Lab3 实验报告

倪煜晖 523030910170

2025 年 5 月 29 日

1 控制逻辑

在 Lab3 LC-3 Simulator 的整个实现过程中，总体的控制逻辑没有使用新的全局变量，所有功能集成在 process_instruction() 中实现。整体的控制逻辑如下：

1. Fetch: 读取 MEMORY[CURRENT_LATCHES.PC], 使用整型的 instr 存储读取到的指令，取出前四位存储到整型 opcode 中，具体实现方法在第二部分中说明。
2. Decode：根据 opcode 的值决定执行的程序类型。
3. Execute：根据 ISA 的要求执行对应指令操作寄存器、内存、条件码、PC 等

在实际实现中，使用的是 switch-case 语句，因此，实际上 Decode 和 Execute 可以说是一起实现的。

2 功能实现

注意到，LC3 中，寄存器的容量与单条指令的长度均为 16 位，在用 C 实现的仿真器中，我们使用一个 int 变量就可以完成存储。这个性质是非常好的。所以，在之后的实现中我们可以很方便的使用 **位运算**来大大简化运算。下面，列举几个关键功能的实现思路。

2.1 Fetch and Decode

位操作与掩码的使用是整个模拟器实现的核心思路。在 Fetch and Decode 中，使用

$$\text{int opcode} = (\text{instr} >> 12)\ \&\ 0\text{xF}$$

取出前四位。之后，使用 switch-case 就可以简洁的处理分支跳转。

2.2 取寄存器

延续**位操作与使用掩码**的思路，只要把指令右移起始位置的位数加上掩码就可以读取到寄存器的值。以加法实现中 DR 的读取为例：

$$\text{int dr} = (\text{instr} >> 9)\ \&\ 0\text{x7}$$

使用上面的指令，读取 [11:9] 处的 DR 值。

2.3 取单个位

取单个位在判断 ADD，JSR，AND 等指令的模式的时候十分重要，判断条件码的值也需要这个操作。它的实现思路仍然是**位操作与使用掩码**。同样以 ADD 指令中的应用举例：

$$\text{int imm\_flag} = (\text{instr} >> 5)\ \&\ 0x1$$

使用上面的指令获得了第 5 位的模式信息。以上，基本说明了取一位与多位的思路，后面的 offset 同样用这种方式得到，不再赘述。

2.4 符号扩展

为了实现正确的带符号运算，在运算前有时候要先实现符号扩展。使用三元运算符?:，配合掩码得到符号扩展的结果。以 BR 指令中的符号扩展为例：

$$\text{pc\_offset} = (\text{pc\_offset}\ \&\ 0x100)\ ?\ (\text{pc\_offset}\ |\ 0xFE00) : \text{pc\_offset}$$

BR 指令采取的是 offset9，用掩码 0x100 得到 sign bit，再根据 offset 的正负进行对应扩展。

2.5 操作数设置

使用取单个位的方式取符号位。直接判断 N 与 Z，用逻辑运算得到 P

$$\text{NEXT\_LATCHES.N} = (\text{result} \gg 15)\ \&\ 0x1;$$
$$\text{NEXT\_LATCHES.Z} = (\text{result} == 0);$$
$$\text{NEXT\_LATCHES.P} = !(\text{NEXT\_LATCHES.N}\ ||\ \text{NEXT\_LATCHES.Z});$$

2.6 PC 自增与跳转逻辑

在 LC3 中，Fetch 后 PC 直接自增，而我们的模拟程序使用 CURRENT_LATCHES 与 NEXT_LATCHES 记录状态。为了解决 BR 等指令中的 offset 实际上是在与自增后的 PC 做加法的问题，在适当的时候直接加一，最后再统一设置 NEXT_LATCHES.PC 的值。

3 验证与测试

编译程序，编写测试 shell, 按照要求编译运行：

```
(base) PS E:\LC3>  gcc -std=c99 -o lc3sim lc3sim.c
(base) PS E:\LC3> .\lc3sim addtest
LC-3 Simulator

Read 10 words from program into memory.

LC-3-SIM>
```

图 1: LC3 Simulator 启动

3.1 ADD 功能测试

执行以下命令：

.ORIG 0x3000 (0x3000)

0x3000 ADD R0,R0,#1 (0x1021)

0x3001 ADD R1,R1,#3 (0x1263)

0x3002 ADD R1,R1,#-1(0x127F)

0x3003 ADD R2,R0,R1(0x1401)

0x3004 ADD R4,R4,#-1(0x193F)

运行结果如图2.

