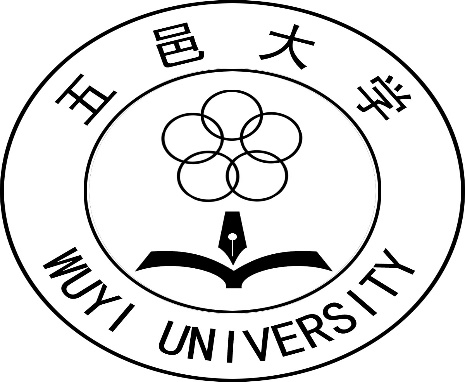
**《计算机组成原理》**

**课程设计报告**

****

**学 院： 智能制造学部**

**专 业： 电子信息工程(信息安全)**

**班 级：**

**学 号：**

**姓 名：**

**指导老师： 杭维颖**

**日 期： 年 月 日**

**20 ~20 学年 第一学期**

**模型计算机系统的设计**

### 一、 实验目的

综合运用所学计算机组成原理知识，设计并实现较为完整的计算机。

### 二、 实验设备

PC机一台，TD-CMA实验系统一套。

### 三、 实验原理

模型计算机的数据格式及指令系统。

**1．数据格式**

模型机规定采用定点补码表示法表示数据，字长为８位，8位全用来表示数据（最高位不表示符号），数值表示范围是： 0≤X≤28－1。

**2．指令设计**

模型机设计三大类指令共十五条，其中包括运算类指令、控制转移类指令，数据传送类指令。运算类指令包含三种运算，算术运算、逻辑运算和移位运算，设计有6条运算类指令，分别为：ADD、AND、INC、SUB、OR、RR，所有运算类指令都为单字节，寻址方式采用寄存器直接寻址。控制转移类指令有三条HLT、JMP、BZC，用以控制程序的分支和转移，其中HLT为单字节指令，JMP和BZC为双字节指令。数据传送类指令有IN、OUT、MOV、LDI、LAD、STA共6条，用以完成寄存器和寄存器、寄存器和I/O、寄存器和存储器之间的数据交换，除MOV指令为单字节指令外，其余均为双字节指令。

1. **指令格式**

所有单字节指令（ADD、AND、INC、SUB、OR、RR、HLT和MOV）格式如下：

|  |  |  |
| --- | --- | --- |
| ７６５４ | ３２ | １０ |
| OP-CODE | RS | RD |

其中，OP-CODE为操作码，RS为源寄存器，RD为目的寄存器，并规定：

|  |  |
| --- | --- |
| RS或RD | 选定的寄存器 |
| 00  01  10  11 | R0  R1  R2  R3 |

IN和OUT的指令格式为：

|  |  |  |  |
| --- | --- | --- | --- |
| ７６５４（1） | ３２（1） | １０（1） | 7—0（2） |
| OP-CODE | RS | RD | P |

其中括号中的1表示指令的第一字节，2表示指令的第二字节，OP-CODE为操作码，RS为源寄存器，RD为目的寄存器，P为I/O端口号，占用一个字节，系统的I/O地址译码原理见图6-1（在地址总线单元）。



图6-1 I/O地址译码原理图

由于用的是地址总线的高两位进行译码，I/O地址空间被分为四个区，如表6-1所示：

表6-1 I/O地址空间分配

|  |  |  |
| --- | --- | --- |
| A7 A6 | 选定 | 地址空间 |
| 00 | IOY0 | 00-3F |
| 01 | IOY1 | 40-7F |
| 10 | IOY2 | 80-BF |
| 11 | IOY3 | C0-FF |

**4．指令系统**

本模型机共有15条基本指令，表6-3列出了各条指令的格式、汇编符号、指令功能。

表6-3 指令描述



### 四、实验步骤

1. 按图6-6连接实验线路，仔细检查接线后打开实验箱电源。

2. 写入实验程序，并进行校验，手动写入

1) **手动写入和校验微程序**

(1) 手动写入微程序

① 将时序与操作台单元的开关KK1置为‘停止’档，KK3置为‘编程’档，KK4置为‘控存’档，KK5置为‘置数’档。

② 使用CON单元的SD05——SD00给出微地址，IN单元给出低8位应写入的数据，连续两次按动时序与操作台的开关ST，将IN单元的数据写到该单元的低8位。

③ 将时序与操作台单元的开关KK5置为‘加1’档。

④ IN单元给出中8位应写入的数据，连续两次按动时序与操作台的开关ST，将IN单元的数据写到该单元的中8位。IN单元给出高8位应写入的数据，连续两次按动时序与操作台的开关ST，将IN单元的数据写到该单元的高8位。

⑤ 重复①、②、③、④四步，将表6-5的微代码写入2816芯片中。

(2) 手动校验微程序

① 将时序与操作台单元的开关KK1置为‘停止’档，KK3置为‘校验’档，KK4置为‘控存’档，KK5置为‘置数’档。

② 使用CON单元的SD05——SD00给出微地址，连续两次按动时序与操作台的开关ST，MC单元的指数据指示灯 M7——M0显示该单元的低8位。

③ 将时序与操作台单元的开关KK5置为‘加1’档。

④ 连续两次按动时序与操作台的开关ST，MC单元的指数据指示灯 M15——M8显示该单元的中8位，MC单元的指数据指示灯 M23——M16显示该单元的高8位。

⑤ 重复①、②、③、④四步，完成对微代码的校验。如果校验出微代码写入错误，重新写入、校验，直至确认微指令的输入无误为止。

2）**手动写入机器程序**

① 将时序与操作台单元的开关KK1置为‘停止’档，KK3置为‘编程’档，KK4置为‘主存’档，KK5置为‘置数’档。

② 使用CON单元的SD7——SD0给出地址，IN单元给出该单元应写入的数据，连续两次按动时序与操作台的开关ST，将IN单元的数据写到该存储器单元。

③ 将时序与操作台单元的开关KK5置为‘加1’档。

④ IN单元给出下一地址（地址自动加1）应写入的数据，连续两次按动时序与操作台的开关ST，将IN单元的数据写到该单元中。然后地址会又自加1，只需在IN单元输入后续地址的数据，连续两次按动时序与操作台的开关ST，即可完成对该单元的写入。⑤ 亦可重复①、②两步，将所有机器指令写入主存芯片中。

