**浙江理工大学**

《计算机组成原理》

课程设计报告

2021 ～2022 学年第2 学期

|  |  |
| --- | --- |
| 学 院 | 信息学院 |
| 班 级 | 20计算机科学与技术(4) |
| 姓 名 | 陈伟剑 |
| 学 号 | 2020333503081 |
| 任课教师 | 李俊松（上课时间：周三1-2、周四6-7 ） |

**一、目的和要求**

通过实践了解计算机各种指令的执行过程，以及控制器的组成，指令系统微程序设计的具体知识，进一步理解和掌握动态微程序设计的概念；完成自定义指令(微程序)从设计、编码到调试的过程。

**二、实验环境**

1. 电脑一台

2. TEC-2模拟软件

**三、理论内容**

1、分析TEC-2机的功能部件组成，分析TEC-2机支持的指令格式等。

**1、TEC-2的控制器基本组成**

(1)7片LS6116芯片组成的56位字长的微程序控制存储器。

用于存放TEC-2机的微程序。其内容在刚加电时不定，加电后将首先从2片ROM(LS2716芯片)中读出固化的、用于实现53条机器指令的做程序，经组织后写入这一控制存器，这一过程称为装入微码。装入完成后，将从监控程序的零地扯执行指令，完成TEC-2机的启动过程。这之后，还可以用LDMC指令按規定的办法向控制存俙器写入新的微程序，以实现新的机器指令。从简化逻辑框图上可以看到，控制存儲器的地址为µRA9～O，读出的信息送微指令流水线寄存器PLR。

(2)7片8位的寄存器芯片(6片LS374和1片LS273)组成微指令寄存器PLR。

用于存放当前做指令的内容

(3)微程序定序器AM2910

核心功能是依据机器的运行状态与当前微指令的有关内容等，正确地形成下一条微指令的地址，以保证微程序按要求的微指令序列关系自动地逐条衔接执行。

(4)程序计数器PC和当前指令地址记忆寄存器IP。

用运算器通用寄存器组中的两个选定的寄存器R5和R6实现。

(5)指令寄存器IR。

存放当前正在执行的指令内容

(6)AM2910输入相关的电路（三个输入通过互斥变量VECT、MAP、PL进行选择）：

6.1、2片LS2716ROM组成的MAPROM。它变指令寄存器中的操作码转换成一段微程序的入口地。

6.2、1片LS125和1片LS244组成的接收内部总线IB9-0选择门电路(跟手拨开关相关)

6.3、1片LS125和微指令寄存器PLR55-48组成的地址输入，把当前微指令中的后续地址B55-46送入AM2910的地址输入端。1片LS125(共4位独立的输入和输出端)分成两组(每组两位)分别用于②和③两项用途。

上面三组信号均为10位宽，且为互斥关系，分别由AM2910芯片提供的3个互斥控制信号/MAP、/VECT和/PL加以选通。

(7)AM2910输出相关的电路：

7.1、1片LS175、1片LS374寄存器组构成的记忆电路。用于保存当前微指令的地址

7.2、3片LS257芯片构成的微控存地址选择形成电路。它实现在AM2910的10位输出地址与存储器地址寄存器的低10位地址之间的选择，结果送往微控存的地址输入端，用于完成微控存单元的读写操作;选择信号是Smux。

7.3、3片LS161计数器芯片组成的地址计数器电路。其输出(共10位)通过两片LS244与刚提到的3片LS257的10位输出形成“线或＂关系，用计数器的一个输出端Y11实现二者之间的选择，Y11为0时，pRA9pRAO是计数器的输出信号，提供完成装入微码过程的微控存的地址，Y11为1时，表明装入微码的过程已结束，微控存的地址uRA9uRA0为3片LS257的输出信号，以完成机器指令运行过程中的微控存的读、写(写仅用于LDMC指令)操作。

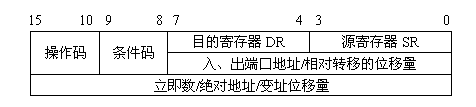
(8)由2片LS2716ROM芯片组成的、固化的微码保存电路及读写控制电路。这是为机器加电后完成装入微码所配备的专用线路，除2片LS2716外，还有前边提到的3片LS161芯片(计数器)，1片LS61芯片，2片LS244、1片Gal20v8、1片LS74、1片LS23和1片LS00.其连接关系在逻辑线路图(二)的右下部分。

**2、TEC-2机支持的指令格式**

TEC-2机指令系统，采用6位操作码，故最多支持64条基本指令，其中53条指令已由设计者实现，微程序固化在ROM芯片中，其余11条留给实验人员执行实现。

TEC-2机的基本指令格式比较固定。从指令长度区分，有单字指令和双字指令，用户也可以实现三字指令；从操作数的个数区分，有无操作数指令、单操作数指令和双操作数指令；从支持的基本寻址方式区分，有寄存器寻址、寄存器间接寻址、立即数寻址、变址寻址、相对寻址、绝对寻址和堆栈寻址等方式；从指令功能上看，最常用的指令类型和运算是比较齐全的。

为了方便理解和记忆，把TEC-2机的指令格式归纳为如下形式：



①单字指令仅用一个指令字。

②双字指令用两个指令字，此时第二个指令字的内容可能是立即数、一个绝对地址或一个变址位移量。

第一个指令字分为三个主要部分。最高6位是操作码。从这个意义上讲，TEC-2机的基本指令是固定长度的操作码结构。最多支持64条基本指令。

中间的两位，即9、8两位是条件码，只能把他用作条件转移指令的判断条件。这两位的值为00、01、10和11时，分别选择以处理机状态字中的C、Z、V和S的值作为判断条件。从这个意义上讲，我们也可以认为这两位是指令的扩展操作码。由于除了条件转移指令之外，其余的指令均不使用这两位，故我们可以用这两位扩展新的指令，而不会影响原来的指令功能。

最低的8位有多种用法：

（1）这8位用于给出入/出指令的入/出端口地址。例如，已规定第一个串行口地址为80h和81h，第二个串行口地址为82h和83h。并行口、DMA口等都可以被分配几个合理的地址，以实现入/出接口的各种实验。

（2）这8位用于给出相对寻址的位移量，其范围是从-128到+127之间，因此相对地址应在当前指令地址向前向后总共256个字的范围之内。实现相对地址计算时，这个位移量的最高位用符号位，补码形式，与16位的当前指令地址（放在IP寄存器中，即增量前的PC值）相加时，这一符号位要扩展到15~8这高8位上去。

（3）这8位被分为两个4位的字段，用于给出的所用的通用寄存器编号。对双操作数指令，这里可是给出目的与源两个操作数所在的寄存器编号。对单操作数指令，只用源或目的中的一个操作数，此时，可能用到某一个4位字段，另一个4位字段不用。需要强调的一点是，寄存器用于给出操作数、操作数地址，或用作变址寄存器。

