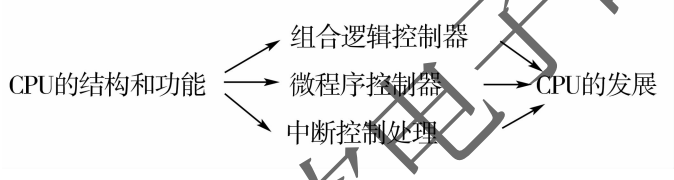
第 6 章　中央处理器

[单元概述]

冯·诺依曼体系结构的计算机硬件由控制器、运算器、存储器、输入设备和输出设备 5 部分组成，随着集成电路的出现及其集成度的提高，设计者将控制器和运算器集成在一个电路上，称做微处理器，这就是通常所称的中央处理器(Central Processor Unit——CPU)，主要完成信息处理的流程控制功能。

CPU 是计算机的核心部件，也是计算机组成原理课程研究的主要内容之一。在本章中，首先从层次结构上介绍一个计算机的功能，然后再分别介绍中央处理器的基本概念，如 CPU 的结构、功能，工作原理和发展等。

本章线索：



[教学重点与难点]

重点：

(1)CPU 的功能和基本结构。

(2)指令执行的过程。

(3)控制器的功能和基本原理。

(4)中断系统。

(5)指令流水线。

难点：

(1)指令执行的过程。

(2)控制器的功能和基本原理。

(3)中断系统。

6.1　CPU 的结构和功能

完整的计算机应用体系包含软件和硬件两大部分，软件和硬件的分界线就是指令系统，通

常,高级语言编写的源程序不能被计算机的指令系统所直接执行,必须经过编译后才能由指令系统执行。而对于指令系统中每一条指令的执行,则需要硬件的支持才能得以实现。CPU 做为计算机的大脑,就由它来处理、控制其他各部件的协调工作,连续不断地执行指令,并依据当前正在执行的指令和它所处的执行步骤,形成并提供在这一个时刻整个机器中各个部件需要使用到的各种控制信号和时序信号。

一般来说,CPU 具有指令控制、操作控制、时间控制和数据加工四个主要功能。

(1)指令控制:根据上述论述,计算机解决某个问题的过程,就是程序的顺序控制、执行的过程,而程序的顺序控制称为指令控制。

(2)操作控制:程序的执行过程最终会分解成一条具体指令的执行,其中要涉及到计算机中的若干个部件,控制这些部件协同工作,要靠各种操作信号的有机配合,CPU 管理并产生由内存取出的每条指令对应的操作信号,把各种操作信号送往相应部件,从而控制这些部件按指令的要求进行动作。

(3)时间控制:计算机中机器指令的操作过程是有严格时间规定的,对各种操作实施时间上的控制,称为时间控制。

(4)数据加工:要完成具体的任务,就必须要涉及到数值数据,逻辑变量,以及其他非数值数据(如字符、字符串)的处理,这些处理就是数据的加工处理。

6.1.1 CPU 的组成

CPU 是用来执行指令的部件,它的组成和实现与计算机的指令系统关系密切,那么,在其组成部件中就要保证包含实现指令系统中所有指令的各种部件、数据通路和控制信号。所以,作为计算机系统的核心部件,CPU 必须包括能对指令进行分析处理的一套硬件设备。具体如下:

(1)指令寄存器 IR(Instruction Register):用来存放由内存中取出的指令,在指令执行的过程中指令应该一直保存在 IR 中。指令是控制其工作的依据,IR 的内容的改变就意味着一条新的指令的开始。

(2)指令译码器 ID(Instruction Decoder):暂存在指令寄存器中的指令,其操作码部分经译码后才能识别出当前要执行的指令是一条什么指令,这就是指令译码器的主要功能。

(3)程序计数器 PC(Program Counter):用来存放即将执行的指令的地址,具有计算的功能,PC 的值是程序执行位置的体现。当程序开始执行的时候,PC 内装有程序的起始地址。当程序开始执行的时候,每执行一条指令,PC 增加一个量,这个量就等于指令所含的字节数;当程序执行转移类指令的时候,该类指令执行的最终结果就作为 PC 的值,以实现转移。

(4)程序状态字寄存器 PSW(Program Status Word):也可以叫做标志寄存器 FR。PSW 用来存放两类信息:一类是体现当前指令执行结果的各种状态信息。另一类是存放控制信息,比如是否允许中断等。

(5)算术逻辑运算单元:CPU 内需要一个算术逻辑运算单元,以便可以增加 PC 的值,使其指明下一条指令的地址;并且执行各种算术和逻辑运算。

(6)寄存器组:不同形式的指令都需要不同数量的寄存器存储操作数,所以需要一组用于读写的寄存器。

(7)时序部件:用于产生计算机系统所需的各种时序(定时)信号,计算机中的各种控制信号

都有很强的定时性。指令告诉计算机要做什么，并由时序线路确定什么时刻去做。时序线路由脉冲源、启停线路和时序信号形成线路组成。脉冲源产生的时钟频率作为主频，它是时序信号的基础。启停线路负责时钟信号的通断。时序形成线路负责产生周期信号、节拍信号和节拍脉冲信号。

(8)微操作控制部件：根据 IR 的内容(指令的内容)、PSW 的内容(状态的内容)以及时序线路的内容(执行时刻的信息)可以由微操作控制形成部件产生控制整个计算机系统所需的各种控制信号(称为微命令)，根据设计方法不同，其形成部件也不大相同，主要可分为：组合逻辑控制器、微程序控制器和门阵列控制器。本章只介绍组合逻辑控制器和微程序控制器。

(9)数据通路：CPU 要完成某一特定的功能，除了上述的寄存器外，还要使信息在各寄存器之间流动，所以还会需要不同的数据通路，指挥和控制 CPU、主存及输入输出部件之间的数据流动方向。

综上所述，CPU 由寄存器、运算器和程序计数器、指令寄存器、指令译码器、时序产生器和操作控制器等几个部分组成(如图 6.1 所示)。寄存器组用来存放各种数据；指令寄存器和指令译码器用来存放和分析 CPU 正在执行的指令；运算器(算术逻辑运算单元 ALU)用于实现对数据的算术运算和逻辑运算。定时与控制部分中的操作控制器产生各类操作控制信号，以便在各寄存器之间建立数据通路，时序产生器对各类操作控制信号进行定时，以便进行时间上的协调工作。因为运算器在本书中已经介绍过，本章的重点将放在控制器的介绍上。

6.1.2　指令的执行过程

计算机的工作过程就是执行机器指令的过程，总是遵循着“取指令，执行指令，取下一条指令，执行下一条指令……”周而复始的工作，直到停机为止。仔细分析指令的执行过程，才能进一步了解控制器的组成和工作原理。

一条指令的执行过程应该是从主存中取出该指令开始到执行完该指令功能为止。指令的执行步骤一般要经过到内存中读取指令、控制器分析指令、控制器按指令要求的具体功能，用一到几个执行步骤，“驱动”计算机中相关部件完成指令的运算、操作工作，并在这个过程中准备好下一条指令的地址到程序计数器 PC 中，至此本条指令的功能执行完毕。这个过程包括如下两个阶段：

取指令阶段：取指令操作对所有指令都是相同的。

执行指令阶段：这个阶段是根据指令操作码的不同，对操作数实施不同的算术运算和逻辑、移位运算。

6.1.3　时序控制系统

6.1.3.1　控制方式

控制单元用于完成一条指令执行的过程，这个过程实质上就是依次执行一条确定的微操作序列的过程。由于不同指令所对应的微操作数及其复杂程度不尽相同，所以每条指令和每个微操作所需的执行时间也不一样。通常把用于形成控制不同微操作序列的时序控制的方式称为控制单元的控制方式。一般分为同步控制方式、异步控制方式和组合控制方式。

(1)同步控制方式。同步控制方式，就是所有指令的所有操作都是事先确定的，并受统一的

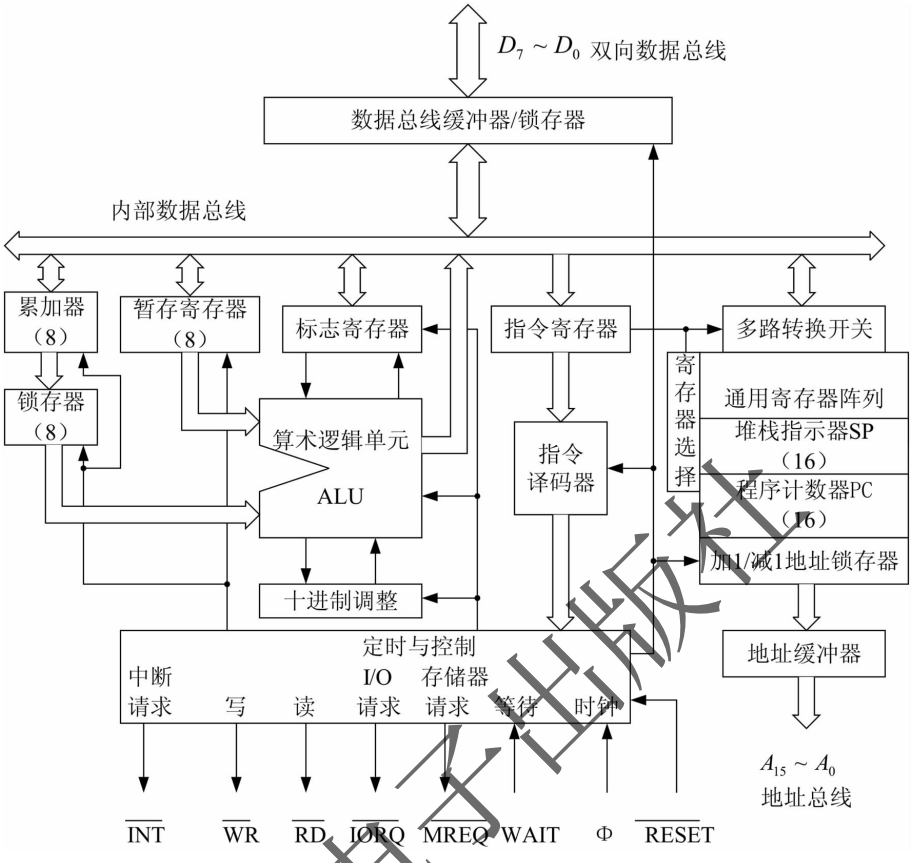


图 6.1 CPU 主要组成部件逻辑结构示意图

基准时间的控制。

(2)异步控制方式。在这种控制方式中,不存在基准时间信号,也没有固定时钟用于同步指令,每条指令执行多长时间都可以。

(3)组合控制方式。这种方式是同步控制方式和异步控制方式的组合,这是计算机中较为常用的方式,它对不同的指令采用了微操作大部分统一,小部分区别对待的方式。

6.1.3.2 基本概念

计算机是一个非常复杂的系统,CPU 执行指令时,各种操作都按精确的定时进行的。这时就需要所有指令执行的过程中有一个基准时间,如果把所有指令都能完成的时间段作为基准时间,显然很不合理,也是对计算机中各种资源的巨大浪费。通过进一步的分析可以得出,计算机内的各种操作一般可以分为在 CPU 内部的操作和对主存的操作。其中,在 CPU 内部的操作速度快,而访问主存的操作时间长,所以通常把一次访问主存的时间定为指令的基准时间,又称为机器周期。这样的话,无论执行一条什么样的指令,都会包含一个从主存中取指令的机器周期,再包含若干个执行指令的机器周期,每个机器周期内又可以完成若干个微操作,这些微操作完成的时间可以用时钟信号控制。这种由指令周期、机器周期、时钟周期组成的时序系统就组成了多级时序系统。

(1)指令周期。指令周期就是一条指令从取出到执行结束所需要时间,称为指令周期。指令周期的大小依指令不同而异,不是一个固定的值,因此一般不用它作为时序一级控制信号使用。

(2)机器周期。CPU 访问一次存储器所需的时间称为一个机器周期,它是根据指令执行的基本过程划分的,根据指令执行的各个阶段可以将一个指令周期划分为取指令周期、取操作数周期和执行周期。例如,取出指令时,将程序计数器的内容送到地址缓冲器,由此所选择地址的内容通过数据缓冲器送到指令寄存器,这个过程所需要的时间就是一个 CPU 周期。有些计算机将 CPU 周期的长短定义为一次内存的写操作,但无论读操作还是写操作,都必然是一次系统总线的传送操作,因此也常常被称为总线周期。

