The procedure of how the CPU's hardware processes an instruction
can be streamlined into a small number of stages.
Let's look at an example.

Note -- each step of an instruction should take 1 cycle.

```
loop:
        lw $t2, 0($10)              # inst 1
        add $12, $12, $11           # inst 2
        addi $10, $10, 4            # inst 3
        bne $10, $13, loop          # inst 4
```

We can write a time diagram to see what happens during each cycle

cycle	fetch	decode	execute	mem or result	result	comments
1	1					
2		1				
3			1			address calculation
4				1		load
5					1	
6	2					
7		2				
8			2			add
9				2		update register
10	3					
11		3				sign extend constant
12			3			add
13				3		update register
14	4					
15		4				calculate target
16			4			compare
17	1					loop repeats
18		1				
19			1			
20				1		
21					1	

etc.

Notice we could save time if we could do multiple steps simultaneously.
For example, why do we have to wait for instruction 1 to finish before fetching instruction 2?
This is the motivation for **pipelining.**