

2007

微指令系统设计



目录：

1. [概要说明](#)
2. [TEC2000 缺陷所在及修改方案](#)
3. [指令系统的设计](#)
4. [运算器设计](#)
5. [组合逻辑的设计](#)
6. [微程序的设计](#)

胡菊花、孟必平、黄宇飞

计算机科学与技术系

计算机组成原理微指令系统设计

一、概要说明

我们在计算机组成原理实验中所使用的 TEC2000 计算机组成原理实验系统机包括了硬件和软件两个部分。其中，硬件部分包括运算器、控制器（包括组合逻辑控制器和微程序控制器）、存储器三个功能部件以及接口和电源；软件部分包括微程序监控程序、监控程序、PC 机与 TEC2000 机通讯程序、交叉汇编程序以及 TEC2000 模拟器程序。从整个学期组成原理实验的过程看来，它还是能不错地满足实验所需，从而帮助组成原理实验的教学并使同学们对于计算机组成原理有了更进一步的了解。

但是，通过实验，我们不难发现它在设计方面还是存在一定的问题，尤其有些设计违反了大家公认的设计原则；这样不仅给教学带来一定的麻烦，同时容易使学习中的同学产生使用混乱的感觉，而在实验中引起一些没有必要的错误。

所以我们需要对于 TEC2000 指令系统、组合逻辑以及微程序进行一系列的改进设计。

二、TEC2000 的缺陷及修改方案

通过实验我们发现 TEC2000 机的缺陷主要有：

➤ 寄存器

- 1) 专用寄存器（程序计数器 PC 与堆栈寄存器 SP）的设置不合理，给编程带来了不少的麻烦（R4 和 R5 容易造成同学们无意间误用而出现错误）。
- 2) 不支持字节存取。

➤ 指令系统

- 3) 基本指令的功能与我们熟知的 8086/8088 指令系统相对薄弱，虽然扩展指令对此有所弥补，但是在组合逻辑中没有提供扩展指令的实现，致使在组合逻辑实验前需要首先进行指令扩展。
- 4) 有些指令名称与我们熟知的 8086/8088 的指令系统的名称差异较大，致使同学们不能方便地记忆和使用。
- 5) 操作码的排列混乱，给同学们自己添加指令带来了不少麻烦。
- 6) 寻址方式过于简单，指令使用不方便。
- 7) 不能进行多位的移位运算，在组成原理实验编成中需要进行循环，给同学们带来了一定的麻烦。

➤ 运算器

- 8) 运算功能太过薄弱，不支持乘、除、十进制加减等。
- 9) 如果要实现例如十进制加减的功能，TEC2000 机却少标志位。
- 10) 有些标志位的设计与我们的常识不符，容易造成判断错误，给编程中的转移等指令的使用造成混乱，在实验中给同学们带来了不少麻烦。
- 11) 在 I/O 输入输出时，是以字节为单位所表达的 ASCII 值进行的，但是 TEC2000 中没有提供字节运算，致使需要首先屏蔽高八位。
- 12) 对于左移、右移选择（逻辑、算术、循环、带 C、联合）的编码过于笼统。

➤ 组合逻辑控制

- 13) 控制信号编码不符合一般常识，给用微指令设计机器指令时带来了一定的不方便。

➤ 微程序

- 14) 对于转移功能的编码过于笼统，不具有唯一代表性。

根据以上缺陷，经过分析和研究，我们决定对其进行如下修改：

- 1) 修改专用寄存器的位置，为了支持字节运算，设计字节寄存器。
- 2) 更改指令名称为同学们熟知的 8086/8088 指令的名称，从而方便同学们的使用。
- 3) 不进行基本指令和扩展指令的划分，在将扩展指令融入基本指令的同时，添加 8086/8088 指令系统中一些必要的指令，从而增强指令的功能。
- 4) 在添加指令的同时，进行合理的操作码排序。
- 5) 改进运算器，使得运算器支持字节运算，并且增加了一些运算功能，通过提高运算器（硬件）的功能，提高计算机的整体性能。
- 6) 相应地，增加了字节运算指令，内存按字节编址。
- 7) 对于一些不合理的标志位、逻辑控制和微程序控制的编码进行重新设计。

三、指令系统的设计

（一）将指令分为 A 类（算术运算）、B 类（逻辑运算和移位）、C 类（数据传送）、D 类（程序控制）、E 类（处理器控制）五类，并按 8086/8088 的命名方式给它们命名。