(3)时钟周期(节拍)。计算机操作的最小时钟单位。通常把定时振荡器两个相邻脉冲上升沿之间的间隔称为一个时钟周期,在这个周期内完成 CPU 内部一些最基本操作。例如:数据通过 ALU 完成一次算术逻辑运算的时间,或者数据从一个寄存器传到另一个寄存器并建立可靠连接的时间。在任何一个计算机系统中,这个值都是常量。

图 6.2 所示为用 74LS138(3－8 线译码器)产生节拍信号的电路图,其中 C,B,A 为该译码器的三个输入端,Y_0～Y_7为译码器的 8 个输出端。C,B,A 的输入值分别来自于 D 触发器的 Q 端(这 3 个 D 触发器串连,形成了一个计数器电路)。上电后,可以产生控制信号“0”,将 3 个 D 触发器的输出端 Q 清“0”,此时根据 3－8 线译码器的性质,$Y_0=0$,其他输出均为“1”;当第一个主时钟的上升沿到来时,将第一个 D 触发器的输入值 D(也就是 $\overline{Q}=1$)送入输出端 Q,此时 A=Q=1,B,C 不变,$Y_1=0$;当第二个主时钟的上升沿到来时,第一个 D 触发器的输入值翻转,并且第二个 D 触发器在 $\overline{Q}$ 上升沿的触发下,发生翻转,此时 A=0,B=1,C=0,$Y_2=0$,这样周而复始的循环,就可以产生图 6.3 所示的节拍电路图。其中 Y_0～Y_7是一个机器周期中的 8 个节拍,Y_0为节拍 0,Y_1为节拍 1……以此类推,Y_7为节拍 7。在同一个机器周期中 Y_0～Y_7只出现一次,周而复始。

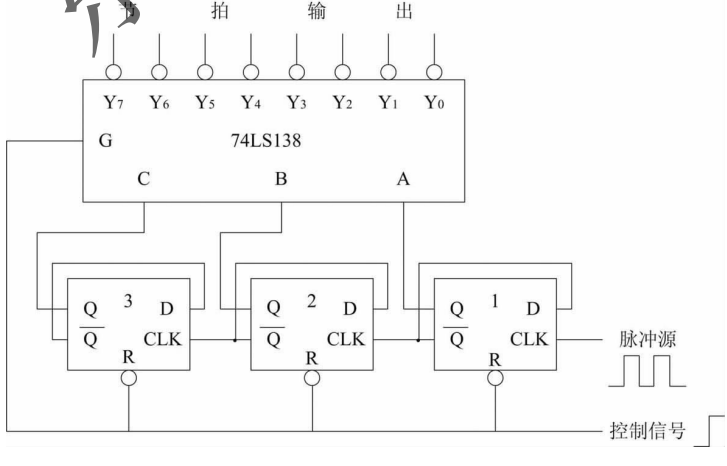


图 6.2　节拍信号发生器举例

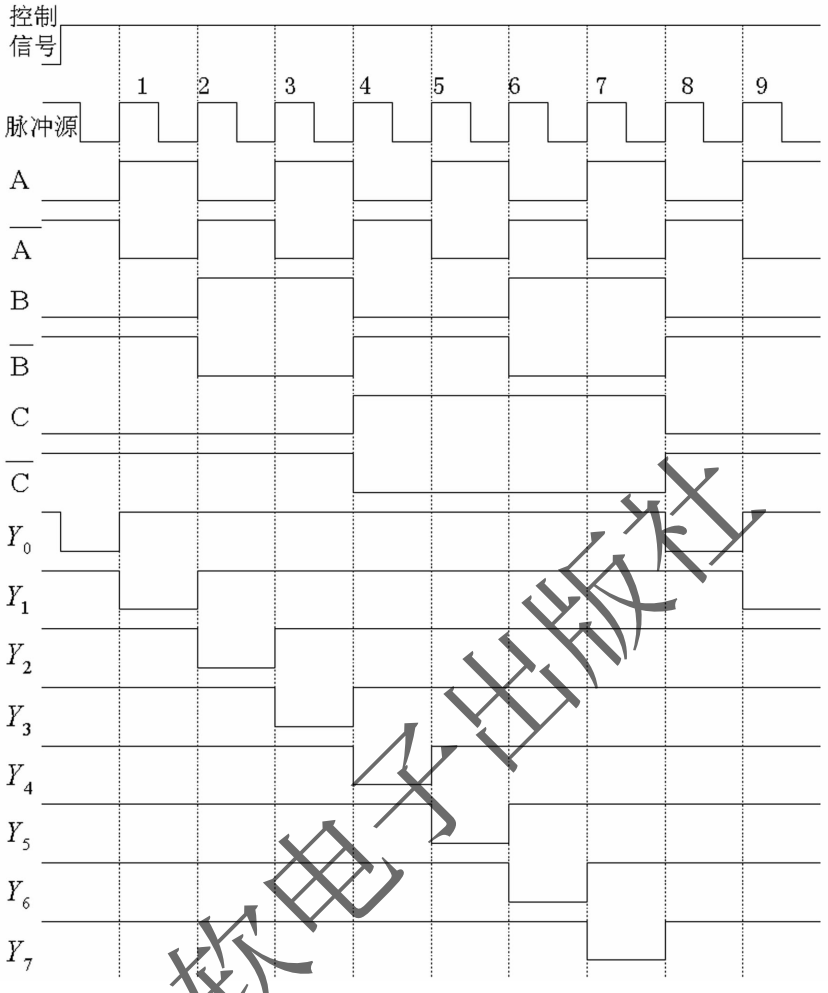


图 6.3 节拍电路图

6.1.3.2 一条完整指令操作流程图

任何一条机器指令的执行过程，实际上是由控制器产生一系列微操作控制信号的过程，将这些微操作控制信号合理地分配在各个机器周期的各个节拍中，便可以构成各条机器指令的操作流程图。通常一条指令的执行可以分成以下几个周期。

(1)取指周期。取指周期用来完成把指令从内存中取出并对操作数寻址的过程，具体为：

① 把现行的指令地址送到主存储地址寄存器(MAR)，记作 PC→MAR；

② 向主存发出读命令；

③ 形成下一条指令的地址，记作 PC+1→PC；

④ 将主存中的内容通过数据总线读至数据缓冲器(MDR)，记作 M→MDR；

⑤ 将 MDR 中的内容送到 IR，记作 MDR→IR；

⑥ 将指令中操作码进行译码。

(2)取操作数周期。取操作数周期用来完成取操作数的有效地址的任务,具体为:

① 把指令的地址码送给 MAR;

② 向主存发出读命令;

③ 将主存中的内容通过数据总线读至 MDR,记作(MAR)→MDR;

④ 将有效地址送至指令寄存器的地址字段。

(3)执行指令周期。执行指令周期用来完成指令的具体功能,由于不同指令的执行指令的周期也不一样,可以按非访存周期、访存周期和转移类指令周期分别讨论。

① 非访存指令。这种指令不需要在执行过程中访问主存。如清除累加器指令 CLA,这条指令只需要完成清除累加器的工作就可以了。

② 访存类指令。这类指令在执行阶段需要访问主存。如加法指令 ADD X,R1 指令,该指令在执行时就需要完成把累加器中的内容与对应主存 X 中的内容相加,结果送给累加器的操作。

③ 转移类指令。这种指令在执行的过程中也不需要访问主存,如无条件转移指令 JMP X,这条指令在执行时就将指令的地址码字段送到 PC 中去。

例 6.1:假定有指令 ADD　R1,NUM,该指令的功能是把主存单元 NUM 的内容与寄存器 R1 中的内容相加,结果送到 R1,也就是:R1←(NUM)+(R1)。

运行此指令要求下列动作:

(1)读指令、分析指令;

(2)读取第一个操作数;

(3)进行"加法"操作;

(4)把结果装入 R1。

假定某计算机系统中每个机器周期内包含 4 个节拍(T_1～T_4),任何一条指令的取指令周期完成的操作是一样的。其公共操作可以描述如下:

• T_1拍:PC→MAR(地址寄存器)→外部地址总线。

• T_2拍:CPU 由外部控制总线向主存储器发出读指令。

• T_3拍:等待主存读出指令,并完成修改 PC 功能。

• T_4拍:从存储器读出的指令经数据缓冲器(MDR)置入指令寄存器 IR,并对其操作码进行译码,完成分析指令功能。

任何指令的其他机器周期的操作与指令功能有关,下面只讨论该取机器数周期的执行过程。

• T_1拍:将指令中的地址码 NUM 置入主存储器的地址寄存器(MAR)。

• T_2拍:向主存储器发出读命令。

• T_3拍:等待主存读出数据到 MDR。

• T_4拍:将 MDR 的内容置入寄存器中,进行加法,并把结果放回 R1 中。

应该注意的是,同样是取数指令,采用的寻址方式不同,需要的机器周期数也不同。

6.1.3.3 控制信号的产生

控制的运行原理,可以从几个方面来理解。

首先,从计算机的基本功能的层次考虑。计算机的基本功能就是执行程序,程序是机器指令的一个序列,因此,计算机应能自动地和连续地执行该机器指令序列中的每一条指令。自动地和连续地执行该机器指令的核心问题,在于应该按指令的序列关系,自动地逐条从内存中取出每一条指令并执行,而且这个过程应该是连续的。

再从计算机系统的层次结构上看,控制器的任务是在微体系结构层上分析 CPU 的构成,在该层次上,看到的是寄存器、ALU 和数据通路;而 CPU 执行一条指令的过程也就相应地变成执行一系列微命令的过程,比如选择两个寄存器的内容作为 ALU 的操作数,并将他们相加,然后将结果存回某个寄存器中。在一些机器上,这些信号直接由硬件产生的控制信号来控制,相应的控制部件称为组合逻辑控制器;而另外一些机器上,这些功能是由微程序产生的控制信号来控制,这样的控制部件称为微程序控制器。

该层次可以看成是指令系统层指令的解释器。在微程序控制器中,微程序就是上一层指令的解释器。而在组合逻辑控制器中,是由硬件直接解释执行指令,并不存在一个真正的程序来解释上一层的指令。总之,这一层的主要功能就是解释执行存放在主存器中的机器指令序列,这个过程是由控制器产生的一组微命令序列来控制实现的。

6.1.3.4 控制单元的外特性

图 6.4 是反映控制单元外特性的框图。

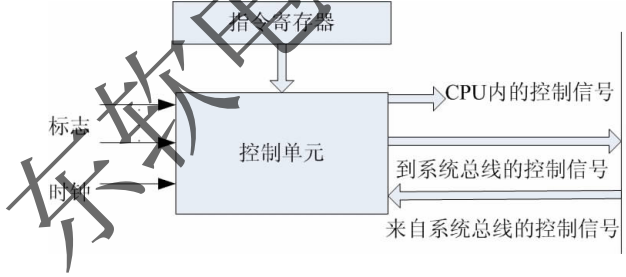


图 6.4 控制单元外特性

在图 6.4 中,输入信号包括:

(1)时钟。该信号可以使控制单元按一定的先后顺序、一定的节奏发出各个控制信号。

(2)指令寄存器。在该寄存器中存储的指令操作码决定了不同的指令周期需要完成的不同的操作。

(3)标志。这个输入信号可以让控制单元根据 CPU 现在的不同状态产生不同的控制信号。

(4)来自系统总线的控制信号,比如,中断请求、DMA 请求。

输出信号包括:

(1)CPU 内的控制信号。可以用于 CPU 内的寄存器之间的传送等操作。

(2)送至系统总线的信号。比如说中断响应信号等。

6.2　组合逻辑控制器

6.2.1　组合逻辑控制器的概念

组合逻辑控制器也称为硬布线控制器，它是由门电路和寄存器构成，它是早期设计计算机的一种方法，采用不同的逻辑电路产生固定的时序控制信号，从而实现不同的指令操作。一旦组合逻辑电路设计好之后，它的操作功能就固定不变了。也就是说，组合逻辑控制器设计好后无法实现新的逻辑电路功能。正因为如此，这种设计的方法使得整个控制器的设计硬件的代价很高，属于早期设计计算机的一种方法。随着新一代机器及超大规模电路技术的发展，组合逻辑设计思想又得到了重视。近年来，在某些超高速新型计算机结构中，又选用了这种硬布线的控制思想，或与微程序控制器混合使用。

作为操作控制器，组合逻辑控制器需要根据指令的要求和指令执行流程，按照一定顺序发出各种操作控制信号，控制有关部件完成指定的操作。一般组合逻辑控制器的输入有：操作码译码器的输出、时序信号、运算结果的标志、状态和其他条件。