**四、实践内容**

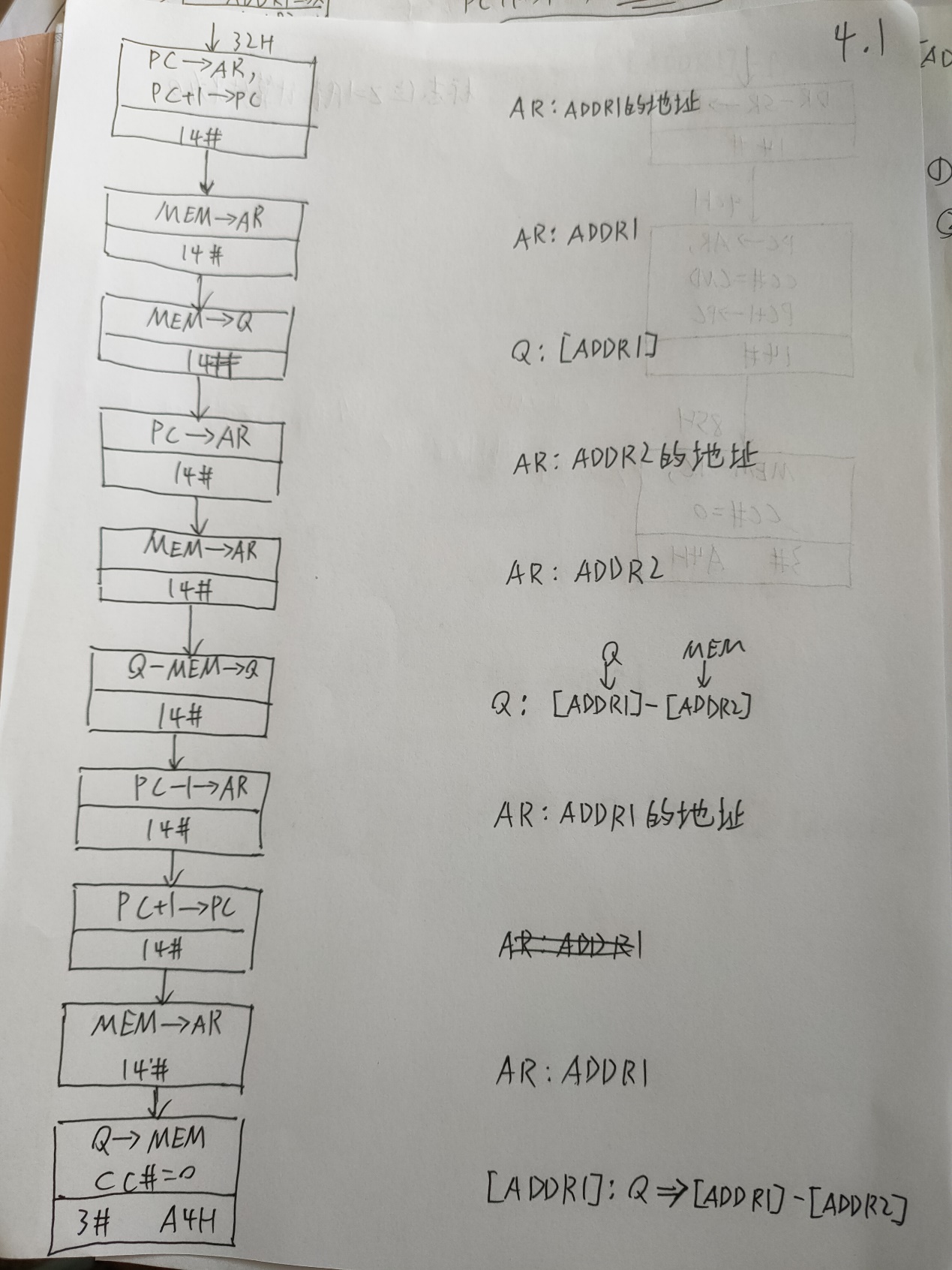
4.1 把用绝对地址表示的内存单元ADDR1中的内容与内存单元ADDR2中的内容相减，结果存于内存单元ADDR1中。

指令格式：D5××，ADDR1，ADDR2， 三字指令（控存入口100H）

功能： [ADDR1]=[ADDR1] -[ADDR2]

指令结构：

|  |  |
| --- | --- |
| 指令首地址 | D5XX |
| 指令首地址+1(PC当前指向值) | ADDR1 |
| 指令首地址+2 | ADDR2 |



微程序：

PC🡪AR，PC+1🡪PC 0000 0E00 A0B5 5402 //AR：ADDR1的地址

MEM🡪AR 0000 0E00 10F0 0002 //AR:ADDR1

MEM—>Q 0000 0E00 00F0 0000 //Q:[ADDR1]

PC🡪AR 0000 0E00 90B0 5002 //AR:ADDR2的地址

MEM🡪AR 0000 0E00 10F0 0002 //AR:ADDR2

Q-MEM🡪Q 0000 0E01 01E0 0000 //Q=[ADDR1](来自Q) - [ADDR2](MEM根据AR获取)

PC-1🡪AR 0000 0E00 91B0 5402 //AR:ADDR1的地址

PC+1🡪PC 0000 0E00 B0B0 5400

MEM🡪AR 0000 0E00 10F0 0002 //AR:ADDR1

Q🡪MEM，CC#=0 0029 0300 1020 0010 //[ADDR1]=Q=[ADDR1]-[ADDR2]

1. PC🡪AR，PC+1🡪PC 0000 0E00 A0B5 5402

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B55-B52 | B51-B48 | B47-B44 | B43-B40 | B39-B36 | B35-B32 | B31-B28 | B27-B24 | B23-B20 | B19-B16 | B15-B12 | B11-B8 | B7-B4 | B3-B0 |
| 0 | 0 | 0 | E | 0 | 0 | A | 0 | B | 5 | 5 | 4 | 0 | 2 |
| 0000 | 0000 | 0000 | 1110 | 0000 | 0000 | 1010 | 0000 | 1011 | 0101 | 0101 | 0100 | 0000 | 0010 |

B2-B0(DC2)：AR接收内容，设为010

B6-B4(DC1)：不向IB总线发送控制信号，设为000

B3(SB)：作AM2901，设为0

B7(SA)：作AM2901，设为0

B9-B8(SSH)：不移位，设为00

B11-B10(SCI)：自加1，设为01

B15-B12(B口)：使用R5寄存器(PC)，设为0101

B19-B16(A口)：使用R5寄存器(PC)，设为0101

B22-B20(MI2-0)：ALU的R端输入0，S端输入B，设为011

B26-B25(MI5-3)：ALU选择“S+R”功能，设为000

B30-B28(MI8-6)：ALU有返回值并传输到B，Y的输出选择A，设为010

B31、B27、B23(MIO、REQ、WE)：不操作数据总线，设为101

B34-B32(SST状态位)：不使用状态判断，设为000

B35：备用位，设为0

B36(SC)：/CC条件码始终低电平有效，设为0

B39-B37(SCC)：同上，设为000

B43-B40(CI3-CI0)：顺序执行(14#命令)，不选用下地址，设为1110

1. MEM🡪AR 0000 0E00 10F0 0002

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B55-B52 | B51-B48 | B47-B44 | B43-B40 | B39-B36 | B35-B32 | B31-B28 | B27-B24 | B23-B20 | B19-B16 | B15-B12 | B11-B8 | B7-B4 | B3-B0 |
| 0 | 0 | 0 | E | 0 | 0 | 1 | 0 | F | 0 | 0 | 0 | 0 | 2 |
| 0000 | 0000 | 0000 | 1110 | 0000 | 0000 | 0001 | 0000 | 1111 | 0000 | 0000 | 0000 | 0000 | 0010 |

