JOINT INSTITUTE
交大密西根学院

Ve370 Introduction to Computer Organization

Homework 1

杨毅文 519370910053

Assigned: September 23, 2021

Due: 2:00pm on September 30, 2021

Submit a PDF file on Canvas

1. (5 points) For the following C statement, write the corresponding RISC-V assembly code. Assume that the C variables f, g, and h, have already been placed in registers x5, x6, and x7 respectively. Use a minimal number of RISC-V assembly instructions.

```
f = g + (h − 5);


addi x7, x7, -5
add x5, x6, x7
```

2. (5 points) For the following C statement, write the corresponding RISC-V assembly code. Assume that the variables f, g, h, i, and j are assigned to registers x5, x6, x7, x28, and x29, respectively. Assume that the base address of the arrays A and B are in registers x10 and x11, respectively.

```
B[8] = A[i−j];


sub x5, x28, x29
slli x30, x5, 2
add x30, x30, x10
lw x6, 0(x30)
sw x6, 8(x11)
```

3. (10 points) For the RISC-V assembly instructions below, what is the corresponding C statement? Assume that the variables f, g, h, i, and j are assigned to registers x5, x6, x7, x28, and x29, respectively. Assume that the base address of the arrays A and B are in registers x10 and x11, respectively.

```
slli x30, x5, 2
add  x30, x10, x30
slli x31, x6, 2
add  x31, x11, x31
lw   x5, 0(x30)
addi x12, x30, 4
lw   x30, 0(x12)
```

```
add  x30, x30, x5
sw   x30, 0(x31)


B[g] = A[f] + A[f + 1]
```

4. (5 points) Show how the value `0xabcdef12` would be arranged in memory of a little-endian and a big-endian machine. Assume the data are stored starting at word address 0.

   Little-endian: Store the value from LSB, i.e., the memory be like:

data	12	ef	cd	ab
address	0	4	8	C

   Big-endian: Store the value from MSB, i.e., the memory be like:

data	ab	cd	ef	12
address	0	4	8	C

5. (5 points) Find the shortest sequence of RISC-V instructions that extracts bits 16 down to 11 from register x5 and uses the value of this field to replace bits 31 down to 26 in register x6 without changing the other bits of registers x5 or x6. (Be sure to test your code using x5 = 0 and x6 = `0xffffffff`. Doing so may reveal a common oversight.)

```
addi x7, x7, 0b111111
slli x8, x7, 26
not x8, x8
and x6, x6, x8
slli x7, x7, 11
and x7, x5, x7
slli x7, x7, 15
or x6, x7, x6
```

6. (10 points) Assume x5 holds the value 0x01010000. What is the value of x6 after the following instructions?

```
      bge x5, x0, ELSE
      jal x0, DONE
ELSE: ori x6, x0, 2
DONE: ......
```

   x6 is 2.

7. Consider the following RISC-V loop:

```
LOOP: beq x6, x0, DONE
      addi x6, x6, -1
      addi x5, x5, 2
      jal x0, LOOP
DONE: ......
```

(1) (10 points) Assume that the register x6 is initialized to the value 10. What is the final value in register x5 assuming the x5 is initially zero?

x5 is 20.

(2) (10 points) For the loop above, write the equivalent C code. Assume that the registers x5 and x6 are integers acc and i, respectively.

```
while (i != 0) {
    acc += 2;
    i --;
}
```

(3) (5 points) For the loop written in RISC-V assembly above, assume that the register x6 is initialized to the value N. How many RISC-V instructions are executed?

4*N

(4) (5 points) For the loop written in RISC-V assembly above, replace the instruction "`beq x6, x0, DONE`" with the instruction "`blt x6, x0, DONE`" and write the equivalent C code.

```
while (i >= 0) {
    acc += 2;
    i --;
}
```

8. (20 points) Translate function f into RISC-V assembly language. Assume the function declaration for `g` is `int g(int a, int b)`. The code for function f is as follows:

```
int f(int a, int b, int c, int d){
    return g(g(a,b), c+d);
}
```

```
f:
    addi sp, sp, -4
    sw x1, 0(sp)
    jal x1, g
```

```
add x11, x12, x13
jal x1, g
lw x1, 0(sp)
addi sp, sp, 4
jalr x0, x1, 0
```

9. (10 points) Right before your function `f` from Problem 8 returns, what do we know about contents of registers `x10-x14, x8, x1`, and `sp`? Keep in mind that we know what the entire function `f` looks like, but for function `g` we only know its declaration.

x10-x14 are functional arguments, x10 stores the result of f(a, b, c, d) and x11 contains c + d, x12 and x13 remains the value of c and d, while x14 is not used thus remaining what it was; x8 is saved register and remains what it was; x1 is the return address which points to the address of PC + 4 (the next instruction after calling f()); sp is the stack pointer and goes back to where it was before f() is called.