University of British Columbia
Electrical and Computer Engineering
ELEC291/ELEC292

Introduction to the AT89LP51RC2 Microcontroller

Dr. Jesús Calviño-Fraga P.Eng.
Department of Electrical and Computer Engineering, UBC
Office: KAIS 3024
E-mail: jesusc@ece.ubc.ca
Phone: (604)-827-5387

Objectives

- Introduction to the AT89LP51RC2 microcontroller.
- Introduction to the 8051 instruction set.
- Understand how an assembler compiler is used and the files it generates.
- Calculate machine code execution time.
- Assemble a bread boarded AT89LP51RC2 microcontroller with LCD.
- Write, compile, download, and run programs into the AT89LP51RC2/LCD microcontroller system.

AT89LP51RC2

Features

- 8-bit Microcontroller Compatible with 8051 Products
- Enhanced 8051 Architecture
  - Single Clock Cycle per Byte Fetch
  - 12 Clock per Machine Cycle Compatibility Mode
  - Up to 20 MIPS Throughput at 20 MHz Clock Frequency
  - Fully Static Operation: 0 Hz to 20 MHz
  - On-chip 2-cycle Hardware Multiplier
  - 16x16 Multiply–Accumulate Unit
  - 256 x 8 Internal RAM
  - On-chip 1152 Bytes Expanded RAM (ERAM)
    - Software Selectable Size (0, 256, 512, 768, 1024 or 1152 Bytes)
  - Dual Data Pointers
  - 4-level Interrupt Priority
- Nonvolatile Program and Data Memory
  - 24KB/32KB of In-System Programmable (ISP) Flash Program Memory
  - 512-byte User Signature Array
  - Endurance: 10,000 Write/Erase Cycles
  - Serial Interface for Program Downloading
  - 2KB Boot ROM Contains Low Level Flash Programming Routines and a Default Serial Bootloader

AT89LP51RC2

- Peripheral Features
  - Three 16-bit Enhanced Timer/Counters
  - Seven 8-bit PWM Outputs
  - 16-bit Programmable Counter Array
    - High Speed Output, Compare/Capture
    - Pulse Width Modulation, Watchdog Timer Capabilities
  - Enhanced UART with Automatic Address Recognition and Framing Error Detection
  - Enhanced Master/Slave SPI with Double-buffered Send/Receive
  - Two Wire Interface 400K bit/s
  - Programmable Watchdog Timer with Software Reset
  - 8 General-purpose Interrupt and Keyboard Interface Pins
- Special Microcontroller Features
  - Dual Oscillator Support: Crystal, 32 kHz Crystal, 8 MHz Internal (AT89LP51IC2)
  - Two-wire On-Chip Debug Interface
  - Brown-out Detection and Power-on Reset with Power-off Flag
  - Selectable Polarity External Reset Pin
  - Low Power Idle and Power-down Modes
  - Interrupt Recovery from Power-down Mode
  - 8-bit Clock Prescaler

AT89LP51RC2

- I/O and Packages
  - Up to 40 Programmable I/O Lines
  - Green (Pb/Halide-free) PLCC44, VQFP44, QFN44. PDIP40
  - Configurable I/O Modes
    - Quasi-bidirectional (80C51 Style), Input-only (Tristate)
    - Push-pull CMOS Output, Open-drain
- Operating Conditions
  - 2.4V to 5.5V V_{CC} Voltage Range
  - -40° C to 85°C Temperature Range
  - 0 to 20 MHz @ 2.4V–5.5V (Single-cycle)

Somewhere in the Datasheet it says that it can run at up to 24 MHz, that is why I selected a 22.1184MHz crystal.

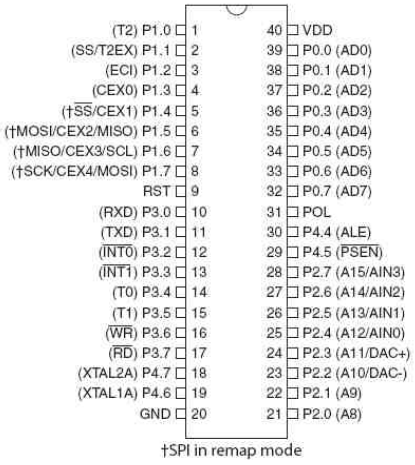
AT89LP51RC2 Features

- Industry standard. The 8051 is still a very common processor.
- Very similar to original 8051/8052 from the 1980's. Uses the well know 8051 instruction set. Can be made backward compatible.
- DIP! Easy to breadboard.
- 34 I/O pins. Also lots of useful peripherals.
- External memory interface. Rare nowadays.
- Lots of tools and resources.
- Not too complicated. User manual is "only" 254 pages.
- 1.7732 CA$ each in quantities of 100.

