ECE M116C- CS M151B Computer Architecture Systems - UCLA

Homework 1 Deadline: Tuesday, Oct. 5, 4 PM

Fall 2021 (Upload it to Gradescope.)

Q1. Write the Swap function -- `Swap(x,y)` -- in the RISC-V assembly. Assume that the inputs are loaded into `a0` and `a1`.

Q1 Answer

```
swap:
        addi t0, a0, 0
        addi a0, a1, 0
        addi a1, t0, 0
        ret
```

or alternatively…

```
swap:
        add t0, a0, zero
        add a0, a1, zero
        add a1, t0, zero
        ret
```

Q2.

a) What is the final value in `a0` after executing this function?
(Assume that `MUL` instruction multiplies `a0` and `t1` and stores the result in `a0`.)

```
main:
       li  a0, 4
       jal ra, func
       ret
func:
       addi sp, sp, -16
       sw   ra, 8(sp)
       sw   a0, 0(sp)
       addi t0, a0, -1
       bge  t0, zero, f2

       addi a0, zero, 1
       addi sp, sp, 16
       ret               // Pay attention to this!

f2:
       addi a0, a0, -1
       jal  ra, func     // … and this!
       addi t1, a0, 0
       lw   a0, 0(sp)
       lw   ra, 8(sp)
       addi sp, sp, 16
       mul a0, a0, t1
       ret
```

b) What does this function do? (e.g., what would be the final value if the initial value of `a0` was 7 instead of 4 in line 1)

Q2 Answer

a)

a0 =24

Steps: a0=4. Then, we store ra (PC of line 3, "ret") on the stack, then a0 (4) on the stack. Then, we subtract 1 from a0 to get a0=3, then we subtract 16 on the stack pointer and restart the process. So a0=3, store ra (PC of line 14, "addi, t1, a0, 0") on the stack, then a0 (3) on the stack. Then a0=2, store ra (PC of line 14) on the stack, then a0 (2) on the stack. This continues until a0=0. Since a0 – 1 < 0, we run lines 9-11. We set a0=1 and

the stack pointer is incremented by 16 (points to portion of memory holding the values 1 and ra).

Then, we set t1 to a0 + 0, which is 1+0=1. Next, we load the portion of memory we are pointing to (1 and ra) into a0 and ra, respectively. We increment the stack pointer by 16 again (points to 2 and ra in memory), and we do a0=a0*t1, which is 1*1 = 1. This process continues (multiply the new a0 by the previous a0). So the next step would be: t1 = 1, a0 = 2, so a0 = 2*1 = 2. Then t1 = 2, a0 = 3, so a0 = 3*2 = 6. Then t1 = 6, a0 = 4, so a0 = 4*6 = 24. Once we get to the bottom of the stack (ra pointing to the 3rd line, "ret"), we finish.

b)

This function multiplies each integer value from 1, up to and including the initial value of a0, which was 4. So a0 = 1 * 2 * 3 * 4 = 24. If a0's initial value was 7, the final value would be 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040.

Q3. Write the RISC-V (RV32I) assembly for the fib(n) function, which computes the Fibonacci number for input n.

See the C code below:

```
int fib(int n)
{
     int a = 0;
     int b = 1;
     int c = a + b;
     while (n > 1) {
     c = a + b;
     a = b;
     b = c;
     n--;
     }
     return c;
}
```

Use the following information in writing your assembly code.

1. The function starts at memory location `0x1100`. Each instruction is 32 bits, thus the second instruction should start at `0x1104` and the third at `0x1108`, and so on. Use this information to correctly compute the offset for jump and branch instructions (you are not allowed to use labels).
2. Value *n* is passed (stored) in register `a0`.
3. The return value, c, should be stored in `a0`.
4. The return address is stored in `ra`.
5. You are free to use saved and temporary registers (don't forget to save values if you are using saved registers).
6. You are allowed to use PSEDUO instructions (e.g., `ret`, `call`, etc.)

Q3 Answer

```
0x1100    li t0, 0
0x1104    li t1, 1
0x1108    add t2, t0, t1
0x110C    li t3, 1
0x1110    ble a0, t3, 24
0x1114    add t2, t0, t1
0x1118    addi t0, t1, 0
0x111C    addi t1, t2, 0
0x1120    addi a0, a0, -1
0x1124    jal zero, -20      (note: "j -20" also ok)
0x1128    addi a0, t2, 0
0x112C    ret
```

Note: "bgt", if implemented correctly, could also work, but results in a more complex instruction set.