目的与要求

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。

要进行这项大型实验，必须清楚地懂得：

1. TEC-2机的功能部件及其连接关系；
2. TEC-2机每个功能部件的功能与具体组成；
3. TEC-2机支持的指令格式；
4. TEC-2机的微指令格式，AM2910芯片的用法；
5. 已实现的典型指令的执行实例，即相应的微指令与其执行次序的安排与衔接；
6. 要实现的新指令的格式与功能。

实验环境

$TEC-2$模拟机

具体内容

请选定指令格式、操作码，按照要求，设计三条指令。

指令一

- **说明**

  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。

  三字指令（控存入口`110H`）。

- **格式**

  `D8××,ADDR1,ADDR2`

- **功能**

  `[ADDR1]=[ADDR1]-[ADDR2]`

微程序设计与实现

执行顺序	微指令	说明
1	$PC \rightarrow AR, PC+1 \rightarrow PC$	为读取第1个操作数的地址做准备
2	$MEM \rightarrow R6$	读取第1个操作数的地址送入R_6
3	$PC \rightarrow AR, PC+1 \rightarrow PC$	为读取第2个操作数的地址做准备
4	$MEM \rightarrow AR$	读取第2个操作数的地址送入AR
5	$R6 \rightarrow AR, MEM \rightarrow R6$	第1个操作数地址送AR，读取第2个操作数送入R_6
6	$MEM - R6 \rightarrow R6$	第1个操作数-第2个操作数送入R_6
7	$R6 \rightarrow MEM$	R_6送第1个操作数所在存储单元

1. $PC \rightarrow AR, PC+1 \rightarrow PC$

```
0000 0E00 A0B5 5402
```

	八								七								六				五							
说明	高8位无用								微指令转移时，转移至下址								备用	用来选择下一条将要执行的微指令的地址			决定CC*如何取值				备用	set status决定标志位取值方式		
									下址								备用	AM2910命令码 CI 3-0			SCC			SC	备用	SST		
微指令	B63	B62	B61	B60	B59	B58	B57	B56	B55	B54	B53	B52	B51	B50	B49	B48	B47	B46	B45	B44	B43	B42	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32
二进制	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
十六进制	0				0				0				0				0				E				0				0			
	四								三								二								一							
说明	是否访问主存或外	AM2901寄存器结果和Y输出选择			访问内存还是IO	运算功能选择			写还是读	运算数来源选择			寄存器选择				寄存器选择				进位选择		移位控制		A口地址选择	选择送往内部总线IB的数据			B口地址选择	选择接受IB数据的寄存器		
	!!MIO	MI 8-6			REQ	MI 5-3			!!WE	MI 2-0			A口				B口				SCI		SSH		SA	DC1			SB	DC2		
微指令	B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
二进制	1	0	1	0	0	0	0	0	x	0	1	1	1	0	0	1	1	0	0	1	0	1	x	x	0	x	x	x	0	0	1	0
十六进制	A				0				B				5				5				4				0				2			

- 七

  顺序执行，下址字段任意取值即可

- 六

  $AM2910$命令码为14，即14号指令，顺序执行

- 五

  - SCC、SC

    顺序执行时不需要条件测试，不需要使用$C\bar{C}$，所以这4位任意取值

  - SST

    此时并非真正的运算，并且第6条微指令才是真正的运算，所以这3位任意即可

- 四、三

  - $M\bar{I}O$、REQ、$W\bar{E}$

    不需要读写存储器或输入输出设备，所以为`10x`

  - MI_{8-6}

    运算结果送B口，运算器Y输出A口

- MI_{5-3}

  选择加法运算

- MI_{2-0}

  运算数选择B口和0，（选A口和0也可以，因为A口和B口是同一个寄存器）

- A口

  选PC，即R_5

- 二

  - B口

    选PC，即R_5

  - SCI

    最低位进位设置为1

  - SSH

    不移位

- 一

  - SA、SB

    选微指令中的A口、B口地址

  - $DC1$

    因为AR只能接收来自运算器的结果输出信号，所以$DC1$可以任意

  - $DC2$

    运算器输出送AR

2. $MEM \rightarrow R6$

```
0000 0E00 30F0 6000
```

	八								七								六				五							
说明	高8位无用								微指令转移时，转移至下址								备用	用来选择下一条将要执行的微指令的地址			决定CC*如何取值				备用	set status决定标志位取值方式		
									下址								备用	AM2910命令码 CI 3-0			SCC			SC	备用	SST		
微指令	B63	B62	B61	B60	B59	B58	B57	B56	B55	B54	B53	B52	B51	B50	B49	B48	B47	B46	B45	B44	B43	B42	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32
二进制	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
十六进制	0				0				0				0				0				E				0				0			
	四								三								二								一							
说明	是否访问主存或外	AM2901寄存器结果和Y输出选择			访问内存还是IO	运算功能选择			写还是读	运算数来源选择			寄存器选择				寄存器选择				进位选择		移位控制		A口地址选择	选择送往内部总线IB的数据			B口地址选择	选择接受IB数据的寄存器		
	!!MIO	MI 8-6			REQ	MI 5-3			!!WE	MI 2-0			A口				B口				SCI		SSH		SA	DC1			SB	DC2		
微指令	B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
二进制	0	0	1	1	0	0	0	0	1	1	1	1	x	x	x	x	0	1	1	0	0	0	x	x	x	x	x	x	0	0	0	0
十六进制	3				0				F				0				6				0				0				0			

- 七

  顺序执行，下址字段任意取值即可

- 六

  $AM2910$命令码为14，即14号指令，顺序执行

- 五

    - SCC、SC

      顺序执行时不需要条件测试，不需要使用$C\bar{C}$，所以这4位任意取值
    - SST

      此时并非真正的运算，并且第6条微指令才是真正的运算，所以这3位任意即可
- 四、三
    - $M\bar{I}O$、REQ、$W\bar{E}$

      读存储器
    - MI_{8-6}

      运算结果送B口，运算器Y输出运算结果（并没有使用）
    - MI_{5-3}

      选择加法运算
    - MI_{2-0}

      运算数选择D和0
    - A口

      任意
- 二
    - B口

      选$R6$
    - SCI

      最低位进位设置为0
    - SSH

      不移位
- 一
    - SA、SB

      A口任意，因为不使用A口；选择微指令中的B口地址
    - $DC1$

      任意
    - $DC2$

      不控制

3. $PC \rightarrow AR, PC + 1 \rightarrow PC$

```
0000 0E00 A0B5 5402
```

这条微指令和本机器指令的第一条微指令完全一样，不再做解释说明

4. $MEM \rightarrow AR$

```
0000 0E00 10F0 0002
```

	八								七								六				五							
说明	高8位无用								微指令转移时，转移至下址								备用	用来选择下一条将要执行的微指令的地址			决定CC*如何取值				备用	set status决定标志位取值方式		
									下址								备用	AM2910命令码 CI 3-0			SCC			SC	备用	SST		
微指令	B63	B62	B61	B60	B59	B58	B57	B56	B55	B54	B53	B52	B51	B50	B49	B48	B47	B46	B45	B44	B43	B42	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32
二进制	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
十六进制	0				0				0				0				0				E				0				0			
	四								三								二								一							
说明	是否访问主存或外	AM2901寄存器结果和Y输出选择			访问内存还是IO	运算功能选择			写还是读	运算数来源选择			寄存器选择				寄存器选择				进位选择		移位控制		A口地址选择	选择送往内部总线IB的数据			B口地址选择	选择接受IB数据的寄存器		
	!!MIO	MI 8-6			REQ	MI 5-3			!!WE	MI 2-0			A口				B口				SCI		SSH		SA	DC1			SB	DC2		
微指令	B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
二进制	0	0	0	1	0	0	0	0	1	1	1	1	x	x	x	x	x	x	x	x	0	0	x	x	x	x	x	x	x	0	1	0
十六进制	1				0				F				0				0				0				0				2			