AT89LP51RC2 Draw Backs

- The built-in ADC (Analog to Digital Converter) is atrocious! Useless for ELEC291/292. We will be using an external ADC.
- 8-bit processor. Well, this is an advantage sometimes!

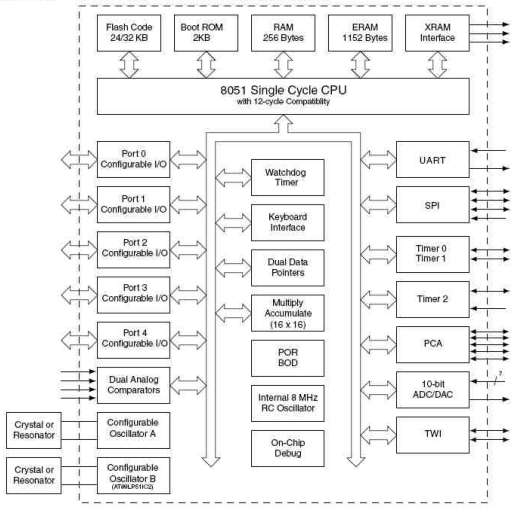
AT89LP51RC2 Pin Out



†SPI in remap mode

AT89LP51RC2 block diagram

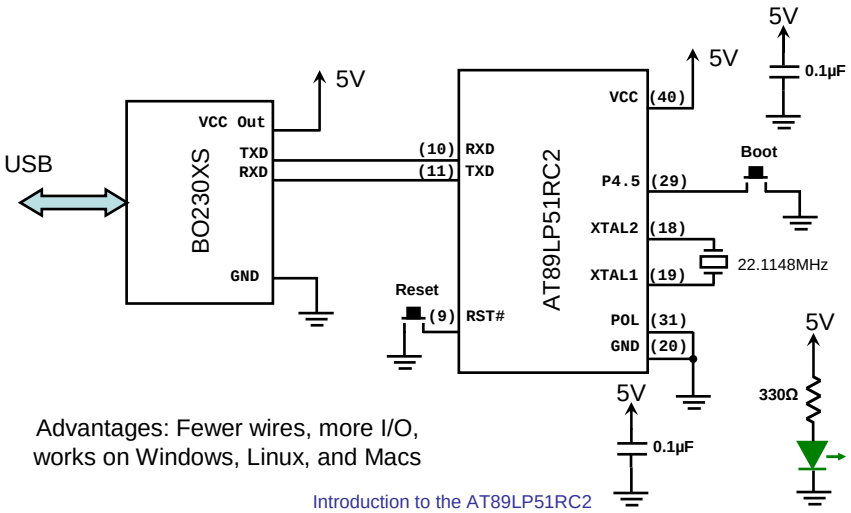
Figure 2-1. Atmel AT89LP51RB2/RC2/IC2 Block Diagram



Warning: it uses different instructions to access the different memory spaces!

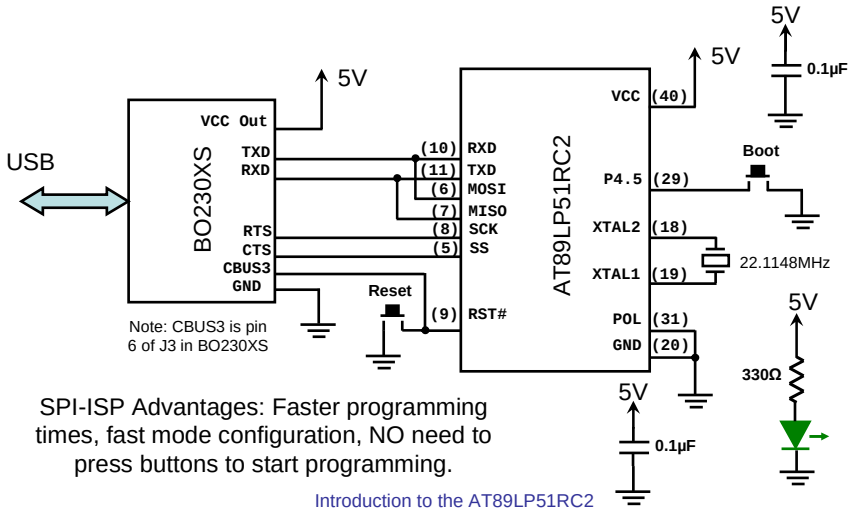
Figure taken from AT89LP51RC2 user manual

Assembling the AT89LP51RC2 Basic System. Using Boot Loader only.