1. 手动校验机器程序

① 将时序与操作台单元的开关KK1置为‘停止’档，KK3置为‘校验’档，KK4置为‘主存’档，KK5置为‘置数’档。

② 使用CON单元的SD7——SD0给出地址，连续两次按动时序与操作台的开关ST，CPU内总线的指数据指示灯 D7——D0显示该单元的数据。

③ 将时序与操作台单元的开关KK5置为‘加1’档。

④ 连续两次按动时序与操作台的开关ST，地址自动加1，CPU内总线的指数据指示灯 D7——D0显示该单元的数据。此后每两次按动时序与操作台的开关ST，地址自动加1，CPU内总线的指数据指示灯 D7——D0显示该单元的数据，继续进行该操作，直至完成校验，如发现错误，则返回写入，然后校验，直至确认输入的所有指令准确无误。

⑤ 亦可重复①、②两步，完成对指令码的校验。如果校验出指令码写入错误，重新写入、校验，直至确认指令的输入无误为止。

 **图6—6 实验接线图**

3） **运行程序**

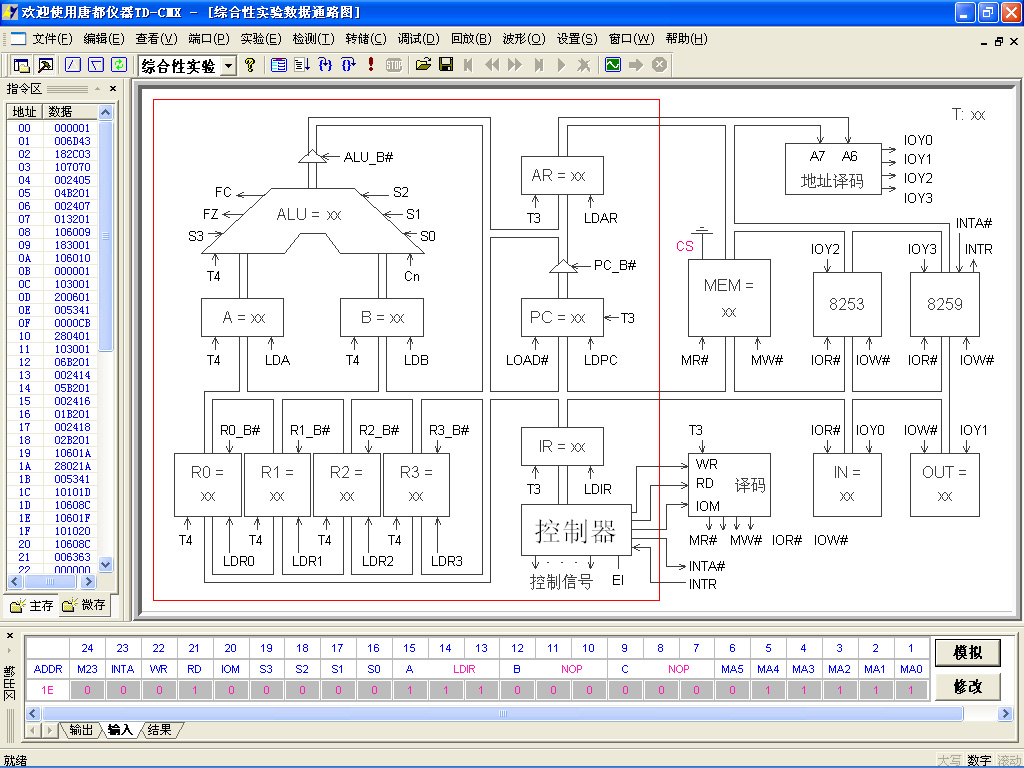
将时序与操作台单元的开关KK1、KK3置为‘运行’档，按动CON单元的总清按钮CLR，将使程序计数器PC、地址寄存器AR和微程序地址为00H，程序可以从头开始运行，暂存器A、B，指令寄存器IR和OUT单元也会被清零。

将时序与操作台单元的开关KK2置为‘单步’档，每按动一次ST按钮，即可单步运行一条微指令，对照微程序流程图，观察微地址显示灯是否和流程一致。每运行完一条微指令，观测一次数据总线和地址总线，对照数据通路图，分析总线上的数据是否正确。

当模型机执行完OUT指令后，检查OUT单元显示的数是否正确，按下CON单元的总清按钮CLR，改变IN单元的值，再次执行机器程序，从OUT单元显示的数判别程序执行是否正确。

**五、总体设计**

本模型机的数据通路框图如图6—2所示。



**图6—2 数据通路框图**

和前面的实验相比，复杂模型机实验指令多，寻址方式多，只用一种测试已不能满足设计要求，为此指令译码电路需要重新设计。如图6-3所示在IR单元的INS\_DEC中实现。





**图6-3 指令译码原理图**

本实验中要用到四个通用寄存器R3…R0，而对寄存器的选择是通过指令的低四位，为此还得设计一个寄存器译码电路，在IR单元的REG\_DEC（GAL16V8）中实现，如图6-4所示。