B2-B0(DC2)：AR接收内容，设为010

B6-B4(DC1)：不向IB总线发送控制信号，设为000

B3(SB)：作AM2901，设为0

B7(SA)：作AM2901，设为0

B9-B8(SSH)：不移位，设为00

B11-B10(SCI)：不进位，设为00

B15-B12(B口)：不使用寄存器，设为0000

B19-B16(A口)：不使用寄存器，设为0000

B22-B20(MI2-0)：ALU的R端输入D，S端输入0，设为111

B26-B25(MI5-3)：ALU选择“S+R”功能，设为000

B30-B28(MI8-6)：ALU无返回值，Y的输出选择F，设为001

B31、B27、B23(MIO、REQ、WE)：从存储器读内容，设为001

B34-B32(SST状态位)：不使用状态判断，设为000

B35：备用位，设为0

B36(SC)：/CC条件码始终低电平有效，设为0

B39-B37(SCC)：同上，设为000

B43-B40(CI3-CI0)：顺序执行(14#命令)，不选用下地址，设为1110

1. MEM—>Q 0000 0E00 00F0 0000

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B55-B52 | B51-B48 | B47-B44 | B43-B40 | B39-B36 | B35-B32 | B31-B28 | B27-B24 | B23-B20 | B19-B16 | B15-B12 | B11-B8 | B7-B4 | B3-B0 |
| 0 | 0 | 0 | E | 0 | 0 | 0 | 0 | F | 0 | 0 | 0 | 0 | 0 |
| 0000 | 0000 | 0000 | 1110 | 0000 | 0000 | 0000 | 0000 | 1111 | 0000 | 0000 | 0000 | 0000 | 0000 |

B2-B0(DC2)：寄存器不接收内容，设为000

B6-B4(DC1)：不向IB总线发送控制信号，设为000

B3(SB)：作AM2901，设为0

B7(SA)：作AM2901，设为0

B9-B8(SSH)：不移位，设为00

B11-B10(SCI)：不进位，设为00

B15-B12(B口)：不使用寄存器，设为0000

B19-B16(A口)：不使用寄存器，设为0000

B22-B20(MI2-0)：ALU的R端输入D，S端输入0，设为111

B26-B25(MI5-3)：ALU选择“S+R”功能，设为000

B30-B28(MI8-6)：ALU有返回值，输出到Q，Y的输出选择F，设为000

B31、B27、B23(MIO、REQ、WE)：从存储器读内容，设为001

B34-B32(SST状态位)：不使用状态判断，设为000

B35：备用位，设为0

B36(SC)：/CC条件码始终低电平有效，设为0

B39-B37(SCC)：同上，设为000

B43-B40(CI3-CI0)：顺序执行(14#命令)，不选用下地址，设为1110

1. PC🡪AR 0000 0E00 90B0 5002

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B55-B52 | B51-B48 | B47-B44 | B43-B40 | B39-B36 | B35-B32 | B31-B28 | B27-B24 | B23-B20 | B19-B16 | B15-B12 | B11-B8 | B7-B4 | B3-B0 |
| 0 | 0 | 0 | E | 0 | 0 | 9 | 0 | B | 0 | 5 | 0 | 0 | 2 |
| 0000 | 0000 | 0000 | 1110 | 0000 | 0000 | 1001 | 0000 | 1011 | 0000 | 0101 | 0000 | 0000 | 0010 |

B2-B0(DC2)：AR接收内容，设为010

B6-B4(DC1)：不向IB总线发送控制信号，设为000

B3(SB)：作AM2901，设为0

B7(SA)：作AM2901，设为0

B9-B8(SSH)：不移位，设为00

B11-B10(SCI)：不进位，设为00

B15-B12(B口)：使用R5寄存器(PC)，设为0101

B19-B16(A口)：不使用寄存器，设为0000

B22-B20(MI2-0)：ALU的R端输入0，S端输入B，设为011

B26-B25(MI5-3)：ALU选择“S+R”功能，设为000

B30-B28(MI8-6)：ALU无返回值，Y的输出选择F，设为001

B31、B27、B23(MIO、REQ、WE)：不操作，设为101

B34-B32(SST状态位)：不使用状态判断，设为000

B35：备用位，设为0

B36(SC)：/CC条件码始终低电平有效，设为0

B39-B37(SCC)：同上，设为000

B43-B40(CI3-CI0)：顺序执行(14#命令)，不选用下地址，设为1110

1. MEM🡪AR同②
2. Q-MEM🡪Q 0000 0E01 01E0 0000

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B55-B52 | B51-B48 | B47-B44 | B43-B40 | B39-B36 | B35-B32 | B31-B28 | B27-B24 | B23-B20 | B19-B16 | B15-B12 | B11-B8 | B7-B4 | B3-B0 |
| 0 | 0 | 0 | E | 0 | 1 | 0 | 1 | E | 0 | 0 | 0 | 0 | 0 |
| 0000 | 0000 | 0000 | 1110 | 0000 | 0001 | 0000 | 0001 | 1110 | 0000 | 0000 | 0000 | 0000 | 0000 |

B2-B0(DC2)：寄存器不接收内容，设为000

B6-B4(DC1)：不向IB总线发送控制信号，设为000

B3(SB)：作AM2901，设为0

B7(SA)：作AM2901，设为0

B9-B8(SSH)：不移位，设为00

B11-B10(SCI)：不进位，设为00

B15-B12(B口)：不使用寄存器，设为0000

B19-B16(A口)：不使用寄存器，设为0000

B22-B20(MI2-0)：ALU的R端输入D，S端输入Q，设为110

B26-B25(MI5-3)：ALU选择“S-R”功能，设为001

B30-B28(MI8-6)：ALU有返回值，输出到Q，Y的输出选择F，设为000

B31、B27、B23(MIO、REQ、WE)：从存储器读内容，设为001

B34-B32(SST状态位)：不使用状态判断，设为000

B35：备用位，设为0

B36(SC)：/CC条件码始终低电平有效，设为0

B39-B37(SCC)：同上，设为000

B43-B40(CI3-CI0)：顺序执行(14#命令)，不选用下地址，设为1110

1. PC-1🡪AR 0000 0E00 91B0 5402

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B55-B52 | B51-B48 | B47-B44 | B43-B40 | B39-B36 | B35-B32 | B31-B28 | B27-B24 | B23-B20 | B19-B16 | B15-B12 | B11-B8 | B7-B4 | B3-B0 |
| 0 | 0 | 0 | E | 0 | 0 | 9 | 1 | B | 0 | 5 | 4 | 0 | 2 |
| 0000 | 0000 | 0000 | 1110 | 0000 | 0000 | 1001 | 0001 | 1011 | 0000 | 0101 | 0100 | 0000 | 0010 |

