控制器部件的功能、组成概述

控制器部件是计算机的五大功能部件之一，其作用是向整机每个部件(包括控制器部件本身)提供协同运行所需要的控制信号。计算机的最本质的功能是连续执行指令，而每一条指令往往又要分成几个执行步骤才得以完成。由此又可以说，计算机控制器的基本功能，是依据当前正在执行的指令和它所处的执行步骤，形成(或称得到)并提供出在这一时刻整机各部件要用到的控制信号。

执行一条指令，要经过读取指令、分析指令、执行指令所规定的处理功能三个阶段完成，控制器还要保证能按程序中设定的指令运行次序，自动地连续执行指令序列。

为此，控制器组成中，必须有一个能提供指令在内存中的地址的部件，通称程序计数器(PC)，服务于读取指令，并接收下条要执行的指令的地址。

还要有一个能保存读来的指令内容的部件，通称指令寄存器(IR)，以提供本指令执行的整个过程中要用到的指令本身的主要信息。

控制器的第三个组成成分，是脉冲源、启停控制逻辑，指令执行的步骤标记线路，它标记出每条指令的各执行步骤的相对次序关系。

控制器的第四个、也是控制器设计中最费力的一个组成成分，是全部时序控制信号的产生部件，它依据指令内容、指令的执行步骤(时刻)，也许还有些别的什么条件信号，来形成并提供出当前各部件本时刻要用到的控制信号。计算机整机各硬件系统，正是在这些信号控制下协同运行，产生予期的执行结果，也就是执行一条又一条的指令。

依据前述控制器的最后两个组成成分的具体组成与运行原理的不同，通常把控制器区分为微程序的控制器和组合逻辑(硬布线)的控制器两大类。教学计算机系统中，分别设计并实现了这两种控制器，为教学提供了比较理想的实例，也为实验人员开辟了选择使用与进行实验的广阔天地。图1.1给出了控制器组成与其在整机中地位的示意表示。

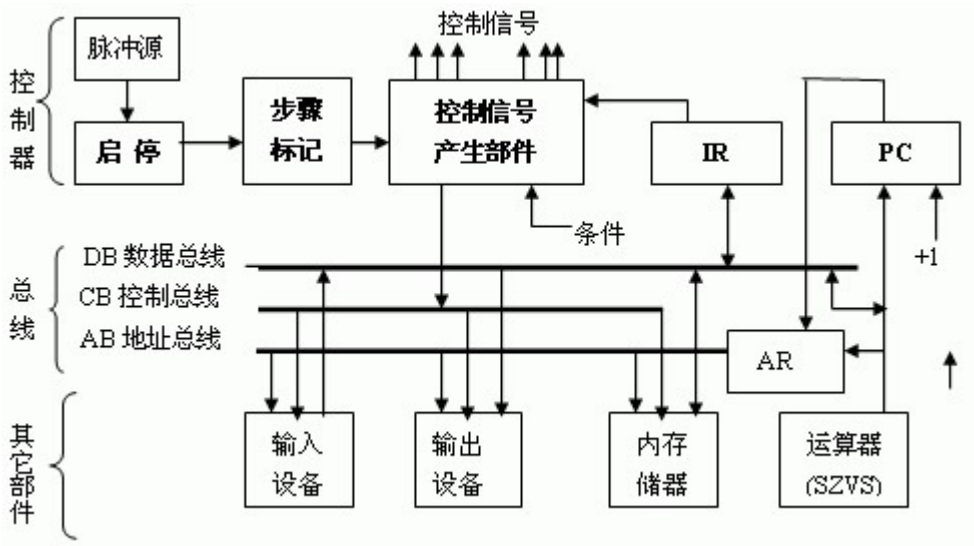

图1.1 控制器组成与其在整机中的地位

微程序控制器的基本知识

1. 微程序控制器的基本工作原理

微程序控制器组成中的核心成分，是控制存储器，通常用ROM器件实现，它用于存储按一定规则组织好的全部的控制信号。该存储器往往由若干个(几百到几千)存储单元(字)组成，每个字的内容，由执行一条指令的一个运行步骤用到的全部控制信号，以及主要用于指明下一个执行步骤的信息共同组成，通常称为一个微指令字，或一条微指令。若把一台计算机中全部指令的每一个执行步骤所用到的信息(控制信号与下一微指令的地址信息)，都有机地安排好次序关系并写进该存储器中，则它将有能力按指令执行情况，准确地提供每一步要用的控制信号，并按正确的执行步骤执行每一条指令。这个存储器大体对应图3.2中的控制信号产生部件。每一个存储字存放一条微指令，全部微指令组成一台计算机的微程序。

微程序控制器的工作原理，是依据读来的机器指令的操作码，找到与之对应的一段微程序的入口地址，并按由指令具体功能所确定的次序，逐条从控制存储器中读出微指令，以"驱动"计算机各功能部件正确运行。

就该控制器对自身的控制而言，其核心问题，是以多种合用的方式，为自己形成并提供出下一次要用到的微指令在控制存储器中的地址,这一功能大体对应图3.2中的步骤标记部件。这是微程序设计中要重点解决的问题之一，既涉及到一定的硬件技术，又与微程序设计技巧和运行性能密切相关，有必要作为单独的一小节进行讲解。

2. 得到下一条微指令地址的有关技术

要保证微指令逐条连续执行，就必须在本条微指令的执行过程中，能取来或临时形成(产生)下一条微指令的地址； 从运行效率考虑，稍后还应能用此地址把下一条微指令的内容从控存中读出来，以便在本条微指令执行完毕后，能尽早地进入下一条微指令的执行过程。

决定下条微指令地址(简称下地址)的因素较多，处理办法各不相同，可能包括:

① 微程序顺序执行时，下地址为本条微指令地址加1;

② 在微程序必定转向某一微地址时，可以在微指令字中的相关字段中给出该地址值;

③ 按微指令(上一条或本条)的某一执行结果的状态，选择顺序执行或转向某一地址，此时必须在微指令字中指明需判断的执行结果及转移地址。 要判断的执行结果，可以是运算器的标志位状态，控制器的执行状态，如多次的微指令循环是否结束，外设是否请求中断等等。

④ 微子程序的调用及返回控制，会用到微堆栈;

⑤ 依条件判断转向多条微指令地址中的某一地址的控制，它可以是前述第③条的更复杂一点的用法，也包括依据取来的机器指令的操作码，找到对应该条指令的执行过程的一段微程序的入口地址。这后一种情况，通常被称为微程序控制中的功能分支转移。此时在微指令字中直接给出多个下地址是不现实的或不合理的，应找出更合理的解决方案。

从上述讨论中可以看出，要得到下一条微指令的地址，至少得从两个方面入手:

. 一是要在微指令字中，分配相应的几个字段，用于给出微指令转移地址(完整的一个地址，或部分的多个地址)，以及是顺序执行，或无条件转移，或条件转移及其判断条件，是否是功能转移， 是否是微子程序调用或返回等等。

. 二是应有相应的专门硬件支持，用于实现微指令地址加1，按判断条件给出判定结果为真还是为假，给出微堆栈组织并实现入栈/出栈管理，解决指令操作码与各自的微程序段入口地址的对应关系以完成微程序中的功能分支转移等等。

