Homework 2

VE370 - Intro to Computer Organization Summer 2022

* Name: Huang Yucheng ID: 519021910885

Exercise 1

Following memory location has address `0x0F000000` and content `0x15C78933`

0	1	2	3
33	89	C7	15

Write RISC-V assembly instructions to load the byte `C7` as a signed number into register `x20`, then show the content of `x20` after the operations.

Answer:
`lui x20, 0x0F000`
`lb x20, 2(x20)`
and the content of `x20` after the operations should be `0xFFFFFFC7`

Exercise 2

The RISC-V assembly program below computes the factorial of a given input n (n!). The integer input is passed through register `x12`, and the result is returned in register `x10`. In the assembly code below, there are a few errors. Correct the errors.

```
FACT:addi sp, sp, 8
     sw x1, 4(sp)
     sw x12, 0(sp)
     add x18, x0, x12
     addi x5, x0, 2
     bge x12, x5, L1
     mul x10, x18, x10
     addi sp, sp, -8
     jalr x0, 0(x1)
  L1:addi x12, x12, -1
     jal x1, FACT
     addi x10, x0, 1
     lw x12, 4(sp)
     lw x1, 0(sp)
     addi sp, sp, -8
     jalr x0, 0(x1)
```

Answer:

```
FACT:addi sp, sp, -8
     sw x1, 4(sp)
     sw x12, 0(sp)
     add x18, x0, x12
     addi x5, x0, 1
     bge x12, x5, L1
     mul x10, x18, x10
     addi sp, sp, 8
     jalr x0, 0(x1)
  L1:addi x12, x12, -1
     jal x1, FACT
     addi x10, x0, 1
     lw x12, 0(sp)
     lw x1, 4(sp)
     addi sp, sp, 8
     mul x10, x12, x10
     jalr x0, 0(x1)
```

Exercise 3

Consider a proposed new instruction named `rpt`. This instruction combines a loop's condition check and counter decrement into a single instruction. For example,

```
rpt x29, loop
```

would do the following:

```
if (x29 > 0) {
   x29=x29-1;
   goto loop;
  }
```

1) If this instruction were to be added to the RISC-V instruction set, what is the most appropriate instruction format?

2) What is the shortest sequence of RISC-V instructions that performs the same operation?

Answer:

1) I think the J format is the most suitable because it is like factorial, using recursive function to keep calling itself.

2)

```
loop: addi x29, x29, -1
      bgt x29, x0, loop
      beq x29, x0, exit
exit: ...
```

Exercise 4

Implement the following C code in RISC-V assembly. Hint: Remember that the stack pointer must remain aligned on a multiple of 16.

```
int fib(int n){
    if (n==0) return 0;
    else if (n==1) return 1;
    else return fib(n-1) + fib(n-2);
   }
```

Answer:

```
   fib: beq x10, x0, Exit
        addi x13, x0, 1
        beq x10, x13, Exit
        addi sp, sp, -8
        sw x1, 0(sp)
        sw x10, 4(sp)
        addi x10, x10, -1
        jal x1, fib
        lw x11, 4(sp)
        sw x10, 4(sp)
        addi x10, x11, -2
        jal x1, fib
        lw x12, 4(sp)
        add x10, x10, x12
        lw x1, 0(sp)
        addi sp, sp, 8
  Exit: jalr x0, 0(x1)
```

Exercise 5

For each function call in above problem, show the contents of the stack after the function call is made. Assume the stack pointer is originally at address `0x7ffffffc`, and follow the register conventions.

Answer:

a. Assume that the variable n is initialized to 0 or 1. Then we do not need to use the stack, the address is always `0x7ffffffc`.

b. Assume that the variable n is initialized to 2. We need to call fib itself once and the stack will grow for 8. the address will be `0x7ffffff4`. After that, it will clear the stack and finally `sp` will be `0x7ffffffc` again.

c. Assume that the variable n is initialized to 3. We need to call fib itself twice and the stack will grow for 16. The deepest address will be `0x7fffffec`. After that, it will clear the stack and finally `sp` will be `0x7ffffffc` again.

d. Assume that the variable n is initialized to n. We need to call fib itself $n-1$ times and the stack will grow for $8(n-1)$. It will copy the process for $n-1$ times.

