硬件编程

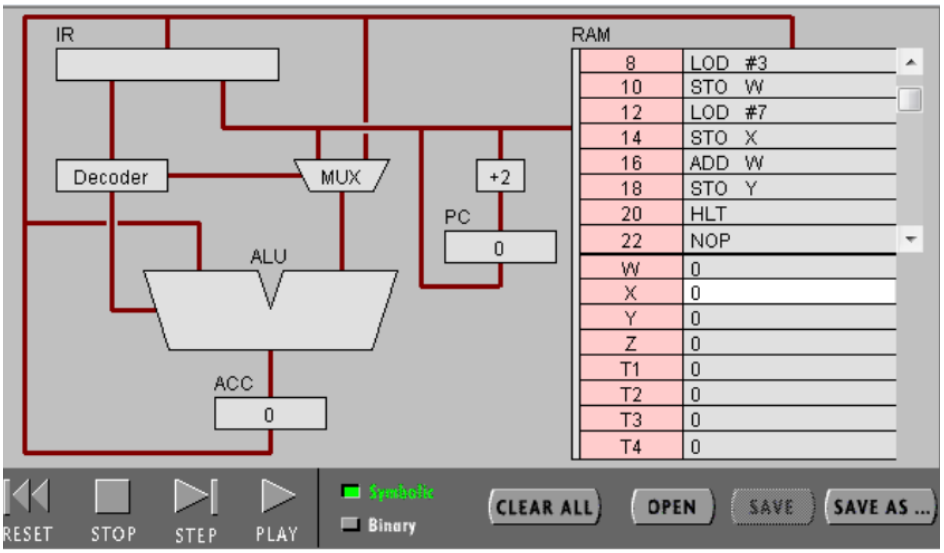
机器指令编程

殷睿

软件工程四班 | 18342122

目录

Program One



STEP BY STEP

PC，IR 寄存器的作用

PC：**Program counter（程序计数器）**，是 CPU 中用于存放下一条指令所在单元的地址的寄存器。在每次获取指令时，需要根据 PC 中存放的指令的地址获取，每个指令被获取后，存储地址加一，指向下一条指令。

IR：**Instruction register（指令寄存器）**，是临时放置从内存里面取得的指令的寄存器。在每次从内存中读取指令后，这条开始执行的指令将被存放在 IR 中进行操作。

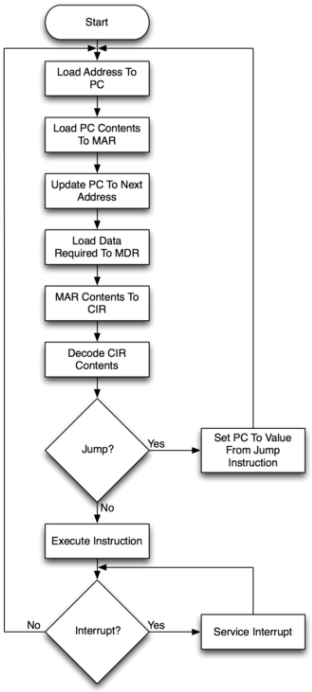
ACC 寄存器的全称与作用

ACC：**Accumulator（累加器）**，是 CPU 中用于储存算术逻辑单元（ALU）计算产生的中间结果的寄存器。ACC 提供了一个 CPU 临时储存计算结果的地方，这样计算结果就不必先储存到内存，之后再调用，节约 CPU 的运行时间。

Fetch-Execute 周期

指令周期:

1. 取得指令：CPU 根据 PC 中的地址读取 RAM 中的指令，PC 地址加一，读取的指令经总线存放到 IR 中。
2. 解码指令：解码器（Decoder）对 IR 中的指令进行解码，将其转换为机器语言（Machine Language）。
3. 执行指令：依据指令调用 ALU 或 ACC 等组件执行。
4. 储存结果：内存或者 ACC 储存相应的计算结果。



LOD #3:

1. 由 PC 中的地址，取得 RAM 中的“LOD #3”指令，PC 地址加一指向下一条，“LOD #3”被储存到 IR 中。
2. “LOD”解码后调用 ALU，“#3”经过数据选择器（MUX）后载入 ALU 等待计算。
3. ALU 进行运算，得到结果 3。
4. ALU 将结果储存到 ACC 当中。

ADD W:

1. 由 PC 中的地址，取得 RAM 中的“ADD W”指令，PC 地址指向下一条，IR 储存“ADD W”指令。
2. 解码器对“ADD”进行解码，编译为机器语言。“W”进入数据选择器。
3. 解码后，CPU 调用 ALU 中的加法器，读取加载 ACC 中储存的数据（3）。数据选择器读取内存中“W”的数据（7），数据经总线加载入 ALU，ALU 对 3 和 7 进行加法运算，获得结果 10。
4. ACC 储存结果 10。

有什么不同?

