MITS Altair Programming Tutorial

By Mihai Pruna

mihaipruna.com

I always wanted to learn to program a computer in Machine Code (just 1s and 0s, the true language of computers). Recently I started reading a biography of Bill Gates (ISBN-10: 0385420757), which starts in the early days of widely available computing. The era of the personal computer and the success of Microsoft were both ushered in by a device that, if purchased without accessories, did not have a keyboard or monitor, yet could be fully programmed using switches and blinking lights: The MITS Altair.

This is a tutorial for a simple program for the MITS Altair computer.

You can use a web-based emulator of the MITS Altair, as well as read the full operating manual, here:

http://www.s2js.com/altair/

The Intel 8080 processor of the Altair is a great device to learn to program in Assembly Language.

Here are more resources for programming the 8080 in Assembly:

http://www.hartetechnologies.com/manuals/Unclassified/8080_Machine_Language_Programming_for_Beginner.PDF

http://altairclone.com/downloads/manuals/8080%20Programmers%20Manual.pdf

This tutorial shows a simple program comparing two numbers. You should do this tutorial after you are able to write and run and understand the first program in the Altair Operating Manual (http://www.classiccmp.org/dunfield/altair/d/88opman.pdf) which adds two numbers, as this tutorial is based on the addition example. This example is more complex than the addition example in the manual, but, also, much simpler than the multiplication example.

Below is a memory map and program and data storage. I found Excel to be a great way to organize and design your code. Just like in the addition tutorial, I used memory locations 128 and 129 to store the two numbers, and memory address 130 to store the result, in this case the largest number. These numbers are in decimal notation. Binary equivalents also given for every address and instruction for inputting via the Altair front panel.

The first column is the contents of the memory of the Altair in decimal. The second in binary. The third is the instruction mnemonic. The instruction can occupy more than one byte if values or addresses are specified as arguments.

Memory Address Dec	Memory Address Bin	Instruction	Binary Value	Comment
0	00000000	LDA	00 111 010	Load accumulator w content at address 128
1	00000001		10 000 000	
2	00000010		00 000 000	
3	00000011	MOV A->B	01 000 111	Move from Accumulator to Register B
4	00000100	LDA	00 111 010	Load accumulator w content at address 129
5	00000101		10 000 001	
6	00000110		00 000 000	
7	00000111	CMP A,B	10 111 000	Compare accumulator with register B . If B>A then carry bit is 1 otherwise carry bit is 0. This means if # at 129> # at 128 carry bit is 0
8	00001000	JC	11 011 010	Jump to address and execute instruction there
9	00001001		00 010 001	
10	00001010		00 000 000	
11	00001011	STA	00 110 010	This is no carry so # at 129 > # at 128 thus we write #129 in address 130. Accumulator already has 129 so we write A to 130
12	00001100		10 000 010	
13	00001101		00 000 000	
14	00001110	JMP	11 000 011	Go back to beginning of program
15	00001111		00 000 000	
16	00010000		00 000 000	
17	00010001	LDA	00 111 010	This is for carry where we jump so we can set 130 to the value of 128
18	00010010		10 000 000	
19	00010011		00 000 000	
20	00010100	STA	00 110 010	Store accumulator which has value from 128 at 130
21	00010101		10 000 010	
22	00010110		00 000 000	

23	00010111	JMP	11 000 011	Go back to beginning of program
24	00011000		00 000 000	
25	00011001		00 000 000	
26	00011010			
27	00011011			
28	00011100			
29	00011101			
…	….	….	….	……
125	01111101			
126	01111110			
127	01111111			
128	10000000		first number	
129	10000001		second number	
130	10000010		largest number	
	….			
255	11111111			

Below are some relevant excerpts from the operating manual:

How to start writing the program (from example)

STEP	SWITCHES 0-7	CONTROL SWITCH
		RESET
0	00 111 010	
		DEPOSIT
1	10 000 000	
		DEPOSIT NEXT

Load accumulator with memory content for number 1:

0. LDA	00 111 010	Load Accumulator with contents
	10 000 000	of: Memory address 128 (2 bytes
	00 000 000	required for memory addresses)

The load accumulator command reference:

LDA (LOAD ACCUMULATOR DIRECT)	00 111 010	(Byte 1)
	(Low Address)	(Byte 2)
	(High Address)	(Byte 3)

Operation: The accumulator is loaded with the contents of the byte at the memory address given by bytes 2 and 3 of the instruction.

Move from accumulator to B

1. MOV (A→B)	01 000 111	Move Accumulator to Register B

Move data command reference:

A	111

MOV (MOVE DATA) 01 DDD SSS (Byte 1)

Operation: The contents of SSS (the source register) are moved to DDD (the destination register). The contents of SSS remain unchanged. The following bit patterns for the source and destination registers apply:

Load accumulator with memory content for number 2

2. LDA	00 111 010	Load Accumulator with contents
	10 000 001	of: Memory address 129
	00 000 000	

Compare B to Accumulator. If B>Accumulator, Carry bit is 1.

CMP (COMPARE REGISTER/MEMORY WITH ACCUMULATOR) 10 111 (reg)

Operation: The content of the specified register is compared with the content of the accumulator by subtracting the former from the latter. The contents of the register and accumulator are unaffected by this operation, and the status bits are set or reset as appropriate.

Code for register B:

Register	Bit Pattern
B	000

The Jump if Carry command goes to specified address if carry bit is 1.

JC	(JUMP IF CARRY)	11 011 010	(Byte 1)
		(Low Address)	(Byte 2)
		(High Address)	(Byte 3)

Write accumulator to result memory address

4. STA	00 110 010	Store Accumulator contents
	10 000 010	at: Memory address 130
	00 000 000	

Store accumulator at memory address command reference:

STA	(STORE ACCUMULATOR DIRECT)	00 110 010	(Byte 1)
		(Low Address)	(Byte 2)
		(High Address)	(Byte 3)

Operation: The contents of the accumulator are stored in the memory at the address specified in bytes 2 and 3.

Jump to beginning of program to keep executing it:

5. JMP	11 000 011	Jump to Memory location 0.
	00 000 000	
	00 000 000	

Jump command reference:

JMP (JUMP)	11 000 011	(Byte 1)
	(Low Address)	(Byte 2)
	(High Address)	(Byte 3)

How to use the program:

The program is now ready to be run, but first it is necessary to store data at each of the two memory addresses which are to be added together. To load the first address, set the DATA/ADDRESS switches to 10 000 000 and actuate EXAMINE. You can now load any desired number into this address by loading the DATA/ADDRESS switches as appropriate. When the number has been loaded into the switches, actuate DEPOSIT to load it into the memory. To load the next address, enter the second number on the DATA/ADDRESS switches and actuate DEPOSIT NEXT. Since sequential memory addresses were selected, the number will be automatically loaded into the proper address (10 000 001). If non-sequential memory addresses had been selected, the procedure for finding the first address would have to be followed (load the address into the DATA/ADDRESS switches and actuate EXAMINE; then load the number into the DATA/ADDRESS switches and actuate DEPOSIT).