（二）对 A—E 五类指令先按类别，再按操作数数目从双目—>单目—>无的顺序进行操作码的编码。

（三）为所有的算术与逻辑运算指令、移位指令和数据传送指令增加字节类指令。

（四）增加不带 C 的循环移位指令共 4 条。

（五）增加十进制加减指令共 4 条。

（六）增加了乘法和除法指令共 16 条，包括带符号和不带符号。

（七）为算术与逻辑运算指令增加立即数寻址方式。

(八) 增加字节寄存器，并把 SP 设置为 R14，PC 设置为 R15。

(九) 在乘除时，Q 寄存器发挥了相当大的作用，具体见下表

寄存器编码如下：

表格 1 寄存器编码表

REG(DR 或 SR)	W=1(字)	W=0(字节)
0000(0H)	R0	R0L
0001(1H)	R1	R1L
0010(2H)	R2	R2L
0011(3H)	R3	R3L
0100(4H)	R4	R4L
0101(5H)	R5	R5L
0110(6H)	R6	R6L
0111(7H)	R7	R7L
1000(8H)	R8	R0H
1001(9H)	R9	R1H
1010(AH)	R10	R2H
1011(BH)	R11	R3H
1100(CH)	R12	R4H
1101(DH)	R13	R5H
1110(EH)	SP	R6H
1111(FH)	PC	R7H

注：W 为字/字节控制信号，若为字操作，W=1；若为字节操作，W=0。字节寄存器与字寄存器利用 W 位区分。例如，编码为 0000 的寄存器，若 W=1，则是字寄存器 R0；若 W=0，则是字节寄存器 R0L。

在进行乘除时，除了字乘时，将结果的高 16 位存入 Q，低 16 位存入目的寄存起，其余的结果都是指定放入 Q；除法时都是把商放入目的寄存起，把余数放入 Q。

指令编码表如下：

表格 2 指令编码表

指令格式	汇编语句	操作数类型	功能说明
算术运算指令			
00000000(00H)	ADD DR,SR	字节	DR+SR->DR
00000001(01H)	ADD DR,SR	字	DR+SR->DR
00000010(02H)	ADD DR,DATA	字节	DR+DATA->DR
00000011(03H)	ADD DR,DATA	字	DR+DATA->DR
00000100(04H)	ADC DR,SR	字节	DR+SR+C->DR
00000101(05H)	ADC DR,SR	字	DR+SR+C->DR
00000110(06H)	ADC DR,DATA	字节	DR+DATA+C->DR
00000111(07H)	ADC DR,DATA	字	DR+DATA+C->DR
00001000(08H)	SUB DR,SR	字节	DR-SR->DR
00001001(09H)	SUB DR,SR	字	DR-SR->DR
00001010(0AH)	SUB DR,DATA	字节	DR-DATA->DR
00001011(0BH)	SUB DR,DATA	字	DR-DATA->DR
00001100(0CH)	SBB DR,SR	字节	DR-SR-C->DR
00001101(0DH)	SBB DR,SR	字	DR-SR-C->DR
00001110(0EH)	SBB DR,DATA	字节	DR-DATA-C->DR
00001111(0FH)	SBB DR,DATA	字	DR-DATA-C->DR
00010000(10H)	CMP DR,SR	字节	DR-SR
00010001(11H)	CMP DR,SR	字	DR-SR