- 七

  顺序执行，下址字段任意取值即可

- 六

  $AM2910$命令码为14，即14号指令，顺序执行

- 五
  - SCC、SC

    顺序执行时不需要条件测试，不需要使用$C\bar{C}$，所以这4位任意取值
  - SST

    此时并非真正的运算，并且第6条微指令才是真正的运算，所以这3位任意即可
- 四、三
  - $M\bar{I}O$、REQ、$W\bar{E}$

    读存储器
  - MI_{8-6}

    运算结果不送寄存器，运算器Y输出运算结果送AR
  - MI_{5-3}

    选择加法运算
  - MI_{2-0}

    运算数选择D和0
  - A口

    任意
- 二
  - B口

    任意
  - SCI

    最低位进位设置为0
  - SSH

    不移位
- 一
  - SA、SB

    都任意，因为不使用A口和B口

- $DC1$

  任意

- $DC2$

  运算器输出送AR

5. $R6 \rightarrow AR, MEM \rightarrow R6$

```
0000 0E00 20F6 6002
```

	八								七								六								五							
说明	高8位无用								微指令转移时，转移至下址								备用		用来选择下一条将要执行的微指令的地址				决定CC*如何取值				备用	set status决定标志位取值方式		
									下址								备用		AM2910命令码 CI 3-0				SCC			SC	备用	SST		
微指令	B63	B62	B61	B60	B59	B58	B57	B56	B55	B54	B53	B52	B51	B50	B49	B48	B47	B46	B45	B44	B43	B42	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32
二进制	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
十六进制	0				0				0				0				0				E				0				0			
	四								三								二								一							
说明	是否访问主存或外设	AM2901寄存器结果和Y输出选择			访问内存还是IO	运算功能选择			写还是读	运算数来源选择		寄存器选择					寄存器选择				进位选择		移位控制		A口地址选择	选择送往内部总线IB的数据			B口地址选择	选择接受IB数据的寄存器		
	!!MIO	MI 8-6			REQ	MI 5-3			!!WE	MI 2-0			A口				B口				SCI		SSH		SA	DC1			SB	DC2		
微指令	B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
二进制	0	0	1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	1	1	0	0	0	x	x	0	x	x	x	0	0	1	0
十六进制	2				0				F				6				6				0				0				2			

- 七

  顺序执行，下址字段任意取值即可

- 六

  $AM2910$命令码为14，即14号指令，顺序执行

- 五

  - SCC、SC

    顺序执行时不需要条件测试，不需要使用$C\bar{C}$，所以这4位任意取值

  - SST

    此时并非真正的运算，并且第6条微指令才是真正的运算，所以这3位任意即可

- 四、三

  - $M\bar{I}O$、REQ、$W\bar{E}$

    读存储器

  - MI_{8-6}

    运算结果送B口，运算器Y输出A口

  - MI_{5-3}

    选择加法运算

  - MI_{2-0}

    运算数选择D和0

  - A口

    R_6

- 二

  - B口

R_6

- SCI

  最低位进位设置为0

- SSH

  不移位

- 一

  - SA、SB

    使用微指令中的A口和B口

  - $DC1$

    任意

  - $DC2$

    运算器输出送AR

6. $MEM-R6\rightarrow R6$

```
0000 0E01 22D6 6000
```

	八								七								六				五							
说明	高8位无用								微指令转移时，转移至下址								备用	用来选择下一条将要执行的微指令的地址			决定CC*如何取值				备用	set status决定标志位取值方式		
									下址								备用	AM2910命令码 CI 3-0			SCC			SC	备用	SST		
微指令	B63	B62	B61	B60	B59	B58	B57	B56	B55	B54	B53	B52	B51	B50	B49	B48	B47	B46	B45	B44	B43	B42	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32
二进制	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
十六进制	0				0				0				0				0				E				0				1			
	四								三								二								一							
说明	是否访问主存或外设	AM2901寄存器结果和Y输出选择			访问内存还是IO	运算功能选择			写还是读	运算数来源选择			寄存器选择				寄存器选择				进位选择		移位控制		A口地址选择	选择送往内部总线IB的数据			B口地址选择	选择接受IB数据的寄存器		
	!!MIO	MI 8-6			REQ	MI 5-3			!!WE	MI 2-0			A口				B口				SCI		SSH		SA	DC1			SB	DC2		
微指令	B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
二进制	0	0	1	0	0	0	1	0	1	1	0	1	0	1	1	0	0	1	1	0	0	0	x	x	0	x	x	x	0	0	0	0
十六进制	2				2				D				6				6				0				0				0			

- 七

  顺序执行，下址字段任意取值即可

- 六

  $AM2910$命令码为14，即14号指令，顺序执行

- 五

  - SCC、SC

    顺序执行时不需要条件测试，不需要使用$C\bar{C}$，所以这4位任意取值

  - SST

    此时是真正的运算，设置标志位

- 四、三

  - $M\bar{I}O$、REQ、$W\bar{E}$

    读存储器

  - MI_{8-6}

    运算结果送B口；运算器Y输出A口（并没有使用）

- MI_{5-3}

  选择减法运算

- MI_{2-0}

  运算数选择D和A口

- A口

  R_6

- 二

  - B口

    R_6

  - SCI

    最低位进位设置为0

  - SSH

    不移位

- 一

  - SA、SB

    使用微指令中的A口和B口

  - $DC1$

    任意

  - $DC2$

    不控制

7. $R6 \rightarrow MEM$

```
0029 0300 1046 0010
```

	八								七								六								五							
说明	高8位无用								微指令转移时，转移至下址								备用		用来选择下一条将要执行的微指令的地址				决定CC*如何取值				备用	set status决定标志位取值方式		
									下址								备用		AM2910命令码 CI 3-0				SCC			SC	备用	SST		
微指令	B63	B62	B61	B60	B59	B58	B57	B56	B55	B54	B53	B52	B51	B50	B49	B48	B47	B46	B45	B44	B43	B42	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32
二进制	x	x	x	x	x	x	x	x	0	0	1	0	1	0	0	1	0	0	x	x	0	0	1	1	0	0	0	x	x	0	0	0
十六进制	0				0				2				9				0				3				0				0			
	四								三								二								一							
说明	是否访问主存或外设	AM2901寄存器结果和Y输出选择			访问内存还是IO	运算功能选择			写还是读	运算数来源选择			寄存器选择				寄存器选择				进位选择		移位控制		A口地址选择	选择送往内部总线IB的数据			B口地址选择	选择接受IB数据的寄存器		
	!!MIO	MI 8-6			REQ	MI 5-3			!!WE	MI 2-0			A口				B口				SCI		SSH		SA	DC1			SB	DC2		
微指令	B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
二进制	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	x	x	x	x	0	0	x	x	0	0	0	1	x	0	0	0
十六进制	1				0				4				6				0				0				1				0			

