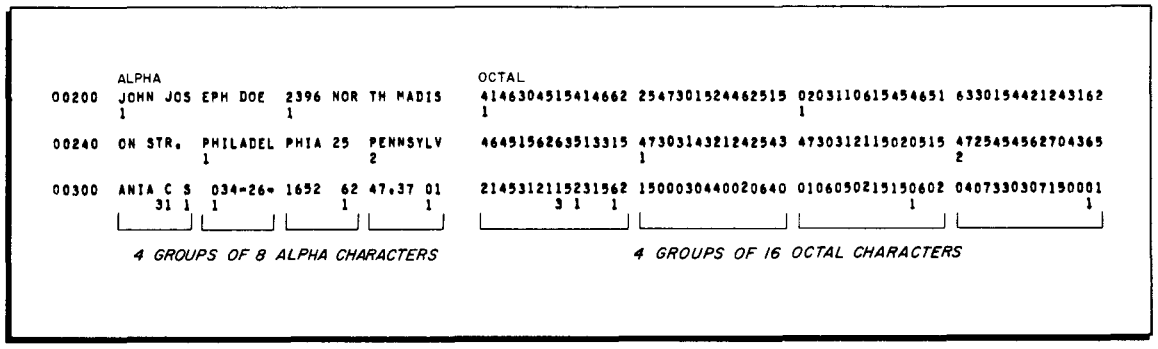
EASYCODER HIGH SPEED MEMORY DUMP ROUTINE

One of the statements which the programmer may use to direct the assembly of an Easycoder program is the Memory Dump statement - HSM. It must be coded immediately preceding the Clear and End statements in the source program. This statement directs the Assembly Program to produce a punched card deck before the object program deck is punched.

THE PROGRAMMER:

1. Writes the mnemonic code (HSM) in the operation field of the coding form.
2. May write an address (which must have been previously defined) in the location field. This address specifies the beginning location of a memory area into which the memory dump routine will be loaded. If the location field is left blank, the routine will be loaded into the area following the location assigned to the last character in the object program.
3. Writes two addresses, separated by a comma, in the operands field. These addresses specify the first and last locations of the memory area whose contents are to be listed.

The printed listing which results from the execution of the memory dump routine (the memory dump) should not be confused with the printed listing produced by the Assembly Program as part of assembly (the program listing). The memory dump is a listing of the actual contents of core memory. The program listing, on the other hand, is a listing of the object program as it is punched on the object deck.



Format of a Memory Dump

Interpreting a Memory Dump - The H-200 memory dump routine edits and prints data and punctuation bit contents of the specified memory area. The dumped output is printed, 32 memory locations per line, in both its alphanumeric and octal representation. (Thirty-two memory locations are represented by 32 alphanumeric characters plus 64 octal characters.) A code number is printed directly beneath each location which contains a punctuation bit, designating punctuation in the following manner: 1 = a word mark, 2 = an item mark, 3 = a record mark.

The leftmost four characters in each printed line represent the octal address of the first memory location whose contents are printed on that line. This is followed by the 32 alpha characters, divided into groups of eight, and then the octal representation of these 32 characters, in four groups of 16. The dump illustrated above begins at decimal location 0128 which, in octal, is memory location 0200.

ASSEMBLY PROGRAM PRINTED LISTING

A printed listing of the assembled program contains symbolic source program statements, assembled (machine-language) equivalents, and error codes. Headings are printed on the first page of the listing. The four types of statements that may appear are symbolized below:

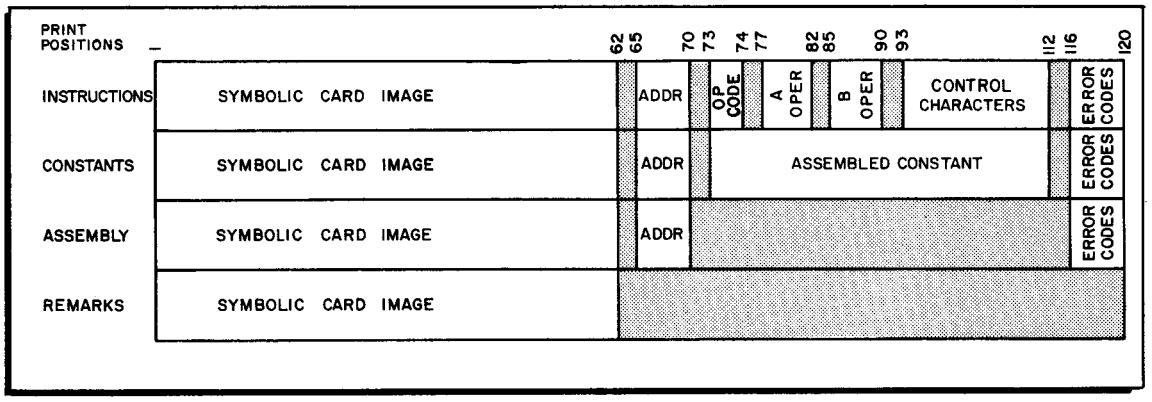


Figure 17. Program Listing Format

Instructions

1-62: The symbolic source program entry is printed within these print positions. Any statements written in these positions on the coding form, are printed in this area.

65-70: This area contains the actual memory address of the assembled instruction (the octal address of the leftmost character).

73-74: The octal representation of the op code is printed in this area.

77-82: The octal representation of the A-operand is printed in this area.

85-90: The octal representation of the B-operand is printed in this area.

93-112: This area contains the octal representation of the control characters, if any, of the instruction. Up to six control characters, separated by blanks, will be printed.

116-120: The error codes, consisting of a series of five zeros and/or numbers from 1 to 9, are printed in this area. If an error exists, a zero will be replaced by a number which denotes the following:

1 = Phase I error.
2 = Phase II error.
3 = Tag table is filled; tag was not entered.

The position in which the number is printed among the five zeros also has particular significance. If the number is printed in place of the:

First zero = error in location field. Second zero = error in op code field.
Third zero = error in A-operand field. Fourth zero = error in B-operand field.
Fifth zero = error in control character

An example of this error coding is the following: 30100 This character that a location field tag was not entered in the tag table. An error was detected during phase I in the A-operand field.

LESSON VII

EASYCODER PROGRAMMING

OP CODE		FUNCTION	TIMING (memory cycles)	DSI-214A PAGE NO.
Octal	Mnemonic			
ARITHMETIC INSTRUCTIONS				
34	BA	Binary Add	$N_i + 1 + N_w + 2N_b$	93
35	BS	Binary Subtract	$N_i + 1 + N_w + 2N_b$	94
36	A	Decimal Add	$N_i + 2 + N_w + 2N_b$ (no recomplement) $N_i + 2 + N_w + 4N_b$ (recomplement)	89
37	S	Decimal Subtract	$N_i + 2 + N_w + 2N_b$ (no recomplement) $N_i + 2 + N_w + 4N_b$ (recomplement)	91
16	ZA	•• Zero and Add	$N_i + 1 + N_w + N_b$	96
17	ZS	•• Zero and Subtract	$N_i + 1 + N_w + N_b$	97
LOGIC INSTRUCTIONS				
31	EXT	Extract (Logical Product)	$N_i + 1 + 3N_w$	100
30	HA	Half Add (Exclusive Or)	$N_i + 1 + 3N_w$	101
33	C	Compare	$N_i + 2 + N_w + N_b$	102
32	SST	Substitute	$N_i + 4$	104
55	BCE	•• Branch if Character Equal	$N_i + 4$	105
65	B	Branch	$N_i + 2$	107
65	BCT	Branch on Condition Test	$N_i + 2$	108
54	BCC	Branch on Character Condition	$N_i + 4$	111
CONTROL INSTRUCTIONS				
22	SW	Set Word Mark	$N_i + 3$	116
20	SI	Set Item Mark	$N_i + 3$	117
23	CW	Clear Word Mark	$N_i + 3$	118
21	CI	Clear Item Mark	$N_i + 3$	119
45	H	Halt	$N_i + 2$	120
40	NOP	No Operation	$N_i + 2$	121
43	CSM	•• Change Sequencing Mode	$N_i + 3$	122
42	CAM	•• Change Addressing Mode	$N_i + 2$	123
41	RNM	Resume Normal Mode	$N_i + 3$	125
14	MCW	Move Character to Word Mark	$N_i + 1 + 2N_w$	127
10	EXM	•• Extended Move	$N_i + 1 + 2N_a$	129
60	MAT	•• Move and Translate	$N_i + 3N_t$	131
15	LCA	Load Characters to A-Field Word Mark	$N_i + 1 + 2N_a$	133
24	SCR	Store Control Registers	$N_i + 5$	135
25	LCR	•• Load Control Registers	$N_i + 5$	136
EDITING				
74	MCE	• Move Characters and Edit	$N_i + 1 + N_a + 2N_b + 2X + 2Y$	140
INPUT/OUTPUT				
66	PDT	Peripheral Data Transfer	$N_i + 1 +$ data transfer time	144
64	PCB	Peripheral Control and Branch	$N_i + 1$ (no branch) $N_i + 2$ (branch)	146

• Individually optional instructions.

•• Optional instructions contained in the Advanced Programming Instructions option. In addition to the instructions listed above, this option contains the following capabilities:

1. Indexed addressing
2. Indirect addressing
3. The ability to test any variant character configuration with the Branch on Character Condition instruction
4. Read reverse capability on 204B half-inch magnetic tape units

NOTE: The Change Addressing Mode instruction (CAM) is available in systems which include either the Advanced Programming Instructions option or a memory capacity greater than 4096 characters

INTRODUCTION

This lesson presents instructions having some degree of similarity with your previous systems instructions. For example, SCR - Store Control Register has the same general purpose as SAR and SBR instructions. Of course, the H-200 may utilize or store any of its 16 control registers. This lesson also discusses the H-200 Branch instruction, and explains the versatile use of variants and alternate formats.

The following lesson introduces instructions previously outside the limits of your experience or prior equipment ability. PDT - Peripheral Data Transfer is an example of the type of instruction explained in Lesson VIII. A 1401 system performs operations serially and only one at a time. Because the H-200 provides simultaneity of operations and has multiple read/write channels, its peripheral instructions are more powerful than those to which you are accustomed. Similarly, Binary Add and Binary Subtract instructions are beyond the capabilities of a 1401 system: Consequently, they are also explained as part of Lesson VIII.

Page 194 provides an index of octal or mnemonic Op. Codes and corresponding memory cycle timing formulas. The column titled "Programmers' Reference Manual" is included for your future utilization of manual DSI 214A. The Honeywell 200 Programmers' Reference Manual is not required in order to complete either Lessons VII or VIII. Those instructions not included in Lessons VII or VIII (A, S, ZA, ZS, SW, CW, H, NOP) generally parallel those to which you are accustomed.

DECIMAL ADDITION

Add instructions perform either a true add or a complement add, depending upon the algebraic signs of the factors as shown by the zone bits (B&A cores). The zone bits in the units position of a field indicate the sign of the field.

DECIMAL SUBTRACTION

The Subtract instruction is analogous to the Add instruction with two exceptions:

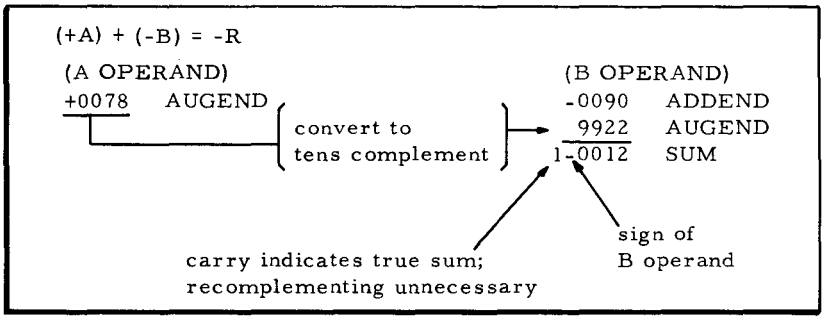
Exception 1. Before the operands are combined, the sign of the A operand is changed. Thus, if the initial sign of the A operand is equal to that of the B operand, the operands are combined by the complement add. If, on the other hand, the initial sign of the A operand is not equal to that of the B operand, the operands are combined by a true add.

Exception 2. If the sign of the A operand is negative and the sign of the B operand is positive, the sign of the result is stored in the B field with the same zero bit configuration that was originally in the B field. Otherwise, the sign of the result is "normalized".

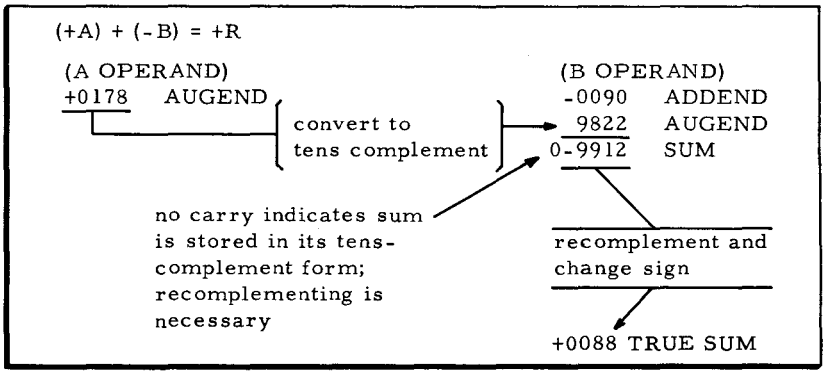
The result of any decimal arithmetic operation is stored with all zone bits, except those in the units position, set to zero. The zone bits in the units position of a field indicate the sign of the field according to the conventions shown in the table below:

A field	00, 11, 01 ⇨ +	10 ⇨ −	00, 11, 01 ⇨ +	10 ⇨ −
B field	00, 11, 01 ⇨ +	10 ⇨ −	10 ⇨ −	00, 11, 01 ⇨ +
Result	Zone bit configuration of B field	10 ⇨ −	10 → − 01 → +	10 → − 01 → +
Type of Add	True	True	Complement	Complement

Sign Convention Table



Complement Add With No Recomplementing



Complement Add With Recomplementing

INDICATORS

Two indicators are set at the completion of every decimal arithmetic operation: the overflow indicator and the zero balance indicator. If a carry is generated beyond the limit of the B field, the overflow indicator is set to "overflow"; if such a carry is not generated, the indicator is unchanged. The zero balance indicator signifies either a zero or a non-zero sum. When a decimal operation produces a result equal to zero (regardless of the sign), the zero balance indicator is set to "yes"; when the result of the operation does not equal zero, this indicator is set to "no." A Branch instruction automatically resets the overflow indicator; the zero balance indicator is not affected by the Branch instruction used to test it but is reset only by the next decimal arithmetic instruction.

1. In a preceding lesson, it was pointed out that certain capital letters separated by /'s provide a convenient method for expressing instruction formats. Remembering that the letter F means "function" and therefore, symbolizes an op. code, express the following format:

OP. CODE A ADDRESS B ADDRESS

F / A / B /

---

16. C COMPARE

		OP CODE	A ADDRESS	B ADDRESS
Format	a.	[redacted]	[redacted]	[redacted]
Format	b.	[redacted]	[redacted]	
Format	c.	[redacted]		

It is important to remember that the data in the B field is compared to an equal number of characters in the A field. The B operand word mark terminates the operation unless A contains fewer characters. In this case, the A operand must have a Word mark, because it is shorter than the B operand.

---

31. CI CLEAR ~~ITEM~~ WORD MARK

CW

		OP CODE	A ADDRESS	B ADDRESS
Format	a.	[redacted]	[redacted]	[redacted]
Format	b.	[redacted]	[redacted]	
Format	c.	[redacted]		

Format a: The locations specified by the A and B addresses are cleared of word marks. The data at these locations is undisturbed.

Format b: The word mark at the location specified by the A address is cleared. The data at this location is undisturbed.

Format c: Word marks are cleared at the locations specified by the contents of the A-and B-address registers. The data at these locations is undisturbed.

Clear the word mark at the location tagged ELEC I.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8 ... 14		15 ... 20	21 ... 62
		CW	ELEC I

---

46. Now, take a closer look at the first three bits and answer the following questions.

1. When checking for a WM ($10_8 = 001000_2$), a branch occurs if a WM is present. Would a branch occur if a RM (IM & WM) were present? _____ (yes/no)
2. When checking for an IM ($20_8 = 010000_2$), a branch occurs if an IM is present. Would a branch occur if a RM (IM & WM) were present? _____ (yes/no)
3. When checking for a RM (IM & WM) a branch occurs if a RM is present. Would a branch occur if only an IM is present? _____ (yes/no)

1.

F/A/B/

16.

B
A
A WORD MARK
B

31.

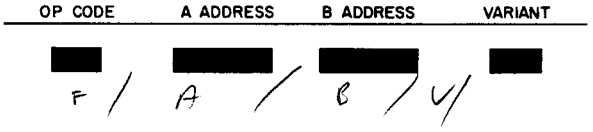
MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
		CW	ELEC1

46.

#1. YES
#2. YES
#3. NO

If any of your answers are incorrect, you can go back and find the reason some other time. Now, CONTINUE TO FRAME 47.

2. Certain instruction formats also indicate that one or more Variant characters are required. Express the following format with letters and /'s.



---

17. With a compare instruction, data characters from the B field are compared bit by bit to the same number of characters of the A field. If the A operand is longer than the B operand, the characters exceeding the word mark in B are not processed.

Three indicators may be turned on by the compare instruction. These are the:

LOW COMPARE (B<A)
EQUAL COMPARE (B = A)
HIGH COMPARE (B>A)

These indicators may be tested by a special branch instruction. The next compare resets the indicators.

---

32. In addition to the word mark, the H-200 provides for two more punctuation marks. They are the Item mark and the record mark.

A group of consecutive characters, treated as a unit, is a word. An instruction address and a word mark define the right and left boundaries respectively. An item, (one or more consecutive words) is defined by an instruction address and an Item mark.

---

47. The preceding questions may have been difficult to answer. It is sufficient to be able to answer the following:

The high order bit (leftmost) is always a zero without Adv. Prog. Instructions option. Consequently, only the bit indicated is tested and if a RM (both IM & WM) is present, a _____ will occur. In other words, if only one bit is to be tested, the presence of the non-tested bit will not prevent a branch.

2.

