JOINT INSTITUTE
交大密西根学院

ECE3700J Introduction to Computer Organization

Homework 1

Assigned: September 22, 2022

Due: 2:00pm on September 29, 2022

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 z, x, and y, have already been placed in registers x16, x17, and x18 respectively. Use a minimal number of RISC-V assembly instructions.

```
z = x + (y - 12);
```

Answer:

```
addi x16, x18, -12
add x16, x17, x16
```

2. (5 points) For the following C statement, write the corresponding RISC-V assembly code. Assume that the variables i and j are assigned to registers x5 and x6 respectively. Assume that the base addresses of the arrays A and B are in registers x16 and x17, respectively.

```
B[7] = A[i+j];
```

Answer:

```
add x30, x5, x6 #compute i+j
slli x30, x30, 2  #multiply by 4
add x3, x30, x16
lw x30, 0(x3)     #load A[i+j]
sw x30, 28(x17)   #store in B[7]
```

3. (10 points) For the RISC-V assembly instructions below, what is the corresponding C statement? Assume that the variables f, a, b, c, and d are assigned to registers x5, x6, x7, x28, and x29, respectively. Assume that the base address of the arrays Y and Z 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)
```

Answer:

```
Z[a] = Y[f] + Y[f+1]
```

4. (5 points) Show how the value `0x5678abcd` would be arranged in memory of a little-endian and a big-endian machine. Assume the data are stored starting at word address 0.

Answer:

Big-endian:

Byte offset	0	1	2	3
Data	56	78	ab	cd

Little-endian:

Byte offset	0	1	2	3
Data	cd	ab	78	56

5. (10 points) Find the shortest sequence of RISC-V instructions that extracts bits 20 down to 11 from register x15 and uses the value of this field to replace bits 31 down to 22 in register x16 without changing the other bits of registers x15 or x16. (Be sure to test your code using x15 = 0 and x16 = `0xffffffff`. Doing so may reveal a common oversight.)

Answer:

```
addi x7, x0, 0x3FF// Create 10 bits of mask
slli x7, x7, 11   // Shift the mask bits to [20:11]
and x28, x15, x7  // extract the bits x15[20:11]
slli x28, x28, 11 // Move selected bits from x15[20:11]
                  // into positions 31 to 22
slli x7, x7, 11   // Shift the mask to cover bits 31 to 22
xori x7, x7, -1   // This is a NOT operation
and x16, x16, x7  // "Zero out" positions 31 to 22 of x16
or x16, x16, x28  // Load bits 31 to 22 from x28 into x16
```

6. (5 points) Assume x5 holds the value 0x11010000. What is the value of x6 after the following instructions?

```
        addi x6, x0, 1
        bge x5, x0, ELSE
        jal x0, DONE
ELSE:   ori x6, x0, 2
DONE:   lui x6, 0xFFFFF
```

Answer:

x6 = 0xFFFFF000

7. Consider the following loop in RISC-V assembly:

```
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 5. What is the final value in register x5 assuming the x5 is initially zero?

Answer:

x5 = 10

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

Answer:

```
i = 0;
j = 5;
while (j ! = 0) {
  i += 2;
  j--;
}
```

(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?

Answer:

4*N + 1 instructions are executed.

(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.

Answer:

```
i = 0;
j = 15;
while (j >= 0) {
  i += 2;
  j--;
}
```

8. (20 points) Translate function `f` into RISC-V assembly language following function calling conventions. 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,c), b-d);
}
```

Answer:

```
Assume a, b, c, d are in x10~x13
f:
addi sp, sp, -8     // Allocate stack space for 2 words
sw x1, 0(sp)        // Save return address
sub x5, x11, x13    // x5 = b-d
sw x5, 4(sp)        // Save b-d on the stack
jal x1, g           // Call x10 = g(a,c)
lw x11, 4(sp)       // Reload x11= b-d from the stack
jal x1, g           // Call x10 = g(g(a,c), b-d)
lw x1, 0(sp)        // Restore return address
addi sp, sp, 8      // Restore stack pointer
jalr x0, 0(x1)
```

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.

Answer:

- It's caller's (function f's) responsibility to save argument registers if necessary. Since none of x10-x14 is saved, callee (function g) can change them as it wants. So, we have no idea what the contents of x10-x14 are.

- We don't know what the precise content of sp is; but we do know that it's identical to what it is when f was called.

- x8 is the frame pointer (fp), its use is optional and actually not used. So, we don't know its content either.

- Similarly, we don't know what the precise content of x1 is; but we do know that it is equal to the return address set by the "jal x1, f" instruction that called f before we entered into this f function.