00010010(12H)	CMP DR, DATA	字节	DR-DATA
00010011(13H)	CMP DR, DATA	字	DR-DATA
00010100(14H)	MUL DR, SR	字节	DRL*SRL->Q (无符号)
00010101(15H)	MUL DR, SR	字	DR*SR->(Q, DR) (无符号)
00010110(16H)	MUL DR, DATA	字节	DRL*DATA->Q (无符号)
00010111(17H)	MUL DR, DATA	字	DR*DATA->(Q, DR) (无符号)
00011000(18H)	IMUL DR, SR	字节	DRL*SRL->Q (带符号)
00011001(19H)	IMUL DR, SR	字	DR*SR->(Q, DR) (带符号)
00011010(1AH)	IMUL DR, DATA	字节	DRL*DATA->Q (带符号)
00011011(1BH)	IMUL DR, DATA	字	DR*DATA->(Q, DR) (带符号)
00011100(1CH)	DIV DR, SR	字节	DRL/SRL->DRL, DRL % SRL->Q
00011101(1DH)	DIV DR, SR	字	DR/SR->DR, DR % SR->Q
00011110(1EH)	DIV DR, DATA	字节	DRL/DATA->DRL, DRL%DATA->Q
00011111(1FH)	DIV DR, DATA	字	DR/DATA->DR, DR % DATA->Q
00100000(20H)	IDIV DR, SR	字节	DRL/SRL->DRL, DRL % SRL->Q
00100001(21H)	IDIV DR, SR	字	DR/SR->DR, DR % SR->Q
00100010(22H)	IDIV DR, DATA	字节	DRL/DATA->DRL, DRL % DATA->Q
00100011(23H)	IDIV DR, DATA	字	DR/DATA->DR, DR % DATA->Q
00100100(24H)	ADDD DR, SR	字节	压缩 B C D 加
00100101(25H)	ADDD DR, SR	字	压缩 B C D 加
00100110(26H)	SUBD DR, SR	字节	压缩 B C D 减
00100111(27H)	SUBD DR, SR	字	压缩 B C D 减

00101000(28H)	INC	DR	字节	DR+1->DR
---------------	-----	----	----	----------

00101001(29H)	INC	DR	字	DR+1->DR
---------------	-----	----	---	----------

00101010(2AH)	DEC	DR	字节	DR-1->DR
---------------	-----	----	----	----------

00101011(2BH)	DEC	DR	字	DR-1->DR
---------------	-----	----	---	----------

逻辑运算和移位指令

00101100(2CH)	AND	DR,SR	字节	DR AND SR->DR
---------------	-----	-------	----	---------------

00101101(2DH)	AND	DR,SR	字	DR AND SR->DR
---------------	-----	-------	---	---------------

00101110(2EH)	AND	DR,DATA	字节	DR AND DATA->DR
---------------	-----	---------	----	-----------------

00101111(2FH)	AND	DR,DATA	字	DR AND DATA->DR
---------------	-----	---------	---	-----------------

00110000(30H)	OR	DR,SR	字节	DR OR SR->DR
---------------	----	-------	----	--------------

00110001(31H)	OR	DR,SR	字	DR OR SR->DR
---------------	----	-------	---	--------------

00110010(32H)	OR	DR,DATA	字节	DR OR DATA->DR
---------------	----	---------	----	----------------

00100011(33H)	OR	DR,DATA	字	DR OR DATA->DR
---------------	----	---------	---	----------------

00110100(34H)	XOR	DR,SR	字节	DR XOR SR->DR
---------------	-----	-------	----	---------------

00110101(35H)	XOR	DR,SR	字	DR XOR SR->DR
---------------	-----	-------	---	---------------

00110110(36H)	XOR	DR,DATA	字节	DR XOR DATA->DR
---------------	-----	---------	----	-----------------

00110111(37H)	XOR	DR,DATA	字	DR XOR DATA->DR
---------------	-----	---------	---	-----------------

00111000(38H)	TEST	DR,SR	字节	DR AND SR
---------------	------	-------	----	-----------

00111001(39H)	TEST	DR,SR	字	DR AND SR
---------------	------	-------	---	-----------

00111010(3AH)	TEST	DR,DATA	字节	DR AND DATA
---------------	------	---------	----	-------------

00111011(3BH)	TEST	DR,DATA	字	DR AND DATA
---------------	------	---------	---	-------------

00111100(3CH)	SHL	DR	字节	DR 逻辑左移
---------------	-----	----	----	---------

00111101(3DH)	SHL DR	字	DR 逻辑左移
00111110(3EH)	SHR DR	字节	DR 逻辑右移
00111111(3FH)	SHR DR	字	DR 逻辑右移
01000000(40H)	SAL DR	字节	DR 算术左移
01000001(41H)	SAL DR	字	DR 算术左移
01000010(42H)	SAR DR	字节	DR 算术右移
01000011(43H)	SAR DR	字	DR 算术右移
01000100(44H)	ROL DR	字节	DR 不带 C 循环左移
01000101(45H)	ROL DR	字	DR 不带 C 循环左移
01000110(46H)	ROR DR	字节	DR 不带 C 循环右移
01000111(47H)	ROR DR	字	DR 不带 C 循环右移
01001000(48H)	RCL DR	字节	DR 带 C 循环左移
01001001(49H)	RCL DR	字	DR 带 C 循环左移
01001010(4AH)	RCR DR	字节	DR 带 C 循环右移
01001011(4BH)	RCR DR	字	DR 带 C 循环右移
01001100(4CH)	NOT DR	字节	DR 取反
01001101(4DH)	NOT DR	字	DR 取反