F/A/B/V/

---

17. NO ANSWER REQUIRED

Both fields must have exactly the same bit configurations to be equal. For example, plus zero is not equal to minus zero. (+"0" 010000, -"0" 100000)

Comparison results and associated branch conditions are listed below:

COMPARISON RESULT	CONDITION FOR BRANCH TEST
B< A	Low Compare
B = A	Equal Compare
B≤ A	Low or Equal Compare
B > A	High Compare
B ≠ A	Unequal Compare
B≥ A	High or Equal Compare

32.

ITEM

ITEM

47.

BRANCH

3. The format F/A/B/, (F/A/B/V/ when appropriate) is often referred to as, "format a." or "the complete format of an instruction".

Instructions designated as "format b." do not contain a B address. Express format b. for an instruction without a variant and an instruction with a variant. F/A/ F/A/V/

---

18. It was stated previously that arithmetic operations set two indicators (zero balance and overflow). These condition indicators are tested by a special branch instruction. Obviously, an unconditional branch instruction is not sufficient. An instruction to Branch on the Condition under TEST is required.

Appropriately, the mnemonic op. code for this instruction is BCT. The letters BCT stand for BRANCH on CONDITION TEST.

---

33. Grouping words to form an item simplifies data transfer within main memory and reduces the number of instructions needed to move consecutive words. Boundaries of the item to be transferred are specified by the programmer using the INSTRUCTION ADDRESS and an Item mark.

---

48. The proper descriptions of the following BCC variants are:

$10_8 = 001000_2$ Branch if _____ mark or _____ mark.

$20_8 = 010000_2$ Branch if _____ mark or _____ mark.

$30_8 = 011000_2$ Branch if _____ mark.

0 = Test only the bit(s) indicated	1 = Test item mark	1 = Test word mark
PUNCTUATION		

3.

Format b. F/A/
Format b. F/A/V/

18.

BRANCH CONDITION TEST

33.

INSTRUCTION ADDRESS
ITEM

48.

Without the Adv. Prog. option, a BCC may test for the three conditions above or any of nine other conditions. Zones may be tested for signs or comginations of punctuation and zones may be tested. Without Adv. Prog. option, bits V6 and V1 must be zero. Consequently, $77_8 = 111111_2$ (among fifty-four other variants) would not be valid.

$V=10_8 = 001000_2$	Branch if WM or RM
$V=20_8 = 010000_2$	Branch if IM or RM
$V=30_8 = 011000_2$	Branch if RM

4. Arithmetic instructions of the format F/A/ are said to "duplicate A". That is, the A operand is arithmetically added to itself.

In other words, saying that the format F/A/ of an ARITHMETIC instruction "DUPLICATES A" means that the A operand is doubled.

19.

BCT BRANCH ON CONDITION TEST

	OP CODE	A ADDRESS	B ADDRESS	VARIANT
Format	[redacted]	[redacted]		[redacted]

The op. code states that a branch is to occur if the condition being tested is present. Therefore, the A address specifies where the BRANCH is to go, if the condition to be tested by the VARIANT character is present.

34.

SI SET ITEM MARK

	OP CODE	A ADDRESS	B ADDRESS
Format a.	[redacted]	[redacted]	[redacted]
Format b.	[redacted]	[redacted]	
Format c.	[redacted]		

Format a: An item mark is set at the location specified by each address.

Format b: An item mark is set at the location specified by the A address.

Format c: Item marks are set at the location specified by the contents of the A and B address registers. (Chaining A and B)

Set an item mark in locations PAY, and PAY + 80. Set an item mark in location ELEC I.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
		SI	PAY,PAY+80
		SI	ELEC I

---

49. With Advanced Programming Instructions option, a BCC variant is unrestricted. That is, 00_8 to 77_8 are valid. Use the chart below and construct any variants from 00 - 77. Describe what each variant will test.

EXAMPLE: 41_8 causes a branch if NO PUNCTUATION AND B BIT IS I.

$\_\__8$ causes a branch if _____.

$\_\__8$ causes a branch if _____.

PUNCTUATION BITS			ZONE BITS		
V_6	V_5	V_4	V_3	V_2	V_1
0 = Test only the bit indicated	Item mark	Word mark	0 = Test only the bit indicated	B bit	A bit
1 = Test both bits			1 = Test both bits		

4.

ARITHMETIC
DUPLICATES

19.

BRANCH
VARIANT

34.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 14	15 20	21 62
		S,I	PAY., PAY.+80,
		S,I	ELEC 1,

49.

ANY of sixty-four variants possible for a BCC are shown by the two tables on the front and back of frame 50. Check whatever variants you constructed by referring to these tables.

NOTE 1. An X represents any octal digit. If X is 0, only the character condition described will be tested; if X is a digit from 1 to 7, the condition described and the condition indicated by the corresponding octal digit in the other table will be tested.

NOTE 2. WITHOUT ADVANCED PROGRAMMING INSTRUCTIONS OPTION - The valid BCC variants are octal: 00, 02, 06, 10, 12, 16, 20, 22, 26, 30, 32, 36.

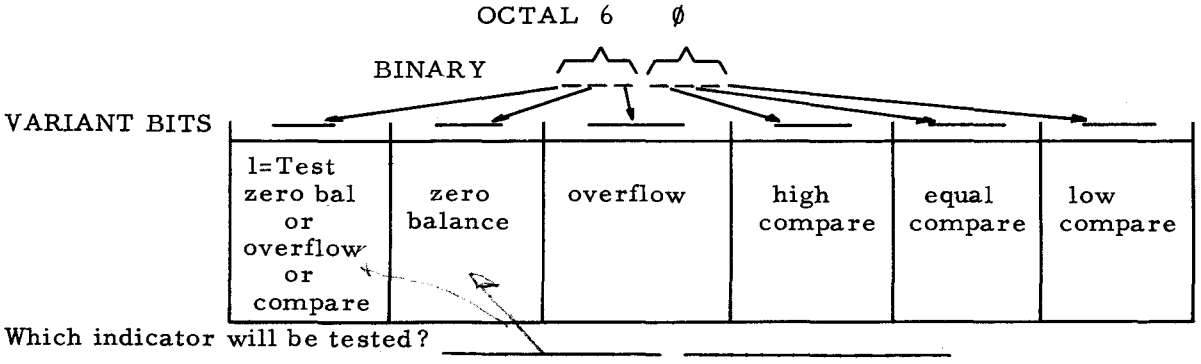
NOTE 3. The instructions constituting the Advanced Programming Instructions option are identified on the index page 194.

5. In non arithmetic instructions, the format F/A/ may indicate "half chaining". That is, the operand at the A address will be involved with the B operand whose address is currently in the B address register.

Consequently, the format F/ can indicate "full chaining". As its name implies, the A operand whose address is currently in the A ADDRESS REGISTER is involved with the B OPERAND whose address is currently in the B ADDRESS REGISTER.

---

20. Suppose that the octal variant 6Ø is written with a BCT instruction. Convert octal 6Ø to six bits and compare it to the variant table below.



Which indicator will be tested? _____

---

35. Setting item marks does not disturb the data stored in that location. However, if you set an item mark _ in a location containing a word mark O , a RECORD mark O will result. Both SW and SI instructions are required to set a O .

50.

Variant Character (octal)	Character Condition
0X	Any punctuation bit configuration.
1X	Word mark bit B character is 1 (either WM or RM present).
2X	Item mark bit of B character is 1 (either IM or RM present)
3X	The character at B contains a record mark.
4X	The character at B contains no punctuation mark.
5X	The character at B contains a word mark.
6X	The character at B contains an item mark.
7X	The character at B contains a record mark. (same as 3X).

5.

A ADDRESS REGISTER
B OPERAND
B ADDRESS REGISTER

20.

ZERO BALANCE indicator is tested by octal variant 6Ø

VARIANT BITS	1	1	0	0	0	0
	1=Test of Zero bal. or overflow or compare	zero balance	overflow	high compare	equal compare	low compare

The first 1 shows that either zero balance or overflow or compare indicator is to be tested. The second 1 shows that it is the zero balance indicator that is being tested.

35.

RECORD

50.

Variant Character (octal)	Character Condition
X0	Any zone bit configuration.
X1	The A bit of the character at B is 1.
X2	The B bit of the character at B is 1.
X3	The B and A bits of the character at B are 11.
X4	The B and A bits of the character at B are 00.
X5	The character at B contains a positive sign (the B and A bits are 01.)
X6	The character at B contains a negative sign (the B and A bits are 10).
X7	The B and A bits of the character at B are 11 (same as X3).

6. An unconditional branch instruction causes the program sequence to BRANCH from the point at which it is encountered to the single address written in the operands field.

Express the format of an unconditional branch instruction and state why that is neither "duplicating A" nor "half chaining". F/A/

Not an arithmetic instruction

"HALF CHAINING" not applied because the address specifies the address for the branch."

21. If the 1 bits show that the zero balance indicator is being tested, would this imply that a zero balance has occurred? _____

Why? N/ - [illegible] tested not result

36.

CI | **CLEAR ITEM MARK**

CI uses the same three formats as CW. A CI instruction will not disturb the data or affect word marks in locations. Clear item marks from locations PAY and PAY + 80, clear item mark from location ELEC I.

WM	LOCATION	OPERATION CODE	OPERANDS
7 8	14	15 20	21 62
		C	PAY, 3+80
		CI	ELEC I

51. Use the format F/A/B/V/, where $V=20_8$ (checking for an item or record mark) to write an instruction as follows:

Branch to address 384 if the character at 402 has the variant specified condition.

LOCATION	OPERATION CODE	OPERANDS
8 14	15 20	21 62
	B	384, 402, 20

6.

BRANCH

F/A/

UNCONDITIONAL BRANCH IS NOT AN ARITHMETIC INSTRUCTION, THEREFORE, BRANCH F/A/ DOES NOT "DUPLICATE A".

"HALF CHAINING" IS NOT IMPLIED BY BRANCH F/A/, BECAUSE THE ADDRESS SPECIFIES THE ADDRESS FOR THE BRANCH.

(or equivalent answers.)

21.

NO

THE VARIANT CHARACTER SPECIFIES WHICH CONDITION IS TO BE TESTED, NOT THE RESULT OF THE TEST.

1=Test of zero bal. or overflow or compare	zero balance	overflow	high compare (>)	equal compare (=)	low compare (<)

36.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 14	15 20	21 62
		CI	PAY,PAY+80
		CI	ELEC1

51.

LOCATION	OPERATION CODE	OPERANDS
8 14	15 20	21 62
	BCC	384,402,20

7. | B | BRANCH |

OP CODE	A ADDRESS	B ADDRESS	VARIANT

Format: [redacted] [redacted]

The op. code of an unconditional branch is the mnemonic B. This type of branch is used to interrupt program sequence and continue at another point. Word marks are not affected.

Because no specific condition is being tested, this type of branch is UNCONDITIONAL Write a branch of this type to the location tagged SUB 6.

MARK	LOCATION	OPERATION CODE	OPERANDS
		B	SUB6

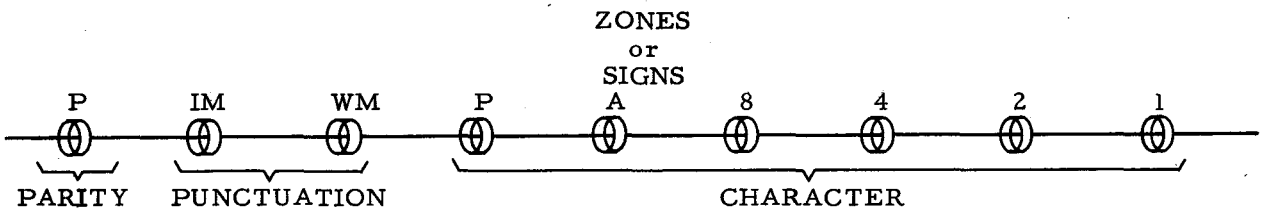
22. Refer to the chart below, and note that three compare indicators may be tested by a properly constructed variant. For example, octal 41 is binary 100001. This will test the LOW COMPARE indicator. A branch occurs if B<A construct the octal variants to:

Branch if B is = or <A 47$_8$ (100011_2)

Branch if zero balance or B>A 64$_8$

1=Test of zero bal. or overflow or compare	zero balance	overflow	high compare (>)	equal compare (=)	low compare (<)

37. Recall that the 9 cores in a memory location are in the following order:



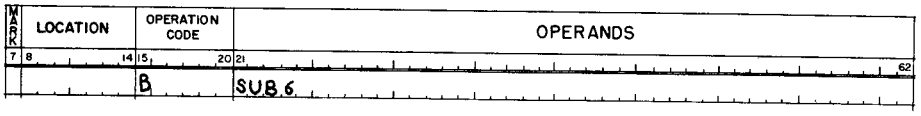
Considering only the punctuation and character cores, their corresponding bits would be 10 0 0 0 0 0 0 if an unsigned ∅ digit was in a memory location with an ITEM MARK. Write the bits for an unsigned zero with a word mark 01 000 000. Write the bits for an unsigned zero with a record mark. 11 0000.

52. In your own words, briefly state the different uses of the instructions: BRANCH, BRANCH ON CONDITION TEST, and BRANCH ON CHARACTER CONDITION.

B - unconditional
BCT - test indicators or sense switch
BBC - check for certain bits in zone bits

7.

UNCONDITIONAL



22.

LOW COMPARE - 41_8 - 100001_2 - B<A

$\underline{43}_8$ B = or < A

$\underline{64}_8$ Zero Balance or B>A

37.

WM Ø = 01 00 0000

RM Ø = 11 00 0000

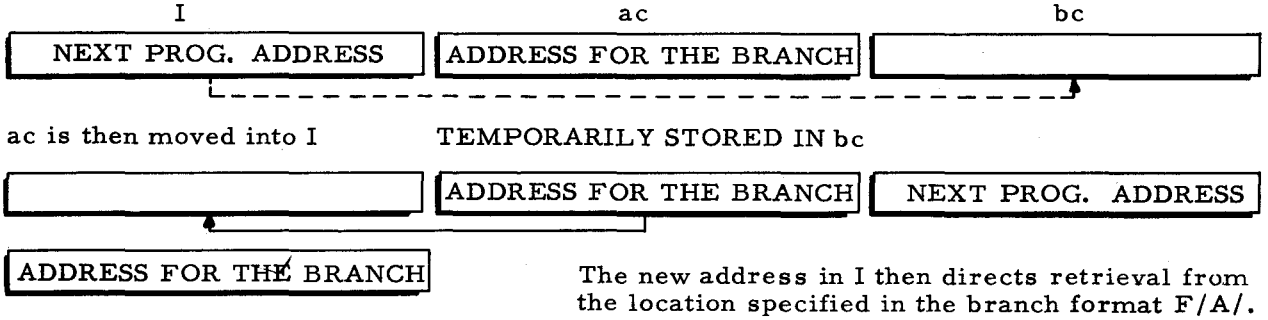
52.

B-BRANCH is an unconditional change in program sequence frequently used for subroutine linkage.

BCT - BRANCH ON CONDITION TEST is used to test the indicators or sense switches.

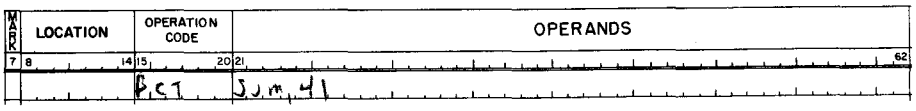
BCC - BRANCH ON CHARACTER CONDITION is used to check punctuation bits and zone bits.

8. Branches are executed within the H-200 in much less time but in a fashion similar to the 1401. The branch execution below uses the letters I, ac, bc, to identify the: Instruction, A, B, Address Registers respectively. The branch instruction format F/A/ is retrieved, then-



The new address in I then directs retrieval from the location specified in the branch format F/A/.

23. Write an instruction to branch to the location tagged SUM if B<A.



38. Perhaps the most obvious use of SW, CW, SI, and CI concerns establishment of word, item, and record limits. Another less obvious but sophisticated use of these instructions is to activate and deactivate a locations' punctuation cores as a "four-way electronic switch".

Assume a programmer wishes to set a "switch" by program instructions instead of manually pressing a sense switch. Perhaps he wishes to indicate a particular routine has been executed or a certain condition has been encountered in the program. He may turn on an "electronic switch" (punctuation bits in a selected location) with a SI or SW instruction.

53.

The final branch instruction to be discussed in this lesson is used to check for equal characters. That is, a branch will occur to the A address if the single character at the B address is the same as the variant character. The Branch if Character Equal instruction does not require construction of specific variant bits. The variant is simply a character to be compared to the B address character.

Formats of this instruction are shown on the answer side of this frame.

8.

NO ANSWER REQUIRED

23.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14	15 20	21 62
		BCT	SUM,4,1

38.

NO ANSWER REQUIRED

53.

BCE — **BRANCH IF CHARACTER EQUAL**

		OP CODE	A ADDRESS	B ADDRESS	VARIANT
Format	a.	[redacted]	[redacted]	[redacted]	[redacted]
Format	b.	[redacted]			

Format a: The single character specified by the B address is compared to the variant character. If the bit configurations of the two characters are equal, the program branches to the location specified by the A address. If the bit configurations are unequal, the program continues in sequence.

Format b: Format b of this instruction is an illegal format unless it is immediately preced by a BCE instruction which did not cause a branch. The single character specified by the contents of the B-address register is compared to a variant character specified in the previous BCE instruction. If the bit configurations of both characters are equal, the program branches to the instruction specified by the contents of the A-address register.

Suppose that the instruction B SUB 6 is stored in memory locations 150, 151, 152, and that SUB 6 refers to the routine beginning in memory location 500. Write the appropriate addresses in the registers below. The branch instruction format F/A/ is retrieved, then-

	I	ac	bc
1.	153	500	
2. ac is moved into I.		500	153
3.	500		153

Refer to the chart in frame 22 as needed to write the following:

Compare Item Number to 4000. If Item Number is equal to 4000, branch to location NITEM.

Description	Tag
Item number	ITEM
4000	CON4

MARK	LOCATION	OPERATION CODE	OPERANDS
		C	CON4, ITEM
		BC 7	NITEM, 47

An "electronic switch" is simply the PUNCTUATION cores of a selected MEMORY LOCATION. The switch may be "turned on" by a SW instruction or a SI instruction. Since it is "turned on" by these instructions, it may be "turned off" by CW and CI instructions. Show the four possible binary conditions of an electronic switch.

