上海大学　计算机学院

《计算机组成原理实验》报告十二

姓名 孔馨怡　　**学号** 22122128

时间 周一 9-11　　**机位** 17　　**指导教师** 顾惠昌

实验名称：建立汇编指令系统

一、实验目的

1.建立一个含中文助记符的汇编指令系统。

2.用建立的指令系统编制一段程序，并运行之。

二、实验原理

1.**编制汇编指令**：在实验三的“举例”中我们编制了一条微指令，它实现“A＋W 的结果右移一位后的值输出到 OUT”，本质上它是编制了这个微指令对应的 24 个控制信号的电平；实验五的任务 2 中，我们修改机器指令 E8，使其成为“输出 A＋W 的结果左移一位后的值到 OUT”指令，它本质上是编制了 μEM 中从 E8 开始的连续 4 个地址中的 24bit 值，即连续的四条微指令；实验六的背景知识 2 使我们知道：“汇编指令是表达机器指令**功能**的指令**助记符**，二者的对应关系由编制的汇编指令表确定”。按步完成这三个编制过程，就定义好一条全新的汇编指令，进一步也可以定义一个汇编指令系统——指令集。

汇编环境 CP226 考虑到了教学上定义汇编指令系统的需求，提供了完成这三个编制任务的集成环境，只要按规定的格式送入编制的符号，系统就会生成相应的汇编指令或汇编指令系统。

2. 汇编表文件:这个文件的后缀为 .DAT,它是一个二维表格式文件，其每一行对应一条指令，这个表共有 3 列，如图 1。第一列是指令的汇编助记符，宽度为 20 个半角字符。第二列是指令的 16 进制编码形式（机器指令），在实验箱系统就是指令的微程序在 μEM 中的起始地址，宽度为 8 个半角字符。第 3 列是这条指令的字节数，宽度为 1 个半角字符，这是本表的重要汇编信息，也是设立本表的原因之一。

这个文件的主要作用是：当编译（汇编）源程序时，查此表把汇编指令翻译成机器指令。即这就是汇编表。

构造这个表文件时也不能带标题行。利用已有.DAT 文件做为模板来构建新指令系统比较方便。具体操作见实验提示。

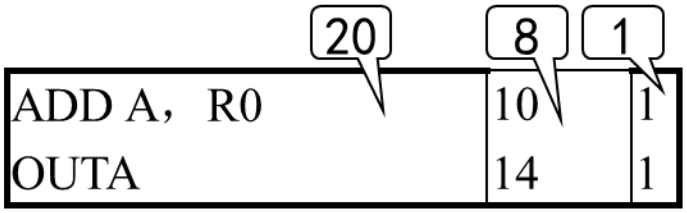

图1. 汇编表文件格式

3. 微程序型指令文件：这个文件的后缀为 .MIC,它也是一个二维表格式的文件，其每一行对应一条微指令，这个表共有 11 列（字段），每一列都定义好了属性和宽度，例如：图 2 是指令集 insfile1.MIC 的格式，这个指令集的全部内容见指导书 103 页到 110 页。

这个表的主要作用是：当系统调用此文件时把其第 4 列“微程序”的内容送入其第 3 列“微地址”指定的μEM（微程序存储器）单元。即初始化μEM。表的第一列为指令的汇编助记符,内容与表 1 的第 1 列一致。5 到 11 列是对本行微指令的说明，内容可以省略。

构造这个表文件时不能带标题行。利用已有.MIC 文件做为模板来构建新指令系统比较方便。具体操作见实验提示。

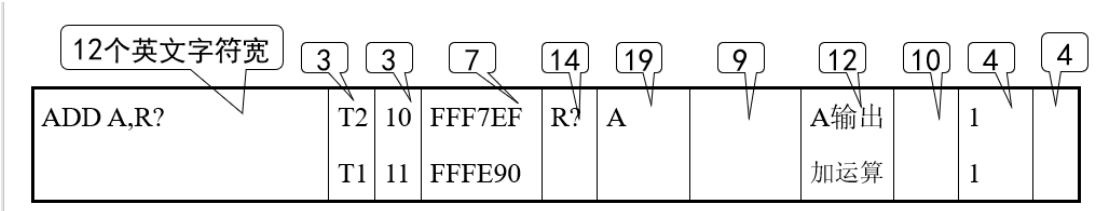

图2. 微程序型指令文件格式

4. 指令的机器码文件:这个文件的后缀为 .MAC, 也是一个二维表格式文件，每一行对应一条指令，表共有 5 列，如图 3。第 1 列是汇编助记符，宽度 14，与表 1 的第 1 列一致。第 2 列是机器码 1，它是指令的微程序在μEM 中起始地址的二进制表示，其最后两位是对 R0～R3 的选择，所以与表 2 的第 3 列一致，宽度为 15。第 3 列是机器码 2，是指令带的立即数或存储器地址。第 4 列是机器码 3，是指令带的第二个存储器地址，宽度 2。第 5 列是注释，宽度 100，用于对指