Advantages: Fewer wires, more I/O, works on Windows, Linux, and Macs

Assembling the AT89LP51RC2 ISP System. Using SPI or Boot Loader.



SPI-ISP Advantages: Faster programming times, fast mode configuration, NO need to press buttons to start programming.

BO230XS J3 and J1 Pins

J3

PIN	Name
1	VCC
2	CBUS2
3	GND
4	CBUS1
5	CBUS0
6	CBUS3

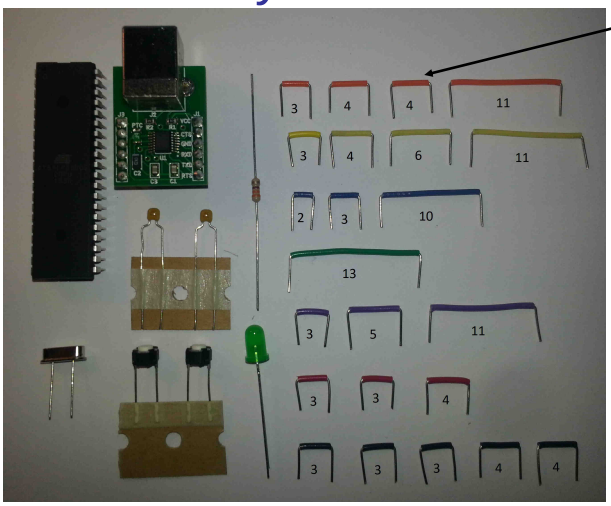
J1

PIN	Name
1	VCC
2	CTS
3	GND
4	RXD
5	TXD
6	RTS

BOM (Bill of Materials)

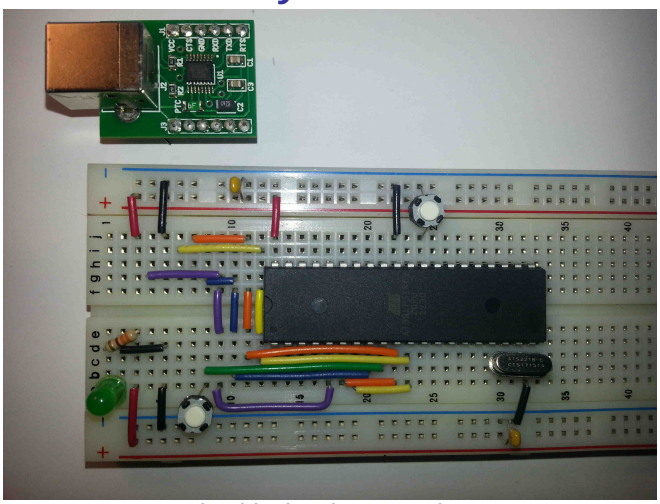
Quantity	Description
2	0.1uF capacitors
1	330Ω resistor
1	LED 5MM GREEN
1	22.1184MHZ Crystal
1	BO230XS serial USB adapter
1	AT89LP51RC2 IC 80C51 MCU 32K FLASH 40-DIP
2	Push button switch

Assembling the AT89LP51RC2 System



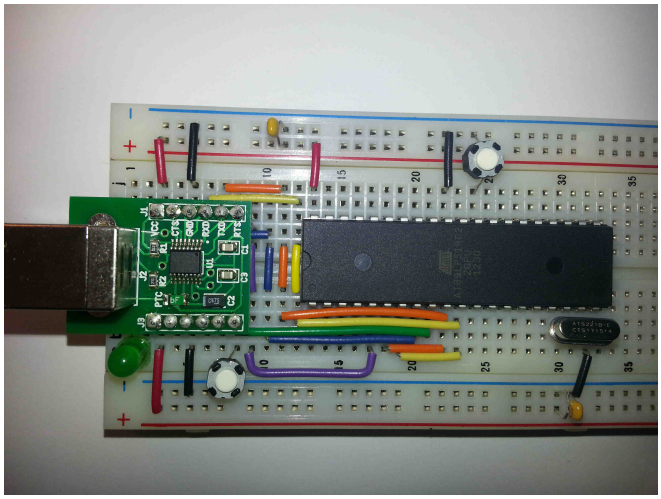
Wire length: 4 means 4 x 0.1 inches. 0.1 inches is the distance between adjacent holes in the breadboard.