IM WM

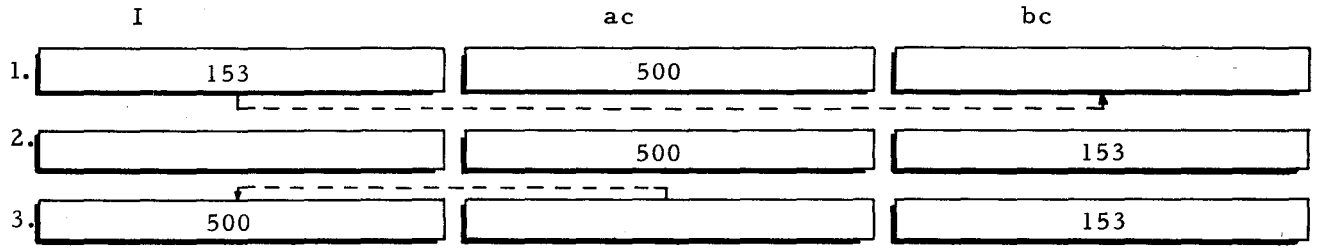
0 0, 0 1, 1 0, 1 1.

A word mark in the location tested has no effect on this instruction.

Determine if the character stored in the location tagged LABEL + 3 is equal to 6. If so, branch to the location tagged P6; otherwise continue the program in sequence.

MARK	LOCATION	OPERATION CODE	OPERANDS
		BCE	P6, LABEL+3, 6

9.



I now contains the address to begin retrieval of SUB6 at location #500. The address to which the program should return after completing SUB6 is temporarily stored in bc.

24.

MARK	LOCATION	OPERATION CODE	OPERANDS
		C	CON4,ITEM
		BCT	NITEM,42

NOTE: 42_8 is $1000\ 10_2$ Tests EQUAL

39.

PUNCTUATION MEMORY LOCATION SW, SI CW CI

	IM	WM	
	0	0	= Both switches "off".
	1	0	= IM "on", WM "off".
	0	1	= IM "off", WM "on".
Record Mark	1	1	= IM "on", WM "on".

These four conditions may be tested by a Branch on Character Condition instruction.

54.

MARK	LOCATION	OPERATION CODE	OPERANDS
		BCE	P6,LABEL+3,6

10. With a 1401 at the point illustrated in frame 9, it would be necessary to write an SBR instruction at the start of the subroutine. This would store bc so that a branch could be written at the end of the subroutine for returning to program sequence (153).

Similar instructions are written for the H-200. However, bc is an H-200 CONTROL REGISTER. To accomplish what you know as SBR, an H-200 STORAGE CONTROL REGISTER instruction is written.

25. If the BCT octal variant has a ∅ as the first digit, (EXAMPLE ∅4), why will none of the indicators be tested? Zero bit must be 1 to test

____ 1 = Test of	____	____	____	____	____
zero bal. or overflow or compare					

40. The initial designation or selection of an electronic switch is accomplished with a data formatting statement and a ∅ in column 21.

Write a define constant instruction to reserve one memory location. Tag it ELEC1.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
	ELEC1	DC	0

55. Determine if any character position in the seven-character field tagged PART contains the letter Q. If so, branch to the location tagged RETRO; otherwise continue the program in sequence.

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
	1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62
1					BCE	RETRO, PART, Q
2					BCE	
3					BCE	
4					BCE	
5					BCE	
6					BCE	
7					BCE	

10.

STORE CONTROL REGISTER

An example of SCR coding is shown below:

MARK	LOCATION	OPERATION CODE	OPERANDS
		B	SUB6
		S	TAX,PAY NEXT INSTRUCTION
	SUB6	CAM	2Ø , START SUBROUTINE
	RETURN	S,CR	3ØØ,7Ø
		≷	
		≷	
		B	3ØØ END OF SUBROUTINE

25.

THE FIRST BIT MUST BE 1, IF ZERO BALANCE OR OVERFLOW OR COMPARE INDICATORS ARE TO BE TESTED

Therefore, a BCT octal variant producing a first bit of 0 is meaningless AS FAR AS INDICATORS ARE CONCERNED.

40.

MARK	LOCATION	OPERATION CODE	OPERANDS
	ELEC1	DC	Ø

55.

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1					BCE	RETRO, PART, 9
2					BCE	
3					BCE	
4					BCE	
5					BCE	
6					BCE	
7					BCE	

11.

	OP CODE	A ADDRESS	B ADDRESS	VARIANT
Format	[redacted]	[redacted]		[redacted]

SCR stores the control register designated by the octal variant character at the address written in the operands field.

In the example at the left (frame 10) evidently the variant character 7 0 designates bc. This control register (containing the address of the return point after the subroutine) is to be stored in memory at address # 800.

SCR	STORE CONTROL REGISTERS

---

26. The testing of individual or combinations of SENSE SWITCHES occurs when a BCT octal variant has a first digit that will produce a first bit zero. In other words, if the first bit of a BCT variant character is not a 1, SENSE switches are to be tested - not ZERO BALANCE, OVERFLO, or COMPARE indicators.

---

41. If it is desired to initially turn on the Item Mark switch, an L could be written in the MARK column of the DC statement. Similarly, if a Word Mark switch were desired to initially be on, a DCW statement could be written.

Write the statement to select an electronic switch tagged ELEC 2 that is initially have both IM and WM on.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
L	ELEC2	DCW	0

---

56. H-200 MCW and LCA instructions are used in much the same manner as their 1401 counterparts. The difference between LCA and MCW is that:

LCA terminates transfer with the word marked A operand character.

MCW terminates transfer when the first word mark in either A or B is reached.

11. 70 - Octal variant designating bc.

800 - bc to be stored at address 800 in example.

The partial tables below and in frame 12 list the control registers designated by SCR variants.

Variant Character (octal)	Control Register
67	A-Address Register
70	B-Address Register
77	Instruction Address Register 1
64	Instruction Address Register 2
01	RWC 1 - Current Location Counter
11	RWC 1 - Starting Location Counter
02	RWC 2 - Current Location Counter
12	RWC 2 - Starting Location Counter

26.

SENSE
ZERO BALANCE
OVERFLOW
COMPARE

41.

WAR	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
	LELEC2	DCW	Ø

56.

NO ANSWER REQUIRED

12.

Variant Character (octal)	Control Register
03	RWC 3 - Current Location Counter
13	RWC 3 - Starting Location Counter
05	RWC 1' - Current Location Counter
15	RWC 1' - Starting Location Counter
66	Interrupt Register
67	Word Register 1
76	Work Register 2
60	Unassigned

Continue to the answer side of this frame.

27. Understanding the purpose of the first variant bit permits alternate utilization of the chart below:

1 = Test of zero bal. or overflow or compare		Sense Switch 4	Sense Switch 3	Sense Switch 2	Sense Switch 1

Construct a BCT variant to test sense switch 4 for on. $0\,0\,1\,0\,0\,0_2$ $1\,0_8$

42. An electronic switch occupies a memory location and as such, may be thought of as a "character". The purpose of one of these "characters" is to provide four possible conditions that may be set by program instructions and then be checked by a special branch instruction. Since this Branch is determined by a Character Condition, it is known as a Branch on CHARACTER CONDITION instruction.

57. The following statements concern an LCA instruction:

1. This instruction (in any format) is the only instruction that always moves both a field and its defining punctuation mark.
2. A record mark appearing in the A field will terminate the operation.
3. All punctuation (word marks, item marks and record marks) initially stored in B-field locations will be cleared if the corresponding A field characters do not include identical punctuation.
4. The B address must never fall within the A field. The A address may fall within the B field, however, if desired.

Continue to the answer side of this frame.

12.

Any of the 16 control memory registers may be stored in memory by SCR and the appropriate variant.

Because it may be desirable to load any of these 16 control memory registers, a Load Control Register (LCR) instruction is available.

27.

$$\underline{001000}_2 = \underline{10}_8$$

The variant above tests sense switch 4.

42.

CHARACTER CONDITION

57.

	OP CODE	A ADDRESS	B ADDRESS
a.	[redacted]	[redacted]	[redacted]
b.	[redacted]	[redacted]	
c.	[redacted]		

LCA	LOAD CHARACTERS TO A-FIELD WORD MARK

Format a: The data and punctuation in the A field are transferred to the B field.

Format b: The data and punctuation in the A field are transferred to the field specified by the contents of the B-address register.

Format c: The data and punctuation in the field specified by the contents of the A-address register are transferred to the field specified by the contents of the B-address register.

13. **LCR** LOAD CONTROL REGISTERS

	OP CODE	A ADDRESS	B ADDRESS	VARIANT
Format	[redacted]	[redacted]		[redacted]

LCR variant characters which designate control registers are the same as those listed for SCR. The contents of the field specified by the A address (containing either a two, three, or four character address depending on the present addressing mode) are loaded into the control register designated by the variant.

Refer to the preceding tables as needed to write:

1. An SCR for RWC - 1 Current Location Counter to be stored at the address tagged CLC I.
2. An LCR for Instruction Address Register 2 to be loaded from the address.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62

28.

When testing for a multiple sense switch condition with a single BCT, a branch occurs only if all the designated switches are on. When testing multiple indicators with a single BCT, the branch occurs if any of the indicators shows the desired condition. *

A complete BCT variant chart is provided on the answer side of this frame. Complete tables of all variant combinations in the Programmers' Reference Manual.

43. **BCC** BRANCH ON CHARACTER CONDITION

	OP CODE	A ADDRESS	B ADDRESS	VARIANT
Format a.	[redacted]	[redacted]	[redacted]	[redacted]
Format b.	[redacted]			

Format a: The single character specified by the B address is examined for the condition specified by the variant character. If the condition is present, the program branches to the instruction specified by the A address. If the condition is not present, the program continues in sequence.

Format b: The single character specified by the contents of the B-address register is examined for the condition specified by the variant character in the previous BCC instruction. If the condition is present, the program branches to the instruction specified by the contents of the A-address register. Otherwise the program continues in sequence.

Note that format b. chains the A and B addresses, but it retains the VARIANT CHARACTER of the PREVIOUS BCC instruction

58. Set punctuation defining the proper field, then move TAX to PAY. The rightmost memory location of the four character A operand is tagged TAX. The rightmost memory location of the four character B operand is tagged PAY.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
		SW	TAX-3
		LCR	TAY,PAY

13.

MARK	LOCATION	OPERATION CODE	OPERANDS
		SCR	CLC1,Ø1
		LCR	COSEQ,77

SCR instructions may also be used for determining the length of records. Assume that it is desired to know the length of a record that has been transferred from the tape unit on RWC #2.

By storing the starting location counter and the current location counter of RWC #2, their contents could then be arithmetically subtracted. The difference between these amounts equals the length of the record transferred.

28.

NO ANSWER REQUIRED

Refer to this chart for the next frame

V_6	V_5	V_4	V_3	V_2	V_1
0 = Test Sense Switch	Not Used	Sense Switch 4	Sense Switch 3	Sense Switch 2	Sense Switch 1
1 = Test Zero Balance, Overflow or Compare	Zero Balance	Overflow	High Compare	Equal Compare	Low Compare

---

43.

Format b. chains A and B addresses.

It also retains the VARIANT CHARACTER of the PREVIOUS BCC.

58.

MARK	LOCATION	OPERATION CODE	OPERANDS
		SW	TAX-3,
		LCA	TAX, PAY

14. Mnemonics for the three instructions covered to this point are: B, SCR, LCR.
Their respective formats are: F F/A/N F/A/N
A control register is designated by an octal VARIANT CHARACTER.

---

29. A multiple sense switch variant tests all designated switches for on, and the program will not branch until all designated switches are set. Conversely, a multiple indicator variant causes a branch if any designated indicator is set. The following example conditions may require more than one BCT instruction. Construct appropriate BCT variants to branch if:

#1. Sense switches 1, 2, and 4 are on 13_8
#2. Sense switch 1 or 2, or 4 is on $01_8, 02_8, 10_8$
#3. Overflow or B≠ (UNEQUAL) A 55_8
#4. Overflow or zero balance or B A 71_8

How may BCT instructions are needed for each of the above?
#1. 1, #2. 3, #3. 1, #4 1.

---

44. Only a single punctuation core (WM) is available in a 1401 memory location. When used as an electronic switch it could only be checked with one BWZ /A/B/1^d (d character 1 causes branch if WM). Additionally, a 1401 BWZ for eight other conditions (word mark or zeros, zone checks) making a total of nine possible tests. Without Advanced Programming Instructions option, the H-200 may check 12 character conditions. With the option, the H-200 may check 64 character conditions.

---

59. Formats of an MCW instruction are illustrated on the answer side of this frame.

14.

B SCR LCR
F/A/ F/A/V F/A/V

VARIANT CHARACTER

29. If any of the below are incorrect, review frames 28 and 29.

#1. Sense switches 1, 2, and 4 are on. 13_8 (001011_2)

#2. Sense switches 1 or 2 or 4 is on. 01_8, 02_8, 10_8

#3. Overflow or B≠A. 55_8 (101101_2) see note.

NOTE: Logically, if B is either HIGH or LOW compared to A, then B could not be equal to A (B≠A). Unequal tests are made by testing condition of HIGH COMPARE and LOW COMPARE indicators.

#4. Overflow or zero balance or B<A. 71_8 (111001_2)

#1. ONE #2. THREE #3. ONE #4. ONE

44.

IF THE H-200 WITH WHICH YOU WILL BE WORKING HAS THE ✓ ADVANCED PROGRAMMING INSTRUCTIONS OPTION, SKIP TO FRAME 49, PAGE 203; OTHERWISE, CONTINUE TO FRAME 45.

0=Test only the bit (s) indicated	1=Test Item Mark	1=Test Word Mark	0=Test only bit indicated 1=Test both bits	B bit	A bit
PUNCTUATION			ZONES		

59. MCW — MOVE CHARACTERS TO WORD MARK

	OP CODE	A ADDRESS	B ADDRESS
Format a.	[redacted]	[redacted]	[redacted]
Format b.	[redacted]	[redacted]	
Format c.	[redacted]		

Format a: The data and item marks in the A field are moved to the B field.

Format b: The data and item marks in the A field are moved to the field specified by the contents of the B-address register.

Format c: The data and item marks in the field specified by the contents of the A-address register are moved to the field specified by the contents of the B-address register.

15. The instruction used to compare B field data to the same number of characters in the A field is known as the COMPARE instruction. This instruction has three formats. The first format is "complete" and does not include a variant. The second format provides "half chaining" and the third format allows "full chaining".

Show and identify these formats.

Format a. F/A/B/ is COMPLETE.

Format b. F/A provides HALF CHAINING.

Format c. F/ allows FULL CHAINING.

---

30. **SW** SET WORD MARK

	OP CODE	A ADDRESS	B ADDRESS
Format a.	[redacted]	[redacted]	[redacted]
Format b.	[redacted]	[redacted]	
Format c.	[redacted]		

Format a: A word mark is set at the location specified by each address. The data at each location is undisturbed.

Format b: A word mark is set at the location specified by the A address. The data at this location is undisturbed.

Format c: Word marks are set at the locations specified by the contents of the A- and B-address registers. The data at each location is undisturbed.

Set a word mark in the location tagged ELEC I.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14	15 20	21 62
		SW	ELEC I

---

45. The first discussion of BCC variants assumes a system without advanced programming instructions option. For the moment, only punctuation checks are considered. Refer to the chart in frame 44, page 224. Then, construct the proper variant to write octal variants for the following:

Check for a WM ___ ___ ___ 000 $_2$ = ___ 0 $_8$

Check for a IM ___ ___ ___ 000 $_2$ = ___ 0 $_8$

Check for a RM ___ ___ ___ 000 $_2$ = ___ 0 $_8$

---

60. Formats a, b, and c: A word mark is required in the shorter of the two fields. The operation terminates when this word mark is sensed. Item marks initially stored in B-field locations will be cleared if the corresponding A-field characters do not include item marks

Assume that the B fields below are unpunctuated and that each A field contains a WM in its leftmost location. Move the A fields to the sequential B fields by the appropriate MCW instruction format.

A FIELD	B FIELD
150 - 155	800 - 805
160 - 168	806 - 814
173 - 180	815 - 822
185 - 187	823 - 825

MARK	LOCATION	OPERATION CODE	
7 8	14	15 20	21
		MCW	187,825
		MCW	180
		MCW	160
		MCW	155

15.

COMPARE (mnemonic C)

F/A/B/	COMPLETE
F/A/	HALF CHAINING
F/	FULL CHAINING

30.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8 14		15 20	21 62
		SW	ELEC1

45.

CHECK WM V = 001000_2 = 10_8

CHECK IM V = 010000_2 = 20_8

CHECK RM V = 011000_2 = 30_8

60.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8 14		15 20	21 62
		MCW	187,825
		MCW	180
		MCW	168
		MCW	155

• MOVE CHARACTERS AND EDIT

(If the H-200 system with which you will be working does not include an MCE option, continue to LESSON VIII.)

MCE	MOVE CHARACTERS AND EDIT

	OP CODE	A ADDRESS	B ADDRESS
Format	[redacted]	[redacted]	[redacted]

FUNCTION

MCE is used to insert identifying symbols and punctuation, also for suppression of unwanted zeros in a data field. The A address contains the information to be edited. The B address contains a control word providing a framework for the edit operation. When an MCE is executed, the data in A is moved to B. There it is punctuated and formatted according to the edit control word already in B.

An LCA instruction can be used to load the control word into B. For instance, if edited information is to be printed, the control word should be loaded into the print image area. The address of this area should be used as the B address of the MCW instruction.

Editing is performed according to the following rules:

Rule 1. Any character in the H-200 character set can be used in the edit control word. Characters having special meanings are listed below. All other characters, if included in the edit control word, remain in the edited result in the position where written.

Rule 2. A word mark in the high-order position of B controls the edit operation.

Rule 3. The number of replaceable characters in the edit control word must be at least as large as the number of characters in A.

Rule 4. Data is transferred from A, character by character, from right to left. If a zero suppression symbol is not sensed in the edit control word, the edit operation terminates when the B word mark is sensed. A zero suppression symbol causes the edited result field to be scanned from left to right. During this scan, high-order zeros and commas are automatically replaced by blanks (unless an asterisk appears immediately to the left of the zero suppression symbol -- see rule 5). Zero suppression is terminated by any of the following:

a. a decimal digit from one through 9,
b. a decimal point, or
c. the location that initially contained the zero suppression symbol.

