

硬件编程-机器指令编程

学号：18342015

姓名：邓智平

目录

任务 1：简单程序.....	3
(1) 打开网页 The PIPPIN User's Guide ， 然后输入 Program 1: Add 2 number.....	3
(2) 点 step after step。观察并回答下面问题：	4
(3) 点击“Binary”,观察回答下面问题.....	5
任务 2：简单循环.....	8
(1) 输入程序 Program 2， 运行并回答问题：	8
(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.....	10
实验小结	13

任务 1：简单程序

(1) 打开网页 The PIPPIN User's Guide ，然后输入
Program 1: Add 2 number

win7-64 [正在运行] - Oracle VM VirtualBox

管理 控制 视图 热键 设备 帮助

CPU Simulator Applet - Windows Internet Explorer

http://www.science.smith.edu/~jcardell/Courses, Bing

收藏夹 建议网站 网页快讯

CPU Simulator Applet

The screenshot shows a CPU simulator interface. On the left, a block diagram of the CPU architecture is displayed, including an Instruction Register (IR), a Decoder, a Multiplexer (MUX), an Arithmetic Logic Unit (ALU), an Accumulator (ACC), and a Program Counter (PC). The PC is currently at address 0. On the right, a table shows the contents of RAM, with addresses 0 through 14 and their corresponding instructions. Below the table, registers W, X, Y, Z, T1, T2, T3, and T4 are listed, all containing the value 0. At the bottom, there are control buttons for RESET, STOP, STEP, and PLAY, along with a 'Binary' checkbox and buttons for CLEAR ALL, OPEN, SAVE, and SAVE AS ...

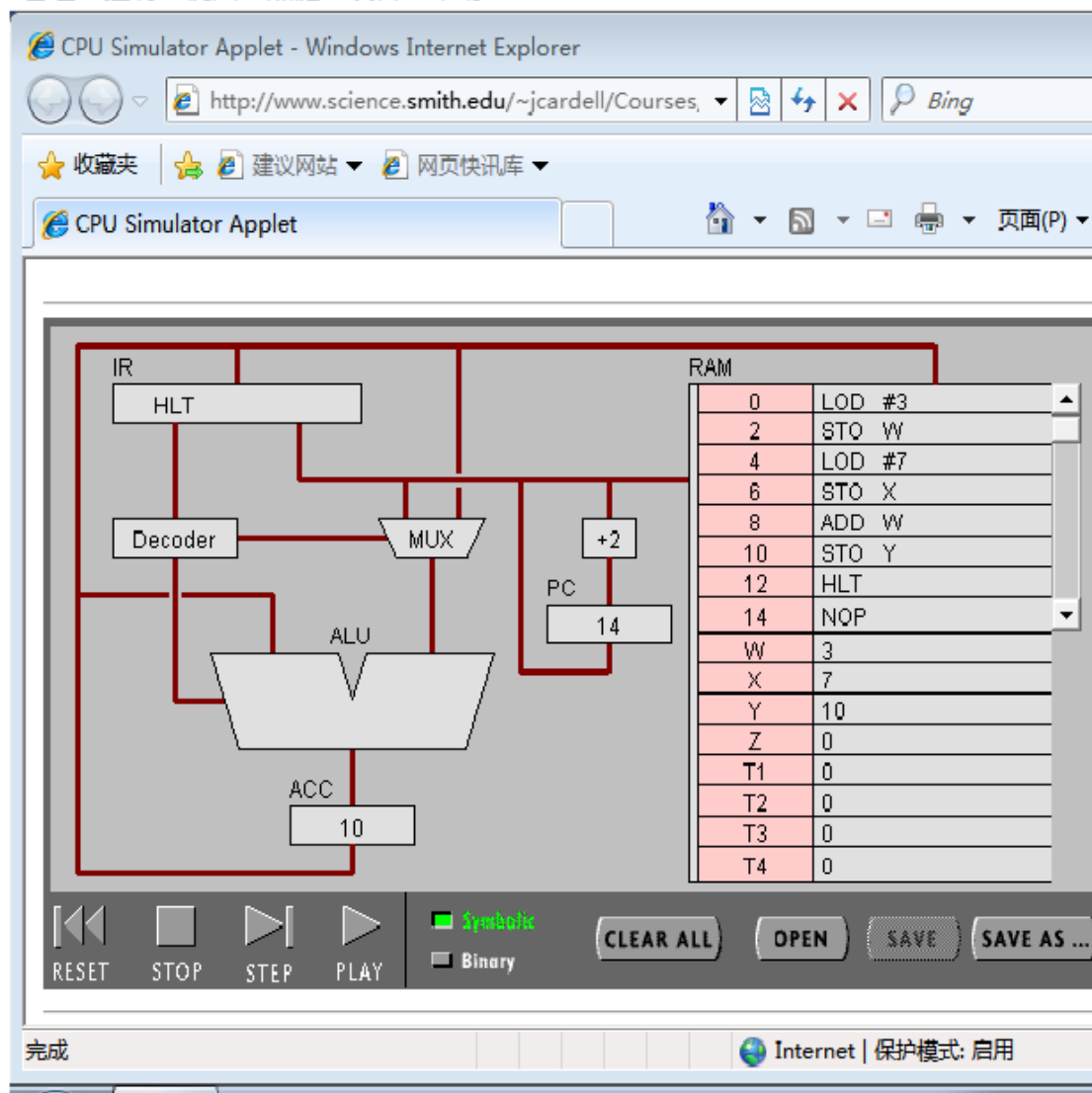
Address	Instruction
0	LOD #3
2	STO W
4	LOD #7
6	STO X
8	ADD W
10	STO Y
12	HLT
14	NOP

