Lab Report - 04

PB20111686 Ruixuan Huang

Lab Goal

Task 1 : 4 bits in the given code are missing. Using given code and the result state of registers to find out the missing bits. Before the program starts, values in the registers except PC are 0.

Task 2 : The given code is used to calculate the remainder of one number divided by 7. 15 bits in the code are missing. Find out the missing bits.

Task 1

Firstly, I translated the machine code into assembly.

```
; For easily speaking, I assume ".ORIG x3000"
1110 010 000001110      ; x3000, R2 <- x300F; R2 = x300F
0101 000 000 1 00000    ; x3001, R0 <- R0 & #0; R0 = 0
0100 1 0000000000x      ; x3002, R7 <- x3003, PC <- x3003 + ?(0 or 1)
1111 0000 00100101      ; x3003, TRAP x25, HALT;
```

If `x` in instruction `x3002` is zero, the program will simply halt at present, and no function carried. That sounds abnormal. So the `x` in this instruction is `1`.

```
1110 010 000001110      ; x3000, R2 <- x300F; R2 = x300F
0101 000 000 1 00000    ; x3001, R0 <- R0 & #0; R0 = 0
0100 1 00000000001      ; x3002, R7 <- x3003, PC <- x3004
1111 0000 00100101      ; x3003, TRAP x25, HALT;
0111 111 010 000000     ; x3004, M[R2] <- R7; M[x300F] <- x3003
0001 010 010 1 0x001    ; x3005, R2 <- R2 + ?(1 or 9)
0001 000 000 1 00001    ; x3006, R0 <- R0 + 1;
0010 001 000010001      ; x3007, R1 <- M[x3008 + x11] = M[3019] = #5
0001 001 x01 1 11111    ; x3008, R1 <- R?(1 or 5) - 1
```

In instruction `x3008`, `x01` is one oprand register. It can be `R1` or `R5`. However, `R5` doesn't occurred in the context. So here I can determine the missing bit is `0`.

```
1110 010 000001110      ; x3000, R2 <- x300F; R2 = x300F
0101 000 000 1 00000    ; x3001, R0 <- R0 & #0; R0 = 0
0100 1 00000000001      ; x3002, R7 <- x3003, PC <- x3004
1111 0000 00100101      ; x3003, TRAP x25, HALT;

0111 111 010 000000     ; x3004, M[R2] <- R7; M[x300F] <- x3003
0001 010 010 1 0x001    ; x3005, R2 <- R2 + ?(1 or 9)
0001 000 000 1 00001    ; x3006, R0 <- R0 + 1;
0010 001 000010001      ; x3007, R1 <- M[x3008 + x11] = M[3019] = #5
0001 001 001 1 11111    ; x3008, R1 <- R1 - 1
0011 001 000001111      ; x3009, M[x300A + xF] = R1; M[x3019] = R1
0000 010000000001       ; x300A, if(R1 == 0) goto x300C
0100 1 11111111000      ; x300B, R7 <- x300C, PC <- x3004

0001 010 010 1 11111    ; x300C, R2 <- R2 - 1
01x0 111010000000       ; x300D
1100 000 111 000000     ; x300E, PC <- R7
```

Let's talk about instruction `x300D`. If `x` is zero, the instuction will be explained as `[0100 1 11010000000]`. When carried, following influence will be made.

```
;R7 = x300E
;PC = x300E - #384
```

It is impossible. Because the counter has excceeded the address space. If `x` is one, the instuction will be explained as `[0110 111 010 000000]`. When carried, following influence will be made.

```
;R7 = M[R2]
```