Rule 5. An asterisk immediately to the left of the zero suppression symbol in the control word causes high-order zeros and commas to be replaced by asterisks instead of blanks in a zero suppression operation. High-order blanks are also replaced by asterisks.

Rule 6. A dollar sign immediately to the left of the zero suppression symbol in the control word is replaced with an A-field character causing the edited result to be rescanned following zero suppression. During this scan, the dollar sign is "floated" to the left of the high-order significant digit in the edited result.

Note 1. Zone Bits in the units position of A are cleared to zero when moved to B. Therefore, the value of the character in the units position of A may change when moved to B. For example, an F in the units position of A will appear as a 6 in the result.

Note 2. Floating dollar sign insertion and automatic asterisk insertion cannot be performed in the same edit operation.

CONTROL CHARACTER	FUNCTION
b (blank)	Blanks are replaced with A-field characters such that the rightmost character in the A field replaces the rightmost blank in the edit control word and all higher-order A-field characters replace successively higher-order blanks.
0 (zero)	This symbol specifies zero suppression. Its location in the control word is interpreted as the rightmost limit of zero suppression. It is replaced with an A-field character.
. (decimal point)	The decimal point remains in the edited field in the position where written.
, (comma)	Commas remain in the edited field where written unless zero suppression is specified (see rule 4). Commas in control word positions to the left of the high-order character transferred from the A field are replaced by blanks.
C_R, CR (credit) $\bar{0}$ (minus) Note: $\bar{0}$ is printed as a minus symbol.	The credit or minus symbol is undisturbed if the sign in the units position of the A field is negative. If the sign is positive, the credit (or minus) symbol is blanked out. A credit (or minus) symbol transferred from the A field is not subject to sign control.
& (ampersand)	The ampersand is replaced by a blank in the edited field.
* (asterisk)	The asterisk remains in the edited field in the position where written unless it appears immediately to the left of the zero suppression symbol (see rule 5).
$ (dollar sign)	The dollar sign remains in the edited field in the position where written unless it appears immediately to the left of the zero suppression symbol (see rule 6).

WORD MARKS

Both the A field and the B field must have defining word marks. The A-field word mark terminates the transfer of data from the A field. The B field word mark terminates the edit operation if no zero suppression symbol is sensed in the edit control word or if automatic dollar sign insertion is specified in conjunction with zero suppression. The B-field word mark is erased after terminating the edit.

If zero suppression is specified, a word mark is automatically set in the location containing the zero suppression symbol. When this word mark is sensed during the reverse scan associated with the zero suppression operation it is erased, and if automatic dollar sign insertion is not called for, the edit operation terminates.

Write the result of the edits in PROBLEMS 1 - 4 below: (refer to preceding pages as needed)

Example:

Data Field (A Field)	⓪000099
Control Word (B Field)	ⓑbb,bb0.bb&&0
Result of Edit	:99 –

PROBLEM 1.

Data Field (A Field)	②5454986
Control Word (B Field)	ⓑbb&bb&bbb
Result of Edit	254 54 986

PROBLEM 2.

Data Field (A Field)	⓪00450+
Control Word (B Field)	Ⓢb,bb0.bb&CR*
Result of Edit	$4.50 *

PROBLEM 3.

Data Field (A Field)	⓪0897445
Control Word (B Field)	ⓑbbb,b$0.bb
Result of Edit	$8,974.45

PROBLEM 4.

Data Field (A Field)	⓪010450
Control Word (B Field)	ⓑb,b*0.bb
Result of Edit	***104.50

Result of Edits:

PROBLEM 4. ⓪010450 / ⓑb,b*0.bb / ***104.50

PROBLEM 3. ⓪0897445 / ⓑbbb,b$0.bb / $8,974.45

PROBLEM 2 ⓪00450+ / Ⓢb,bb0.bb&CR* / $ 4.50 *

PROBLEM 1. ②5454986 / ⓑbb&bb&bbb / 254 54 986

LESSON VIII

EASYCODER PROGRAMMING

1. 1401 memory size is limited and its "coded" addressing may be awkward to modify (000 to 999, then ≠00 to z99 . . . !0? to R91 etc.).

As you have seen for the H-200, two memory locations with 12 bits may address up to #4096. Then, 18 binary digits to #32000, 24 binary digits to #65000, etc. permits readily expandable memory. The H-200 addressing system is more flexible because it is based on BINARY digits.

---

20. List the word mark conventions for the following BA formats:

F/A/B/ B, A if smaller

F/A/ A

F/ B, A if smaller

---

39. Possibly the S op. code (Octal 37, 011111_2) is to become NOP (Octal 40, 100000_2) later in the program. What binary value would be required as a constant to accomplish this change with a HA? 111111

Show the operands and arithmetically perform HA.

B OPERAND = 011111

A OPERAND = 111111

HA SUM = 100000

---

58. CARDIN

721	722	723	724	725	726	727	728	729	730
J	D	O	E			H	4	9	

Write the constant and the instruction to extract the complete character H, passing the numeric portions but blocking the zones of characters 4 and 9.

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
	1 2 3 4 5	6	7	8 14	15 20	21 62
1				CARDIN	RESV	80
2				BFLD	RESV	40
3					≡	
4				MASK	DC W	#3C7171717
5					≡	
6					LCA	MASK,BFLD
7					EXT	CARINT8,BFLD

1.

BINARY

20.

F/A/B/ WORD MARK THE B OPERAND. ALSO, WORD MARK THE A OPERAND IF A IS SHORTER THAN B.

F/A/ WORD MARK THE A OPERAND.

F/ SAME AS F/A/B/.

39.

111111_2 (77_8)

B OPERAND = 0 1 1 1 1 1

A OPERAND = 1 1 1 1 1 1

HA SUM = 1 0 0 0 0 0 = 40_8 = NOP

58. MASK is defined with DCW providing a word mark in the shorter operand of EXT.

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1			CARDIN	RESV	80
2			BFLD	RESV	40
3				3	
4			MASK	DCW	#3C77171717
5				3	
6				LCA	MASK,BFLD
7				EXT	CARDIN+8,BFLD

Note: The EXT A address is relative (CARDIN + 8). This could have been written as an absolute address (720).

2. Without indexing, H-200 address modification may be accomplished through an arithmetic capability not available in your previous equipment. Because its addresses are binary, the H-200 has a BINARY ARITHMETIC capability. Operations of this type may be used to MODIFY an address as well as to accomplish other programming applications.

---

21. You should recall that the opposite of binary addition is BINARY SUBTRACTION. This follows the rules: 0 + 0 = 0, 1 + 0 = 1, 0 + 1 = 1, 1 + 1 = 0 with a carry of 1.

However, producing the complement of a binary number is called COMPLEMENTATION and re-adding a high order carry is called END AROUND CARRY.

---

40. Half Add A operands containing all 1 bits will produce the complement of any binary value. As an example of this, perform the HA below remembering that carries are not propagated and a WM in the shorter operand terminates the operation.

		WM		
B OPERAND		100101	000111	110010
A OPERAND	111111	111111	111111	111111

---

59. HA instructions change operands but do not propagate a carry. SST instructions can move or not move bits of a single character. An EXT instruction can move or not move character bits or groups of characters.

HA, SST, EXT, deal with six of nine cores in a memory location. They affect any of the six character bits but do not directly affect punctuation or parity.

2.

BINARY ARITHMETIC
MODIFY

21.

BINARY SUBTRACTION
COMPLEMENTATION
END AROUND CARRY

40.

		WM		
B OPERAND		100101	000111	110010
A OPERAND	111111	111111	111111	111111
WORD MARK TERMINATES		0 1 10 10	1 1 1000	00 1 10 1

Notice that the HA sum is the complement of the B operand.

59.

NO ANSWER REQUIRED

3. **BA** BINARY ADD

Unlike your previous equipment, the Honeywell 200 can perform binary as well as DECIMAL arithmetic operations. Binary Add instructions (mnemonic BA) may be written in the three formats a, b, or c. Show each of these formats.

Format a. F/ A/ C/

Format b. F/ A/

Format c. F/

---

22. **BS** BINARY SUBTRACT

The same formats and word mark conventions you learned with BA are applicable in Binary Subtract (mnemonic BS) operations. Before discussing computer performance of complementation and end around carry, see if you recall how to do binary subtraction.

Binary subtract the subtrahend 000010 from the minuend 101001.

111101
100111

---

41. Mnemonic SST represents the SUBSTITUTE logic instruction. Your previous system was able to move a characters' numerical or zone bits; the H-200 with its SST instruction can:

1. MOVE NUMERICAL BITS OF A CHARACTER.
2. MOVE ZONE BITS OF A CHARACTER.
3. MOVE OR NOT MOVE ANY CHARACTER BIT EXCEPT PUNCTUATION AND PARITY.

---

60. HA, EXT, and SST are considered to be logic instructions. Another instruction provides greater control and is extended in function to move punctuation and data bits. The mnemonic op. code of this instruction is EXM, which stands for EXTENDED MOVE. EXM is considered to be a CONTROL instruction rather than a logic instruction.

3.

DECIMAL

F/A/B/
F/A/
F/

22.

The SUBTRAHEND (number to be subtracted) 000010 is complemented:

111101, and then added to the minuend.

```
                c c c   c
                1 0 1 0 0 1
COMPLEMENTED SUBTRAHEND =
                1 1 1 1 0 1
END AROUND CARRY
              1 1 0 0 1 1 0
                        1
              -------------
              1 0 0 1 1 1
```

41.

SUBSTITUTE

60.

EXTENDED MOVE
CONTROL

4. The H-200 follows binary addition rules presented in Lesson IV.
That is, 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, and 1 + 1 = 0 with a "carry" of 1.

---

23. Binary Subtraction requires an end around carry. Since this "1" bit is always to be added, the computer automatically inserts a 1 bit in the ADDER at the start of BS.
In format F/A/B/, the subtrahend is the A OPERAND, consequently the minuend is the B OPERAND.

---

42. SST SUBSTITUTE

An H-200 SST instruction moves or does not move any of the character bits stored at the A address to the B address. The move is specified by the programmer according to the variant he constructs.

Indicate the format of the SST instruction described above.

F/A/B/V

61.

OP CODE	A ADDRESS	B ADDRESS	VARIANT
[redacted]	[redacted]	[redacted]	[redacted]

EXM EXTENDED MOVE

The Extended Move instruction - (EXM) - uses one format only and moves the A field to the B field under condition specified by the six bit VARIANT character.

4.

1 + 1 = 0 with a "carry" of 1.

23.

A OPERAND (ADDRESS) is the subtrahend.
B OPERAND (ADDRESS) is the minuend.

42.

F/A/B/V/

SST does not perform its single character operation arithmetically. The variant 1 bits permit the movement of corresponding A character bits to the B character.

61.

VARIANT

5. Punctuation and parity bits are not involved with producing the sum for a BA instruction. Therefore, each memory location of the A or B operand will contribute six bits. For example, an operand three memory locations long would not be treated as "three characters" in a BA operation. Instead, it would appear as a binary value comprising a total of 18 bits.

---

24. Sensing op. code BS causes the A operand to be complemented before it enters the adder. Op. code BS also causes a 1 bit to be automatically inserted in the ADDER as the end around carry.

---

43. Each 1 bit in an SST variant permits corresponding A character bits to be moved to the B character. The numerical portion of the character at address #2396 is moved to address #3000 because of octal variant 17 in the instruction below.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14	15 20	21 62
		SST	2396,3000,17

Show the binary digits of octal 17. 00 1 1 1 1

---

62. Termination, direction of move, movement of punctuation and/or data bits are all determined by the six bit VARIANT CHARACTER. A programmer may construct a variant to accomplish and terminate the move as required. Possible six bit configurations are shown in the table on the answer side of this frame.

5.

18

Perhaps a way to visualize the difference between a decimal Add and BA may be as follows:

DECIMAL ADDITION: Bits are combined character by character (memory location be memory location).

BINARY ADDITION: Bits are combined BIT by BIT.

24.

A operand

1

ADDER

43.

$17_8 = 00\ 1111_2$

Since variant 1 bits permit corresponding A character bits to be moved into the B character, 17_8, will move a numerical portion of the A character.

62.

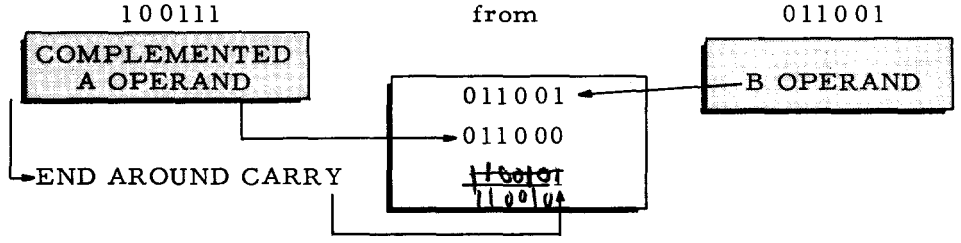
VARIANT CHARACTER

EXTENDED MOVE (EXM) CONDITIONS	VARIANT BITS V_6	V_5	V_4	V_3	V_2	V_1
Type of Move						
1. A-field data bits→B	X	X	X	X	X	1
2. A-field word-mark bits →B	X	X	X	X	1	X
3. A-field item-mark bits →B	X	X	X	1	X	X
Direction of Move						
1. right to left	X	X	0	X	X	X
2. left to right	X	X	1	X	X	X
Termination of Move						
1. automatic after single-character move	0	0	X	X	X	X
2. A-field word mark	0	1	X	X	X	X
3. A-field item mark	1	0	X	X	X	X
4. A-field record mark	1	1	X	X	X	X

6. A simple distinction between decimal and binary addition is the number of bits involved per memory location. In decimal operations, high order two bits (B and A cores) of 00 denote a numeric character. For example, 3890_{10} is stored in memory as four characters. Complete the bits in the memory locations below to show 3890_{10} stored as a decimal value.

ADDRESS	466	467	468	469
CONTENTS	B A 8 4 2 1 0 0 0 0 1 1	B A 8 4 2 1 0 0 1 0 0 0	B A 8 4 2 1 0 0 1 0 0 1	B A 8 4 2 1 0 0 0 0 0 0
	3	8	9	0

25. Retrieval of a BS op. code causes the A operand to be complemented and an end around carry 1 enters the adder. Determine the answer for the following BS as accomplished by the computer. Binary subtract:



44. Construct an octal variant for later movement of zone bits from the character at address #450 to the character at address #943. Write the instruction to accomplish this move.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14 15	20 21	62
		SST	450,943,60

63. An EXM variant written as octal 13 represents the binary digits:

0 0 1 0 1 1
V_6 V_5 V_4 V_3 V_2 V_1

Refer to the chart in frame 62 and explain bits of the variant above.

TERMINATION (V_6 V_5) automatic after Termination of single character
DIRECTION OF MOVE (V_4) left to right
TYPE OF MOVE (V_3 V_2 V_1) Fill with zeros data both → B

6.

ADDRESS	466	467	468	469
CONTENTS	BA 8421 00 0011	BA 8421 00 1000	BA 8421 00 1001	BA 8421 00 0000
	3	8	9	0

25.

```
 cc   c
011001
011000
      1
------
110010
```

44.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8 ... 14	15 ... 20	21 ... 62	
	SST	450,943,60	

Bits of a character already in the B memory location may be changed with an SST instruction. Assume that the character at the address tagged MORP is alphanumeric M (100100_2). What will this character be changed to by the SST instruction if frame 45?

63.

$$\frac{0}{V_6}\ \frac{0}{V_5}\ \frac{1}{V_4}\ \frac{0}{V_3}\ \frac{1}{V_2}\ \frac{1}{V_1}$$

TERMINATION (V_6 V_5) AUTOMATIC AFTER SINGLE CHARACTER MOVE.
DIRECTION OF MOVE (V_4) LEFT TO RIGHT.
TYPE OF MOVE (V_3 V_2 V_1) A FIELD WORD MARK AND DATA BITS ⟶ B.

7. Binary arithmetic operations use all six bits in a memory location. In other words, the B and A cores do <u>not</u> denote numeric or Hollerith groups. Instead, they have appropriate binary positional value. For example, convert 3890_{10} to its binary value below. Show its storage in addresses 480 and 481, then compare to the decimal storage example in frame 6.

480	481

26. A BS instruction in format F/A/ duplicates A. Carries beyond the A operand word mark are lost in the answer. Remembering to complement and to add 1 for around carry, determine the answer to the following BS instruction.

BS 111111

45.

MARK	LOCATION	OPERATION CODE	OPERANDS
	ONES	DC	#1C77 (ONE MEMORY LOCATION SET TO ALL 1 BITS)
		SST	ONES,MORP,Ø3

CONSTANT 1 1 1 1 1 1

VARIANT 0 0 0 0 1 1 (VARIANT 1 BITS PASS BITS)

"B" CHARACTER 1 0 0 1 0 0 (ALPHANUMERIC "M")

"B" RESULT IS, 1 0 0 1 1 1 WHICH IS THE LETTER ?

64. Construct an EXM binary variant to: 1 1 0 1 0 1 $_2$ = 6 5 $_8$

TERMINATE (V_6 V_5) with A field record mark.
DIRECTION OF MOVE (V_4) to be from right to left.
TYPE OF MOVE (V_3 V_2 V_1) A field item and data bits ⟶ B.

Type of Move						
1. A-field data bits⟶B	X	X	X	X	X	1
2. A-field word-mark bits⟶B	X	X	X	X	1	X
3. A-field item-mark bits⟶B	X	X	X	1	X	X
Direction of Move						
1. right to left	X	X	0	X	X	X
2. left to right	X	X	1	X	X	X
Termination of Move						
1. automatic after single-character move	0	0	X	X	X	X
2. A-field word mark	0	1	X	X	X	X
3. A-field item mark	1	0	X	X	X	X
4. A-field record mark	1	1	X	X	X	X

7.

```
    0    R = 1
  2 1    R = 1
  2 3    R = 1
  2 7    R = 1
 2 15    R = 0
 2 30    R = 0
 2 60    R = 1
2 121    R = 1
2 243    R = 0
2 486    R = 0
2 972    R = 1
2 1945   R = 0
2 3890
```

480	481
1 1 1 1 0 0	1 1 0 0 1 0

26.