数据传送类指令

01001110(4EH)	MOV DR,SR	字节	DR->SR
01001111(4FH)	MOV DR,SR	字	DR->SR
01010000(50H)	MOV DR,DATA	字节	DATA->DR
01010001(51H)	MOV DR,DATA	字	DATA->DR

01010010(52H)	MOV DR,[SR]	字节	[SR]->DR
01010011(53H)	MOV DR,[SR]	字	[SR]->DR
01010100(54H)	MOV [DR],SR	字节	SR->[DR]
01010101(55H)	MOV [DR],SR	字	SR->[DR]
01010110(56H)	MOV DR,[ADR]	字节	[ADR]->DR
01010111(57H)	MOV DR,[ADR]	字	[ADR]->DR
01011000(58H)	MOV [ADR],SR	字节	SR->[ADR]
01011001(59H)	MOV [ADR],SR	字	SR->[ADR]
01011010(5AH)	MOV DR,OFFSET[SR]	字节	[OFFSET+SR]->DR
01011011(5BH)	MOV DR,OFFSET[SR]	字	[OFFSET+SR]->DR
01011100(5CH)	MOV OFFSET[DR],SR	字节	[OFFSET+DR]->SR
01011101(5DH)	MOV OFFSET[DR],SR	字	[OFFSET+DR]->SR
01011110(5EH)	PSHF		标志寄存器入栈
01011111(5FH)	PUSH SR	字	SR 入栈
01100000(60H)	POPF	字	标志寄存器出栈
01100001(61H)	POP DR	字	出栈到 DR
01100010(62H)	IN PORT	字节	(PORT)->R0L
01100011(63H)	OUT PORT	字节	R0L->(PORT)
控制转移指令			
01100100(64H)	CALL ADR	字	调用起始地址为 ADR 的子程序
01100101(65H)	CALL SR	字	调用起始地址为 SR 的子程序
01100110(66H)	JMP ADR	字	无条件长跳转到 ADR

01100111(67H)	JMP SR	字	无条件长跳转到 SR 所指示地址
01101000(68H)	JR ADR	字	无条件短跳转到 ADR
01101001(69H)	JC ADR	字	C=1 时跳转
01101010(6AH)	JNC ADR	字	C=0 时跳转
01101011(6BH)	JZ ADR	字	Z=1 时跳转
01101100(6CH)	JNZ ADR	字	Z=0 时跳转
01101101(6DH)	JS ADR	字	S=1 时跳转
01101110(6EH)	JNS ADR	字	S=0 时跳转
01101111(6FH)	JO ADR	字	O=1 时跳转
01110000(70H)	JNO ADR	字	O=0 时跳转
01110001(71H)	RET		子程序返回
01110010(72H)	IRET		中断返回

处理器控制指令

01110011(73H)	CLC		0->C
01110100(74H)	STC		1->C
01110101(75H)	SLI		关中断
01110110(76H)	STI		开中断
01110111(77H)	NOP		空操作

注：标志寄存器入栈指令 PSHF 没有改成 8086/8088 标志寄存器入栈指令 PUSHF，而依然使用原来 TEC2000 中原来的指令名称，是为了简化监控程序中汇编命令的编写而限制指令名长度，做出的妥协。

并没有将 8086/8088 指令系统中所有的指令都加上，是因为上述这些指令足以满足我们实验的需要，而且相对于原来 TEC2000 的设计改良了很多。

因为我们将对运算器的功能进行改进，在其中直接实现压缩 BCD 加减，并且直接实现有符号和无符号乘除，所以指令有所改变。

操作码并没有和 8086/8088 指令编码一致。

对于乘除设置了有符号和无符号运算，并且规定做乘时，只有在字乘的时候，高 16 位送入 Q 寄存器，低 16 位送入目的寄存器，其它情况都是将结果放入 Q 寄存器，主要是为降低设计微指令时控制信号的复杂度而作的妥协；在做除法时，不管是字节除还是字除都将商放入目的寄存器，把余数放入 Q 寄存器。

