

Not Another Microprocessor Emulator User Manual

Revision 1
20 May 2019

Full Opcode Reference

SINGLE

0000	0000	0000	0000	NOP	
0000	0000	0000	0001	RST	RING 0
0000	0000	0000	0010	HLT	
0000	0000	0000	0011	RET	
0000	0000	0000	0100	RTI	RING 0
0000	0000	0000	1xxx	INT x	

SINGLE REGISTER

0000	0000	0001	nnnn	CLR	$R[n] \leftarrow 0$
0000	0000	0010	nnnn	INC	$R[n] = R[n] + 1$
0000	0000	0011	nnnn	DEC	$R[n] = R[n] - 1$
0000	0000	0100	nnnn	NOT	$R[n] = !R[n]$
0000	0000	0101	nnnn	BRA	$PC = PC + 2 * R[n]$
0000	0000	0110	nnnn	BNE	$PC \leftarrow R[n]$ if !E
0000	0000	0111	nnnn	BRE	$PC \leftarrow R[n]$ if E
0000	0000	1000	nnnn	BLE	$PC \leftarrow R[n]$ if L
0000	0000	1001	nnnn	BGE	$PC \leftarrow R[n]$ if !L
0000	0000	1010	nnnn	BLT	$PC \leftarrow R[n]$ if Z
0000	0000	1011	nnnn	BGT	$PC \leftarrow R[n]$ if !Z
0000	0000	1100	nnnn	PSH	$m[SP] \leftarrow R[n]$
0000	0000	1101	nnnn	POP	$R[n] \leftarrow m[SP]$
0000	0000	1110	nnnn	CAL	$RL = PC + 1, PC = R[n]$

DOUBLE REGISTER

0000	0001	xxxx	yyyy	ADD	$R[x] = R[x] + R[y]$
0000	0010	xxxx	yyyy	SUB	$R[x] = R[x] - R[y]$
0000	0011	xxxx	yyyy	LSL	$R[x] = R[x] \ll R[y]$
0000	0100	xxxx	yyyy	RSR	$R[x] = R[x] \gg R[y]$
0000	0101	xxxx	yyyy	AND	$R[x] = R[x] \& R[y]$
0000	0110	xxxx	yyyy	ORR	$R[x] = R[x] R[y]$
0000	0111	xxxx	yyyy	EOR	$R[x] = R[x] \wedge R[y]$
0000	1000	xxxx	yyyy	XCH	$R[x]$ SWAP $R[y]$
0000	1001	xxxx	yyyy	MOV	$R[x] \leftarrow R[y]$
0000	1010	xxxx	yyyy	LDR	$R[x] \leftarrow m(R[y], R[y+1])$
0000	1011	xxxx	yyyy	STR	$R[x] \rightarrow m(R[y], R[y+1])$
0000	1100	xxxx	yyyy	LDB	$R[x] = m(R[y])$
0000	1101	xxxx	yyyy	STB	$m(R[x]) = R[y]$

IMMEDIATE

0001	####	xxxx	xxxx	BRI	$PC = PC + 2 * x$
0010	rrrr	xxxx	xxxx	ADD	$R[r] = R[r] + x$
0011	rrrr	xxxx	xxxx	SUB	$R[r] = R[r] - x$
0100	rrrr	xxxx	xxxx	LSL	$R[r] = R[r] \ll x$
0101	rrrr	xxxx	xxxx	RSR	$R[r] = R[r] \gg x$
0110	rrrr	xxxx	xxxx	AND	$R[r] = R[r] \& x$
0111	rrrr	xxxx	xxxx	ORR	$R[r] = R[r] x$
1000	rrrr	xxxx	xxxx	EOR	$R[r] = R[r] \wedge x$
1001	rrrr	xxxx	xxxx	MOV	$R[r] = x$
1010	rrrr	xxxx	xxxx	CMP	Compare $R[r]$ and x

CPU Architecture

NAME is a 16-bit CPU structured for highest simplicity. All instructions are 16-bit constant length, and all registers are 16-bit wide. As the address bus is 16 bits wide, there are 64 Kilobytes of addressable memory, which is split into several distinct portions, as shown in the following table:

START	END	DEVICE
0x0000	0x7FFF	RAM (32k)
0x8000	0xBFFF	ROM (32k)
0xC000	0xD7FE	VIDEO
0xD7FF	0xDEEE	MMU
0xE000	0xFFFF	GPIO

RAM: Upon startup, RAM is cleared.