- 七

  这是最后一条微指令，下一条微指令是`A4H`。

- 六

  $AM2910$命令码为3，即3号指令，条件转移

- 五

  - SCC、SC

$\bar{CC}$设为0，进行转移

- SST

  此时并不是运算，所以标志位不变

- 四、三
  - $\bar{MIO}$、REQ、$\bar{WE}$

    写存储器

  - MI_{8-6}

    运算结果不送寄存器；运算器Y输出A口

  - MI_{5-3}

    选择加法运算

  - MI_{2-0}

    运算数选择0和A口

  - A口

    R_6

- 二
  - B口

    任意

  - SCI

    最低位进位设置为0

  - SSH

    不移位

- 一
  - SA、SB

    使用微指令中的A口；B口任意

  - $DC1$

    运算器输出送IB

  - $DC2$

    不控制

指令测试

如下图所示，`[A00H]`存储单元（$ADDR1$）存储了`0045H`，`[A01H]`存储单元（$ADDR2$）存储了`0023H`，最终指令运行后`[A00H]`为`0022H`，实现`[ADDR1]=[ADDR1]-[ADDR2]`。

```
TEC-2 CRT MONITOR
Version 1.2 ,Jun.2005

>E900
0900    0000:0000  0000:0E00  0000:A0B5  0000:5402  0000:0000
0905    0000:0E00  0000:30F0  0000:6000  0000:0000  0000:0E00
090A    0000:A0B5  0000:5402  0000:0000  0000:0E00  0000:10F0
090F    0000:0002  0000:0000  0000:0E00  0000:20F6  0000:6002
0914    0000:0000  0000:0E01  0000:22D6  0000:6000  0000:0029
0919    0000:0300  0000:1046  0000:0010
>D900
0900    0000  0E00  A0B5  5402  0000  0E00  30F0  6000  ......T.....0.'.
0908    0000  0E00  A0B5  5402  0000  0E00  10F0  0002  ......T.....;.;.
0910    0000  0E00  20F6  6002  0000  0E01  22D6  6000  ..... '.....".'.
0918    0029  0300  1046  0010  0000  0000  0000  0000  .)...F..........
0920    0000  0000  0000  0000  0000  0000  0000  0000  ................
0928    0000  0000  0000  0000  0000  0000  0000  0000  ................
0930    0000  0000  0000  0000  0000  0000  0000  0000  ................
0938    0000  0000  0000  0000  0000  0000  0000  0000  ................
0940    0000  0000  0000  0000  0000  0000  0000  0000  ................
0948    0000  0000  0000  0000  0000  0000  0000  0000  ................
0950    0000  0000  0000  0000  0000  0000  0000  0000  ................
0958    0000  0000  0000  0000  0000  0000  0000  0000  ................
0960    0000  0000  0000  0000  0000  0000  0000  0000  ................
0968    0000  0000  0000  0000  0000  0000  0000  0000  ................rsity
0970    0000  0000  0000  0000  0000  0000  0000  0000  ................
>A800
0800: MOV R1,900
0802: MOV R2,7
0804: MOV R3,110
0806: LDMC
0807: RET
0808:
>G800
>A820
0820: MOV R0,0045
0822: MOV [A001],R0
0824: MOV R1,0023
0826: MOV [A011],R1
0828: NOP
0829: NOP
082A: NOP
082B: RET
082C:
>E828
0828    0000:D800  0000:0A00  0000:0A01
>U820
0820: 2C00 0045  MOV    R0,     0045
0822: 3400 0A00  MOV    [0A00], R0
0824: 2C10 0023  MOV    R1,     0023
0826: 3401 0A01  MOV    [0A01], R1
0828: D800       DW     D800
0829: 0A00       ADC    R0,     R0
```

```
082A: 0A01       ADC    R0,     R1
082B: AC00       RET
082C: 0000       NOP
082D: 0000       NOP
082E: 0000       NOP
082F: 0000       NOP
0830: 0000       NOP
0831: 0000       NOP
0832: 0000       NOP
0833: 0000       NOP
>G820
>DA00
0A00    0022  0023  0000  0000  0000  0000  0000  0000  ."#.............
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  ................
>
```

指令二

- **说明**

  将一通用寄存器内容加上某内存单元内容，结果放在另一寄存器中。

  双字指令（控存入口`130H`），SR和DR分别为源、目的寄存器（各4位）。

- **格式**

  `E0 DR SR,ADDR`

- **功能**

  `DR=SR+ [ADDR]`

微程序设计与实现

执行顺序	微指令	说明
1	$PC \rightarrow AR, PC+1 \rightarrow PC$	为读取操作数的地址做准备
2	$MEM \rightarrow AR$	操作数的地址送入AR
3	$SR+MEM \rightarrow DR$	计算SR与操作数的和存入DR

1. $PC \rightarrow AR, PC+1 \rightarrow PC$

```
0000 0E00 A0B5 5402
```

这条微指令和第一条机器指令的第一条微指令完全一样，不再做解释说明

2. $MEM \rightarrow AR$

```
0000 0E00 10F0 0002
```

这条微指令和第一条机器指令的第四条微指令完全一样，不再做解释说明

3. $SR+MEM \rightarrow DR$

```
0029 0301 30D0 0088
```

	八								七								六								五							
说明	高8位无用								微指令转移时，转移至下址								备用	用来选择下一条将要执行的微指令的地址							决定CC*如何取值				备用	set status决定标志位取值方式			
									下址								备用	AM2910命令码 CI 3-0							SCC			SC	备用	SST			
微指令	B63	B62	B61	B60	B59	B58	B57	B56	B55	B54	B53	B52	B51	B50	B49	B48	B47	B46	B45	B44	B43	B42	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32
二进制	x	x	x	x	x	x	x	x	0	0	1	0	1	0	0	1	0	0	x	x	0	0	1	1	0	0	0	x	x	0	0	1
十六进制	0				0				2				9				0				3				0				1			
	四								三								二								一							
说明	是否访问主存或外设	AM2901寄存器结果和Y输出选择			访问内存还是IO	运算功能选择			写还是读	运算数来源选择			寄存器选择				寄存器选择				进位选择		移位控制		A口地址选择	选择送往内部总线IB的数据			B口地址选择	选择接受IB数据的寄存器			
	!!MIO	MI 8-6			REQ	MI 5-3			!!WE	MI 2-0			A口				B口				SCI		SSH		SA	DC1			SB	DC2			
微指令	B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
二进制	0	0	1	1	0	0	0	0	1	1	0	1	x	x	x	x	x	x	x	x	0	0	x	x	1	x	x	x	1	0	0	0
十六进制	3				0				D				0				0				0				8				8			