B2-B0(DC2)：AR接收内容，设为010

B6-B4(DC1)：不向IB总线发送控制信号，设为000

B3(SB)：作AM2901，设为0

B7(SA)：作AM2901，设为0

B9-B8(SSH)：不移位，设为00

B11-B10(SCI)：进位1(用于减法)，设为01

B15-B12(B口)：使用R5寄存器(PC)，设为0101

B19-B16(A口)：不使用寄存器，设为0000

B22-B20(MI2-0)：ALU的R端输入0，S端输入B，设为011

B26-B25(MI5-3)：ALU选择“S-R”功能，设为001

B30-B28(MI8-6)：ALU无返回值，Y的输出选择F，设为001

B31、B27、B23(MIO、REQ、WE)：不操作，设为101

B34-B32(SST状态位)：不使用状态判断，设为000

B35：备用位，设为0

B36(SC)：/CC条件码始终低电平有效，设为0

B39-B37(SCC)：同上，设为000

B43-B40(CI3-CI0)：顺序执行(14#命令)，不选用下地址，设为1110

1. PC+1🡪PC 0000 0E00 B0B0 5400

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B55-B52 | B51-B48 | B47-B44 | B43-B40 | B39-B36 | B35-B32 | B31-B28 | B27-B24 | B23-B20 | B19-B16 | B15-B12 | B11-B8 | B7-B4 | B3-B0 |
| 0 | 0 | 0 | E | 0 | 0 | B | 0 | B | 0 | 5 | 4 | 0 | 0 |
| 0000 | 0000 | 0000 | 1110 | 0000 | 0000 | 1011 | 0000 | 1011 | 0000 | 0101 | 0100 | 0000 | 0000 |

B2-B0(DC2)：寄存器不接收内容，设为000

B6-B4(DC1)：不向IB总线发送控制信号，设为000

B3(SB)：作AM2901，设为0

B7(SA)：作AM2901，设为0

B9-B8(SSH)：不移位，设为00

B11-B10(SCI)：自加1，设为01

B15-B12(B口)：使用R5寄存器(PC)，设为0101

B19-B16(A口)：不使用寄存器，设为0000

B22-B20(MI2-0)：ALU的R端输入0，S端输入B，设为011

B26-B25(MI5-3)：ALU选择“S+R”功能，设为000

B30-B28(MI8-6)：ALU有返回值，返回至B，Y的输出选择F，设为011

B31、B27、B23(MIO、REQ、WE)：不操作，设为101

B34-B32(SST状态位)：不使用状态判断，设为000

B35：备用位，设为0

B36(SC)：/CC条件码始终低电平有效，设为0

B39-B37(SCC)：同上，设为000

B43-B40(CI3-CI0)：顺序执行(14#命令)，不选用下地址，设为1110

1. MEM🡪AR 同②
2. Q🡪MEM，CC#=0 0029 0300 1020 0010

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B55-B52 | B51-B48 | B47-B44 | B43-B40 | B39-B36 | B35-B32 | B31-B28 | B27-B24 | B23-B20 | B19-B16 | B15-B12 | B11-B8 | B7-B4 | B3-B0 |
| 2 | 9 | 0 | 3 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 1 | 0 |
| 0010 | 1001 | 0000 | 0011 | 0000 | 0000 | 0001 | 0000 | 0010 | 0000 | 0000 | 0000 | 0001 | 0000 |

B2-B0(DC2)：寄存器不接收内容，设为000

B6-B4(DC1)：打开ALU和主存之间的路，设为001

B3(SB)：作AM2901，设为0

B7(SA)：作AM2901，设为0

B9-B8(SSH)：不移位，设为00

B11-B10(SCI)：不进位，设为00

B15-B12(B口)：不使用寄存器，设为0000

B19-B16(A口)：不使用寄存器，设为0000

B22-B20(MI2-0)：ALU的R端输入0，S端输入Q，设为010

B26-B25(MI5-3)：ALU选择“S+R”功能，设为000

B30-B28(MI8-6)：ALU无返回值，Y的输出选择F，设为001

B31、B27、B23(MIO、REQ、WE)：存储器写，设为000

B34-B32(SST状态位)：不使用状态判断，设为000

B35：备用位，设为0

B36(SC)：/CC条件码始终低电平有效，设为0

B39-B37(SCC)：同上，设为000

B43-B40(CI3-CI0)：条件转移(3#命令)，设为011

B55-B44(下地址)：这是该指令的最后一条微命令，执行完后要跳转到A4H，其下地址为29h

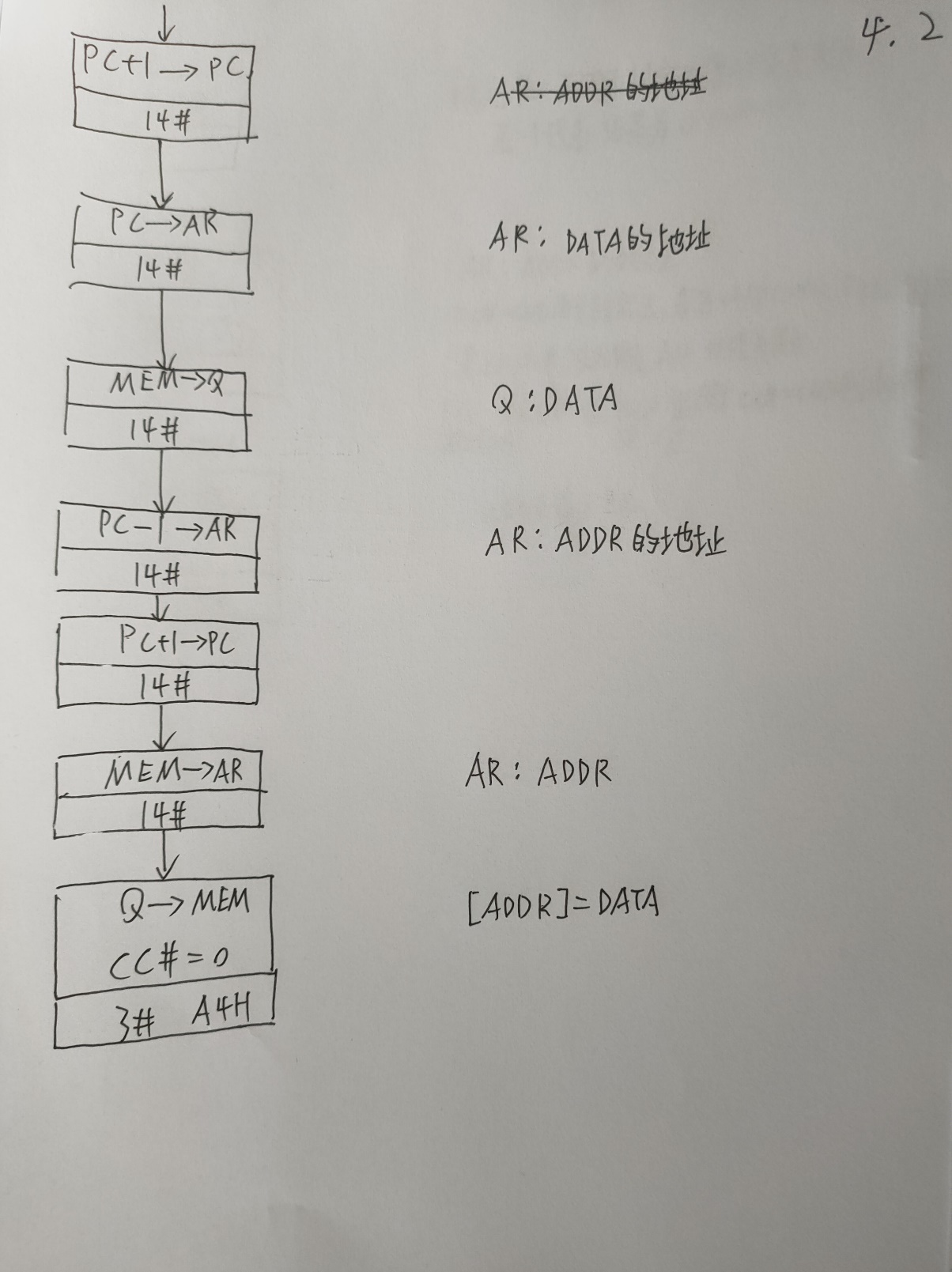
* 1. 把立即数DATA传送至地址为ADDR的内存单元中保存。

指令格式：D8××，ADDR，DATA， 三字指令（控存入口110H）

功能： [ADDR]←DATA

指令结构：

|  |  |
| --- | --- |
| 指令首地址 | D8XX |
| 指令首地址+1(PC当前指向值) | ADDR |
| 指令首地址+2 | DATA |