W 0
X 0
Y 0
Z 0
T1 0
T2 0
T3 0
T4 0

RESET STOP STEP PLAY

CLEAR ALL OPEN SAVE SAVE AS ...

完成 Internet | 保护模式: 启用



(2) 点 step after step。观察并回答下面问题：

1) PC, IR 寄存器的作用。

IR 指令寄存器，用来保存当前正在执行的一条指令。

PC 程序计数器，存放下一条要执行指令在内存中的地址。

2) ACC 寄存器的全称与作用。

全称：The accumulator(A register)，用于存放操作的数据和结果。

3) 用“LOD #3”指令的执行过程，解释 Fetch-Execute 周期

从 RAM 获取指令及数据 (LOD #3) -> IR 寄存器 -> Decoder(Decode instruction(LOD #3)) -> 数字 3 存在 MUX(数据选择器)中 -> Decoder 将 LOD 指令传给 ALU -> ALU 从 MUX 得到数字 3 -> 累加到 ACC 上 -> PC 累加 2, 用于存放下一条指令的地址

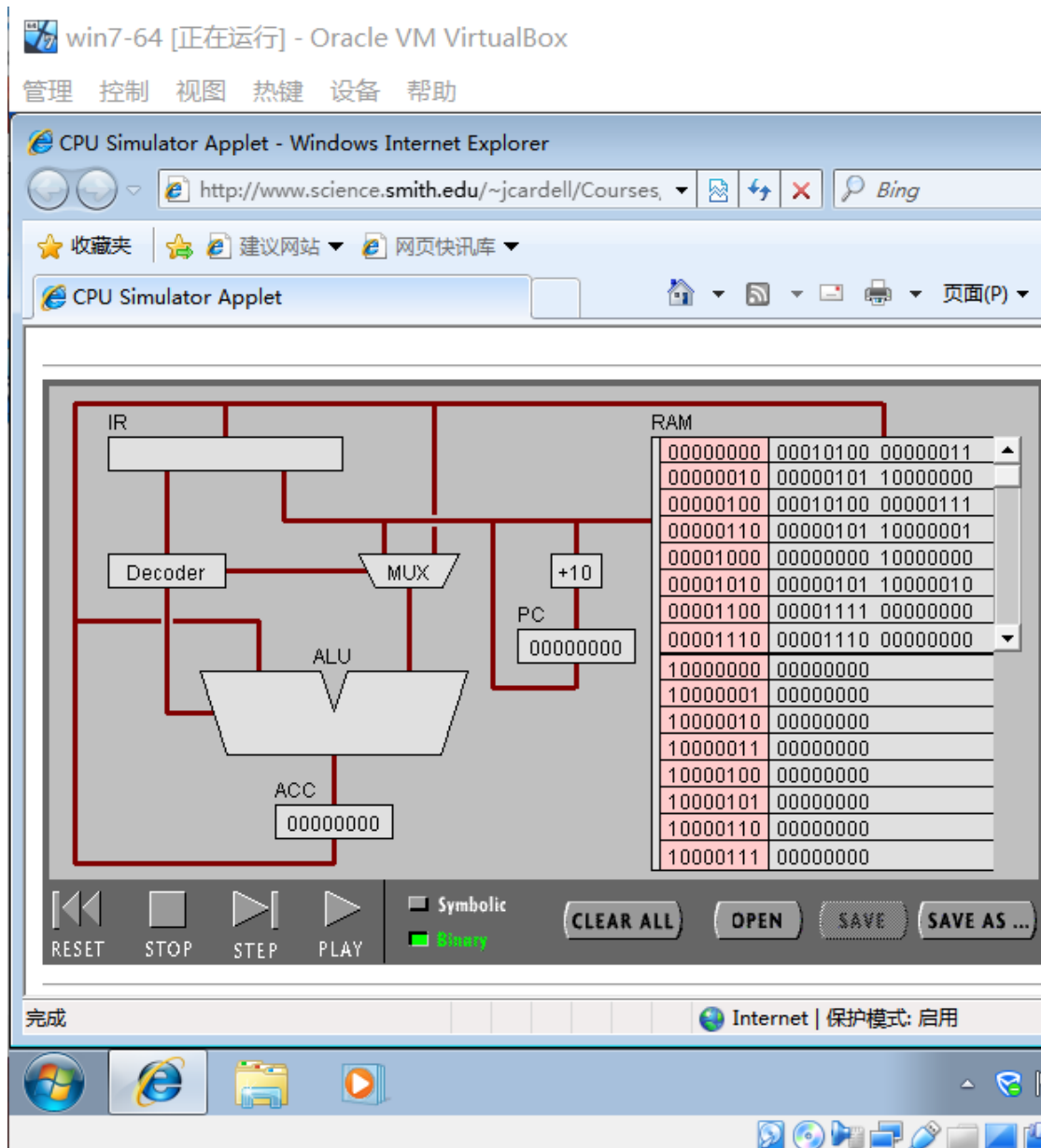
4) 用“ADD W”指令的执行过程, 解释 Fetch-Execute 周期。

从 RAM 获取指令及数据 (ADD W) -> IR 寄存器 -> Decoder(Decode instruction(ADD W)) -> MUX(数据选择器)从 RAM 中获取 W 对应地址的内容, 即数值 3 -> Decoder 将 ADD 指令传给 ALU -> ALU 从 MUX 得到数字 3 -> 累加到 ACC 上 -> PC 累加 2, 用于存放下一条指令的地址

5) “LOD #3”与 “ADD W”指令的执行在 Fetch-Execute 周期级别, 有什么不同。

指令“LOD #3”在经过 IR 寄存器时, 将数据 3 直接传给 MUX, 使 ALU 直接提取, 给 ACC 赋值; 指令“ADD W”在经过 IR 寄存器时, 将 ADD 操作信号传给 ALU, 所以 MUX 还需再访问 RAM 获取 W 的内容, 再使 ALU 提取, 将 W 的值累加到 ACC 上。

(3) 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。

LOD #7 的二进制：0001 0100 0000 0111

第一个字节的含义：指令说明符中的操作码，对应 LOD 操作

第二个字节的含义：操作数说明符，此处为数值 7 的二进制码

2) 解释 RAM 的地址

RAM 的地址被分为两部分，其中第一部分地址用来储存指令，包括

指令说明符以及操作数说明符；第二部分地址用来标识变量的地址，具体存储变量的值。

3) 该机器 CPU 是几位的？（按累加器的位数）

8 位

4) 写出该程序对应的 C 语言表达

```
#include<stdio.h>
```

```
int main(){
```

```
    int w=3;
```

```
    w+=7;
```

```
    return 0;
```

```
}
```