- 七

  这是最后一条微指令，下一条微指令是`A4H`。

- 六

  $AM2910$命令码为3，即3号指令，条件转移

- 五

  - SCC、SC

    $\bar{C}C$设为0，进行转移

  - SST

    此时进行了运算，需要对标志位进行设置

- 四、三

  - $M\bar{I}O$、REQ、$W\bar{E}$

读存储器

- MI_{8-6}

  运算结果送B口；运算器Y输出运算结果（并没有使用）

- MI_{5-3}

  选择加法运算

- MI_{2-0}

  运算数选择D和A口

- A口

  使用机器指令中设置的SR，微指令中不需要给A口地址

- 二

  - B口

    使用机器指令中设置的DR，微指令中不需要给B口地址

  - SCI

    最低位进位设置为0

  - SSH

    不移位

- 一

  - SA、SB

    使用机器指令中的SR、DR

  - $DC1$

    任意

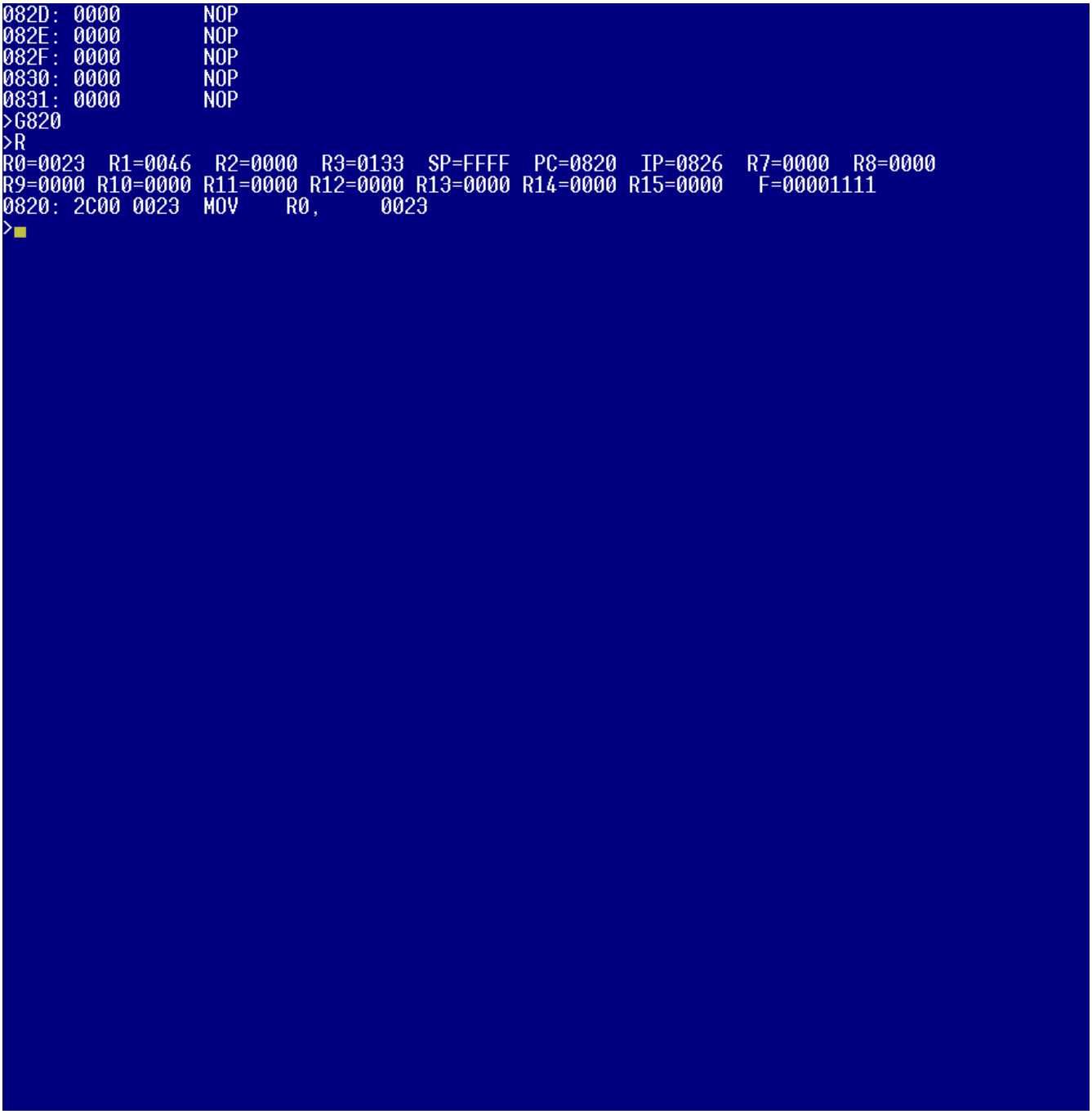
  - $DC2$

    不控制

指令测试

如图示，R_0（SR）为`0023H`，`A00H`（$ADDR$）存储`0023H`，程序运行后R_1（DR）为`0046H`，实现`DR=SR+[ADDR]`。

```
TEC-2 CRT MONITOR
Version 1.2 ,Jun.2005

>E900
0900    0000:0000  0000:0E00  0000:A0B5  0000:5402  0000:0000
0905    0000:0E00  0000:10F0  0000:0002  0000:0029  0000:0301
090A    0000:30D0  0000:0088
>D900
0900    0000  0E00  A0B5  5402  0000  0E00  10F0  0002  .....T.........
0908    0029  0301  30D0  0088  0000  0000  0000  0000  .)..0...........
0910    0000  0000  0000  0000  0000  0000  0000  0000  ................
0918    0000  0000  0000  0000  0000  0000  0000  0000  ................
0920    0000  0000  0000  0000  0000  0000  0000  0000  ................
0928    0000  0000  0000  0000  0000  0000  0000  0000  ................
0930    0000  0000  0000  0000  0000  0000  0000  0000  ................
0938    0000  0000  0000  0000  0000  0000  0000  0000  ................
0940    0000  0000  0000  0000  0000  0000  0000  0000  ................
0948    0000  0000  0000  0000  0000  0000  0000  0000  ................
0950    0000  0000  0000  0000  0000  0000  0000  0000  ................
0958    0000  0000  0000  0000  0000  0000  0000  0000  ................
0960    0000  0000  0000  0000  0000  0000  0000  0000  ................
0968    0000  0000  0000  0000  0000  0000  0000  0000  ................
0970    0000  0000  0000  0000  0000  0000  0000  0000  ................
>A800
0800: MOV R1,900tual TEC-2 By Guiheng Zhou, Jun. 2005, Sun Yat-sen University
0802: MOV R2,3
0804: MOV R3,130
0806: LDMC
0807: RET
0808:
>G800
>A820
0820: MOV R0,0023
0822: MOV [A00],R0
0824: NOP
0825: NOP
0826: RET
0827:
>E824
0824    0000:E010  0000:0A00
>U820
0820: 2C00 0023  MOV    R0,     0023
0822: 3400 0A00  MOV    [0A00], R0
0824: E010       DW     E010
0825: 0A00       ADC    R0,     R0
0826: AC00       RET
0827: 0000       NOP
0828: 0000       NOP
0829: 0000       NOP
082A: 0000       NOP
082B: 0000       NOP
082C: 0000       NOP
```



指令三

- **说明**
  - 转移指令

    判断两个通用寄存器内容是否相等，若相等则转移到指定绝对地址，否则顺序执行。
  - 双字指令（控存入口`140H`），`SR`和`DR`分别为源、目的寄存器（各4位），`ADDR`为绝对地址。
  - 提示

