一、目的和要求

深入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；完成微程序控制的特定功能计算机的指令系统设计和调试。

二、实验环境

Windows10, Tec-2 模拟程序，Tec-2 实验系统监控程序，微指令分析器 analyzer

三、具体内容

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。

（一）TEC-2机控制器部件的关键内容包括：

（1）由7片LS6116随机读写存储器芯片组成的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条机器指令的微程序，经组织后写入这一控制存储器，这一过程称为装入微码。装入完成后，将从监控程序的零地址执行指令，完成TEC-2机的启动过程。这之后，还可以用LDMC指令按规定的办法向控制存储器写入新的微程序，以实现新的机器指令。控制存储器的地址为μRA9~μRA0，读出的信息送微指令流水线寄存器PLR。

（2）微指令寄存器PLR由7片8位的寄存器芯片（6片LS374和1片LS273）组成，用于存放当前微指令的内容。

（3）微程序定序器AM2910芯片是微程序控制器中非常关键、也是稍微难懂一点的部分。它的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和当前指令地址寄存器IP，是用运算器通用寄存器组中的两个选定的寄存器R5和R6实现的。

（5）指令寄存器IR，用于存放当前正在执行的指令内容。

（6）为AM2910提供输入地址信号的配套线路，包括：

① 由两片LS2716 ROM芯片组成的MAPROM，它将指令寄存器中的操作码转换成一段微程序的入口地址；

② 由l片LS125和1片LS244组成的接收内部总线的IB9~IB0信号的选择门电路，它把由水平板上的开关提供的微指令地址送AM2910的地址输入端；

③ 由1片LS125和微指令寄存器的PLR55~48组成的一组地址输入，把当前微指令中的后续地址B55~46送入AM2910的地址输入端；请注意，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。

这三组信号均为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）由AM2910芯片的10位地址输出信号驱动的配套电路，包括：

① 由一片LSl75和一片LS374寄存器组构成的记忆电路，用于保存当前微指令的地址，其输出仅送往显示灯部分，以显示当前微指令的地址；

② 由3片LS257（四位的二选一电路）芯片构成的微控存的地址选择形成电路，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。

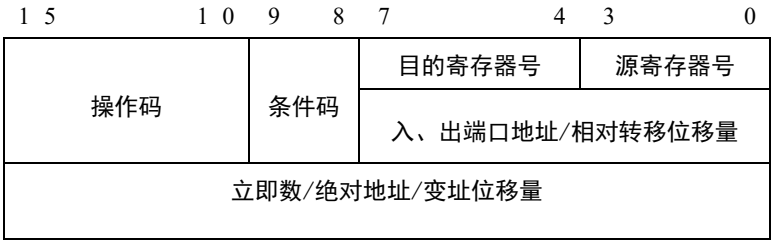
③ 与此有关的还有3片计数器芯片LS161组成的地址计数器电路，其输出（共10位）通过两片LS244与刚提到的3片LS257的10位输出形成“线或”关系，用计数器的一个输出端Y11实现二者之间的选择，Y11为0时，μRA9~μRA 0是计数器的输出信号，提供完成装入微码过程的微控存的地址，Y11为1时，表明装入微码的过程已结束，微控存的地址μRA9~μRA 0为3片LS257的输出信号，以完成机器指令运行过程中的微控存的读、写（写仅用于LDMC指令）操作。

（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LSl61芯片，2片LS244、1片Gal20v8、1片LS74、1片LSl23和1片LS00。

（二）微指令格式

TEC-2 机类 PC 机基本指令系统，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，其相应微程序固化在 ROM 芯片中，其余 11 条将留给实验人员自行设计实现。已实现的指令与实验人员实现的指令能方便地用在同一实验程序中，为控制器部件的实验带来很大方便。

TEC-2 机的基本指令的格式比较固定。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，常用指令类型和运算还是比较齐全的。其指令的基本格式如下：



单字指令仅用一个指令字。

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

第一个指令字的最高 6 位是操作码，9、8 两位是条件码，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 的值作判断条件。

最低 8 位有如下的用法：

（1）分成两个 4 位的字段，用于给出所用的通用寄存器编号。对双操作数指令，这里可以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，另外一个 4 位字段则不用。

（2）用于给出输入输出指令的端口地址。

（3）用于给出相对寻址的位移量，位移量用 8 位补码表示，其范围从－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。

TEC-2 机上现有的软件，包括监控程序都是用已实现的 53 条指令设计完成的，而且在 PC 机上实现的 TEC-2 机的交叉汇编程序，能在 PC 机中直接汇编生成出 TEC-2 机的指令代码，即 TEC-2 机上的执行程序。

每条微指令由56位组成，从高向低各位标记为B55~B0，分为13个字段，如下所示：

表1　微指令的56位微码

B55~B46 下地址	B45 B44 备用	B43~B40 CI3~ CI 0	B39~B37 SCC	B36 SC	B35 备用	B34~B32 SST	B31 MIO	B30~B28 MI8~MI6	B27 REQ

B26-B24 MI5~ MI3	B23 WE	B22~B20 MI2~MI0	B19~B16 A口地址	B15~B12 B口地址	B11B10 SCI	B9B8 SSH	B7 SA	B6~B4 DC1	B3 SB	B2~B0 DC2

56位微码分放在7片存储器芯片中。

18位微码用于控制与给出每条微指令的下地址，供控制器部件本身使用。其中：B55~B46的10位微码是下地址字段；B45、B44位备用；

B43~B40为CI3~CI0，是用于给出AM2910芯片的16种命令码的编号；

B39~B37、B36分别为3位的SCC和1位的SC，给出AM2910芯片的条件判断信号/CC的选择码，用于保证微指令的条件转移等；

提供给运算器部件的控制信号有26位，分别是：

A口地址、B口地址、A口和B口地址选择控制信号SA、SB，合计共10位；

3组3位的AM2910的控制信号MI8~MI6、MI5~MI3、MI2~MI0共9位；

控制标志寄存器写入的SST、最低位进位控制SCi、移位信号形成的SSH 3个字段共7位。它们的使用方法已在运算器部件的讨论中讲述清楚。

还有3位微码/MIO、REQ和/WE用于控制内存的读写、外设接口的读写、以及微码的装入。其规定如下表2所示：

表2

/MIO	REQ	/WE	操作功能	/MIO	REQ	/WE	操作功能
0	0	0	存储器写	0	1	1	I/O读
0	0	1	存储器读	1	0	×	不操作
0	1	0	I/O写	1	1	×	装入微码