图 6.5 是组合逻辑控制器与其他部件的关系图，从图中可以看出，组合逻辑控制器的输入有：(1)来自指令译码器的译码输出信号 Im；

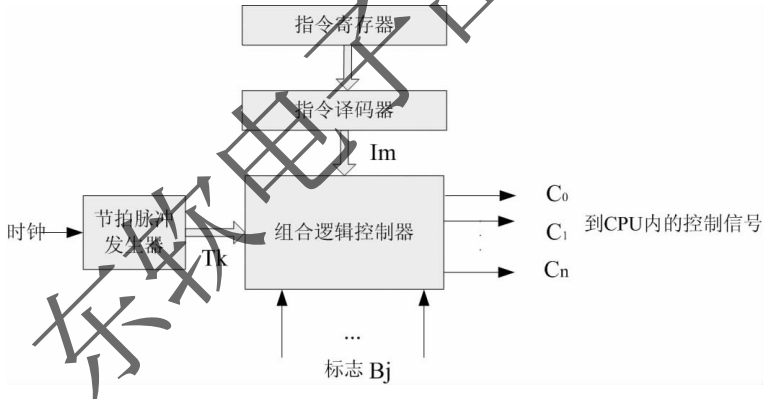


图 6.5　组合逻辑控制器关系图

(2)来自执行部件的操作反馈信息 Bj(也就是标志寄存器中的值)；

(3)来自时序控制部件的时序信号 Tk。

组合逻辑控制器的输出有：

(1)微操作控制信号 Cn：对执行部件进行操作控制；

(2)其他控制信号：根据条件变量改变时序发生器的计数顺序，以缩短指令周期。

综上所述，微操作控制信号 Cn 是 Im、Tk 和 Bj 的函数，即：Cn＝F(Im，Tk，Bj)。

6.2.2　组合逻辑控制器设计举例

一般说来，组合逻辑控制器的设计过程可以分为三个步骤。第一步是进行微操作的综合，第二步是将这些逻辑表达式简化为最简单的逻辑表达式，第三步是用逻辑电路去实现所有的逻辑表达式，便形成了“组合逻辑控制器”。

例 6.2：设计某模型机的控制器，该机指令系统包含五条指令，设计时采用组合逻辑控制器，并采用同步控制，每个机器周期包含 4 个节拍。

指令如下：

CLA	清零指令	指令功能：0→AC；
ADD AC,M	加法指令	(AC)+(M)→AC；
SUBAC,M	减法指令	(AC)−(M)→AC；
JMP m	跳转指令	m→PC；
NOP	空指令	延时

(1)分析指令，列出所有指令的操作流程，如图 6.6 所示。

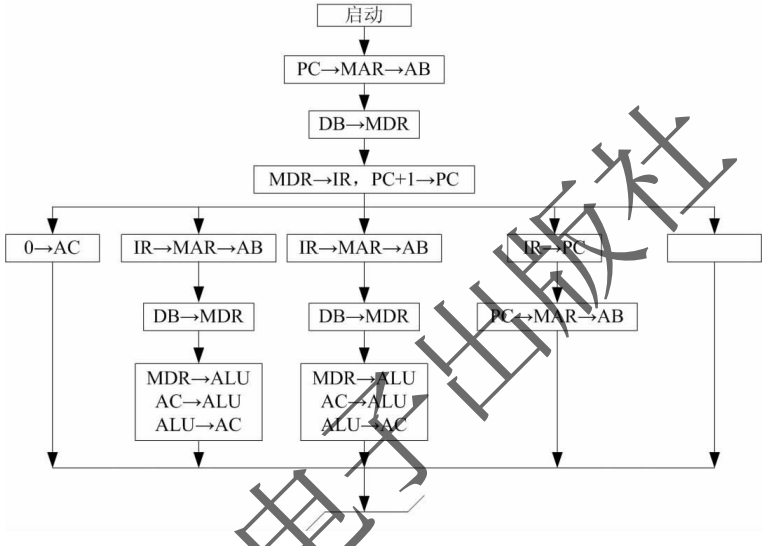


图 6.6 指令操作流程图

(2)对执行流程进行时序划分(分配周期和脉冲)，如图 6.7 所示。

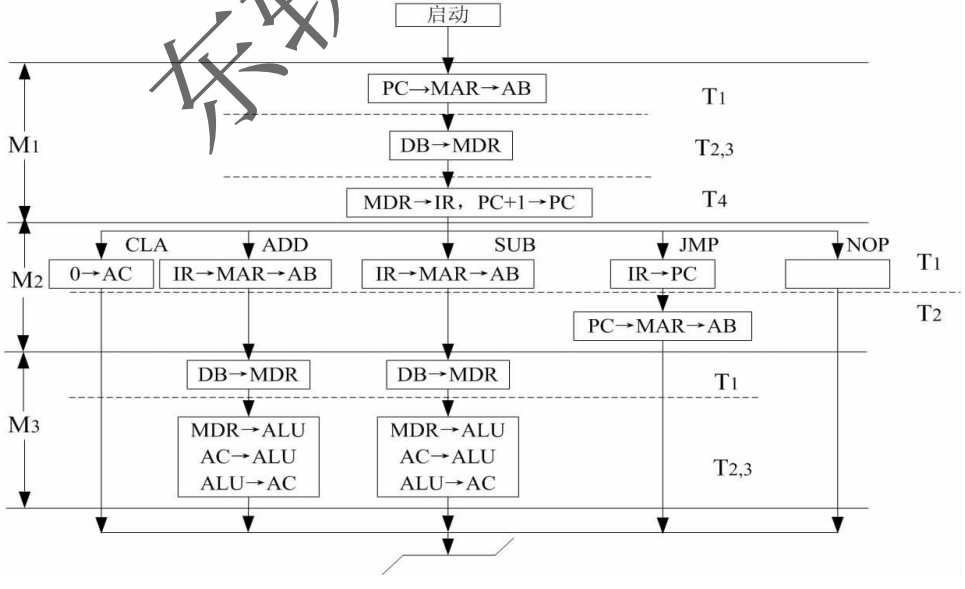


图 6.7 执行流程的时序划分

(3)总结所有微操作如表 6.1 所示。

表 6.1　　微操作表

微操作	功能	微操作	功能
C0	PC→MAR→AB	C6	DB→MDR
C1	PC+1→PC	C7	AC→ALU
C2	IR→PC	C8	ALU→AC
C3	IR→MAR→AB	C9	0→AC
C4	MDR→IR		
C5	MDR→ALU		

(4)列出各微操作的逻辑函数表达式：

C0：PC→MAR→AB＝M1 · T1＋M2 · T2 · JMP

C1：PC＋1→PC＝M1 · T4

C2：IR→PC＝M2 · T1 · JMP

C3：IR→MAR→AB＝M2 · T1 · (ADD＋SUB)

C4：MDR→IR＝M1 · T4

C5：MDR→ALU＝M3 · T3 · (ADD＋SUB)

C6：DB→MDR＝M1 · T2 · T3＋M3 · T1 · T2 · (ADD＋SUB)

C7：AC→ALU＝M3 · T3 · (ADD＋SUB)

C8：ALU→AC＝M3 · T3 · (ADD＋SUB)

C9：0→AC＝M2 · T1 · CLA

一般来说，微操作还应考虑状态条件的约束，此例未涉及。

(5)设计逻辑电路实现控制器功能。

① 采用可变 CPU 周期，固定时钟脉冲方式，设计时序产生器；

② 由微操作逻辑函数表达式生成逻辑电路，产生各微操作控制信号。

6.3　微程序控制器

从上述分析可以看出，组合逻辑控制器设计好后无法实现新的逻辑电路功能。正因为此，这种设计的方法使得整个控制器的设计硬件的代价很高，属于早期设计计算机的一种方法。这些组合逻辑控制电路存在的缺点，促使人们不断地找寻不同的替代方案，微程序控制的概念是英国科学家威尔克斯(Wilkes)于 1951 年提出的，主要思想是用保存在只读存储器中的专用程序替代逻辑控制电路，因其设计思想与组合逻辑控制器相比，具有很大的灵活性，所以在计算机设计中逐渐取代了早期的组合逻辑控制器。

实际上，微程序设计技术是利用软件方法来设计硬件的一门技术，即仿照通常的程序解题方法，把操作控制信号编成所谓的“微指令”，存放在一个只读存储器里(称为控制存储器 CM)。当计算机运行时，逐条地读出这些微指令，从而产生全机所需要的各种操作控制信号，使相应部件执行所规定的操作。其工作原理就是编写每一条机器指令的微程序，这个微程序是按执行每

一条机器指令所需的微操作的先后顺序而编写的。这时,一条机器指令对应一个微程序。

6.3.1 微程序设计中的有关术语

(1)微命令和微操作。控制部件通过控制线向执行部件发出各种控制命令,通常把这种控制命令叫做微命令,例如打开或关闭某部件通路的控制门的电位,或是对触发器、寄存器进行同步打入、置位、复位的控制脉冲。而执行部件接受微命令后所进行的操作,叫做微操作。微操作在执行部件中是最基本的操作。微操作可分为相容性的微操作和相斥性的微操作。其中,相容性的微操作是指在同时或同一个 CPU 周期内可以并行执行的微操作;相斥性的微操作是指不能在同时或不能在同一个 CPU 周期内并行执行的微操作。

(2)微指令。在计算机的一个 CPU 周期中,一组实现一定操作功能的微命令的组合,构成一条微指令。一般由操作控制和顺序控制两大部分组成。

(3)微程序。一条机器指令的功能是用许多条微指令组成的序列来实现的,这个微指令序列通常叫做微程序。

(4)微程序存储器。存放微程序的专用存储器又叫作控制存储器(CM—Control Memory),通常用只读存储器 ROM 实现,一般对它的要求是速度快、工作可靠。

在微程序设计中还要注意以下几个问题:

(1)CPU 周期与微指令周期的关系。在串行方式的微程序控制器中,微指令周期等于读出微指令的时间加上执行该条微指令的时间。一般来讲,一个微指令周期时间应该设计得恰好和 CPU 周期时间相等。

(2)机器指令与微指令的关系。

① 一条机器指令对应一个微程序,这个微程序是由若干条微指令序列组成的。因此,一条机器指令的功能是由若干条微指令组成的序列来实现的。简而言之,一条机器指令所完成的操作划分成若干条微指令来完成,由微指令进行解释和执行。

② 从指令与微指令、程序与微程序、地址与微地址的一一对应关系来看,前者与内存储器有关,后者与控制存储器有关。

③ 每一个 CPU 周期就对应一条微指令。

6.3.2 微程序控制器的原理

微程序控制器的主要特点是用控制存储器等替代组合逻辑的操作控制部件。此时指令的执行过程是由微程序中微指令执行的先后次序决定的。微程序控制器的基本工作原理为:当计算机执行程序时,总是按照指令,把计算机中 PC 的内容作为指令的地址,从主存储器中取出指令,放到指令寄存器 IR 中。微程序控制器根据机器指令操作码决定微程序的入口。选择控制存储器 CM 中该机器指令微程序的第一条微指令,送入微指令寄存器 uIR。微指令的操作控制字段,给出各种控制信号(微命令),控制计算机各个部件完成指定操作。同时微指令的顺序控制字段给出后继微指令的地址,按照规定的顺序由控制存储器中取出第二条微指令,直到完成机器指令的功能为止。

微程序控制器原理框图如图 6.8 所示。它主要由控制存储器、微指令寄存器和地址转移逻辑组成。

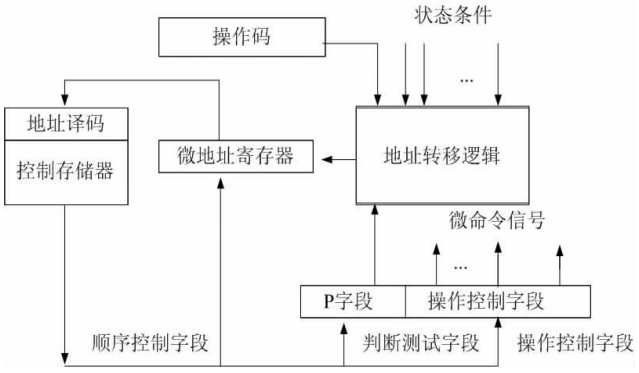


图 6.8　微程序控制器原理框图

其中：

• 控制存储器：用来存放实现全部指令系统的微程序，它是一种只读型存储器。要求速度快，读出周期短。

• 微指令寄存器：微指令寄存器用来存放由控制存储器读出的一条微指令信息，分为微地址寄存器和微命令寄存器两个部分。其中微地址寄存器决定将要访问的下一条微指令的地址，微命令寄存器则保存一条微指令的操作控制字段和判别测试字段的信息。