This explanation is more trustable. And since we haven't used instruction `x3003`, here `R2` is loaded with `x3003` and then PC is loaded with it. So that's reasonable. Here we note that `R2` stores one address. If the `x` in instruction `x3005` is one, it will obviously excceed the address space easily. So we finally determine the last missing bit.

```
;Full code
1110 010 000001110      ; x3000, R2 <- x300F; R2 = x300F
0101 000 000 1 00000    ; x3001, R0 <- R0 & #0; R0 = 0
0100 1 00000000001      ; x3002, R7 <- x3003, PC <- x3004
1111 0000 00100101      ; x3003, TRAP x25, HALT;

0111 111 010 000000     ; x3004, M[R2] <- R7; M[x300F] <- x3003
0001 010 010 1 00001    ; x3005, R2 <- R2 + 1
0001 000 000 1 00001    ; x3006, R0 <- R0 + 1;
0010 001 000010001      ; x3007, R1 <- M[x3008 + x11] = M[3019] = #5
0001 001 001 1 11111    ; x3008, R1 <- R1 - 1
0011 001 000001111      ; x3009, M[x300A + xF] = R1; M[x3019] = R1
0000 010000000001       ; x300A, if(R1 == 0) goto x300C
0100 1 11111111000      ; x300B, R7 <- x300C, PC <- x3004

0001 010 010 1 11111    ; x300C, R2 <- R2 - 1
0110 111010000000       ; x300D, R7 = M[R2]
1100 000 111 000000     ; x300E, PC <- R7
0000000000000000        ; x300F
0000000000000000        ; x3010
0000000000000000        ; x3011
0000000000000000        ; x3012
0000000000000000        ; x3013
0000000000000000        ; x3014
0000000000000000        ; x3015
0000000000000000        ; x3016
0000000000000000        ; x3017
0000000000000000        ; x3018
0000000000000101        ; x3019
```

Using LC-3 tools to verify the correctness of the program code.

Registers		
R0	x0005	5
R1	x0000	0
R2	x300F	12303 =x300f
R3	x0000	0
R4	x0000	0
R5	x0000	0
R6	x0000	0
R7	x3003	12291 =x3003

Task 2

I will use similar method as task 1.

```
; For easily speaking, I assume ".ORIG x3000"
0010 001 000010101    ;x3000, R1 = M[x3016] = x0120
0100 1 00000001000    ;x3001, R7 = x3002, PC= x300A(goto x300A)
```

```
0101 010 001 1 00111  ;x3002, R2 = R1 & 7
0001 001 010 000 100  ;x3003, R1 = R2 + R4
0001 000 0xx 1 11001  ;x3004, R0 = R? - 7
0000 001 111111011    ;x3005, if(R0 > 0) goto x3002
0001 000 0xx 1 11001  ;x3006, R0 = R? - 7
0000 100 000000001    ;x3007, if(R0 < 0) goto x3009
0001 001 001 1 11001  ;x3008, R1 = R1 - 7
1111 0000 00100101    ;x3009, TRAP x25, halt

0101 010 010 1 00000  ;x300A, R2 = 0
0101 011 011 1 00000  ;x300B, R3 = 0
0101 100 100 1 00000  ;x300C, R4 = 0
0001 010 010 1 00001  ;x300D, R2 = 1
0001 011 011 1 01000  ;x300E, R3 = R3 + 4
0101 101 011 000 001  ;x300F, R5 = R3 & R1
0000 010 000000001    ;x3010, if(R5 == 0)goto x3012
0001 100 010 000 100  ;x3011, R4 = R2 + R4
0001 010 010 000 010  ;x3012, R2 = R2 << 1
0001 011 011 000 011  ;x3013, R3 = R3 << 1
0000 101 111111010    ;x3014, if(R3 != 0) goto x300F
1100 000 111 000000   ;x3015, PC = R7(x3002)

0000000100100000      ;x3016
```

