**上海大学 计算机学院**

**《计算机组成原理实验》报告六**

**姓名 刘远航 学号 22121883**

**时间 20240117 机位 20 指导教师 刘跃军**

**实验名称: 机器语言程序实验**

**一、实验目的**

1.编制机器语言简单程序。

2.成功运行机器语言程序。

3.理解计算机执行程序的实际过程。

**二、实验原理**

**1. 指令的形式化表示：**

在前几次实验课中我们用中文自然语言表述了各种操作，中国人很容易理解它们，但印度人很难理解它们，机器也无法理解它们，而且自然语言用多了容易产生歧义，所以不适合用来表述大量的操作——写程序。另一方面，在实验三中我们使用微指令的二进制具体编码(16进制数形式)来表示它，在实验五中用机器指令的微程序入口地址(16进制数形式)来表示它们，这几种表示可以直接被机器“理解”，但人理解起来就非常吃力。

表面上看，人理解16进制数形式指令很困难的原因是这些指令太相似了，其实根本原因是16进制数形式的指令没有对指令功能的任何反映，所以“人”用起来不方便，尤其是在使用大量指令来编制解决实际问题的程序时，这种不反映指令逻辑功能的形式更显得难以使用。如何解决这个问题呢？任何人都会想到一个简单的方法——给每条(微)指令再规定一个反映其逻辑功能的“符号”，习惯上称这套符号为“汇编指令”。

**2. 汇编：**

由于汇编指令和16进制数指令是同一条指令的两种形式化符号，所以二者之间存在着一张符号对应表——翻译工具。厂家为实验箱的机器指令规定的两种形式化符号对应表见101～102页的表，其中“助记符”列就是“汇编指令”符号，“机器码1”和“机器码2”是指令的二进制数形式符号。有了这张对应表，人就可以用汇编指令符号来编写程序，以方便编程时的逻辑思维，编好程序后，只要对照这张表按顺序逐条将指令翻译成指令的二进制形式，然后将二进制数形式的程序送入计算机，计算机就能“理解”程序的逻辑操作了。这个翻译过程叫“汇编”。

**三、实验内容**

**1．实验任务一、编写并运行机器语言程序，将R1中的数值左移n次送OUT，n是R2中的数值。(指令码从10开始)**

1. **实验步骤**

①注视仪器，打开电源，手不要远离电源开关，随时准备关闭电源，注意各数码管、发光管的稳定性，静待10秒，确信仪器稳定、无焦糊味。

②根据实验任务设计汇编语言。

|  |
| --- |
| LOOP:MOV A,R1 |
| RL A |
| MOV R1,A |
| MOV A,R2 |
| SUB A,#01 |
| MOV R2,A |
| JZ LOOP1 |
| JMP LOOP |
| MOV A,R1 |
| LOOP1:OUT |

③将汇编语言翻译成机器码并写入相应内存单元。

|  |  |  |  |
| --- | --- | --- | --- |
| 单元 | 机器码 | 单元 | 机器码 |
| 10 | 71 | 17 | A4 |
| 11 | D4 | 18 | 1B |
| 12 | 81 | 19 | AC |
| 13 | 72 | 1A | 10 |
| 14 | 3D | 1B | 71 |
| 15 | 01 | 1C | C4 |
| 16 | 82 |  |  |

④在 μPC 模式下验证功能。赋初值μPC(00)，PC(10)，R1(01)，R2(03)。按STEP 观察微指令执行的过程。

⑤记录实验过程中的现象，关闭实验箱。

1. **实验现象**

赋初值R1=1，R2=3，

当pc从10运行到1D时，out显示为8。

1. **数据记录、分析与处理**

MOV A, R1: 将寄存器R1的值移动到累加器A中。

RL A: 将累加器A中的值左循环移位。

MOV R1, A: 将累加器A中的值移动回寄存器R1。

MOV A, R2: 将寄存器R2的值移动到累加器A中。

SUB A, #01: 用1减去累加器A中的值。

MOV R2, A: 将累加器A中的结果移动回寄存器R2。

JZ LOOP1: 如果累加器A中的值为零，则跳转到标签LOOP1。

JMP LOOP: 无条件跳转到标签LOOP。

1. **实验结论**

很好的完成了实验任务。

**2．实验任务二、编写并运行机器语言程序，完成R3＝R0×R1,结果送OUT。 (指令码从30开始)**

1. **实验步骤**

①注视仪器，打开电源，手不要远离电源开关，随时准备关闭电源，注意各数码管、发光管的稳定性，静待10秒，确信仪器稳定、无焦糊味。

②根据实验任务设计汇编语言。

|  |
| --- |
| MOV A,R0 |
| SUB A,#00 |
| JZ LOOP1 |
| MOV A,R1 |
| SUB A,#00 |
| JZ LOOP1 |
| MOV A,#00 |
| LOOP:MOV A,R3 |
| ADD A,R0 |
| MOV R3,A |
| MOV A,R1 |
| SUB A,#01 |
| MOV R1,A |
| JZ LOOP1 |
| JMP LOOP |
| MOV A,R3 |
| LOOP1:OUT |
| LOOP1:OUT |

