第六章 指令执行实例

time：2025.10

王冬青

第四周--课后思考：第六章 指令执行实例（后半节课）

1、 初步理解机器指令（01 串）的封装分为**操作码**和**操作数**（也叫*地址码*）两大部分，一般操作码决定指令功能，操作数决定指令执行对象位置，操作数又分为**源操作数**和**目的操作数**；

2、 什么是低级语言？理解汇编语言中的指令是机器指令的助记方式，以本章中 6 条指令为例，操作码分别对应 load、store、add、jump 等指令。

3、 什么是源操作数和目的操作数？本章中 6 条指令用到的数据位置在哪里？它们对应下一章学习的数据寻址方式。

4、 目前我们将每条指令的执行粗略分为：取指、译码、执行；

5、 将本章中 6 条指令的执行过程试着自己不看图解，解释一遍；体会 PC 寄存器、通用寄存器组、运算器、译码器、IR、MAR、MDR，以及内存等在指令执行过程中的作用，控制器的工作原理在此时不用纠结，会在 8、9 章学习；

6、 通过前两条 load 指令的执行过程，理解指令执行复杂度不仅与指令功能（操作码）有关，还与指令操作对象的获取方式（地址码）有关。

7、 本章中 PC 的更新发生在什么时候？PC 每次为什么是+4？

8、 这 6 条示例指令有哪些用到了加法器？分别用加法器做什么用？最后一条 Jump 指令用到加法器了吗？有同学问 jump 之后还需要对 1000 加 4 吗？这里不用再+4 了，按照指令后面的注释理解就可以了，jump 后，会到 1000 地址处去取指，本例中没给出该地址指令，所以就不接着讨论了，该例中给出 jump 指令只是让读者看到非顺序执行的场景。

9、 这 6 条指令执行过程分别访问了几次内存？两次 load 和两次 store 访问的内存次数一样吗？

10、 思考一条指令的执行，哪些操作是在 CPU 内部做的？哪些操作是需要访问存储器的？哪些操作是需要访问 IO 的？

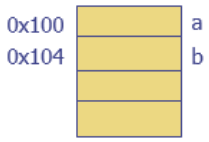
说明；注释部分[Rx]表示取 Rx 的值，这种表述方式对初学者不太友好，容易将其和**寄存**

器间接寻址混淆，所以统一为 **Rx 表示寄存器直接寻址**，即取 Rx 的值做为操作数，而**[Rx]是寄存器间接寻址**，即到内存地址为 Rx 的地方进行取数。

尝试：

试着把下面一段代码用本章掌握的汇编指令（本模型机的）来执行：

```
int a = 1;
int b = 2;
a = a + b;
```



第六章作业：6.1、6.2（提交时间待定）