令进行说明。实验箱默认的指令系统 insfile1 没有机器码 3，其此表的具体内容见指导书 101 页和 102 页。

这个文件的主要作用是：解释汇编表的机器码细节，所以当编译源程序中的多字节指令时，可能要查此表。

构造这个表文件时也不能带标题行。利用已有.MAC 文件做为模板来构建新指令系统比较方便。具体操作见实验提示。

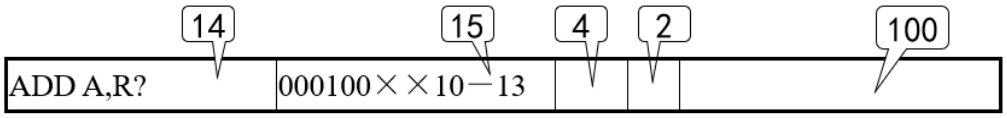

图3. 机器码文件格式

三、实验内容

1．实验任务一：编制一个汇编指令系统，包含下列助记符：

指令助记符	指令意义描述
输入 A A，#*	将立即数打入累加器
减法 A，#*	累加器 A 减立即数
零跳转*	运算结果为零跳转指令
跳到*	无条件跳转指令
输出	累加器 A 输出到 OUT

用所编制的指令系统，写出源程序，完成OUT寄存器交替显示5、4、3、2、1。交替显示时间为1秒。

(1) 实验步骤

1、编写汇编表文件 mission.DAT，文件如下：

输入 A A,#*　　04　　2

减法 A,#*　　08　　2

跳到 *　　0C　　2

输出　　10　　1

零跳转*　　14　　2

分析：

第一列是我们编的指令的名称和格式；

第二列是所在地址；

第三列是这条指令所需的机器码的条数，例如输入的指令中，第一条机器码的目的是表明这是输入的指令，第二条机器码的含义就是输入的立即数；

零跳转的实现：查表可知，零跳转和跳转的机器码前面都是一致的都为 000011XX，这时候要利用前几次实验中学到的内容，最后“XX”两位由 PC 控制，只要最后地址的末尾为 4，在相同机器码 000011XX 的情况下就可以变成 000011X4，从而实现零跳转，而非无条件跳转。

2、编写微程序型指令文件 mission.MIC,文件内容如下：

```
_FATCH_      T0 00 CBFFFF
                01 FFFFFF
                02 FFFFFF
                03 FFFFFF
输入 A A,#*   T1 04 C7FFF7
             T0 05 CBFFFF
                06 FFFFFF
                07 FFFFFF
减法 A,#*    T2 08 C7FFEF
             T1 09 CBFE91
             T0 0A CBFFFF
                0B FFFFFF
跳到 *       T1 0C C6FFFF
             T0 0D CBFFFF
                0E FFFFFF
                0F FFFFFF
输出         T1 10 FFDF9F
             T0 11 CBFFFF
```

```
                    12 FFFFFF
                    13 FFFFFF
零跳转 *     T1     14 C6FFFF
                T0  15 CBFFFF
                    16 FFFFFF
                    17 FFFFFF
```

分析：

第一列是我们编的指令的名称和格式，注意这里要加上最开始的_FATCH_指令；

第二列是有效机器码的条数，由下往上从 T0 开始计数，取址指令也包含在内；

第三列是地址；

第四列是指令的机器码，注意每个指令的最后一条有效指令是 CBFFFF，是为了完成取址的操作，每个助记符下面最多四条指令，不足四条的全部由 FFFFFF 补齐，有效操作的机器码如何编写前面实验中已经学到过，由各个功能的 10（开关）组成，转换为十六进制即可，当然也可以去查看系统中原本的 MIC 文件中对应功能的机器码是哪一条，可以复制过来，因为功能实现一样，指令机器码就是一样的；