微程序：

PC+1🡪PC 0000 0E00 B0B0 5400

PC🡪AR 0000 0E00 90B0 5002 //AR:DATA的地址

MEM🡪Q 0000 0E00 00F0 0000 //Q:DATA

PC-1🡪AR 0000 0E00 91B0 5402 //AR:ADDR的地址

MEM🡪AR 0000 0E00 10F0 0002 //AR:ADDR

PC+1🡪PC 0000 0E00 B0B0 5400

Q🡪MEM，CC#=0 0029 0300 1020 0010 //[ADDR]=DATA

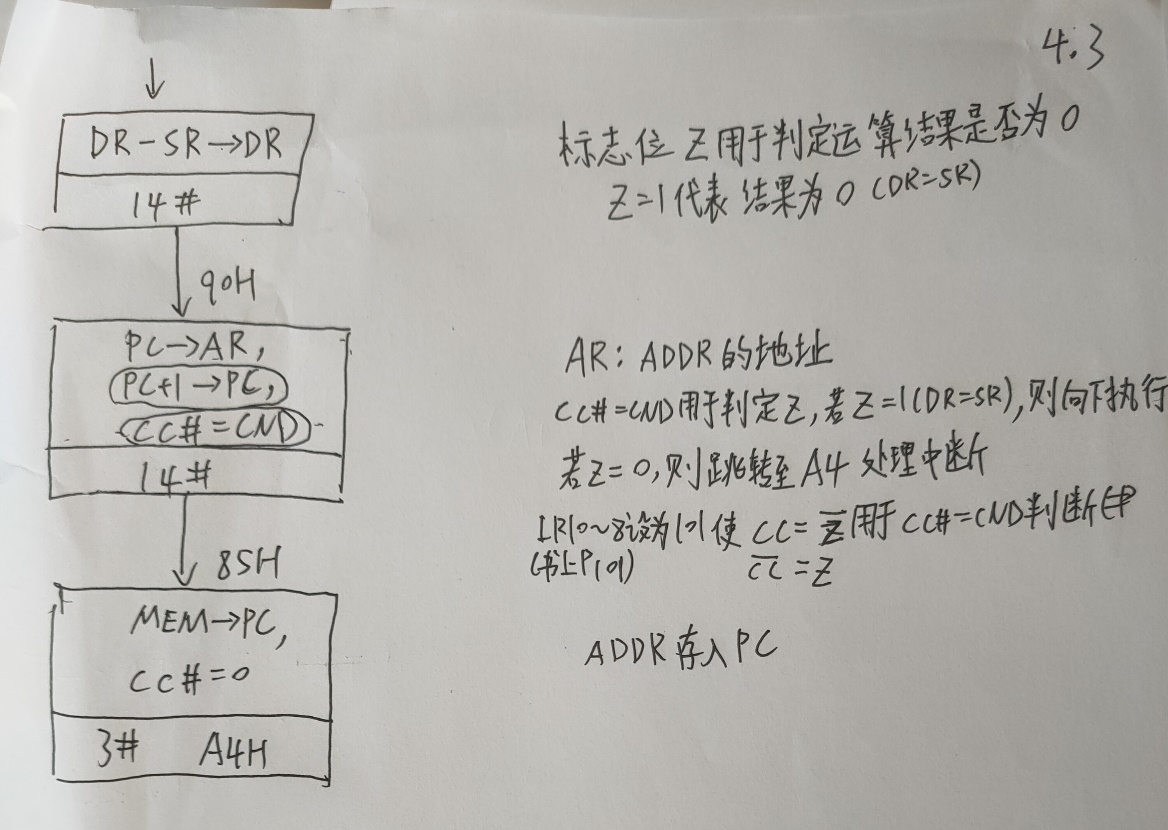
1. PC+1🡪PC 同4.1第⑧步
2. PC🡪AR 同4.1第④步
3. MEM🡪Q 同4.1第③步
4. PC-1🡪AR 同4.1第⑦步
5. MEM🡪AR 同4.1第②步
6. PC+1🡪PC 同①
7. Q🡪MEM，CC#=0 同4.1第⑩步

4.3 转移指令。判断两个通用寄存器内容是否相等，若不相等则转移到指定目的地址，否则顺序执行。

指令格式：E1 DR SR，ADDR 双字指令（控存入口130H, ADDR为绝对转移地址）

功能： if DR!=SR goto ADDR else　顺序执行。

|  |  |
| --- | --- |
| 指令首地址 | E1XX |
| 指令首地址+1(PC当前指向值) | ADDR |



**微程序：**

DR-SR🡪DR 0000 0E01 9190 0088

PC🡪AR，PC+1🡪PC，CC#=CND 0029 03E0 A0B5 5402

MEM🡪PC，CC#=0 0029 0300 20F0 5000

第二步需要使用到CND字段（见附录1），CND所占的3个二进制位从IR10~IR8中获取。通过查表2.6(P101)，在SCC=111且SC=0的前提下，令IR10~IR8=001（操作码见测试第5步），可以使 ，当Z=1时，=0可以进行跳转。

**编码：（下面的表述中默认省略前8位）**

1. DR-SR🡪DR 0000 0E01 B190 0088

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B55-B52 | B51-B48 | B47-B44 | B43-B40 | B39-B36 | B35-B32 | B31-B28 | B27-B24 | B23-B20 | B19-B16 | B15-B12 | B11-B8 | B7-B4 | B3-B0 |
| 0 | 0 | 0 | E | 0 | 1 | 9 | 1 | 9 | 0 | 0 | 0 | 8 | 8 |
| 0000 | 0000 | 0000 | 1110 | 0000 | 0001 | 1001 | 0001 | 1001 | 0000 | 0000 | 0000 | 1000 | 1000 |