• 地址转移逻辑：地址转移逻辑就承担自动完成修改微地址的任务。

• 微指令执行过程：微指令的执行过程与机器指令的执行过程很类似，任何一条微指令的执行过程也可以分为取微指令和执行微指令两个阶段，只是微指令是从控制存储器中取出后置入微指令寄存器中，才能被执行。

上述工作涉及到两个层次，一个层次是程序员所看到的传统机器级，包括指令工作程序和主存储器；另一个层次是设计者所看到的微程序级，包括微指令、微程序和控制存储器。

6.3.3　微指令的结构

微程序设计的关键是如何确定微指令的结构。

微指令与机器指令类似，由微操作码和微地址码构成，其中微操作码部分用来确定该微指令所能产生的微命令，因此可以称之为“控制字段”，微地址码用来确定将要执行的下条微指令在控制存储器中的地址，其作用和机器指令的地址码不太相同。

6.3.3.1　微指令的控制字段

根据微指令中控制字段构成方式的不同可以分为水平型和垂直型两种格式。

(1)水平型微指令。水平型微指令是指能并行产生多个微命令的微指令，又分为直接表示法、编码表示法和混合表示法，如图 6.9 所示。该类型指令适用于大中型高速计算机，而且要求计算机具有充分的并行操作部件和数据通路。

① 直接表示法。微指令操作控制字段中的每一位代表一个微命令，该位为 1，表示选用该命令，为 0 表示不选用该命令。在这里，控制字段具有最高的并行性，多个微命令可以同时为 1，简单直观，其输出可直接用于控制，不需要译码，因此速度快；但是微指令字较长，因而使控制存储器容量较大，而且控制字段的总位数应是全机微命令的总个数，而它们同时置 1 的可能性很少，并且指令字比较长，实现转移和立即数比较困难。

② 编码表示法。为了克服直接表示法的缺点，通常可以把控制命令按其操作性质分为几组，每组内的微命令是互斥的且采用二进制编码来表示，然后通过小组(字段)译码器产生操作

控制信号。这样，就用较少的二进制信息位表示较多的微命令信号，使微指令字长大大缩短；但是由于增加了译码延时，微程序的执行速度略有减慢。

③ 混合表示法。把直接表示法与编码表示法混合使用，以便综合考虑微指令字长、灵活性和执行微程序速度等方面的要求。

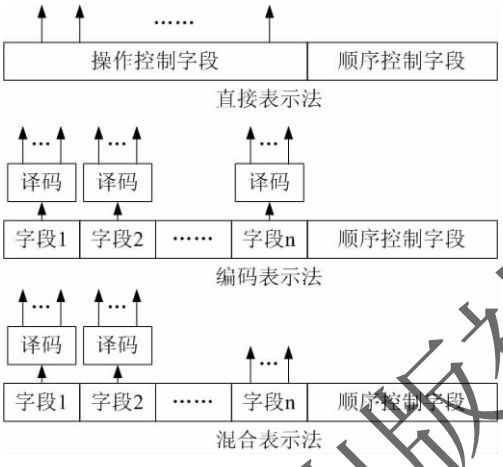


图 6.9 水平微指令的编码方法

例 6.3：某机的微指令中包含 4 个控制字段，每个字段可分别激活 2，4，3，1 种控制信号。分别采用直接表示法和编码表示法设计控制字段，需要各取几位控制字段。

① 采用直接表示法，需要的控制字段位数与控制信号数相同，为 2＋4＋3＋1＝10 位。

② 采用编码表示法时，根据每个字段需要激活的控制信号数，再加上每个字段中预留一个码字表示不激活任何一条控制线，则 4 个控制字段需要表示 3，5，4，2 种状态，对应位数为 2，3，2，1 位，则控制字段总位数为 2＋3＋2＋1＝8 位。

无论什么编码方式，其追求的目标是共同的，就是提高编码的效率，压缩微指令的宽度，保持微命令必须的并行性，且硬件电路尽可能简单。

(2)垂直型微指令。垂直型微指令功能比较简单，类似于机器指令的结构，一般适用于速度要求不高或结构简单的计算机。由微操作码决定该条微指令的功能，通常一条微指令中只有 1 个～2 个微操作命令，功能简单。垂直型微指令字短，但解释执行一条机器指令所需的微指令条数多。其特点为：

① 微指令字比较短，控制存储器 CM 位码利用率高；

② 实现转移和立即数操作方便；

③ 编制微程序容易，用户容易掌握；

④ 便于采用高级微程序设计语言，容易实现微程序设计的自动化。

综上所述，水平型微指令与垂直型微指令的区别在于：

(1)水平型微指令并行操作能力强，效率高，灵活性强，相比较下垂直型微指令则较差；

(2)水平型微指令执行一条指令的时间短，垂直型微指令执行时间长；

(3)由水平型微指令解释指令的微程序,具有微指令字比较长,但微程序短的特点,垂直型微指令则相反,微指令字比较短而微程序长;

(4)水平型微指令用户难以掌握,而垂直型微指令与指令比较相似,相对来说,比较容易掌握。

6.3.3.2　微指令的地址字段

计算机程序的执行大部分为顺序执行,因此在 CPU 中设立程序计数器 PC,根据 PC 的内容,取出一条指令,同时 PC 自增指向下一条指令的位置。若需要实现转移,由转移类指令完成。对于大多数下情况下为顺序执行的程序来说,设立 PC 和转移指令无疑是一个最好的选择。

而微指令的执行情况有所不同,任何一条指令都要取指令,取指令的操作对任何指令都是相同的,而各条指令中取数的操作与指令无关,只由寻址方式决定。若将这些相同的操作在每条指令对应的微程序中都重复存储,必然加大控制存储器的容量,这样显然不可取。合理的做法是将这些共同的操作只存储一次,由各条指令共同使用。这就意味着在微程序中,转移是经常发生的,为了这种频繁的转移,通常在每条微指令中设置下一条指令的地址部分,用以指明下一条微指令地址的产生方式和具体地址。因此微指令执行的顺序控制问题,也就是如何确定下一条微指令的地址。一般来说,由于微程序在控制存储器中有两种不同的存放方式:连续存放和不连续存放。在这两种方式下的地址字段格式不同。

(1)断定方式。断定方式是指组成一个微程序的多条指令在控制存储器中不连续存放,由各条微指令的地址字段给定下一条微指令在控制存储器中的地址,考虑到条件转移类微指令的需要,其地址字段应包含条件测试字段和下条微指令地址字段两部分。

(2)增量方式。也称计数器方式,这种方式同用程序计数器来产生机器指令地址的方法相类似。这时要求组成一个微程序的多条微指令在控制存储器中连续存放。在这里,微指令中可以不包含地址字段,但需要增加一个独立的微程序计数器(upc),顺序执行时,后继微地址直接由微程序计数器加一给出;转移分支时,按微指令判别字段和"状态条件"来形成后继微地址并修改微程序计数器。这种控制方式的优点在于微指令的顺序控制字段较短,微地址产生机构简单;但是由于多路并行转移功能较弱,也存在着速度较慢,灵活性较差的缺点。

以上介绍的技术属于静态微程序设计(对应于一台计算机的机器指令只有一组微程序,而且这一组微程序设计好之后,一般无需改变而且也不能改变,这种微程序设计技术称为静态微程序设计),相比之下,采用 EPROM 等可改写存储器作为控制存储器,通过改变微指令和微程序来改变计算机的指令系统,这种微程序设计技术称为动态微程序设计。它可根据需要改变指令系统,在一台计算机上实现不同的指令系统,扩大计算机功能。

6.3.4　微程序控制器实例

对于采用微程序控制方式的计算机系统,"微程序设计"的过程就是设计用来解释各条微指令的微程序的过程,一般来说,指令系统中包含多少条机器指令,就应该设计出多少个微程序,因此微程序设计的过程仍然是一个非常繁琐的过程,其设计工作量也是非常大的,但是其修改

起来也很方便。

例 6.4：设计某模型机的控制器，而该机指令系统包含五条指令，设计时采用微程序控制器。微指令采用断定方式，直接控制水平型微指令格式，并假定控制存储器的容量为 256×22 位。指令功能如表 6.2 所示。

表 6.2　　模型机指令功能表

指令助记符	功能说明
CLA	0→AC，把寄存器 AC 中的内容清零
SUB x	(AC)+(x)→AC，把寄存器 AC 的内容和存储单元 x 的内容相加，结果送到寄存器 AC
ADD x	(AC)+(x)→AC，把寄存器 AC 的内容和存储单元 x 的内容相加，结果送到寄存器 AC
JMP x	x→PC，无条件转移
NOP	空指令

第一步：分析指令格式。

若 CPU 的指令流程图如图 6.10 所示，其指令的格式如图 6.11 所示。

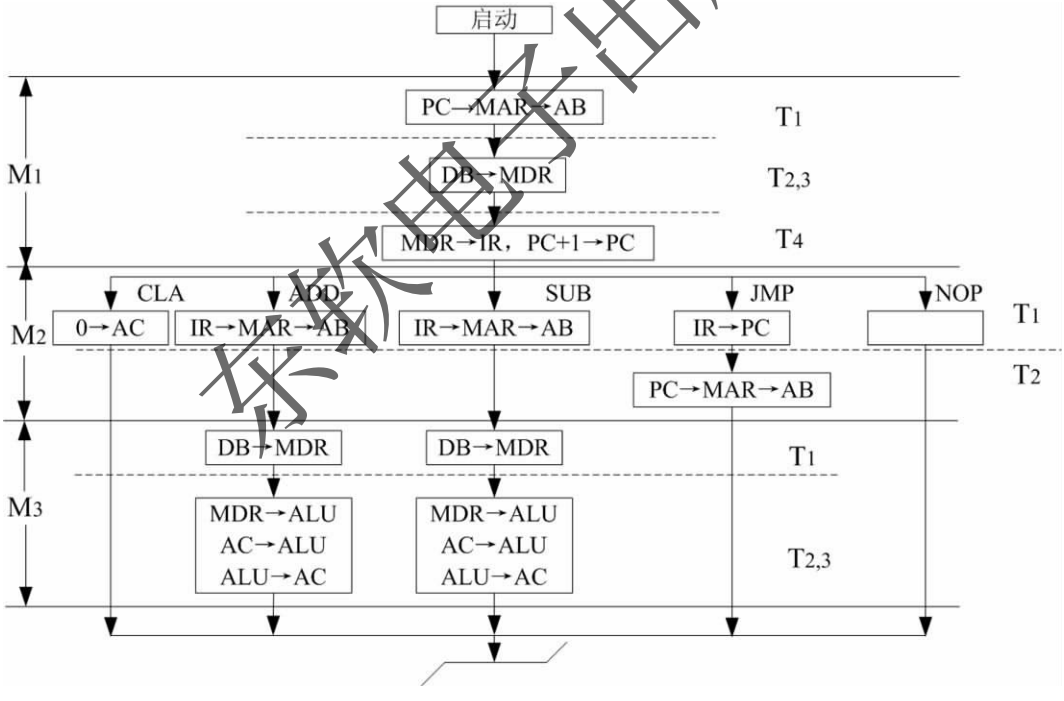


图 6.10　模型机指令流程图

操作码	地址码

图 6.11　模型机指令格式

在指令执行的过程中，使用如下的操作控制命令，给其逐一编号，得到共 10 个微指令。

(1)PC→MAR→AB，用于把 PC 中的送入地址总线。

(2)PC＋1→PC，用于自增 PC 中的值。

(3)IR→PC，用于把指令寄存器中下一条指令的地址送入 PC 寄存器。

(4)IR→MAR→AB，用于把指令寄存器中的操作数据的地址值送入地址总线。

(5)MDR→IR，用于把数据缓冲器中的值送入指令寄存器。

(6)MDR→ALU，用于把数据缓冲器中的值送入 ALU。

(7)DB→MDR，用于把数据总线中的值送入数据缓冲器。

(8)AC→ALU，用于把 AC 中的值送入 ALU。

(9)ALU→AC，用于把 ALU 中的值送入 AC。

(10)0→AC，用于把 AC 中的值清零。

根据图 6.10 可以看出，任何一条指令在取指令周期(M1)中都产生一条用于取指的微指令来完成，而第二、第三周期用于完成指令自身的执行指令周期。

第二步：设计微程序控制器逻辑图。

微指令格式一般采用水平微指令，下一条指令的地址 12 位，控制字段 10 位，每位表示一个微命令。微指令字长 22 位，微程序控制器逻辑框图如图 6.12 所示。