2. 使用 TEC-2 仿真软件进行微指令级的设计和调试，完成微程序控制的特定功能计算机的指令设计。

选定指令格式、操作码，设计如下指令：

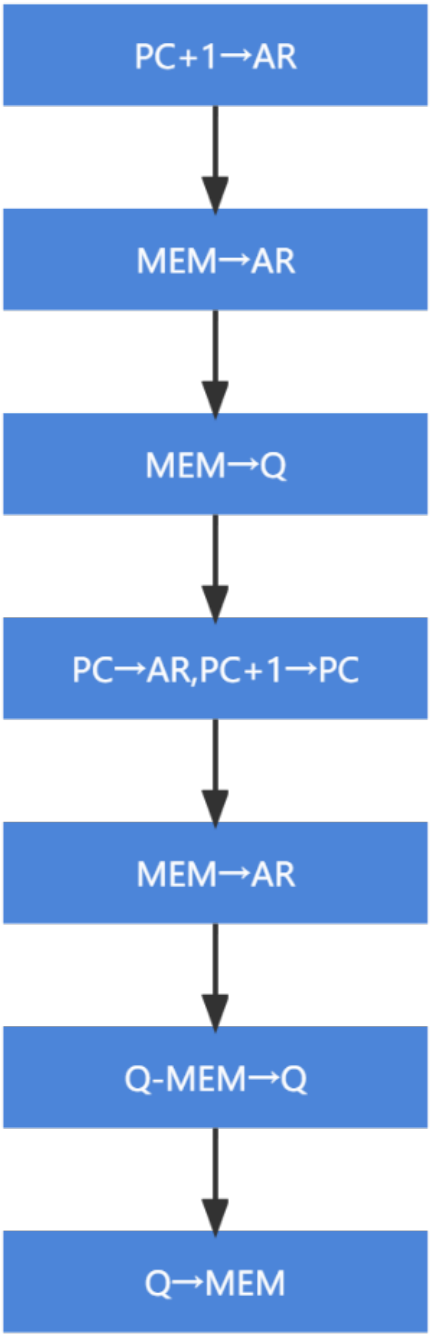
（1）把用绝对地址表示的内存单元 ADDR1 中的内容与内存单元 ADDR2 中的内容相减，结果存于内存单元 ADDR1 中。

指令格式：D5xx，ADDR1，ADDR2，　　三字指令（控存入口 100H）

功能：　　[ADDR1]=[ADDR1] -[ADDR2]

设计思路：该指令为三字指令，需要取出两个地址中的内容进行相减，并储存回第一个地址位置。由此可见第一个地址直至最后都需要使用到，为了增加效率，可以先将第二个操作数取出储存至 Q 寄存器，再取用第一个操作数

微指令流程图



微指令详细设计

序号	微指令功能	微指令代码	详细功能
1	$PC+1 \rightarrow AR$	0000 0E00 90B5 5402	为 AR 读入第二个操作数的地址做准备
2	$MEM \rightarrow AR$	0000 0E00 10F0 0002	AR 读入第二个操作数的地址
3	$MEM \rightarrow Q$	0000 0E00 00F0 0000	读第二个操作数并送 Q 寄存器
4	$PC \rightarrow AR, PC+1 \rightarrow PC$	0000 0E00 A0B5 5402	为 AR 读入第一个操作数的地址做准备
5	$MEM \rightarrow$ AR	0000 0E00 10F0 0002	AR 读入第一个操作数的地址
6	$Q-MEM \rightarrow Q$	0000 0E01 02E0 0000	读第一个操作数并进行减法，结果送 Q 寄存器
7	$Q \rightarrow MEM, \overline{CC}=0$	0029 0300 1020 0010	Q 寄存器结果送回第一个操作数位置

Instruction#1

#1								下地址字段										备用		CI3-0				SCC			SC	备用	SST		
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	X	X	X
0				0				0				0				0				E				0				0			
/MIO	MI8-6			/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	X	X	X	X	X	X	X	0	1	0
9				0				B				5				5				4				0				2			

B55-B46	下地址字段	顺序执行时，下地址段任意
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	顺序执行
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	不操作
B30-B28	MI8-6	运算器 Y 输出送 F 口
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 0 和 B 口
B19-B16	A 口	R5(PC)
B15-B12	B 口	R5(PC)
B11-B10	SCi	进位设置为 1(实现 PC+1)
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未向总线发送控制，可任意
B3	SB	选择 B 口地址
B2-B0	DC2	运算器输出送 AR，选/GAR

Instruction#2

-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	X	X	X
0				0				0				0				0				E				0				0			
/MIO	MI8-6			/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	0	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	1	0
1				0				F				0				0				0				0				2			

B55-B46	下地址字段	顺序执行时，下地址段任意
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	顺序执行
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	储存器读取
B30-B28	MI8-6	运算器 Y 输出送 F 口
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 D 和 0
B19-B16	A 口	未用，任意
B15-B12	B 口	未用，任意
B11-B10	SCi	进位为 0
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未向总线发送控制，可任意
B3	SB	选择 B 口地址
B2-B0	DC2	运算器输出送 AR，选/GAR

Instruction#3

#3								下地址字段										备用		CI3-0				SCC			SC	备用	SST		
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	X	X	X
0				0				0				0				0				E				0				0			
/MIO	MI8-6			/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0				0				F				0				0				0				0				0			

B55-B46	下地址字段	顺序执行时，下地址段任意
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	顺序执行
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	储存器读取
B30-B28	MI8-6	运算器 Y 输出送 F 口
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 D 和 0
B19-B16	A 口	未用，任意
B15-B12	B 口	未用，任意
B11-B10	SCi	进位为 0
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未向总线发送控制，可任意
B3	SB	选择 B 口地址
B2-B0	DC2	未使用

Instruction#4

#4								下地址字段										备用		CI3-0				SCC			SC	备用	SST		
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	X	X	X
0				0				0				0				0				E				0				0			
/MIO	MI8-6				/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	X	X	X	X	X	X	X	0	1	0
A				0				B				5				5				4				0				2			

B55-B46	下地址字段	顺序执行时，下地址段任意
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	顺序执行
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	不操作
B30-B28	MI8-6	运算器 Y 输出送 F 口,寄存器结果返回至 B 口
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 0 和 B 口
B19-B16	A 口	R5(PC)
B15-B12	B 口	R5(PC)
B11-B10	SCi	进位设置为 1(实现 PC+1)
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未向总线发送控制，可任意
B3	SB	选择 B 口地址
B2-B0	DC2	运算器输出送 AR，选/GAR

Instruction#5

#3								下地址字段										备用		CI3-0				SCC			SC	备用	SST		
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	X	X	X
0				0				0				0				0				E				0				0			
/MIO	MI8-6				/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0				0				F				0				0				0				0				0			

B55-B46	下地址字段	顺序执行时，下地址段任意
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	顺序执行
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	储存器读取
B30-B28	MI8-6	运算器 Y 输出送 F 口
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 D 和 0
B19-B16	A 口	未用，任意
B15-B12	B 口	未用，任意
B11-B10	SCi	进位为 0
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未向总线发送控制，可任意
B3	SB	选择 B 口地址
B2-B0	DC2	未使用

Instruction#6

#6								下地址字段										备用		CI3-0				SCC			SC	备用	SST		
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	0	0	1
0				0				0				0				0				E				0				1			

/MIO	MI8-6			/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0				2				E				0				0				0				0				0			

B55-B46	下地址字段	顺序执行时，下地址段任意
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	顺序执行
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	接受 ALU 标志位输出值
B31,B27,B23	/MIO、/REQ、/WE	储存器读取
B30-B28	MI8-6	运算器 Y 输出送 F 口，寄存器结果返回至 Q
B26-B24	MI5-3	运算功能选择为“R-S”
B22-B20	MI2-0	Am2901 的运算数来源选择 D 和 Q
B19-B16	A 口	未用，任意
B15-B12	B 口	未用，任意
B11-B10	SCi	进位为 0
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未向总线发送控制，可任意
B3	SB	选择 B 口地址
B2-B0	DC2	未使用

Instruction#7

#7								下地址字段										备用		CI3-0				SCC			SC	备用	SST		
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	1	0	1	0	0	1	0	0	X	X	0	0	1	1	1	1	1	X	X	X	X	X
										A				4																	
0				0				2				9				0				3				0				0			

/MIO	MI8-6			/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	0	0	0	1	0	X	X	X	X	X	X	X	X	0	0	X	X	X	0	0	1	X	X	X	X
1				0				2				0				0				0				1				0			