任务 2：简单循环

(1) 输入程序 Program 2，运行并回答问题：

win7-64 [正在运行] - Oracle VM VirtualBox

管理 控制 视图 热键 设备 帮助

CPU Simulator Applet - Windows Internet Explorer

http://www.science.smith.edu/~jcardell/Courses, Bing

收藏夹 建议网站 网页快讯库

CPU Simulator Applet

The screenshot shows a CPU Simulator Applet running in a Windows Internet Explorer browser. The applet displays a circuit diagram with components: IR (Instruction Register), Decoder, MUX (Multiplexer), ALU (Arithmetic Logic Unit), ACC (Accumulator), PC (Program Counter), and RAM. The PC is set to 0, and the ACC is set to 0. The RAM table is as follows:

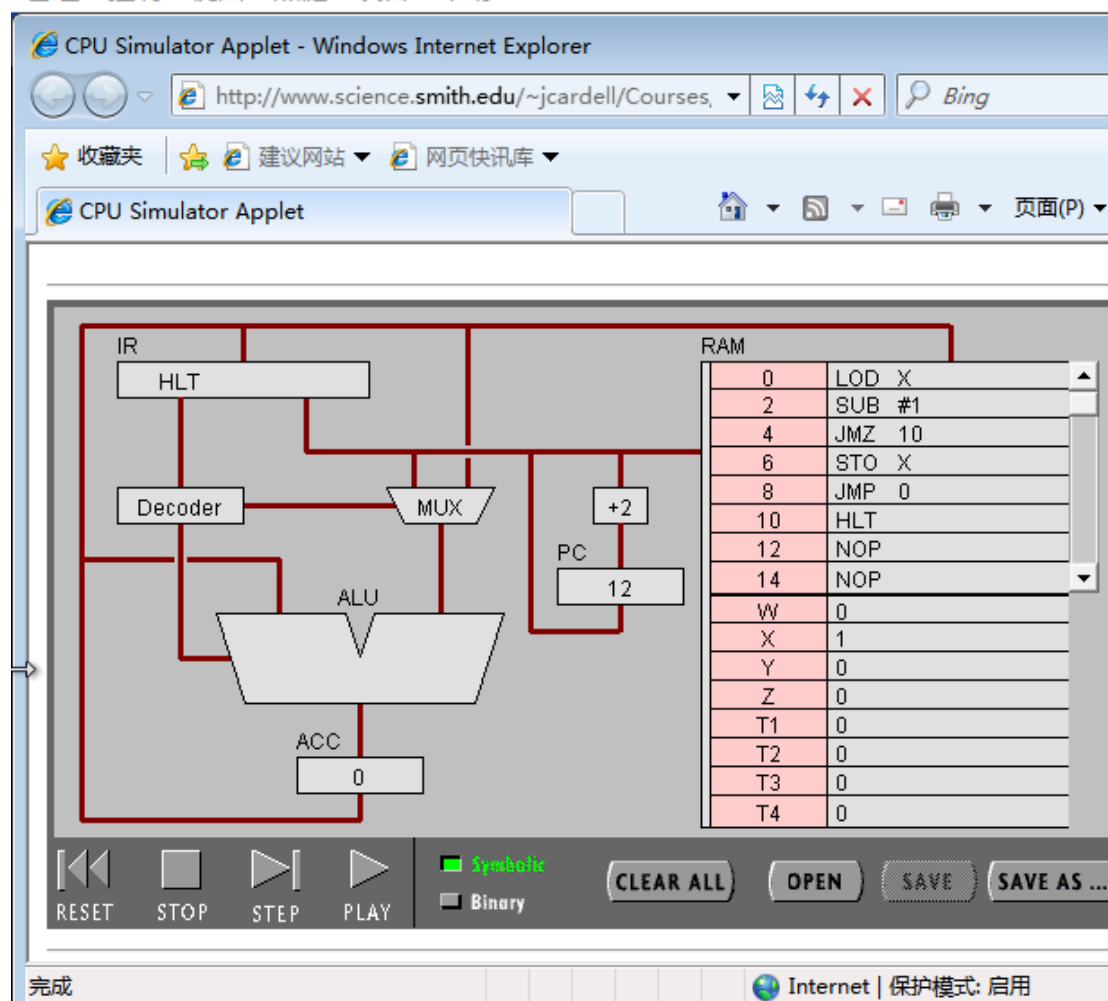
Address	Operation
0	LOD X
2	SUB #1
4	JMZ 10
6	STO X
8	JMP 0
10	HLT
12	NOP
14	NOP
W	0
X	3
Y	0
Z	0
T1	0
T2	0
T3	0
T4	0

