hw5 ans

t1

AND 可能会被作为操作码而不是标签，因此可能会导致汇编器报错。

t2

a) .END 不是指令，不会被执行

b) 不会让机器停止

c) 只是告诉汇编器到哪里停止汇编的一个标识符。

t3

可以存在多个.END伪指令。

如果想使用多个.END伪指令，则需要为每一个 .END伪指令配对一个.ORIG伪指令来使用。类似于C语言中 `{}` 括号的用法。

t4

a) ADD 指令立即数只有5位。对于 ADD R3, R3, #30，立即数太大。
我们可以将指令拆分为两条 ADD R3, R3, #15。

b) 汇编器会检测到立即数编码错误。

t5

每个模块有自己的符号表，在没有 `.EXTERNAL` 声明的前提下，只会从自己的符号表中查找对应标签的地址，因此不会产生问题。

t6

`DATA .FILL x8002` 的情况下，x8002 = (无符号数) 32770 = (有符号数)-32766 第一次执行LOOP后R4为x7FFF,溢出变为正数继续循环，LOOP执行 10924次

`DATA .FILL x8003` 的情况下，x8003 = (无符号数) 32771 = (有符号数)-32765 第一次执行LOOP后R4为x8000,为负数，执行跳转，LOOP执行 1次

t7

```
MUL         ST  R7, SAVER7
            ST  R0, SAVER0
            ST  R1, SAVER1
            ST  R2, SAVER2
            ST  R5, SAVER5
            AND R2, R2, #0
            JSR POP
            ADD R1, R0, #0
            JSR POP
            ADD R1, R1, #0
            BRz DONE
AGAIN       ADD R2, R2, R0
            ADD R1, R1, #-1
            BRp AGAIN
DONE        ADD R0, R2, #0
            JSR PUSH
            LD R7, SAVER7
            LD R0, SAVER0
            LD R1, SAVER1
            LD R2, SAVER2
            LD R5, SAVER5
            RET
```

t8

a) `R7` 是调用者保存寄存器，`R0` 和 `R2` 是被调用者保存寄存器。

b) $f(n) = f(n-1) + f(n-2) + 21$，包含了递归调用两个 `n` 更小的子过程的开销以及维护寄存器，调用结果相加的开销。

c) 使用内存记录 $f(n)$ 是否被计算过以及 若 $f(n)$ 被计算过，它对应的值。

t9

```
PUSH        ADD R6, R6, #-2
            STR R0, R6, #0
            STR R1, R6, #1

POP         LDR R0, R6, #0
            LDR R1, R6, #1
            ADD R6, R6, #2
```

t10

a) ABCD → BDCA

```
push A
push B
pop  B
push C
push D
pop  D
pop  C
pop  A
```

b) 不可能输出DBCA。因为D出栈时，BC都在栈中，且由于C在B后入栈，C必然比B先出栈，不可能出现DBC的顺序。

c) 由 $C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$ 当 $n = 5$ 时，有 $C_5 = \frac{1}{5+1}\binom{10}{5} = \frac{10!}{5!\cdot 6!} = 42$

T11

memory access: 2 cycles.

cycle number	state number	information
11	27	LD.REG:1 DRMUX: 00 GateMDR: 1 LD.CC: 1 GateALU: 0 GatePC: 0
16	30	LD.MDR: 0 MDR: x2209 IR: x2009 LD.IR: 1
50	1	LD.REG:1 MDR: x14A1 BUS: x0001 DR: 010 GateMDR: 0
57	1	PC: x3007 IR: x1040 BUS: x0003 GateALU: 1 GatePC: 0
65	22	ADDR1MUX: 0 ADDR2MUX: 10 LD.PC: 1 PC: x3008 PCMUX: ADDER

a) ADD R2,R2,#1

b) ADD R0,R1,R0

c) B .fill #2

The student was trying to divide the value at A by the value and B and store the quotient at C. To fix the program, the BRnzp AGAIN should be changed to BRp AGAIN