Assembling the AT89LP51RC2 System



It should take about 30 minutes!

Assembling the AT89LP51RC2 System

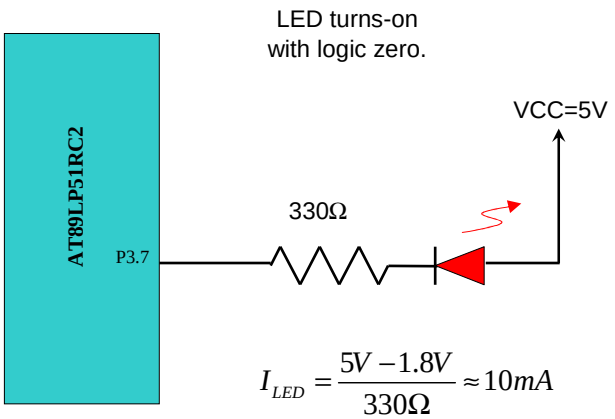


BO230XS

- Very simple circuit that allows to connect to many different types of microcontrollers.
- Adds serial port connection to computer via USB.
- Allows for In System Programming (ISP) for many microcontrollers. Some examples: The whole AT89LP family from Atmel/Microchip, The C8051/EFM8 family from Silabs, Some STM32F ARMs from ST Microelectronics, the LPC824 ARM from NXP, all PIC32MX from Microchip, the ATMega328 (and other members of the family) from Atmel/Microchip, and some MSP430 from Texas Instruments.

Add LED/Resistor to P3.7

LED voltage drop
@ 10mA (approx):
IR=1.4V
Red=1.8V
Yellow=1.9V
Green=2.0V
Blue=2.9V



$$I_{LED} = \frac{5V - 1.8V}{330\Omega} \approx 10mA$$

Blinky

```
; Blinky.asm: toggles an LED attached to pin 17 (P3.7)
$MODLP51

org 0000H
    ljmp myprogram

; When using a 22.1184MHz crystal in fast mode
; one cycle takes 1.0/22.1184MHz = 45.21123 ns
WaitHalfSec:
    mov R2, #89
L3: mov R1, #250
L2: mov R0, #166
L1: djnz R0, L1 ; 3 cycles->3*45.21123ns*166=22.51519us
    djnz R1, L2 ; 22.51519us*250=5.629ms
    djnz R2, L3 ; 5.629ms*89=0.5s (approximately)
    ret

myprogram:
    mov SP, #7FH
M0:
    cpl P3.7
    lcall WaitHalfSec
    sjmp M0
END
```


Compile “Blinky.asm” and flash “Blinky.hex”.

- Download and install CrossIDE from the course web page.
- Start CrossIDE and create a new assembly file. copy and paste the code from previous slides. Save as “Blinky.asm”.
- Click Build->ASM51. Click the “Browse” button beside “Complete path of assembler” and select a51.exe from the “Call51\bin” folder where you installed CrossIDE. Then press “Ok”.

Flash and run into the AT89LP51RC2 using Boot Loader

- While pressing button “BOOT”, press and release button “RESET”.
- On Crosside, click “fLash”->”Atmel 89LP Bootloader”. Make sure the file “Blinky.hex” is selected. Also make sure that the “Port” box is set to the USB port # assigned to the BO230XS. Click the “Flash” button and wait until it finishes.
- Press and release “RESET”. The program starts running!

Flash and run into the AT89LP51RC2 using SPI-ISP

- On Crosside, click “fLash”->”Atmel 89LP Using SPI”. Make sure the file “Blinky.hex” is selected. Click the “Flash” button and wait until it finishes. Done!

(Works only on Windows)

Summary: Blinky

- To run “Blinky” in the AT89LP51RC2 we need to complete the following steps:
  1. Assemble the AT89LP51RC2 Microcontroller system in a bread board. Attach LED to P3.7. (Or any other available pin, just change the code!)
  2. Edit and Compile “Blinky.asm”.
  3. Flash and run “Blinky.hex” into the AT89LP51RC2 processor.