RESET STOP STEP PLAY

Binary

CLEAR ALL OPEN SAVE SAVE AS ...

完成 Internet | 保护模式: 启用



1) 用一句话总结程序的功能

给赋 X 一非零初值，进入循环，每次循环 X 减去 1，直到 X 为 1 时 ACC 寄存器存着 0，程序结束。

2) 写出对应的 c 语言程序

```
#include<stdio.h>
```

```
int main(){
```

```
    int x=3;
```

```
    while((x-1)!=0){
```

```
        x--;
```

```

    }

    return 0;

}

```

(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

win7-64 [正在运行] - Oracle VM VirtualBox

管理 控制 视图 热键 设备 帮助

CPU Simulator Applet - Windows Internet Explorer

http://www.science.smith.edu/~jcardell/Courses, Bing

收藏夹 建议网站 网页快讯

CPU Simulator Applet

The screenshot shows the CPU Simulator Applet interface. The main window displays the internal architecture of a simple CPU. The IR (Instruction Register) is connected to a Decoder, which outputs to a MUX (Multiplexer). The MUX also receives input from the PC (Program Counter). The output of the MUX goes to the ALU (Arithmetic Logic Unit). The ALU also receives input from the ACC (Accumulator). The output of the ALU goes back to the ACC. The PC is incremented by 2. The RAM (Random Access Memory) is shown on the right, containing a list of instructions and data. The instructions are: 0: LOD Y, 2: ADD X, 4: STO Y, 6: LOD X, 8: SUB #1, 10: JMZ 16, 12: STO X, 14: JMP 0. The data is: W: 0, X: 3, Y: 0, Z: 0, T1: 0, T2: 0, T3: 0, T4: 0. The ACC is currently 0. The PC is currently 0. The MUX is currently selecting the PC input. The ALU is currently performing an addition. The RAM is currently at address 0.

Address	Instruction
0	LOD Y
2	ADD X
4	STO Y
6	LOD X
8	SUB #1
10	JMZ 16
12	STO X
14	JMP 0