可以看到，第一、二条指令验证了立即数模式的正确性，第三条验证了加负数功能的正确性，第四条验证了寄存器模式的正确性，最后一条说明 Low16bits（x）((x) & 0xFFFF）的处理方式运行正常。此外，各指令均正确处理了条件码。

3.2 NOT 功能测试

在之前程序的基础上继续执行：

0x3005 NOT R3,R3(0x96FF)

运行结果如图3.

可见，程序正确把 0x0000 反转为 0xFFFF 并且正确设置条件码。

3.3 AND 功能测试

在之前程序的基础上继续执行：

0x3006 AND R5,R0,R1(0x5A01)

0x3007 AND R5,R1,#2(0x5AA2)

运行结果如图4.

程序在寄存器模式与立即数模式下均正常运行并且正确设置条件码。

3.4 BR 功能测试

在之前程序的基础上继续执行：

0x3008 BRN #4(0x0804)

0x3009 BRP #4(0x0204)

运行结果如图5.

在上一条 AND 后，N=0,Z=0,P=1, 所以 BRN 不做任何事情，BRP 正确跳转到自增后 PC+offset 的位置。

```
(base) PS E:\LC3> .\lc3sim addtest
LC-3 Simulator

Read 10 words from program into memory.

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 0
PC                : 0x3000
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x0000
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

(a) 图 1

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 1
PC                : 0x3001
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x0001
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

(b) 图 2

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 2
PC                : 0x3002
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x0001
1: 0x0003
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

(c) 图 3

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 3
PC                : 0x3003
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x0001
1: 0x0002
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

(d) 图 4

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 4
PC                : 0x3004
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x0001
1: 0x0002
2: 0x0003
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

(e) 图 5

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 5
PC                : 0x3005
CCs: N = 1  Z = 0  P = 0
Registers:
0: 0x0001
1: 0x0002
2: 0x0003
3: 0x0000
4: 0xffff
5: 0x0000
6: 0x0000
7: 0x0000
```

(f) 图 6

图 2: ADD 功能测试

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 6
PC                : 0x3006
CCs: N = 1  Z = 0  P = 0
Registers:
0: 0x0001
1: 0x0002
2: 0x0003
3: 0xffff
4: 0xffff
5: 0x0000
6: 0x0000
7: 0x0000
```

图 3: NOT 功能测试

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 7
PC                : 0x3007
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x0001
1: 0x0002
2: 0x0003
3: 0xffff
4: 0xffff
5: 0x0000
6: 0x0000
7: 0x0000
```

(a) 寄存器模式

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 8
PC                : 0x3008
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x0001
1: 0x0002
2: 0x0003
3: 0xffff
4: 0xffff
5: 0x0002
6: 0x0000
7: 0x0000
```

(b) 立即数模式

图 4: AND 功能测试

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 9
PC                : 0x3009
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x0001
1: 0x0002
2: 0x0003
3: 0xffff
4: 0xffff
5: 0x0002
6: 0x0000
7: 0x0000
```

(a) 不跳转

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 10
PC                : 0x300e
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x0001
1: 0x0002
2: 0x0003
3: 0xffff
4: 0xffff
5: 0x0002
6: 0x0000
7: 0x0000
```

(b) 跳转

图 5: BR 功能测试

3.5 TRAP 功能测试

运行如下代码：

.ORIG 0x3000 (0x3000)

0x3000 TRAP x25(0xF025)

运行结果如图6.

程序正确保存 PC 到 R7. 由于全部初始化为 0，所以无论 TRAP 什么，都会跳转到 0x0000 然后触发 HALT。

```
(base) PS E:\LC3> .\lc3sim trap
LC-3 Simulator

Read 1 words from program into memory.

LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 1
PC                : 0x0000
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x0000
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x3001
```

```
LC-3-SIM> r 1

Simulating for 1 cycles...