    利用指令的`CND`字段，即IR_{10-8}，令$IR_{10-8}=101$，即$C\bar{C}=Z$。

    当`DR==SR`时`Z=1`，微程序不跳转，接着执行$MEM\rightarrow PC$（即$ADDR\rightarrow PC$）；

    而当`DR!=SR`时`Z=0`，微程序跳转至`A4H`。

- **格式**

  `E5 DR SR,ADDR`

- **功能**

  `if DR==SR goto ADDR` `else` 顺序执行

微程序设计与实现

执行顺序	微指令	说明
1	$SR-DR$	测试SR与DR是否相等
2	$PC\rightarrow AR, PC+1\rightarrow PC$，如果$Z=0$，则微程序跳转到$A4H$	为读取转移地址做准备
3	$MEM\rightarrow PC$	程序转移到指定地址

1. $SR-DR$

```
0000 0E01 9210 0088
```

	八								七								六								五							
说明	高8位无用								微指令转移时，转移至下址								备用	用来选择下一条将要执行的微指令的地址				决定CC*如何取值				备用	set status决定标志位取值方式		
									下址								备用	AM2910命令码 CI 3-0				SCC			SC	备用	SST		
微指令	B63	B62	B61	B60	B59	B58	B57	B56	B55	B54	B53	B52	B51	B50	B49	B48	B47	B46	B45	B44	B43	B42	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32
二进制	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
十六进制	0				0				0				0				0				E				0				1			
	四								三								二								一							
说明	是否访问主存或外设	AM2901寄存器结果和Y输出选择			访问内存还是IO	运算功能选择			写还是读	运算数来源选择			寄存器选择				寄存器选择				进位选择		移位控制		A口地址选择	选择送往内部总线IB的数据			B口地址选择	选择接受IB数据的寄存器		
	!!MIO	MI 8-6			REQ	MI 5-3			!!WE	MI 2-0			A口				B口				SCI		SSH		SA	DC1			SB	DC2		
微指令	B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
二进制	1	0	0	1	0	0	1	0	x	0	0	1	x	x	x	x	x	x	x	x	0	0	x	x	1	x	x	x	1	0	0	0
十六进制	9				2				1				0				0				0				8				8			

- 七

  顺序执行，下址字段任意

- 六

  $AM2910$命令码为14，即14号指令，顺序执行

- 五

  - SCC、SC

    顺序执行，不需要条件测试

  - SST

    此时进行了运算，需要对标志位进行设置，来判断SR和DR是否相等

- 四、三

  - $M\bar{I}O$、REQ、$W\bar{E}$

    不进行存储器或IO操作

  - MI_{8-6}

    运算结果不送寄存器；运算器Y输出运算结果（并没有使用）

  - MI_{5-3}

选择减法运算

- MI_{2-0}

  运算数选择A口和B口

- A口

  使用机器指令中设置的SR，微指令中不需要给A口地址

- 二

  - B口

    使用机器指令中设置的DR，微指令中不需要给B口地址

  - SCI

    最低位进位设置为0

  - SSH

    不移位

- 一

  - SA、SB

    使用机器指令中的SR、DR

  - $DC1$

    任意

  - $DC2$

    不控制

2. $PC \to AR, PC+1 \to PC$，如果$Z=0$，则微程序跳转到$A4H$

```
0029 03E0 A045 5412
```

	八								七								六				五							
说明	高8位无用								微指令转移时，转移至下址								备用	用来选择下一条将要执行的微指令的地址			决定CC*如何取值				备用	set status决定标志位取值方式		
									下址								备用	AM2910命令码 CI 3-0			SCC			SC	备用	SST		
微指令	B63	B62	B61	B60	B59	B58	B57	B56	B55	B54	B53	B52	B51	B50	B49	B48	B47	B46	B45	B44	B43	B42	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32
二进制	x	x	x	x	x	x	x	x	0	0	1	0	1	0	0	1	0	0	x	x	0	0	1	1	1	1	1	x	x	0	0	0
十六进制	0				0				2				9				0				3				E				0			
	四								三								二								一							
说明	是否访问主存或外设	AM2901寄存器结果和Y输出选择			访问内存还是IO	运算功能选择			写还是读	运算数来源选择			寄存器选择				寄存器选择				进位选择		移位控制		A口地址选择	选择送往内部总线IB的数据			B口地址选择	选择接受IB数据的寄存器		
	!!MIO	MI 8-6			REQ	MI 5-3			!!WE	MI 2-0			A口				B口				SCI		SSH		SA	DC1			SB	DC2		
微指令	B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
二进制	1	0	1	0	0	0	0	0	x	1	0	0	0	1	0	1	0	1	0	1	0	1	x	x	0	0	0	1	0	0	1	0
十六进制	A				0				4				5				5				4				1				2			

- 七

  微程序转移，下址字段为`A4H`

- 六

  $AM2910$命令码为3，即3号指令，条件转移

- 五

  - SCC、SC

测试条件设置为IR_{10-8}，指令中将IR_{10-8}设为5，即$C\bar{C} = Z$

- SST

  此时并不是计算，状态位保持即可

- 四、三
  - $M\bar{I}O$、REQ、$W\bar{E}$

    不进行存储器或IO操作

  - MI_{8-6}

    运算结果送B口；运算器Y输出A口

  - MI_{5-3}

    选择加法运算

  - MI_{2-0}

    运算数选择A口和0

  - A口

    A口设置为PC，即R_5。

- 二
  - B口

    B口设置为PC，即R_5。

  - SCI

    最低位进位设置为1

  - SSH

    不移位

- 一
  - SA、SB

    使用微指令中的A口和B口地址

  - $DC1$

    由于是写AR，这里把运算器输出送IB。经过前两条机器指令的试验，实际上$DC1$可以任意。

  - $DC2$

    写AR

3. $MEM \rightarrow PC$

```
0029 0300 30F0 5000
```

	八								七										六				五							
说明	高8位无用								微指令转移时，转移至下址								备用		用来选择下一条将要执行的微指令的地址				决定CC*如何取值				备用	set status决定标志位取值方式		
									下址								备用		AM2910命令码 CI 3-0				SCC			SC	备用	SST		
微指令	B63	B62	B61	B60	B59	B58	B57	B56	B55	B54	B53	B52	B51	B50	B49	B48	B47	B46	B45	B44	B43	B42	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32
二进制	x	x	x	x	x	x	x	x	0	0	1	0	1	0	0	1	0	0	x	x	0	0	1	1	0	0	0	x	x	0	0	0
十六进制	0				0				2				9				0				3				0				0			
	四								三								二								一							
说明	是否访问主存或外设	AM2901寄存器结果和Y输出选择			访问内存还是IO	运算功能选择			写还是读	运算数来源选择			寄存器选择				寄存器选择				进位选择		移位控制		A口地址选择	选择送往内部总线IB的数据			B口地址选择	选择接受IB数据的寄存器		
	!!MIO	MI 8-6			REQ	MI 5-3			!!WE	MI 2-0			A口				B口				SCI		SSH		SA	DC1			SB	DC2		
微指令	B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
二进制	0	0	1	1	0	0	0	0	1	1	1	1	x	x	x	x	1	0	0	1	0	0	x	x	x	x	x	x	0	0	0	0
十六进制	3				0				F				0				5				0				0				0			

