

作业 5

2025 年 10 月 4 日

前言

虽然英文版也是我写的，但我初版直接用英文完成，没有中文原稿。故使用 AI 翻译成了本文档，并进行了一些修订。语言表达看起来可能很人机，但是我大概看了一下不影响理解，请放心使用。

T1

以下 LC-3 程序有什么问题？执行时可能发生什么？给出两种可能的解决方法。

```
1 .ORIG x3000
2     AND R1, R1, #0
3 PROGRAM ADD R1, R1, #1
4     TRAP x23          ; IN
5     LEA R0, MESSAGE
6     TRAP x22          ; PUTS
7 MESSAGE .STRINGZ "Program reporting"
8 .END                 ; HALT
```

T2

在 LC-3 编程实践中，使用 LC3Tools 运行和调试程序时，你可能会观察到以下现象：程序执行 **HALT** 指令并停止后，寄存器中存储的值与你的预期有很大差异。在这之前我们一般通过设置断点来解决这个问题。

请参考以下截图。

Registers				Memory			
R0	x0000	0		① ▶ x3000	x5020	20512	AND R0, R0, #0
R1	x7FFF	32767		① ▶ x3001	x1021	4129	ADD R0, R0, #1
R2	x0003	3		① ▶ x3002	x1221	4641	ADD R1, R0, #1
R3	x0004	4		① ▶ x3003	x1461	5217	ADD R2, R1, #1
R4	x0005	5		① ▶ x3004	x16A1	5793	ADD R3, R2, #1
R5	x0006	6		① ▶ x3005	x18E1	6369	ADD R4, R3, #1
R6	x2FFE	12286		① ▶ x3006	x1B21	6945	ADD R5, R4, #1
R7	x0008	8		① ▶ x3007	x1D61	7521	ADD R6, R5, #1
PSR	x0002	2	CC: Z	① ▶ x3008	x1FA1	8097	ADD R7, R6, #1
PC	x036C	876		① ▶ x3009	xF025	61477	HALT
MCR	x0000	0		① ▶ x300A	x0000	0	

- 1) 请解释为什么会发生这种情况。
- 2) 下面给出了 **HALT** 服务例程的片段。通常情况下，地址 **x036C** 处的指令会执行吗？如果不会，请解释这条指令存在的意义。

地址	指令
x0366	<i>LEA R0, TRAP_HALT_MSG</i>
x0367	<i>PUTS</i>
x0368	<i>LDI R0, OS_MCR</i>
x0369	<i>LD R1, MASK_HI</i>
x036A	<i>AND R0, R0, R1</i>
x036B	<i>STI R0, OS_MCR</i>
x036C	<i>BRnzp TRAP_HALT</i>

T3

请列出 TRAP 和中断的至少三点区别。

T4

- 1) 什么是系统空间？
- 2) 什么是用户空间？
- 3) 在用户模式下哪些内存地址是不可访问的？
- 4) 什么是 ACV？
- 5) 是什么决定了当前状态下的可访问内存范围？
- 6) 列出从用户模式转换到特权模式的所有方法。

T5

以下是一个 LC-3 程序片段。执行完标签 F 处代码后，寄存器 R0 的值是多少？

注意：你需要展示你的解题过程。

```

1  A  LD R0, E
2      LEA R7, E
3  B  .FILL x1021
4      ADD R2, R0, #0
5      LD R1, D
6      LD R3, B
7      ADD R3, R1, R3
8      ST R3, C
9  C  .BLKW 1
10     RET
11  D  .STRINGZ "!"
12  E  LD R1, A
13  F  ADD R0, R0, R1

```

T6

在执行 LC-3 程序期间，程序中的一条指令从时钟周期 T 开始执行，需要 15 个周期才能完成。表格列出了处理此指令期间需要使用总线的全部五个时钟周期。表格显示了每个时钟周期的：哪个时钟周期、状态机的状态、总线上的值，以及在该时钟周期期间激活的重要控制信号。