B2-B0(DC2)：寄存器不接收内容，设为000

B6-B4(DC1)：不向IB总线发送控制信号，设为000

B3(SB)：用指令给ALU 的S端赋值，设为1

B7(SA)：用指令给ALU 的R端赋值，设为1

B9-B8(SSH)：不移位，设为00

B11-B10(SCI)：不进位，设为00

B15-B12(B口)：不使用寄存器，设为0000

B19-B16(A口)：不使用寄存器，设为0000

B22-B20(MI2-0)：ALU的R端输入A，S端输入B，设为001

B26-B25(MI5-3)：ALU选择“S-R”功能，设为001

B30-B28(MI8-6)：ALU无返回值，Y的输出选择F，设为001

B31、B27、B23(MIO、REQ、WE)：不操作数据总线，设为101

B34-B32(SST状态位)：保存运算器的状态，设为001

B35：备用位，设为0

B36(SC)：/CC条件码始终低电平有效，设为0

B39-37(SCC)：同上，设为000

B43-40(CI3-CI0)：顺序执行(14#命令)，不选用下地址，设为1110

1. PC🡪AR， PC+1🡪PC，CC#=CND 0029 03E0 A0B5 5402

CC#=CND用于判定Z，若Z=1(DR=SR)，则向下执行，否则跳转至A4处理中断

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B55-B52 | B51-B48 | B47-B44 | B43-B40 | B39-B36 | B35-B32 | B31-B28 | B27-B24 | B23-B20 | B19-B16 | B15-B12 | B11-B8 | B7-B4 | B3-B0 |
| 2 | 9 | 0 | 3 | E | 0 | A | 0 | B | 5 | 5 | 4 | 0 | 2 |
| 0010 | 1001 | 0000 | 0011 | 1110 | 0000 | 1010 | 0000 | 1011 | 0101 | 0101 | 0100 | 0000 | 0010 |

B2-B0(DC2)：AR寄存器接收内容，设为010

B6-B4(DC1)：不向IB总线发送控制信号，设为000

B3(SB)：作AM2901，设为0

B7(SA)：作AM2901，设为0

B9-B8(SSH)：不移位，设为00

B11-B10(SCI)：自加1，设为01

B15-B12(B口)：使用R5寄存器(PC)，设为0101

B19-B16(A口)：使用R5寄存器(PC)，设为0101

B22-B20(MI2-0)：ALU的R端输入0，S端输入B，设为011

B26-B25(MI5-3)：ALU选择“S+R”功能，设为000

B30-B28(MI8-6)：ALU有返回值并传输到B，Y的输出选择A，设为010

B31、B27、B23(MIO、REQ、WE)：不操作数据总线，设为101

B34-B32(SST状态位)：不使用状态判断，设为000

B35：备用位，设为0

B36(SC)：需要用到IR10-8，设为0

B39-37(SCC)：需要用到IR10-8，设为111

B43-40(CI3-CI0)：需要条件转移(3#命令)，设为0011

B55-B44(下地址)：判定是否相等，是则顺序执行，否则跳转到A4H，其下地址为29

1. MEM🡪PC，CC#=0 0029 0300 20F0 5000

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. B55-B52 | B51-B48 | B47-B44 | B43-B40 | B39-B36 | B35-B32 | B31-B28 | B27-B24 | B23-B20 | B19-B16 | B15-B12 | B11-B8 | B7-B4 | B3-B0 |
| 2 | 9 | 0 | 3 | 0 | 0 | 2 | 0 | F | 0 | 5 | 0 | 0 | 0 |
| 0010 | 1001 | 0000 | 0011 | 0000 | 0000 | 0010 | 0000 | 1111 | 0000 | 0101 | 0000 | 0000 | 0000 |

B2-B0(DC2)：寄存器不接收内容，设为000

B6-B4(DC1)：不向IB总线发送控制信号，设为000

B3(SB)：用指令给ALU 的S端赋值，设为1

B7(SA)：用指令给ALU 的R端赋值，设为1

B9-B8(SSH)：不移位，设为00

B11-B10(SCI)：不进位，设为00

B15-B12(B口)：使用R5寄存器(PC)，设为0101

B19-B16(A口)：不使用寄存器，设为0000

B22-B20(MI2-0)：ALU的R端输入D，S端输入0，设为111

B26-B25(MI5-3)：ALU选择“S+R”功能，设为000

B30-B28(MI8-6)：ALU有返回值并传输到B，Y的输出选择A，设为010

B31、B27、B23(MIO、REQ、WE)：不操作数据总线，设为101

B34-B32(SST状态位)：保存运算器的状态，设为001

B35：备用位，设为0

B36(SC)：/CC条件码始终低电平有效，设为0

B39-37(SCC)：同上，设为000

B43-40(CI3-CI0)：条件转移(3#命令)，设为011

B55-B44(下地址)：这是该指令的最后一条微命令，执行完后要跳转到A4H，其下地址为29h

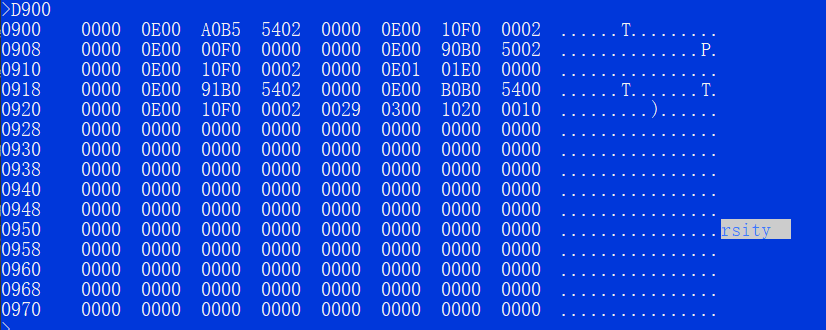
**五、测试**

5.1 把用绝对地址表示的内存单元ADDR1中的内容与内存单元ADDR2中的内容相减，结果存于内存单元ADDR1中。

(1)用E命令输入微码(E900，微码在内存中的首地址设置为900)