- 七

  微程序转移，下址字段为`A4H`

- 六

  $AM2910$命令码为3，即3号指令，条件转移

- 五

  - SCC、SC

    测试条件设置为0，一定转移

  - SST

    此时并不是计算，状态位保持即可

- 四、三

  - $M\bar{I}O$、REQ、$W\bar{E}$

    读存储器

  - MI_{8-6}

    运算结果送B口；运算器Y输出运算结果（并没有使用）

  - MI_{5-3}

    选择加法运算

  - MI_{2-0}

    运算数选择D和0

  - A口

    A口任意，并不使用。

- 二

  - B口

    B口设置为PC，即R_5。

  - SCI

    最低位进位设置为0

  - SSH

    不移位

- 一

  - SA、SB

    使用微指令中的B口地址，A口任意（因为不使用A口）。

- $DC1$

  任意

- $DC2$

  不控制

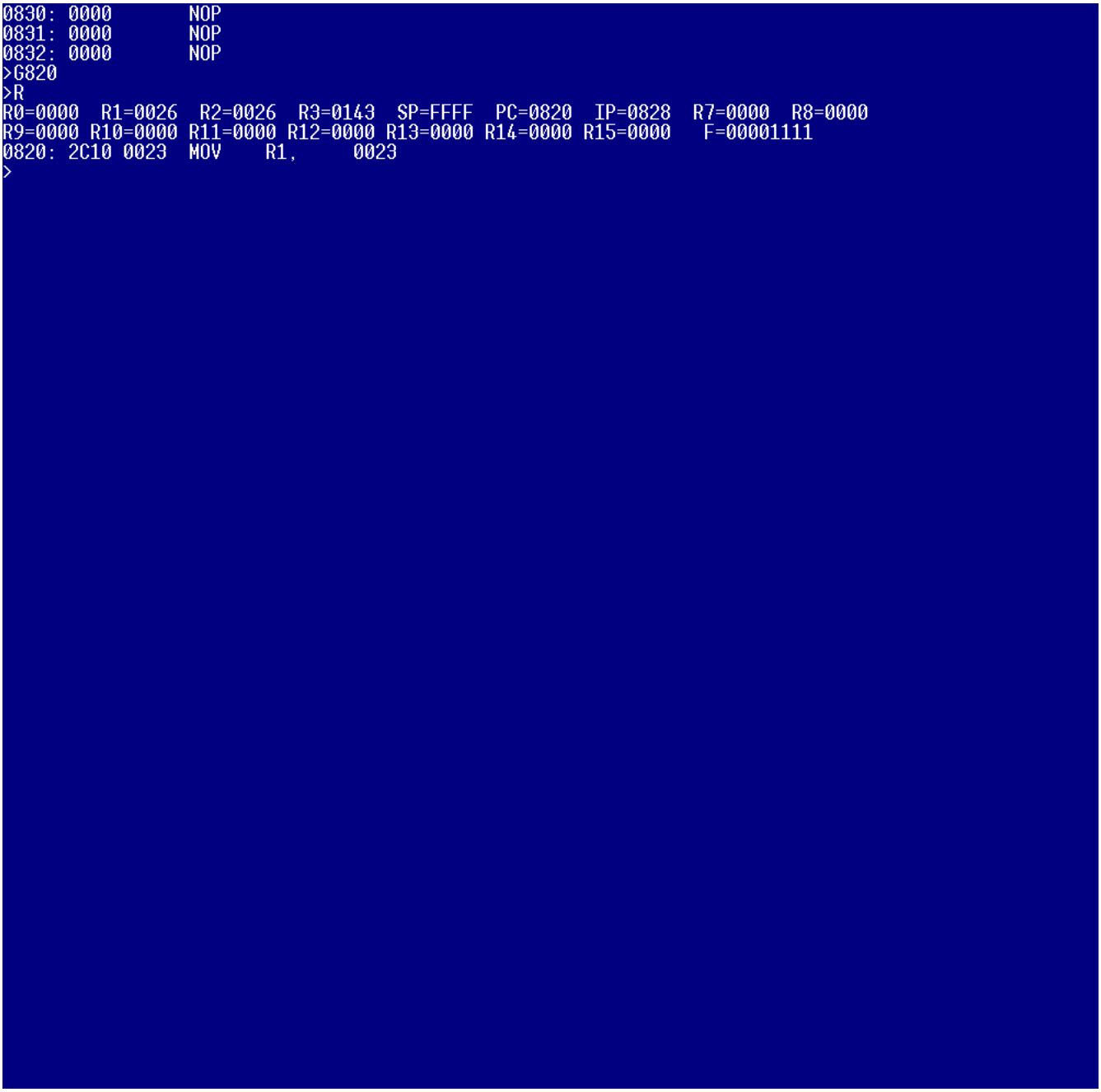
指令测试

当R_1（SR）存储`0023H`、R_2（DR）存储`0026H`时，即$DR \neq SR$时，程序顺序执行，运行了`MOV R1,0026`，所以程序运行后R_1为`0026H`。

当R_1（SR）存储`0023H`、R_2（DR）存储`0023H`时，即$DR = SR$时，程序跳转至`828H`（$ADDR$），所以未运行`MOV R1,0026`，所以程序运行后R_1仍为`0023H`。