A OPERAND 1 1 1 1 1 1

A COMPLEMENTED 0 0 0 0 0 0

←C C C C C

1 1 1 1 1 1

1 END AROUND CARRY

0 0 0 0 0 0

WM

Consequently, format F/A/ zeros out a location when used with a BS instruction.

45.

CONSTANT 1 1 1 1 1 1

VARIANT 0 0 0 0 1 1 (VARIANT 1 BITS PASS BITS)

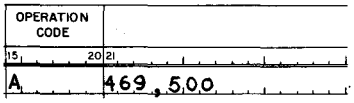
"B" CHARACTER 1 0 0 1 0 0 (ALPHANUMERIC "M")

"B" RESULT IS, 1 0 0 1 1 1 WHICH IS THE LETTER P

64. $110101_2 = 65_8$

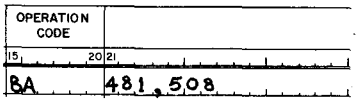
EXTENDED MOVE (EXM) CONDITIONS	VARIANT BITS V_6	V_5	V_4	V_3	V_2	V_1
Type of Move						
1. A-field data bits→B	X	X	X	X	X	1
2. A-field word-mark bits →B	X	X	X	X	1	X
3. A-field item-mark bits→B	X	X	X	1	X	X
Direction of Move						
1. right to left	X	X	0	X	X	X
2. left to right	X	X	1	X	X	X
Termination of Move						
1. automatic after single-character move	0	0	X	X	X	X
2. A-field word mark	0	1	X	X	X	X
3. A-field item mark	1	0	X	X	X	X
4. A-field record mark	1	1	X	X	X	X

8. An A operand of 3890_{10} and a B operand of 200_{10} are shown below as added by decimal addition (mnemonic A) and binary addition (mnemonic BA).



500	501	502
00 0010	00 0000	00 0000
2	0	0

B OPERAND 200_{10} =



507	508
000011	001000

466	467	468	469
00 0011	00 1000	00 1001	00 0000
3	8	9	0

A OPERAND 3890_{10} =

480	481
111100	110010

$$\begin{array}{r} 200 \\ +\ 3890 \\ \hline 4090 \end{array}$$

Perform the addition ⇨

$$\begin{array}{r} 0\,0\,0\,0\,1\,1\,0\,0\,1\,0\,0\,0 \\ +\,1\,1\,1\,1\,0\,0\,1\,1\,0\,0\,1\,0 \\ \hline 1\,1\,1\,1\,1\,0\,1\,0 \end{array}$$

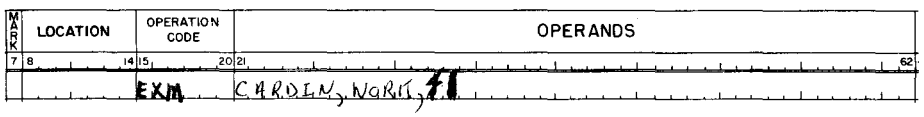
27. List the word mark conventions for BS in format F/.

B operand SW, A operand SW if omitted

Since the A and B addresses are not indicated on the coding form for BS in format F/, how are operands obtained? A ADDRESS Reg. (B ✓ A ✓)

46. Format F/ may be used for an SST instruction if it follows an SST of F/A/B/V/. In format F/, the A and B addresses are provided by the A and B ADDRESS registers. The variant for format F/ is the same as the preceding SST instruction.

65. Refer to the chart in frame 64, then write an EXM instruction to move data bits from the field tagged CARDIN to the field tagged WORK. Move the data from right to left and terminate when the first item mark of CARDIN is sensed.



8.

```
  000011001000
+ 111100110010
  111111111010
```

27.

THE B OPERAND SHOULD BE WORD MARKED AND
THE A OPERAND ALSO, IF SHORTER THAN B.
A ADDRESS FROM A ADDRESS REGISTER
B ADDRESS FROM B ADDRESS REGISTER

46.

A and B ADDRESS

65.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
		EXM	CARD IN, WORK, 41

9. A total of seven memory locations are involved for decimal addition of 200 plus 3890. In contrast, only four memory locations are needed for binary addition of the same values. Scientific applications may advantageously use binary arithmetic. In your business data processing, you may frequently use binary arithmetic to modify addresses. You may also use "counters" operating in binary.

As a check of the previous frame, what does 111111111010_2 equal in decimal?

---

28. BA and BS are used for similar purposes. BA increases an address being modified, BS Decreases an address being modified. Similarly, BA may be used to increment a binary counter and BS may be used to decrement a binary counter. Neither BA nor BS utilizes overflow or zero balance indicators. If a high order carry is generated, it will be lost.

---

47. SST may be written in format F/ to chain A & B addresses if the format F/A/B/V precedes it. The variant for format F/ is provided by the preceding instruction. Each SST operates on a single character basis. Write the instructions to move the numerical portions of the five character field in addresses #601 - #605 to the area tagged NOZONE.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14	15 20	21 62
		SST	605, NOZONE, 1
		SST	
		SST	
		SST	
		SST	

---

66. H-200 versatility is exemplified by its capability of accepting a character code that is foreign and converting to the comparable H-200 character. A single H-200 instruction will move characters having a different bit configuration and translate them into Honeywell characters. The mnemonic op. code for this instruction is MAT which stands for MOVE and TRANSLATE.

9.

4090_{10}

$$\begin{array}{rl} 000011001000_2 & = \quad 200_{10} \\ \underline{111100110010_2} & = \underline{3890_{10}} \\ 111111111010_2 & = 4090_{10} \end{array}$$

Therefore, the answer is the same whether decimal or binary addition is used.

28.

DECREASES
DECREMENT
BINARY COUNTER

47.

F/A/B/V/

VARIANT OCTAL 17 = 001111. SINCE VARIANT 1 BITS PERMIT A CHARACTER BIT TO MOVE, 17_8 MOVES NUMERICAL

MARK	LOCATION	OPERATION CODE	OPERANDS
		SST	605,NOZONE,,17
		SST	
		SST	
		SST	
		SST	

66.

MOVE
TRANSLATE

10. Word mark conventions for the various BA formats are given below:

F/A/B/ - The B operand should have a word mark in its leftmost memory location to terminate the operation. If the A operand is shorter than B, the A operand should also be word marked.

F/A/ - Word mark the A operand.

F/ - Word marks as for F/A/B/.

In formats F/ and F/A/B/, if the A operand exceeds the length of the B operand, will the exceeding portion of A be processed? No

Why? B Word mark terminate operation

29. Before using BS to decrement a counter, do the following:

Write the coding to set a binary 1 in a memory location and tag it ONE. (This will be a one memory location A Operand.)

Write coding establishing a binary counter of two memory locations containing binary for 500_{10} and tag it COUNT. (This will be a two memory location B operand.)

MARK	LOCATION	OPERATION CODE	OPERANDS
	ONE	DCW	#1B1
	COUNT	DCW	#2B500

48. SST instructions on the coding form lines 2-5 in frame 47 are chained. Show the register addresses below assuming NOZONE is address #700.

CODING FORM LINE	OP. CODE REGISTER	A ADDRESS REGISTER	B ADDRESS REGISTER	VARIANT REGISTER
#1	SST	605	700	17
#2	SST	604	699	
#3	SST	603	698	
#4	SST	602	697	
#5	SST	601	696	

67.

OP CODE	A ADDRESS	B ADDRESS	VARIANT 1	VARIANT 2
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

MAT MOVE AND TRANSLATE

MAT will read a field of characters expressed in code foreign to that of the Honeywell product line, automatically translate into Honeywell code, and store translated values in the same or different field. Change from one code to another involves use of a table stored in memory to TRANSLATE characters.

10.

NO

THE B OPERAND WORD MARK TERMINATES THE OPERATION, THEREFORE, THAT PORTION OF A EXCEEDING B WILL NOT BE PROCESSED.

(Or equivalent answer.)

29.

DCW statements are used so that the A and B operands will be word marked.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
	ONE	DCW	#1B1
	COUNT	DCW	#2B5ØØ

48.

CODING FORM LINE	OP. CODE REGISTER	A ADDRESS REGISTER	B ADDRESS REGISTER	VARIANT REGISTER
#1	SST	605	700	17
#2	SST	604	699	
#3	SST	603	698	
#4	SST	602	697	
#5	SST	601	696	

67.

TRANSLATE

11. Remember that BA is an arithmetic operation, then briefly explain why only the A operand needs a terminating word mark in format F/A/.

A duplicate

30. Assume some repeating operation is being performed and the counter is to be decremented for each operation. Write the instruction to accomplish decrementing.

MARK	LOCATION	OPERATION CODE	OPERANDS
	ONE	DCW	#1B1
	COUNT	DCW	#2B5ØØ
			REPETITIVE OPERATION
		AS	ONE, COUNT

49. An SST instruction may be chained, and the preceding SST variant will be retained. SST operates on a single character basis. That is, any desired character bits in a single memory location may be moved into another location as specified by the 1 bits in the VARIANT character.

68. A translation table is created by the programmer and stored in memory. Since each character consists of 6 bits (00_8 through 77_8), a possible 64 different configuration of characters may be expressed and stored as a TRANSLATION TABLE.

11.

SINCE BA IS AN ARITHMETIC OPERATION, FORMAT F/A/ "DUPLICATES" A. THE A OPERAND IS THE ONLY OPERAND INVOLVED.

(Or equivalent answer.)

30.

MARK	LOCATION	OPERATION CODE	OPERANDS
	ONE	DCW	#1B1
	COUNT	DCW	#2B5ØØ
			REPETITIVE OPERATION
		BS	ONE,COUNT

49.

VARIANT

68.

64

TRANSLATION TABLE

12. A Compare instruction (C 1283, 113) stored in memory locations 500 - 504, is to have its B address "modified" from 113_{10} to 188_{10}.

Because addresses are binary, this increase of the B address by 75_{10} may be accomplished with a BINARY ADD instruction whose mnemonic op. code is [illegible].

---

31. BA and BS are considered to be arithmetic instructions. The group of instructions which contained BCT, BCC, BCE, etc. deal with binary digits for logic rather than arithmetic operations.

Instructions called EXTRACT (EXT), HALF ADD (HA), and SUBSTITUTE (SST) are part of the BCT, BCC, BCE group. Even though these instructions deal with binary digits, they are referred to as LOGIC rather than ARITHMETIC instructions.

---

50. SST operates on a single character basis, consequently word marks are not required to terminate the operation.

The next logic instruction discussed (EXTRACT, mnemonic EXT) is similar in function to SST. However, EXT may move or not move any desired character bits from one or more memory locations. Because this instruction may involve more than one memory location, WORD marks are required to TERMINATE the operation.

---

69. MAT format F/A/B/V_1/ V_2/ may be defined as follows:

a. The type of operation to be performed is indicated by the OP CODE.
b. The memory location of the field to be Moved and Translated is specified by the A ADDRESS.
c. The location into which the translated characters are to be stored is specified by the B ADDRESS.
d. V_1 and V_2 provide the base address of the TRANSLATION table in memory.
e. A word mark within the A-field terminates the MAT process after the word marked character is translated.

12.

BINARY ADD
B A

31.

LOGIC
ARITHMETIC

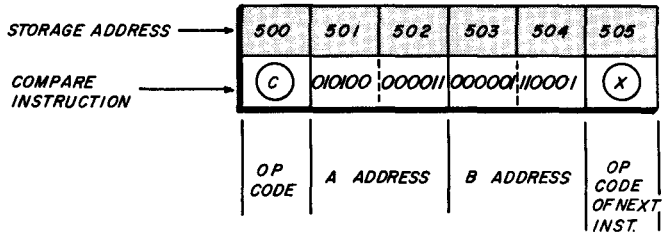
50.

WORD
TERMINATE

69.

OP. CODE
A ADDRESS
B ADDRESS
TRANSLATION

13. BA format F/A/B/ may be used to modify the B address of the Compare instruction shown below:



Assume that a binary constant equal to 75_{10} is stored in memory and has the tag B75. Write the instruction changing the B address of the Compare instruction to 188_{10}.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14 15	20 21	62
		BA	B75,504

---

32. EXT (Extract), HA (Half Add) and SST (Substitute) provide logic operations not available in your previous system. The versatility of the H-200 permits equally versatile programming applications of these LOGIC instructions.

51.

EXT	EXTRACT (Logical Product)

EXT is non arithmetic and may be written in the standard three formats not incorporating a variant.

EXT format F/A/ chains the B address rather than duplicating A.

Addresses for format F/ are obtained from the A and B A DDRESS Register

---

70. Three character addressing is required to specify the translation table address of the equivalent Honeywell code.

The base address is formed by the two VARIANT CHARACTERS.

The "foreign" code from the location specified by the A ADDRESS provides the low order six bits of the translation table address.

13.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 14	15 20	21 62
		BA	B75,504

32.

LOGIC

51.

B

A

A and B ADDRESS REGISTERS

70.

VARIANT CHARACTERS

A ADDRESS

14. To use tag B75 (in previous example) as the A operand, it must have been defined as a binary constant equal to 75. Considering its use in modifying an address, would the data formatting op. code have been DC or DCW? DCW Why? Operand Size

MARK	LOCATION	OPERATION CODE	OPERANDS
	B75		#2B75
		BA	B75,5Ø4

33. HA, EXT, and SST are available to the programmer to increase his flexibility of programming. Since these logic instructions are used at a programmer's discretion, their explanations in following frames suggest rather than specify applications.

Write the full name of the logic mnemonics:

HA, HALT, EXT, EXTRACT, SST, SUBSTITUTE

52. All EXTRACT instructions follow these rules and store the result at the B address.

Bit in A-field	Bit in B-field	Bit in Result field
1	1	1
1	0	0
0	1	0
0	0	0

A word mark is required for the shorter of the two operands. The operation terminates when this word mark is sensed.

Format a: A-field is combined bit by bit with B-field.

Format b: A-field is combined bit by bit with data specified by B address register.

Format c: Data specified by contents of A and B address registers combined bit by bit.

71. Any "foreign" six bit character has a binary value. For example, 1401 alphanumeric character "A" has a bit configuration of 110001 which equals 49_{10}. The corresponding Honeywell alphanumeric character "A" (010001) will be stored in the forty-nineth translation table address. An illustration of accessing the proper memory location in the translation table is shown on the answer side of this frame.

14.

DCW

THE A OPERAND (BINARY CONSTANT) MAY BE SHORTER THAN THE ADDRESS (2, 3, or 4 CHARACTERS) THAT IS TO BE MODIFIED. WHEN THE A OPERAND OF BA INSTRUCTIONS IS SHORTER THAN THE B OPERAND, THE A OPERAND SHOULD BE WORD MARKED. DEFINING THE BINARY CONSTANT WITH A DCW ELIMINATES THE NECESSITY OF ALSO WRITING A SW.

(Or equivalent answer.)

33.

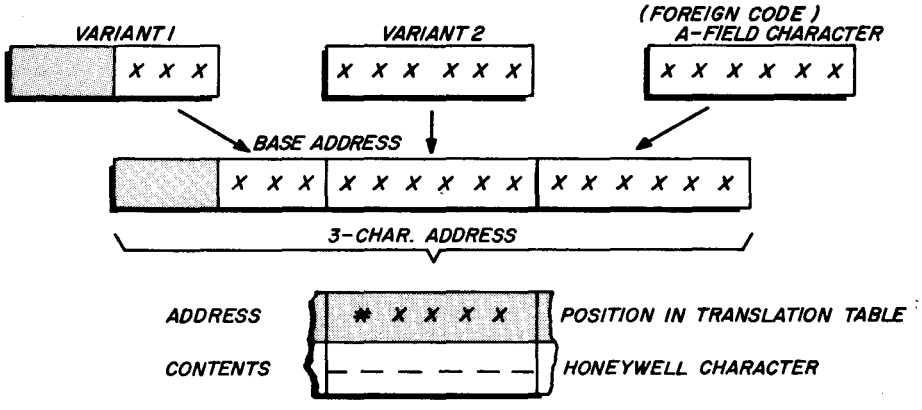
HALF ADD
EXTRACT
SUBSTITUTE

The following frames suggest applications of these logic instructions and explain their operation.

52.

NO ANSWER REQUIRED

71.



15. Propagation of "carries" is related to the length and word mark placement of A or B operands. For example, assume equal length operands and the B operand word marked. If the operand values are such that their high order bits (leftmost) propagate a "carry", will the sum of B+A be correctly stored? NO Why? carry will be lost, B operand wrong the operation

34.

HA	HALF ADD (exclusive or)

Half Add instructions add the A operand to the B operand without propagating carries. The result is stored in the B field. Basic rules for half adding are extremely simple since carries are disregarded. Complete this chart.

A-FIELD BIT		B-FIELD BIT		RESULT BIT
1	+	1	=	0
1	+	0	=	1
0	+	1	=	1
0	+	0	=	0

53. Rather than using a variant to specify which bits are to be passed from the A field into the B field, EXT uses a constant in the B field. Each 1 bit in the constant permits the corresponding A field bit to pass into B. Determine the result in the problem below.

A FIELD	41_8, 72_8, 60_8 =	100001	111010	110100
CONSTANT IN B FIELD	03_8, 77_8, 14_8 =	000011	111111	001111
	RESULT	000001	111010	000100

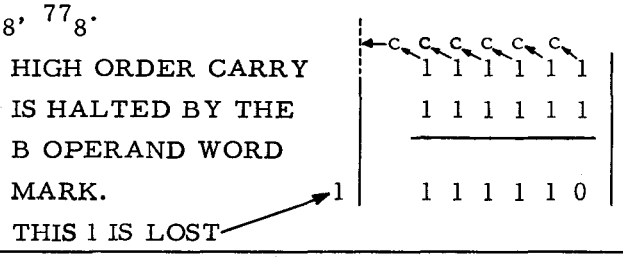
72. The translation table "base address" is established with a MORG statement. MORG directs assembly to assign addresses to following coding beginning with the next multiple of the address written in the operands field. What will the low order six bits be (regardless of which multiple, 128, 256, 512, . . . 4096 etc. is assigned by assembly) for the MORG below? ___ ___ ___ ___ ___ ___

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
	TABLE	MORG	64

15.