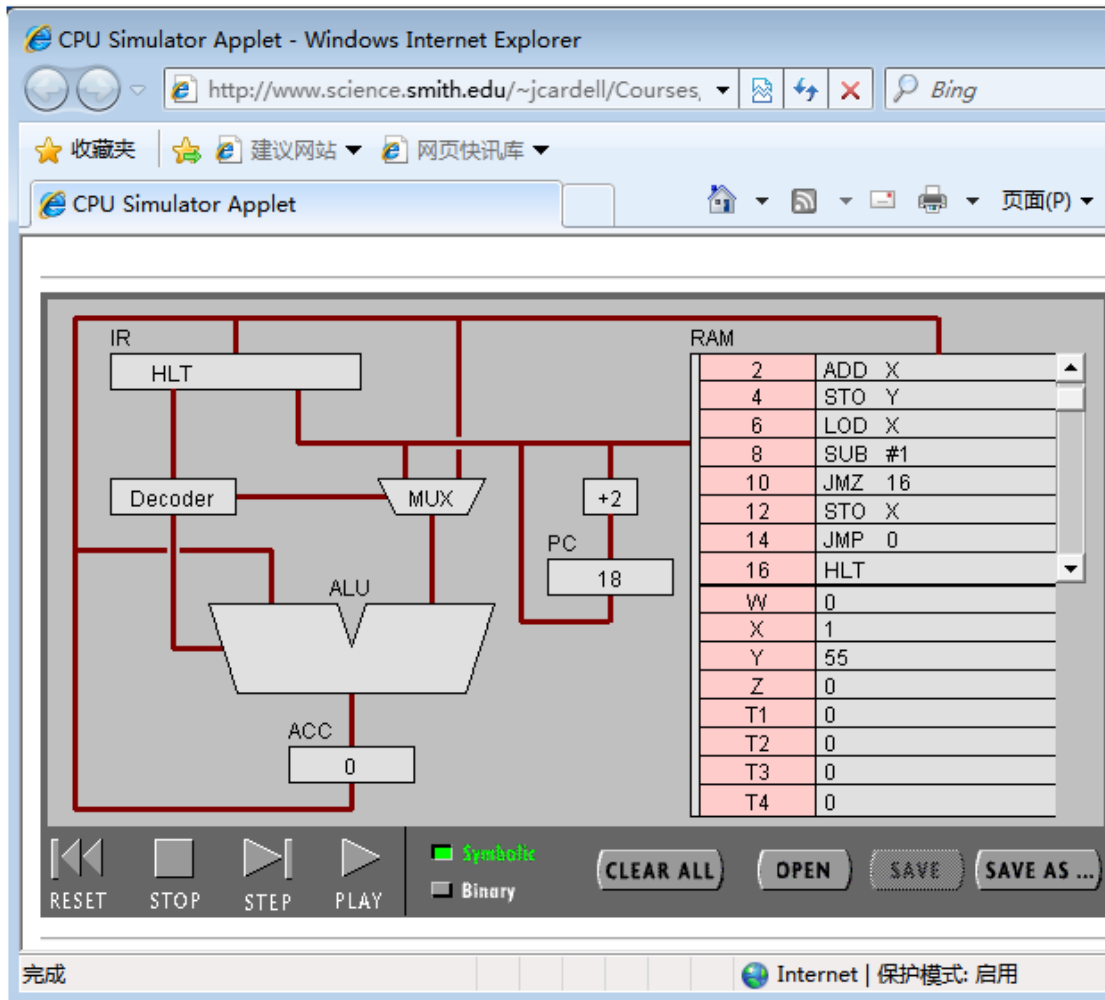
Register	Value
W	0
X	3
Y	0
Z	0
T1	0
T2	0
T3	0
T4	0

RESET STOP STEP PLAY

Binary

CLEAR ALL OPEN SAVE SAVE AS ...

完成 Internet | 保护模式: 启用



1) 写出 c 语言的计算过程

```
#include<stdio.h>
```

```
int main(){
```

```
    int x,y;
```

```
    for(x=10;(x-1)!=0;--x){
```

```
        y+=x;
```

```
    }
```

```
    y+=x;
```

```
    return 0;
```

}

2) 写出机器语言的计算过程

对变量 X 和 Y 分别赋初值 10 和 0，再用循环，每次先 LOD Y，再 ADD X 然后将该值赋给 Y，即 STO Y，然后 X 减去 1，每次均设条件跳转 JMZ 16 语句判断 ACC 当前值是否为 0，若为零，则跳出循环，此时 Y 即为结果，若不为零，则经过无条件跳转语句 JMP 0 回到 LOD X 那一步，直至 X=1 即 ACC 寄存器存着 0 时。

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

区别：高级语言是编一些应用软件，与硬件不直接打交道的语言，比较直接，易懂。机器语言是计算机直接能够接受和应用的语言，一般人难以理解。

联系：高级语言从机器语言发展而来，高级语言最终都通过编译器转换成机器语言。

实验小结

通过 vbox 在 win7 虚拟机上成功创建作业环境。

通过提供的网站软件成功完成 program1：add 2 number 和 program2 的模拟实验。

通过以上任务充分了解汇编语言和高级语言的差别。