Lab1 Report

作者: 罗文杰
专业: 计算机科学与技术
学号: 3210102456

1 Introduction

In this lab, we will write a program with LC-3 machine code to identify whether a word is F-word. As far as we can see, one word is F-word if it contains 4 continuous 1.

2 Solution1

To identify a word whether a F-word, we can use iterators. For a loop, we could distinguish a word whether a minus word to identify its bit whether 1, if so, then take iterators add 1, if not, then take iterators to 0. When iterators added to 4, it means the word has 4 continuous 1.

The codes is as follow.

```
0011000000000000; .orig x3000
1010 001 000010100; LDI,R3,PC+21
1010 001 000010100; LDI,R1,PC+20
0101 010 010 1 00000; AND R2 R2 #0
0001 010 010 1 00100; ADD R2 R2 #4
0001 001 001 1 00000; ADD R1 R1 #0
0000 010 000001010; BRz PC+10
0000 100 000000100; BRz PC+4
0101 010 010 1 00000; AND R2 R2 #0
0001 010 010 1 00100; ADD R2 R2 #4
0001 001 001 0 00 001; ADD R1 R1 R1
0000 111 111111010;BRz PC-6
0001 010 010 1 11111; ADD R2 R2 #-1
0001 001 001 0 00 001; ADD R1 R1 R1
0001 010 010 1 00000; ADD R2 R2 #0
0000 010 000000001;BRz PC+1
0000 111 111110100;BRz PC-11
0001 010 010 1 00000; ADD R2 R2 #0
0000 010 000000010;BRz PC+2
0101 010 010 1 00000; AND R2 R2 #0
0000 111 000000001;BRz PC+1
0001 010 010 1 00001; ADD R2 R2 #1
```

```
1111000000100101; halt
0011000100000000; 0X3100 #test
0011000100000000; 0X3101 #test
```

3 Solution2

In Solution1, we have some problems: the program divides the characters into positive and negative cases, discussing too many cases and creating code redundancy; the program cannot handle 4 numbers at the same time, resulting in high space overhead.

To cover this problems, we have another solution: take AND operation on R1 and 1111000000000000, add the result by 1111000000000000, then take NOT operation on the result, if the answer is 0, is means R1 has 4 continuous 1 in its 4 first bits.

The codes is as follow.

```
0011000000000000; .orig x3000
0010 100 000001101;L LD,R4,PC+13
0010 011 000001101;D LD,R3,PC+13
1010 001 000001010; LDI,R1,PC+10
0101 010 001 0 00 011; AND R2 R1 R3
0001 010 010 0 00 100; ADD R2 R2 R4
1001 010 010 111111; NOT R2 R2
0000 010 000000100; BRz PC+4
0001 001 001 0 00 001; ADD R1 R1 R1
0000 101 111111010;BRz PC-5
0101 010 010 1 00000; AND R2 R2 #0
0000 111 000000001;BRz PC+1
0001 010 010 1 00001; ADD R2 R2 #1
1111000000100101; halt
0011000100000000; 0X3100 #test
0000111111111111; L
1111000000000000; D
```