NO

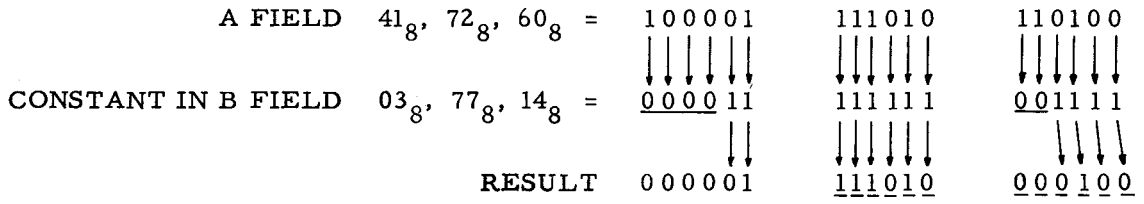
THE B OPERAND WORD MARK TERMINATES THE OPERATION. IF A CARRY IS GENERATED TO BE PROPAGATED BEYOND THE HIGH ORDER BIT, THE CARRY WILL BE "LOST". EXAMPLE: BA 77_8, 77_8.



34.

A-FIELD BIT		B-FIELD BIT		RESULT BIT
1	+	1	=	0
1	+	0	=	1
0	+	1	=	1
0	+	0	=	0

53.



PARTS OF CHARACTERS, CHARACTERS, AND GROUPS OF CHARACTERS MAY BE EXTRACTED BY A PROPERLY CONSTRUCTED CONSTANT "MASK" IN THE B FIELD OF AN EXT INSTRUCTION.

72.

Low order six bits = 0 0 0 0 0 0 for a MORG of 64, regardless of which multiple is assigned.

EXAMPLE:

		THREE CHARACTERS ADDRESS		
128_{10} =		0 0 0	0 0 0 0 0 0	0 0 0 0 0 0
1024_{10} =		0 0 0	0 1 0 0 0 0	0 0 0 0 0 0
4096_{10} =		0 0 0	0 0 0 0 0 0	0 0 0 0 0 0

16. Loss of a high order carry is prevented by using zeros in the high order position. Establish the required A and/or B operand punctuation for the following instruction, then calculate the binary sum.

MARK	LOCATION	OPERATION CODE	OPERANDS
		BA	720,601

	#600	#601
ADDRESS		
B OPERAND (0077_8)	000000	111111

	#720
ADDRESS	
A OPERAND (77_8)	111111

$0077_8 = 000000111111$

$78_8 =$ 111111

SUM 111110

600 → Determine operation
720

35. The HA instruction uses the three standard formats for instructions and does not require a variant. Indicate these formats below:

F/A/B/
F/A/
F/

54. A constant to be used as a "mask" in an EXT instruction is constructed in relation to what the programmer desires to pass. For example, the portion of a constant that is octal 77 will move a corresponding character unchanged. 77_8 passes the character (in the same relative position) without change because a 1 bit permits the corresponding bit to pass. Octal 77 = 1 1 1 1 1 1 $_2$, therefore, all bits in the corresponding character are permitted to pass.

73. Since V_1 and V_2 of a MAT instruction specify the translation table base address, the MORG 64 statement may be tagged. Then, this tag may be written in a MAT instruction instead of the two VARIANT characters to specify the BASE ADDRESS of the translation table. The six bits in the LOW order of the address designate a specific position of the translation table and are supplied by the foreign CHARACTER that is to be TRANSLATED.

16. The word mark in address #600 (B operand) terminates the operation. The A operand (address #720) is word marked because it is shorter than B.

MARK	LOCATION	OPERATION CODE	OPERANDS
		SW	72Ø,6ØØ
		BA	72Ø,6Ø1

	#600	#601
ADDRESS		
B OPERAND (0077_8)	000000	111111

	#720
ADDRESS	
A OPERAND (77_8)	111111

ccccccc

0077_8 = 000000111111

77_8 = 111111

SUM = 000001111110

35.

F/A/B/

F/A/

F/

54.

Octal 77 = 111111 $_2$ therefore, all bits in the CORRESPONDING character are permitted to PASS.

73.

VARIANT

BASE ADDRESS

LOW order

CHARACTER to be TRANSLATED

17. The word mark of a shorter A operand does not terminate the BA operation, it simply stops supplying bits. This is sometimes referred to as "zeros implied" by an A operand word mark.

ACTUAL ZEROS TO PROVIDE FOR CARRY ⇨ 000000111111 B OPERAND = 0077_8

ZEROS IMPLIED BY A OPERAND WORD MARK ⇨ 000000111111 A OPERAND = 77_8

Inadvertantly supplying unrelated adjacent bits is prevented by word marking an A operand that is shorter than a B operand. The A operand word mark means that Zeros are implied.

---

36. HA is a logical rather than arithmetic instruction. The major difference is in the F/A/ instruction format. Rather than "duplicating" the A operand, the B operand is "chained". The B operand address is obtained from the B ADDRESS REGISTER.

---

55. Octal 17 as a mask in the B field of an EXT instruction will pass the NUMERIC portion of a corresponding character and block the ZONE bits. This passing and blocking is accomplished due to the binary digits of 17_8 equaling 0 0 1 1 1 1$_2$.

---

74. Construction of a translation table is not difficult. The foreign characters to be translated are listed in ascending order of their binary values. The equivalent Honeywell characters (to be entered into the translation table) are then listed to correspond with the order of the foreign code. For example:

FOREIGN CODE	1401 CHARACTER	H-200 CHARACTER	H-200 OCTAL
000000	blank	blank = 001101	15
000001	1	1 = 000001	01
000010	2	2 = 000010	02
etc.	etc.	etc.	etc.
001010	ø	ø = 000000	00
001011	#	# = 101010	52
etc.	etc.	etc.	etc.

Will all the H-200 characters be listed in ascending binary order? No

Why? No direct relationship between 1401 binary character values & H-200 characters.

17.

ZEROS
IMPLIED

36.

B ADDRESS REGISTER

55.

NUMERIC
ZONE
001111_2

74.

NO

THERE IS NOT A DIRECT RELATIONSHIP BETWEEN 1401 BINARY CHARACTER VALUES AND H-200 CHARACTERS. IT IS FOR THIS REASON THAT TRANSLATION IS REQUIRED.

(Or equivalent answer.)

18. BA instructions may be used to increment a binary counter which is counting some repeating operation. Write the appropriate instruction on line 6 below.

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1				ONE	DCW	#1B1 PUTS A BINARY 1 IN A MEMORY LOCATION
2				COUNT	DCW	#2BØ SETS TWO LOCATIONS TO Ø AS A BINARY COUNTER
3						
4						
5						
6					BA	ONE, COUNT
7						
8						
9						
10						

37. HA formats require a word mark defining the limit of the shorter operand. Suppose that at some point in a program the op. code of an instruction is to be changed to another op. code. To accomplish this with a HA instruction, will a SW instruction be required? NO

Why? op code has word mark

56. Notice which bits (all, numeric, zone, etc.) will be passed by each two octal digits in mask specified below. MASK is loaded into the area tagged BFLD so that the constant (MASK) will not be changed by the EXT operation. Continue to the answer side of this frame.

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1				MASK	DCW	#4C771700606Ø
2					BSS	
3					LCA	MASK, BFLD
4					EXT	CARDIN, BFLD

75. Honeywell characters may be entered into the translation table using octal (#16C . . . . .) or alphanumeric (#32A . . . .) DC statements as shown below.

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	
1				TABLE	MORG	64
2					DC	#16C150102030405
3					DC	#16C166162636465
4					DC	#16C4041424344445
5					DC	#16C172122232425

} V_1 V_2 BASE ADDRESS
} OCTAL DESIGNATION OF 64 H-200 CHARACTERS IN 1401 SEQUENCE

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	
1				TABLE	MORG	64
2					DC	#32A61234567890#
3					DC	#32A-JKLMNOPQR1$

} V_1 V_2 BASE ADDRESS
} ALPHANUMERIC DESIGNATION OF 64 H-200 CHAR., 1401 SEQUENCE

A total of 64 memory locations will be required in either case. H-200 characters are entered in 1401 sequence because this is an example of 1401 translation.

18. The BA instruction (on line 6) increments the counter as each card is read until the number of cards equals the value in the location tagged TOTAL.

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1				ONE	DCW	#1B1 PUTS A BINARY 1 IN A MEMORY LOCATION	
2				COUNT	DCW	#2BØ SETS TWO LOCATIONS TO Ø AS A BINARY COUNTER	
3				READ	PDT	CARDIN,51,41 THESE INSTRUCTIONS CAUSE THE	
4					PCB	*,ØØ,41,1Ø CARD READER TO READ A CARD	
5					PCB	ERROR,ØØ,41,41 AND THEN CHECK FOR ERRORS ETC.	
6					BA	ONE,COUNT A BINARY 1 IS ADDED TO THE COUNTER	
7					C	TOTAL,COUNT THE VALUE IN TOTAL IS COMPARED TO THE COUNTER	
8					BCT	NEXT,42 BRANCH IF AN EQUAL COMPARE IS INDICATED	
9					B	READ BRANCH TO READ ANOTHER CARD IF UNEQUAL COMPARE	
10				NEXT	DSM	PROGRAM CONTINUES	

37.

NO

AN OP. CODE IS A SINGLE CHARACTER. THEREFORE, IT IS THE SHORTEST POSSIBLE OPERAND, AND ALREADY CONTAINS A WORD MARK.

(Or equivalent answer)

56. Refer to the coding form below to answer frame #57.

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1				MASK	DCW	#4C77177ØØ6Ø	
2					DSM		
3					LCA	MASK,BFLD	
4					EXT	CARDIN,BFLD	

75.

64

1401

1401

19. In the previous example, suppose that the binary value in the locations tagged TOTAL equals 129_{10}. Show the binary contents of the location tagged COUNT when the program continues to NEXT.

0 0 0 0 1 0 0 0 0 0 0 1

38. As an example of HA, suppose that a decimal add op. code (Mnemonic A, octal 36) is to be changed to become a decimal subtract op. code (Mnemonic S, octal 37). The address of the op. code A is #500. Determine what binary value is needed as the constant for the HA instruction below:

MARK	LOCATION	OPERATION CODE	OPERANDS
	NEWOP	DC	#161 ← ANSWER
		HA	NEWOP, 500

57. For example, the first character at CARDIN will have all of its six character bits passed into BFLD because 77 contains all 1 bits.

17 will pass the Numeric portion of the second character of CARDIN and Block the Zone bits.

00 will Block the second character of CARDIN.

60 will Pass the Zone bits of the Third character of CARDIN and block the Numeric portion.

76. Assume MORG 64 has been accomplished and tagged TABLE to refer to the base address of V_1 V_2. Also assume a translation table has been filled with H-200 characters written in the foreign code sequence.

Write an instruction to move and translate from the area tagged FCODE into the area tagged HCODE.

MARK	LOCATION	OPERATION CODE	OPERANDS
		MAT	FCODE, HCODE, TABLE

19.

$$000010000001_2 = 129_{10}$$

(Return to frame 20, page 233)

38.

Since octal 36 (011110) is to be changed to octal 37 (011111) a binary constant of 1 is required.

MARK	LOCATION	OPERATION CODE	OPERANDS
	NEWOP	DC	#1,B,1
		HA	NEWOP,500

(Return to frame 39, page 233)

57.

17 will pass the NUMERIC portion of the second character of CARDIN and BLOCK the ZONE bits.
ØØ will BLOCK the second character of CARDIN.
6Ø will PASS the ZONE bits of the THIRD character of CARDIN and block the NUMERIC portion.

(Return to frame 58, page 233)

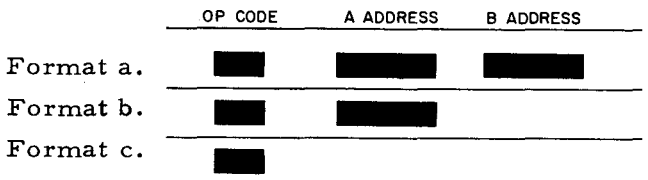
76.

MARK	LOCATION	OPERATION CODE	OPERANDS
		MAT	FCODE,HCODE,TABLE

MAT is written in format F/A/B/V_1 V_2, where the A field contains characters to be translated and the B field is the area for the translated characters to be stored. MAT terminates with the character in A or B that is word marked.

(Continue to page 272)

CSM	CHANGE SEQUENCING MODE



FUNCTION

Format a: The A address is loaded into the A-address register and the B address is loaded into the B-address register. Contents of the I- and I'-address registers are interchanged, and the program branches to the instruction whose op code address was previously stored in the I'-address register.

Format b: The A address is loaded into the A-address register. The B address is chained from the B-address register.

Format c: Both A and B addresses are chained from the A and B-address registers.

WORD MARKS - Formats a, b, and c: Word marks are not affected by this instruction.

CSM instruction execution is explained in Lesson V, along with the 16 control memory registers. However, the H-200 contains an automatic change sequencing mode instruction referred to as "Op. Code Trapping". This capability is available when the addressing mode variant (CAM) is specified as octal 24, 04, 64, for two, three, and four character addressing respectively.

Op. code trapping involves the record marking of a non standard operation code. Sensing of a record marked op. code has the effect of an automatic CSM instruction, in that it causes I and I' registers to be interchanged. For example, the letters M (multiply) and D (divide) could be record marked and then used as appropriate op. codes.

In the example of M and D above, the I' register would contain the address of a co-sequence routine to determine whether the trapped op. code (record marked) is an M or a D. After determining whether the desired operation is multiplication or division, the co-sequence routine branches to the address of an appropriate multiply or divide subroutine.

Upon completion of the specified subroutine, the contents of I are restored and I' is again loaded with its co-sequence routine address. The program then continues in sequence as directed by the I register.

RNM	RESUME NORMAL MODE

Format

FUNCTION

Certain conditions, such as transfer of information to or from a remote data communication device, make it necessary for the main program to be interrupted. The H-200 can be equipped with an interrupt indicator which is automatically turned on when an interrupt signal is sensed. For example, a communication control can generate an interrupt signal whenever it requires access to the main memory. When an interrupt signal occurs, the following activities are automatically initiated:

1. The instruction being executed is completed.
2. The interrupt indicator is turned on.
3. Settings of arithmetic and comparision indicators are preserved in auxiliary storage areas.
4. Contents of the I-address and the interrupt register are interchanged. The program branches to the instruction whose address was initially stored in the interrupt register.
5. The machine switches to three-character addressing mode if the normal sequence of instructions is stored in the two-character addressing mode.

The interrupt indicator remains on during execution of the subroutine. It indicates to the central processor that a priority routine is being performed and any further interrupt signals should be rejected. Upon completion of the subroutine, the normal sequence of instructions can be returned to via an RNM instruction. This instruction reverses the effect of activities 2, 3, 4, and 5 listed above.

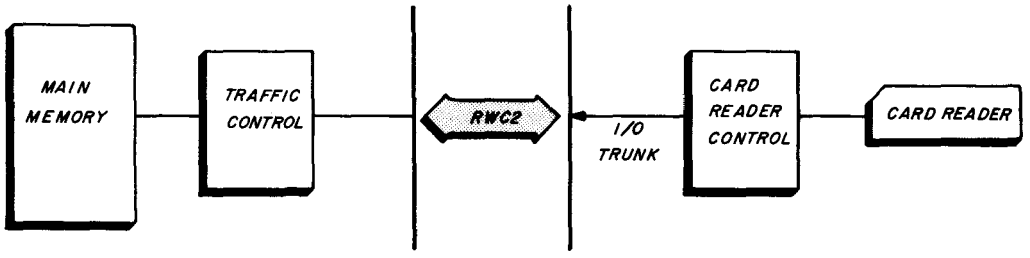
The Resume Normal Mode instruction should be coded with zeros in the A address, the B address, and the variant character. The first two instructions in the interrupt subroutine should be SCR instructions which store the contents of the A- and B-address registers in the A and B addresses of the RNM instruction. Immediately following these two instructions there should be a routine to test for the previous variant character and to store that character in the variant field of the RNM instruction. Thus, the coded zeros of the RNM instruction are replaced by the contents of control memory registers, providing re-entry to the main program.

EXAMPLE: After the execution of the RNM instruction, the interrupt register contains the address of the op code which immediately follows the instruction. The sample coding below illustrates a convenient method of restoring the starting address of the interrupt subroutine (ENTER) in the interrupt register when the normal program sequence is resumed. Note that the first two instructions of the subroutine are SCR (Store Control Registers) instructions which store the contents of the A- and B-address registers in the A address (RESUME + 3) and the B address (RESUME + 6) of the RNM instruction.

INPUT/OUTPUT OPERATIONS

Simultaneity of peripheral operations together with central processor computing is achieved through the principle of "time sharing" illustrated in Lesson II. Input/output operations require access to memory for only a fraction of the total time required for mechanical functions of peripheral devices. Consequently, central processor time is shared among the read/write channels. If no time is required by a peripheral unit, the central processor is granted additional computing time.

It should be remembered that units are permanently connected to any of 16 input or output trunks. However, the programmer has complete freedom to assign or reassign read/write channels by means of program instructions. When the programmer writes an input/output instruction, he specifies, among other things, the read/write channel over which data transfer is to take place, and the peripheral control that is to receive or transmit data.



As soon as data transfer is complete in the example above, RWC 2 is automatically removed from the interface. The programmer may then reassign RWC 2 to another peripheral control in another input/output instruction if desired.

Transfer of information between memory and a peripheral device is either input to or output from the central processor. Regardless of whether the operation is input or output and whichever device is to be directed, the H-200 uses a single instruction. The mnemonic op. code of this single input/output instruction is PDT, which stands for Peripheral Data Transfer.

Another instruction is used in conjunction with PDT to either set up controls, or for testing the condition of peripheral devices. These dual operations of initial control of devices and subsequent testing of devices have the mnemonic op. code PCB. This op. code represents the function of Peripheral Control and Branch.

At the conclusion of the following frames (which primarily discuss card reading), you will be directed to the appropriate section of the Programmers' Reference Manual for information concerning other peripheral devices.

