JOINT INSTITUTE
交大密西根学院

Ve370 Introduction to Computer Organization

Homework 1 Solutions

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);
```

Answer:

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

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];
```

Answer:

```
sub x30, x28, x29 #compute i-j
slli x30, x30, 2  #multiply by 4
add x3, x30, x10
lw x30, 0(x3)     #load A[i-j]
sw x30, 32(x11)   #store in B[8]
```

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)
```

Answer:

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.

Answer:

Big-endian:

Byte offset	**0**	**1**	**2**	**3**
Data	**ab**	**cd**	**ef**	**12**

Little-endian:

Byte offset	**0**	**1**	**2**	**3**
Data	**12**	**ef**	**cd**	**ab**

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.)

Answer:

```
addi x7, x0, 0x3f // Create bit mask for bits 16 to 11
slli x7, x7, 11   // Shift the masked bits
and x28, x5, x7   // Apply the mask to x5
slli x7, x6, 15   // Shift the mask to cover bits 31 to 26
xori x7, x7, -1   // This is a NOT operation
and x6, x6, x7    // "Zero out" positions 31 to 26 of x6
slli x28, x28, 15 // Move selection from x5 into positions
                  // 31 to 26
or x6, x6, x28    // Load bits 31 to 26 from x28
```

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:  ......
```

Answer:

x6 = 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?

Answer:

x5 = 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.

Answer:

```
acc = 0;
i = 10;
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?

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:

```
acc = 0;
i = 10;
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);
}
```

Answer:

```
f:
addi x2, x2, -8     // Allocate stack space for 2 words
sw x1, 0(x2)        // Save return address
add x5, x12, x13    // x5 = c+d
sw x5, 4(x2)        // Save c+d on the stack
jal x1, g           // Call x10 = g(a,b)
lw x11, 4(x2)       // Reload x11= c+d from the stack
jal x1, g           // Call x10 = g(g(a,b), c+d)
lw x1, 0(x2)        // Restore return address
addi x2, x2, 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:

- We have no idea what the contents of x10-x14 are, g can set them as it wants.

- We don't know what the precise contents of x8 and sp are; but we do know that they are identical to the contents when f was called.

- Similarly, we don't know what the precise contents of x1 are; but, we do know that it is equal to the return address set by the "jal x1, f" instruction that invoked f.