Exercise 2 Solution

Byte Codes Translation

Please write down the byte codes of the following Y86-64 instructions.

Y86-64 instructions	Byte codes (hex value)
popq %rbx	0xb03f
jne 0x1234567	0x746745230100000000
subq %rax, %rdx	0x6102
mrmovq 0x432(%rax), %rcx	0x501032040000000000
rrmovq %rdx, %rbx	0x2023
call 0x8765432	0x8032547608

Instruction Design

Suppose we are going to implement **cirmovxx V, rB**, which conditionally moves value V to register rB, in our SEQ Y86_64 processor.

Stage	cirmovxx V, rB
Fetch	icode:ifun <- M1[PC] rA:rB <- M1[PC+1] valC <- M8[PC+2] valP <- PC+10
Decode	–
Execute	Cnd <- Cond(CC,ifun) valE <- 0 + valC
Memory	–
Write back	R[rB] <- Cnd?valE:-
PC update	PC <- valP

Jump Table

In Section 3.6.8, we saw a common way to implement switch statements is to create a set of code blocks and then index those blocks using a jump table. Consider the C code shown below for a function `switchv`.

```
long switchv(long idx) {
    long result = 0;
```

```
    switch(idx) {
        case 3:
            result = 0xa;
            break;
        case 1:
        case 5:
        Case 7:
            result = 0xb;
            break;
        case 4:
            result = 0xc;
            break;
        default:
            result = 0xd;
    }
    return result;
}
```

Alice wants to implement **switchv** in Y86-64 using jump table. Since Y86-64 instruction set does not include indirect jump instruction, she decides to get the same effect by combining several of them. Below is part of her solution.

```
jtable:
    .quad LD
    .quad L0
    .quad L1
    .quad L2
    .quad L3
    .quad L4
    .quad L5
    .quad L6
switchv:
    irmovq [1], %r8
    irmovq [2], %r10
    irmovq [3], %r11
    irmovq $0, %rax
    irmovq jtable, %rcx
    rrmovq %rdi, %rdx
    subq %r8, %rdx
    jg jump
    subq %r10, %rdi
    jl jump
locate:
    irmovq $0x8, %r8
```

```
jump:
    mrmovq (%rcx), %rdi
    # "Question 2"
L0:
    [4]
    ret
L1:
    [5]
L2:
    [6]
    ret
L3:
    irmovq $0xc, %rax
    ret
L4:
    jmp L0
L5:
    jmp LD
L6:
     jmp L0
LD:
    irmovq $0xd, %rax
```

addq %r8, %rcx
subq %r11, %rdi
jl jump
jmp locate
	ret

1. Please fill in the blanks.

(1) $0x7 (2) $0x1 (3) $0x1

(4) irmovq $0xb, %rax

(5) Jmp LD

(6) irmovq $0xa, %rax

2. The marks "Question 2" stands for indirect jump to *%rdi, please write down a combination of Y86-64 instructions to make that effect. (Hint: use two Y86-64 instructions)

pushq %rdi

ret

Y86-64 Assembly

Consider the following assembly.

```
0x000:                          |       .pos 0
0x000: 30f4 00100000 00000000   | Init: irmovq 0x1000, %rsp
0x00a: 30f5 00100000 00000000   |       irmovq 0x1000, %rbp
0x014: _[1]_                    |       call Main
0x01d: 00                       |       halt
                                |
_[2]_:                          |       .align 8
_[3]_: 33010000 00000000        | Array:  .quad 0x133
0x028: fc0d0000 00000000        |       .quad 0xdfc
0x030: 2f0f0000 00000000        |       .quad 0xf2f
0x038: 33020000 00000000        |       .quad 0x_[4]_
                                |
_[5]_: a05f                     | Main:  pushq %rbp
0x042: 2045                     |       rrmovq %rsp, %rbp
0x044: a03f                     |       ____[6]____
0x046: ____[7]____              |       irmovq Array, %rdx
0x050: 5002 00000000 00000000   |       mrmovq (%rdx), %rax
0x05a: 5032 08000000 00000000   |       mrmovq 8(%rdx), %rbx
0x064: 5012 10000000 00000000   |       mrmovq 0x10(%rdx), %rcx
0x06e: 5022 18000000 00000000   |       mrmovq 0x18(%rdx), %rdx
```

```
_[8]_: ____[9]____              |       addq %rax, %rbx
0x07a: 6112                     |       subq %rcx, %rdx
0x07c: 6131                     |       ____[10]____
0x07e: b03f                     |       popq %rbx
0x080: 70 55000000 00000000     |       jmp End
0x08a: 2054                     | End:  rrmovq %rbp, %rsp
0x08c: b05f                     |       popq %rbp
0x08e: 90                       |       ret
```

1. Please fill in the blanks within above Y86-64 binary and assembly code.

(1) 80 40000000 00000000
(2) 0x020
(3) 0x020
(4) 233
(5) 0x040
(6) pushq %rbx
(7) 30f220000000 00000000
(8) 0x078
(9) 6003
(10) subq %rbx, %rcx

2. Suppose the entry point is 0x000, please calculate the value of below registers after the program HALT.

%rdx	0xffffffff fffff304
%rsp	0x1000
CC.ZF	1
CC.SF	0
CC.OF	0