Jump Instructions

❓ Distinguish between j, jr, jal, jalr

1. 首先，我们先区分一下JAL和JALR

   JAL: **jal rd offset**

   JALR: **jalr rd rs offset**

jal	UJ	Jump & Link	R[rd] = PC+4; PC = PC + {imm,1b'0}
jalr	I	Jump & Link Register	R[rd] = PC+4; PC = R[rs1]+imm

我们发现，这两个指令只差了一个寄存器$rs，JAL跳转的位置是**PC+offset**，JALR跳转的位置是**R[$rs]+offset**，两个指令都会将返回的地址连接到rd（默认是$ra）。下面这两段代码是等价的，左边黄色的下划线venus是在建议用label代替offset。

```
1 mv x1, x0
2 addi t1,t1,12
3 jal ra, 4
4 main:
5     addi a0, a0,17
6     mv a1,t1
7     ecall
```

```
1 mv x1, x0
2 addi t1,t1,12
3 jalr ra, t1, 0
4 main:
5     addi a0, a0,17
6     mv a1,t1
7     ecall
```

2. 然后我们会想，这么简单的程序我要ra做什么，或者说我都搞寄存器了还加offset做什么，于是就有了J和JR这两个伪指令。

j	Jump	PC = {imm,1b'0}	jal
jr	Jump register	PC = R[rs1]	jalr

J: **j offset = jal x0 offset**

JR: **jr rs = jalr x0 rs 0**

于是上面的代码就可以写成

```
1 mv x1, x0
2 addi t1,t1,12
3 j main
4 main:
5     addi a0, a0,17
6     mv a1,t1
7     ecall
```

```
1 mv x1, x0
2 addi t1,t1,12
3 jr t1
4 main:
5     addi a0, a0,17
6     mv a1,t1
7     ecall
```

可以看到他们在assemble的时候是这个样子的

```
0x8        0x0040006F          jal x0 4                    j main
```

```
0x8        0x00030067          jalr x0 x6 0                jr t1
```

3. 那么在写代码的时候，这四个指令该怎么用呢？我个人习惯是
    - 如果是不用回头的那种跳转，直接J就完事，比如说 j exit
    - 如果是调用函数，就 jal function, 函数结束的时候再jr ra
    - jalr狗都不用（不是

差不多了，有问题欢迎讨论，我学学就来