**图6-4 寄存器译码原理图**

根据机器指令系统要求，设计微程序流程图及确定微地址，如图6-5所示。

按照系统建议的微指令格式，见表6-4，参照微指令流程图，将每条微指令代码化，译成二进制代码表，见表6-5，并将二进制代码表转换为联机操作时的十六进制格式文件

**表6-4 微指令格式**



表6-5 二进制代码表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 十六进制表示 | 高五位 | S3-S0 | A字段 | B字段 | C 字段 | UA5-UA0 |
| 00 | 00 00 01 | 00000 | 0000 | 000 | 000 | 000 | 000001 |
| 01 | 00 6D 43 | 00000 | 0000 | 110 | 110 | 101 | 000011 |
| 03 | 10 70 70 | 00010 | 0000 | 111 | 000 | 001 | 110000 |
| 04 | 00 24 05 | 00000 | 0000 | 010 | 011 | 000 | 000101 |
| 05 | 04 B2 01 | 00000 | 1001 | 011 | 001 | 000 | 000001 |
| 06 | 00 24 07 | 00000 | 0000 | 010 | 011 | 000 | 000111 |
| 07 | 01 32 01 | 00000 | 0010 | 011 | 001 | 000 | 000001 |
| 08 | 10 60 09 | 00010 | 0000 | 110 | 000 | 000 | 001001 |
| 09 | 18 30 01 | 00011 | 0000 | 011 | 000 | 000 | 000001 |
| 0A | 10 60 10 | 00010 | 0000 | 110 | 000 | 000 | 010000 |
| 0B | 00 00 01 | 00000 | 0000 | 000 | 000 | 000 | 000001 |
| 0C | 10 30 01 | 00010 | 0000 | 011 | 000 | 000 | 000001 |
| 0D | 20 06 01 | 00100 | 0000 | 000 | 001 | 100 | 000001 |
| 0E | 00 53 41 | 00000 | 0000 | 101 | 001 | 101 | 000001 |
| 0F | 00 00 CB | 00000 | 0000 | 000 | 000 | 011 | 001011 |
| 10 | 28 04 01 | 00101 | 0000 | 000 | 010 | 000 | 000001 |
| 11 | 10 30 01 | 00010 | 0000 | 011 | 000 | 000 | 000001 |
| 12 | 06 B2 01 | 00000 | 1101 | 011 | 001 | 000 | 000001 |
| 13 | 00 24 14 | 00000 | 0000 | 010 | 011 | 000 | 010100 |
| 14 | 05 B2 01 | 00000 | 1011 | 011 | 001 | 000 | 000001 |
| 15 | 00 24 16 | 00000 | 0000 | 010 | 011 | 000 | 010110 |
| 16 | 01 B2 01 | 00000 | 0011 | 011 | 001 | 000 | 000001 |
| 17 | 00 24 18 | 00000 | 0000 | 010 | 011 | 000 | 011000 |
| 18 | 02 B2 01 | 00000 | 0101 | 011 | 001 | 000 | 000001 |
| 1B | 00 53 41 | 00000 | 0000 | 101 | 001 | 101 | 000001 |
| 1C | 10 10 1D | 00010 | 0000 | 001 | 000 | 000 | 011101 |
| 1D | 10 60 8C | 00010 | 0000 | 110 | 000 | 010 | 001100 |
| 1E | 10 60 1F | 00010 | 0000 | 110 | 000 | 000 | 011111 |
| 1F | 10 10 20 | 00010 | 0000 | 001 | 000 | 000 | 100000 |
| 20 | 10 60 8C | 00010 | 0000 | 110 | 000 | 010 | 001100 |
| 28 | 10 10 29 | 00010 | 0000 | 001 | 000 | 000 | 101001 |
| 29 | 00 28 2A | 00000 | 0000 | 010 | 100 | 000 | 101010 |
| 2A | 04 E2 2B | 00000 | 1001 | 110 | 001 | 000 | 101011 |
| 2B | 04 92 8C | 00000 | 1001 | 001 | 001 | 010 | 001100 |
| 2C | 10 10 2D | 00010 | 0000 | 001 | 000 | 000 | 101101 |
| 2D | 00 2C 2E | 00000 | 0000 | 010 | 110 | 000 | 101110 |
| 2E | 04 E2 2F | 00000 | 1001 | 110 | 001 | 000 | 101111 |
| 2F | 04 92 8C | 00000 | 1001 | 001 | 001 | 010 | 001100 |
| 30 | 00 16 04 | 00000 | 0000 | 001 | 011 | 000 | 000100 |
| 31 | 00 16 06 | 00000 | 0000 | 001 | 011 | 000 | 000110 |
| 32 | 00 6D 48 | 00000 | 0000 | 110 | 110 | 101 | 001000 |
| 33 | 00 6D 4A | 00000 | 0000 | 110 | 110 | 101 | 001010 |
| 34 | 00 34 01 | 00000 | 0000 | 011 | 010 | 000 | 000001 |
| 35 | 00 00 35 | 00000 | 0000 | 000 | 000 | 000 | 110101 |
| 36 | 00 6D 51 | 00000 | 0000 | 110 | 110 | 101 | 010001 |
| 37 | 00 16 12 | 00000 | 0000 | 001 | 011 | 000 | 010010 |
| 38 | 00 16 13 | 00000 | 0000 | 001 | 011 | 000 | 010011 |
| 39 | 00 16 15 | 00000 | 0000 | 001 | 011 | 000 | 010101 |
| 3A | 00 16 17 | 00000 | 0000 | 001 | 011 | 000 | 010111 |
| 3B | 00 00 01 | 00000 | 0000 | 000 | 000 | 000 | 000001 |
| 3C | 00 6D 5C | 00000 | 0000 | 110 | 110 | 101 | 011100 |
| 3D | 00 6D 5E | 00000 | 0000 | 110 | 110 | 101 | 011110 |
| 3E | 00 6D 68 | 00000 | 0000 | 110 | 110 | 101 | 101000 |
| 3F | 00 6D 6C | 00000 | 0000 | 110 | 110 | 101 | 101100 |

