ICS HW5

PB22111679 孙婧雯

T1

`.END` 是一个伪指令，不同于真正的汇编指令，不会被执行，也不会让机器停止，只是告诉汇编器到哪里停止汇编的一个标识符。

`HALT` 是一个汇编指令，机器运行到这一句会真正停止运行，在这一句后面的指令都不会被执行。

T2

仿照栈的概念（访问规则），可以定义 `queue` 为：先存入的东西先取走，称为“入队”；后存入的东西后取走，称为“出队”。和栈相似，队列也是由一段连续的内存空间和寄存器组成的，但是有两个寄存器（队列头、尾指针）。

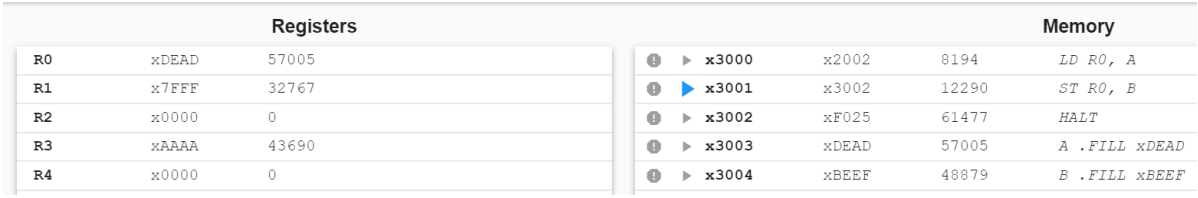
T3

根据 `.FILL` 后的十六进制数推断，该程序想要上载 `R0` `xDEAD` 地址处的值，并储存到 `xBEEF` 。伪指令 `.FILL` 不应该放在程序前端，应该放在 `HALT` 后面。

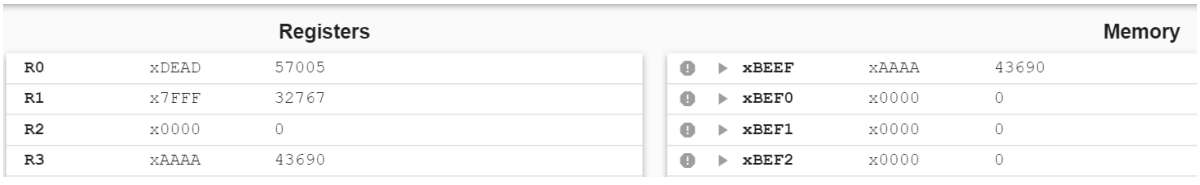
修改之后我们可以看到（预先在 `xDEAD` 处存入一个值 `xAAAA` ）：

① `LD` 语句生效，`R0` 可以取到 `xDEAD`

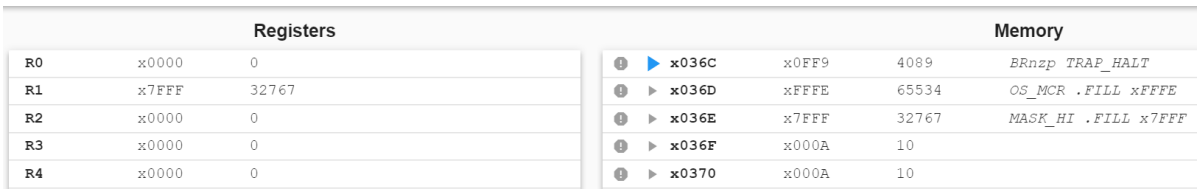
(图中 `R3` 存的 `xAAAA` 是做题过程中调试存入的，可以忽略)



② `ST` 语句生效，`xBEEF` 被存入了 `xAAAA`



如果不修改，按照原程序在 LC-3 tools 运行，程序运行到第一行就自动进入 `HALT` 的上下文了。



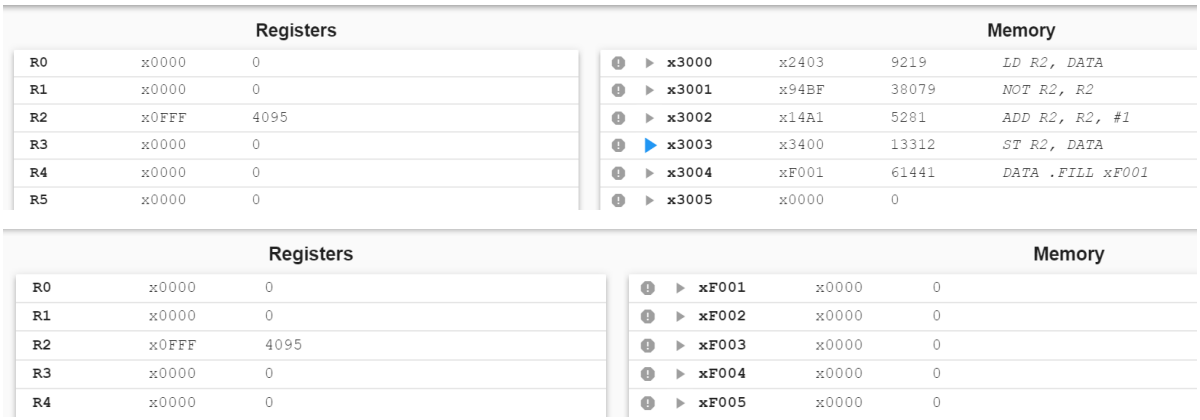
T4

每个模块有自己的 `LABLE` 表，在没有 `.EXTERNAL` 声明的前提下，只会从自己的 `LABLE` 表中查找对应标签的地址，所以两个模块运行时不会有冲突。

T5

1）`xF001` 的补码是 `x0FFF`。

2）程序无法把补码返存到 `xF001` 即 `DATA`，如图：`R2` 可以正确存到 `x0FFF`，但地址 `xF001` 处没有值。



3）程序将不会结束，因为末位只有 `.END`，没有 `HALT`。

T6

`.FILL`：让汇编器把后面跟的十六进制数放在引用此伪指令的位置。只有一个数。

`.BLKW`：提示汇编器从引用此伪指令的位置开始，有多少连续的内存会被占用。可以有多个数（内存空间）。

`.STRINGZ`：让汇编器把后面跟的字符串依次放入内存空间，以引用此伪指令的位置为起始位置，并在字符串结尾补 `NULL`。可以有多个数（内存空间）。

T7

补全如下：

```
    .ORIG x3000
    LD R0, A
    LD R1, B
X   NOT R2, R0       (a)
    ADD R2, R2, #1   (b) ; R2 = -R0
    ADD R2, R2, R1       ; R2 = R1 - R0
    BRz DONE         (c) ; if R2 == 0 done
    ADD R1, R1, #-1      ; else increamenting and decrementing
    ADD R0, R0, #1   (d)
    BRnzp X
    DONE ST R1,C
```

```
    TRAP x25
    A .BLKW 1
    B .BLKW 1
    C .BLKW 1
.END
```

T9

1）`PUSH H` 之前，A B 入栈，B 弹出，C 入栈，C 弹出，D E F 入栈，F 弹出，G 入栈， D E G 弹出，H 入栈，则栈中所含元素为 A H 。

2）`PUSH` 指令数和 `POP` 指令数之差最大时，栈内所含元素最多，即 `PUSH F` 或 `PUSH G` 语句执行之后，有 4 个元素。

3）一共执行了 13 次元素入和元素出的操作（包括 `PUSH` `POP` `ENQUEUE` `DEQUEUE` ），并且其中没有非法操作，则最终队列为空，不含任何剩余元素。