B55-B46	下地址字段	跳转执行，下地址段为 A4，进行中断检测
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	条件转移
B39-B37	SCC	判断$\overline{CC} = 0$
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	储存器写入
B30-B28	MI8-6	运算器 Y 输出送 F 口
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 0 和 Q
B19-B16	A 口	未用，任意
B15-B12	B 口	未用，任意
B11-B10	SCi	进位为 0
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	向总线发送控制，运算器输出至[AR]
B3	SB	选择 B 口地址
B2-B0	DC2	未使用

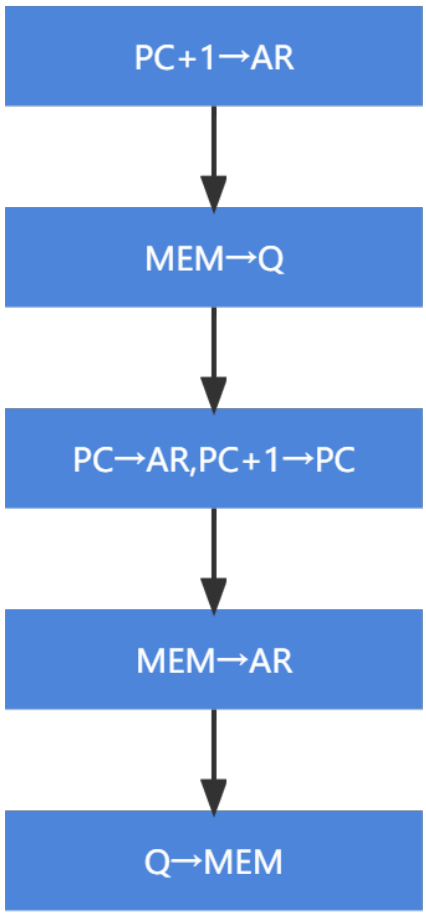
（2）把立即数 DATA 传送至地址为 ADDR 的内存单元中保存。

指令格式：D8xx，ADDR，DATA，三字指令（控存入口 110H）

功能：　　[ADDR]←DATA

设计思路：该指令为三字指令，类似于实验一，需要先取出第二个地址中的内容，并储存回第一个地址位置。由此可见第一个地址直至最后都需要使用到，为了增加效率，可以先将第二个操作数取出储存至 Q 寄存器，再取用第一个地址

微指令流程图



微指令详细设计

序号	微指令功能	微指令代码	详细功能
1	$PC+1\rightarrow AR$	0000 0E00 90B5 5402	为 AR 读入第二个操作数的地址做准备
2	$MEM\rightarrow Q$	0000 0E00 00F0 0000	读第二个操作数并送 Q 寄存器
3	$PC\rightarrow AR, PC+1\rightarrow PC$	0000 0E00 A0B5 5402	为 AR 读入第一个操作数的地址做准备
4	$MEM\rightarrow$ AR	0000 0E00 10F0 0002	AR 读入第一个操作数的地址
5	$Q\rightarrow MEM, \overline{CC}=0$	0029 0300 1020 0010	Q 寄存器结果送回第一个操作数位置

Instruction#1

#1								下地址字段										备用		CI3-0				SCC			SC	备用	SST		
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	X	X	X
0				0				0				0				0				E				0				0			

/MIO	MI8-6			/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	X	X	X	X	X	X	X	0	1	0
9				0				B				5				5				4				0				2			

B55-B46	下地址字段	顺序执行时，下地址段任意
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	顺序执行
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	不操作
B30-B28	MI8-6	运算器 Y 输出送 F 口
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 0 和 B 口
B19-B16	A 口	R5(PC)
B15-B12	B 口	R5(PC)
B11-B10	SCi	进位设置为 1(实现 PC+1)
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未向总线发送控制，可任意
B3	SB	选择 B 口地址
B2-B0	DC2	运算器输出送 AR，选/GAR

Instruction#2

#2								下地址字段										备用		CI3-0				SCC			SC	备用	SST		
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	X	X	X
0				0				0				0				0				E				0				0			

/MIO	MI8-6			/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0				0				F				0				0				0				0				0			

B55-B46	下地址字段	顺序执行时，下地址段任意
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	顺序执行
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	储存器读取
B30-B28	MI8-6	运算器 Y 输出送 F 口
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 D 和 0
B19-B16	A 口	未用，任意
B15-B12	B 口	未用，任意
B11-B10	SCi	进位为 0
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未向总线发送控制，可任意
B3	SB	选择 B 口地址
B2-B0	DC2	未使用

Instruction#3

#3								下地址字段										备用		CI3-0				SCC				SC	备用	SST			
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	X	X	X
0				0				0				0				0				E				0				0			

/MIO	MI8-6			/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	X	X	X	X	X	X	X	0	1	0
A				0				B				5				5				4				0				2			

B55-B46	下地址字段	顺序执行时，下地址段任意
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	顺序执行
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	不操作
B30-B28	MI8-6	运算器 Y 输出送 F 口,寄存器结果返回至 B 口
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 0 和 B 口
B19-B16	A 口	R5(PC)
B15-B12	B 口	R5(PC)
B11-B10	SCi	进位设置为 1(实现 PC+1)
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未向总线发送控制，可任意
B3	SB	选择 B 口地址
B2-B0	DC2	运算器输出送 AR，选/GAR

Instruction#4

#4								下地址字段										备用		CI3-0				SCC				SC	备用	SST			
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	X	X	X
0				0				0				0				0				E				0				0			

/MIO	MI8-6			/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	0	1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	1	0
1				0				F				0				0				0				0				2			

B55-B46	下地址字段	顺序执行时，下地址段任意
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	顺序执行
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	储存器读取
B30-B28	MI8-6	运算器 Y 输出送 F 口
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 D 和 0
B19-B16	A 口	未用，任意
B15-B12	B 口	未用，任意
B11-B10	SCi	进位为 0
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未向总线发送控制，可任意
B3	SB	选择 B 口地址
B2-B0	DC2	未使用

Instruction#5

#5								下地址字段								备用		CI3-0				SCC			SC	备用	SST			
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	1	0	1	0	0	1	0	0	X	X	0	0	1	1	1	1	1	X	X	X	X	X
										A				4																	
0				0				2				9				0				3				0				0			

/MIO	MI8-6			/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi		SSH		SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	0	0	0	1	0	X	X	X	X	X	X	X	X	0	0	X	X	X	0	0	1	X	X	X	X
1				0				2				0				0				0				1				0			

B55-B46	下地址字段	跳转执行，下地址段为 A4，进行中断检测
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	条件转移
B39-B37	SCC	判断$\overline{CC}=0$
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	储存器写入
B30-B28	MI8-6	运算器 Y 输出送 F 口
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 0 和 Q
B19-B16	A 口	未用，任意
B15-B12	B 口	未用，任意
B11-B10	SCi	进位为 0
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	向总线发送控制，运算器输出至[AR]
B3	SB	选择 B 口地址
B2-B0	DC2	未使用

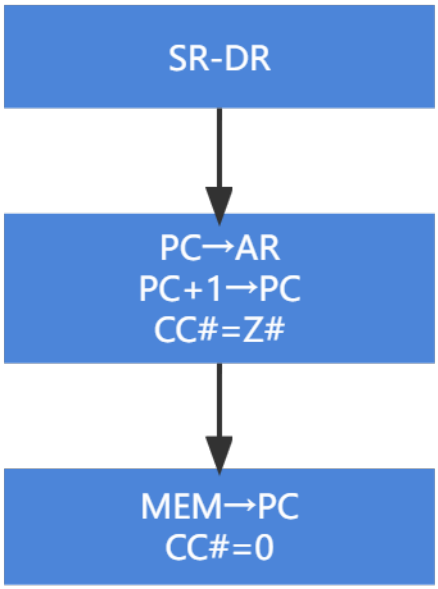
（3）转移指令。判断两个通用寄存器内容是否相等，若不相等则转移到指定目的地址，否则顺序执行。