由于不同的计算机所用的微程序设计技术不尽相同，故上述两个方面内容的处理与实现方法也会有明显差异，请有兴趣者参阅有关资料。我们只介绍在教学计算机系统中用到的有关硬件与具体微程序设计的知识。

3. 微程序定序器Am2910芯片的组成与功能

在教学计算机的微程序控制器中，用于形成下一条微指令地址的核心硬件，是一片Am2910器件，正确掌握该器件的内部组成与它所提供的功能，是学懂该教学机微程序设计技术，设计与实现自己的新指令的微程序段的重要环节。

(1). Am2910芯片的内部组成

Am2910芯片的内部组成如图1.2所示。

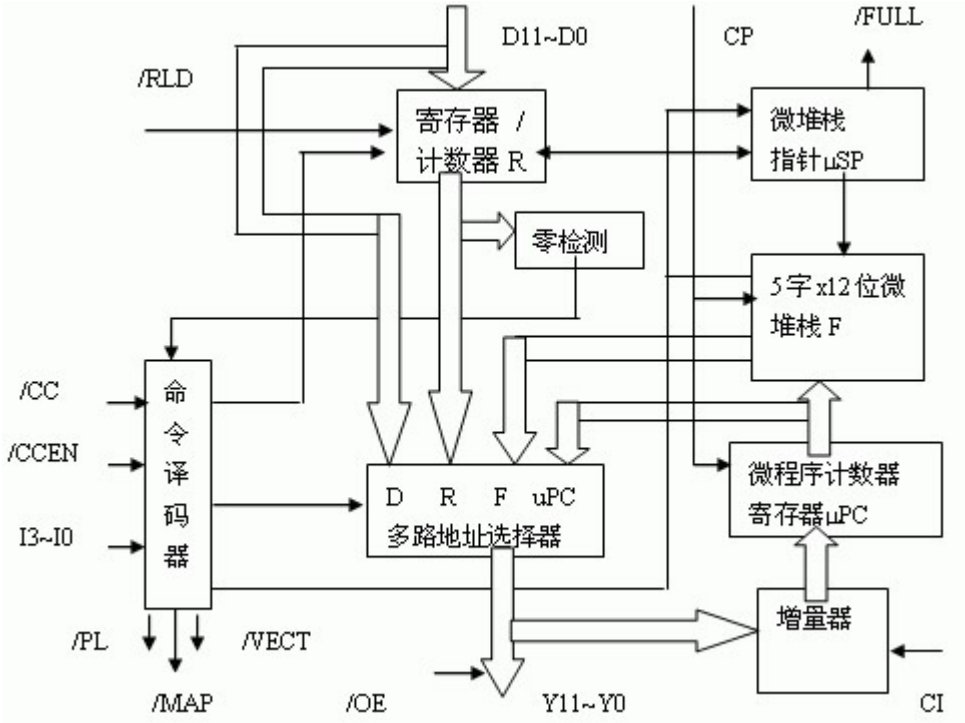

图 1.2 Am2910芯片的内部组成

Am2910是一片能提供12位微指令地址的器件，即它的输入输出的地址位数和器件内的部件位数均为12位，能直接寻址4096条微指令字的空间范围。

AM2910包括一个四输入的多路地址选择器，用来从寄存器/计数器(R)，直接输入(D)，微程序计数器(μPC)或微堆栈(F)四个输入中，选择其一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，用以实现微程序转移；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，则可能执行N+1次循环。

微程序计数器由12位的增量器和12位的寄存器μPC组成。当增量器的进位输入C1为高电平时，多路器的输出Y加1后装入μPC (即 μPC←Y+1)，用于实现微程序的顺序执行； 而当C1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位的寄存器堆栈和微堆栈指针?SP组成，主要用于保存微子程序调用时的返回地址和微程序循环的首地址。微堆栈指针?SP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。当微堆栈中的数据达到5个时，就发出微堆栈已满信号 (/FULL=0)，这时，任何压入操作都将覆盖掉栈顶的原有数据。

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的操作码找到相应的微程序段首地址的转移；
当 /VECT有效时，(即/VECT=0)，原意来源于中断向量，现用于接收手拨微地址。

命令译码器接收外部送来的命令码I3~I0，条件输入 /CC和条件允许 /CCEN信号，并对其译码，产生芯片内工作需要的控制信号，和外部要用的三个控制选择信号/PL、/MAP和 /VECT。

（2）AM2910引脚的定义

输入信号 :
D11-D0: 外部直接输入的数据，既可作为寄存器/计数器的初值，也可以经过地址多路选择器直接从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。
/CCEN和 /CC：共同确定测试条件是否通过，若/CCEN为低且 /CC为高，则指明测试失效；而 /CCEN为高或 /CC为低，均表明测试通过。若把 /CCEN 接地，即使其恒为低电平， 则可以只使用 /CC判断测试结果， /CC为低是测试通过，/CC为高则表明测试失效，我们在教学计算机中就是这样用的。
/PLD: 寄存器/计数器装入信号，当其为低电平时，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Am2910内部的寄存器/计数器。
CI: 增量器进位输入，当其为高电平时，控制微指令地址增量，即执行 μPC←Y+1，当为低电平时,执行 μPC←Y。
/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。

输出信号 :
Y11-Y0: 下一条微指令的地址，它直接被用作为读控制存储器的地址。
/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。

（3）Am2910的功能与具体用法

表1.1给出了Am2910所完成的功能，这些功能由命令码I3-I0，条件输入/CC，/CCEN 以及计数器当前值组合的结果来决定。

表1.1

	完成功能	R/C 内容	R/C 操作	使能信号	/CCEN			
					/CC 高		/CC 低	
					Y 输出	堆栈	Y 输出	堆栈
0	初始化			PL	0	清除	0	清除
1	条件转微子			PL	μPC		D	压入
2	指令功能分支			MAP	D		D	
3	条件转移			PL	μPC		D	
4	入栈 R/C 装数			PL	μPC	压入	μPC	压入
6	转中断向量			VECT	μPC		D	
8	R/C	非零	-1	PL	F		F	
	非零循环	零		PL	μPC	弹出	μPC	弹出
10	条件返回			PL	μPC		F	弹出
14	顺序执行			PL	μPC		μPC	
15	三路转移	非零	-1	PL	F		μPC	
		零		PL	D	弹出	μPC	弹出

注1：若测试失败则保持，否则就装数。 PC：微指令寄存器 D：直接输入
注2：图中符号 / 表示保持原内容不变。 R/C：寄存器/计数器 F：微堆栈栈顶

Am2910提供了16条命令，用来控制Am2910内部的操作和选择下一条将要执行的微指令的地址。其中只用少数命令（如0，2，14号命令）的执行结果仅由命令码本身决定，大部分命令还都要受到测试条件 (/CC和/CCEN) 为真还是为假的控制，有些命令（如 8，9和15号命令）的执行结果，则要受到内部计数器当前值是否为零的控制，其中15号命令同时受到内部计数器的值是否为零和测试条件是否通过的双重控制，以实现三路微程序转移的功能。

现将教学计算机设计中要用到的0，2，3，4，6，8，14，15这8条命令的功能和我们的具体用法说明如下:

◆ 0号命令: 用于初始化，即无条件清除内部微堆栈，并使Y的输出一定为零，用于系统加电时， 确保此时系统从0号微地址开始执行微程序。

◆ 2号命令: 用于指令功能分支，即输出信号/MAP为低,使D输入信号从MAP ROM（微地址映射部件）得到，并将其作为输出微地址Y的值，实现用指令操作码找到对应该指令的微程序段的入口地址，从而开始该条指令的特定的执行过程。

◆ 3号指令: 用于条件微转移控制，当条件成立，即/CC为低时，用/PL把微指令字中的下地址字段的内容(转移地址)经过D输入并送到Y，实现微程序转移。当/CC为高时，微程序顺序执行，即把已增1后的微指令地址作为下地址。若外部电路确保送入的/CC的状态为低，3号条件微转移命令也可以用于实现无条件的必定微程序转移来使用。

◆ 4号命令: 进栈并条件装入计数器，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。

◆ 6号命令：条件转至中断向量，用于为外部输入D选择一个新的微地址来源，即输出信号/VECT 为低，原意取某一向量地址，我们用其从教学机主板上的开关取一个手拨的微程序地址。

◆ 8号命令: 用于微程序循环控制。为了使用这条命令，前面必须用一条命令(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)，顺序执行下一条微指令。

◆ 14号命令：顺序执行，即执行紧跟在本条微指令后面的那条微指令。

◆ 15号命令: 三路转移，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。该命令要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)，并选择增1后的微指令地址作为下条微指令的地址，否则，选择微栈顶的内容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，则用增1后的微指令地址为下条微指令的地址，否则，下地址来自当前微指令的下地址字段。

微程序控制器的设计与实现

1. 基本组成部分

基本组成指的是该控制器的必不可少的组成部分，如图3.4所示。它与微程序设计的最基本的原理直接相关，是课程的重点内容。
从图3.1可以看出该控制器的基本组成部件及它们相互间的逻辑关系。

最核心的部分是控制存储器，用于存放教学机的微程序，由56位组成，用7片74LS6116随机读写的8位×2048字的内存芯片实现。通常控存都是用ROM芯片实现，把厂家设计好的微程序固化在里边，仅提供读操作功能，可靠性更高些。教学机要支持动态微程序设计，即允许实验人员把自己设计的微程序写进控存，我们只能用可读写存储器支持这一要求。这还带来一个新的问题，即在实验机加电启动时，首先必须把已设计好的53条机器指令用到的微程序调入控存，这个问题将到该控制器的辅助组成部分去进一步讲解。正常执行微程序时，该控存将依据Am2910提供给它的10位地址，在读写命令W/R(为高是读操作)控制下读出相应单元的一条微指令。

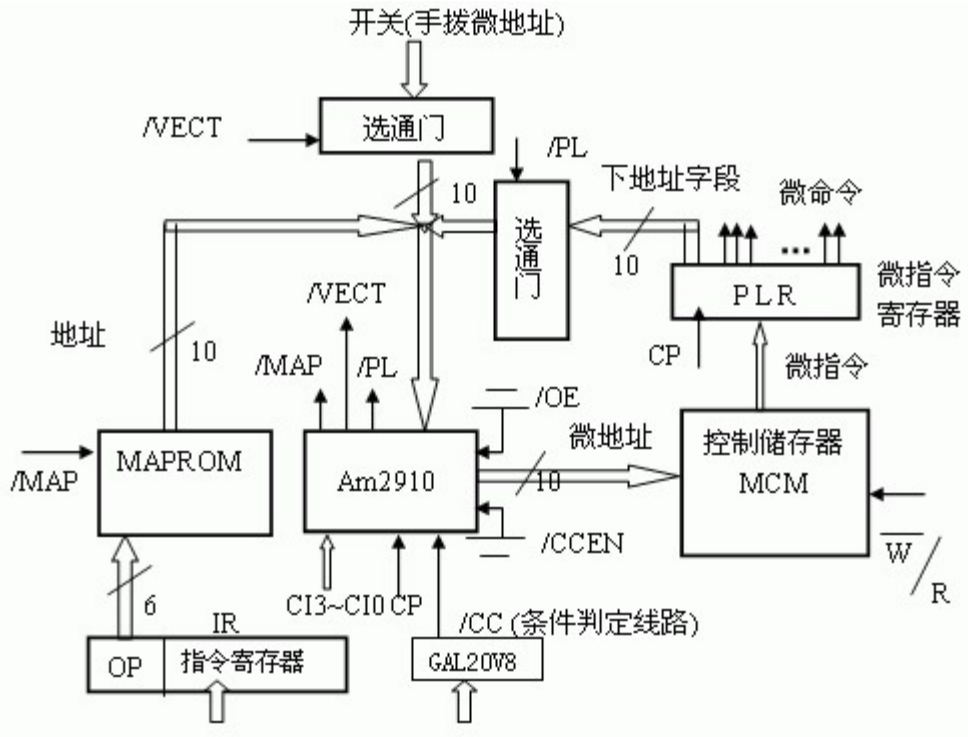

图 3.1 微程序控制器的基本组成

读出的微指令将通过CP脉冲，打入该控制器的第二个组成部件——微指令寄存器PLR中。前边已提到过，一条微指令的内容通常由下地址和控制信号(微命令)两部分组成。下地址可能又被分成几个小字段，它们被用于控制、形成、或直接提供下一条微指令的地址，是供给控制器部件本身使用的。微命令也由若干个小字段组成，提供对计算机其它功能部件，如运算器、主存、输入/输出设备的控制信号。在第二章，我们已看到用于控制运算器的许多控制信号。在本小节稍后部分，将详细介绍教学计算机的微指令的格式和每个字段的内容及其控制功能。

该控制器的第三个组成成分，是微程序定序器Am2910器件及其配套电路。

在本教学计算机的实现中，已把Am2910的/OE端接地，使其输出Y11-Y0总保持有效(实用Y9-Y0共10位)。

把/CCEN接地，使Am2910的条件判断结果只取决于/CC。

把CI接电源，使微指令地址 +1总是执行。

用/VECT信号把通过水平板上的开关给出的10位微指令的手拨地址接通到Am2910的D输入端。

分别用/MAP和/PL两个信号选通MAPROM和微指令下地址字段的输出送到Am2910的D输入端，从而形成D输入端的3选1逻辑功能。

与Am2910配套的电路，主要包括MAPROM和用于形成/CC信号逻辑值的条件判定线路。MAPROM被用作为指令微地址映射部件，它变换指令的操作码为该指令对应的微程序段入口地址，由两片74LS2716 ROM芯片组成，其地址为指令的操作码，对应单元中存放相应微程序段的入口地址，执行读操作，并用/MAP选通读出的信息，解决的是指令功能分支问题。

关于/CC条件码的形成问题，需解决指明判定条件和完成条件判定两个方面的需求。要判断的条件相当多，是通过微指令中下地址字段中的SCC(3位)和SC(1位)两个子字段指明的，其具体规定如表3.1所示,采用专门的硬件电路，即一片Gal20v8器件实现表中所规定的功能

表3.1

SCC	SC	/CC	SCC	SC	/CC
0		0	7	IR10-8	
1		1		0	/C
2	SC=0	/FS1		1	/Z
3	SC=0	/FS2		2	/V
4	SC=0	/FS3		3	/S
5	SC=0	/WAIT		4	C
2	SC=1	/C		5	Z
3	SC=1	/Z		6	V
4	SC=1	/V		7	S
5	SC=1	/S			
6		/INT			