8051 Instruction Set

1. Arithmetic Operations
2. Logical Operations
3. Data Transfer
4. Boolean Variable Manipulation
5. Program Branching and Machine Control

WARNING: I did my best to change the number of cycles per instruction to the actual values for the AT89LP51RC2, but there may be errors. The ultimate reference is the AT89LP51RC2 user manual!

Note: not showing AT89LP51RC2 extended instructions.

Arithmetic Operations (ADD)

Mnemonic	Opcode	B	C	Function
ADD A, Rn	28H-2FH	1	1	(A) = (A) + (Rn)
ADD A, direct	25H	2	2	(A) = (A) + (direct)
ADD A, @Ri	26H-27H	1	2	(A) = (A) + ((Ri))
ADD A, #data	24H	2	2	(A) = (A) + #data
ADDC A, Rn	38H-3FH	1	1	(A) = (A) + (C) + (Rn)
ADDC A, direct	35H	2	2	(A) = (A) + (C) + (direct)
ADDC A, @Ri	36H-37H	1	2	(A) = (A) + (C) + ((Ri))
ADDC A,#data	34H	2	2	(A) = (A) + (C) + #data

Arithmetic Operations (SUB)

Mnemonic	Opcode	B	C	Function
SUBB A, Rn	98H-9FH	1	1	(A) = (A) - (C) - (Rn)
SUBB A, direct	95H	2	2	(A) = (A) - (C) - (direct)
SUBB A, @Ri	96H-97H	1	2	(A) = (A) - (C) - ((Ri))
SUBB A, #data	94H	2	2	(A) = (A) - (C) - #data

Arithmetic Operations (INC)

Mnemonic	Opcode	B	C	Function
INC A	04H	1	1	(A) = (A) + 1
INC Rn	08H-0FH	1	1	(Rn) = (Rn) + 1
INC direct	05H	2	2	(direct) = (direct) + 1
INC @Ri	06H-07H	1	2	((Ri)) = ((Ri)) + 1
INC DPTR	A3H	1	2	(DPTR) = (DPTR) + 1

Arithmetic Operations (DEC)

Mnemonic	Opcode	B	C	Function
DEC A	14H	1	1	(A) = (A) - 1
DEC Rn	18H-1FH	1	1	(Rn) = (Rn) - 1
DEC direct	15H	2	2	(direct) = (direct) -1
DEC @Ri	16H-17H	1	2	((Ri)) = ((Ri)) - 1

Arithmetic Operations (Other)

Mnemonic	Opcode	B	C	Function
MUL AB	A4H	1	2	(B15–8), (A7–0) = (A) x (B)
DIV AB	84H	1	4	(B15–8), (A7–0) = (A) / (B)
DA A	D4H	1	1	Converts (A) to BCD

Logic Operations (AND)

Mnemonic	Opcode	B	C	Function
ANL A, Rn	58H-5FH	1	1	(A) = (A) AND (Rn)
ANL A, direct	55H	2	2	(A) = (A) AND (direct)
ANL A, @Ri	56H-57H	1	2	(A) = (A) AND ((Ri))
ANL A, #data	54H	2	2	(A) = (A) AND #data
ANL direct, A	52H	2	2	(direct) = (direct) AND A
ANL direct, #data	53H	3	3	(direct) = (direct) AND #data

Logic Operations (OR)

Mnemonic	Opcode	B	C	Function
ORL A, Rn	48H-4FH	1	1	(A) = (A) OR (Rn)
ORL A, direct	45H	2	2	(A) = (A) OR (direct)
ORL A, @Ri	46H-47H	1	2	(A) = (A) OR ((Ri))
ORL A, #data	44H	2	2	(A) = (A) OR #data
ORL direct, A	42H	2	2	(direct) = (direct) OR (A)
ORL direct, #data	43H	3	3	(direct) = (direct) OR #data

Logic Operations (XOR)

Mnemonic	Opcode	B	C	Function
XRL A, Rn	68H-6FH	1	1	(A) = (A) XOR (Rn)
XRL A, direct	65H	2	2	(A) = (A) XOR (direct)
XRL A, @ Ri	66H-67H	1	2	(A) = (A) XOR ((Ri))
XRL A, #data	64H	2	2	(direct) = (A) XOR #data
XRL direct, A	62H	2	2	(direct) = (direct) XOR (A)
XRL direct, #data	63H	3	3	(direct) = (direct) XOR #data

Logic Operations (Other)