Simulator halted

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 1
PC                : 0x0000
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x0000
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x3001

LC-3-SIM>
```

图 6: TRAP 功能测试

3.6 LEA 与 JMP 功能测试

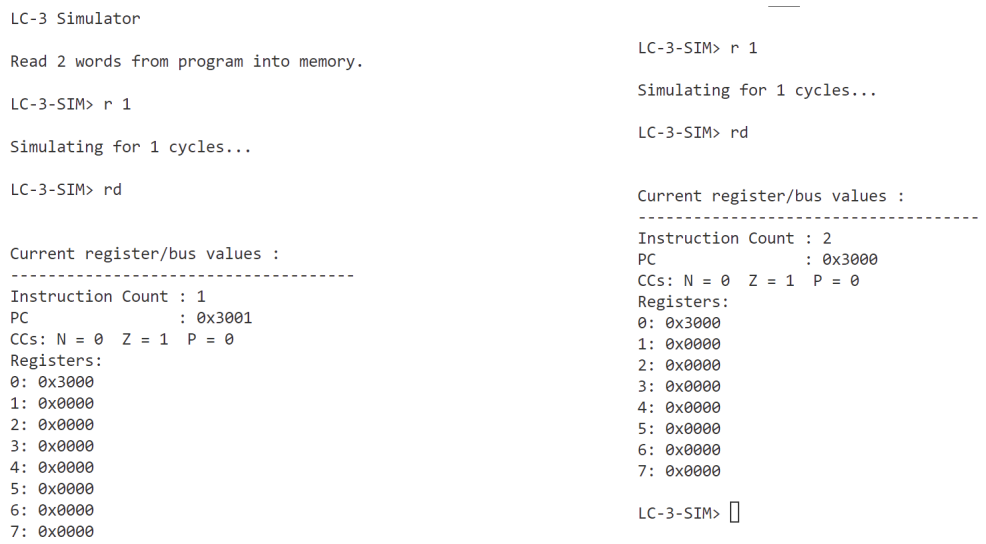
运行如下代码：

.ORIG 0x3000

0x3000 LEA R0,# -1(0xE1FF)

0x3001 JMP R0(0xC000)

运行结果如图7：



(a) LEA 功能测试 (b) JMP 功能测试

图 7: LEA 与 JMP 功能测试

则 LEA 指令正确把地址加载到指定寄存器中，JMP 正确跳转到对应寄存器中存储的地址，不改变条件码。由于 RET 指令就是 JMP R7, 后面不再单独测试。

3.7 ST 与 LD 功能测试

运行如下代码：

.ORIG 0x3000

0x3000 LEA R0,# -1(0xE1FF)

0x3001 ST R0, # 7(0x3007)

0x3002 LD R1, #6(0x2206)

运行结果如图8：则 ST 指令正确把源寄存器的值存储到指定地址中，LD 正确把数据加载到指定寄存器并设置条件码。

3.8 STI 与 LDI 功能测试

由于全部初始化为 0，所以 STI 和 LDI 都会访问 0x0000. 运行如下代码, 使用 mdump 查看内存：

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 2
PC                : 0x3002
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x3000
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

(a) ST 功能测试

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 3
PC                : 0x3003
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x3000
1: 0x3000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

(b) LD 功能测试

图 8: ST 与 LD 功能测试

.ORIG 0x3000

0x3000 ADD R0,R0,# 1(0x1021)

0x3001 STI R0, # 7(0xB007)

0x3002 LDI R1, #6(0xA206)

运行结果如图9：

则 STI 与 LDI 能正确存储与加载，且 LDI 正确设置条件码。

3.9 STR 与 LDR 功能测试

运行如下代码, 使用 mdump 查看内存. 加入 ADD R4,R4,#1 是为了改变条件码，便于对比查看。

.ORIG 0x3000

0x3000 LEA R1,# -1(0xE3FF)

0x3001 ADD R0,R0,# -1(0x103F)

0x3002 ADD R4,R4,#1(0x1921)

0x3003 STR R0,R1,#7(0x7047)

0x3004 LDR R2,R1,#7(0x6447)

运行结果如图10：

则 STR 与 LDR 能正确存储与加载，且 LDR 正确设置条件码。

3.10 JSR 与 JSRR(与 RET) 功能测试

执行以下命令, 其中的 BR 是 0x0000, 无论如何不跳转，发挥 NOP 的作用。

.ORIG 0x3000

```
LC-3-SIM> r 2

Simulating for 2 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 2
PC                : 0x3002
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x0001
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

(a) STI

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 3
PC                : 0x3003
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0x0001
1: 0x0001
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