表中的FS1、FS2和FS3是水平板上的3个功能开关，用于选择教学机执行不同的操作功能，其具体规定如表3.2所示

表3.2

FS1	FS2	FS3	FS4	功　能
X	X	X	1	脱机运算器部件实验(X 代表该位可取任意值)
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 地址连续执行程序
1	1	0	0	从指定地址单指令执行程序
1	1	1	0	从指定地址连续执行程序

/WAIT是教学处于单步执行时，用于单步控制线路的等待状态(等待按下STEP微型按键)。

C、Z、V、S或它们的取反值/C、/Z、/V、/S是运算器中的四个状态标志位。当SCC的3位微码为111,即十进制编码值为7时，通过条件转移指令的指令操作码 IR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所用的微指令中。

/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。

采用专门的硬件电路，即一片Gal20v8器件实现表3.2所规定的功能。

微指令字下地址字段中还有一个子字段CI3-CI0，用于给出Am2910的命令码，它与/CC的取值、Am2910内部的R/C的内容是否为零等一起，共同决定Am2910芯片内部的操作和形成下

一条微指令地址的具体办法。

2. 辅助组成部分

辅助部分指的是该控制器中的、用于向控制存储器写入微指令的逻辑电路，这是向学生提供必要的实验手段，即允许学生设计自己的微程序段和将其写入控存，并使其正确运行所应该具备的部分。包括支持通过开关和按键用手工方式向控存写入微指令，支持用LDMC(装入微码)指令向控存写入微指令，支持教学机加电启动时向控存写入已设计好的53条指令用到的全部微程序的有关电路。

实现中要解决的三个问题是:

◆ 控存读写在时间上的矛盾，即执行每一条微指令的过程，也正是读取下一条微指令的过程，把二者的执行过程从时间上重迭起来，以提高执行微指令的速度，增加计算机处理性能，这就是流水线技术的概念。为此，必须设置微指令寄存器，使读出的下一条微指令的内容送到该寄存器的输入端，等待CP脉冲的到来，以完成打入操作； 而该寄存器的输出正是本条微指令的内容，正在控制计算机各部件完成本步骤的操作功能。如果此时还要向控存写入微指令，与上述读操作是矛盾的。解决办法不外如下两种:

在执行一条微指令的期间(微周期)，分前后两段时间，分别完成控存的写、读操作，这使每条微指令的最短执行时间，至少要等于两个控存访问周期。

或设置一个不读控存的微指令周期，专门用于控存写操作。在此周期内，封锁有关的几处的CP脉冲，并由少量专门的电路提供控存写入操作要用到的控制信号。看来，在教学计算机中选用这后一种方案更合理些，但我们还是用了前一种方案，详述从略。

◆ 获得控存写入操作的地址
在手工拨入微指令前，要用水平板上的16位开关的低10位拨入控存地址。此时，是把开关拨好的地址值经选通门送到教学机的内部总线，再送入程序计数器PC中和地址寄存器AR中，至此就把控存地址送到了地址总线，并作为控存的地址来源之一。以后每写入一条微指令，使PC加1送PC和AR，作为写入下一条微指令的控存地址，支持连续写入操作。

◆ 提供控存写入操作的数据(微指令内容)
在手工拨入微指令的过程中，微指令是56位字长，而拨数开关只有16位。为此，需分4次拨入。为了尽量少影响控存的读操作，在教学机中设置了一个56位的控存写入寄存器LDR，分成4个16位的段(最高位的段只有8位)，各段依次接收16位数据开关拨入的16位微码，待4次拨入完成后，集中一次56位同时写入控存。具体实现方案是:

16位开关所拨内容经选通门送到内部总线IB，16位的IB直接接到了四段LDR的输入端。

控存写入寄存器LDR由7片8D寄存器器件实现，每两片组成一段(最高段仅用1片)，分别用不同的打入时钟信号控制打入操作。LDR的56位输出直接送到控存的数据入出端，用W/R读写信号控制写入(W/R为低是写)操作。

图3.2 是实现控存写入的有关逻辑电路示意图。

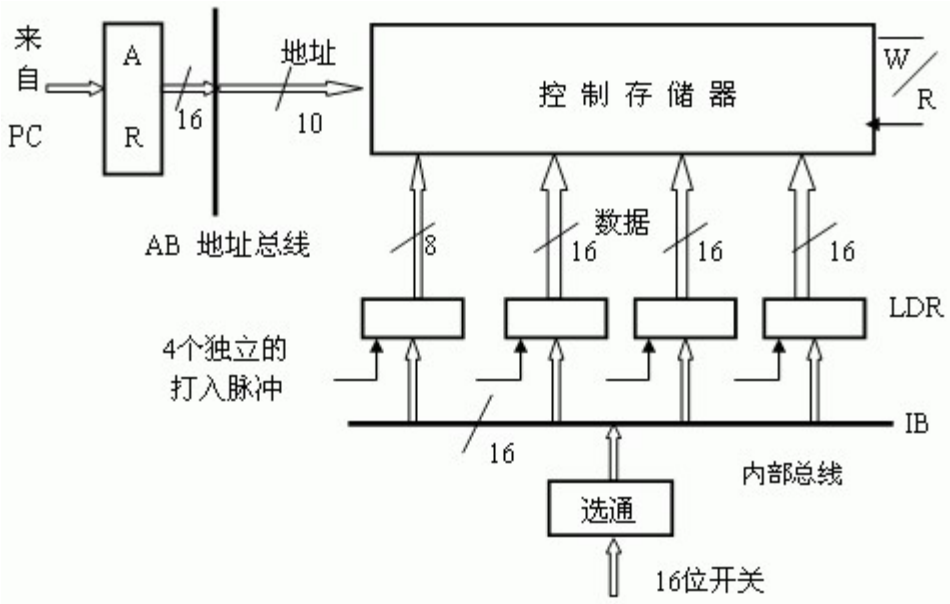

图 3.2 实现控存写入的逻辑线路

在通过LDMC指令向控存写入微指令时，用到了LDMC指令的三个默认参数，即微码在内存区的首地址，要写入的微指令的条数，要写的微指令在控存中的首地址，它们分别放在通用寄存器R1,R2和R3中。这条指令执行时，存在与手工拨入微指令同样的三个问题，即用R1内容作地址，依次4次读内存内容到内部总线，并依次写进LDR的四段，拼出一条待写入的微指令字； 然后用R3的内容作控存地址(用R3内容送地址寄存器实现)，并且一次完成控存写入操作。用到的电路与图3.5类似，只是变开关内容送IB为内存读出内容送IB，变PC内容送AR为R3内容送AR，变手工按单步按键操作为计算机执行指令，请参见逻辑图有关部分与LDMC指令相应的微程序段。

至于教学计算机加电启动过程中的微码初始化装入过程，涉及到几个专用的逻辑部件。一个是微码固化器件μCR，用两片74LS2716 ROM芯片组成16位字长的，2K字容量的专用电路，每4 个字存放一条微指令。

另一个是地址计数器部件，用完成4位计数功能的三片74LS6116器件串行计数实现，它被同时用作为读μCR和写控制存储器的两个地址。这两个地址有如下关系:

. μCR的内容从0地址顺序读出，控存的内容也从0地址顺序地写入，二者初值均为0。

. 读4次μCR写一次控存，二者操作频率为4:1，满足用10位地址读μCR，并用高8位地址写控存的条件。
第三个组成部件，是提供μCR的读命令信号，LDR的4个打入脉冲和控存的写入命令的信号，这些信号只能在这里提供，因为此时微码尚未装入控存，教学机的指令还不能执行。要保证此时提供的控制信号与将来微指令所提供的信号的互斥性。

图3.3是微码初始化装入的逻辑电路。10位地址用于读μCR，8位地址用于写控存，初值均为0。10位地址的低2位产生的4分频信号作为高8位地址的计数脉冲，其译码值的4个状态用于产生LDR的4个打入脉冲。 最后一次计数期间还用于写控存。这就是图上的读命令和写命令的形成逻辑。

图3.3的计数器输出用于指明微码初始化装入完成，并用以解决这里给出的控制信号与微指令

给出的信号的互斥问题。

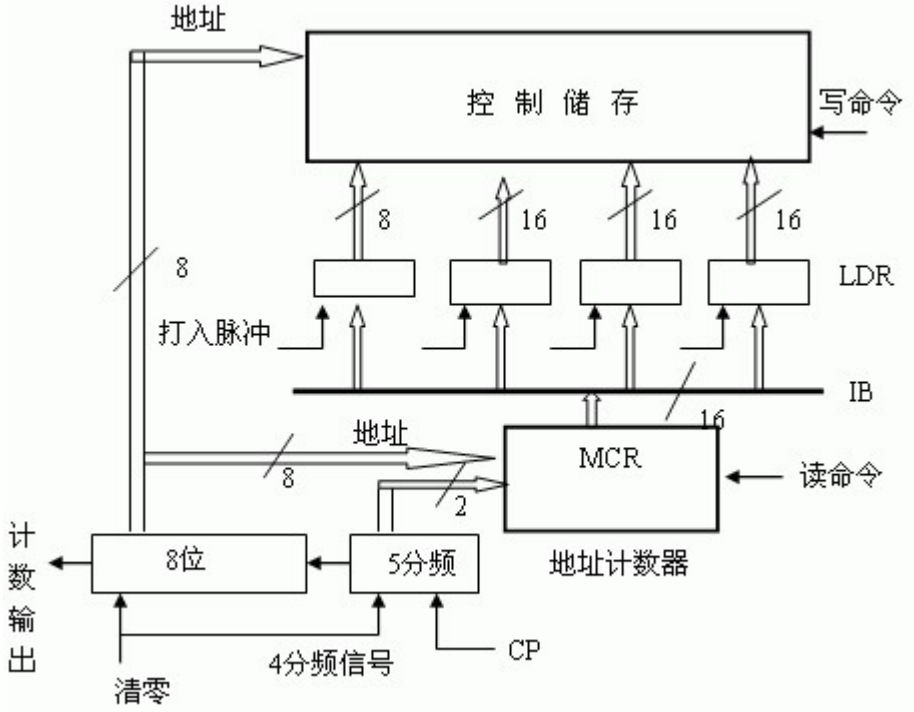

图 3.3

若把教学机微程序控制器的基本组成与辅助部分两项合起来，则可得到如图3.4所示的该控制器组成的逻辑框图。

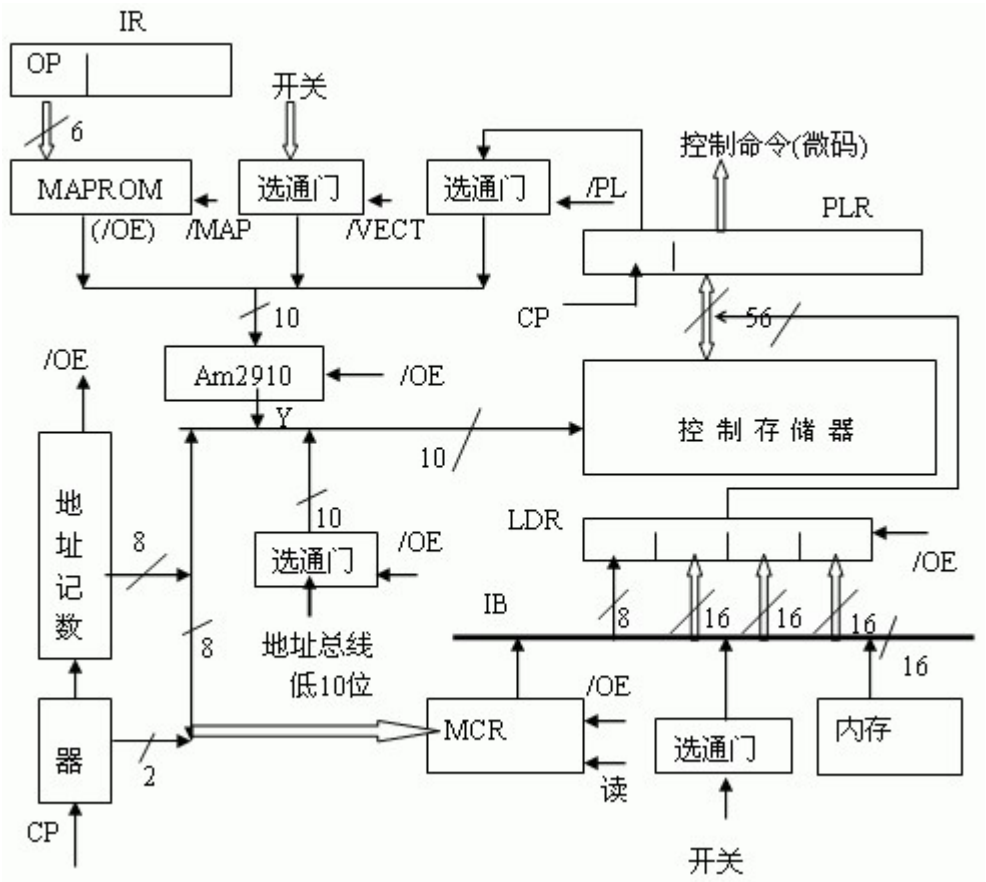

图3.4 教学机的微程序控制器的完整组成

图上半部分为其基本组成电路，用于读控存，执行微指令以解释指令的执行过程，是教学机正常运行所需要的。此时，由Am2910提供控存地址，从控存读出的微指令送微指令寄存器PLR，保证下边部分的电路的输出均被禁止，包括它送给控存的地址和56位的数据，通过给出正确的/OE信号实现。

下边部分为该控制器的辅助组成电路，支持向控存写入微指令。写控存期间，要禁止上半部分电路与控存发生作用，包括禁止Am2910的10位地址输出，不向PLR提供打入脉冲CP，以避免把写向控存的内容接收到PLR中，还包括对Am2910器件的一点特殊控制。

3. 教学计算机的微程序设计

(1) 微指令格式

一条微指令，通常由下地址字段与控制码字段两大部分内容组成，这每一部分又会分成若干个更小的子字段。

教学机的微指令由56位组成，其中最高的20位(含两位备用位)用于控制微程序下地址的形成。有10位用于给出微指令转移用的转移地址或微程序执行用到的数据(例如微程序循环次数)。4位用作Am2910的命令码CI3-CI0，另有4位(3位SCC和1位SC)用于给出微指令转移的判别条件。

