Hw4

T1

1. `LDR`: 取址 + 取操作数，总共两次；
2. `STI`: 取址 + 取操作数 + 存储结果，总共三次；
3. `TRAP`: 取址 + 取操作数，总共两次。

T2

1. 0001 001 001 1 00000; ADD R1, R1, #0
2. 0101 001 001 000 001; AND R1, R1, R1
3. 0000 111 000000000; BRnzp #0

T3

1. x123B;
2. 1010 110 00011 1111; LDI R6, #63

T4

1. Similarity: 都将程序跳到 PC - #171 的位置继续执行；
2. Difference:
    - **JSR** 操作还将当前 **PC** 值存到 **R7** 中，可以通过 **RET** 指令返回；
    - **BRnzp** 仅跳转，无返回。

T5

（以下答案均以当前指令位置为原点）

- 对于 **BR**、**LD**、**LDI**、**LEA**、**ST**、**STI** 操作的 PCoffset9，偏移范围为 $\text{当前地址} - 255 \sim \text{当前地址} + 256$；
- 对于 **JSR** 操作的 PCoffset11，偏移范围为 $\text{当前地址} - 1023 \sim \text{当前地址} + 1024$；
- 对于 **LDR**、**STR** 操作的 offset6，偏移范围为 $\text{当前地址} - 31 \sim \text{当前地址} + 32$。

T6

- $R_0 = \text{x}0D31$
- $R_1 = \text{x}D000$
- $R_2 = \text{x}2002$
- $R_3 = \text{x}3002$

T7

- ```
  ; use sequential construct

  0101 000 000 1 00000; AND R0, R0, #0
  0001 000 000 1 00001; ADD R0, R0, #1 ; R0 = F1
  0001 001 000 1 00000; ADD R1, R0, #0 ; R1 = F2
  0001 011 000 000 001; ADD R3, R0, R1 ; R3 = F3 = F1 + F2
  0001 010 001 000 011; ADD R2, R1, R3 ; R2 = F4 = F2 + F3
  ```

- ```
  ; use iterative construct

                       ;         .ORIG x3000
  0101 000 000 1 00000;          AND R0, R0, #0
  0001 011 000 1 10110;          ADD R3, R0, #-10 ; 计数器
  0001 000 000 1 00001;          ADD R0, R0, #1
  0001 001 000 1 00000;          ADD R1, R0, #0
  0001 011 011 1 00010;          ADD R3, R3, #2

  0000 011 000000101  ; LOOP     BRzp DONE
  0001 010 000 000 001;          ADD R2, R0, R1
  0001 000 001 1 00000;          ADD R0, R1, #0
  0001 001 010 1 00000;          ADD R1, R2, #0
  0001 011 011 1 00001;          ADD R3, R3, #1
  0000 111 111111010  ;          BRnzp LOOP

  1111 0000 00100101  ; DONE     TRAP x25
                       ;         .END
  ```

T8

```
1001 100 001 111111
1001 101 010 111111
0101 110 100 000 101
1001 011 110 111111
```

T9

- 根据条件码判断 **BR** 指令是否进行条件跳转。

T10

- $R_1(R_1-1)/2$