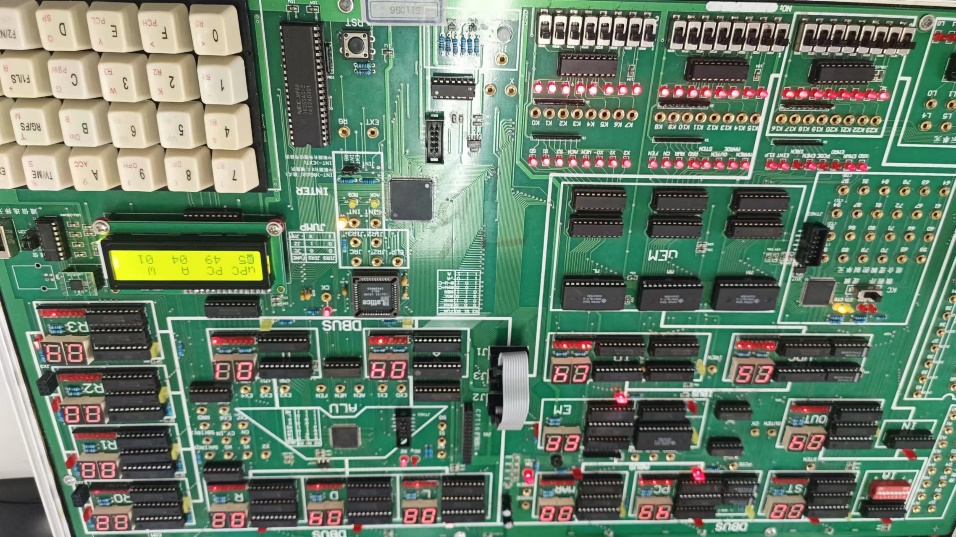
③将汇编语言翻译成机器码并写入相应内存单元。

|  |  |  |  |
| --- | --- | --- | --- |
| 单元 | 机器码 | 单元 | 机器码 |
| 30 | 70 | 3D | 10 |
| 31 | 3C | 3E | 83 |
| 32 | 00 | 3F | 71 |
| 33 | A4 | 40 | 3C |
| 34 | 48 | 41 | 01 |
| 35 | 71 | 42 | 81 |
| 36 | 3C | 43 | A4 |
| 37 | 00 | 44 | 47 |
| 38 | A4 | 45 | AC |
| 39 | 48 | 46 | 3C |
| 3A | 7C | 47 | 73 |
| 3B | 00 | 48 | C4 |
| 3C | 73 |  |  |

④在 μPC 模式下验证功能。赋初值μPC(00)，PC(10)，R0(02)，R1(02)。按STEP 观察微指令执行的过程。

⑤记录实验过程中的现象，关闭实验箱。

1. **实验现象**



最后out显示为4.

1. **数据记录、分析与处理**

MOV A, R0: 将寄存器R0的值移动到累加器A中。

SUB A, #00: 用0减去累加器A中的值。

JZ LOOP1: 如果累加器A中的值为零，则跳转到LOOP1。

MOV A, R1: 将寄存器R1的值移动到累加器A中。

SUB A, #00: 同样，用0减去累加器A中的值，没有实际影响。

JZ LOOP1: 如果累加器A中的值为零，则跳转到LOOP1。

MOV A, #00: 将立即数0移动到累加器A中

接下来是一个循环结构：

LOOP: MOV A, R3: 将寄存器R3的值移动到累加器A中。

ADD A, R0: 将寄存器R0的值加到累加器A中。

MOV R3, A: 将累加器A中的值移动回寄存器R3。

MOV A, R1: 将寄存器R1的值移动到累加器A中。

SUB A, #01: 用1减去累加器A中的值。

MOV R1, A: 将累加器A中的结果移动回寄存器R1。

JZ LOOP1: 如果累加器A中的值为零，则跳转到LOOP1。

JMP LOOP: 无条件跳转到标签LOOP。

MOV A, R3: 将寄存器R3的值移动到累加器A中。

OUT: 输出累加器A中的值。

1. **实验结论**

很好的完成了任务二，达到了实验目的。

**四、建议和体会**

1.感觉汇编语言语法比较少，理解起来容易，但自己编起来可能不太熟悉。

2.运算起来只能用A寄存器，低级语言玩的果然时“资源”。

3.觉得学校应该把汇编语言改成必修。

**五、思考题**

问题：建立“中文汇编指令”需要哪些条件？

答：根据本实验有关内容，可以了解到，汇编语言仅仅只是一种符号和机器 码之间的简单映射。所以如果要建立中文汇编指令，那么我们需要编写中文汇编 指令的解析器，而这需要我们知道机器本身的指令集，换言之我们需要找到机器 的机器指令表，然后将我们的中文符号与这张表上的指令进行对应，从而形成符 号对应表并将其在硬件与软件上实现。