Mnemonic	Opcode	B	C	Function
CLR A	E4H	1	1	(A) = 0
CPL A	F4H	1	1	(A) = (A')
RL A	23H	1	1	Rotate (A) Left
RLC A	33H	1	1	Rotate (A) Left Through Carry
RR A	03H	1	1	Rotate (A) Right
RRC A	13H	1	1	Rotate (A) Right Through Carry
SWAP A	C4H	1	1	(A3-0) ↔ (A7-4)

Data Transfer (MOV 1 of 3)

Mnemonic	Opcode	B	C	Function
MOV A, Rn	E8H-EFH	1	1	(A) = (Rn)
MOV A, direct	E5H	2	2	(A) = (direct)
MOV A, @Ri	E6H-E7H	1	1	(A) = ((Ri))
MOV A, #data	74H	2	2	(A) = #data
MOV Rn, A	F8H-FFH	1	1	(Rn) = (A)
MOV Rn, direct	A8H-AFH	2	2	(Rn) = (direct)
MOV Rn, #data	78H-7FH	2	2	(Rn) = #data
MOV direct, A	F5H	2	2	(direct) = (A)
MOV direct, Rn	88H-8FH	2	2	(direct) = (Rn)

Data Transfer (MOV 2 of 3)

Mnemonic	Opcode	B	C	Function
MOV direct1, direct2	85H	3	3	(direct1) = (direct2) (source) (destination)
MOV direct, @Ri	86H-87H	2	2	(direct) = ((Ri))
MOV direct, #data	75H	3	3	(direct) = #data
MOV @Ri, A	F6H-F7H	1	1	((Ri)) = A
MOV @Ri, direct	A6H-A7H	2	2	((Ri)) = (direct)
MOV @Ri, #data	76H-77H	2	2	((Ri)) = #data
MOV DPTR, #data16	90H	3	3	(DPTR) = #data15-0 (DPH) = #data15-8 (DPL) = #data7-0
MOVC A, @A + DPTR	93H	1	3	(A) = ((A) + (DPTR))

Data Transfer (MOV 3 of 3)

Mnemonic	Opcode	B	C	Function
MOVC A, @A + PC	83H	1	4	(A) = ((A) + (PC))
MOVX A, @Ri	E2H-E3H	1	2	(A) = ((Ri))
MOVX A, @DPTR	E0H	1	2	(A) = ((DPTR))
MOVX @Ri, A	F2H-F3H	1	1	((Ri)) = (A)
MOVX @DPTR,A	F0H	1	1	((DPTR)) = (A)

Data Transfer (Other)

Mnemonic	Opcode	B	C	Function
PUSH direct	C0H	2	2	(SP) = (SP) + 1 ((SP)) = (direct)
POP direct	D0H	2	2	(direct) = ((SP)) (SP) = (SP) - 1
XCHA, Rn	C8H-CFH	1	1	(A) = (Rn)
XCHA, direct	C5H	2	2	(A) = (direct)
XCHA, @Ri	C6H-C7H	1	2	(A) = ((Ri))
XCHD A, @Ri	D6H-D7H	1	2	(A3-0) = ((Ri3-0))

Boolean Variable Manipulation

Mnemonic	Opcode	B	C	Function
CLR C	C3H	1	1	(C) = 0
CLR bit	C2H	2	2	(bit) = 0
SETB C	D3H	1	1	(C) = 1
SETB bit	D2H	2	2	(bit) = 1
CPL C	B3H	1	1	(C) = (C)
CPL bit	B2H	2	2	(bit) = (bit)
ANL C, bit	82H	2	2	(C) = (C) AND (bit)
ANL C, bit’	B0H	2	2	(C) = (C) AND (bit’)
ORL C, bit	72H	2	2	(C) = (C) OR (bit)
ORL C, bit’	A0H	2	2	(C) = (C) OR (bit’)
MOV C, bit	A2H	2	2	(C) = (bit)
MOV bit, C	92H	2	2	(bit) = (C)

Program Branching (1 of 3)

Mnemonic	Opcode	B	C	Function
ACALL addr 11	AAA10001	2	3	(PC) = (PC) + 2 (PC10-0) = page address
LCALL addr 16	12H	3	4	(PC) = addr15-0
RET	22H	1	4	Return from ACALL/CALL
RETI	32H	1	4	Return from interrupt
AJMP addr 11	AAA00001	2	3	(PC) = (PC) + 2 (PC10-0) = page address
LJMP addr 16	02H	3	4	(PC) = addr15-0
SJMP rel	80H	2	3	(PC) = (PC) + 2 (PC) = (PC) + rel
JMP @A + DPTR	73H	1	2	(PC) = (A) + (DPTR)