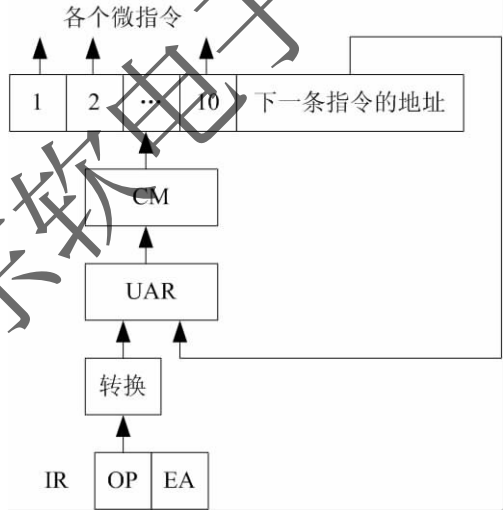


图 6.12　微程序控制器逻辑图

第三步：微程序编码。

以 ADD 指令为例，根据指令流程设计微程序中该条微指令内容，如表 6.3 所示。

表 6.3 微程序编码

	1 2 3 4 5 6 7 8 9 10	11－22
$(00)_8$	1 1 0 0 1 0 1 0 0 0	07
$(01)_8$	0 0 0 0 0 1 1 1 1 0	00
$(02)_8$	……	…
$(03)_8$	……	…
$(04)_8$	……	…
$(05)_8$	……	…
$(06)_8$	……	…
$(07)_8$	0 0 0 1 0 0 0 0 0 0	01

第一机器周期 M1 中执行 00 号地址中的微指令，该号地址中的内容主要用于完成取机器指令的过程，在这里共产生了 4 条微命令，并确定下一条为指令在控制存储器中的地址为 07。07 号地址中的微指令，主要用于完成寻址过程，在这里共产生了 1 条微命令，并确定下一条为指令在控制存储器中的地址为 01。01 号地址中的微指令，主要用于完成操作数和加法的过程，在这里共产生了 4 条微命令，并确定下一条为指令在控制存储器中的地址为 00。

利用上述的方法，可以设计完成一个计算机所需的所有机器指令，然后可以把它们写入控制存储器中，这样就完成了微程序设计的整个过程。微程序一旦设计完毕，一般情况下只允许读出来执行，这样的话，控制存储器一般由只读存储器构成。

6.4 中断控制处理

6.4.1 中断的概念

6.4.1.1 中断的提出

中断(Interrupt)是现代计算机输入输出的常用控制方式。早期的计算机没有中断系统，CPU 与外设之间交换信息常用的传送方式是程序查询方式。CPU 必须通过不断的查询，才能得知外部设备的工作状态。这种方式固然简单，但却存在着几个明显的缺点：

(1)CPU 的工作速度为每秒几十万次到每秒几百万次，而 CPU 启动外设工作仅需一两条指令，耗时几微秒，因此在查询过程中，CPU 长期处于踏步等待状态，使系统效率大大降低。

(2)CPU 在一段时间内只能和一台外设交换信息，其他设备不能同时工作。

(3)不能发现和处理预先无法估计的错误和异常情况。

因此，为了解决这些问题，就需要发展中断的概念。

6.4.1.2 中断的概念

中断是指 CPU 在执行主程序的过程中，由于计算机系统的外部某种原因，或计算机的一些异常事故及其他的内部原因，有必要尽快终止主程序的执行，转而为该外部事件或内部原因服

务，执行相应的处理程序，待处理完毕后，又返回继续执行被终止的主程序。

事件发生时向 CPU 发出的请求，称为中断请求。能接收中断请求并对中断事件进行处理的部件，称作中断系统。中断系统的功能越来越强，是计算机系统中必不可少的部件。中断源是指能够向 CPU 发出中断请求的事件。通常中断源有以下几种：

(1)一般的 I/O 设备，如键盘、打印机等；

(2)数据通道，如磁盘、光盘等；

(3)需要经确定时间而采用的外部实时时钟；

(4)类似电源掉电等故障源；

(5)为了调试程序而设置的单步调试断点。

6.4.1.3　中断与调用子程序的区别

从表面上看，中断类似于程序设计中的子程序调用，但是它们之间有着本质的区别：

(1)子程序的调用是程序员事先安排好的，而中断是随机产生的；

(2)子程序的执行受到主程序或上层子程序的控制，而中断服务程序一般与被中断的现行程序无关；

(3)不存在同时调用多个子程序的情况，而有可能发生多个外设同时请求 CPU 为自己服务的情况。

总之，中断的处理比子程序的调用复杂得多。

6.4.2　中断系统

6.4.2.1　中断系统的功能

中断系统的功能可以概括为以下几个方面：

(1)及时发现和处理计算机中的软硬件故障；

(2)计算机系统中主机与输入、输出设备之间可采用中断方式相互交换信息，以完成输入输出功能；

(3)应用在实时控制系统中的计算机，可由中断系统来接收从外部实时输入的信息，并进行必要的处理；

(4)多用户计算机系统中，可通过中断系统实现多道程序之间的调度以实现多道程序的并行执行；

(5)计算机系统中的虚拟存储器可用中断方式向 CPU 报告存储空间的使用情况；

(6)计算机软件的调试与维护过程中，可利用中断系统提供人工干预的途径，以实现人机会话功能。

6.4.2.3　中断系统的分类

中断的分类方法很多，根据不同的标准有不同的分类方法。

(1)根据中断源或中断产生原因的不同可分为内中断与外中断。

内中断是指在 CPU 内部产生的中断请求，或者是中断指令 INT n，又称之为软件中断。外中断是指由 CPU 外部引起的中断，包括外部设备引起的中断、定时器/计数器引起的中断等，又

称之为硬件中断。在 8086 系统中，常见的中断源如图 6.13 所示：

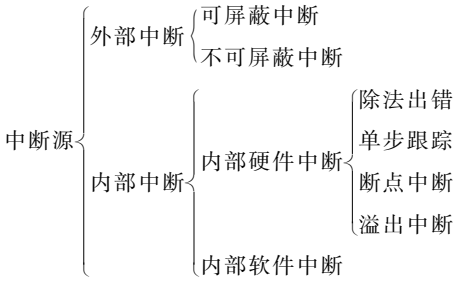


图 6.13　常见的中断源

出现在 8086CPU 可屏蔽中断请求信号线 INTR 上的中断请求信号为可屏蔽中断，它可以被程序用指令禁止或开放，即 CPU 是否响应这些中断取决于标志位 I 的状态，若 I＝1（开中断），CPU 就可以响应可屏蔽中断；若 I＝0（关中断），CPU 就不响应中断，一般的输入、输出设备中断可以安排为可屏蔽中断。中断屏蔽用来对各级中断实现有选择的屏蔽（例如将低级中断和同级中断屏蔽掉，允许高级中断嵌套进入等），在硬件排队方式中，一般用中断屏蔽寄存器实现，也可用中断屏蔽码表示，当中断屏蔽码为“1”时表示该中断源被禁止，当中断屏蔽码为“0”时表示该中断源开放。

出现在 8086CPU 不可屏蔽中断请求信号线 NMI 上的中断称之为不可屏蔽中断，它不受中断标志位 I 的影响，在当前指令执行完后，CPU 就响应。这种中断总是用于某些非常紧急的事件，例如电源掉电、存储错误、总线错误等。

当 CPU 执行除法指令时，若除数为 0 或者商超出了寄存器所能表达的范围，则产生除法错中断。

若单步跟踪标志 T＝1，CPU 进入单步跟踪状态，在每条指令执行完了之后，引起一个单步跟踪中断，直到 T＝0 为止。CPU 响应单步中断之后，先将标志寄存器压栈保护，然后将标志位 T 和 I 清零，以禁止可屏蔽中断和单步中断，再将代码段寄存器 CS 和指令指针 IP 压栈保护，接下来执行中断处理程序，程序执行完后，单步中断结束，将 CS 和 IP 出栈恢复，并恢复标志寄存器的值。

断点中断主要用于软件调试中，程序员可以根据需要在关键的地方设置断点指令，当程序执行到该处时，就会停下来，转去执行相应的中断处理程序。

若在进行带符号的二进制数加、减法指令时，指令产生的结果溢出，使溢出标志 O＝1，便产生一个溢出中断，并转去执行相应的中断处理程序。

内部软件中断是在指令中直接给出中断类型编码产生的中断，也是最常用的中断方式。

(2)根据中断源的性能可分为故障中断和正常中断。

故障中断是由于硬件或软件故障而产生的中断，例如，前面所讲的电源掉电、除法错以及某些内部软件中断。

正常中断是正常情况下产生的中断，例如由键盘等外设发出的中断或者是在程序中设置的中断。

(3)根据中断的进入方式又分为自愿中断和强迫中断。

自愿中断又称程序自中断，它不是随机产生的，而是在程序中安排的有关指令，这些指令可以使计算机进入中断处理的过程，如指令系统中的软件中断指令等。

强迫中断是随机产生的中断，不是程序事先安排好的。当这种中断产生后，由中断系统强迫计算机终止现行程序并转入中断服务程序。

(4)根据处理中断的繁简程度可分为简单中断和程序中断。

程序中断就是前面所提到的中断，主机在响应中断请求后，通过执行一段中断服务程序来处理更紧迫的任务，这样的中断处理过程需要占用一定的 CPU 时间。

简单中断就是主存与外设之间直接进行信息交换的方法，即 DMA 方式。这种中断不去执行中断服务程序，故不破坏现行程序的状态。主机发现有简单中断请求时，让出一个或几个主存的存取周期供外设与主存交换信息，然后继续执行程序。

(5)根据中断源的可屏蔽与否可分为可屏蔽中断和不可屏蔽中断。

6.4.2.4　中断类型与中断向量表

8086 系统中，根据中断源的不同，为每种中断源指定了一个编号，这就是中断类型码，例如前面提到的几种中断源的中断类型码，如表 6.4 所示。

表 6.4　　几种中断类型码

中断源	中断类型码
除法出错	0
单步中断	1
非屏蔽中断	2
断点中断	3
溢出中断	4

中断类型共有 256 种，有些是系统统一规定的(例如表 6.4 中所示)，有些可以由用户指定中断类型(例如多数的内部软件中断)。除了上面 5 种专用的中断类型之外，中断类型码 5～31 是系统的保留部分，留待今后开发系统使用，程序员最好不要使用，中断类型 32～255 供程序员使用。

每种不同的中断源都有一个对应的中断处理程序，也称为中断服务程序，每个中断服务程序都有一个确定的入口地址，这称作中断向量。8086CPU 在内存的最小的 1K 字节(地址为 00000H～003FFH)空间中开辟了一个区域，用来存放这些中断向量，这就是中断向量表，中断向量表确定了中断类型号与中断源所对应的中断向量之间的关系，如图 6.14 所示。

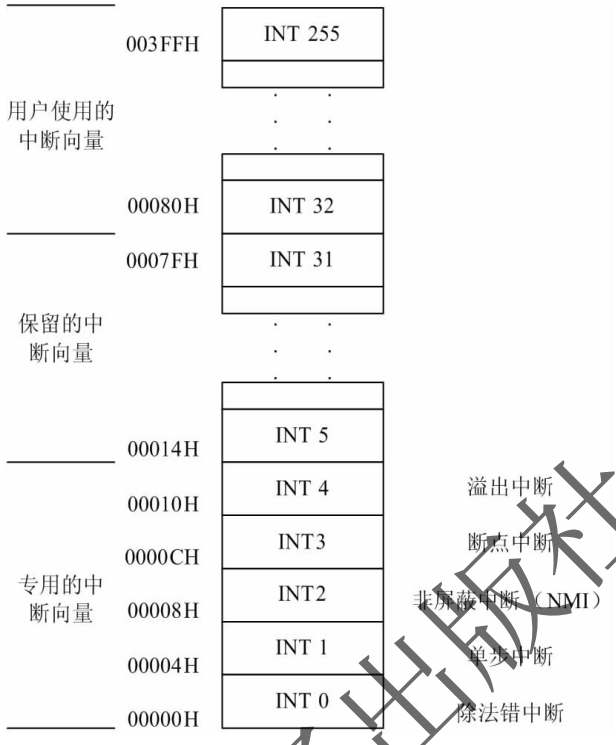


图 6.14　中断向量表

由图 6.14 可以看出，每个中断向量占用 4 个字节，两个低位字节存放中断处理程序入口地址的段内偏移量，两个高位字节存放中断处理程序入口地址的代码段基址。CPU 响应中断后，把表中对应 4 个字节的内容分别送入 CS 和 IP，便可以进入相应的中断处理程序执行。