**图6—5 微程序流程图**

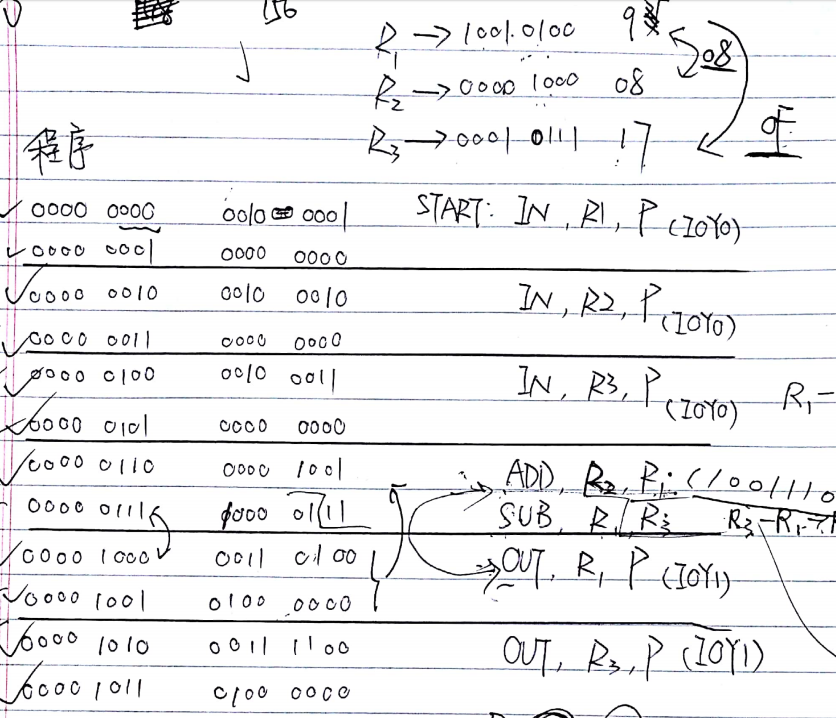
**六、指令系统的设计**

### 方案一：编程实现8位数的X+Y-Z的计算并显示输出的程序

**（1）程序设计及说明：**

|  |  |  |  |
| --- | --- | --- | --- |
| 地址 | 内容 | 助记符 | 说明 |
| 0000 0000 | 0010 0001 | START: IN, R1, P(IOYO) | 从IN单元读入计数初值，并送入到R1寄存器中 |
| 0000 0001 | 0000 0000 |
| 0000 0010 | 0010 0010 | IN, R2, P(IOYO) | 从IN单元读入计数初值，并送入到R2寄存器中 |
| 0000 0011 | 0000 0000 |
| 0000 0100 | 0010 0011 | IN, R3, P(IOYO) | 从IN单元读入计数初值，并送入到R3寄存器中 |
| 0000 0101 | 0000 0000 |
| 0000 0110 | 0000 1001 | ADD, R2, R1 | 将R1和R2的数据相加，并送到R1寄存器中 |
| 0000 0111 | 1000 0111 | SUB, R1, R3 | 将R1和R3的数据相加，并送到R3寄存器中 |
| 0000 1000 | 0011 0100 | OUT, R1, P(IOY1) | 输出R1寄存器中的数据 |
| 0000 1001 | 0100 0000 |
| 0000 1010 | 0011 1100 | OUT, R3, P(IOY1) | 输出R3寄存器中的数据 |
| 0000 1011 | 0100 0000 |