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。

设计思路：该指令为二字指令，第一步让 SR-DR，把结果是否为零的信息记录在标志位，接着根据标志位来条件跳转。$\bar{Z}$等于 1，说明两个数不相等，微程序跳转，取出需要跳转的位置至 PC。反之，顺序执行。

微指令流程图



微指令详细设计

序号	微指令功能	微指令代码	详细功能
1	$SR-DR$	0000 0E01 9190 0088	通过 SR-DR 改变 Z 状态值
2	$PC \rightarrow AR$ $PC+1 \rightarrow PC$ $\overline{CC}=\bar{Z}$	0029 03E0 A0B5 5402	为顺序执行做准备 并通过判断 Z 状态值判断是否要跳转
3	$MEM \rightarrow \mathrm{PC}, \overline{CC}=0$	0029 0300 30F0 5000	跳转

Instruction#1

#1								下地址字段										备用		CI3-0				SCC				SC	备用	SST			
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	0	0	1
0				0				0				0				0				E				0				1			
/MIO	MI8-6				/REQ	MI5-3			/WE	MI2-0				A口				B口				SCi		SSH		SA	DC1			SB	DC2			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	1	1	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X	1	X	X	X
9				1				9				0				0				0				8				8			

B55-B46	下地址字段	顺序执行时，下地址段任意
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	顺序执行
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	接受 ALU 标志位输出值
B31,B27,B23	/MIO、/REQ、/WE	不操作
B30-B28	MI8-6	运算器 Y 输出送 F 口
B26-B24	MI5-3	运算功能选择为“S-R”
B22-B20	MI2-0	Am2901 的运算数来源选择 A 口和 B 口
B19-B16	A 口	未用，任意
B15-B12	B 口	未用，任意
B11-B10	SCi	进位为 0
B9-B8	SSH	无需移位
B7	SA	指令给 R 赋值
B6-B4	DC1	未向总线发送控制，可任意
B3	SB	指令给 S 赋值
B2-B0	DC2	未使用

Instruction#2

#2								下地址字段										备用		CI3-0				SCC				SC	备用	SST			
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	1	0	1	0	0	1	0	0	X	X	0	0	1	1	1	1	1	X	X	X	X	X
										A				4																	
0				0				2				9				0				3				E				0			
/MIO	MI8-6				/REQ	MI5-3			/WE	MI2-0				A口				B口				SCi		SSH		SA	DC1			SB	DC2			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	X	X	X	X	X	X	X	0	1	0
A				0				B				5				5				4				0				2			

B55-B46	下地址字段	跳转执行，下地址段为 A4，进行中断检测
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	条件转移
B39-B37	SCC	判断$\overline{CC} = \bar{Z}$
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	不操作
B30-B28	MI8-6	运算器 Y 输出送 A 口,寄存器结果送回 B
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 0 和 B
B19-B16	A 口	R5(PC)
B15-B12	B 口	R5(PC)
B11-B10	SCi	进位设置为 1(实现 PC+1)
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未使用
B3	SB	选择 B 口地址
B2-B0	DC2	运算器输出送 AR，选/GAR

Instruction#3

#3								下地址字段								备用		CI3-0				SCC			SC	备用	SST		
-	-	-	-	-	-	-	-	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
X	X	X	X	X	X	X	X	X	X	1	0	1	0	0	1	0	0	X	X	0	0	1	1	X	X	X	X	X	X	X	X
										A				4																	
0				0				2				9				0				3				0				0			
/MIO	MI8-6			/REQ	MI5-3			/WE	MI2-0			A口				B口				SCi	SSH			SA	DC1			SB	DC2		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	0	1	1	1	1	X	X	X	X	0	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X
3				0				F				0				5				0				0				0			

B55-B46	下地址字段	跳转执行，下地址段为 A4，进行中断检测
B45-B44	备用	备用位任意
B43-B40	CI3~CI0	条件转移
B39-B37	SCC	任意
B36	SC	任意
B35	备用	备用位任意
B34-B32	SST	不是运算，任意即可
B31,B27,B23	/MIO、/REQ、/WE	储存器读取
B30-B28	MI8-6	运算器 Y 输出送 F 口，寄存器输出 F 至 B
B26-B24	MI5-3	运算功能选择为“R+S”
B22-B20	MI2-0	Am2901 的运算数来源选择 D 和 0
B19-B16	A 口	未用，任意
B15-B12	B 口	R5(PC)
B11-B10	SCi	进位为 0
B9-B8	SSH	无需移位
B7	SA	选择 A 口地址
B6-B4	DC1	未使用
B3	SB	选择 B 口地址
B2-B0	DC2	未使用

3. 测试程序设计及调试。

实验一

测试样例解释：
将 365 放至 0A00 内，50 放至 0A01 内
输入命令 D500 0A00 0A01 即将 0A00 内容减去 0A01 内容，放回至 0A00

（一）输入微码

```
>E900
0900    0000:0000  0000:0E00  0000:90B5  0000:5402  0000:0000
0905    0000:0E00  0000:10F0  0000:0002  0000:0000  0000:0E00
090A    0000:00F0  0000:0000  0000:0000  0000:0E00  0000:A0B5
090F    0000:5402  0000:0000  0000:0E00  0000:10F0  0000:0002
0914    0000:0000  0000:0E01  0000:02E0  0000:0000  0000:0029
0919    0000:0300  0000:1020  0000:0010
```

（二）将微码加载到微控存中

```
>A800
0800: MOV R1,900
0802: MOV R2,7
0804: MOV R3,100
0806: LDMC
0807: RET
0808:
>G800
```

（三）运算准备

```
>A820
0820: MOV R0,0365
0822: MOV R1,0050
0824: MOV [A00],R0
0826: MOV [A01],R1
0828: RET
0829: tor of Virtua
>G820
```

（四）输入程序，运行新指令

```
>E829
0829    0000:D500  0000:0A00  0000:0A01  0000:AC00
>G829
```

（五）观察结果

```
>DA00
0A00    0315  0050  0000  0000  0000  0000  0000  0000  ...P............
0A08    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A10    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A18    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A20    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A28    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A30    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A38    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A40    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A48    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A50    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A58    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A60    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A68    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A70    0000  0000  0000  0000  0000  0000  0000  0000  ................
```

实验二

测试样例解释：

输入命令 D800 0A00 6666 即将立即数 6666 放至 0A00

（一）输入微码

```
>E900
0900    0000:0000  0000:0E00  0000:90B5  0000:5402  0000:0000
0905    0000:0E00  0000:00F0  0000:0000  0000:0000  0000:0E00
090A    0000:A0B5  0000:5402  0000:0000  0000:0E00  0000:10F0
090F    0000:0002  0000:0029  0000:0300  0000:1020  0000:0010
```

（二）将微码加载到微控存中

```
>A800
0800: MOV R1,900
0802: MOV R2,5
0804: MOV R3,110
0806: LDMC
0807: RET
0808:
>G800
>
```

（三）输入程序，运行新指令

```
>E820
0820    0000:D800  0000:0A00  0000:6666  0000:AC00
>G820
>
```

