

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:

$rd \leftarrow PC + 4$ ($rd \leftarrow$ we use x1)

$PC \leftarrow PC + Imm \ll 1$ ($Imm \leftarrow$ we use Function Label)

Before jumping to the function:

```
jal x1, FunctionLabel
```

2. jalr

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

do:

$rd \leftarrow PC + 4$ (we use x0)

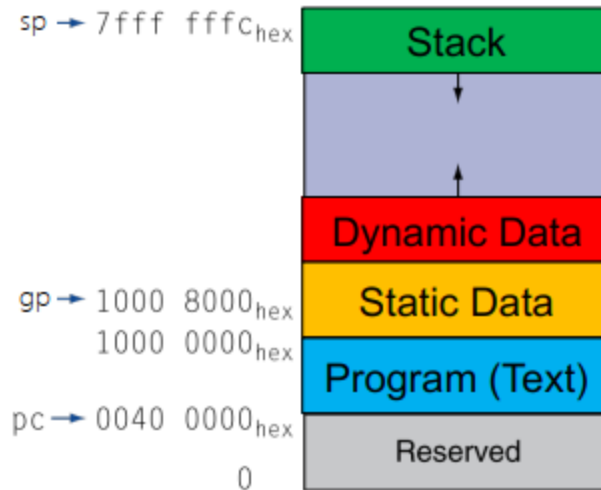
$PC \leftarrow rs1 + Imm$ (rs1 \leftarrow we use x1, Imm \leftarrow offset \leftarrow we use 0)

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. $PC \leftarrow PC + 4$
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

Caller VS Callee

| | |
|-----------------------------|--|
| 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:

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`.

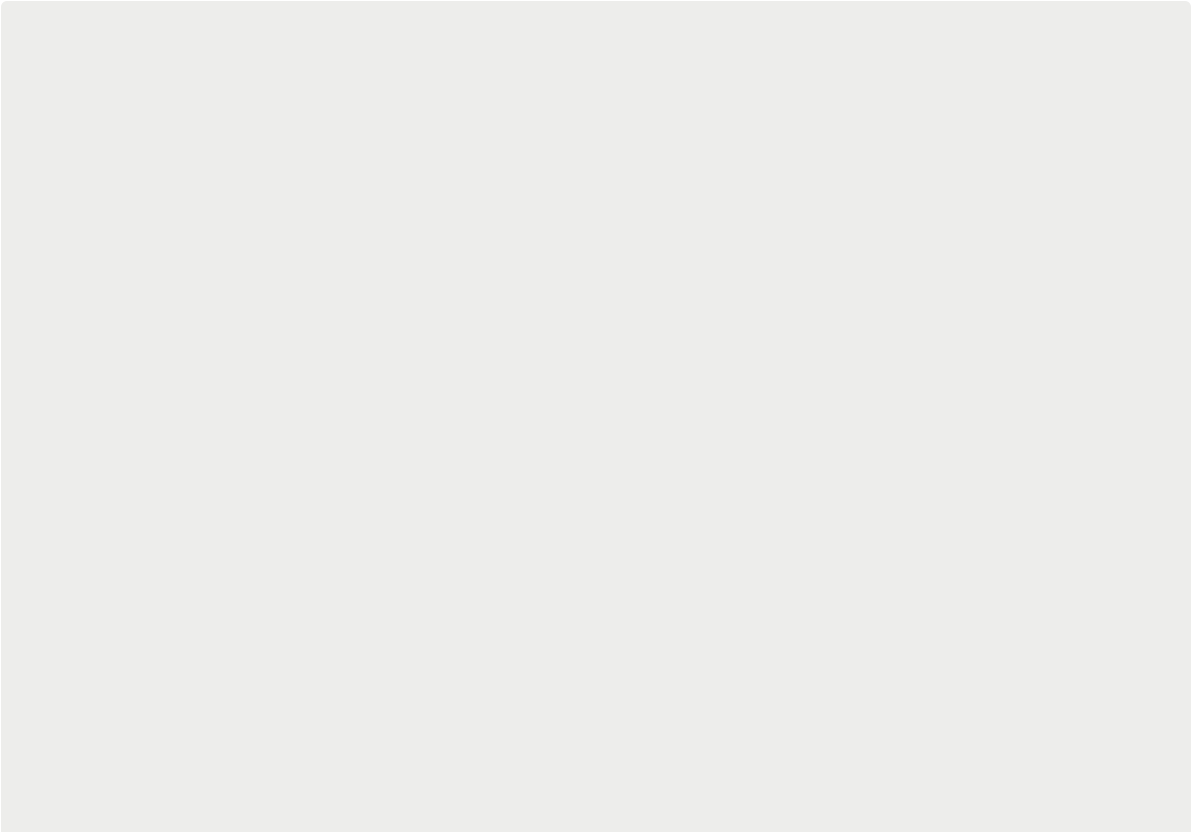


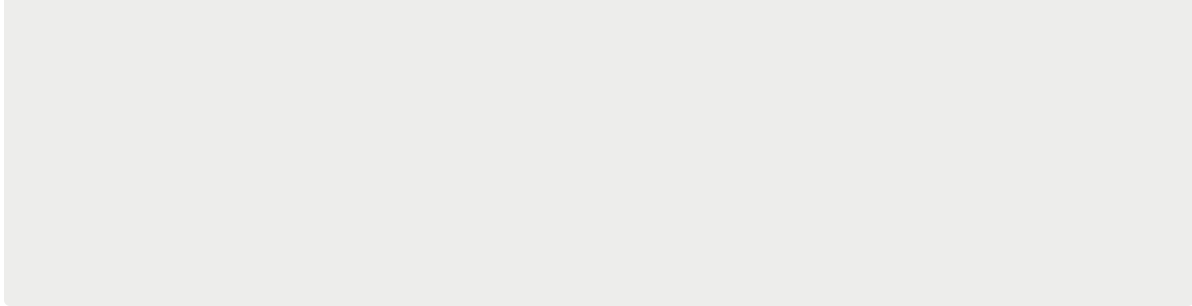
Exercise 3

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.





References

ECE3700JFA23 Slides T3