**说明：**通过IN指令，将数据传入寄存器中，然后通过逻辑运算单元ALU，进行数据的加减操作，最后再将数据传回寄存器中，然后输出寄存器中的数据。



**方案一的程序设计图**

**（2）输入的数据：**

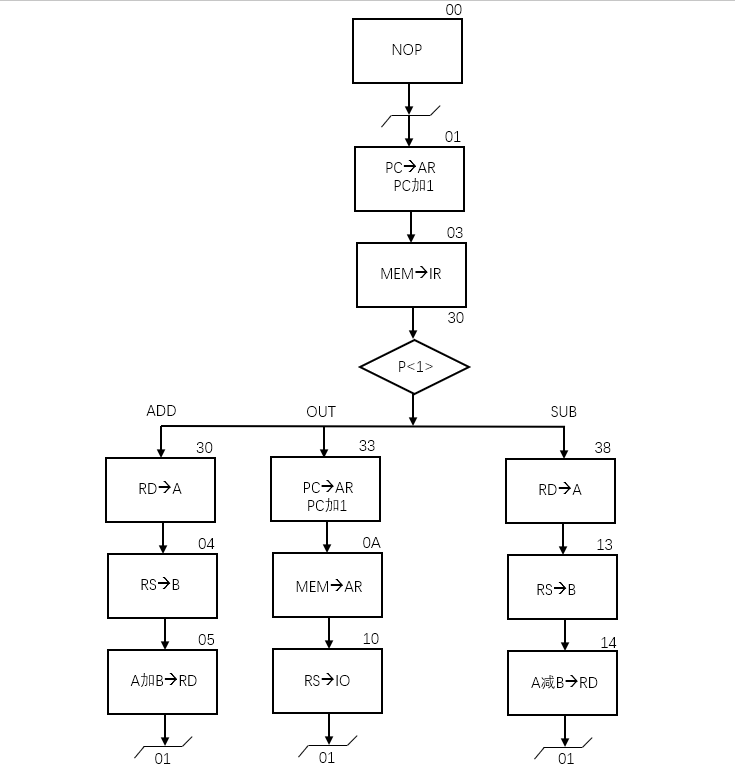
（94）16=（1001 0100）2 →R1

（08）16=（0000 1000）2 →R2

（17）16=（0001 0111）2 →R3

**（3）微程序设计及说明：**

1.微程序流程图



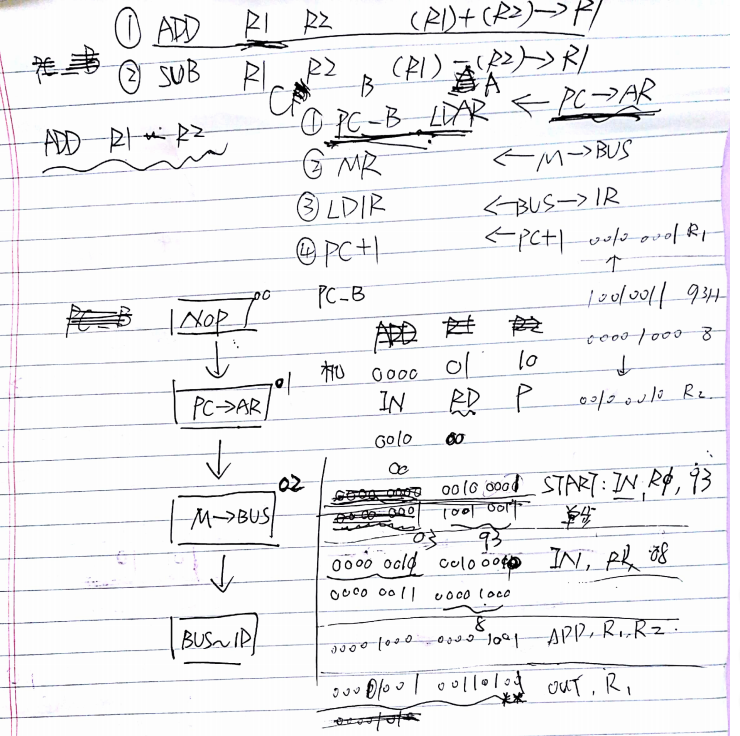
2.二进制代码表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 十六进制表示 | 高五位 | S3-S0 | A字段 | B字段 | C 字段 | UA5-UA0 |
| 00 | 00 00 01 | 00000 | 0000 | 000 | 000 | 000 | 000001 |
| 01 | 00 6D 43 | 00000 | 0000 | 110 | 110 | 101 | 000011 |
| 03 | 10 70 70 | 00010 | 0000 | 111 | 000 | 001 | 110000 |
| 04 | 00 24 05 | 00000 | 0000 | 010 | 010 | 000 | 000101 |
| 05 | 04 B2 01 | 00000 | 1001 | 011 | 001 | 000 | 000001 |
| 08 | 10 60 09 | 00010 | 0000 | 110 | 000 | 000 | 001001 |
| 09 | 18 30 01 | 00011 | 0000 | 011 | 000 | 000 | 000001 |
| 0A | 10 60 10 | 00010 | 0000 | 110 | 000 | 000 | 010000 |
| 10 | 28 04 01 | 00101 | 0000 | 000 | 010 | 000 | 000001 |
| 30 | 00 16 04 | 00000 | 0000 | 001 | 011 | 000 | 000100 |
| 32 | 00 6D 48 | 00000 | 0000 | 110 | 110 | 101 | 001000 |
| 33 | 00 6D 4A | 00000 | 0000 | 110 | 110 | 101 | 001010 |

**(4)调试过程及说明：**

**1.调试过程**

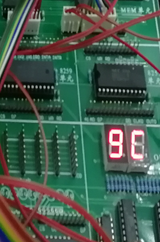
**在整个过程中，从开始写机器指令，到后面机器指令下的微指令，进行一步一步的调整，最后得出正确的程序和指令系统。**

****

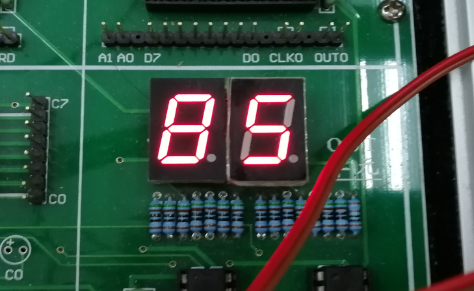
**从ADD，SUB再到IN，OUT指令，从机器指令到微指令，进行了一定的讨论。在刚开始的时候，是打算改变微指令的代码的，但后面的时候，发觉还是先做一份最基本，不改变微代码的方案，先熟悉整个流程。**