(2)用D命令检查输入的微码



(3)用A命令输入加载微码的程序，将微码加载到微控存中（TEC-2机主存储器中RAM地址分配为0800H-0FFFH）。并用G命令运行。

>A800↙

MOV R1,900↙ //微码在内存中的首地址

MOV R2,10↙ //有10条微命令

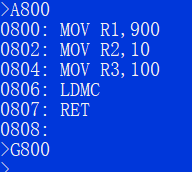
MOV R3.100↙ //微码在微控存中的首地址，题目指定入口为100H

LDMC↙ //加载微码

RET↙ //返回

↙

>G800↙



(4)用A命令输入测试程序

>A840

MOV R0,EB5A↙

MOV R1,5834↙

MOV [A00],R0↙

MOV [A01],R1↙

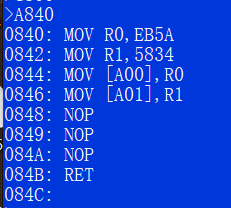
NOP↙ //指令占3个字节，用三个NOP占位

NOP↙

NOP↙

RET↙

↙



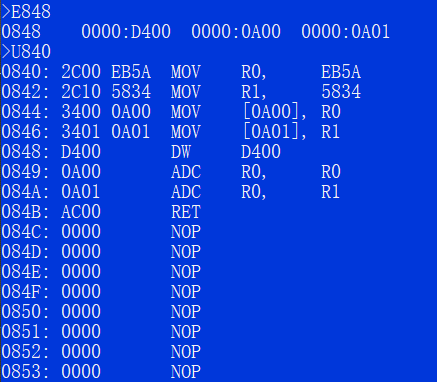
(5)用E命令输入自定义的指令

E848

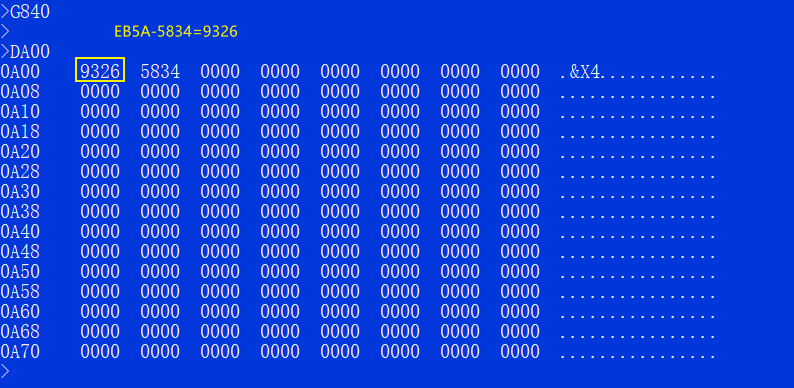
依次输入D400 0A00 0A01



(6)用U命令查看装入新指令后的程序

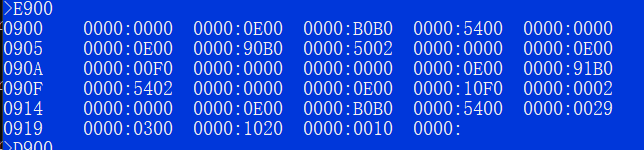


(7)用G命令运行程序，然后用D命令查看运行成果

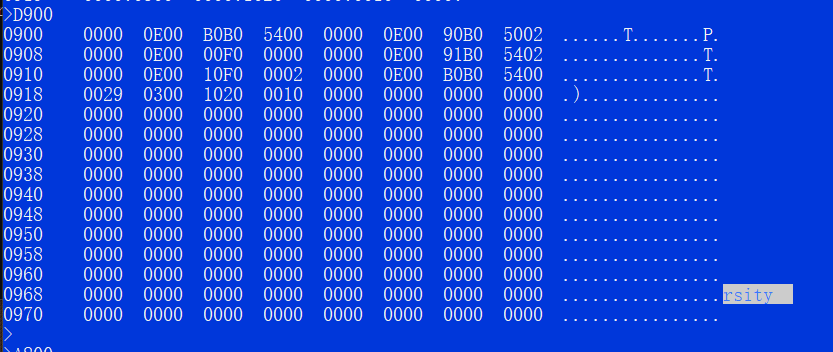


5.2把立即数DATA传送至地址为ADDR的内存单元中保存。

(1)用E命令输入微码(E900，微码在内存中的首地址设置为900)



(2)用D命令检查输入的微码



(3)用A命令输入加载微码的程序，将微码加载到微控存中（TEC-2机主存储器中RAM地址分配为0800H-0FFFH）。并用G命令运行。

>A800↙

MOV R1,900↙ //微码在内存中的首地址

MOV R2,7↙ //有7条微命令

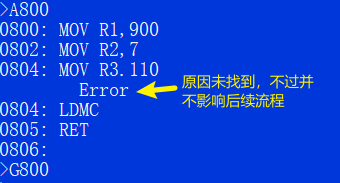
MOV R3.110↙ //微码在微控存中的首地址，题目指定入口为110H

LDMC↙ //加载微码

RET↙ //返回

↙

>G800↙



(4)用A命令输入测试程序

>A840

MOV R0,7777↙

MOV [A00],R0↙

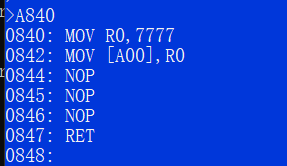
NOP↙ //指令占3个字节，用三个NOP占位

NOP↙

NOP↙

RET↙

↙



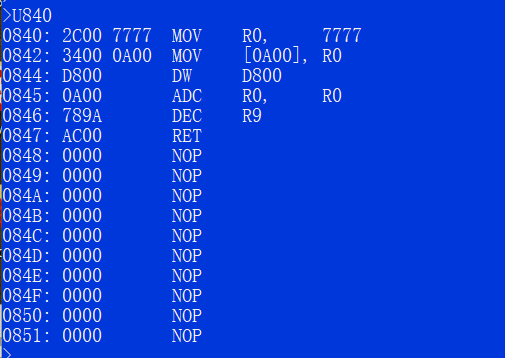
(5)用E命令输入自定义的指令

E844

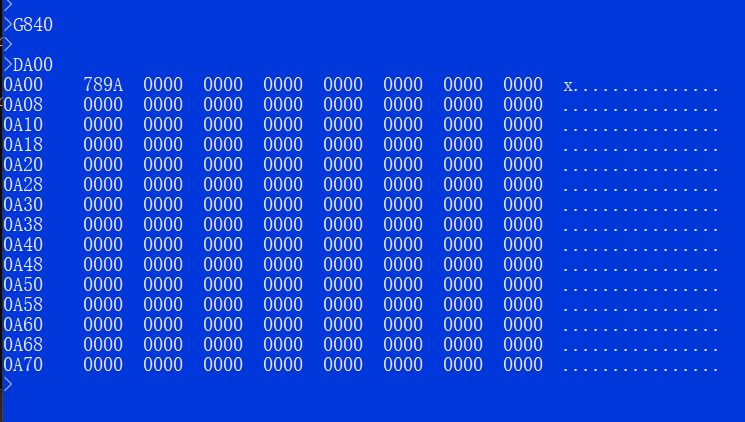
依次输入D800 0A00 789A



(6)用U命令查看装入新指令后的程序

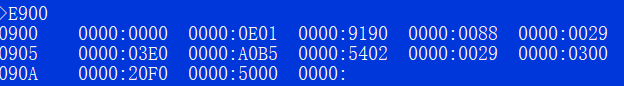


(7)用G命令运行程序，然后用D命令查看运行成果

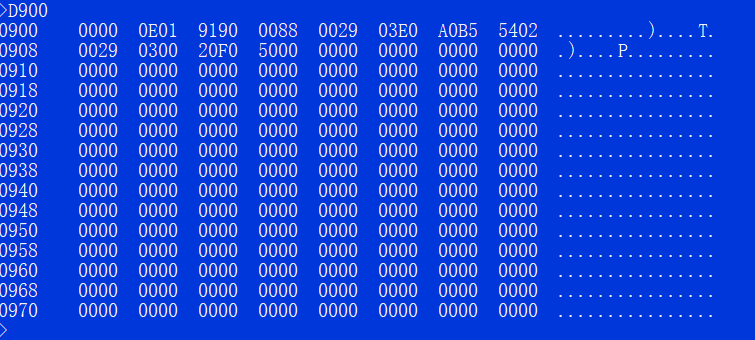


5.3转移指令。判断两个通用寄存器内容是否相等，若不相等则转移到指定目的地址，否则顺序执行。

(1)用E命令输入微码(E900，微码在内存中的首地址设置为900)