- Firstly I can determine that the third `x` in instructions `x3004` and `x3006` is `1`, because bit[4:3] = `11` instead of `00` (if `x` is zero).
- The HINT in the task note that the program used the term "divided by 8", I determine that the third part of the program is used to do this. To do so, we should use double pointer. One used to read and the other used to write. Here `R3` plays the role as the latter and `R2` the former. So the missing bits in instruction `x3013` represent the oprand `R3`.
- Once the read pointer is zero, the work is done. So the missing bits in instruction `x3014` represent the `R3` is not zero.
- When carrying `x3005`, the PC is `x3006`. Let's find the missing bits in instruction `x3005`. The PC offset 9 is `1xxx11011` so I will find the bits by enumerating the missing part and verify whether the address is possible.

missing parts	goto where	missing parts	goto where
000	`x2F22`	100	`x2FA2`
001	`x2F42`	101	`x2FC2`
010	`x2F62`	110	`x2FE2`
011	`x2F82`	111	`x3002`

Obviously, only when the missing parts being `111` will the instruction be possible.

```
; First two parts of the program
0010 001 000010101    ;x3000, R1 = M[x3016] = x0120, R1 is to be find the remainder
0100 1 00000001000    ;x3001, R7 = x3002, PC= x300A
; After carrying the 3rd part, R4 is R1 / 8
0101 010 001 1 00111  ;x3002, R2 = R1 & 7, get the low 3 bits of R1
0001 001 010 000 100  ;x3003, R1 = R2 + R4
0001 000 001 1 11001  ;x3004, R0 = R1 - 7
0000 001 111111011    ;x3005, if(R0 > 0) goto x3002
0001 000 001 1 11001  ;x3006, R0 = R1 - 7
0000 100 000000001    ;x3007, if(R0 < 0) goto x3009
0001 001 001 1 11001  ;x3008, R1 = R1 - 7
1111 0000 00100101    ;x3009, TRAP x25, halt
```

- In the second part, we note that `R4` is `R1 / 8`, and `R2` is a constantly updated reg so the missing parts in instructions `x3004` and `x3006` are all `01` (representing reg `R1`).

```
;Full code
0010 001 000010101    ;x3000, R1 = M[x3016] = x0120
0100 1 00000001000    ;x3001, R7 = x3002, PC= x300A(goto x300A)
; After carrying the 3rd part, R4 is R1 / 8
0101 010 001 1 00111  ;x3002, R2 = R1 & 7, get the low 3 bits of R1
0001 001 010 000 100  ;x3003, R1 = R2 + R4
0001 000 001 1 11001  ;x3004, R0 = R1 - 7
0000 001 111111011    ;x3005, if(R0 > 0) goto x3001
0001 000 001 1 11001  ;x3006, R0 = R1 - 7
0000 100 000000001    ;x3007, if(R0 < 0) goto x3009
0001 001 001 1 11001  ;x3008, R1 = R1 - 7
1111 0000 00100101    ;x3009, TRAP x25, halt

0101 010 010 1 00000  ;x300A, R2 = 0
0101 011 011 1 00000  ;x300B, R3 = 0
0101 100 100 1 00000  ;x300C, R4 = 0
0001 010 010 1 00001  ;x300D, R2 = 1
0001 011 011 1 01000  ;x300E, R3 = R3 + 8
0101 101 011 000 001  ;x300F, R5 = R3 & R1
0000 010 000000001    ;x3010, if(R5 == 0)goto x3012
0001 100 010 000 100  ;x3011, R4 = R2 + R4
0001 010 010 000 010  ;x3012, R2 = R2 << 1
0001 011 011 000 011  ;x3013, R3 = R3 << 1
0000 101 111111010    ;x3014, if(R3 != 0) goto x300F
1100 000 111 000000   ;x3015, PC = R7(x3002)

0000000100100000      ;x3016
```

Using LC-3 tools to verify the correctness of my program code.

Registers

R0	xFFFA	65530
R1	x0001	1
R2	x0000	0
R3	x0000	0
R4	x0001	1
R5	x0000	0
R6	x0000	0
R7	x3002	12290

The result is stored in `R1`, and $288 \equiv 1 \pmod 7$. So the program is correct.