**刚开始熟悉的机器指令以及微指令，概念一直很混乱，后面渐渐知道，怎样去做一份指令了。**

**2.调试的结果及数码管显示结果照片：**

****

**加的结果：（94）16+（08）16=（9C）16**

****

**减的结果：（9C）16-(17)16=(85)16**

**方案二**：**编程实现16位数**X+Y**的计算，数X,Y存储在主存单元种，并显示输出的程序**

**（1）程序设计及说明：**

|  |  |  |  |
| --- | --- | --- | --- |
| 地址 | 内容 | 助记符 | 说明 |
| 0000 0000 | 1100 1000 | START: LAD, R0, R[I], 0EH | 从MEM读入数据送R0，变址寻址，偏移量为0EH |
| 0000 0001 | 0000 1110 |
| 0000 0010 | 1100 1001 | LAD, R1, R[I], 0FH | 从MEM读入数据送R1，变址寻址，偏移量为0FH |
| 0000 0011 | 0000 1111 |
| 0000 0100 | 1100 1010 | LAD, R2, R[I], 10H | 从MEM读入数据送R2，变址寻址，偏移量为10H |
| 0000 0101 | 0001 0001 |
| 0000 0110 | 1100 1011 | LAD, R3, R[I], 11H | 从MEM读入数据送R3，变址寻址，偏移量为11H |
| 0000 0111 | 0001 0001 |
| 0000 1000 | 0000 0001 | ADD, R0, R1 | 将R0和R1的数据相加，并送到R1寄存器中 |
| 0000 1001 | 0000 1011 | ADD, R2, R3 | 将R2和R3的数据相加，并送到R3寄存器中 |
| 0000 1010 | 0011 0100 | OUT, R1, P(IOY1) | 输出R1寄存器中的数据 |
| 0000 1011 | 0100 0000 |
| 0000 1100 | 0011 1100 | OUT, R3, P(IOY1) | 输出R3寄存器中的数据 |
| 0000 1101 | 0100 0000 |
| 0000 1110 | 0001 0111 | **数据** | |
| 0000 1111 | 1001 0100 |
| 0001 0000 | 0000 1000 |
| 0001 0001 | 0000 0111 |

