

Lab03 2017/11/13

PB16030899 Zhu Heqin

- [Lab03 2017/11/13](#)
 - [PB16030899 Zhu Heqin](#)
 - [PURPOSE](#)
 - [PRINCEPLE](#)
 - [PROCEDURE](#)
 - [RESULT](#)
 - [ATTACHMENT](#)

PURPOSE

This lab let us use **ASSEMBLE** language to write a program that can sort the data stored in two locations. To some degree ,it's a bit hard . Through this lab, so fluently of coding and deep understanding of the assemble language that I've gained.

Also, I feel a sense of achievement aftering the correct result appeared . THough it took me about two hour to accomplish this lab.

PRINCEPLE

linstructions I used are as follows

- LD
- LD
- LDR
- STR
- ADD
- NOT
- AND
- BR
- JSR
- RET

and **persudo-op**

- .FILL
- .ORIG
- .END

There are some beautiful subroutines I wrote to cooperate the program to solve the problem.

- **CMP**:CMP MEM[R0],MEM[R1], SAVE RST IN R6
- **EQBGN**:CMP IF THE CONTENT OF R0 IF THE SRCEND,AND STORE THE RST IN R6

- **WRITE**:WRITE MEM[R1] TO MEM[R4] ,TWO LOCATION,AND MOVE THE HIGHEST SRC DATA (R5) TO THE POSITION OF MAX (R1)

PROCEDURE

Firstly ,I write a small programm to set the test case so that I won't waste too much time to set data manually.

Then I wrote the programm with some subroutines concentrately .

I learned the asemble language for some time. Then I quickly write the program . Three has been some problems when debugging.

I got some experience:

- Behind the persudo-op **.ORIG** ,threr must be a memory address
- Whatever the instruction is. it will be invalid when it 's after the persudo-op **.END**
- make full use of **.STRINGZ**

RESULT

Before executing the program:

The image shows two side-by-side screenshots of the LC3 Simulator. The left window, titled 'LC3 Simulator - SETVAL.obj', shows the initial state of the program. The right window, titled 'LC3 Simulator - lab03.obj', shows the initial state of the program. Both windows display the initial register values and the first few instructions of the program.

Register	Value	Register	Value	Register	Value	Register	Value
R0	x7FFF	32767	R4	x0000	0	PC	xFD79
R1	xFFFF	-1	R5	x0000	0	IR	xB02C
R2	x0000	0	R6	x0000	0	PSR	xB001
R3	x3200	12800	R7	xFD75	-651	CC	P

The left window shows the first few instructions of the program:

```

x3200 0000000000000000 x0000 NOP
x3201 0000000010000000 x0080 NOP
x3202 0000000001111111 x007F NOP
x3203 0000000001111111 x007E NOP
x3204 0000000001111101 x007D NOP
x3205 0000000001111100 x007C NOP
x3206 0000000001111011 x007B NOP
x3207 0000000001111010 x007A NOP
x3208 0000000001111001 x0079 NOP
x3209 0000000001111000 x0078 NOP
x320A 0000000001110111 x0077 NOP
x320B 0000000001110110 x0076 NOP
x320C 0000000001110101 x0075 NOP
x320D 0000000001110100 x0074 NOP
x320E 0000000001110011 x0073 NOP
x320F 0000000001110010 x0072 NOP
x3210 0000000001110001 x0071 NOP
x3211 0000000001110000 x0070 NOP
x3212 0000000001101111 x006F NOP
x3213 0000000001101110 x006E NOP
x3214 0000000001101101 x006D NOP
x3215 0000000001101100 x006C NOP
x3216 0000000001101011 x006B NOP
x3217 0000000001101010 x006A NOP
x3218 0000000001101001 x0069 NOP
x3219 0000000001101000 x0068 NOP
x321A 0000000001100111 x0067 NOP
x321B 0000000001100110 x0066 NOP
x321C 0000000001100101 x0065 NOP
x321D 0000000001100100 x0064 NOP
x321E 0000000001100011 x0063 NOP
x321F 0000000001100010 x0062 NOP