（四）观察结果

```
>DA00
0A00    6666  0000  0000  0000  0000  0000  0000  0000  ff..............
0A08    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A10    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A18    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A20    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A28    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A30    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A38    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A40    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A48    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A50    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A58    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A60    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A68    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A70    0000  0000  0000  0000  0000  0000  0000  0000  ................
>
```

实验三

测试样例解释：
R1,R2,R3 内容分别未 1111,6666,6666
首先 E112 0829 即判断 R1,R2 内容是否相等，若相等顺序执行，若不等则转移至 0829(0829 内容为把 6666 赋给 R1，即最后若 R1 为 6666 微程序运行则正常)
再 E123 082E 即判断 R2,R3 内容是否相等，若相等顺序执行，若不相等则转移至 082E(082E 内容为把 1111 赋给 R3，即最后若 R3 为 6666 则微程序运行正常)
总之，若最后 R1-R3 全为 6666 则运行正常

（一）输入微码

```
>E900
0900    0000:0000  0000:0E01  0000:9190  0000:0088  0000:0029
0905    0000:03E0  0000:A0B5  0000:5402  0000:0029  0000:0300
090A    0000:30F0  0000:5000
>
```

（二）将微码加载到微控存中

```
>A800
0800: MOV R1,900
0802: MOV R2,3
0804: MOV R3,130
0806: LDMC
0807: RET
0808:
>G800
```

（三）运算准备

```
>A820
0820: MOV R1,1111
0822: MOV R2,6666
0824: MOV R3,6666
0826: NOP
0827: NOP
0828: RET
0829: MOV R1,6666
082B: NOP
082C: NOP of Virtual TEC-2 B
082D: RET
082E: MOV R3,1111
0830: RET
0831:
>
```

（四）输入程序

```
>E826
0826    0000:E112  0000:0829
>E82B
082B    0000:E123  0000:082E
>
```

（五）运行新指令，观察结果

```
>R
R0=0000  R1=090C  R2=0000  R3=0133  SP=FFFF  PC=0800  IP=0807  R7=0000  R8=0000
R9=0000 R10=0000 R11=0000 R12=0000 R13=0000 R14=0000 R15=0000   F=01001111
0800: 2C10 0900  MOV    R1,     0900
>G820
>R
R0=0000  R1=6666  R2=6666  R3=6666  SP=FFFF  PC=0820  IP=082D  R7=0000  R8=0000
R9=0000 R10=0000 R11=0000 R12=0000 R13=0000 R14=0000 R15=0000   F=01001111
0820: 2C10 1111  MOV    R1,     1111
```

四．实验心得

通过本次课程设计，我对 TEC-2 机的指令系统有了更深入的了解，也加深了对 Am2901 结构的掌握，在最开始之初我对于老师上课讲的有关实验的内容感到十分困惑，不明白其具体的用处，难以将他们串联在一起。这次设计让我们把课堂上学习的理论拿到了实践中来,将很多以前感到很抽象的内容加深了理解，在与同学讨论，争辩的过程中，使我对所学的内容理解更加的透彻,同时更找到了自己过去的不足和知识上的漏洞。在查找参考资料同时，我也掌握了很多新的内容。这使我获益颇丰。我也明白，要成功设计出一条指令，不但需要扎实的理论知识，更需要严谨的思维和态度，不断将其优化。

首先最重要的搞清楚各个指令的含义

E 指令：逐“字”输入

```
>E900
0900    0000:
```

最前面的 0900 是提示你当前位置，后面的 0000 是提示你，当前位置的内容是"0000"

```
>E900
0900    0000:1234  0000:5678  0000:9ABC  0000:1234  0000:5678
0905    0000:9ABC
```

Enter 代表改动结束，Space 代表改下一个，每一行会显示 5 个“字”（我对这个“字”的定义是 4 位的一个 16 进制数）。比如上面这张图，第二行的五个就是 0900-0904 位置的内容（很奇怪，一个微指令是 4 个字，为什么非要一行显示 5 个字……）

肯定有人会好奇每个前面这个 0000 到底是什么意思，他其实就是再告诉你，这个位置原来储存的东西是什么，比如上面这个图运行完之后，我如果再从 900 开始执行 E 命令：

```
>E900
0900    0000:1234  0000:5678  0000:9ABC  0000:1234  0000:5678
0905    0000:9ABC
>E900
0900    1234:6666  5678:6666  9ABC:6666  1234:6666  5678:6666
```

大概就能明白是什么意思了。

A,E 指令的作用其实是一样的，可以理解为

E 输入的是 machine code language，

A 输入的是高级语言

比如下图

```
>E800
0800    0000:AC00
>A801
0801: RET
0802:
>D800
0800    AC00  AC00  0000  0000  0000  0000  0000  0000  ................
0808    0000  0000  0000  0000  0000  0000  0000  0000  ................
0810    0000  0000  0000  0000  0000  0000  0000  0000  ................
0818    0000  0000  0000  0000  0000  0000  0000  0000  ................
```

（D 指令：显示内容，“D800”会从 0800 位置向后显示 128 个“字”）

E 指令中，直接把“AC00”这个内容存到 0800 中

A 指令中，会把 RET 这个高级指令，“编译”成对应的 machine code，也就是“AC00”，存到 0801 中

也就是说，殊途同归，哪个方便来哪个，E 指令底层，A 指令直观。

U 指令是 A 指令的逆

```
>U800
0800: AC00        RET
0801: AC00        RET
0802: 0000        NOP
0803: 0000        NOP
0804: 0000        NOP
```

他会“反编译”一下，告诉你你每一个字（或者几个）代表什么意思（比如上面的两个 RET）

G 指令

刚才前面的指令，都是在对内存进行修改或者读取，并没有真正意义上的运行。

G 指令会运行你的代码至 RET 为止，也就是说“RET”其实代表了一段代码的终止（当作 c 里面的 return 就行），如果你没有 RET，使用 G 指令会导致 dead loop。

```
>R
R0=0000  R1=0000  R2=0000  R3=0000  SP=FFFF  PC=0000  IP=0000  R7=0000  R8=0000
R9=0000 R10=0000 R11=0000 R12=0000 R13=0000 R14=0000 R15=0000   F=00001111
0000: 0000        NOP
>G800
>R
R0=0000  R1=0000  R2=0000  R3=0000  SP=FFFF  PC=0800  IP=0800  R7=0000  R8=0000
R9=0000 R10=0000 R11=0000 R12=0000 R13=0000 R14=0000 R15=0000   F=00001111
0800: AC00        RET
```

(R 指令：显示所有寄存器的值)

上面这张图显示了我在运行 800（还是之前存的 RET）之后，各个寄存器内容的变化，首先是 PC 变成了 800，也就是这次运行开始的地方，IP 代表这次运行结束的地方，由于 800 只有一个 RET(也就是 return)，运行还没有开始就结束了。

实验

每个实验基本上就是，告诉你一个要求，让你编写相应的微程序，然后整个例子运行一下，证明能跑就行。大概流程分为以下这么几个：

设计微程序

把微程序整到控存里面

跑个 demo

一点一点来

首先看帮助真的很重要，内置的帮助能解决绝大多数的问题。

我觉得最好的方法就是先去分析一个现成的，把到底发生了什么搞清楚再去设计新的

（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：D5xx，ADDR1，ADDR2，　　三字指令（控存入口 100H）↵