余下的36位，1位备用，其它35位用于给出对控制器之外的其它几个功能部件的控制命令，包括对运算器、主存、入/出接口、总线等的控制。

其中有26位用于运算器，在第四章讲运行器部件时已看到过24位，这里还有两位SA和SB，用于指明送给Am2101的A口地址、B口地址的信息是来自微指令的A口字段、B口字段，还是来自指令寄存器的SR、DR字段。通常，单或双操作数指令中用到SR、DR内容时，Am2901的A口、B口地址需要由指令寄存器提供。当另一些指令用到默认的寄存器内容时(如IN和OUT指令默认使用通用寄存器RO，LDMC指令要用到R1,R2和R3等)，这些寄存器的编号，就必须在相关的微指令字的A口地址或者 B口地址字段给出。此外，由于PC、IP和SP等是用R5、R6和R4实现的，在涉及到PC和SP的计算、读写等操作时，这些寄存器的编号也必须通过相关微指令字的A口地址、B口地址字段来提供。为此，特规定分别用SA和SB完成这一选择，如表3.4所示。

为了对内存和输入/输出接口的读写进行控制，用了/MIO、REQ和/WE三位微码，具体规定如表3.3所示，即这三位用于指明正常总线周期的类型和支持动态微程序设计用到的一个写控制存储器的特殊操作周期。

表3.3

/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	X	不操作
0	1	0	I／O写	1	1	X	装入微码

表3.4

当 SA=0 时，Am2901 的 A 口地址来自微指令的 A 口字段 SA=1 时，则来自指令寄存器的 SR 字段
当 SB=0 时，Am2901 的 B 口地址来自微指令的 B 口字段 SB=1 时，则来自指令寄存器的 DR 字段

与内部总线和寄存器接收操作有关的微码共6 位。DC1用3位，选择把哪类信息送往内部总线IB，DC2用3位，选择哪一个寄存器接收出现在内部总线上的内容，其具体规定如表3.5和表3.6所示。其中DC2的011编码表明INT(中断优先级)寄存器接收外部送来的信息。

表 3.5

DC1 编码			控制端	送往内部总线 IB 的数据
0	0	0	/SWTOIB	开关手拨数据
0	0	1	/RTOIB	运算器的输出
0	1	0	/ITOIB	指令的低 8 位
0	1	1	/FTOIB	状态寄存器
1	0	0	/INTA	中断向量
1	0	1	NC	未使用
1	1	0	/EI	转用于开中断
1	1	1	/DI	转用于开中断

表3.6

DC2 编码			控制端	寄存器接收来自 IB 的数据
0	0	0	NC	未使用（NC）
0	0	1	GIR	指令寄存器 IR
0	1	0	GAR	地址寄存器 AR
0	1	1	GINTP	中断优先权
1	0	0	CPLDR0	LDR6
1	0	1	CPLDR1	LDR5，LDR4
1	1	0	CPLDR2	LDR3，LDR2
1	1	1	CPLDR3	LDR1，LDR0

这是用3-8译码器实现选择控制的。要指出的是，当DC1为000编码时，仅当不是内存或I/O读操作，即/MIO REQ /WE不为0X1时，才是开关内容送内部总线IB，在读内存或读外设时，要禁止DC1译码，使原DC1所确定的选择均不起作用，以便把外部总线DB上的内容，经过一个双向三态门送往内部总线IB。

这56位微码的具体安排如图3.5所示。

书写这56位微码时，经常采用16进制的表示形式。各字段的划分要有利于按16进制编码识别各段微码的控制功能。

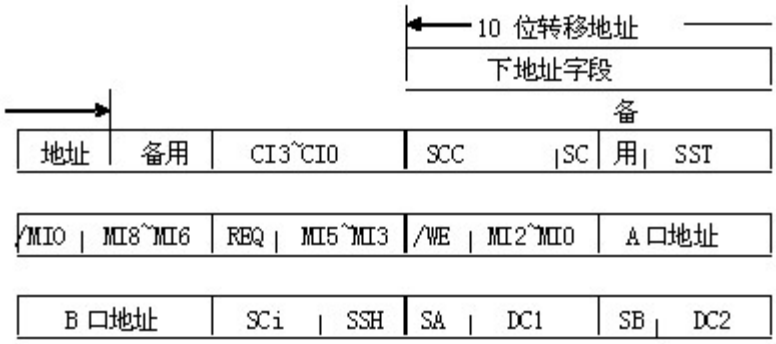

图3.5 微指令格式

(2) 教学机的微程序分析

教学机的微程序清单前面已给出

每条微指令在清单中的内容是:

从左数第一列为每条微指令在控存中的地址，用2位16进制数给出。

第二列为本条微指令完成功能的简单说明，包括选用什么条件形成/CC的值。

第三列是本条微指令中AM2910选用的命令码。

第四列是在本条微指令字的B55-B46给出的下地址字段的内容，经常是一个微转移地址。

再接下来给出用16进制表示的56位微码的具体内容。每两位数字表示一片8位的控存芯片中的信息。其排列关系已在微指令格式一小节给出。显而易见，前面刚说过的第三列和第四列的内容已包括在这56位微码的高14位部分。仅是由于这两个字段的内容对阅读微程序更加重要，才将其再单独表示在第三、第四两列。
在最右部分，有些微指令中还有简单注释，表明操作功能选择或该微程序段入口地址所对应的汇编语句。

微程序执行过程

现在我们先具体看一看已实现的微程序的执行过程。

假定教学机的功能开关拨在1010的位置，表明在从主存的0地址连续执行程序。

加电后按LDMC/RESET键的信号将首先启动装入微码，装入完成的信号，或RESET按键给出的信号/RESET将清零微指令寄存器PLR的B47-B40，使CI的四位为0000，得到AM2910的0#命令码，它使AM2910芯片内的微堆栈清空，并使微地址输出Y11-Y0 (教学机仅用Y9-Y0共10位)为0，故首先要读出并执行保存在控存0号单元中的微指令，从而启动了微程序的执行过程。

头一条微指令是判FS1所处的状态，它为1(开始已假定)，按表3.2规定，是CC为1，故/CC为低，按AM2910的3#命令规定，应转移，转移地址为10h。接下来应执行10h地址中的微指令。(0地址中的微指令要实现的IB→PC，→AR功能暂不讨论)。

10h中的微指令判FS2为0，使/CC为高，不满足转移条件，应顺序执行下一条微指令11h。

11h中的微指令判FS3为1，则转移到17h。

以上3条微指令完成了判定FS1、FS2、FS3开关为101位置，应进入从主存0地址连续执行程序的操作过程。
17h中的微指令完成把0值送入PC的操作。实现的方案是在运算器中执行R5-R5→R5的操作。R5即是PC。该微指令选用14#命令，表明应顺序执行下一条微指令。

18h中的微指令完成PC送地址寄存器，并将PC的内容保存到IP(即R6)中，以备后面可能有相对转移的地址计算之用。还用14#命令。

19h完成取指 (即读主存储器并将读出来的指令内容打入指令寄存器IR中) 和PC增量的操作。至此，当前指令已取到指令寄存器IR中，并准备好顺序执行指令的指令地址。还用14#命令。