- 1) 填写表格中缺失的条目。
- 2) 正在处理的是什么指令？
- 3) 该指令在内存中的什么位置？
- 4) 内存读取或写入需要多少个时钟周期？
- 5) 上面的信息足够让你知道三个内存位置的内容。它们的地址是什么？它们的内容是什么？

周期	状态	总线	该周期的重要控制信号
T	18	x3010	LD.MAR=1,LD.PC=1,PCMux=PC+1,GatePC=1
T+4			
T+6		x3013	
T+10		x4567	
T+14		x0000	LD.REG=1,LD.CC=1,GateMDR=1,DR=001

T7

阅读以下程序：

```

1      .ORIG x3000
2      LD R0, A
3      LD R1, B
4      BRz DONE
5      ----- (a)
6      ----- (b)
7      BRnzp AGAIN
8 DONE  ST R0, A
9      HALT
10 A    .FILL x0--- (c)
11 B    .FILL x0001
12      .END

```

该程序只使用 R0 和 R1 寄存器。注意 (a) 和 (b) 行表示两条缺失的指令。完成 (c) 行。还要注意程序中必须有一条指令标记为 AGAIN，该标签缺失。

程序执行后，A 的内容是 x1800。

在执行过程中，我们在每个时钟周期都检查计算机并记录了某些时钟周期的信息，产生了下表。表格按收集信息的周期数排序。注意每次内存访问需要五个时钟周期。

填写程序中缺失的指令，并通过添加标签 AGAIN 来完成程序。同时，填写表格中缺失的信息。给定 A 和 B 的值，你能否推断出这个程序的功能？

Cycle Number	State Number	Control Signals			
1	18	LD.MAR: <input type="text"/>	LD.REG: <input type="text"/>	GateMDR: <input type="text"/>	
		LD.PC: <input type="text"/>	PCMUX: <input type="text"/>	GatePC: <input type="text"/>	
<input type="text"/>	0	LD.MAR: <input type="text"/>	LD.REG: <input type="text"/>	BEN: <input type="text"/>	
		LD.PC: <input type="text"/>	LD.CC: <input type="text"/>		
<input type="text"/>		LD.REG: <input type="text" value="1"/>	DR: <input type="text" value="000"/>	GateMDR: <input type="text"/>	
		GateALU: <input type="text"/>	GateMARMUX: <input type="text"/>		
54	1	LD.MAR: <input type="text"/>	ALUK: <input type="text"/>	GateALU: <input type="text"/>	
		LD.REG: <input type="text"/>	DR: <input type="text"/>	GatePC: <input type="text"/>	
86	22	ADDR1MUX: <input type="text"/>	ADDR2MUX: <input type="text"/>		
77		LD.PC: <input type="text"/>	LD.MAR: <input type="text"/>	PCMUX: <input type="text"/>	
101	15				

T8

单链表中的一个元素有两个部分：数据和指向下一个元素的指针。数据和指针位于相邻的内存位置，数据在指向下一个元素的指针之前。它们都是 16 位整数。列表元素的地址是该元素的起始地址。示例如下：

List with 4 elements

x4000	x0041
x4001	x4100
x4100	x0042
x4101	x5000
x5000	x0043
x5001	x5090
x5090	x0044
x5091	x0000

你需要实现下面的函数，它以 R0 作为唯一参数。R0 是现有列表元素的地址，你需要从列表中删除该元素（该元素不是最后一个元素）。你还需要将该元素的数据输出到监视器（为方便起见，我们将数据视为 8 位字符并忽略其他位）。当函数返回时，R0 应该携带被删除元素的下一个元素的地址。所有其他寄存器均为 caller save。

注意：元素的地址允许变更。

```
1  ----  ----  ----  #0
2  ----  ----  ----
3  TRAP  x21
4  ----  --,  R2,  --
5  ----  R1,  R0,  --
6  ----  R2,  ----
7  ----  ----  ----  #0
8  ----  ----  ----  #1
9  ----  R2,  ----
10 RET
```