功能：　　[ADDR1]=[ADDR1] -[ADDR2]↵

这是课设第一题，以此为基础来研究一下。

```
>E900
0900    0000:0000  0000:0E00  0000:90B5  0000:5402  0000:0000
0905    0000:0E00  0000:10F0  0000:0002  0000:0000  0000:0E00
090A    0000:00F0  0000:0000  0000:0000  0000:0E00  0000:A0B5
090F    0000:5402  0000:0000  0000:0E00  0000:10F0  0000:0002
0914    0000:0000  0000:0E01  0000:02E0  0000:0000  0000:0029
0919    0000:0300  0000:1020  0000:0010
>A800
0800:   MOV R1,900
0802:   MOV R2,7
0804:   MOV R3,100
0806:   LDMC
0807:   RET
0808:
>G800
>A820
0820:   MOV R0,0365
0822:   MOV R1,0050
0824:   MOV [A00],R0
0826:   MOV [A01],R1
0828:   RET
0829:
>G820
>E829
0829    0000:D500  0000:0A00  0000:0A01  0000:AC00005, Sun Yat-sen University
>G829
>DA00
0A00    0315  0050  0000  0000  0000  0000  0000  0000  ...P............
0A08    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A10    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A18    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A20    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A28    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A30    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A38    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A40    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A48    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A50    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A58    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A60    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A68    0000  0000  0000  0000  0000  0000  0000  0000  ................
0A70    0000  0000  0000  0000  0000  0000  0000  0000  ................
```

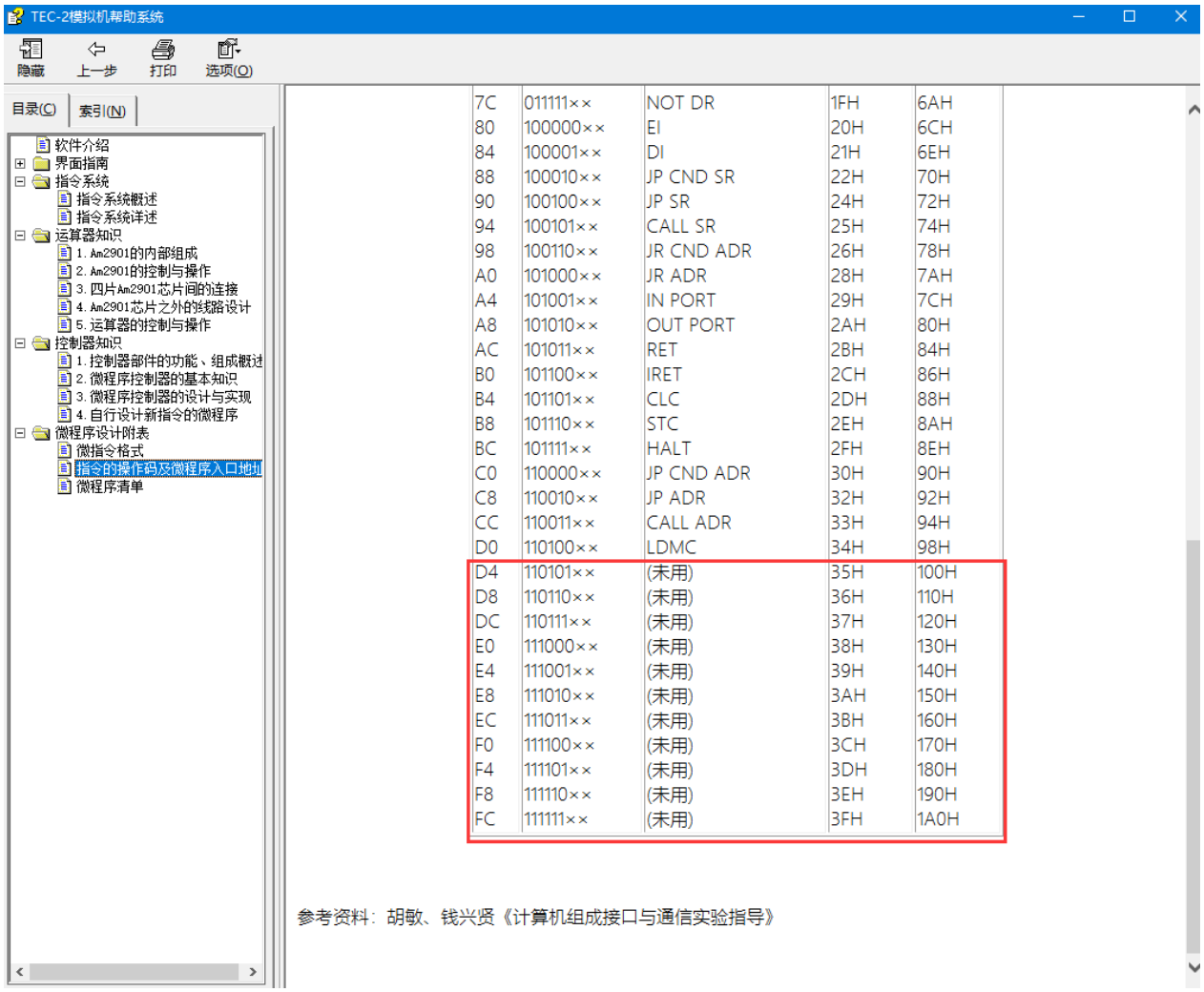
先不要管微程序的设计，和这个微程序具体构成，先看明白每一步发生了什么事情

首先第一步是把微程序代码装到了 0900 这个位置，4 个字为一条微指令(红框里面就是一条微指令)，数一下一共装入了 7 条微指令

```
>E900
0900    0000:0000  0000:0E00  0000:90B5  0000:5402  0000:0000
0905    0000:0E00  0000:10F0  0000:0002  0000:0000  0000:0E00
090A    0000:00F0  0000:0000  0000:0000  0000:0E00  0000:A0B5
090F    0000:5402  0000:0000  0000:0E00  0000:10F0  0000:0002
0914    0000:0000  0000:0E01  0000:02E0  0000:0000  0000:0029
0919    0000:0300  0000:1020  0000:0010
```

接下来的东西是一些比较死的一些东西

Tec-2 的设计者内置了很多指令，但也留下了给用户自定义的空间



7C	011111××	NOT DR	1FH	6AH
80	100000××	EI	20H	6CH
84	100001××	DI	21H	6EH
88	100010××	JP CND SR	22H	70H
90	100100××	JP SR	24H	72H
94	100101××	CALL SR	25H	74H
98	100110××	JR CND ADR	26H	78H
A0	101000××	JR ADR	28H	7AH
A4	101001××	IN PORT	29H	7CH
A8	101010××	OUT PORT	2AH	80H
AC	101011××	RET	2BH	84H
B0	101100××	IRET	2CH	86H
B4	101101××	CLC	2DH	88H
B8	101110××	STC	2EH	8AH
BC	101111××	HALT	2FH	8EH
C0	110000××	JP CND ADR	30H	90H
C8	110010××	JP ADR	32H	92H
CC	110011××	CALL ADR	33H	94H
D0	110100××	LDMC	34H	98H
D4	110101××	(未用)	35H	100H
D8	110110××	(未用)	36H	110H
DC	110111××	(未用)	37H	120H
E0	111000××	(未用)	38H	130H
E4	111001××	(未用)	39H	140H
E8	111010××	(未用)	3AH	150H
EC	111011××	(未用)	3BH	160H
F0	111100××	(未用)	3CH	170H
F4	111101××	(未用)	3DH	180H
F8	111110××	(未用)	3EH	190H
FC	111111××	(未用)	3FH	1A0H