ROM: Prior to loading the CPU, the image file “rom.bin” will be loaded into ROM memory. Mindful of the 4K ROM limit, should the file exceed this size, it will be truncated. **Writes to ROM are not possible and will result in a bus error (System Interrupt 2).**

VIDEO: The video memory serves a 40x33 text-only screen with EGA color graphics modes. For more information on its use, see the *Video Device* section.

MMU: The MMU acts as an intermediary between user instructions and their impact on the internal states of the CPU. It contains tables allowing for the protection of memory based on the current ring level. For more information on this, please see the *Supervisor Mode* section or the *MMU Device* section.

GPIO: The GPIO device allows for input and output to certain devices.

When the CPU is started, it will begin in HALT mode, and no instructions will be executed. The CPU will also begin in supervisor mode. The program counter will begin at position 0x8000. Because instructions are 2 bytes wide, the program counter is incremented by 2 each cycle.

Note that the lowest 0x1000 bytes of RAM are reserved for system usage as follows:

0x0000:0x00022	Interrupt Vector Table
→ 0x0000:0x0010	System Reserved Interrupts
→ 0x0010:0x0022	User (Software) Interrupts

Supervisor & User Mode

The NAME CPU implements certain protective features which can be enabled/disabled by entering or exiting supervisor mode (ring 0) and User mode (ring 1).

In supervisor mode, all memory locations and registers are writeable, and all opcodes can be used freely.

In user mode, certain registers are read-only and certain opcodes cannot be called. In addition to these protection features, while in user mode, the CPU will first audit the MMU before writing to memory locations. Certain things to keep in mind:

- The MMU can only be initialized/updated while in supervisor mode.
- Supervisor mode can be exited by clearing the supervisor bit in register RF.
- Supervisor mode can only be entered by means of an exception or system reset.
- Supervisor mode has its own stack, the supervisor stack, beginning at 0x1000

For more information, see the Interrupts section.

For example, assuming one programs a kernel, the kernel would:

- Set up the MMU to protect its own memory space
- Set up the IVT so that programs can jump to kernel space
- Enter user mode and jump to the main program

In this setup, the kernel is protected from user mode operations and can only be entered by means of an interrupt, which will automatically elevate the CPU privileges to supervisor and (if the IVT is set up properly) jump to kernel code.

FLAGS Register

The FLAGS register (RF) contains bits that describe several internal states of the CPU. Its format is as follows:

15	X	X	X	X	I	I	I	SV	X	X	X	X	N	Z	O	C	0
----	---	---	---	---	---	---	---	----	---	---	---	---	---	---	---	---	---

- (C) CARRY:** This bit is enabled if an arithmetic operation has performed a carry. Example trigger: $1111 + 0001 = 0000 + CR$
- (O) OVERFLOW:** This bit is enabled if an arithmetic operation has overflowed into a sign bit. Example trigger: $0100 + 0100 = 1000 + OV$
- (Z) ZERO:** The zero bit is set if the destination register becomes zero after an arithmetic operation or comparison.
- (N) NEGATIVE:** The negative bit is set if the destination register becomes negative after an arithmetic operation or comparison.
- (SV) SUPERVISOR:** The supervisor bit is set if the CPU is in supervisor mode.
- (I₁, I₂, I₃):** Interrupt masking bits. See interrupt section for more information.






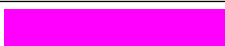

Video Memory

The video memory, addressable from 0xC000 to 0xD7FE, serves the video device. This device is a 93x33 character, ASCII text only graphical terminal with EGA color mode.

The formatting of each character, which occupies 16 bits, is as follows:

16	0	
BACKGROUND_COLOR	FOREGROUND_COLOR	CHARACTER
4	4	8

Unfortunately, not all EGA colors are supported natively by ncurses. Those available are as follows:

CODE	NAME	HEX	SAMPLE
0x0	BLACK	0x000000	
0x1	BLUE	0x0000AA	
0x2	GREEN	0x00AA00	
0x3	CYAN	0x00AAAA	
0x4	RED	0xAA0000	
0x5	MAGENTA	0xAA00AA	
0x6	YELLOW	0xAA5500	
0xF	WHITE	0xFFFFFFFF	

As an example, to produce the following character with red text on a black background: 

The associated 16 bit code would be:

0000 0100 0101 0010 → 0x0452

To place the character at the top left corner, one would simply write 0x0452 to memory location 0xC000, as such:

```

XOR  RX,  RX      ;Clear RX
ADD  RX, 192
LSL  RX,  8       ;RX now contains 0xC000
ADD  RA,  4
LSL  RA,  8
ADD  RA, 82       ;RA now contains 0x0452
STR  RA,  RX      ;Store RA at RX
    
```

Assembler Directives

The assembler contains several directives that support the creation of variables and offsets to the code in the ROM image. So far, the supported directives are:

- .org n Must only be used on beginning of assembly source. Will shift the assembled machine code $n*2$ bytes.
- .data n Begins the data segment containing variables at position $n*2$.
- .str x s Allocates (appends) a null-terminated string to the data segment with name x.

