阶段	通用	具体
	ret	ret
取指	icode:ifun ← M_1[PC] valP ← PC+1	icode:ifun ← M_1[0x041]=9:0 valP ← 0x041+1=0x042
译码	valA ← R[%rsp] valB ← R[%rsp]	valA ← R[%rsp]=120 valB ← R[%rsp]=120
执行	valE ← valB+8	valE ← 120+8=128
访存	valM ← M_8[valA]	valM ← M_8[120]=0x040
写回	R[%rsp]← valE	R[%rsp]← 128
更新 PC	PC ← valM	PC ← 0x040

跟踪记录表明这条指令的效果就是将 PC 设为 0x040，halt 指令的地址。同时也将%rsp置为了 128。

4.3.2　SEQ 硬件结构

实现所有 Y86-64 指令所需要的计算可以被组织成 6 个基本阶段：取指、译码、执行、访存、写回和更新 PC。图 4-22 给出了一个能执行这些计算的硬件结构的抽象表示。程序计数器放在寄存器中，在图中左下角(标明为“PC”)。然后，信息沿着线流动(多条线组合在一起就用宽一点的灰线来表示)，先向上，再向右。同各个阶段相关的硬件单元(hardware units)负责执行这些处理。在右边，反馈线路向下，包括要写到寄存器文件的更新值，以及更新的程序计数器值。正如在 4.3.3 节中讨论的那样，在 SEQ 中，所有硬件单元的处理都在一个时钟周期内完成。这张图省略了一些小的组合逻辑块，还省略了所有用来操作各个硬件单元以及将相应的值路由到这些单元的控制逻辑。稍后会补充这些细节。我们从下往上画处理器和流程的方法似乎有点奇怪。在开始设计流水线化的处理器时，我们会解释这么画的原因。

硬件单元与各个处理阶段相关联：

取指：将程序计数器寄存器作为地址，指令内存读取指令的字节。PC 增加器(PC incrementer)计算 valP，即增加了的程序计数器。

译码：寄存器文件有两个读端口 A 和 B，从这两个端口同时读寄存器值 valA 和 valB。

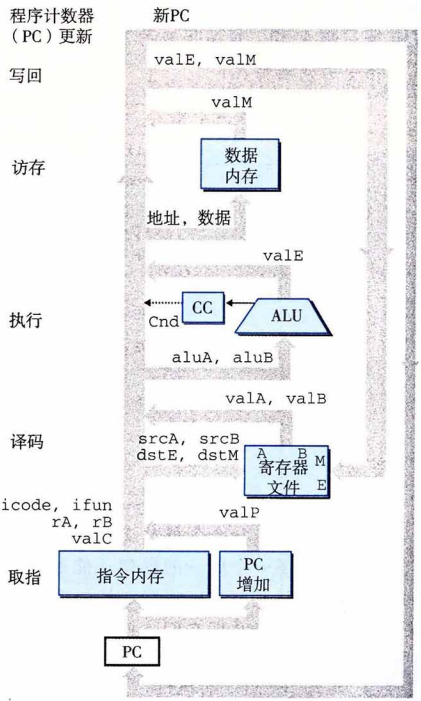

图 4-22　SEQ 的抽象视图，一种顺序实现。指令执行过程中的信息处理沿着顺时针方向的流程进行，从用程序计数器(PC)取指令开始，如图中左下角所示