20h选用2#命令，它使输出控制信号/MAP为低电平，则完成从MAPROM取得当前指令对应的微程序段的入口地址。

MAPROM是由2片8位的ROM芯片组成的微指令地址映射部件，其地址输入为IR给出的指令操作码，其输出为该条机器指令对应的微程序段的入口地址，指令操作码与相应微程序段的入口地址的对应关系给出在指令汇总表中(表2.2)。

接下来的微指令开始执行指令的具体功能。

例如，ADD指令将导致进入1Ch微指令，完成 DR+SR→DR的功能；

而ADC指令将导致进入1Eh微指令，完成 DR+SR+C→DR 的功能；

SUB和SBB指令，将通过20h、22h微指令分别完成DR-SR→DR、DR-SR-C→DR的功能，如此等等。

上述4条指令,完成具体的相应运算功能的控制方法，是正确地选择AM2901芯片的3组3位的控制码，和正确选择最低位的进位信号，并使/MIO、REQ、/WE为101，即不执行内存或外设的读写操作，这在运算器部件的教学实验过程中多次处理过。

上述4条指令的相应4条微指令的下地址字段均给出A4，微指令下地址的控制码选为3#，即按测试条件决定是顺序执行还是转移，判断条件是/CC=0，即测试条件通过，则一定转移，故下一条微指令的地址为A4h。
A4h微指令的功能是依据有无中断请求，决定进入中断处理过程，还是顺序执行下 一条指令，这是每条机器指令完成后都应该执行的一项操作。

没有中断请求，则执行A5h微指令。它要依据功能开关FS3的状态，检查是连续方式执行程序(1010或1110)还是单条指令方式执行程序(1100)，并完成PC内容送地址寄存器,保留PC值到IP中。为连续执行时，则转回19h进入下一条指令的取指过程。

教学机53条基本指令的微程序总体结构

下面看一看教学机53条基本指令的微程序的总体结构：

① 从00h-15h这16条微指令,用于判别教学机上3个功能开关FS1、FS2、FS3的位置状态,以决定教学机的运行方式，具体规定在前面已给出,并表示在教学机的简明操作卡上。这包括运行微程序、装入微码、存储器读写(包括在监控命令支持下读写或用开关、按钮方式读写),控制程序运行的方式(单步还是连续)，运行程序的首地址等。前面还以FS1 FS2 FS3处于101,且FS4处于0状态为例，讲解了这里有关的几条微指令的功能与运行次序。

② 17h和18h两条微指令用于加电后启动监控程序时，把0地址值送入PC，PC值送地址寄存器AR，做好读取头一条指令的准备工作。

③ 19h这条微指令完成取指,即读内存，把读来的指令字送入指令内存器IR中并实现PC值增量。这公用于所有指令,是所有指令必须首先完成的共同操作,它与指令的操作码无关。

④1Ah微指令,实现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，是接在取指后每一条指令都必须完成的一项操作。

⑤ 1Ch-A3h这些微指令用于实现教学机53条基本指令各自的执行功能。每一条机器指令用1到多条微指令实现，各自有自己的一个入口地址。在前边我们已具体看了几条指令，诸如ADD、ADC、SUB、SBB的执行过程。

⑥ A4h-ABh这几条微指令用于每条指令结束后,判有无中断请求,和是连续还是单指令执行程序，公用于所有指令。A4h还用于保存状态寄存器到Q。有中断请求时，转向由ADh-B6h去实现中断响应功能。无中断请求，则完成PC→AR，为顺序执行或转移做好取下一条指令的准备。A5h完成PC→AR和PC→IP的功能，与18h这条微指令的功能有些类似,差别在于，18h在初始加电后准备监控程序的首地址,A5h是执行完一条指令后,准备下一条指令的地址。A5h还要判连续还是单指令执行程序，即判FS3功能开关处于1还是0状态。连续时直接转19h完成取指,单指令执行时，用下边的几条的微指令处理等待，即等用户按下STEP 按键。

⑦ADh-B6h完成中断响应，包括处理机状态入栈，中断入口地址(由中断优先级拼接固定的高位地址形成，参见逻辑图纸)送进PC等。中断实验中有更详细的说明。

?B8h-BAh 3条微指令是手工装入微码和写存储器用到的等待微子程序； BCh-BEh是手动读存储器用到的等待微子程序。

有了上述说明，我们在下面给出教学机七条典型指令的微程序的流程图，如图3.6 所示。这里用到的有关符号是:

一个方框表示一条微指令,箭头指出微指令的衔接次序，方框内给出微指令所执行的操作，在箭头附近给出该微指令的地址，在方框内或旁边给出微指令字中的AM2910的命令码。