```

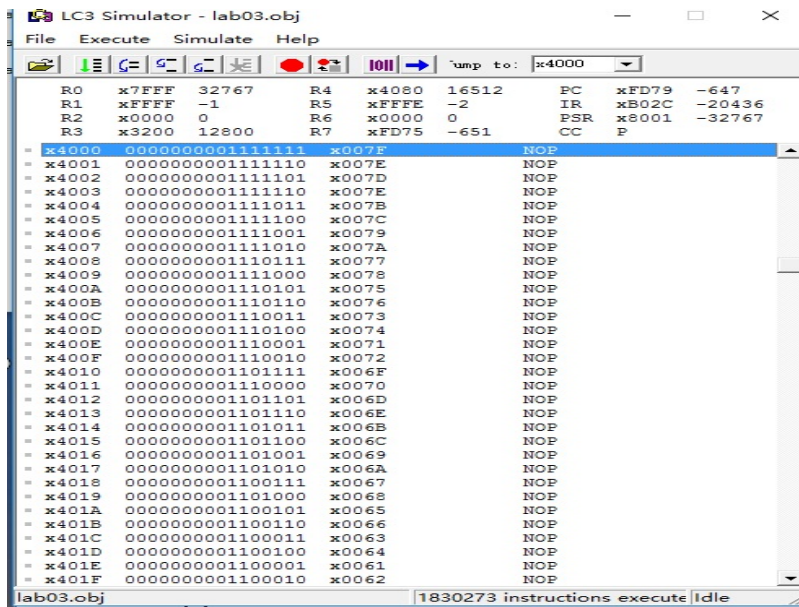
The right window shows the first few instructions of the program:

```

x3262 0000000001100011 x0063 NOP
x3263 0000000001100100 x0064 NOP
x3264 0000000001100101 x0065 NOP
x3265 0000000001100110 x0066 NOP
x3266 0000000001100111 x0067 NOP
x3267 0000000001101000 x0068 NOP
x3268 0000000001101001 x0069 NOP
x3269 0000000001101010 x006A NOP
x326A 0000000001101011 x006B NOP
x326B 0000000001101100 x006C NOP
x326C 0000000001101101 x006D NOP
x326D 0000000001101110 x006E NOP
x326E 0000000001101111 x006F NOP
x326F 0000000001110000 x0070 NOP
x3270 0000000001110001 x0071 NOP
x3271 0000000001110010 x0072 NOP
x3272 0000000001110011 x0073 NOP
x3273 0000000001110100 x0074 NOP
x3274 0000000001110101 x0075 NOP
x3275 0000000001110110 x0076 NOP
x3276 0000000001110111 x0077 NOP
x3277 0000000001111000 x0078 NOP
x3278 0000000001111001 x0079 NOP
x3279 0000000001111010 x007A NOP
x327A 0000000001111011 x007B NOP
x327B 0000000001111100 x007C NOP
x327C 0000000001111101 x007D NOP
x327D 0000000001111110 x007E NOP
x327E 0000000001111111 x007F NOP
x327F 0000000001111110 x007E NOP
x3280 0000000000000000 x0000 NOP
x3281 0000000000000000 x0000 NOP

```

After executing the program:



It works well. Through this lab , I learned a lot. Assembly language is so brief and useful.

ATTACHMENT

This is the small program to set the test data.

```
.ORIG X3000
LD R5,NUM
LD R3,SRCBGN
LD R1,NUM

TESTVAL ADD R0,R5,R3
        ADD R1,R1,#-1
        STR R1,R0,#0
        ADD R5,R5,#-1
        BRp TESTVAL

NUM      .FILL #127
SRCBGN   .FILL x3200

        HALT
        .END
```