ICS_Lab1_Report

Xiaoma

2022.10.28

实验目的

使用 LC-3 机器语言计算数字 A 的低 B 位的 1 的数量，并将得到的结果存储在内存中。
具体要求：

- 数字 A 存储在内存位置 x3100
- 数字 B 存储在内存位置 x3101
- 计算结果存储在内存位置 x3102

例子：

- 内存位置 x3100 包含值 **0000 0000 0000 1101**
- 内存位置 x3101 包含值 **0000 0000 0000 0011**
- 结果存储在 x3102, 值为 **0000 0000 0000 0010**

实验原理

如果想求一个数字 A 的低 B 位的 1 的个数，我们通常将该数字按位与 1 进行与操作，来判断每一位位是否为 1。按照该思想可以得到伪代码：

Algorithm 1: HammingWeight of Lower B Bits

```
Initialize A, B;
weight = 0;
judge = 1;
while i = 0 to B - 1 do
    temp = A & judge;
    judge *= 2;
    if temp != 0 then
        ++weight;
return weight;
```

实验步骤

伪代码转换

将算法思想用 LC-3 机器语言实现，需要解决的问题：

- 循环的实现
- judge 值的更新

循环的实现

使用条件跳转指令 BRz，当判断 A 中某位是否为 1 时，每判断一次，B 减 1，当 B 不为 0 时，跳转至前面进行循环，当 B 为 0 时，结束循环，存储结果。

judge 值的更新

由于 LC-3 没有移位指令，若想将 judge 右移，则需要使用 ADD 指令，将 judge 的值变为原来的二倍以后写回。

实验代码

解决两个问题以后，我们可以得到最终的 LC-3 程序

```
0011000000000000
0010000011111111 ;LD R0, x00F
0101100100100000 ;AND R4, R4, x00
0010001011111110 ;LD R1, x00E
0000010000001010 ;BRz R1, x00A
0101010010100000 ;AND R2, R2, x00
0101011011100000 ;AND R3, R3, x00
0001010010100001 ;ADD R2, R2, x01
0000001000000001 ;BRp x001
0001010010000010 ;ADD R2, R2, R2
0101011000000010 ;AND R3, R0, R2
0000010000000001 ;BRz x001
0001100100100001 ;ADD R4, R4, x01
0001001001111111 ;ADD R1, R1, x1F
0000001111111010 ;BRp x1FA
0011100011110011 ;ST R4, x0F3
1111000000100101
```

实现代码的一些细节

- 使用 LD 指令从内存位置 x3100,x3101 中读取所需数据至 R0,R1(**R0 代表 A,R1 代表 B**)。

```
0010000011111111 ;LD R0, x00F
0010001011111111 ;LD R1, x00F
```

- 首先判断 $R1$ 是否为 0，若为 0 则直接返回 0，结束程序。

```
0010001011111110 ;LD R1, x00E
0000010000001010 ;BRz R1, x00A
```

- 使用 ADD 清空 R2，R3，R4 的值，其中 (**R2 代表 judge,R3 代表判断结果，R4 代表 weight**)，使其变为 0，排除寄存器初始值对结果的影响

```
0101010010100000 ;AND R2, R2, x00
0101100100100000 ;AND R4, R4, x00
0101011011100000 ;AND R3, R3, x00
```

- 使用 BRp 指令进行判断，若 R2 首次进入循环则不需要进行移位

```
0000001000000001 ;BRp x001
```

- 使用 ADD 指令对 R2 中数值进行移位操作

```
0001010010000010 ;ADD R2, R2, R2
```

- 使用 ADD 指令判断 R0 某位是否为 1

```
0101011000000010 ;AND R3, R0, R2
```

- 使用 BRz 指令，若结果非 0，则 R4 加 1，若结果为 0，R4 值不增加

```
0000010000000001 ;BRz x001
0001100100100001 ;ADD R4, R4, x01
```

- 使用 ADD 指令，将 R1 的值减一

```
0001001001111111 ;ADD R1, R1, x1F
```

- 使用 BRp 指令，若 R1 不为 0 则继续以上循环，若 R1 为 0 则结束

```
0000001111111010  ;BRp x1FA
```

- 将得到的结果存储至内存位置 x3102

```
0011100011110100  ;ST R4, x0F4
```

实验结果

依次对实验文档给出的例子进行测试，结果如下：

机器码评测

3 / 3 个通过测试用例

- 平均指令数: 54
- 通过 13:3, 指令数: 24, 输出: 2
- 通过 167:6, 指令数: 41, 输出: 4
- 通过 32767:15, 指令数: 97, 输出: 15

自行编写了部分测试例子，结果如下：

机器码评测

3 / 3 个通过测试用例

- 平均指令数: 21.333333333333332
- 通过 25:1, 指令数: 13, 输出: 1
- 通过 245:4, 指令数: 29, 输出: 2
- 通过 456:3, 指令数: 22, 输出: 0