四、运算器的设计

由于增加了乘除运算，所以需要将控制码扩展到 10 位，控制信号的含义如下：

表格 3 运算器运算功能控制码

控制码编码				ALU
I6	I5	I4	I3	function
$\mu 35$	$\mu 34$	$\mu 33$	$\mu 32$	
0	0	0	0	$R + S$
0	0	0	1	$S - R$
0	0	1	0	$R - S$
0	0	1	1	$R * S$
0	1	0	0	$R (*) S$
0	1	0	1	R / S
0	1	1	0	$R (/) S$
0	1	1	1	INVE R
1	0	0	0	$R (+) S$
1	0	0	1	$R (-) S$

051221028 胡菊花 051221065 孟必平 051221033 黄宇飞

1	0	1	0	$R \vee S$
1	0	1	1	$R \wedge S$
1	1	0	0	$!R \wedge S$
1	1	0	1	$R \oplus S$
1	1	1	0	$!(R \oplus S)$
1	1	1	1	$[R]$

注：+加，-减，*无符号乘，(*)带符号乘，/无符号除，(/)带符号，INVE 反置，(+) 压缩

BCD 加，(-) 压缩 BCD 减， \vee 或， \wedge 与，!非， \oplus 异或，[]取补。

最后一个运算主要是为了凑足 16 个运算。

表格 4 运算器结果输出控制码

I10 $\mu 39$	I9 $\mu 38$	I8 $\mu 37$	I7 $\mu 36$	寄存器 结果选择	Q 结果 的选择	Y 输出选择
0	0	0	0	无	F1	F1
0	0	0	1	无	无	F1
0	0	1	0	$F1 \rightarrow B$	无	A
0	0	1	1	$F1 \rightarrow B$	无	F1
0	1	0	0	$F1/2 \rightarrow B$	$Q/2 \rightarrow Q$	F1
0	1	0	1	$F1/2 \rightarrow B$	无	F1
0	1	1	0	$2 * F1 \rightarrow B$	$2 * Q \rightarrow Q$	F1
0	1	1	1	$2 * F1 \rightarrow B$	无	F1
1	0	0	0	$F1 \rightarrow B$	F2	F1

注：因为加入了字乘除运算，所以设了 F1 和 F2，从而实现乘法结果高 16 位和低 16 位的输出；除法时，能实现商和余数的分别存放。为了微指令实现的方便，所以选择将字乘的结果放入 Q 寄存器。

表格 5 数据来源控制码

I2	I1	I0	数据来源	
$\mu 30$	$\mu 29$	$\mu 28$	R	S
0	0	0	A	Q
0	0	1	A	B
0	1	0	0	Q
0	1	1	0	B
1	0	0	0	A
1	0	1	D	A
1	1	0	D	Q
1	1	1	D	0

由于内存编址采用了字节方式，堆栈操作和程序运行需要 SP+2，SP-2 和 PC+2 的功能。为了实现加减 2 的操作，填加了 C2 信号。C2=1 时运算结果会加上 2；C2=0 时运算结果不变。C2 信号与最低位进位信号 CIN 一起由 SCI 提供，因此扩充 SCI 信号，如下表所示：

表格 6 SCI 控制信号

SCI2	SCI1	SCI0	C2	Cin
$\mu 8$	$\mu 7$	$\mu 6$		
0	0	0	0	0
0	0	1	0	1
0	1	0	0	C
0	1	1	0	/C
1	0	0	1	0

表格 7 SST 控制信号

SST2	SST1	SST0	W	C	Z	O	S	说明
$\mu 14$	$\mu 13$	$\mu 12$	$\mu 15$					
0	0	0	-	C	Z	O	S	标志位不变

0	0	1	0	Cy	Zr	Ov	F7	接收 ALU 的标志位输出 (字节)
0	0	1	1	Cy	Zr	Ov	F15	接收 ALU 的标志位输出 (字)
0	1	0	-	IBH7	IBH6	IBH5	IBH4	恢复标志位原现场的值
0	1	1	-	0	Z	O	S	C 置为 0
1	0	0	-	1	Z	O	S	C 置为 1
1	0	1	0	RAM8	Z	O	S	左移操作, 其他标志位不变(字节)
1	0	1	1	RAM15	Z	O	S	左移操作, 其他标志位不变(字)
1	1	0	-	RAM0	Z	O	S	右移操作, 其他标志位不变
1	1	1	-	Q0	Z	O	S	联合右移, 其他标志位不变