参考资料：胡敏、钱兴贤《计算机组成接口与通信实验指导》

（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：D5xx，ADDR1，ADDR2，　　三字指令（控存入口 100H）↵

功能：　　[ADDR1]=[ADDR1] -[ADDR2]↵

观察题目里面有两个比较重要的东西，一个是指令格式 D5XX，一个是控存入口 100H，你会发现在帮助中，100H 对应的指令格式却是 D4，他的底层逻辑是这样的：指令 D400-D7FF 所对应的是**同一个函数**，也就是说，你无论是输入 D500,D501….D600….只要在这个范围内，他所对应的内容都是在 100H 这个控存里面的，那他存在的意义到底是什么，用我的话来说就是，“把部分参数固定在指令里面”。在本题，你指令输入 D500,D501….D600….都是无所谓的，因为在你的微指令里面没有用到这些东西。但在第三题中

（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。↵

指令格式为 E1XX,查询上面的帮助后你会发现他对应的位置其实 E000-E3FF 都对应的是这个函数，但最后需要你设计出来的，比如 E158,就是判断寄存器 R5 和寄存器 R8 的内容，也就是我说的“把部分参数固定在指令里面”。而指令是 E1 不是 E0,E2,E3 也是老师有意为止，具体就自己去研究研究吧。

回到刚才说的，你想要把一个微程序加载到控存里面，是需要固定的做法，也就是下面这张图

```
>A800
0800:   MOV R1,900
0802:   MOV R2,7
0804:   MOV R3,100
0806:   LDMC
0807:   RET
0808:
>G800
```

LDMC 这条指令，就是加载自定义指令的指令，调用时，会用到 R1,R2,R3 里面的内容，其分别是
R1:微指令起始位置
R2:微指令条数
R3:要加载到哪个控存
G800 就是之前说过的，从 800 开始运行，运行到 RET 结束。
所以这个图翻译出来就是“从 0900(R1 内容)位置开始，读取 7(R2 内容)条微指令，固定到控存 100(R1 内容)处
而且更重要的一点是，当你把这些加载到控存之后，你就算再改变 0900 后面这些位置的东西，控存 100 的这个地方的指令也不会再变，你可以理解为当你运行 LDMC 时，就是给控存里 make 了一份 copy
也就是说，到这步为止，你已经成功把 D400-D7FF 这些语句所对应的指令准备好了

接下来这部分就是做一些准备工作，把 0A00 位置里放 365 这个数字（随便的），0A01 里放 50

```
>A820
0820:   MOV R0,0365
0822:   MOV R1,0050
0824:   MOV [A00],R0
0826:   MOV [A01],R1
0828:   RET
0829:
>G820
```

再接下来这一步很重要一定要看懂，从 0829 位置开始，分别是

指令格式：D5xx，ADDR1，ADDR2，　　三字指令

D500 刚刚说的指令名字，在这道题里面，你都 D400-D7FF 都可以
0A00 第一个操作数 ADDR1
0A01 第二个操作数 ADDR2
AC00 RET,程序结束
作用就是用(0A00)里面的数字减去(0A01)里面的数字，再存回(0A00)里面

```
>E829
0829    0000:D500  0000:0A00  0000:0A01  0000:AC00
>G829
```

最后结果，之前 0A00 里面是 365，365-50=315 存回 0A00（全文都是 hex 啊）

```
>DA00
0A00    0315  0050  0000  0000  0000  0000  0000  0000
0A08    0000  0000  0000  0000  0000  0000  0000  0000
```

五．附加材料

（一）微程序设计与测试步骤：

1.依照需求，利用微指令分析器设计好微程序

2.进入监控程序，用 E 命令输入微码，如

```
>E900
```

输入首地址为 900 的微码，回车后输入微程序，以空格隔开，回车表示输入完毕。

3.输入加载微码的程序，如

```
>A800
```

可在首地址 800 开始输入加载微码的程序，如

```
0800: MOV R1, 900      ; 微码在内存中首地址为 900，即上步 E900 的 900
0802: MOV R2, 7        ; 一共有 7 条微指令
0804: MOV R3, 100      ; 微码在微控存中的首地址为 100（对应操作码 D4）
0806: LDMC             ;加载微码指令，将微码指令加载到控存
0807: RET
0808:
```

4.运行加载微码的程序，如

```
>G800
```

其中 800 是第 3 步的 A800 的 800，这样微码便装入了微控存中（在上例中即 D4 对应的 100H 首地址）

5.用另一个程序测试新指令，先准备测试数据，如输入

```
>A820
```

然后输入

```
0820: MOV R0, 0011          ; 将 0011 存入 R0
0822: MOV [0890], R0        ; 将 R0 的内容存入地址为 0890 的内存单元中
0824: MOV [0891], R0        ; 将 R0 的内容存入地址为 0891 的内存单元中
0826: NOP
0827: NOP
0828: NOP
0829: RET
```

6.在第 5 步已输入的指令后（NOP 开始的地方）调用新指令，如在上步后输入

```
>E826
```

即将新指令的调用放在 0824: MOV [0891], R0 之后，接着输入

```
D400 0890 0891
```

这样就调用了新指令的操作码 D4，配合操作数 0890,0891。

7.运行测试程序，如依照上例，输入

```
>G820
```

就运行了首地址为 820 的微程序。

8.查看结果，可以用 D 或者 R 命令查看程序运行后寄存器或内存状态。

（二）TEC-2 机的微程序设计

① 下一条微指令的地址的形成

微程序设计的关键技术之一，是处理好每条微指令的下地址，以保证微程序正确、高效地执行。在TEC-2机中，这是通过一片微程序定序器AM2910芯片实现的。

·AM2910芯片的内部结构

AM2910芯片的内部结构框图如图1所示。

AM2910包括一个四输入的多路地址选择器，用来选择寄存器/计数器（R），直接输入（D），微程序计数器（uPC）或微堆栈（F）中的一个作为下一条微指令的地址。

寄存器/计数器由12个D型触发器组成。当它用作寄存器时，主要用于保存一个微地址，用以实现微程序分支；当它用作计数器时，具有减一功能(何时减一，取决于AM2910的命令码)，主要用于控制微程序的循环次数，若装入的初值为N，则可执行N+1次。

微程序计数器由12位增量器和12位寄存器组成。当增量器的进位输入CI为高电平时，多路器的输出Y加1后装入μPC（即μPC←Y十1），用于实现微程序的顺序执行；而当CI为低电平时，多路器的输出Y直接装入μPC（即μPC←Y），用于实现同一条微指令的多次执行。

微堆栈是由5字×12位的寄存器堆和微堆栈指针μSP组成，主要用于保存微子程序调用的返回地址和微程序循环的首地址。微堆栈指针μSP总是指向最后一次压入的数据，因此，执行微程序循环时，允许不执行弹出操作而直接访问微堆栈的栈顶。当堆栈中的数据达到5个时，就发出堆栈已满信号（/FULL=0)，这时，任何压入操作都将覆盖掉栈顶的数据。

AM2910输出3个使能信号：/PL，/MAP和/VECT，用以决定直接输入D的来源。

当/PL有效时（即/PL＝0），D来源于微指令的下地址字段，用于实现微程序转移；当/MAP有效时（即/MAP＝0），D来源于MAPROM，用于实现从机器指令到相应的微程序段的转移；