Program Branching (2 of 3)

Mnemonic	Opcode	B	C	Function
JZ rel	60H	2	2/3	(PC) = (PC) + 2 IF (A) = 0 THEN (PC) = (PC) + rel
JNZ rel	70H	2	2/3	(PC) = (PC) + 2 IF (A) ≠ 0 THEN (PC) = (PC) + rel
JC rel	40H	2	2/3	(PC) = (PC) + 2 IF (C) = 1 THEN (PC) = (PC) + rel
JNC rel	50H	2	2/3	(PC) = (PC) + 2 IF (C) ≠0 THEN (PC) = (PC) + rel
JB bit, rel	20H	3	3/4	(PC) = (PC) + 3 IF (bit) = 1 THEN (PC) = (PC) + rel
JNB bit, rel	30H	3	3/4	(PC) = (PC) + 3 IF (bit) = 0 THEN (PC) = (PC) + rel
JBC bit, rel	10H	3	3/4	(PC) = (PC) + 3 IF (bit) = 1 THEN (bit) = 0 and (PC) = (PC) + rel

Program Branching (3 of 3)

Mnemonic	Opcode	B	C	Function
CJNE A, direct, rel	B5H	3	4	Compare and jump if not equal rel
CJNE A, #data, rel	B4H	3	4	Compare and jump if not equal rel
CJNE Rn, #data, rel	B8H-BFH	3	4	Compare and jump if not equal rel
CJNE @Ri, #data, rel	B6H-B7H	3	4	Compare and jump if not equal rel
DJNZ Rn, rel	D8H-DFH	3	3	Decrement and Jump if not zero
DJNZ direct,rel	D5H	3	4	Decrement and Jump if not zero
NOP	00H	1	1	(PC) = (PC) + 1

The 8051 Assembly Language Instruction

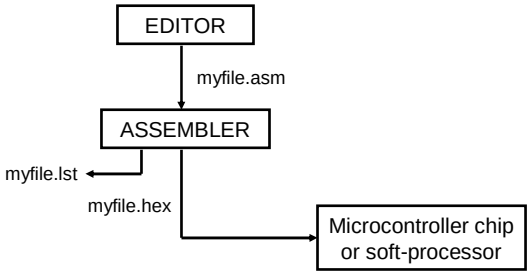
- **An Assembly language instruction consists of four fields:**

  [label :] mnemonic [operands] [;comment]

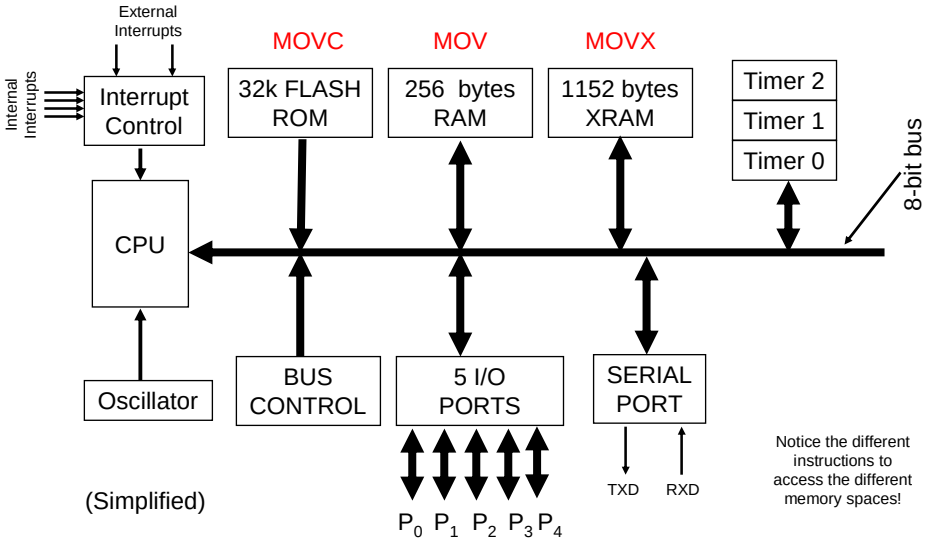
- **Examples:**

```
L1:  MOV A, #'*' ; Load Acc with ASCII for '*'
     NOP
     ; We can just have comments!
L3: ; We can also place labels anywhere
     DJNZ R0, L3
```

