P264-268

Y86-64 的顺序实现

- 通常，处理一条指令包括很多操作，简略分成如下阶段：
  - 取指。即从内存读取指令字节
  - 译码。从寄存器读入最多两个操作数
  - 执行。该阶段要么执行指令指明的操作，计算内存引用的有效地址，要么增加或减少栈指针。
  - 访存。该阶段将数据写入内存，或从内存读出数据
  - 写回。
  - 更新 pc。将 pc 设置成下一跳指令地址

  由此 无限循环。

- 从硬件角度，既然执行指令是一个无线循环的操作，流程相似，自然是希望硬件数量最少，降低复杂度。其中一个方法是：**让不同指令共享尽可能多的硬件，让框架更加通用。**比如下述几个指令：

阶段	OPq rA, rB	rrmovq rA, rB	irmovqV, rB
取指	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valP ← PC+2	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valP ← PC+2	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valC ← M_8[PC+2] valP ← PC+10
译码	valA ← R[rA] valB ← R[rB]	valA ← R[rA]	
执行	valE ← valB OP valA Set CC	valE ← 0+valA	valE ← 0+valC
访存			
写回	R[rB]← valE	R[rB]← valE	R[rB]← valE
更新 PC	PC ← valP	PC ← valP	PC ← valP

4-18 Y86-64 指令 OPq、rrmovq 和 irmovq 在顺序实现中的计算。这些指令计算了一个值，并将结果存放在寄存器中。符号 icode:ifun 表明指令字节的两个组成部分，而 rA:rB 表明寄存器指示符字节的两个组成部分。符号 M_1[x]表示访问(读或者写)内存位置 x 处的一个字节，而 M_8[x]表示访问八个字节