中断类型与中断向量之间有着确定的关系，它们之间的关系是：中断类型号×4＝中断向量在中断向量表中的首地址，其中段内偏移量的地址单元是（中断类型号×4），代码段基址所在单元地址是（中断类型号×4＋2）。例如，已知软件中断类型号为 12，则存放段内偏移量的地址单元是 12×4＝48＝00030H，存放代码段基地址的地址单元是 00032H，如图 6.15 所示，即中断处理程序的入口地址 CS:IP 为 926B:A533。

00034H	80	FD	CE	12
00030H	33	A5	6B	92

图 6.15　中断向量实例

6.4.2.5　中断系统的分级

（1）中断优先级。计算机可以响应外部和内部多种中断源发出的中断，当多个中断源同时向 CPU 请求中断时，一般要按照中断的优先级进行排队后再处理。

确定中断优先级的原则是：对那些提出中断请求后需要立即处理，否则就会造成严重后果的中断源，规定最高的优先级；而对那些可以延迟响应和处理的中断源，规定较低的优先级。例如 8086 系统中的优先级的次序如图 6.16 所示。

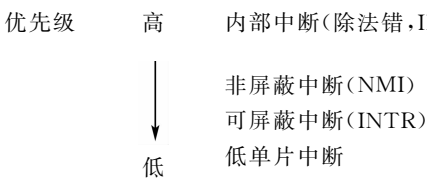


图 6.16　8086 系统的中断优先级

(2)中断排队的处理原则。

① 如果不同优先级别的中断请求同时达到,那么 CPU 应先响应和处理高级中断,后响应和处理低级中断。

② 如果不同优先级的中断请求嵌套产生,可称为多重中断,那么应先允许高级中断打断低级中断而被优先处理,禁止低级中断打断高级中断,也禁止打断同级中断。

6.4.2.5　中断系统举例

Intel 8259A 是可编程中断控制器,它可以管理 8 级硬件中断,若使用 9 片 8259A 组成级联方式,可以实现 64 级的优先权中断管理,它适用于 8080/8085 和 8086/8088 系列微机。

8259A 的内部结构图如图 6.17 所示。

① 数据总线缓冲器。由一个双向 8 位缓冲器构成,是 8259A 与 CPU 之间的数据接口,进行与 CPU 之间的信息交换。

② 读/写逻辑控制电路。用来接收来自 CPU 的读/写逻辑控制命令和片选、选端口控制信息。由于一片 8259A 只占两个 I/O 端口地址,只需要用一位地址 A_0 来选端口,其他高位地址经译码后作为片选信号 $\overline{CS}$ 输入。当 CPU 执行 OUT 命令时,$\overline{WR}$ 信号有效与 A_0 配合,将 CPU 通过数据总线 $D_7 \sim D_0$ 送来的控制字写入 8259A 中的控制寄存器。当 CPU 执行 IN 指令时,$\overline{RD}$ 信号有效与 A_0 配合,将 8259A 中内部寄存器的内容通过数据总线 $D_7 \sim D_0$ 传送给 CPU。

③ 级联缓冲器/比较器。一片 8259A 只能接收 8 级中断 $IR_0 \sim IR_7$,当输入的中断超过 8 级时,可用多片 8259A 级联使用,构成主从关系。对于主 8259A,级联信号 $CAS_0 \sim CAS_2$ 是输出信号;对于从 8259A,$CAS_0 \sim CAS_2$ 是输入信号。$\overline{SP}/\overline{EN}$ 是一个双功能信号,当 8259A 处于缓冲状态,$\overline{EN}$ 有效,表示允许 8259A 通过缓冲器输出;$\overline{EN}$ 无效时,表示允许 CPU 写 8259A。当 8259A 处于非缓冲状态时,$\overline{SP}$ 用作表明主从关系,$\overline{SP}=1$ 表示是主 8259A,$\overline{SP}=0$ 表示是从 8259A。

④ 中断请求寄存器 IRR。这个 8 位寄存器用来存放由外部输入的中断请求信号 $IR_0 \sim IR_7$,当某一个 IR_i 端呈现高电平时,该寄存器相应位置“1”。当 8 个中断信号同时进入时,IRR 寄存器将被置成全“1”。

⑤ 中断服务寄存器 ISR。这个 8 位寄存器用来记录正在处理的中断请求。任何一级中断被响应时,ISR 寄存器中相应位置“1”,一直到该中断处理完毕。中断嵌套情况下,允许 ISR 寄存器中有多位被同时置“1”。

⑥ 中断屏蔽寄存器 IMR。这个 8 位寄存器用来存放 8 级中断请求的屏蔽信号,当禁止某

一级中断进入系统，该寄存器相应位置置“1”，这样可以很方便地实现对各级中断有选择地屏蔽，也方便改变各级中断级别。

⑦ 优先权电路。优先权电路用来识别各中断请求信号的优先级别，当有多个中断请求同时发生时，由它进行中断排队，判定优先级最高的中断，由 CPU 来响应该中断。当有中断嵌套时，由它判定是否允许打断当前中断。

⑧ 控制电路。8259A 的内部控制器，根据中断请求寄存器 IRR 的置位情况和优先权电路判定的结果，向 8259A 内部各部件发出各种控制信号，并向 CPU 发出中断请求和接收来自 CPU 的中断响应信号。

8259A 是可编程的中断控制器，功能比较强大，被广泛应用于微机系统中。

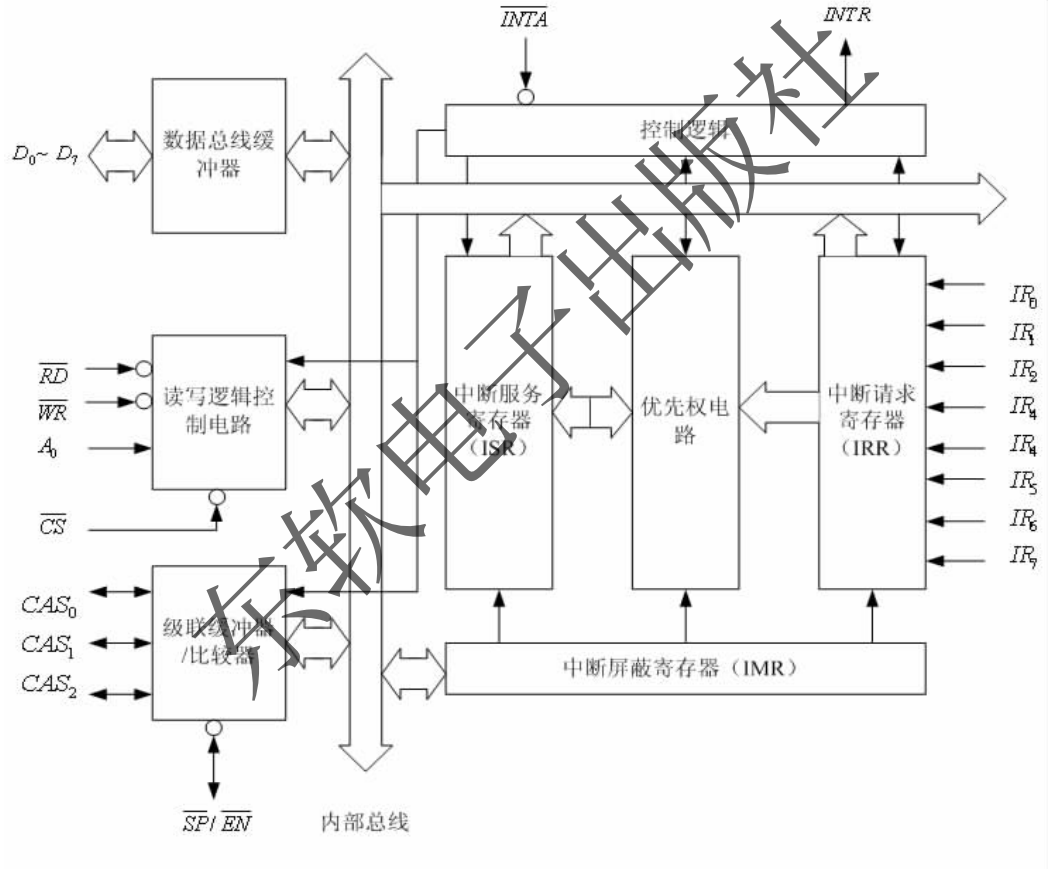


图 6.17 8259A 内部结构图

8259A 的引脚图如图 6.18 所示，引脚名称如表 6.5 所示。

$IR_0 \sim IR_7$ 是 8 条外部中断请求输入信号线。

$D_7 \sim D_0$ 是双向三态数据线，可以直接与系统的数据总线相连。

$\overline{CS}$，$\overline{RD}$，$\overline{WR}$ 和 A_0 一起控制着 CPU 对 8259A 的读/写操作，使 CPU 能向 8259A 写入初始化命令字 ICW 和工作命令字 OCW，或读入 8259A 的内部信息。它们的控制作用如表 6.6

所示。当 $\overline{CS}=1$ 或 $\overline{RD}$，$\overline{WR}$ 同时为 1 时，$D_7 \sim D_0$ 呈现高阻态。

$CAS_0 \sim CAS_2$ 为双向级联地址线，当 8259A 做主片时，为输出线；当 8259A 做从片时，为输入线。

$\overline{SP}/\overline{EN}$ 做输入线使用时，低电平表明该片是从片；作输出线使用时，用于启动 8259A 与 CPU 之间的数据总线缓冲器。该引脚与 $CAS_0 \sim CAS_2$ 配合使用，可实现 8259A 的级联。

INT 是送至 CPU 或主 8259A 的中断请求信号。

$\overline{INTA}$ 是来自 CPU 的中断响应信号。

A_0 通常与系统地址总线 A_0 相连，$\overline{CS}$ 经译码器与系统地址总线相连，$\overline{RD}$，$\overline{WR}$，INT 和 $\overline{INTA}$ 与相应的系统控制总线相连。

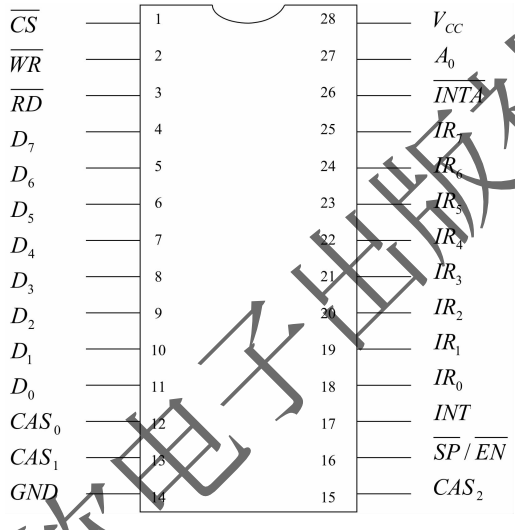


图 6.18　8259A 引脚图

表 6.5　　8259A 引脚名称

引脚号	引脚	名称
4～11	$D_7 \sim D_0$	数据总线(双向)
3	$\overline{RD}$	读允许
2	$\overline{WR}$	写允许
27	A_0	命令选择地址
1	$\overline{CS}$	片选信号
12,13,15	$CAS_0 \sim CAS_2$	级联线
16	$\overline{SP}/\overline{EN}$	从程序/允许缓冲
17	INT	中断请求
26	$\overline{INTA}$	中断响应
18～25	$IR_0 \sim IR_7$	中断请求输入

表 6.6　　8259A 的读写功能 *

$\overline{CS}$	$\overline{WR}$	$\overline{RD}$	$\overline{A_0}$	$\overline{D_4}$	$\overline{D_3}$	
0	0	1	0	0	0	数据总线→OCW_2
0	0	1	0	0	1	数据总线→OCW_3
0	0	1	0	1	X	数据总线→ICW_1
0	0	1	1	X	X	数据总线→OCW_1、ICW_2、ICW_3、ICW_4
0	1	0	0			IRR、ISR 或中断级别码→数据总线
0	1	0	1			IMR→数据总线

* 详见编程说明

6.4.3 中断的嵌套

6.4.3.1 中断嵌套

中断嵌套也称为多重中断，其过程如图 6.19 所示，中断嵌套的层次可以有多层，越在里层的中断越急迫，优先级越高，越能优先得到 CPU 的服务。

CPU 进行中断嵌套时，要按照先发生的中断请求的断点，先保护后恢复，后发生的中断请求的断点，后保护先恢复的原则保护多个断点。同时，为了保证优先级别高的中断源首先得到 CPU 的服务，需要用软件或者硬件实现中断判优，并且在 CPU 进入某一中断服务程序之后，系统必须处于开中断状态。