Exercise 6

Given a 32-bit RISC-V machine instruction:

```
1111 1111 0110 1010 0001 1010 1110 0011
```

1) What does the assembly instruction do?
2) What type of instruction is it?

Answer:

1) `bne x20, x22, Target` and the Target is `PC-12`
2) B-type

Exercise 7

Given RISC-V assembly instruction:

```
lw x21, -32(sp)
```

1) What is the corresponding binary representation?
2) What type of instruction is it?

Answer:

1) `1111 1110 0000 0001 0010 1010 1000 0011`
2) I-type

Exercise 8

If the RISC-V processor is modified to have 128 registers rather than 32 registers:

1) Show the bit fields of an R-type format instruction assuming opcode and funcfields are not changed.

2) What would happen to the I-type instruction if we want to keep the total number of bits for an instruction unchanged?

3) What is the impact on the range of addresses for a beq instruction? Assume all instructions remain 32 bits long and the size of opcode and func fields don't change.

Answer:

1)

funct7	rs2	rs 1	funct3	rd	opcode
7 bits	7 bits	7 bits	3 bits	7 bits	7 bits

2)

immediate[7:0]	rs 1	funct3	rd	opcode
8 bits	7 bits	3 bits	7 bits	7 bits

Since we cannot change the total bits(32). The immediate will be 4 bits shorter and can only have 8 bits.

3) Since rs2 and rs1 are both 2 bits longer. The immediate will be 4 bits shorter(from 12 to 8). As a result, the range of addresses is shorter (from $[-2^{12}, 2^{12}-1]$ to $[-2^8, 2^8-1]$)

Exercise 9

Convert the following assembly code fragment into machine code, assuming the memory location of the first instruction (LOOP) is `0x1000F400`

```
LOOP: blt x0, x5, ELSE
      jal x0, DONE
ELSE: addi x5, x5, -1
      addi x25, x25, 2
      jal x0, LOOP
DONE: ...
```

Answer:

1) `LOOP: blt x0, x5, ELSE`. It is B-type, and the immediate is 4 = `0000 0000 0100`

imm[11]	imm[9:4]	rs2	rs 1	funct3	imm[3:0]	imm[10]	opcode
0	000000	00101	00000	100	0100	0	1100011

Finally, the machine code should be `0000 0000 0101 0000 0100 0100 0110 0011`, or `0x00504463`

2) `jal x0, DONE`. It is J-type, and the immediate is 8 = `0000 0000 0000 0000 1000`

imm[19]	imm[9:0]	imm[10]	imm[18:11]	rd	opcode
0	0000001000	0	00000000	00000	1101111

Finally, the machine code should be `0000 0001 0000 0000 0000 0000 0110 1111`, or `0x0100006F`

3) `ELSE: addi x5, x5, -1`. It is I-type, and the immediate is -1 = `1111 1111 1111`

imm[11:0]	rs1	funct3	rd	opcode
111111111111	00101	000	00101	0010011

Finally, the machine code should be `1111 1111 1111 0010 1000 0010 1001 0011`, or `0xFFF28293`

4) `addi x25, x25, 2`. It is I-type, and the immediate is 2 = `0000 0000 0010`

imm[11:0]	rs1	funct3	rd	opcode
000000000010	11001	000	11001	0010011

Finally, the machine code should be `0000 0000 0010 1100 1000 1100 1001 0011`, or `0x002C8C93`

5) `jal x0, LOOP`. It is J-type, and the immediate is -8 = `1111 1111 1111 1111 1000`

imm[19]	imm[9:0]	imm[10]	imm[18:11]	rd	opcode
1	1111111000	1	11111111	00000	1101111

Finally, the machine code should be `1111 1111 0001 1111 1111 0000 0110 1111`, or `0xFF1FF06F`