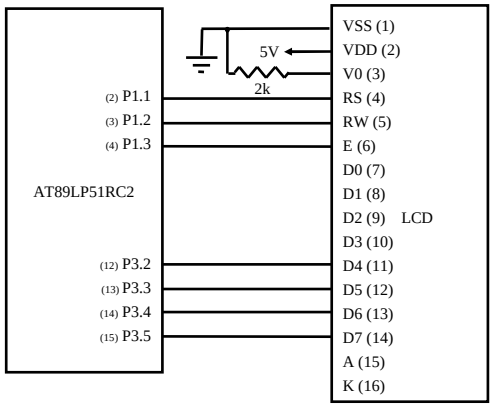
Writing and compiling an 8051 assembly language program



AT89LP51RC2 Block Diagram

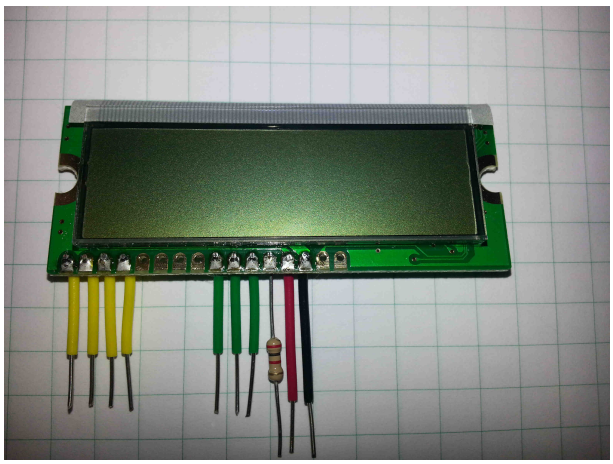


Adding an LCD

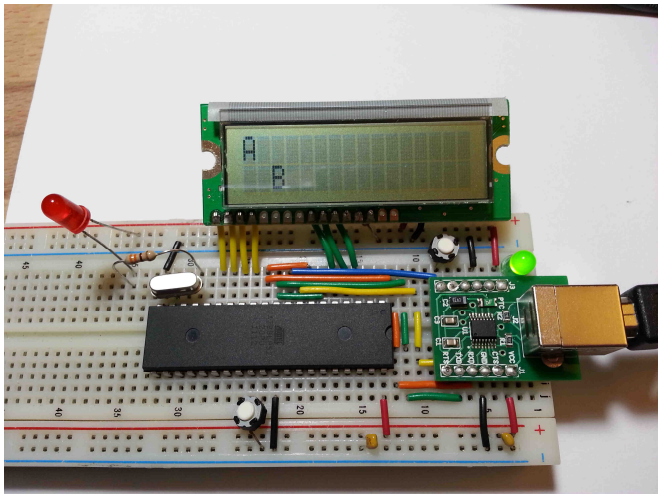


- Example configures LCD in 4-bit mode. Saves pins and wires!
- Standard Hitachi HD44780 controller
- Pin assignments arbitrary, but source code must match wiring
- Pins 7, 8, 9, 10, 15, 16 not connected.

Soldering Wires to LCD



AT89LP51RC2/LCD



Lab 1 Demo

- 'LCD_test_4bit.asm': this a test program for the AT89LP51RC2/LCD with the pins as shown in previous slides. Test program prints 'A' in column 1 of row 1, and 'B' in column 3 of row 2.
- Write your name (or a short version of it!) on row 1. Write your student number in row 2.
- Add extra functionality/features for extra marks.
- This lab could take you a long time to complete. Start early!
- Be as neat as possible. You'll be using the AT89LP51RC2/LCD system for the next three weeks!

Save your ICs!

- Integrated circuits (ICs) are delicate devices that can be damaged easily by:
  - Applying an inappropriate voltage.
  - Connecting live signals to the IC when not power up.
  - Electrostatic discharges. This happens frequently!

Chairs Available in labs

Electrostatic Discharges and The Blue/Gray Chairs of Death!

- Some of the chairs in MCLD labs are blue/gray.
- The blue/gray chairs are NOT really designed for an electronics lab use: they don't dissipate electrostatic charge. The black foamy chairs are designed for lab use: they dissipate electrostatic charge.
- If you are wearing a sweater, the air is dry, and you are seating in a blue chair you will likely get charged. If you discharge through your circuit it may permanently damage the ICs. Solution: before touching your circuit, touch the grounded metal rack in the work bench.