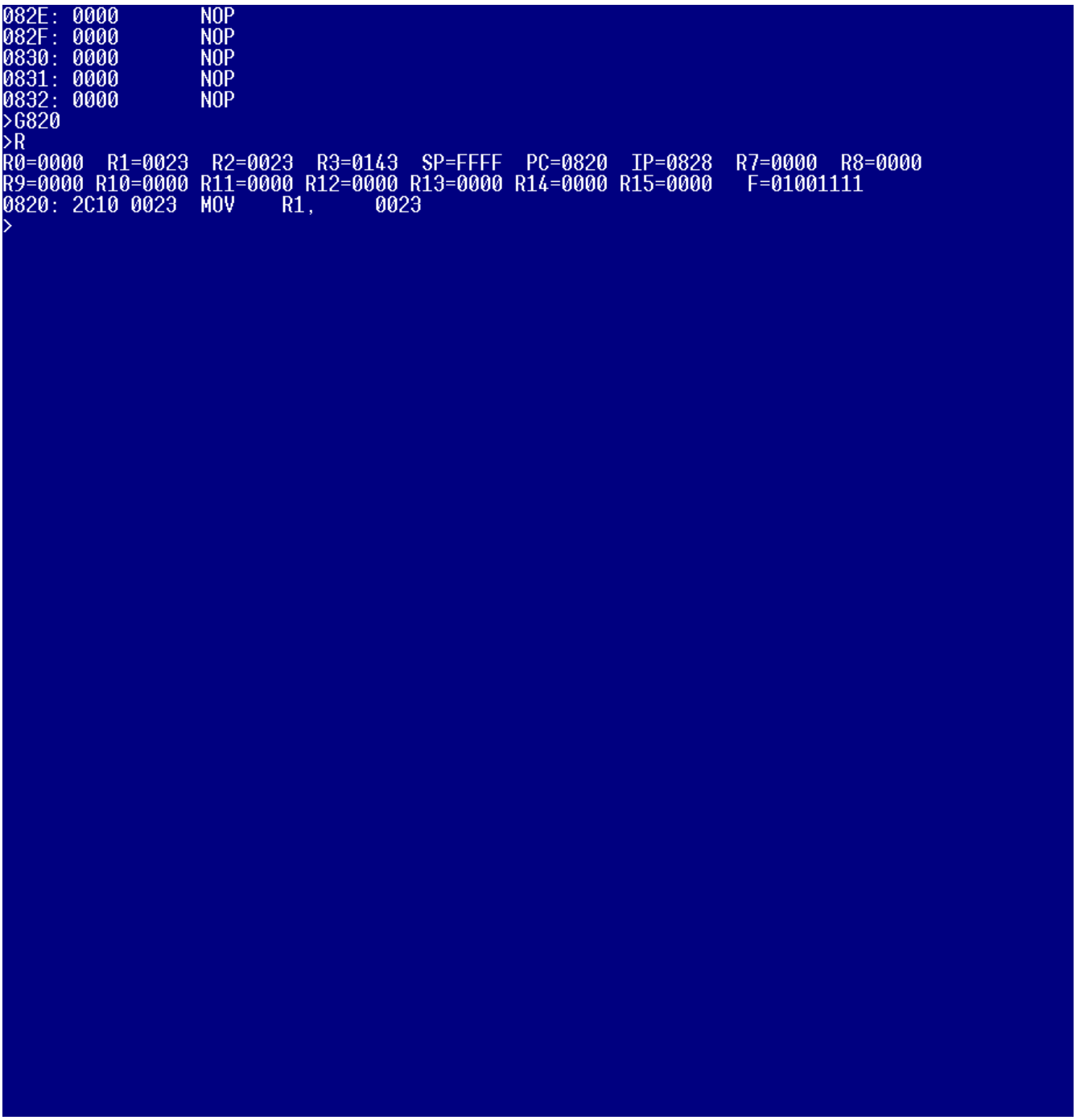
- $DR \neq SR$时

```
TEC-2 CRT MONITOR
Version 1.2 ,Jun.2005

>E900
0900    0000:0000  0000:0E01  0000:9210  0000:0088  0000:0029
0905    0000:03E0  0000:A045  0000:5412  0000:0029  0000:0300
090A    0000:30F0  0000:5000
>D900
0900    0000  0E01  9210  0088  0029  03E0  A045  5412  .........)...ET.
0908    0029  0300  30F0  5000  0000  0000  0000  0000  .)..0.P.........
0910    0000  0000  0000  0000  0000  0000  0000  0000  ................
0918    0000  0000  0000  0000  0000  0000  0000  0000  ................
0920    0000  0000  0000  0000  0000  0000  0000  0000  ................
0928    0000  0000  0000  0000  0000  0000  0000  0000  ................
0930    0000  0000  0000  0000  0000  0000  0000  0000  ................
0938    0000  0000  0000  0000  0000  0000  0000  0000  ................
0940    0000  0000  0000  0000  0000  0000  0000  0000  ................
0948    0000  0000  0000  0000  0000  0000  0000  0000  ................
0950    0000  0000  0000  0000  0000  0000  0000  0000  ................
0958    0000  0000  0000  0000  0000  0000  0000  0000  ................
0960    0000  0000  0000  0000  0000  0000  0000  0000  ................
0968    0000  0000  0000  0000  0000  0000  0000  0000  ................
0970    0000  0000  0000  0000  0000  0000  0000  0000  ................
>A800
0800: MOV R1,900tual TEC-2 By Guiheng Zhou, Jun. 2005, Sun Yat-sen University
0802: MOV R2,3
0804: MOV R3,140
0806: LDMC
0807: RET
0808:
>G800
>A820
0820: MOV R1,0023
0822: MOV R2,0026
0824: NOP
0825: NOP
0826: MOV R1,0026
0828: RET
0829:
>E824
0824    0000:E512  0000:0828
>U820
0820: 2C10 0023  MOV    R1,     0023
0822: 2C20 0026  MOV    R2,     0026
0824: E512       DW     E512
0825: 0828       ADC    R2,     R8
0826: 2C10 0026  MOV    R1,     0026
0828: AC00       RET
0829: 0000       NOP
082A: 0000       NOP
082B: 0000       NOP
082C: 0000       NOP
082D: 0000       NOP
082E: 0000       NOP
082F: 0000       NOP
```


0830: 0000 NOP
0831: 0000 NOP
0832: 0000 NOP
>G820
>R
R0=0000 R1=0026 R2=0026 R3=0143 SP=FFFF PC=0820 IP=0828 R7=0000 R8=0000
R9=0000 R10=0000 R11=0000 R12=0000 R13=0000 R14=0000 R15=0000 F=00001111
0820: 2C10 0023 MOV R1, 0023
>

- $DR = SR$时

```
TEC-2 CRT MONITOR
Version 1.2 ,Jun.2005

>E900
0900    0000:0000  0000:0E01  0000:9210  0000:0088  0000:0029
0905    0000:03E0  0000:A045  0000:5412  0000:0029  0000:0300
090A    0000:30F0  0000:5000
>D900
0900    0000  0E01  9210  0088  0029  03E0  A045  5412  ........)...ET.
0908    0029  0300  30F0  5000  0000  0000  0000  0000  .)..0.P.........
0910    0000  0000  0000  0000  0000  0000  0000  0000  ................
0918    0000  0000  0000  0000  0000  0000  0000  0000  ................
0920    0000  0000  0000  0000  0000  0000  0000  0000  ................
0928    0000  0000  0000  0000  0000  0000  0000  0000  ................
0930    0000  0000  0000  0000  0000  0000  0000  0000  ................
0938    0000  0000  0000  0000  0000  0000  0000  0000  ................
0940    0000  0000  0000  0000  0000  0000  0000  0000  ................
0948    0000  0000  0000  0000  0000  0000  0000  0000  ................
0950    0000  0000  0000  0000  0000  0000  0000  0000  ................
0958    0000  0000  0000  0000  0000  0000  0000  0000  ................
0960    0000  0000  0000  0000  0000  0000  0000  0000  ................
0968    0000  0000  0000  0000  0000  0000  0000  0000  ................
0970    0000  0000  0000  0000  0000  0000  0000  0000  ................
>A800
0800: MOV R1,900tual TEC-2 By Guiheng Zhou, Jun. 2005, Sun Yat-sen University
0802: MOV R2,3
0804: MOV R3,140
0806: LDMC
0807: RET
0808:
>G800
>A820
0820: MOV R1,0023
0822: MOV R2,0023
0824: NOP
0825: NOP
0826: MOV R1,0026
0828: RET
0829:
>E824
0824    0000:E512  0000:0828
>U820
0820: 2C10 0023  MOV    R1,     0023
0822: 2C20 0023  MOV    R2,     0023
0824: E512       DW     E512
0825: 0828       ADC    R2,     R8
0826: 2C10 0026  MOV    R1,     0026
0828: AC00       RET
0829: 0000       NOP
082A: 0000       NOP
082B: 0000       NOP
082C: 0000       NOP
082D: 0000       NOP
```