**说明：**带进制的加法，由F=A加B加FC可得，将1001改为1010即可实现带进位加

地址 十六进制表示 高五位 S3-S0 A字段 B字段 C字段 UA5-UA0

**05 05 B2 01 00000 1010 011 001 000 000001**

**(带进位加)**

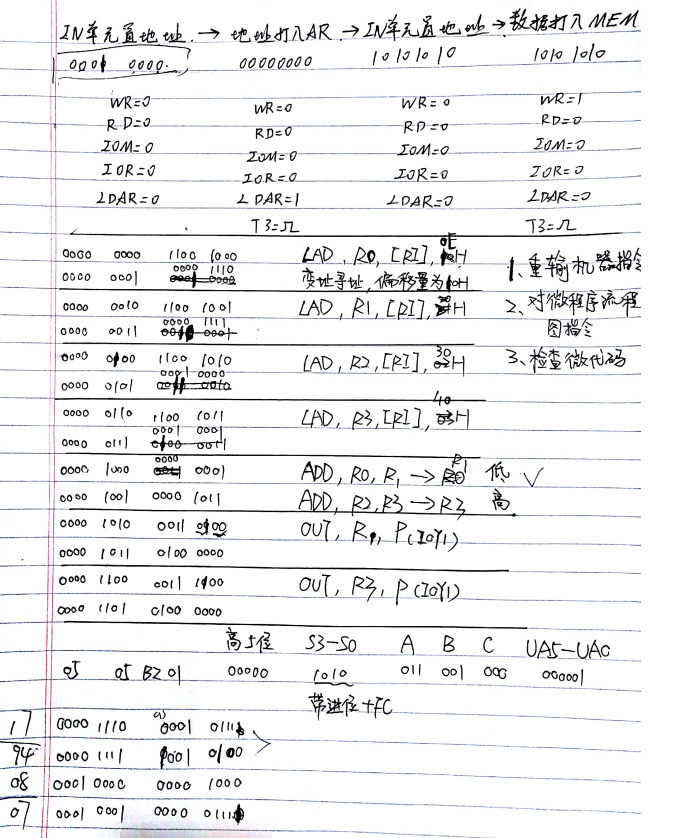
**（2）输入的数据：**

**（17）16=（0001 0111）2  0000 1110**

**（94）16=（1001 0100）2 0000 1111**

**（08）16=（0000 1000）2 0001 0000**

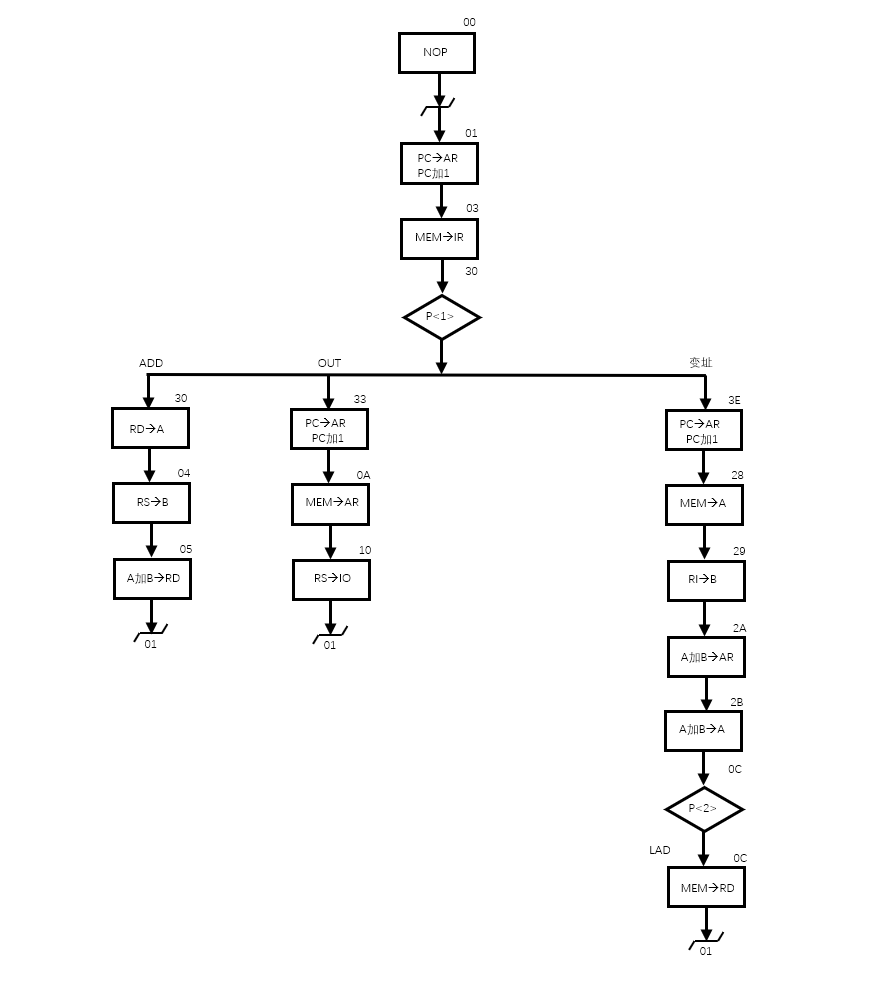
**（07）16=（0000 0111）2 0001 0111**



方案二的设计图

**（3）微程序设计及说明：**

1.微程序流程图

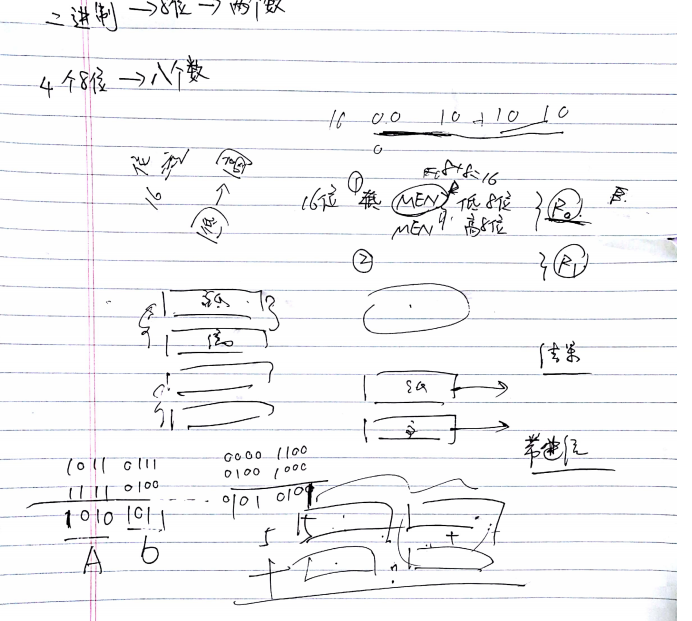


2.二进制代码表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 十六进制表示 | 高五位 | S3-S0 | A字段 | B字段 | C 字段 | UA5-UA0 |
| 00 | 00 00 01 | 00000 | 0000 | 000 | 000 | 000 | 000001 |
| 01 | 00 6D 43 | 00000 | 0000 | 110 | 110 | 101 | 000011 |
| 03 | 10 70 70 | 00010 | 0000 | 111 | 000 | 001 | 110000 |
| 04 | 00 24 05 | 00000 | 0000 | 010 | 010 | 000 | 000101 |
| 05 | 04 B2 01 | 00000 | 1001 | 011 | 001 | 000 | 000001 |
| 08 | 10 60 09 | 00010 | 0000 | 110 | 000 | 000 | 001001 |
| 09 | 18 30 01 | 00011 | 0000 | 011 | 000 | 000 | 000001 |
| 0A | 10 60 10 | 00010 | 0000 | 110 | 000 | 000 | 010000 |
| 0C | 10 30 01 | 00010 | 0000 | 011 | 000 | 000 | 000001 |
| 10 | 28 04 01 | 00101 | 0000 | 000 | 010 | 000 | 000001 |
| 13 | 00 24 14 | 00000 | 0000 | 010 | 010 | 000 | 010100 |
| 14 | 05 B2 01 | 00000 | 1011 | 011 | 001 | 000 | 000001 |
| 28 | 10 10 29 | 00010 | 0000 | 001 | 000 | 000 | 101001 |
| 29 | 00 28 2A | 00000 | 0000 | 010 | 100 | 000 | 101010 |
| 2A | 04 E2 2B | 00000 | 1001 | 110 | 001 | 000 | 101011 |
| 2B | 04 92 8C | 00000 | 1001 | 001 | 001 | 010 | 001100 |
| 30 | 00 16 04 | 00000 | 0000 | 001 | 011 | 000 | 000100 |
| 32 | 00 6D 48 | 00000 | 0000 | 110 | 110 | 101 | 001000 |
| 33 | 00 6D 4A | 00000 | 0000 | 110 | 110 | 101 | 001010 |
| 38 | 00 16 13 | 00000 | 0000 | 001 | 011 | 000 | 010011 |
| 3E | 00 6D 6C | 00000 | 0000 | 110 | 110 | 101 | 101000 |