零跳转的实现：我们可以发现完成零跳转的机器码和完成无条件跳转的机器码都是 C6FFFF，我们也可以发现零跳转的地址是 14，由上面 DAT 文件已经分析过的原理，即可实现零跳转操作。

3、编写指令的机器码文件 mission.MAC,具体文件如下：

```
_FATCH_          000000XX
输入 A  A,#*     000001XX     将立即数放入到 A 寄存器
减法  A,#*       000010XX     累加器 A 减立即数
跳到 *           000011XX     无条件跳转指令
输出             000100XX     累加器 A 输出到 OUT
零跳转 *         000011XX     运算结果为零跳转指令
```

分析：

该文件解释汇编表的机器码细节，所以如果文件为空的话其实不影响实际实验，但是这个表的完善是为了更好的传播和保存你的汇编指令系统。

4、编写微程序函数 mission.ASM，编写文件如下：

指令		分析
LOOP:	输入A A,#05H	显示05H
	输出	
	输入A A,#10H	延时功能实现
T1:	减法 A,#01H	
	零跳转 T2	
	跳到 T1	
T2:	输入A A,#04H	显示04H
	输出	
	输入A A,#10H	延时功能实现
T3:	减法 A,#01H	
	零跳转 T4	
	跳到 T3	
T4:	输入A A,#03H	显示03H
	输出	
	输入A A,#10H	延时功能实现
T5:	减法 A,#01H	
	零跳转 T6	
	跳到 T5	
T6:	输入A A,#02H	显示02H
	输出	
	输入A A,#10H	延时功能实现
T7:	减法 A,#01H	
	零跳转 T8	
	跳到 T7	
T8:	输入A A,#01H	显示01H
	输出	
	输入A A,#10H	延时功能实现
T9:	减法 A,#01H	
	零跳转 T10	
	跳到 T9	
T10:	跳到 LOOP	继续交替显示

分析：

按照自己编写的汇编指令系统，编写 ASM 文件如右，该过程其实我们已经在这学期前面几次实验中很熟悉了，这里不过多赘述。

但是需要特别强调的是因为我们的要求是只能运用前面汇编指令系统中的指令进行编写，所以没有其他寄存器去存储 A 中输出的数字，而延时的操作又要用到 A 寄存器，所以我们只能通过像右边这样的方式，在每次显示完 5、4、3、2、1 中的每个数字后，手动移动下一个数字后显示，然后又继续完成延时的操作，最后五个数字全部显示完又回到显示 05H 处即可。

如何计算延时所需的次数和时间：

根据频率 114.8Hz，再根据公式：频率*周期=1，求出周期，然后可以得出 1s 内会需要多少次

周 期 T = 1/f = 1/114.8 = 0.0087108 s ；

那么一秒就需要 11 次左右。

5、选择通信口，打开 mission.ASM 文件。

6、将提前编制的指令系统（mission.DAT，mission.MIC,mission.MAC）调入。

7、点击编译下载，观察试验箱的变化。

⑵ 实验分析

实验分析在前面的操作过程中顺便记录了，此处略，可以往前查找翻看。

⑶ 实验现象

最后现象：

OUT 寄存提交替显示 5、4、3、2、1…

其他小发现：

详见建议与体会。

⑷ 实验结论

成功编制一个汇编指令系统，OUT 寄存提交替显示 5、4、3、2、1…

四、建议和体会

1.关于中文字占两个字符：在我编写 DAT、ASM、MIC、MAC 等文件时，光标移动到已经打好的中文字的中间时，会发现光标卡在了中文字中间，而且如果我选中了中文字，此时中文字将由一个字变成两个看起来像乱码的字符，这一现象可以说明中文是两个字符组成的。

2.关于缩进和对齐：汇编指令系统在编写的过程中，需要有严格的字符数量要求。在我编写 DAT、ASM、MIC、MAC 等文件时，文件的内容类似表格，表格的每一列都需要严格的对齐,这时候不能用缩进来对齐,只可以用空格来手动对齐，如果用了缩进，在调入该汇编指令系统将会显示不合法或者装载不成功、有不合规字符等的提示。

3.编写 DAT、MIC、MAC 等文件时，文件名必须一致，放在同一个文件夹中。

五、思考题

为什么汇编指令中可以用“中文符号”？

答：当你编写汇编代码时，你可能注意到可以使用中文符号来表示指令。这是因为汇编语言中的助记符最多可以包含 20 个半角字符，尽管一个中文字符占据两个字符的空间，但计算机仍然可以通过查找表格找到相应的机器指令。。