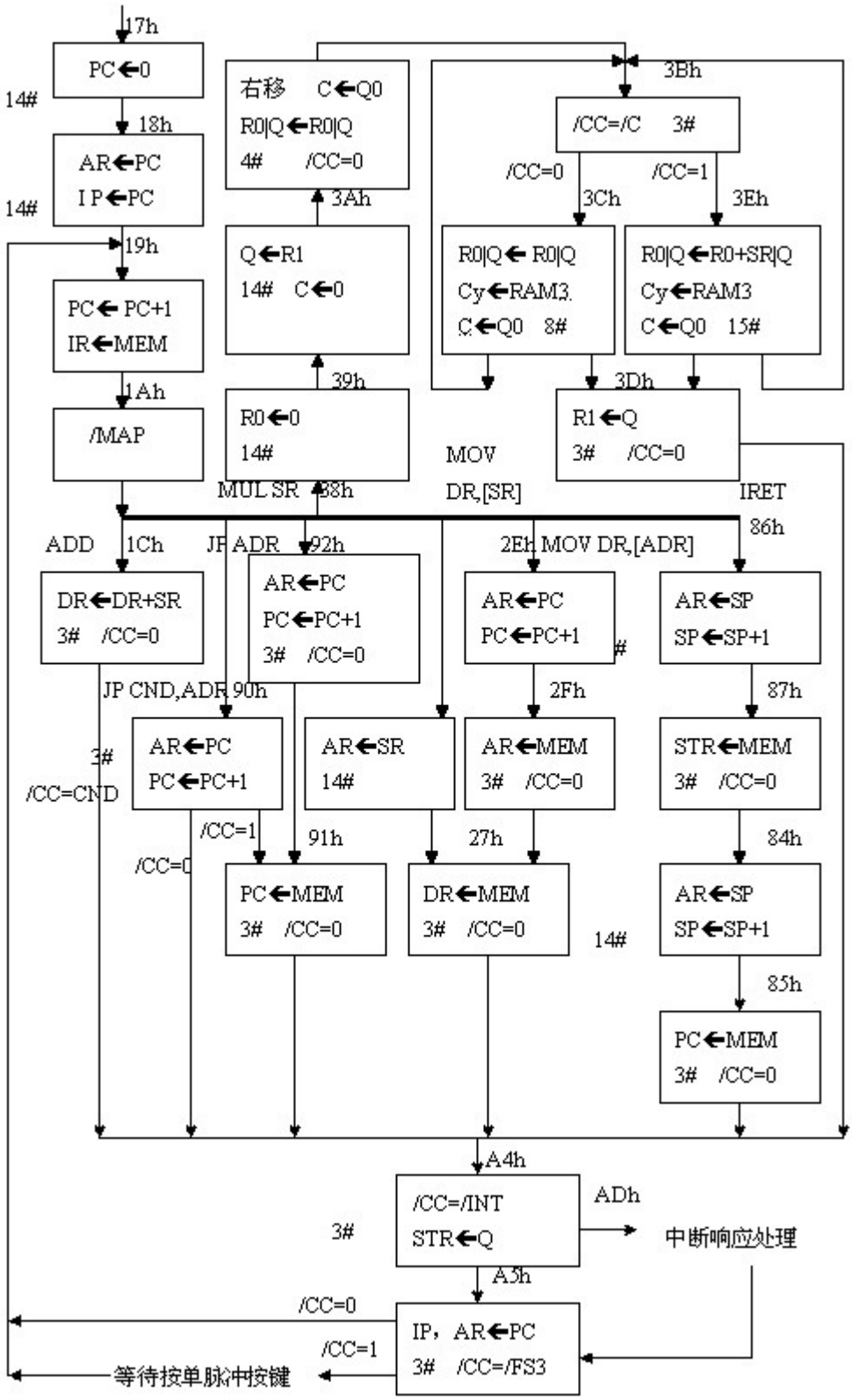

图3.6 七条指令的微程序执行流程图

自行设计新指令的微程序

所谓新指令，是指教学机支持的64条基本指令中未实现、留给学生自行设计与实现的11条机器指令，即指令汇总表中最后的11条指令。这11条指令的情况是:

—— 6位操作码已定，为D4、D8、DC、E0、E4、E8、EC、F0、F4、F8和FC，这是按8位长度的16进制方式给出的。其最低两位，可用于选择C、Z、V、S四个标志位作条件转移指令的判别条件。

—— 这11条指令的微程序段的入口地址已定为100h，110h，120h,...1FEh。这是由MAPROM器件的内容限定的,这些内容已写好在该器件的相应单元中。

—— 这11条指令没有相应的汇编语句名，执行的功能也未定义。但在使用时，必须使其指令格式与已实现的53条指令的格式类同，如要用C、Z、V、S作为判别条件，只能用指令寄存器的第9,8两位编码加以标明，作为写入用的寄存器编号只能通过IR7-IR4标明等等。使用不当，目前已给出的硬件可能无法直接支持。
设计新指令的微程序段将涉及以下几个问题:

—— 选定指令格式及功能，包括确定要用的操作码，指令中其它字段的内容分配与使用，本指令要实现的具体功能。

—— 按新指令的功能与格式，设计该指令的执行过程，即分成几步完成，每一步要实现的详细操作细节，各步之间的衔接次序等。

—— 将每一步中的操作，用一条微指令实现，即具体设计每条微指令各字段的具体编码值，既包括控制码的各字段，也包括下地址字段，形成下地址用到的条件码等等。

—— 将设计好的微码，装入控制存储器的相应单元。

—— 设计一个使用新、旧指令的用户程序，检查程序运行的正确性，以确定新指令是否正确执行，对新指令的执行过程仔细调试，直到得到满意的结果。

这一过程中的向控存中装入新指令的微码有两种方法，一是通过水平板上的开关与按键直接拨入，具体操作方法参见教学计算机补充材料；二是在程序中用LDMC指令指令实现自动装入，其具体操作步骤介绍如下。

作为例子，最简单的方法，是抄一条现有指令作为新指令予以实现。例如，操作码选D4，指令格式选为D4 DR，SR,,实现DR+SR→DR的功能。它就是ADD DR，SR那条指令，差别仅是操作码由04变为D4。查指令汇总表，D4操作码的微指令的入口地址应为100h,故将1Ch地址中的微指令(实现ADD DR，SR的操作)中的内容复制到100h单元，就完成了这条新指令的微程序设计的过程。看下面一个程序例子。

```
<A800
800: MOV R8, 240 ； 为指令的目的寄存器赋初值
MOV R9， 360 ； 为指令的源寄存器赋初值
MOV R1， 900 ； 微码在内存的首地址
MOV R2, 1 ； 微指令条数
MOV R3， 100 ； 微码在控存中的首地址
LDMC ； 用R1,R2,R3作为参数,装入微码
D489 ； 新指令的二进制执行码
RET
<E900
```

900： 0029…0301….b090….0088

以上用到的数值均为16进制。在监控命令工作时，输入均用16进制数，且都不能跟h字符。

该程序的功能是将240和360两个16进制形式的整数分别送入R8和R9。用新指令 (机器码为D489：操作码为D4，DR选R8,SR选R9) 完成两个寄存器的内容相加，结果写入R8。

该程序当中的4条指令，实现的是装入新指令的微码。微码在内存的首地址为900，四个字的内容为0029，0301，b090，0088，是控存1CH单元的内容，可以用监控程序的E命令键入。

该程序可以用监控程序的命令打入。倒数第2行的D489是新指令的机器码，不能在A命令方式下打入。具体操作过程，可以在A命令方式下，先在此处打入任何一条单字指令，例如，MOV R0，R0。整个程序输入后，再将该单元的内容用E命令改为D489，该程序运行过程中，在为R8,R9赋值后，接着装入新指令的微码，再执行新指令，最后返回监控程序以结束该程序的执行过程。

该程序运行结束后，用R命令检查程序的执行结果，R8的值应变为05A0。

从这个例子可以得出以下几个结论:

—— 新旧指令可以用在同一程序中；
—— 新指令在每次教学机重新加电后，至少得重新装入一次对应的微码； 仅在装入相应微码后，新指令才能执行，即已将新指令追加到教学的指令系统中；
—— 新指令无汇编码(因汇编程序实现在前，新指令实现在后)，故在程序中，只能通过机器码使用新指令；否则必须去扩展有关的汇编程序。
—— 装入新的微指令与使用新指令变得非常容易，同学的精力就可以全部集中到微程序设计方面来。但必须想到，新、旧指令的微程序之间存在着如下协调与配合关系:
前边的例子中，只设计了新指令的具体执行功能，执行前的取指过程和执行后的判中断、与下条指令的衔接等均使用了原微程序的有关内容。从同学学习微程序设计的角度看，取指过程与每条指令完成后的相应处理是公用于所有指令的，而且比较简单，看懂原来的实现方法与细节，以及与每条指令执行过程的衔接方式，也就达到了深入掌握的程度，故一般不必在自己设计的微程序中考虑这一部分内容。若有的学生想在自己的微程序中实现自己设计的取指等处理过程，必须保证在新旧指令衔接时不出现矛盾。最简单的方法，是在自己的多条新指令中，有几条指令有自己设计的取指与后续处理，它们不能与原有指令的微程序段正确衔接。但有一条新指令用原有指令的取指处理完成取指过程，有另一条新指令用原有指令的后续处理完成判中断，保证能正确与原有指令的取指过程衔接。当用这样的两条指令"夹"起的其它新指令序列出现在任何程序中时，每条指令均将能正确衔接执行。

设计不同格式与功能的新指令的执行步骤，以及每一步中的微指令字的各字段的编码，是学习计算机微程序设计的重点，也是学懂计算机指令执行过程的核心内容。