**(4)调试过程及说明：**

**相比于第一个实验，我们对程序的设计更加的熟练了，在老师的指导下，解决了将数据存放在寄存器中的操作，解决了十六位数带进位加的操作。**

****

**2.调试的结果及数码管显示结果照片：**

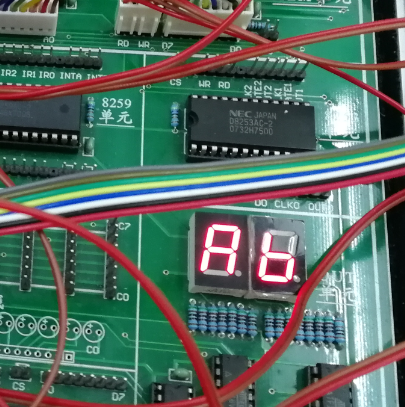
**（1）第一组数据（没有进位）：**

**（17）16=（0001 0111）2  0000 1110**

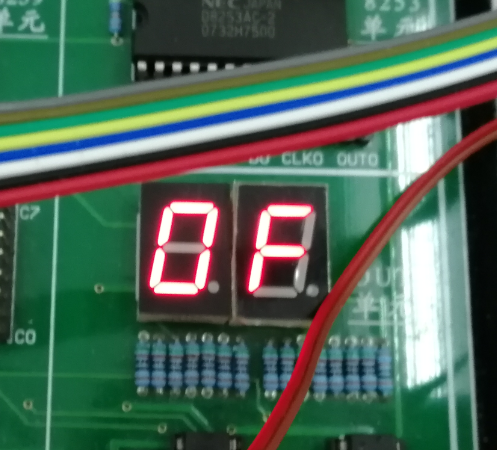
**（94）16=（1001 0100）2 0000 1111**

**（08）16=（0000 1000）2 0001 0000**

**（07）16=（0000 0111）2 0001 0111**

****

**低八位加的结果：（17）16+（94）16=（AB）16**

****

**高八位加的结果：（08）16+（07）16=（0F）16**

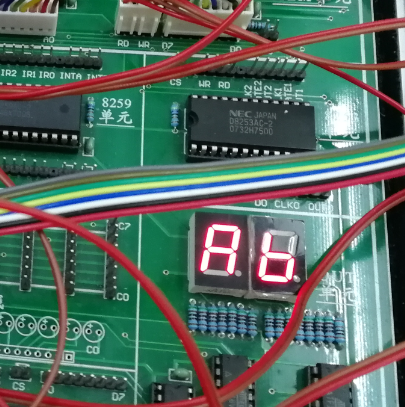
**（1）第二组数据（有进位）:**

**(B7)16 = (1011 0111)2**

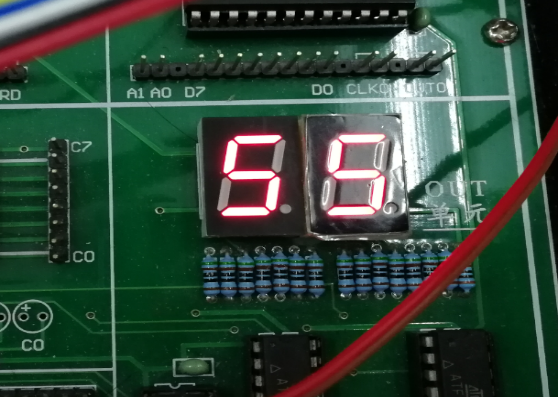
**(F8)16 = (1111 0100)2**

**(0C)16 = (0000 1100)2**

**(48)16 = (0100 1000)2**

****

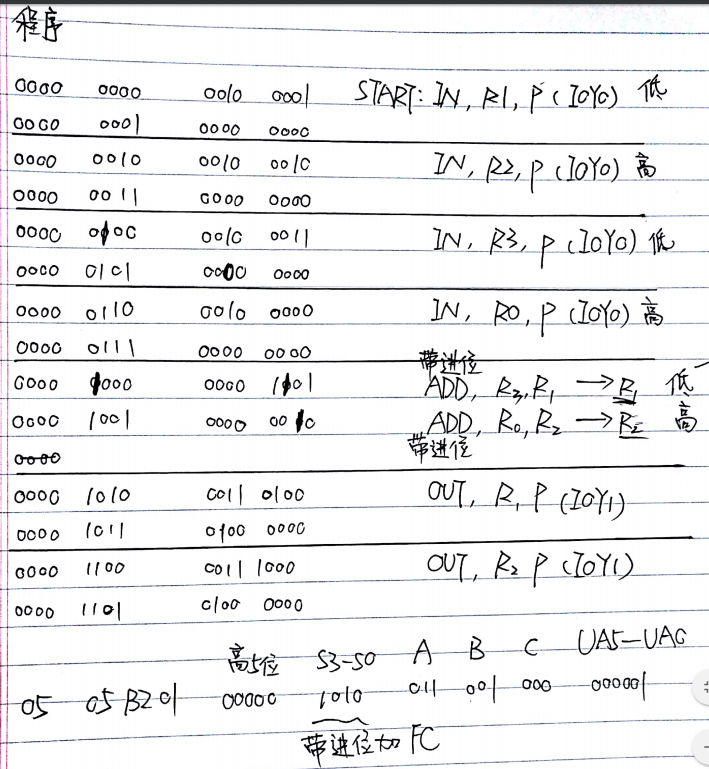
**低八位加的结果：（B7）16+（F8）16=（AB）16 （有进位）**

****

**高八位加的结果：（0C）16+（48）16=（55）16**

**方案三：用IN指令编程实现16位数**X+Y**的计算，并显示输出的程序**

**说明：**在进行方案二之前，我们首先实现了IN指令的16位数X+Y的计算，但因为考虑到没有存放在MEN单元中，达不到要求，所以重新设计了方案二，方案三有进行实践，但数据没有写下来，只有一页程序设计纸张。



**七、总结**

通过这次设计我学习了如何进行原码一位乘和原码一位除的计算机操作。经过实践我发现实际操作与学习书本知识还是有许多不同，只有加强实际训练才能更好的理解书本知识。

我想说本次课程设计不仅让我们复习了计算机组成原理的知识还让我们复习了算法设计与分析以及提前了解了汇编语言的一些内容。还增强了我们的动手能力，这是一次收获颇丰的课程设计。