# Lab 6 Report

In Lab 6b, we need to write a program to execute LC-3binary code. As the TRAP and RTI instruction aren’t need to be implemented, this Lab is actually easy. Just because there are 13 instructions, the program may be a big one.

**Arithmetic**

We have learned the composition of LC-3, also the state machine for the LC-3, so in fact, the most thing have been done. That is, we can declare variables and implement instructions according to the state machine because there are a lot of useful information. The only problem is that we need to convert the input characters into numbers that are easy to calculate, also we have to notice the conversion of binary numbers in different bits. All in all, data type is what we need to pay attention to.

**Code**

1. typedef struct memnode\* memnodep;
2. struct memnode{
3. int loc;   *//address*
4. char data[17];
5. memnodep next;
6. memnodep front;
7. };

As I don’t want to have many empty memory, a linked list is used to simulate LC-3 memory. That means the memory is certainly be used but it also takes up other memory to show other information. And I need to write functions to create the list and add node to it.

The “loc” is the address of a 16-bits data. [17] makes it convenient to use function “strcpy”. The two memnodep is used to access the front and the back memory nodes.

1. memnodep MEM;   *//Start location of memory*
2. short int reg[8]={30583,30583,30583,30583,30583,30583,30583,30583};         *//register*
3. int PC;
4. memnodep  memp;      *//A pointer used to access memory*
5. int  MAR;
6. char MDR[17];
7. int nzp[3]={0,1,0};     //condition code

As register is used to calculate, it may be a negative number so its type is short int. MAR is the address of the memory we want to access, it won’t be negative, so it is int or unsigned short int.

1. int numconvert(char snum[], int n);
2. void dectobits(int dec,char bits[17]);

The first one will convert the string we input to integer. The “n” is the length of the string. The string can represent a complement binary number or unsigned binary number. The function makes it easy to calculate.

The next one will convert an integer to a complement binary number and then to a string. This function is useful especially when we need to store the data in the register into memory as the types of them are different.

1. void executor();     *//Executing instructions one by one*
2. memnodep findloc(int n);  *//find the pointer to the address which store the data we neeed*
3. int readfun(char ch[17]);  *//According to bit[15:12] to distinguish instructions*
4. void BR(char ch[17]);   *//0000  0*
5. void ADD(char ch[17]);   *//0001  1*
6. void AND(char ch[17]);   *//0101  5*
7. void NOT(char ch[17]);   *//1001  9*
8. void JMP(char ch[17]);   *//1100  12*
9. void JSR(char ch[17]);   *//0100  4*
10. void LD(char ch[17]);   *//0010  2*
11. void LDI(char ch[17]);   *//1010  10*
12. void LDR(char ch[17]);   *//0110  6*
13. void LEA(char ch[17]);   *//1110  14*
14. void ST(char ch[17]);   *//0011  3*
15. void STI(char ch[17]);   *//1011  11*
16. void STR(char ch[17]);   *//0111  7*

The executor is a while loop. When we detect a halt instruction, we will break it.

1. void ADD(char ch[17])   *//0001  1*
2. {
3. char numch[5];
4. int DR,SR1;
5. int i;
6. for(i=0;i<3;i++){
7. numch[i]=ch[i+4];
8. }
9. DR=numconvert(numch,3);
10. for(i=0;i<3;i++){
11. numch[i]=ch[i+7];
12. }
13. SR1=numconvert(numch,3);
15. if(ch[10]=='0'){
16. int SR2;
17. for(i=0;i<3;i++){
18. numch[i]=ch[i+13];
19. }
20. SR2=numconvert(numch,3);
21. reg[DR]=reg[SR1]+reg[SR2];
22. }
23. else if(ch[10]=='1'){
24. int imm;
25. for(i=0;i<5;i++){
26. numch[i]=ch[i+11];
27. }
28. imm=numconvert(numch,5);
29. reg[DR]=reg[SR1]+imm;
30. }
31. short int cmp=reg[DR];
32. for(i=0;i<3;i++){
33. nzp[i]=0;
34. }
35. if(cmp>0) nzp[2]=1;
36. else if(cmp<0) nzp[0]=1;
37. else if(cmp==0) nzp[1]=1;
38. return ;
40. }

The others instruction is just like the ADD instruction. First, convert string to integer we will use and check some condition bits. Then make calculation. Finally, change the condition code if we need.