软件工程导论

实验

机器语言编程

喻勇强
18342123

目录

任务 1

（1）点 step after step 观察并回答问题
（2）点击"Binary"，观察并回答问题

任务 2

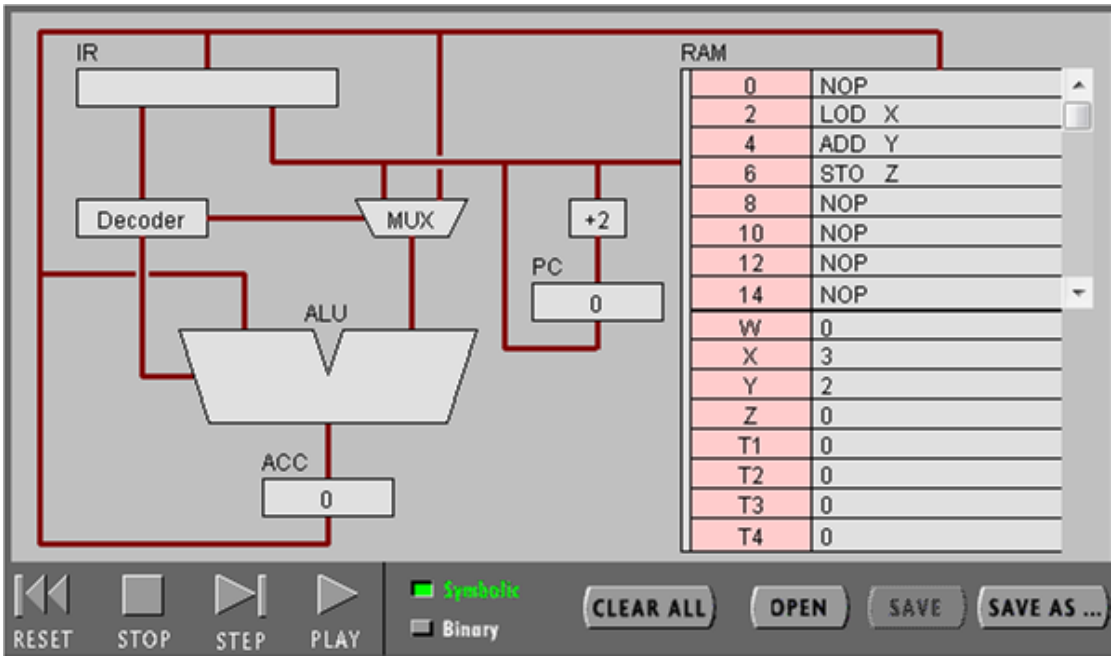
（1） 输入程序 Program2 运行并回答问题
（2） 修改程序，用机器语言实现 10+9+8+…1，输出结果存放于内存 Y

实验目标:

- 理解冯·诺伊曼计算机的结构
- 理解机器指令的构成
- 理解机器指令执行周期
- 用汇编编写简单程序

任务 1:

两个数的加法:



观察并回答下面问题:

1. PC，IR 寄存器的作用。
2. ACC 寄存器的全称与作用。
3. 用“LOD #3”指令的执行过程，解释 Fetch-Execute 周期。
4. 用“ADD W”指令的执行过程，解释 Fetch-Execute 周期。
5. “LOD #3”与“ADD W”指令的执行在 Fetch-Execute 周期级别，有什么不同。

1. IR 指令寄存器是用来存放指令的，存放当前正在执行的指令，包括指令的操作码，地址码，地址信息

   PC 程序计数器，是用来计数的，指示指令在存储器的存放位置，也就是个地址信息

2. ACC (Accumulator)也就是累加器，这是一个具有特殊用途的二进制 8 位寄存器，专门用来存放操作数和运算结果。

3.

PC 中的地址码就是该指令的地址码，计算机按照这个地址去 RAM 中寻找相应的指令这指令运送到了 IR 中。

指令从 IR 中被转移到了 Decoder 中，对指令进行解码过程，判断该指令要进行何种操作，并且判断该指令中用到的操作数在哪里。

又 LOD #3 中的操作数本身就在这条指令中，所以不再需要额外从内存中读取。

最后通过数据选择器存放到了 ACC 中，完成了 Execute the instruction 的操作

4. PC 根据指令的地址码去 RAM 中读取出这一条指令，并且传送到 IR 中

指令从 IR 被送到了 Decoder 中, 对指令进行解码, 发现这条指令中的操作数并不在指令中, 而是需要到内存中重新读取。

根据上一步的结果, W 的值需要到内存中再读取, 并且同样通过数据选择器进入到 ALU 中。

最后, 在 ALU 中完成了算术运算后结果被保存在了 ACC 中, 完成了 Execute the instruction。

4. 相比于“LOD #3”计算机在读取”ADD W”的指令的过程中，只能读到 w 的地址码，并不能得到具体的数值。

点击“Binary”,观察回答下面问题

1.写出指令“LOD #7”的二进制形式，按指令结构，解释每部分的含义。
2.解释 RAM 的地址。
3.该机器 CPU 是几位的？（按累加器的位数）
4.写出该程序对应的 C 语言表达。

1. 00010100 00000111 第一个字节中的四位数代表寻址模式，前四位中的“1”表示操作数是数值后四位代表操作码 0100 表示 LOD 操作，第二个字节因而表示的是数 7.
2. RAM 中所有存储的数据和指令都有自己的地址，计算机可以根据地址来读取指令和数据。
3. 8 位
4. C 语言表达

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

任务 2:
（1） 输入程序 Program 2，运行并回答问题:
1. 用一句话总结程序的功能
2. 写出对应的 c 语言程序

1. 通过循环从 1 加到 n,并输出结果
2. Int main（）{

```
    int sum=0;
    int i;
    for(i=0;i<n;i++)
        sum=sum+i;
     printf(“%d”,sum);
}
```

（2） 修改该程序，用机器语言实现 10+9+8+..1 ，输出结果存放于内存 Y
1. 写出 c 语言的计算过程
2. 写出机器语言的计算过程
3. 用自己的语言，简单总结高级语言与机器语言的区别与联系。

1. int main(){

```
    int sum=0;
    int i;
    for(i=10,i>0,i--)
        sum=sum+i;
    printf(“%d”,sum);
}
```

2. LOD #0

```
  STO Y
  LOD #1
  STO X
  JMP 16
  LOD X
```

ADD #1
STO X
SUB W
JMZ 28
LOD Y
ADD X
STO Y
JMP 10
HLT

3.机器语言是根据计算机 CPU 结构而使计算机能够执行的语言，高级语言的使用模式符合人们的语言习惯，要使计算机运行程序必须把高级语言转换成机器语言。