(2)用D命令检查输入的微码



(3)用A命令输入加载微码的程序，将微码加载到微控存中（TEC-2机主存储器中RAM地址分配为0800H-0FFFH）。并用G命令运行。

>A800↙

MOV R1,900↙ //微码在内存中的首地址

MOV R2,3↙ //有7条微命令

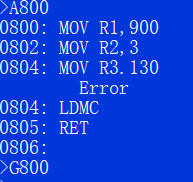
MOV R3.130↙ //微码在微控存中的首地址，题目指定入口为140H

LDMC↙ //加载微码

RET↙ //返回

↙

>G800↙



(4)用A命令输入测试程序

>A820

MOV R1,1111↙

MOV E2,2222↙

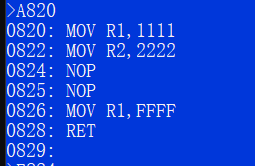
NOP↙ //指令占2个字节

NOP↙

MOV R1,FFFF↙

RET↙

↙



(5)用E命令输入自定义的指令

E824

依次输入E112 0828

对照附录1，E1的1对应0001，即IR10~IR8=1，再对照P101的CC设置表，5对应 。

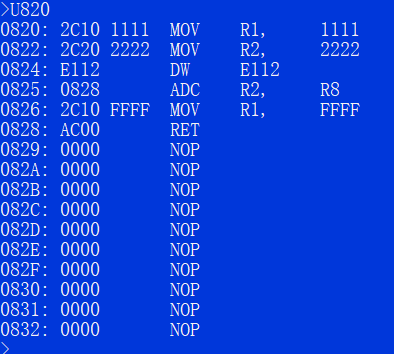
E112中后两位，1表示将R1视为DR，2表示将R2视为SR。



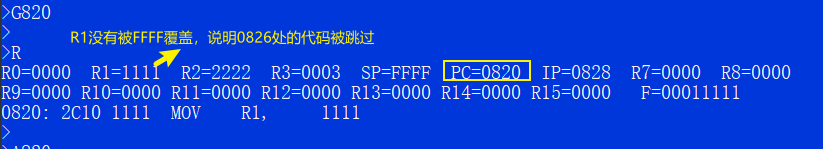
**若DR!=SR，则会直接跳转到0828，对应测试程序的RET，MOV R1,0023会被省略**

**反之，R1的内容会被FFFF覆盖**

(6)用U命令查看装入新指令后的程序

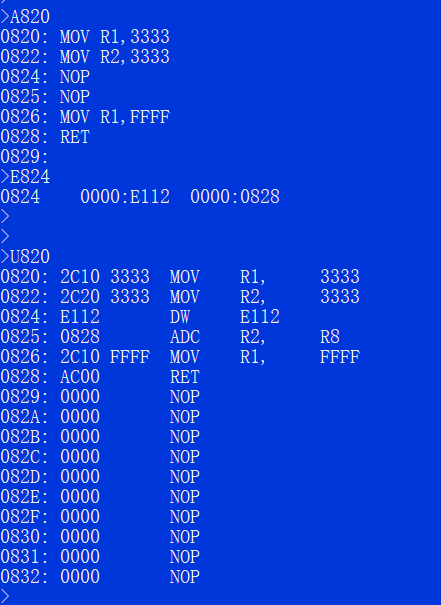


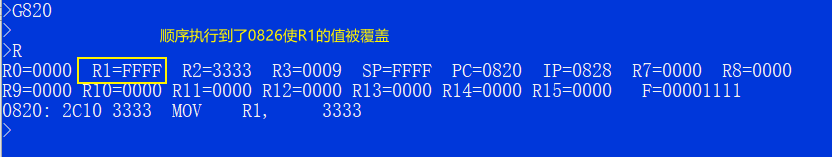
(7)用G命令运行程序，然后用R命令查看运行成果



(8)修改A820处的内容重新测试

这次DR=SR，R1会被FFFF覆盖（即0826处的代码不会被跳过）





**六、心得与体会**

通过本次课程设计，我基本熟悉了TEC-2机微程序从设计到使用的过程。要实现一个自定义指令，本质就是写一段可以调用的微程序(跟函数调用差不多)，微程序由微命令构成，TEC-2机的微命令固定为64位，通过对64位微命令的设计可以实现控制TEC-2机的各个部分。虽然一些常见的微命令可以通过查表解决，但是要深入理解某个微命令为什么要这样编码，就需要先了解：①TEC-2机的各个部件(运算器am2901、控制器am2910、存储器、总线等)的功能。②了解这些部件如何通过数字信号运行或传输数据。③了解常见微命令的编码、入口地址等内容。

课程设计主要有设计、编码、测试三个环节。

对于设计，微程序不同于平常的程序设计，微程序中会频繁用到两层的地址操作。比如，对于一个自定义指令ORDER ADDR1，进入这个指令后，PC会指向ADDR1**的地址**，如果要获得ADDR1指向的内容[ADDR1]，得先将ADDR1的地址通过PC🡪AR传入AR，然后内存MEM通过AR中的地址获取到ADDR1并传入AR，然后MEM通过AR中的新内容ADDR1获取到[ADDR1]。以上过程用到了两层的地址引用，理清这种执行流程后，我对PC、AR、MEM这些部件的印象更为深刻，它们独自的功能都非常的简单，但是通过互相联系产生了复杂的功能。

对于编码，一开始看到附录中的微命令表时是一头雾水的，因为它直接给了微命令的16进制编码，而按照一般的编码逻辑，应当是对照实验册192页的编码表完成二进制编码，然后再转换成16进制，16进制本身没有特别大的意义，它只是简化了二进制命令的书写。对于一个微命令，我们需要设计它的56位编码(还有8位默认为0放在头部)，一个命令的不同部分代表对不同组件的控制信号，比如，B30-28、B26-24、B22-20这三串二进制编码会对运算器am2901进行控制，具体如何编码可以对照192页的表。总结下来，一个微命令有如下五要素：对输入来源的控制、对总线(数据传输)的控制、对运算器am2901的控制、对控制器am2910的控制(确定下一步如何执行，详情见192页)、下一条微命令的地址。

测试，测试流程比较固定，之前已经通过实验三基本熟悉了实验的过程，不过每次测试一开始都要对着编码结果一个个输入16进制数，这很容易出现打错字而导致后续测试出问题。因此，在测试开始之前，我重新浏览了一下测试的流程：①写入自定义指令的编码。②编写微码装入程序并运行。③编写测试程序(汇编)。④向上一步的测试程序装入自定义指令。⑤运行测试程序并查看结果。过程并不难，主要是都交给人工输入的话，效率会很低。为此考虑编写一个程序来自动化输入，查资料(跟郑尉欣同学合作完成)发现python有一个pyautogui库，它可以实现自动输入内容。具体的测试内容可以先写入txt文件，然后通过pyautogui库结合当前位于TEC-2控制台的鼠标指针的位置，读取txt文件的内容实现测试内容的自动输入。这样一来测试的效率就可以变得非常高。相比设计和编码环节，测试环节更好地锻炼了我的计算思维能力和资料搜集能力，当我发现一项工作繁琐重复时，可以想一想能不能考虑用代码辅助我的工作，需求明确后，可以上网查询有没有可以直接拿来用的库(各种语言的库能够满足大部分的需求)，有就直接拿来用。

最后，通过本次课程设计，加深了我对计算机底层运行过程的理解，计算机对我而言不再是一个大黑箱，而是一条分工明确的流水线工程。现在的计算机都遵循冯诺依曼体系，通过熟悉TEC-2的底层，后续也可以举一反三拓展到其他计算机上。希望此次经历能够让我更加从容地面对将来的专业知识。