当/VECT有效时（即/VECT＝0），原意D来源于中断向量，现用于接收手拨微地址。

·AM2910引脚的定义

输入线：

D11~D0：外部直接输入的数据，既可作为寄存器/计数器的初值，也可直接经地址多路选择器从Y输出，作为下一条微指令的地址。

I3~I0：AM2910的命令码，来自微指令字的有关字段，用以选择AM2910的16条命令之一。

/CC：条件输入，若为低电平，则表示测试成功，否则，表示测试失效。

/CCEN：/CC允许信号，若为低电平，则表示/CC有效，否则，不管/CC是什么状态，测试条件为永真。

/RLD：寄存器/计数器装入信号，当为低电平时，不管AM2910所执行的命令和测试条件如何，都强制把直接输入D11~D0装入。

/OE：Y输出允许信号，低电平有效，当为高电平时，Y输出为高阻态。

CP：时钟脉冲信号，由低变高的上升边沿触发所有内部状态的变化。

输出线：

Y11~Y0：下一条微指令的地址，它直接作为控制存储器的地址。

/FULL：微堆栈满信号，低电平有效。

/PL、/MAP、/VECT：3个使能信号，用于决定直接输入D的来源。

·AM2910的功能与具体用法

AM2910的功能是由命令码I3~I0，条件输入/CC、/CCEN以及计数器当前值组合的结果。

AM2910提供了16条命令，用来选择下一条将要执行的微指令的地址。其中，有4条命令（0、2、12、

14）是无条件命令，其功能仅取决于命令本身的功能；有3条命令（8、9、15）受到内部计数器当前值的控制；有10条命令（1、3、4、5、6、7、10、11、13、15）

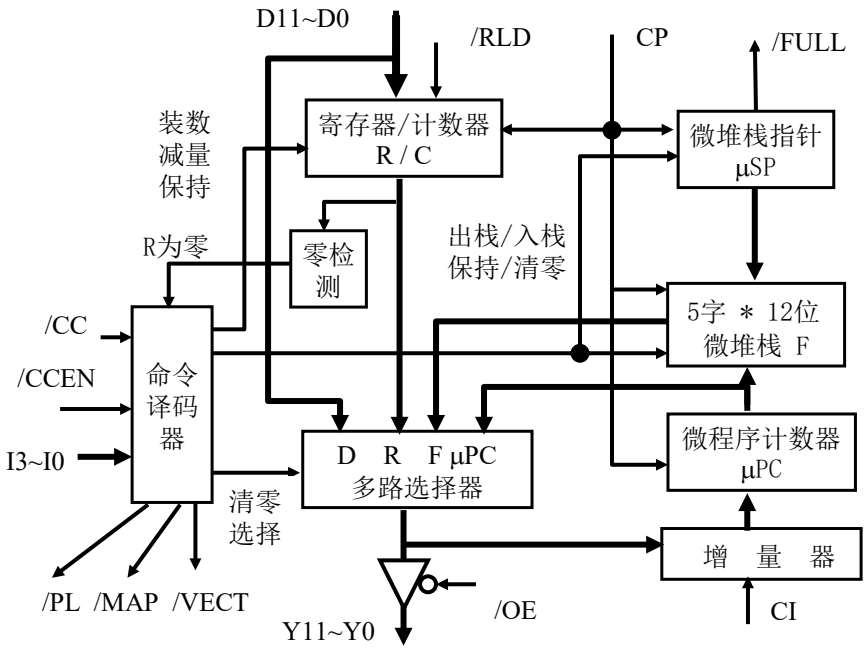


图 1 AM2910的结构图

受到条件输入（/CC和/CCEN）的控制。命令15同时受到条件输入和计数器当前值的控制，可以实现3路（F、D、μPC）转移。

现将要用到的2、3、4、8、14、15这6条命令的功能说明如下：

2号命令：无条件转MAP（JMAP）。

3号命令：条件转移。条件测试时，为假顺序执行，为真则按下地址D转移。

4号命令：进栈并条件装入计数器（PUSH）。

把下条微指令的地址压入堆栈中，同时若条件测试通过，则把当前微指令地址字段的内容（通常作为循环次数）装入计数器中，然后顺序执行。该命令通常是为建立微程序循环作准备。

8号命令：重复循环（RFCT）。

为了使用这条命令，前面必须有一条命令（通常是4号命令）把循环首地址压入堆栈，把循环次数装入计数器。该命令执行时，先检测计数器的值是否为零，若不为零，则计数器减一，并取微栈顶的内容作为下条微指令的地址（即转循环首地址继续循环），否则，把微堆栈指针减一（即弹出循环地址），顺序执行下条微指令。

14号命令：顺序执行。

15号命令：3路转移（TWB）

与8号命令类似，在该命令执行之前，前面必须有一命令（如4号命令）已把转移地址压入堆栈，把循环计数值装入计数器。该命令同时受到条件输入和计数器当前值的控制。当计数器不为零时，计数器减一，同时，若条件测试通过，则退栈（即弹出转移地址），并选择μPC作为下条微指令的地址，否则，选择微栈顶作为下条微指令的地址；当计数器为零时，微堆栈指针减一（即退栈），此时，若条件测试通过，则下条微指令的地址来自μPC，否则，来自当前微指令的下地址字段。

在TEC-2机的实现中，已把AM2910的/OE端接地，使其输出Y11~Y0总是有效的（实用Y9~Y0共10位）。

并把/CCEN接地，使AM2910的条件判断结果一定取决于/CC。把CI接电源，使μPC+1总是执行的。用/VECT信号把通过水平板上开关给出的10位的微指令的入口地址接通到AM2910的D输入端。

关于/CC条件码的形成，主要是通过微码位SCC和SC来选择判断条件，其具体规定见下表3。

表3　　/CC条件码的形成

SCC B39 B38 B37	SC或 IR10~IR8	/CC	SCC B39 B38 B37	SC或 IR10~IR8	/CC
0		0	7	IR10~IR8=000	/C
1		1		IR10~IR8=001	/Z
2	SC=0	/FS1		IR10~IR8=010	/V
3	SC=0	/FS2		IR10~IR8=011	/S
4	SC=0	/FS3		IR10~IR8=100	C
5	SC=0	/WAIT		IR10~IR8=101	Z
2	SC=1	/C		IR10~IR8=110	V
3	SC=1	/Z		IR10~IR8=111	S
4	SC=1	/V			
5	SC=1	/S			
6		/INT			

表3中的FSl、FS2和FS3是水平板上的3个功能开关，用于选择TEC-2机执行不同的功能，其具体规定如表4所示（功能选择）：

WAIT是TEC-2处于单步执行时，单步控制线路的等待状态（等待按下STEP微型按键）经控制线路（Gal2）给出的信号。

表 4

FS1	FS2	FS3	FS4	执行功能
×	×	×	1	运算器实验
0	0	0	0	装入微程序
0	0	1	0	执行微程序
0	1	0	0	存储器写（单步）
0	1	1	0	存储器读（单步）
1	0	0	0	存储器读（连续）
1	0	1	0	从 0 地址连续执行
1	1	0	0	从指定地址单步执行
1	1	1	0	从指定地址连续执行

C、Z、V、S或它们的取反值/C、/Z、/V、/S是运算器中的4个状态标志位。当SCC的3位微码为111时，通过条件转移指令的指令寄存器的第10~8位选择它们，以形成条件码/CC的值。

当SC为1时，通过SCC三位编码的2、3、4和5状态选择/C、/Z、/V、/S形成条件码CC的值，用于非条件转移指令的微指令中。

/INT为中断请求信号，低电平有效，在每条指令结束时，判有无中断请求，以确定转中断处理还是执行下一条指令。

/CC条件码的形成是用一片Gal20v8器件实现的。