注：联合移位时，通用寄存器在左，Q 寄存器在右。

由于加了不带进位 C 的移位，SSH 控制信号需要重新设置，具体如下所示：

表格 8 SSH 控制信号（注：此处的 X 代表不必处理。左、右移是由指令功能确定的）

SSH2 μ11	SSH1 μ10	SSH0 μ9	W μ15	左移			右移			说明
				RAM0	RAM8	Q0	RAM15	RAM7	Q15	
0	0	0	0	0	X	X	X	0	X	通用寄存器逻辑移位(字节)
0	0	0	1	0	X	X	0	X	X	通用寄存器逻辑移位(字)
0	0	1	0	0	X	X	X	F7	X	通用寄存器算术移位(字节)
0	0	1	1	0	X	X	F15	X	X	通用寄存器算术移位(字)
0	1	0	0	RAM8	X	X	X	RAM0	X	通用寄存器循环移位(字节)
0	1	0	1	RAM15	X	X	RAM0	X	X	通用寄存器循环移位(字)
0	1	1	0	C	X	X	X	C	X	通用寄存器与 C 循环移位(字节)
0	1	1	1	C	X	X	C	X	X	通用寄存器与 C 循环移位(字)
1	0	0	1	Q15	X	RAM15	Q0	X	RAM0	通用寄存器与 Q 循环联合移位
1	0	1	1	Q15	X	F15	F15	X	RAM0	通用寄存器与 Q 算术联合移位
1	1	0	1	Q15	X	0	0	X	RAM0	通用寄存器与 Q 逻辑联合移位

五、组合逻辑设计

(一) 下表是内存及 IO 访问控制信号：

表格 9 内存及 IO 访问控制信号

REQ μ42	/MIO μ41	/WR μ40	译码信号	功能	控制对象
0	-	-	无	无读写操作	-
1	1	1	/WE、/MREQ	内存写信号，内存读写请求	存储器的写信号引脚
1	1	0	/RD、/MREQ	内存读信号，内存读写请求	存储器的读信号引脚
1	0	1	/IOWE、/IOREQ	I/O 写信号，I/O 读写请求	I/O 的写信号引脚
1	0	0	/IORD、/IOREQ	I/O 读信号，I/O 读写请求	I/O 的读信号引脚

注：内存读写请求还是 I/O，可以直接通过/MIO 信号判断；一旦 REG 置为 1，则说明有读写，这相对原来 TEC2000 的设计更为合理。

(二) 总线操作、中断有关的控制信号：DC12-DC10、DC22-DC20

TEC2000 既可以作为 16 位机，又可以作为 8 位机，因此它的控制信号比较复杂。考虑到 8 位机的实用意义不大，我们决定重新设计为单纯的 16 位机。控制信号也得到相应简化。

如下表所示：

表格 10 DC12-DC10 控制信号

DC12 μ2	DC11 μ1	DC10 μ0	译码信号	功能说明
0	0	0	/SWTOIB	开关到内部总线
0	0	1	/RTOIB	ALU 输出到内部总线
0	1	0	/STOIB	16 位机扩展符号到内部总线
0	1	1	/FTOIB	状态到内部总线
1	0	0	/INTVH	中断高位到内部总线
1	0	1	/INTVL	中断低位到内部总线
其它			(无)	无操作

表格 11 DC22-DC20 控制信号

DC22 μ5	DC21 μ4	DC20 μ3	译码信号	功能说明
0	0	0	(无)	无操作
0	0	1	/GIR	指令寄存器接收
0	1	0	/GAR	地址寄存器接收
0	1	1	/INTR	恢复中断优先级
1	0	0	/INTN	新中断优先级
1	0	1	/EI	开中断, 置中断允许位 INTE=1
1	1	0	/DI	关中断, 置中断允许位 INTE=0
1	1	1	/GAR /GI	地址寄存器接收, 下条指令取值

注：将原本 DC23 信号的作用，转化到了 DC22-DC20 (111)。

六、微程序设计

(一) 对微程序的改进主要在于原来的条件判断电路设计过于笼统，没有唯一确定性，

所以进行如下改进：

表格 12 条件判断控制信号

SCC3 μ47	SCC2 μ46	SCC1 μ45	SCC0 μ44	功能
0	0	0	0	必转
0	0	0	1	/INT=0 时, 转
0	0	1	0	C, 转 (JRC)
0	0	1	1	/C, 转 (JRNC)
0	1	0	0	Z, 转 (JRZ)
0	1	0	1	/Z, 转 (JRNZ)
0	1	1	0	S, 转 (JRS)
0	1	1	1	/S, 转 (JRNS)
1	0	0	0	O, 转 (JRO)
1	0	0	1	/O, 转 (JRNO)
1	0	1	1	IRH2=0, 转
1	1	0	0	IRH0=1, 转