实验心得

1. 磨刀不误砍柴工。在设计微程序前，我把实验书看了一遍，有些地方不是很懂；之后着手去设计微程序，然后再去看书，理解到的更多了；最终微程序设计完之后，我又把书看了一遍，又发现了很多之前理解不到位的地方，正所谓温故而知新。我想这最能说明实践和理论的关系吧，多动手多动脑，才能巩固对知识的理解。
2. 通过本次对微指令码一位位的设计，我对于计算机的底层实现有了更深的理解，更进一步地了解了计算机的工作原理，特别是运算器、控制器、内存、寄存器等部件之间的协作。本学期也学了汇编语言，它几乎是最靠近计算机硬件的编程语言了，在本次课程设计中，自主设计硬件间的协作，又算是直接接触了硬件，巩固了汇编和计组两门学科的知识与其联系。
3. 通过本次课程设计，我巩固了微程序设计的方法与思想，比如微指令、微操作、微命令、水平型微指令、微控存等基本概念，了解了微程序层次上程序员所看到的的计算机系统结构，加深了对程序员一词的理解。不同层次的程序员由于工作的“机器”不同，所以面对的问题以及其实现往往也是极为不同的。

附加材料

以下为$TEC-2$部分知识点总结。

字长

$TEC-2$机字长为16位，运算器、主存、数据与地址总线均为16位。

指令

$TEC-2$机指令有6位操作码，故支持64条指令，其中53条已实现，用于写出该机的监控程序。

IR：指令寄存器，存储当前正在执行的指令。

指令最高6位（IR_{15-10}）是操作码，之后两位IR_{9-8}是条件码，把它用作条件转移指令的判断条件，因此可以认为这两位是指令的扩展操作码。除条件转移指令之外，其余指令不使用这两位。

主存

主存支持$64K$字，$4K\times 8$的$ROM(2732)$存放监控程序，$2K\times 8$的$RAM(2716)$存放用户数据及数据。

运算器

运算器主要由4片$AM2901$级联而成，可实现8种运算功能，16个双端口（A、B）读出、单端口（B）写入的通用寄存器（R_{0-15}，其中R_{4-6}作为SP、PC、IP），另配有1片$AM2902$实现快速进位。

- IP

  保存当前正在运行的指令的地址，**用于转移变址的目的。**

  所以在课设第一条指令中可以使用R_6暂存数据，因为此时不需要转移变址，用不到IP。

- I_{8-6}

  进行寄存器结果选择和Y输出选择

- I_{5-3}

  进行运算功能选择

- I_{2-0}

  进行数据来源选择

引脚信号

- D_{3-0}

  外部送给$AM2901$的数据信号，比如从内存读出来的数据

- Y_{3-0}

  $AM2901$向外送出的数据，受$\bar{OE}$控制。但在$TEC-2$中，$\bar{OE}$已接地

- A、B

选择寄存器组中的源与目的寄存器。

当A、B同值时，被选中的同一个寄存器的内容将被同时送到A、B两个数据输出端口

控制器

- 控制器主要由一片$AM2910$、7片$6116(RAM,2K\times 8)$（微控存）、16位的指令寄存器IR和2片2716（存储用于实现53条机器指令的微程序，加电后读取送入微控存）等组成。
- 程序计数器PC用运算器中的通用寄存器R_5代替，保存下一条指令的地址。
- 指令地址寄存器IP由运算器中的R_6代替。
- 控存字长56位，已实现的53条指令的微程序存放在2片单独的8位ROM中，加电的过程自动调入控存（装入微码）。
- 地址总线的输入信号仅有一组，即地址寄存器（AR），而**AR只能接收来自运算器的结果输出信号。**

$AM2910$

$AM2910$，微程序定序器，**作用：形成下一条微指令的地址。**

三个输出使能信号

作用：决定直接输入D的来源

- $M\bar{A}P$

  当其有效时，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。

- $VE\bar{C}T$

  当其有效时，原意为D来源于中断向量，现用于接收手拨微地址。

- $P\bar{L}$

  当其有效时，D来源于微指令的下地址字段，用于实现微程序转移。

引脚定义

- $C\bar{C}$

  条件测试，当其为低电平时，测试成功，转移（使用下址字段）；否则，顺序执行。

- $O\bar{E}$

  Y输出允许信号，低电平有效，已接地

- $CC\bar{E}N$

  $C\bar{C}$允许信号，已接地

16条命令

- 2号命令

  指令功能分支，无条件转MAP

- 3号命令

  条件转移，条件测试$C\bar{C}$为1时顺序执行，否则按下地址D转移。

- 14号命令

  顺序执行下一条微指令

微程序

微指令格式

每位的用处已在EXCEL表格中说明。

- $DC1$和$DC2$

  关于这两个控制位我还有两个问题

  - $DC2$为2时

    写AR时应把$DC2$设为2，即AR接收来自IB的数据；但$DC1$课本上都是设为0，即微型开关送IB。

    这样不就是把微型开关送到了AR嘛？这不太对啊！？以下是我的猜测：

    我猜在这里$DC1$应该是任意的。因为AR只能接收运算器的输出，所以当$DC2$设置为写AR时，就忽略$DC1$的设置，硬件应该可以实现。

  - $DC2$为0时

    此时代表NC，即无寄存器接收IB的数据，所以此时$DC1$应该也是任意，书上写的是0。

- SSH

常用微指令

- 19H

  取指令，PC增量。

  该条微指令公用于所有指令。

- 1AH

  按新取来的指令的操作码找到该条指令本身的微程序段的入口地址。

  19H之后一定是1AH。

- A4H

  根据有无中断请求，决定是进入中断处理过程，还是顺序执行。

  任何一条机器指令执行完都要去A4H检测中断

程序调试

以第一条指令为例

前期准备

1. $S_2S_1S_0$

   设为`100`

2. $FS_1FS_2FS_3FS_4$

   设为`1010`

3. $STEP/CONT$

设为`CONT`

输入并查看微码

1. `E900`

   将微码输入到`900H`开始的内存单元中

2. `D900`

   查看内存

将微码加载到微控存

1. `A800`

   输入加载微码的程序

```
0800: MOV R1,900     ; 900是微码在内存中地址
0802: MOV R2,7       ; 共7条微指令
0804: MOV R3,110     ; 微码在微控存中的首地址
0806: LDMC           ;加载微码
0807: RET
```

2. `G800`

   运行上边的代码，把微码装入微控存`110H`开始的单元中

测试指令

1. `A820`

   输入测试指令的程序到`820H`开始的内存单元中

```
0820: MOV R0,0045
0822: MOV [A00],R0  ; [A00]单元存储45H
0824: MOV R1,0023
0826: MOV [A01],R1  ; [A01]单元存储23H
0828: NOP           ; 占一个字
0829: NOP
082A: NOP
082B: RET
```

2. `E826`

   把新指令写到`826H`开始的内存单元

```
0828    0000:D800  0000:0A00  0000:0A01
```

3. `U820`

   反汇编，查看我们输入的测试指令的程序

```
0820: 2C00 0045  MOV    R0,     0045
0822: 3400 0A00  MOV    [0A00], R0
0824: 2C10 0023  MOV    R1,     0023
0826: 3401 0A01  MOV    [0A01], R1
0828: D800       DW     D800
0829: 0A00       ADC    R0,     R0   ; 高8位为000010XX，对应指令为ADC，实际上是我们自己设计的指令
082A: 0A01       ADC    R0,     R1
082B: AC00       RET
```

4. `G820`

   运行从`820H`开始的程序

观察运算结果

`DA00`