1. Transfer of data to or from peripheral units is accomplished by an instruction with the mnemonic op. code PDT. The letters P, D, T, stand for Peripheral Data Transfer. The several tasks accomplished by this instruction involve specifying the memory location at which transfer is to begin, designating the read/write channel, identifying the control unit, and when more than one device is controlled, the particular device is also selected.

---

13. The purpose of CI is to designate which read/write channel is to be used and whether it should be interlocked or not. If a PDT is to be performed on RWC #1 and the system contains optional RWC #1', CI MUST show interlocking.

Advisedly, programmers may interlock RWC #1 even if the system does not contain RWC #1'. In this manner, programs will not have to be reviewed and CI's rewritten if optional RWC #1' is acquired later.

---

25. In a "small" H-200 system, PCB format F/A/C1/ may imply the busy status of a particular peripheral device because each read write channel is assigned one specific device

In a larger H-200 system, PCB format F/A/C1/ is only used if a programmer desires to know the busy status of a read / write channel.

---

37. An initializing PCB controls a peripheral device in much the same manner that switches or dials might be manually set to control a device. Consequently, an initializing PCB is written at the start of a program (and whenever operation of a device is to be changed).

Format of an initializing PCB is the same as that of a PDT. Briefly state the purpose of each part of the PCB format: F/A/C1/C2/C3 - Cn/.

1.

PERIPHERAL DATA TRANSFER

13.

NO ANSWER REQUIRED

25.

ONE DEVICE MAY BE ASSOCIATED WITH EACH RWC IN A SMALL H-200 SYSTEM.

(Or equivalent answer.)

READ/WRITE CHANNEL

37. F/A/C1/C2/C3 - Cn/

F is the op. code causing the computer to perform PCB.
A is the address for the branch, if conditions specified by control characters are not satisfied.
C1 specified the read/write channel. (ØØ)
C2 designates input or output of a particular trunk connected to the desired peripheral device.
C3 - Cn are control characters, the number of which depends on the needs of the device.

(Or equivalent answer.)

2. The other instruction used with PDT has the mnemonic op. code PCB. The letters P, C, B, stand for Peripheral control and branch. The purpose of PCB is to either control or test a peripheral unit and to provide a branch address in case control can not be effected or the test is not as desired.

---

14. As pointed out earlier, assignment of RWC's with control character C1 is at the discretion of the programmer.

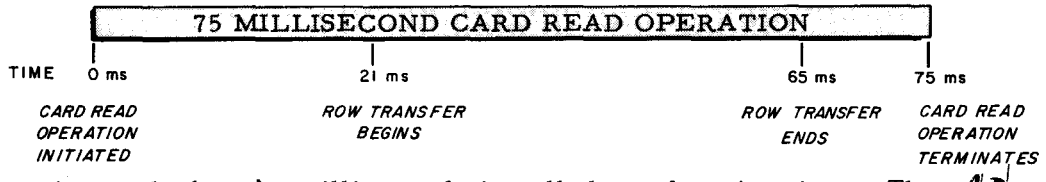
C2's purpose is to designate an input or ourput trunk to which some control unit is attached. The programmer writes C2 for either an input or output trunk as determined by the type unit attached to the trunk. As examples: A card reader provides INPUT to the central processor and is attached to an INPUT TRUNK. A card punch receives OUTPUT from the central processor and is attached to an OUTPUT TRUNK.

---

26. PCB instructions need not specify any particular RWC. If control character C1 is written as ØØ, C1 has no effect. The control unit connected to the input/output trunk designated by C2 will then be tested. PCB format F/A/ØØ/C2/ causes the program to branch to the A address if the control unit specified by control character C2 is busy. ØØ as C1 means no READ/WRITE CHANNEL is to be tested.

---

38. Timing of peripheral operations is the next subject to be discussed. Consider the purely MECHANICAL functions of a card reader whose times are shown below.



The first interval of 21 milliseconds is called acceleration time. The 10 millisecond interval between the end of row transfer and termination of the card read is deceleration time. Since these operations are purely MECHANICAL functions of the card reader, the control processor is free for a total of 31 milliseconds.

2.

PERIPHERAL CONTROL BRANCH

14.

INPUT
INPUT TRUNK
OUTPUT
OUTPUT TRUNK

Note: Units providing both input and output (tape units, etc.) are attached to both an input and an output trunk.

26.

C2
READ/WRITE CHANNEL

38.

21
10
MECHANICAL
31

3. In functioning as either a control or test of peripheral units, PCB may be used in three applications. It may initialize a unit, such as setting the card reader to read Hollerith code, etc. It may test a unit to see if it is still busy with a pervious instruction. It may check for errors such as illegal card punches or hole count errors, etc.

Listing the three uses of PCB, it first can initialize a unit, PCB can also test a unit to see if it is still busy, or PCB can check for error. In all three applications, PCB provides the address of a branch.

---

15. The purpose of C1 is to specify which RWC is to be used and whether it should be interlocked ($5X_8$) or not ($1X_8$). In Lesson II, you saw the second I/O control character (C2) used as a trunk designation. A basic H-200 has four pairs of input/output trunks that are designated: 00&40, 01&41, 02&42, 03&43.

Do you remember what was meant by a first digit of 0? OUTPUT. A first digit of 4? INPUT.

---

27. Peripheral Control and Branch instructions (mnemonic P C B) may be written in the same format as PDT. Show this format below and identify its parts.

F/A/C1/C2/C3-CN/

OP CODE

ADDRESS of branch

READ/WRITE channel

Input or output TRUNK designation

C3-CN: Tests or control functions depending on the needs of the particular DEVICE

---

39. With 21 millisecond acceleration and 10 millisecond deceleration, the central processor is not involved for a total of 31 milliseconds. This amount of time is automatically assigned to the central processor.

Transfer of 12 card rows occurs during the remaining 44 millisecond interval. However, central processor time of only .320 milliseconds is required to transfer one card row. Calculate the amount of central processor time used to transfer all 12 rows. 3.84 milliseconds

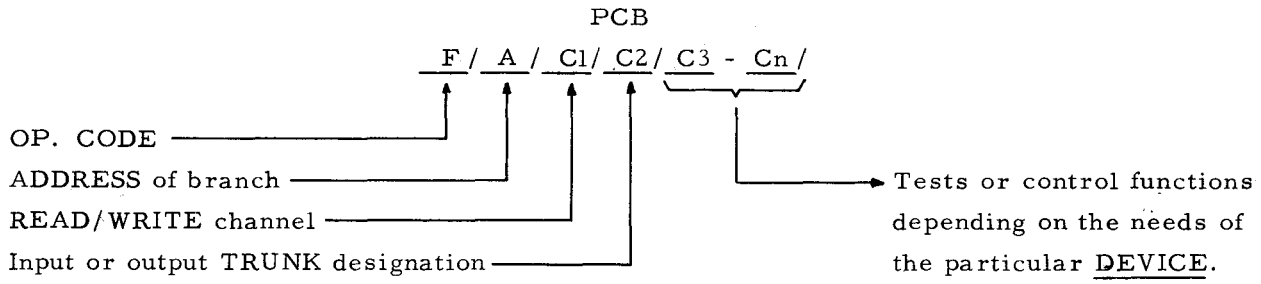
3.

INITIALIZE
BUSY
ERRORS
BRANCH

15.

C2 (trunk designation) first digit of 0 = OUTPUT
C2 (trunk designation) first digit of 4 = INPUT

27.



39.

3.84 MILLISECONDS

Note: For simplicity, this time will be referred to as "nearly 4 milliseconds" in subsequent frames.

4.

OP CODE	A ADDRESS	I/O CONTROL CHARACTERS C1	C2	C3 ⟶ Cn
[redacted]	[redacted]	[redacted]	[redacted]	[ ] [ ]

PDT — PERIPHERAL DATA TRANSFER (Printer)

The op. code of a PDT instruction sets up the operation to be performed. The A address shows either from which or to which main memory address transfer will occur. The Input/Output Control Characters C1 and C2 (variants) designate the desired read write channel and peripheral control unit respectively. Because the needs of devices differ, the number of INPUT / OUTPUT CONTROL CHARACTERS beyond C2 varies.

---

16. C2 designates the trunk number (from 0 to 3) with its second digit. Whether this is INPUT (4) to the central processor or OUTPUT (0) of the central processor is shown by the first digit of C2.

Write C2's showing central processor input and output for each trunk.

CP INPUT = 40, 41, 42, 43.

CP OUTPUT = 00, 01, 02, 03.

---

28. After a PDT instruction, it is necessary to test to see if the PDT has been completed or if the peripheral device is still busy. A PCB C3 of octal 10 tests busy. Remembering that no read/write channel needs to be tested, write a PCB instruction to branch on itself if the card reader attached to trunk #1 is still busy.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14 15	20 21	62
		PDT	CARDIN,51,41
		PCB	*,00,#1,10

---

40. The card reader performs mechanical movement and card reading for 75 milliseconds. The central processor is only involved for nearly 4 milliseconds to transfer information. Consequently, 71 out of 75 milliseconds is automatically granted to the central processor during a card read interval.

Assuming an average instruction execution time of 40 microseconds (.04 milliseconds), approximately how many instructions can be executed by the central processor during a card read? 1775

4.

INPUT/OUTPUT CONTROL CHARACTERS

16.

CP INPUT	40, 41, 42, 43
CP OUTPUT	00, 01, 02, 03

28.

In the PCB, * is the self reference for the PCB itself. Therefore, the program will wait until the device on trunk #41 is no longer busy (1Ø).

MAR	LOCATION	OPERATION CODE	OPERANDS
7	8 14	15 20	21 62
		PDT	CARDIN,51,41
		PCB	*,ØØ,41,1Ø

A C1 of ØØ means no RWC needs to be tested.

40.

1775 instructions with average execution times of 40 microseconds could be accomplished during a card read operation.

5. In format F/A/C1/C2/C3 - Cn/ of a peripheral data transfer instruction:

F refers to the mnemonic op. code PDT.

A is the ADDRESS either to or from which transfer occurs.

Control characters C1 and C2 (to be explained in the following frames) are found in every PDT instruction, regardless of the type of peripheral device being directed.

The number of control characters beyond C2 (That is, control characters C3 through CN) depends on the needs of the particular device.

---

17. The purposes of C1 and C2 should not be confused.

C1 specifies which READ / WRITE channel is desired and a first digit of 5 means that it is to be INTERLOCKED.

C2 designates which of the four input/output TRUNKS is to be used.

For C2, a first digit of 0 means OUTPUT of the CENTRAL PROCESSOR, a first digit of 4 means INPUT to the CENTRAL PROCESSOR.

---

29. A PCB with an * as the A address causes the program to branch and WAIT until the device being tested for busy has completed its PDT.

Write a branch on self PCB to check the busy status of the card punch below.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14 15	20 21	62
		PDT	PCHOUT,12,01
		PCB	*,00,01,10

---

41. The H-200 is much faster than the 1401 even if the H-200 is intentionally (and unnecessarily) caused to operate in a serial manner just for the sake of comparison. As you know, the 1401 operates serially. That is, a card may be read, then processed, then the next card is read. Even with processing overlap, the time available to the 1401 central processor does not compare to the time automatically granted to the H-200 central processor during a card read. Continue to the answer side of this frame.

5.

PDT
ADDRESS
C3 - Cn

17.

C1 = READ/WRITE channel.
51 is INTERLOCKED RWC #1.
C2 = Input/output TRUNKS.
OX = OUTPUT of the CENTRAL PROCESSOR.
4X = INPUT of the CENTRAL PROCESSOR.

(X could be any trunk from 0 to 3)

29.

WAIT

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14 15	20 21	62
		PDT	PCHOUT,12,Ø1
		PCB	*,ØØ,Ø1,1Ø

---

41. You will be given an illustration of H-200 simultanity at the end of this lesson, in which information from a card is processed, formatted, written on tape, and printed by the printer during a single card read interval. This simultaneity is accomplished while maintaining the full rated speed of card reading.

6. Format F/A/C1/C2/ is used for all PDT instructions. However, the number of CONTROL CHARACTERS beyond C2 will vary depending upon the needs of the particular device. C1, C2, and any additional C's are written as two digit octal variants. The number of these octal variants beyond C2 depends on the NEEDS of the particular DEVICE. For example, a printer and a tape unit require different types of control.

---

18. Suppose a card reader and a card punch are attached to the #1 pair of input output trunks. Write the C2 to designate the card reader. 41 Write the C2 to designate the card punch. 01

---

30. The PCB instruction in the preceding frame has the format F/A/C1/C2/C3/. Briefly list the purpose of each of its elements.

F/ (PCB) OP CODE

A/ (*) SELF REFERENCE A ADDRESS

C1/ (ØØ) DO NOT TEST ANY RWC

C2/ (Ø1) TEST DEVICE CONNECTED TO OUTPUT TRUNK #1

C3/ (1Ø) TEST IF DEVICE BUSY

---

42. The following coding shows two areas of memory being reserved to contain card reader information. The area tagged CRB1 (Card Reader Buffer #1) receives card input. Later, this information may be moved to CRB2 to permit the next card to enter CRB1.

MARK	LOCATION	OPERATION CODE	OPERANDS
	CRB1	RESV	8Ø
	CRB2	RESV	8Ø
		PCB	STOP,00,41,27,21,22

Write the instruction initializing the card reader on trunk #1 to branch to location STOP if inoperable. If operable, set to read Hollerith (27) and eject cards with hole count errors (21) or illegal punch (22).

6.

CONTROL CHARACTERS
NEEDS of the DEVICE

18.

CARD READER C2 = 41 because a card reader provides input (4) to the central processor.
CARD PUNCH C2 = 01 because a card punch requires output (0) from the central processor.

30.

F/	(PCB)	OP. CODE TO SET UP OPERATION.
A/	(*)	SELF REFERENCE A ADDRESS.
C1/	(ØØ)	DO NOT TEST ANY RWC.
C2/	(Ø1)	TEST DEVICE CONNECTED TO OUTPUT TRUNK #1.
C3/	(1Ø)	TEST FOR BUSY.

42.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 14	15 20	21 62
	CRB1	RESV	8Ø
	CRB2	RESV	8Ø
		END	
		PCB	STOP,ØØ,41,27,21,22

7. The writing of the op. code PDT and selection of the A address to which or from which transfer should occur needs little explanation. As mentioned previously, the H-200 is not limited to specific reserved processing areas as is the 1401. An H-200 programmer determines where, how many, what size, and how to tag the reserved PROCESSING AREAS he will use for the A ADDRESS in a PDT instruction.

---

19. With a card reader and card punch assigned to trunk #1, write the following:
Read a card into memory starting at location CARDIN using RWC #1 interlocked.
Punch a card with the information from the location tagged PCHOUT using RWC #2.

MARK	LOCATION	OPERATION CODE	OPERANDS
		PDT	CARDIN,51,#1
		PDT	PCHOUT,12,01

---

31. If conditions other than busy (Hole Count Error, Illegal Punch, etc.) are to be tested, they should be accomplished with another PCB. The control character for Hole Count Error is octal 41, for Illegal Punch it is octal 42. Write a PCB to branch to the location tagged ERROR1 if an HCE is found in the preceding card read on trunk below.

MARK	LOCATION	OPERATION CODE	OPERANDS
		PDT	CARDIN,51,41
		PCB	*,00,41,10
		PCB	ERROR1,00,41,41

---

43. Set a word mark at CRB1, then write a PDT to read a card into CRB1 using interlocked RWC#1, card reader attached to trunk #1.

MARK	LOCATION	OPERATION CODE	OPERANDS
	CRB1	RESV	80
	CRB2	RESV	80
		SW	
		PCB	STOP,00,41,27,21,22
		SW	CRB1
		PDT	CRB1,51,41

7.

PROCESSING AREAS

ADDRESS

PDT

A programmer specifies which read/write channel is to be used by constructing C1 after he has written the op. code and A address.

19.

MARK	LOCATION	OPERATION CODE	OPERANDS
		PDT	CARDIN,514,1
		PDT	PCHOUT,12,Ø1

31.

MARK	LOCATION	OPERATION CODE	OPERANDS
		PDT	CARDIN,51,41
		PCB	*,ØØ,41,1Ø
		PCB	ERROR1,ØØ,41,41

43.

MARK	LOCATION	OPERATION CODE	OPERANDS
	CRB1	RESV	8Ø
	CRB2	RESV	8Ø
		END	
		PCB	STOP,ØØ,41,27,21,22
		SW	CRB1
		PDT	CRB1,51,41

8. Because three read/write channels (four with RWC 1' option) are available, the first control character (C1) specifies which READ / WRITE CHANNEL is desired. Control character C1 is easily constructed in its two digit octal form. The rightmost digit specifies which of the three RWC's is desired. Given the leftmost octal digit, construct C1 for:

RWC #1 = 1 1 $_8$
RWC #2 = 1 2 $_8$
RWC #3 = 1 3 $_8$

---

20. A card read or card punch operation terminates when either of two situations is encountered.

1. All 90 columns have been read into or from memory.
2. A record mark is sensed in memory.

Establish the punctuation to terminate a card read with the fortieth column read into the area tagged CARDIN. Then, write a PDT to read a card using interlocked RWC #1 and the card reader on trunk #1.

TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
			SN	CARDIN+39
			SI	CARDIN+39
			PDT	CARDIN,51,41

---

32. Explain each element of the instruction PCB ERROR1, ØØ, 41, 41.

F/ (PCB) OP CODE
A/ (ERROR1) Address branch if
C1/ (ØØ) No - read/write channel for test
C2/ (41) output trunk 1 (test device attached)
C3/ (41) check for HCE or ILP

How does the computer differentiate between the control characters 41 and 41?

position in the instruction

---

44. Write the instruction to wait (branch on itself) and check the card reader for busy (10). Then, write an instruction to branch to the location tagged ERROR if HCE (41) or ILP (42).

MARK	LOCATION	OPERATION CODE	OPERANDS
	CRB1	RESV	8Ø
	CRB2	RESV	8Ø
		PCB	STOP,ØØ,41,27,21,22
		SW	CRB1
		PDT	CRB1,51,41
		PCB	*,00,41,10
		PCB	ERROR,00,41,41,42

8.

READ/WRITE CHANNELS

C1 for RWC #1 = $1\underline{1}_8$

RWC #2 = $1\underline{2}_8$

RWC #3 = $1\underline{3}_8$

20.

MARK	LOCATION	OPERATION CODE	OPERANDS
		SW	CARD1N+39
		S,1	CARD1,N+39
		PDT	CARD1,N,51,41

32.