1. 解码器对“ADD”进行解码后，CPU 对 MUX 进行了调整，是的“W”进入后，转为对内存中数据的调用。而“#3”则是直接将 3 载入到 ALU 中。
2. “LOD”指令并没有调用 ALU 中的加法器，而是将 3 直接储存到 ACC 中。

BINARY

“LOD #7” 的二进制

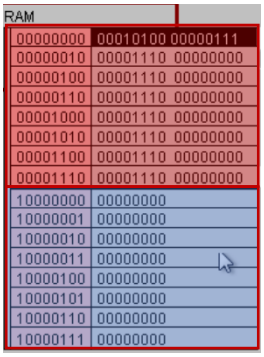
00010100 00000111

0001：立即寻址，对立即数进行操作

0100：在加载一个数，并储存在 ACC 中

00000111：二进制的 7，一个立即数

RAM 的地址



地址开头为 0：单元储存指令
地址开头为 1：单元储存数据

该 CPU 位数

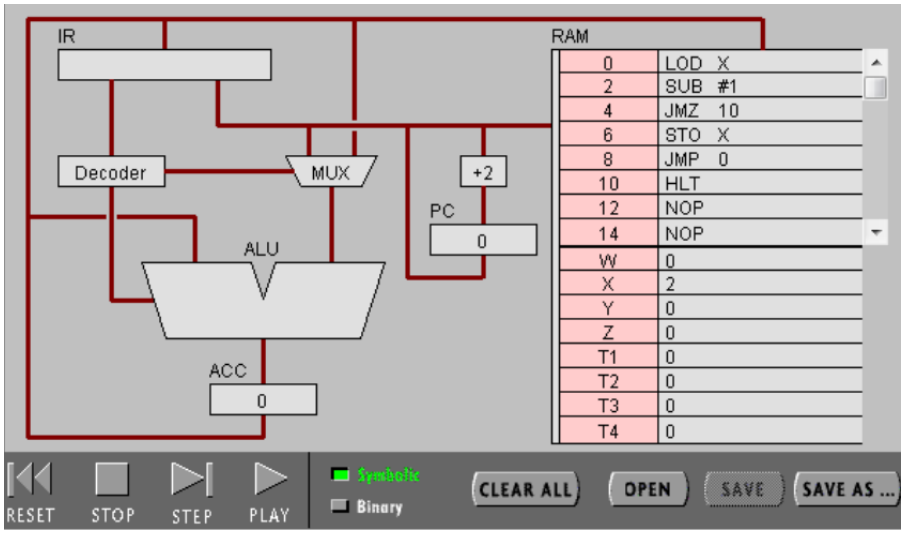
累加器为 8 位，故 CPU 位数为 8 位。

C 语言表达

```c
#include <stdio.h>

int main(){
    int w = 3;
    int x = 7;
    int y = w + x;
}
```

Program Two



原程序

程序的功能

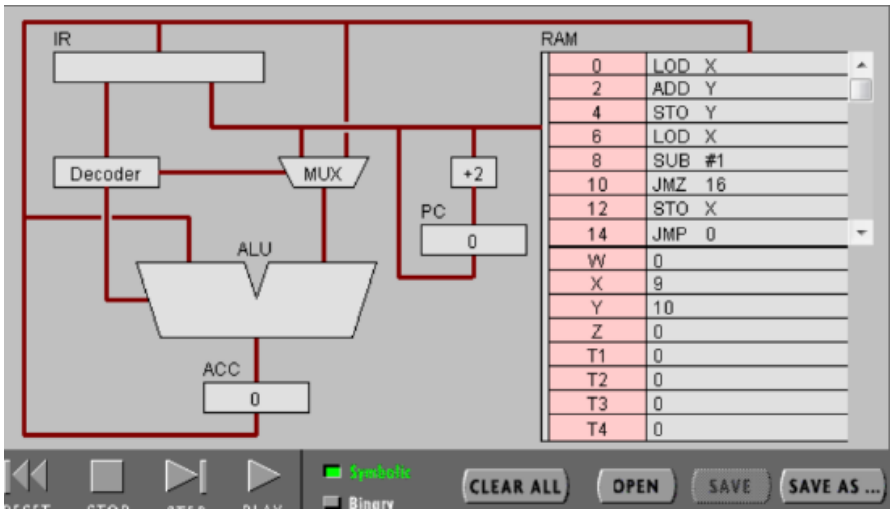
一个执行 3 次的循环。

对应的 c 语言程序

```c
#include <stdio.h>

int main(){
    int x = 3;
    while(x != 0)
        x--;
}
```

修改该程序



c 语言表达

```c
#include <stdio.h>

int main(){
    int x = 10;
    int y = 0;
    while(x != 0){
        y += x;
        x--;
    }
}
```

机器语言表达

00000100 10000001
00000000 10000010
00000101 10000010
00000100 10000001
00010001 00000001
00001101 00010000
00000101 10000001

00001100 00000000
00001111 00000000

高级语言与机器语言的区别与联系

机器语言更加贴近计算机，以计算机和处理器的结构为主要思维模式进行编程，对计算机的 CPU 和内存等结构进行直接指令调用。

高级语言更加贴近人，以人的思维模式进行编程，也因此具有结构化的特点，程序可读性较强，但是不会对计算机的硬件进行直接调用，而是交给计算机自动进行。

但是两者编程思想上可以找到一脉相承的地方，例如上边例子中的循环思想。并且高级语言和机器语言一样，采用指令的形式编程，例如 c 语言中一个带有分号的语句，就可近似看作是一条指令。