6.4.3.2 中断现场的保护和恢复

中断现场指的是发生中断时 CPU 的主要状态，其中最重要的是断点，另外还有一些通用寄存器的状态。对中断现场进行保护和恢复主要是因为 CPU 要先后执行两个完全不同的程序（现行程序和中断服务程序），必须进行两种程序运行状态的转换。

现场的保护与恢复方法有纯软件以及软、硬件相结合两种。纯软件方法是 CPU 响应中断后，用一系列传送指令把要保存的现场参数送到主存某些单元中去，当中断服务程序结束后，再采用传送指令进行相反方向的传送。这种方法不需要硬件支持，但是占用了 CPU 的宝贵时间，速度较慢。现代计算机一般都采用硬件方法来自动快速地保护和恢复部分重要的现场，其余寄存器的状态再由软件完成恢复和保护，这种方法需要堆栈做硬件支持。

软硬件结合的方法往往是和响应中断结合在一起使用的。先把断点和程序状态字自动压入堆栈，这就是保护旧现场；接着根据中断源送来的中断向量自动取出中断服务程序入口地址和新的程序状态字，这就是建立新现场；最后由一些指令实现对必要的通用寄存器的保护。恢复现场则是保护现场的逆过程。

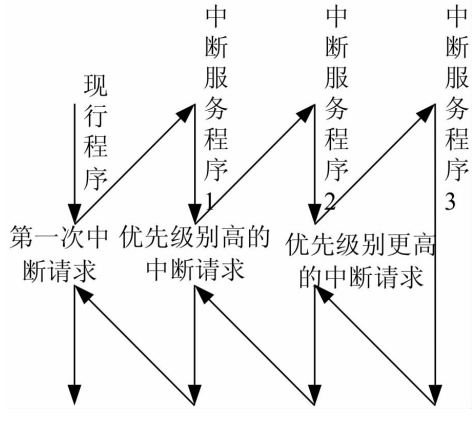


图 6.19　中断嵌套的过程

6.4.4 中断的处理过程

一次完整的中断过程通常分为中断请求、中

断排队、中断响应、中断处理和中断返回五个步骤。

中断请求是由中断源通过中断请求触发器发出的，这样中断源就能够保持中断信号，当中断源发出中断请求时，中断触发器置“1”。

中断排队是在同时有多个中断源发出中断请求时，按照中断源的优先级安排处理的先后顺序，CPU 首先响应优先级高的中断请求，当中断处理完毕后再处理优先级低的中断请求。8086CPU 对中断源优先级的判别是由 8259A 的优先权判别来确定的。中断排队的原则是：

(1)如果不同优先级别的中断请求同时达到，那么 CPU 应先响应和处理高级中断，后响应和处理低级中断。

(2)如果不同优先级的中断请求嵌套产生，可称为多重中断，那么应先允许高级中断打断低级中断而被优先处理，禁止低级中断打断高级中断，也禁止打断同级中断。

中断响应和中断处理比较复杂，下面将详细介绍。

中断返回是使用中断返回指令，将保存的 CS 和 IP 值弹回原处，继续执行主程序。

6.4.4.1　中断的响应

(1)CPU 响应中断的条件。CPU 是否满足中断源的中断请求，必须满足以下几个条件。

① 当前指令执行完毕。

② CPU 处于开中断状态。

③ 当前没有发生复位(RESET)、保持(HOLD)和非屏蔽请求(NMI)。

④ 若当前指令是开中断指令(STI)、中断返回指令(IRET)或前缀指令(LOCK、REP 等)时，执行完毕后，再执行一条指令后 CPU 才能响应中断。

(2)CPU 对中断的响应过程。当满足上述条件时，CPU 响应中断，分为以下几个过程：

① 发出中断响应信号。

② 关中断，将内部中断允许触发器 I 置为 0，以禁止响应其他中断。

③ 保护断点，将断点处的 CS 和 IP 压入堆栈保护。

④ 给出中断服务程序的入口地址，转入执行相应的中断服务程序。

6.4.4.2　中断的处理

中断的处理包括以下几个过程：

(1)保护现场。中断现场指的是发生中断时 CPU 的主要状态，其中最重要的是断点，另外还有一些通用寄存器的状态。现场的保护可以使用纯软件或者软件和硬件结合的方法。一般情况下，主程序的断点程序地址会自动压栈保护，而在中断处理程序中用到的所有寄存器都需要在中断处理程序最开始用硬件或者指令(软件)进行保护。

(2)用软件完成中断处理程序，实现中断的功能。

(3)恢复现场。恢复现场的过程与保护现场的过程相反，按照先入后出的原则，将保护的断点和通用寄存器的状态从堆栈中弹出恢复，主程序的断点地址自动进行恢复。

(4)开中断，将内部中断允许触发器 I 置为 1 或 0，以允许或禁止响应其他中断。

6.4.4.3　一次中断的全过程

一次中断的全过程如图 6.20 所示。

如果用 8086 汇编语言实现一个中断处理程序，程序的结构如下所示：

```
INTERRUP PROC FAR;中断处理程序开始,8086 响应中断时,自动将 F 进栈,并将 IF 置 0
PUSH   AX   ;保护现场
PUSH   BX
```

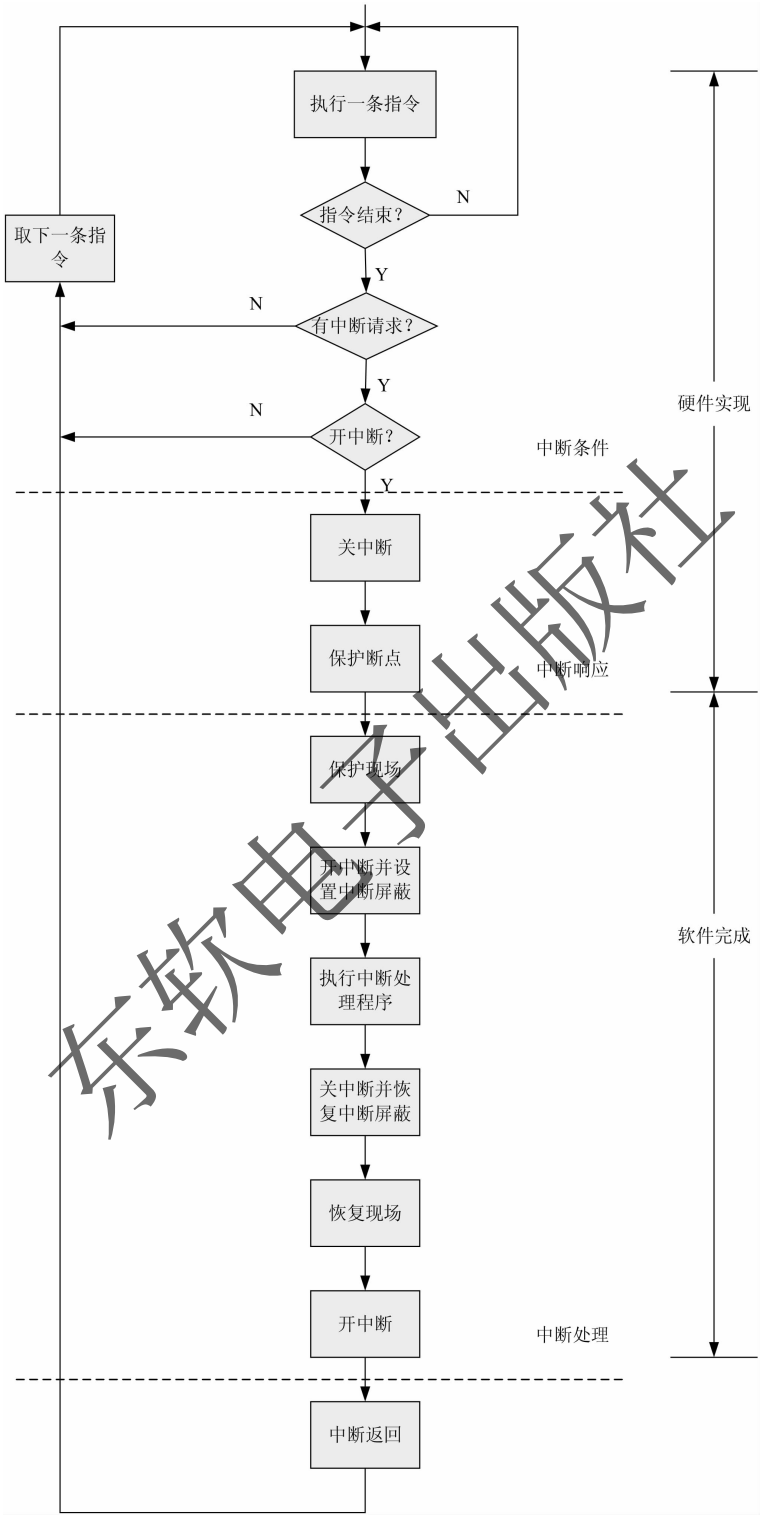


图 6.20 一次中断的全过程

```
PUSH   CX
PUSH   DX
PUSH   SI
PUSH   DI
PUSH   BP
PUSH   DS
PUSH   ES
STI              ;允许多重中断

……               ;中断服务实现

CLI;关中断,恢复现场期间不允许响应中断
POP    ES
POP    DS
POP    BP
POP    DI
POP    SI
POP    DX
POP    CX
POP    BX
POP    AX
IRET       ;中断返回,IRET 指令将自动保存的 F 出栈,并恢复响应中断之前程序的状态
INTERRUP ENDP;中断处理程序结束
```


6.5　CPU 的发展

6.5.1　指令流水线

标准的冯・诺依曼体系结构,采用的是串行处理,即一个时刻只能进行一个操作。而提高计算机性能的根本方向之一是并行处理,因此,近年来人们谋求对传统冯・诺依曼机进行改造,如:

① 采用多个处理部件形成流水处理,依靠时间上的重叠提高处理效率;

② 组成阵列机结构,形成单指令流多数据流,提高处理速度;采用多个冯・诺依曼机组成多机系统,支持并行算法结构等。

由此可见,并行处理技术已成为计算机发展的主流。

并行性有两种含义:

① 同时性:两个以上事件在同一时刻发生。如多机系统中,同一时刻多个进程在运行。

② 并发性:两个以上事件在同一间隔内发生。如并发程序,某一时刻 CPU 中只有一个进程在运行,而在一个时间段内,多个进程同时运行。

一般来说,并行性有以下三种表现形式:

① 时间并行:指时间重叠。实现形式即流水线方式;

② 空间并行:指资源重复。实现形式主要为多处理器系统和多计算机系统。

为了提高计算机访问存储器和执行指令的处理速度,可以使一些需要计算机处理的多项操作在时间上重叠进行,甚至设计出多个功能相同或相近的部件同时对其进行处理,这种技术就是流水线与并行执行技术。

(1)指令流水原理。指令流水类似于工厂的装配线,装配线上利用每个装配段同时可以对不同产品进行加工的特点提高装配效率。如果可以将这种思想用到指令的执行上,也可以提高指令的执行效率,这就是指令流水的概念。

从本章的分析可以看到,完成一条指令的过程实际上也可以分为许多阶段。例如,微处理器中一条指令的执行过程一般可以分为这样几个步骤:先是处理器(CPU)从主存取来指令,接着对此指令进行译码以确定它的操作类型。然后若操作需要存储器操作数时,要确定操作数地址并负责由主存取来操作数。最后完成所需的操作。并将结果写回寄存器或主存的指定位置中。在不采用流水技术的计算机中,这个过程的是串行执行的,如图 6.21 所示。

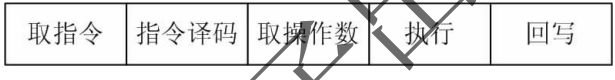


图 6.21 指令的串行执行

这样的执行过程虽然也能完成指令,但执行部件的利用率不高,实际上在指令执行的过程中,不需要访问主存,这时完全可以利用这个时间段来取下一条指令,这时有两条指令重叠执行,这就是流水。

(2)流水线作业。图 6.22 是指令的二级流水线,在这里,指令执行的时间一般要比取指令的时间要长,所以这种方式虽然可以加快指令的执行,但还是不能将执行效率加倍。

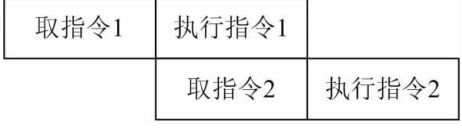


图 6.22 指令的二级流水线

为了进一步提高流水线的处理速度,可以将指令的处理过程再进一步细化,例如,一条指令流水线可由如下 5 个阶段组成:

