ECE3700JFA23 RC2

TA: Xu Weiqing

Q&A about RISC-V instructions

Review about zero/unsigned…

1. What is immediate? What's it used to do?
2. When immediate data signed / unsigned?
3. For shift right and load byte (or halfword), when zero(unsigned) extension / signed extension?
4. For conditional branch, when the reg data signed / unsigned?

Jump

1. jal

   ```
   jal rd, Label
   ```

   do:

   rd ← PC + 4 (rd ← we use x1)

   PC ← PC + Imm<<1 (Imm ← we use Function Label)

   Before jumping to the function:

   ```
   jal x1, FunctionLabel
   ```

2. jalr

```
jalr rd, offset(rs1)
```

do:

rd ← PC + 4 (we use x0)

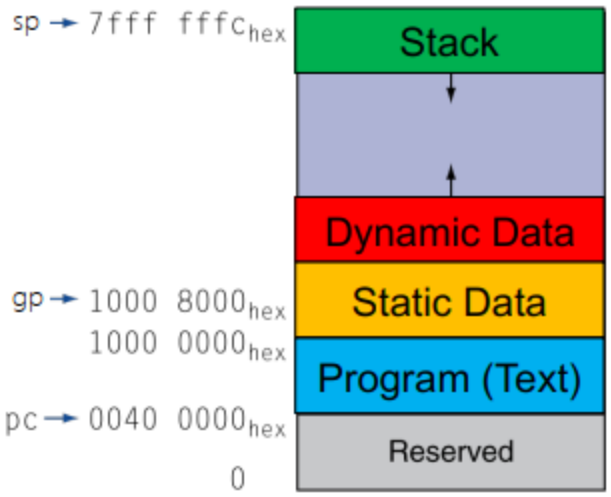
PC ← rs1 + Imm (rs1 ← we use x1, Imm ← offset ← we use 0)

Before leaving the function:

```
jalr x0, 0(x1)
```

Memory

1. PC (program counter)
    a. store the addr. of the instruction to be executed. (like pointer)
    b. PC ← PC + 4
2. Memory layout
    a. stack **x2**: sp 0x7ffffffc going down
    b. dynamic data: heap, going up
    c. static data: global/static variables **x3**: gp 0x10008000
    d. text: program (instructions) **PC**: 0x0040000



Function Call

1. parameters → reg x10-x17
2. control → function
3. storage & stack acquire

   save 3 registers:

   ```
   addi sp, sp, -12
   ```

4. save (push) important reg → stack
5. operating…
6. result → reg x10-x11
7. load (pop) stack → important reg
8. return storage in stack
9. return to the place of function call → reg x1

Caller VS Callee

Caller	Callee
save the registers needed after function call Including: its arguments + temporary registers	save **saved registers** to stack before used; don't need to save temporary registers
save return address	require stack
jal	jalr

- x10 – x11 : function arguments/results
- x12 – x17 : function arguments
- x5 — x7 , x28 — x31 : temporary registers
- x8 — x9 , x18 — x27 : saved registers

1. Make sure the saved registers don't change after function call!
2. frame (activation record): what's saved in this function

Leaf VS Non-Leaf

Leaf	Non-Leaf
Do not call other functions	Nested functions
	x1 change because of calling other functions

Exercise 1

Do it!

C:

```
int add(int *a, int size) {
    //REQUIRES: size is positive integers
    int result = 0;
    for (int i = 0; i < size; i++) {
        result = result + a[i];
    }
    return result;
}
```

Assume two arguments a and size are stored in x11 and x12 respectively, and the returned result should be stored in x10.

Assembly:

```
ADD:
  addi x10, x0, 0
  add x5, x0, x11
  addi x6, x0, 0 # i
LOOP:
  lw x7, 0(x5) # a[i]
  add x10 x10 x7 # result = result + a[i]
  addi x6 x6 1 # i += 1
  addi x5 x5 4 # a[i] -> a[i+1]
  bne x6 x12 LOOP # i < size -> LOOP
  jalr x0 0(x1) # return after add function
```

Exercise 2

C:

```
int fact (int n) {
    //REQUIRES: n is a positive integer
    if (n < 3) return n;
    else return n * fact(n-1);
}
```

Assume the argument n is in x10, and the return result should be in x10.

```
fact:
  addi sp, sp, -8
  sw x1, 4(sp)
  sw x10, 0(sp)
  addi x5, x10, -3 # x5 <- n-3
  bge x5, x0, L1
  addi sp, sp, 8
  jalr x0, 0(x1)
L1:
  addi x10, x10, -1
  jal x1, fact
  addi x6, x10, 0
  lw x10, 0(sp)
  lw x1, 4(sp)
```

```
  addi sp, sp, 8
  mul x10, x10, x6
  jalr x0, 0(x1)
```

Exercise 3

C:

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

Assume the function declaration for g is **int g(int a, int b)**, arguments a, b, c, d are in x10-x13. The return value of f or g should be in x10.

```
addi sp, sp, -24
sw x1 20(sp)
sw x10 16(sp)
sw x11 12(sp)
sw x12 8(sp)
sw x13 4(sp)
add x5 x11 x13 # x5 = b+d
add x11 x12 x0 # x11 = c
sw x5 0(sp)
jal x1 g # x10 = g(a,c)
lw x5 0(sp)
add x11 x5 x0
jal x1 g # x10 = g(g(a,c), b+d)
add x5 x10 x0
lw x13 4(sp)
lw x12 8(sp)
lw x11 12(sp)
lw x10 16(sp)
sub x10 x11 x5
lw x1 20(sp)
addi sp, sp, 24
jalr x0 0(x1)
```

References

ECE3700JFA23 Slides T2