

Operating Systems

CMSC 422 and MSCS 515

Machine Language Instruction Set – Revised



The instruction set for our OS CPU simulation's machine language is based on the op codes of the classic 6502 microprocessor. It was the heart of the Commodore PET, Apple II, and Bender, so we're in good company using it ourselves.

There is an excellent virtual 6502 simulator, assembler, and disassembler at <http://e-tradition.net/bytes/6502>. Make frequent use of this tool so that you can test your machine code there before trying it in your OS. You should spend time debugging your OS, not the user programs. This also saves you the step of writing your own assembler, although we might want to do that later on.

There are three registers: X, Y, and the Accumulator.

Description	Op Code	Mnemonic	Example Assembly	Example Disassembly
Load the accumulator with a constant	A9	LDA	LDA #\$07	A9 07
Load the accumulator from memory	AD	LDA	LDA \$0010	AD 10 00
Store the accumulator in memory	8D	STA	STA \$0010	8D 10 00
Add with carry Adds contents of an address to the contents of the accumulator and keeps the result in the accumulator	6D	ADC	ADC \$0010	6D 10 00
Load the X register with a constant	A2	LDX	LDX #\$01	A2 01
Load the X register from memory	AE	LDX	LDX \$0010	AE 10 00
Load the Y register with a constant	A0	LDY	LDY #\$04	A0 04
Load the Y register from memory	AC	LDY	LDY \$0010	AC 10 00
No Operation	EA	NOP	EA	EA
Break (which is really a system call)	00	BRK	00	00
Compare a byte in memory to the X reg Sets the Z (zero) flag if equal	EC	CPX	EC \$0010	EC 10 00
Branch X bytes if Z flag = 0	D0	BNE	D0 EF	F0 EF
Increment the value of a byte	EE	INC	EE \$0021	EE 21 00
System Call #\$01 in X reg = print the integer stored in the Y register. #\$02 in X reg = print the 00-terminated string stored at the address in the Y register.	FF	SYS		FF

Operating Systems

CMSC 422 and MSCS 515

Machine Language Instruction Set – Revised

Example Three

; OS/1.3 (OS class program three)
; Prints 1, 2 and DONE.

lda #\$3	Acc = 3	0000	LDA #\$03	A9 03
sta \$0041	Mem[41] = 3	0002	STA \$0041	8D 41 00
lda #\$1	Acc = 1	0005	LDA #\$01	A9 01
sta \$0040	Mem[40] = 1	0007	STA \$0040	8D 40 00
loop ldy \$0040	Y = Mem[40]	000A	LOOP LDY \$0040	AC 40 00
ldx #\$01	X = 1	000D	LDX #\$01	A2 01
sys	System Call	000F	SYS	FF
inc \$0040	Mem[40]++	0010	INC \$0040	EE 40 00
ldx \$0040	X = Mem[40]	0013	LDX \$0040	AE 40 00
cpx \$0041	Z bit = (x == Mem[41])	0016	CPX \$0041	EC 41 00
bne loop	if z == 0 goto loop	0019	BNE LOOP	D0 EF
lda #\$44	Acc = \$44 ("D")	001B	LDA #\$44	A9 44
sta \$0042	Mem[42] = \$44	001D	STA \$0042	8D 42 00
lda #\$4F	Acc = \$4F ("O")	0020	LDA #\$4F	A9 4F
sta \$0043	Mem[43] = \$4F	0022	STA \$0043	8D 43 00
lda #\$4E	Acc = \$4E ("N")	0025	LDA #\$4E	A9 4E
sta \$0044	Mem[44] = \$4E	0027	STA \$0044	8D 44 00
lda #\$45	Acc = \$45 ("E")	002A	LDA #\$45	A9 45
sta \$0045	Mem[45] = \$45	002C	STA \$0045	8D 45 00
lda #\$00	Acc = \$00 (null)	002F	LDA #\$00	A9 00
sta \$0046	Mem[46] = \$00	0031	STA \$0046	8D 46 00
ldx #\$02	X = 2	0034	LDX #\$02	A2 02
ldy #\$42	Y = \$42 (address)	0036	LDY #\$42	A0 42
sys	System call	0038	SYS	FF
brk	Break	0039	BRK	00

Remember, SYS does not cause an error (as the real 6502 did not have this), which is nice, but it also does not generate an op code. In order to make our code work in the emulator, we use the op code for NOP in place of SYS. Thus the EA's in the op code stream below.

```
A9 03 8D 41 00 A9 01 8D 40 00 AC 40 00 A2 01 EA EE 40 00 AE 40 00 EC 41 00 D0
EF A9 44 8D 42 00 A9 4F 8D 43 00 A9 4E 8D 44 00 A9 45 8D 45 00 A9 00 8D 46 00
A2 02 A0 42 EA 00
```

In your CPU, you will generate a software interrupt when you see the SYS op code (FF) so that your OS can handle it.

Copy the object code and test it out at <http://www.e-tradition.net/bytes/6502>.