SA、SB：用于指明运算器 A 口、B 口的控制信号是由微指令给出还是由当前指令操作码给出。为了提供立即数操作，对 SA、SB 的进行如下改进：

表格 13 SA、SB 控制信号

SB μ24	SA1 μ26	SA0 μ25	A 口	B 口
0	0	0	微指令中 μ19-μ16 所指定的 A 口	微指令中 μ23-μ20 所指定的 B 口
1	0	1	IRL3-IRL0 所指定的 A 口	IRL7-IRL4 所指定的 B 口
X	1	0	微指令中 μ23-μ20 所指定的 A 口	—
X	1	1	IRL7-IRL4 所指定的 A 口	—

注：上表中 SA 和 SB 是分开设定的。

SA 控制信号既可以选择微指令中 μ19-μ16 所指定的 A 口和机器码中 IRL3-IRL0 所指定的 A 口，也可以微指令中 μ23-μ20 所指定的 A 口和机器码中 IRL7-IRL4 所指定的 A 口，这样就可以实现 DR+D(立即数)->DR 这种操作。

(二) 微指令的各控制信号如下表所示：

μ59 μ52 μ51 μ48 μ47	μ44 μ43	μ40 μ39	μ36 μ35	μ32 μ31	μ28	μ27	μ24
下地址 7-0	μI3-μI0	SCC3-SCC0	0、REQ/MIO/WR	I10 - I7	I6 - I3	0、I2 - I0	0、SA1-SA0、SB
μ23 μ20 μ19	μ16 μ15	μ14	μ12 μ11	μ9	μ8	μ6	μ5
μ3	μ2	μ0					
A 口地址	B 口地址	W、SST2-SST0	SSH2-SSH0	SCI2-SCI0	DC22-DC20	DC12-DC10	

因为控制信号的增加，导致微指令长度增加，为了使之为整数个四位组，所以增加到了 60 位。

其中：

μ59—μ52 的 8 位微码是下地址字段；

μ51—μ48 是用于给出 Am2910 芯片的 16 种命令码的编号,如表格 13 Am2901 微指令所示；

μ43、μ31、μ27 为备用位；

其余见表格 3 至表格 13。

表格 1 3 Am 2 9 1 0 微指令

$\mu 47$ $\mu 44$	功能	寄存器计数器内容	失败 / CCEN = 低 和 / CC = 高		通过 / CCEN = 低 和 / CC = 低		寄存器计数器操作	使能
			Y	堆栈	Y	堆栈		
0	复位	-	0	清除	0	清除	-	PL
1	条件转 PL 子程序	-	μPC	-	D	压入	-	PL
2	跳转 MAP	-	D	-	D	-	-	MAP
3	条件跳转 PL	-	μPC	-	D	-	-	PL
4	压入 / 条件装入计数器	-	μPC	压入	μPC	压入	(注)	PL
5	条件转 R/PL	-	R	压入	D	压入	-	PL
6	条件跳转向量	-	μPC	-	D	-	-	VECT
7	条件跳转 R/PL	-	R	-	D	-	-	PL
8	R! = 0, 重复本条指令	! = 0	F	-	F	-	R 减 1	PL
		= 0	μPC	弹出	μPC	弹出	-	PL
9	R! = 0, 重复本条指令	! = 0	D	-	D	-	R 减 1	PL
		= 0	μPC	-	μPC	-	-	PL
1 0	条件返回	-	μPC	-	F	弹出	-	PL
1 1	条件跳转 PL 和弹出	-	μPC	-	D	弹出	-	PL
1 2	装入计数器和连续	-	μPC	-	μPC	-	装入 R	PL
1 3	测试循环结束	-	F	-	μPC	弹出	-	PL
1 4	连续	-	μPC	-	μPC	-	-	PL
1 5	R! = 0, 重复本条指令 三路分支	! = 0	F	-	μPC	弹出	R 减 1	PL
		= 0	D	弹出	μPC	弹出	-	PL

注：若 / CC = 高并 / CCEN = 低，保持不变；否则装入 R 计数器。

微程序指令流程表，参见：[微程序指令流程表.xls](#)