F/ (PCB) OP. CODE TO SET UP THE OPERATION

A/ (ERROR1) ADDRESS FOR BRANCH

C1/ (ØØ) NO RWC IS TO BE TESTED

C2/ (41) TEST THE DEVICE ATTACHED TO INPUT (4) TRUNK #1 (1)

C3/ (41) TEST FOR A HOLE COUNT ERROR

POSITION IN THE INSTRUCTION AS EITHER C2 OR C3 DETERMINES WHETHER 41 DESIGNATES INPUT TRUNK #1 OR TEST CONTITION HCE.

(Or equivalent answer.)

44.

Note: The busy test has been tagged TEST for subsequent reference.

MARK	LOCATION	OPERATION CODE	OPERANDS
	CRB1	RESV	8Ø
	CRB2	RESV	8Ø
		SW	
		PCB	STOP,ØØ,41,27,21,22
		SW	CRB1
		PDT	CRB1,51,41
	TEST	PCB	*,ØØ,41,1Ø
		PCB	ERROR,ØØ,41,41,42

9. The optional fourth read/write channel (RWC #1') is designated with a second octal digit of 5. As such, C1 for RWC #1' is written 15$_8$.

When RWC #1' is available, it will alternate 2 micro-second memory cycles with RWC #1.

---

21. You should remember the distinction for a record mark when information is being read into or out of memory. When information is transferred into memory, transfer terminates with the memory location containing the record mark. When information is being transferred out of memory, transfer terminates at the memory location before the record mark.

Establish punctuation to terminate a card punch after 40 characters have been transferred from the area starting at address #201. Then, write a PDT to punch a card using RWC #2, card punch attached to trunk #1.

MARK	LOCATION	OPERATION CODE	OPERANDS
		SW	241
		SI	241
		PDT	12,01

---

33. PCB control characters of octal 10, 41, and 42, test a card reader for busy, Hole error ERROR and Illegal Punch (ILP) respectively. A PCB containing control character 10 as a test should be written separately from a PCB testing 41 and/or 42. What is the difference between the two forms of PCB shown below.

	F/	A/	C1/	C2/	C3/	C4/
Form #1.	PCB	ERROR	ØØ	41	41	42
Form #2.	PCB	ERROR1	ØØ	41	41	
	PCB	ERROR2	ØØ	41	42	

Form #1 - Branch to error if either, Hole or Illegal punch.
#2 " " error 1 " " " error 2

---

45. When a card has been read into CRB1, an LCA can be written to move CRB1 information into CRB2. Because the next card to be read has an acceleration interval of 10 milliseconds, there is more than enough time to retrieve the PDT, then perform the LCA of the first card. Write the PDT to read the next card into CRB1. Then, write an LCA to move the previous information from CRB1 +79 into CRB2 +79.

		SW	CRB1
		PDT	CRB1,51,41
	TEST	PCB	*,ØØ,41,1Ø

9.

C1 for RWC #1' = 15 (8)

21. #201 through #240 equals 40 characters. Therefore, the forty-first memory location is record marked.

MARK	LOCATION	OPERATION CODE	OPERANDS
		SW	241
		SI	241
		PDT	201,12,01

The format F/A/C1/C2/ contains all the control characters needed for a card read or punch. Other types of peripheral equipment use additional control characters. These are listed in the Programmers' Reference Manual to which you will refer later.

33.

HOLE COUNT ERROR

FORM #1 BRANCHES TO ONE LOCATION IF EITHER HCE OR ILP.

FORM #2 BRANCHES TO A LOCATION IF HCE, ANOTHER LOCATION IF ILP.

(Or equivalent answer.)

45. Note: If the LCA is written on line 9, acceleration time of the card will not be used to advantage.

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1				CRB1	RESV	80	
2				CRB2	RESV	80	
3					SW		
4					PCB	STOP,00,41,27,21,22	INITIALIZES CARD READER
5					SW	CRB1	SETS A FLD WM FOR SUBSEQUENT LCA
6					PDT	CRB1,51,41	READS CARD INTO CRB1
7				TEST	PCB	*,00,41,10	WAITS UNTIL CARD READ IS COMPLETE
8					PCB	ERROR,00,41,41,42	TESTS FOR HCE AND ILP
9					PDT	CRB1,51,41	STARTS NEXT READ INTO CRB1
10					LCA	CRB1+79,CRB2+79	MOVES PREVIOUS INFO TO CRB2

10. Availability of RWC #1' introduces a concept called "interlocking". It is possible to construct a C1 for RWC #1 that will exclude RWC #1' from sharing alternate memory cycles. "Interlocking" temporarily removes RWC #1' and permits RWC #1 to operate undistrubed.

Exclusion of RWC #1' is accomplished by INTERLOCKING RWC #1 with a specially constructed Control Character C1.

---

22. Now that you have written PDT instructions for a card read and a card punch, it is appropriate to discuss the instruction which tests to see if a PDT has been completed and if any errors were detected. What is the mnemonic op. code for the instruction used to either control or test a PDT? PCB

---

34. The preceding frames discussed PCB as used to test a device. PCB is also used to control (INITIALIZE) devices. A partial table of card reader initializing control characters is illustrated on the answer side of this frame.

---

46. Refer to the coding form in frame 45 and notice that the previous card information has been moved to card read buffer area #2 (CRB2). Another card read interval is just beginning into CRB1 for the PDT on line 10.

The coding to be written following line 10 will process the information now in CRB2. Because acceleration time is being used advantageously, more than 75 milliseconds are available to the central processor before a full card can enter CRB1.

If the coding on lines 9 and 10 were reversed, how much time would still be available to process CRB2 before a full card has been read into CRB1? _____ milliseconds.

10.

INTERLOCKING

Note: A C1 of 51_8 "interlocks" RWC #1.

The first digit excludes RWC #1'.

The second digit designates RWC #1.

A C1 of 51_8 is not to be confused with the C1 of 15_8 which designates RWC #1'.

22.

PCB

(Peripheral Control and Branch)

34.

Control Character (octal)	Function
	Control Functions
27	Branch if device inoperable. If operable, set control unit to read Hollerith code.
26	Branch if device inoperable. If operable, set control unit to read special code.
21	Branch if device inoperable. If operable, set control unit to reject cards with hole-count errors automatically.
22	Branch if device inoperable. If operable, set control unit to reject cards with illegal punches automatically.

46.

75 Milliseconds

75 milliseconds is equal to 37,500 memory cycles. As much processing as reasonably desired may be accomplished and still maintain the full rated speed of card reading. At the end of the desired processing, a branch is simply written to return to TEST and test for busy. This loop continues until the last card has been read and processed.

11. Does illustration A or B show interlocking of RWC #1 to exclude RWC #1'? B

RWC TIME SHARING (2 microsecond intervals)

A.	RWC #1	RWC #2	RWC #3	RWC #1'	RWC #2	RWC #3
B.	RWC #1	RWC #2	RWC #3	RWC #1	RWC #2	RWC #3

Write a CI for each RWC in illustration A. ✓

Write a CI for each RWC in illustration B. ✓

23. The shortest format of a PCB is when the only test to be performed concerns the status of a read/write channel. Asking, "Is such and such a read/write channel busy?" and providing the alternative, "If busy (performing some operation) branch to the location specified", is accomplished by the format F/A/C1/.

F/ is the OP CODE PCB.

A/ is the ADDRESS for the BRANCH.

C1/ specifies the READ / WRITE CHANNEL to be tested.

35. An initializing PCB is similar to manually setting switches and dials to control a device, except that initializing PCB's are accomplished with program instructions.

Format F/A/C1/C2/C3 - Cn/ is written for an initializing PCB, and RWC is designated (ØØ). Assume a card reader is attached to trunk #1, then refer to the preceding table to write an initializing PCB to read Hollerith code. Tag the instruction INIT, write the A address as a tag of HALT.

MARK	LOCATION	OPERATION CODE	OPERANDS
	INIT	PCB	HALT,0,0,41,27

47. Write the instruction to branch back to TEST after the desired processing.

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1			CRB1	RESV	8Ø
2			CRB2	RESV	8Ø
3					
4				PCB	STOP,ØØ,41,27,21,22
5				SW	CRB1
6				PDT	CRB1,51,41
7			TEST	PCB	*,ØØ,41,1Ø
8				PCB	ERROR,ØØ,41,41,42
9				PDT	CRB1,51,41
10				LCA	CRB1+79,CRB2+79
11					PROCESSING OF CRB2 INFORMATION

11. Illustration B shows interlocking for RWC #1 to exclude RWC #1' from alternate read/write cycles.

Without interlocking, C1 for RWC #1 = 11_8
RWC #2 = 12_8
RWC #3 = 13_8
RWC #1' = 15_8

Without interlocking, C1 for RWC #1 = 51_8 RWC #2 = 12_8 RWC #3 = 13_8

23.

OP. CODE
ADDRESS for the BRANCH
READ/WRITE CHANNEL

35.

MARK	LOCATION	OPERATION CODE	OPERANDS
7	8 ... 14	15 ... 20	21 ... 62
	INIT	PCB	HALT,ØØ,41,27

47.

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1			CRB1	RESV	8Ø	
2			CRB2	RESV	8Ø	
3						
4				PCB	STOP,ØØ,41,27,21,22	
5				SW	CRB1	
6				PDT	CRB1,51,41	
7			TEST	PCB	*,ØØ,41,1Ø	
8				PCB	ERROR,ØØ,41,41,42	
9				PDT	CRB1,51,41	
10				LCA	CRB1+79,CRB2+79	
11					PROCESSING OF CRB2 INFORMATION	
12						
13				B	TEST	

12. If necessary, refer to frame 11 to answer the following:

In a system without optional RWC #1', RWC #1 is granted 2 microseconds every 6 microseconds.

With RWC #1' and RWC #1 not interlocked, RWC #1 is granted 2 microseconds every 12 microseconds.

In a system with RWC #1', but RWC #1 interlocked, RWC #1 is granted 2 microseconds every 6 microseconds.

---

24. PCB format F/A/C1/ may be used in a "small" H-200 system having a limited number of peripheral devices. Each device may as well always be assigned a certain read/write channel, if the system does not contain more peripheral devices than read/write channels. For example, an H-200 system with only a card reader, card punch, and printer would have little need for the programmers' freedom of RWC reassignment. (Of course, the programmer could change RWC assignments if desired.)

Continue to the answer side of this frame.

---

36. In the case of a card reader initializing PCB, control character C3 must be octal 27 (Hollerith code). Initializing PCB's will branch to the A address if the device is inoperable. Write an initializing PCB for the following:

Branch to the location tagged STOP if the card reader attached to trunk #1 is inoperable. If operable, set to read Hollerith code and to automatically reject cards with hole count errors (21_8).

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8	14 15	20 21	62
		PCB	STOP,00,41,27,21

---

48. Briefly explain what is accomplished by each line of coding in frame 47.

Line #1 _____

Line #2 _____

Line #4 _____

Line #5 _____

Line #6 _____

Line #7 _____

Line #8 _____

Line #9 _____

Line #10 _____

Line #11+ _____

Last Line _____

12.

W/O RWC #1', RWC #1 is granted 2 microseconds every 6 microseconds.
With RWC #1', RWC #1 not interlocked, RWC #1 is granted 2 microseconds every 12 microseconds.
With RWC #1', RWC #1 interlocked, RWC #1 is granted 2 microseconds every 6 microseconds.

(Return to frame 13, page 275.)

24. If there is no need to change RWC assignements (to accommodate more peripheral devices) the same RWC may always be assigned to a particular device. In effect then, testing of a RWC channel for busy actually checks whether its associated device is busy, WHEN APPLIED TO A SMALL SYSTEM PCB. In a larger H-200 system (where multiple peripheral devices are accommodated by the programmers' freedom of RWC reassignment) PCB format F/A/C1/ only checks status of a RWC and does not imply the status of a particular device.

(Return to frame 25, page 275.)

36.

MARK	LOCATION	OPERATION CODE	OPERANDS
7 8 14		15 20	21 62
		PCB	STOP,ØØ,41,27,21

(Return to frame 37, page 275.)

48.

Line #1 RESERVE 80 MEMORY LOCATIONS, TAGGING THE LEFTMOST CRB1.
Line #2 RESERVE 80 MEMORY LOCATIONS, TAGGING THE LEFTMOST CRB2.
Line #4 INITIALIZE CARD READER (TRUNK #1), HOLLERITH, REJECT HCE & ILP.
Line #5 SET WM IN CRB1 FOR THE A FIELD OF THE SUBSEQUENT LCA.
Line #6 READ CARD INTO CRB1 USING INTERLOCKED RWC #1, TRUNK #1.
Line #7 WAIT IF CARD READER BUSY.
Line #8 TEST FOR HOLE COUNT ERROR OR ILLEGAL PUNCH.
Line #9 STARTS NEXT CARD READ.
Line #10 MOVE PREVIOUS INFORMATION FROM CRB1 TO CRB2.
Line #11+ PROCESS CRB2 FOR UP TO 75+ MILLISECONDS.
Last Line BRANCH TO TEST CARD READER FOR BUSY (LINE 9 PDT).

(Continue to page 300.)

SIMULTANEITY AND DOUBLE BUFFERING

The preceding PDT and PCB frames explained an operation involving only one read/write channel and a single peripheral unit. The concept of double buffering was also explained. A card was read into a card read area then moved to a card read buffer area (⟶CRB1--⟶CRB2). This provided more than 75 milliseconds to process the information from CRB2 while the next card entered CRB1.

On this and the following page, the principle of double buffering is expanded. Three peripheral units (card reader, tape unit, and printer) are operated simultaneously on three read/write channels and roughly 60 milliseconds of processing time is provided while maintaining full speed of card reading!

The coding form below establishes two buffer areas in memory for each peripheral unit. Punctuation of these areas is not detailed in this example. Read the coding and notice that it is EXECUTED, then continue to page 301.

EASYCODER
CODING FORM

PROBLEM SIMULTANEOUS CARD READ, TAPE WRITE, & PRINT PROGRAMMER T. ELLIOTT DATE 15 JULY 1964 PAGE 1 OF 3

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1	0 1 0 1 0				PROG	DBLBUF	
2							
3					ADMODE	2	
4							
5					CAM	2	
6							
7				CRB1	RESV	80	
8							
9				CRB2	RESV	80	
10							
11				TPB1	RESV	60	
12							
13				TPB2	RESV	60	
14							
15				PRB1	RESV	120	
16							
17				PRB2	RESV	120	
18							
19				INIT		} DATA FORMATING, PUNCTUATION, ETC.	
20							
21							
22					PCB	*,00,00,21 REWIND TAPE DRIVE 1	
23							
24					PCB	*,00,41,27,21 SET CARD READER TO HOLLERITH & TO REJECT	
25							
26					PDT	*,03,02,57 SKIP TO HEAD OF FORM ON PRINTER	
27							
28					B	BOOT (BRANCH TO LOADING ROUTINE AFTER INIT EX)	
29							
30					EX	INIT (EXECUTE FROM INIT THROUGH B BOOT)	

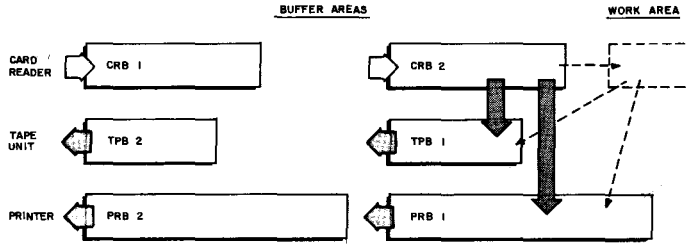
Following execution of the coding on the preceding page, the six buffer areas CRB1, CRB2, TPB1, TPB2, PRB1, PRB2 are reserved in memory. The coding below is then overlaid on the preceding coding for more efficient utilization of memory. Instructions on this coding form are assembled and the operations illustrated below the form take place. Review the form and illustration.

Note: Even including additional coding for the tape unit and printer, nearly 60 milliseconds are available as PROCESSING TIME with card reading at full rated speed!

EASYCODER
CODING FORM

PROBLEM SIMULTANEOUS CARD READ, TAPE WRITE, & PRINT PROGRAMMER T. ELLIOTT DATE 15 JULY 1964 PAGE 2 OF 3

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
02010				ORG	INIT	OVERLAY PREVIOUSLY EXECUTED SEGMENT
			START	PDT	CRB1,51,41	READ CARD INTO CARD READ BUFFER #1
			TEST	PCB	*,00,41,10	WAIT ON CARD READER BUSY
				PCB	ERROR,00,41,41	TEST FOR CARD READ ERROR (HCE)
				PDT	CRB1,51,41	START NEXT CARD READ
				LCA	CRB1+79,CRB2+79	DURING ACCELERATION, MOVE PRIOR CRB1 TO CRB2
			PROCESSING TIME			
				PCB	*,00,00,01	WAIT ON TAPE WRITE BUSY
				PCB	ERROR2,00,00,41	TEST FOR TAPE WRITE ERROR
				LCA	TPB1+59,TPB2+59	MOVE DATA TO TAPE BUFFER #2
				PDT	TPB2,02,00,61	WRITE TAPE FROM TPB2
				PCB	*,00,02,10	WAIT ON PRINTER BUSY
				PCB	ERROR3,00,02,40	TEST FOR PRINTER ERROR
				LCA	PRB1+119,PRB2+119	MOVE DATA TO PRINT BUFFER #2
				PDT	PRB2,03,02,21	PRINT FROM P B2
				B	TEST	BRANCH TO TEST CARD READER BUSY



Lesson VIII explained input/output operations as related to card reading and punching. Coding for the other peripheral units PDT's and PCB's is explained in the Honeywell 200 Programmers' Reference Manual DSI-214A.

As an exercise, you may refer tape unit or printer PDT's and PCB's illustrated on the preceding pages to their appropriate explanations in the INPUT/OUTPUT section of the Programmers' Reference Manual.

The table of contents for this programmed text lists frame and page numbers of instructions presented in Lessons VII and VIII.

A table of respective page numbers for the Programmers' Reference Manual is given on page 194.

NOTES

NOTES

HONEYWELL
ELECTRONIC
DATA
PROCESSING
WELLESLEY HILLS,
MASSACHUSETTS 02181