Homework 4

PB20111686 Ruixuan Huang

T1

```
set R4 as 0;
set R0 as ~R1;
set R0 as R0 + 1; // 2's complement code of R1
set R0 as R0 + R2; // 2's complement code of R2-R1
if R0 >= 0 then goto 7th instruction(halt);
set R4 as R4 + 1;
halt;
```

Using R4 to represent R2-R1 is negative(if so, R4=1) or not.

T2

```
set R0 as 0;
set R5 as R1 & 1;
if R5 == 0 then goto 5th instruction(halt);
set R0 as R0 + 1;
halt;
```

Using R0 to represent the lowest bit of R1 is true(if so, R0 = 1) or not.

T3

The full value of x1000 should be

```
0001 101 000 1 11000
```

T4

(a).

```
set R0 as 0;
set R0 as R0 + 5; // R0==5
set R1 as x3007; // R1==4
set R0 as R0 + R0; // R0==10
set R1 as R1 - 1; // R1==3
if R1 > 0 then go to 4th instruction;
halt;
```

Caculating $5 \times 2^{\mathrm{M[x3007]}}$ and store answer in R0.

(b).

PC: x3006

R0: 0000000001010000

R1~R7: 0000000000000000

{N,Z,P} = 010

(c).

$10 + 10 + 16 + 10 \times 4 + 10 \times 4 + 11 \times 3 + 10 = 159.$

T5

```
set R0 as R0 - 16; // R0==-16
set R1 as value of (x3002 + xFE = ) x3100;
if R1 == 0 then goto x3007; // x3002
if R1 >= 0 then goto x3005;
set R0 as R0 + 1;
set R1 as R1 << 1; // x3005
goto x3002;
set R0 as ~R0; // x3007
set R0 as R0 + 1;
halt;
```

Count the number of 0 at R1.

> *During each cycle, the program peeks the highest bit of R1, which representing the value of R1 is positive or not. If being negative, the R0 will be added by 1.*

T6

```
set R0 as (xFF + x3001 = ) x3100;
? // x3001
? // x3002
set R0 as R0 + 1;
goto x3001;
set (x3006 + x4A = ) x3050 as the value R0 points;
halt;
```

x3001: 0110 001 000 000000

x3002: 0000 100 000000010

> *x3001 gives the value R0 points to R1, x3002 examines whether R1 is negative or not.*

> *For giving the assumption that there must be one negative value after x3100, we don't need consider whether the pointer exceeds x4000 or not.*

T7

The LDI instruction actually contains two memory reads, and the STI instruction contains one read and one write.

T8

```
// descriptions of 3 instructions
set R1 as R1;
goto PC + 1;
// attention: before excuting, the PC has already ++!
nothing occurred;
```

The 3rd instruction is the suitest. The 2nd instruction is wrong because it will skip one instruction. Even if the 1st instruction does nothing to the data of R1, it may change the condition codes.

T9

```
// descriptions of 4 instructions
set R4 as ~R1
?
set R6 as R4 & R5
?
```

Using $a|b = \overline{\bar{a}\&\bar{b}},$ we can give the answer:

2). 1001 101 010 111111

3). 1001 011 110 111111