Note that all variable names **MUST** be single words, ASCII, no spaces!

Memory Management Unit

The Memory Management Unit (MMU) operates as a watchdog between user commands and the memory that they access. On the bus, the MMU occupies 4 Kilobytes of memory from 0xD000 to 0xDFFF. In order to write to the MMU the CPU MUST be in supervisor (ring 0) mode.

The MMU at its core is simply an array of objects, which can be considered tables, that define which portions of memory belong to the supervisor. Each entry to the MMU has a starting address, and a block size. As such, the formatting of an entry is as follows:

<code>u_int16_t</code> START_ADDRESS	<code>u_int16_t</code> BLOCK_SIZE
--------------------------------------	-----------------------------------

The starting address references where the protected memory segment begins, and the block size determines how many words the segment occupies. For example, if one wishes to protect a segment of memory from 0x2000 to 0x2100, this means the starting address is 0x2000 and the block size is 0x100. The entry is then: 0x20000100

If the CPU is in user mode, and an attempt is made to write to memory that the MMU has defined as protected, system interrupt 0 (PRIVILEGE_VIOLATION) will be triggered.

Interrupt Handling

When an interrupt is triggered, such as a RESET or a software interrupt such as INT n, the CPU will enter supervisor mode. There are several steps taken by the processor for such a context switch, they are:

1. Processor enters Supervisor (Ring 0) mode
2. Base pointer switches to the supervisor stack at base address 0x1000
3. User frame pointer is pushed onto the supervisor stack, new FP is 0x0FFE
4. Flags register is pushed onto the supervisor stack
5. Program counter is pushed onto the supervisor stack
6. Program counter is loaded with the corresponding address from the *Interrupt Vector Table (IVT)* at location 0x0000, starting the *Interrupt Service Routine*

At the end of the interrupt, the command RTI (Return from Interrupt) is issued, and the processor performs the following actions:

- 1) Program counter is popped from the supervisor stack
- 2) Flags register is popped from the supervisor stack
- 3) User stack is returned to its previous state
- 4) Processor enters User (Ring 1) mode

Traditionally, the IVT resides in protected memory space and therefore cannot be modified in ring 1 mode. The proper procedure for initializing the IVT is to load the addresses of the ISRs into the IVT address space and then switch to ring 1, as seen in the following example where we set the ISR for System Interrupt 1 to a function at address 0x0400:

```
ADD RX, 2      ;0x0002 is the IVT entry for S. INT 1
ADD RA, 4
LSL RA, 8      ;Load 0x0400 into RA
STR RA, RX
EOR RF, RF     ;Exit supervisor mode
```

There are system reserved interrupts from 0 to 7. These interrupts occupy address space 0x0000 through 0x0010, after which the software interrupt vector table begins. For example, a call to opcode INT 1 will jump to the location provided at IVT address 0x0026. The following is a list of interrupts:

IVT Address	Interrupt	
0x0000	SYSTEM INT 0	PRIVILEGE VIOLATION
0x0002	SYSTEM INT 1	KEYBOARD PRESS
0x0004	SYSTEM INT 2	BUS ERROR
0x0008	RESERVED	
0x000A	RESERVED	
0x000C	RESERVED	
0x000E	RESERVED	
0x0010	RESERVED	
0x0012 - 0x0022	INT0 - INT 7	Software Interrupts

Registers

The NAME CPU has 11 internal registers, each 16 bits wide. They are:

RA, RB, RC, RD	General purpose registers.
PC	Program counter
FP	User stack frame pointer
BP	Base pointer (can be either user or supervisor base pointer.)
RX	Index register
RF	FLAGS register
RL	Link register, used for RET instruction, set by CAL instruction
SP	Supervisor stack frame pointer

When in user mode, registers PC, RF, and SP cannot be written to by any instruction. They are only writeable in supervisor mode.

General Purpose I/O Ports

The NAME CPU employs memory-mapped I/O residing at address space 0xE000 to 0xFFFF. Currently, there is only one device mapped to this address space: the keyboard.

The keyboard occupies the first word of GPIO memory from 0xE000 to 0xE001, and this word is set to the last key that was pressed. The format is as follows:

Unused ASCII Last Key Pressed	
8	8

To toggle keyboard mode in the UI, press ESC.