(b) LDI

```
LC-3-SIM> mdump 0x3005 0x300F


Memory content [0x3005..0x300f] :
-------------------------------------
  0x3005 (12293) : 0x00
  0x3006 (12294) : 0x00
  0x3007 (12295) : 0x00
  0x3008 (12296) : 0x00
  0x3009 (12297) : 0x00
  0x300a (12298) : 0x00
  0x300b (12299) : 0x00
  0x300c (12300) : 0x00
  0x300d (12301) : 0x00
  0x300e (12302) : 0x00
  0x300f (12303) : 0x00
```

(c) 内存查看

```
LC-3-SIM> mdump 0x0000 0x0010


Memory content [0x0000..0x0010] :
-------------------------------------
  0x0000 (0) : 0x01
  0x0001 (1) : 0x00
  0x0002 (2) : 0x00
  0x0003 (3) : 0x00
  0x0004 (4) : 0x00
  0x0005 (5) : 0x00
  0x0006 (6) : 0x00
  0x0007 (7) : 0x00
  0x0008 (8) : 0x00
  0x0009 (9) : 0x00
  0x000a (10) : 0x00
  0x000b (11) : 0x00
  0x000c (12) : 0x00
  0x000d (13) : 0x00
  0x000e (14) : 0x00
  0x000f (15) : 0x00
  0x0010 (16) : 0x00
```

(d) 内存查看

图 9: STI 与 LDI 功能测试

0x3000 LEA R0,# 5(0xE005)

0x3001 JSR #3(0x4803)

0x3002 JSRR R0(0x4000)

0x3003 BR(0x0000)

0x3004 BR(0x0000)

0x3005 RET(0xC1C0)

0x3006 RET(0xC1C0)

运行结果如图：

3.11 总结

总而言之，ADD,NOT,AND,BR,TRAP,LEA,JMP,ST,LD,STI,LDT,STR,LDR,JSR 指令在各个模式下均工作正常。

```
LC-3-SIM> r 3

Simulating for 3 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 3
PC                : 0x3003
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0xffff
1: 0x3000
2: 0x0000
3: 0x0000
4: 0x0001
5: 0x0000
6: 0x0000
7: 0x0000
```

(a) 前期准备

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 4
PC                : 0x3004
CCs: N = 0  Z = 0  P = 1
Registers:
0: 0xffff
1: 0x3000
2: 0x0000
3: 0x0000
4: 0x0001
5: 0x0000
6: 0x0000
7: 0x0000
```

(b) STR

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 5
PC                : 0x3005
CCs: N = 1  Z = 0  P = 0
Registers:
0: 0xffff
1: 0x3000
2: 0xffff
3: 0x0000
4: 0x0001
5: 0x0000
6: 0x0000
7: 0x0000
```

(c) LDR

```
LC-3-SIM> mdump 0x3000 0x300A


Memory content [0x3000..0x300a] :
-------------------------------------
  0x3000 (12288) : 0xe3ff
  0x3001 (12289) : 0x103f
  0x3002 (12290) : 0x1921
  0x3003 (12291) : 0x7047
  0x3004 (12292) : 0x6447
  0x3005 (12293) : 0x00
  0x3006 (12294) : 0x00
  0x3007 (12295) : 0xffff
  0x3008 (12296) : 0x00
  0x3009 (12297) : 0x00
  0x300a (12298) : 0x00
```

(d) 内存查看

图 10: STR 与 LDR 功能测试

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 1
PC                : 0x3001
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x3006
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x0000
```

(a) 准备工作

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 2
PC                : 0x3005
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x3006
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x3002
```

(b) JSR

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 3
PC                : 0x3002
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x3006
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x3002
```

(c) RET

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 4
PC                : 0x3006
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x3006
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x3003
```

(d) JSRR

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 5
PC                : 0x3003
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x3006
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x3003
```

(e) RET

```
LC-3-SIM> r 1

Simulating for 1 cycles...

LC-3-SIM> rd


Current register/bus values :
-------------------------------------
Instruction Count : 6
PC                : 0x3004
CCs: N = 0  Z = 1  P = 0
Registers:
0: 0x3006
1: 0x0000
2: 0x0000
3: 0x0000
4: 0x0000
5: 0x0000
6: 0x0000
7: 0x3003
```

(f) 顺序执行

图 11: JSR 与 JSRR(与 RET) 功能测试

则 JSR，JSRR 可以正确跳转并保存自增后的 PC 到 R7，RET 可以正确返回。