S1——取指令(IF),由主存取指令;

S2——指令译码(ID),确定指令将要完成的操作;

S3——取操作数(OF),确定存储器操作数地址,取得所需存储器操作数和寄存器操作数;

S4——执行(EX),对操作数完成指定操作;

S5——回写(WB),修改目标操作数。

采用流水线完成 4 条指令的过程如图 6.23 所示。

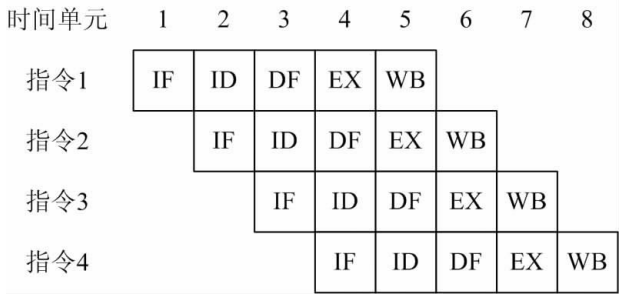


图 6.23　指令流水线

可以看出，完成 4 条指令的执行只用了 8 个时钟周期，若以非流水线的顺序处理则需要 20 个时钟周期。

(3)举例：8088 微处理器的流水线。① 8088 微处理器中，取指令和指令的执行分别由总线接口部件 BIU 和执行部件 EU 独立完成。

② 这两个阶段的执行过程是并行操作的，即在一条指令 i(n)的执行过程中，可以取出下一条指令 i($n+1$)在指令流队列中排队，在指令 i(n)执行完以后就可以立即执行下一条指令 i($n+1$)。

进入流水线的指令流，由于后一指令的第 i 功能步与前一指令的第 i+1 功能步同时进行，从而使一串指令总的完成时间大为缩短。

6.5.2　流水线性能

用于衡量流水线的性能指标主要是吞吐率、加速比和效率。

(1)吞吐率。吞吐率是指单位时间内能处理的指令条数或能输出的结果量。吞吐率越高，计算机系统的处理能力就越强。就流水线而言，吞吐率就是单位时间内能流出的任务数或能流出的结果数。最大吞吐率是指流水线达到稳定状态后可获得的吞吐率。如果流水线中各个子过程所需要的时间都是△t，在流水线正常满负荷工作时，就会每隔△t 时间解释完一条指令，即其最大吞吐率 TPmax=1/△t。而实际上由于各个子过程所完成的工作都不同，所以各自的时间也不同。为了避免各段之间经过的时间不匹配，也为了保证各段之间数据通路宽度上能匹配，通常都要在各子过程之间插入一个锁存器，这些锁存器都受同一个时钟脉冲同步，这样每个子过程所经过的时间也应将锁存器所需要的延迟包括进去。时钟脉冲的周期对流水线的最大吞吐率会有直接影响，其值越小越好。如果各个子过程所需的时间分别为△t1，△t2，△t3，△t4，时钟周期应当为 max{△t1，△t2，△t3，△t4}，则最大吞吐率可用下面公式表示：TPmax=1/max{△t1，△t2，△t3，△t4}。

实际吞吐率就是指流水线完成 n 条指令的实际吞吐率。假设一指令流水线由 m 段组成，且各段经过时间都为△t0，若连续处理 n 条指令，则第一条指令从流入到流出需要 T0=m△t0 的流水建立时间，之后每隔△t0 就可以流出一条指令。因此，完成 n 条指令的解释共需时间 T=m·△t0+(n−1)△t0，在这段时间里，流水线的实际吞吐率为：

$$TP=\frac{n}{m\Delta t_0+(n+1)\Delta t_0}=\frac{1}{\Delta t_0(1+\frac{m-1}{n})}=\frac{TP_{max}}{1+\frac{m-1}{n}}$$

可以看出，不仅实际的吞吐率总是小于最大的吞吐率，而且只有当 n>>m 时，实际的吞吐率才能接近于理想的最大吞吐率。

(2)加速比。加速比是指采用流水线方式后的工作速度与等效的顺序串行方式的工作速度之比。对 n 个求解任务而言，若用串行方式工作需要时间为 T_1，而用 m 段流水线来完成此项工作需要时间为 T_k，则加速比为：

$$S_p=\frac{T_1}{T_k}=\frac{nm\Delta t}{m\Delta t+(n-1)\Delta t}=\frac{nm}{m+n-1}=\frac{m}{1+\frac{m-1}{n}}$$

当 n>>m 时，有 Sp≈m，显然要获得高的加速比，流水线段数 m 应尽可能取大，即应加大流水深度。

(3)效率。效率是指流水线中的各功能段的利用率。由于流水线有建立和排空时间，因此各功能段的设备不可能一直处于工作状态，总有一段空闲时间。所以，整个流水线的效率是：

$$\eta=\frac{\eta_1+\eta_2+\cdots+\eta_m}{m}=\frac{m\cdot\eta_0}{m}=\frac{m\cdot n\Delta t_0}{m\cdot T}$$

其中，分母 m·T 是时空图中 m 各段以及流水总时间 T 所围成的总面积，分子 m·n△t0 则是时空图中 n 个任务实际占用的总面积。因此，从时空图上来看，所谓效率实际就是 n 个任务占用的时空区和 m 个段总的时空区面积之比。显然，只有当 n>>m 时，η 才趋近于 1。同时还可看出，对于线性流水线，每段经过时间相等的情况下，流水线的效率将与吞吐率成正比，即

$$\eta=\frac{n\cdot\Delta t_0}{T}=\frac{n}{n+m-1}=TP\cdot\Delta t_0$$

应当说明的是，对于非线性流水线或线性流水线中各段经过的时间不等时，这种成正比的关系并不存在，此时应通过画出实际工作时的时空图来分别求出吞吐率和效率。

例 6.5：设有四段流水线，△t1＝△t2＝△t3＝△t4＝△t，求有 4 个指令时的吞吐率和加速比。

解：(1)图 6-24 所示为 4 条指令进入流水线的时空图，在 7 个时钟周期可以完成 4 条指令，所以实际吞吐量为 4/(△t·7)。

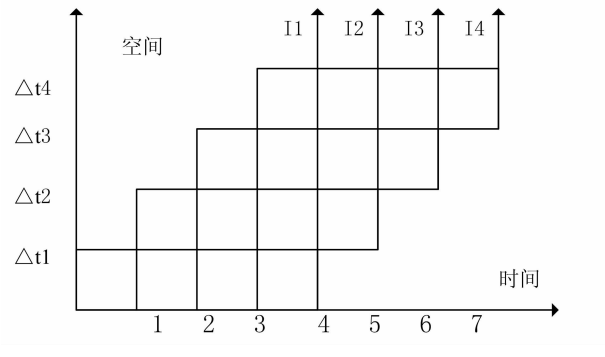


图 6.24 标准流水线时空图

(2)加速比为 4 段流水线的速度/等功能的非流水线的速度＝7△t/(4×4△t)＝7/16。

6.5.3　流水线中的多发技术

流水线技术使计算机的体系结构产生了巨大的变化。为了取得进一步的发展,还可以在流水线中加入多发技术,最常见的多发技术有超标量技术、超流水技术、超长指令字技术等。

(1)超标量技术。这种技术是指在每个时钟周期内可同时并发多条独立指令,也就是将多条指令并行编译并执行。这种技术要求 CPU 内部有多个功能部件和指令译码器,还要有多个寄存器端口和总线。

(2)超流水线技术。这种技术是将一线流水寄存器插入到流水线段中,这样可以将原来的一个时钟周期又再一次分段,这样超流水线的处理器的周期就要比普通流水线的处理器周期短,这样,在原有的时钟周期内,功能部件被使用次数增加,速度也就加快了。

(3)超长指令字技术。超常指令字是由美国 Yale 大学教授 Fisher 提出的。它有点类似于超级标量,是一条指令来实现多个操作的并行执行,之所以放到一条指令是为了减少内存访问。通常一条指令多达上百位,有若干操作数,每条指令可以做不同的几种运算。那些指令可以并行执行是由编译器来选择的。

6.5.4　CPU 的发展和典型 CPU

6.5.4.1　CPU 的发展

个人电脑从 8088(XT)发展到现在的 PentiumIII 时代,只经过了不到 20 年的时间。从生产技术来说,最初的 8088 集成了 29000 个晶体管,而高能奔腾的集成度超过了 750 万个晶体管;从 CPU 的运行速度,以 MIPS(百万个指令每秒)为单位,8088 是 0.75,而速度最快的高能奔腾超过了 1000。

(1)8088 和 8086。Intel 公司于 1981 年推出 8086 与 8088 微处理器,著名的 IBMXT 电脑就是基于 8088。这两种 16 位的微处理器比以往的 8 位机功能更强大,地址线有 20 条,内存寻址范围为 1M 字节。它们的区别在于,8086 外部的数据也是 16 位,而 8088 的外部数据为 8 位。

(2)80286。1982 年,INTEL 推出了 80286 芯片,该芯片含有 13.4 万个晶体管,80286 也是 16 位处理器,其频率比 8086 更高,它有 24 条地址线,内存寻址范围是 16M 字节。

(3)80386。80386 属于 32 位微处理器,其内部和外部数据总线都是 32 位,地址总线也是 32 位,可寻址 4GB 内存。它除具有实模式和保护模式外,还增加了一种叫虚拟 86 的工作方式,可以通过同时模拟多个 8086 处理器来提供多任务能力。它有以下几种:80386SX,它是准 32 位处理器,数据总线是 16 位,其内部 32 位寄存器必须分两个 16 位的总线来读取。它是 286 计算机与 386DX 计算机之间的过渡产品。386DX 是真正的 32 位处理器,它的数据总线和内部寄存器都是 32 位。它还可以配上 80387 数字协处理器,以提高计算速度。386 处理器的主频有 16,20,25,33,40MHz 五种。除 Intel 公司生产 386 芯片外,还有 AMD,Cyrix,Ti 和 IBM 等公司生产的。

(4)80486。80486 简称 486,于 1989 年由 Intel 公司首先推出,集成了 120 万个晶体管。其

时钟频率从25MHz逐步提高到33MHz,50MHz。它也属于32位处理器。80486是将80386和数字协处理器80387以及一个8KB的高速缓存集成在一个芯片内,并且在80X86系列中首次采用了RISC技术,可以在一个时钟周期内执行一条指令。它还采用了突发总线方式,大大提高了CPU与内存的数据交换速度。

(5)Pentium处理器。Pentium(奔腾)是Intel公司于1993年推出的新一代微处理器,它集成了310万个晶体管。Pentium微处理器使用更高的时钟频率,最初为60MHZ和66MHZ,后提高到200MHZ。64位数据总线,16KB的高速缓存。

6.5.4.2 Pentium微处理器

随着超大规模集成电路技术飞速发展,以及在图形图像处理、语音识别、视频处理、CAD/CAE/CAI及网络软件开发中对高速度、大内存、大流量客户机/服务器的迫切需求,Intel公司于1993年推出了新一代微处理器——Pentium。Pentium微处理器是一种高性能32位微处理器,最初的一块Pentium芯片的功能大体相当于两块80486。近十年来,Intel公司又陆续推出了速度更快,性能更高的Pentium的第2代(PII)、第3代(PIII)产品和第4代(P4)产品。目前时钟频率高达1500MHz的P4已成为当前主流PC机的通用芯片。

(1)Pentium特征。Pentium在80486的基础上做了很大改进,不仅增加了片内集成度,而且采用了新的体系结构,其性能更高。其主要特点是:

- 高集成度,片内集成有310万个晶体管。
- 时钟频率高,早期的Pentium为60MHz或66MHz。目前已发展到500MHz,700MHz和1500MHz。
- 采用超标量流水线结构,比相同频率的80486 DX性能提高一倍。
- 数据总线带宽增加。内部总线为32位,外部数据总线宽度为64位。
- 片内采用分立的指令Cache和数据Cache结构,可无冲突地同时完成指令预取和数据读写。
- 采用新型的分页模式。
- 固化常用指令,使指令的运行得到进一步加快。
- 具有分支指令预测功能。采用RISC技术。
- 重新设计的高性能浮点运算部件。
- 在数据的完整性、容错性和节电性等方面采用了新的设计方法。软件向上兼容80386/80486,可以在MS-DOS, Windows 95, Windows NT, OS/2,UNIX和Solaris等操作系统下运行。

(2)Pentium的内部结构。Pentium CPU的结构如图6.25所示,与80486相比,它采用了新的体系结构。

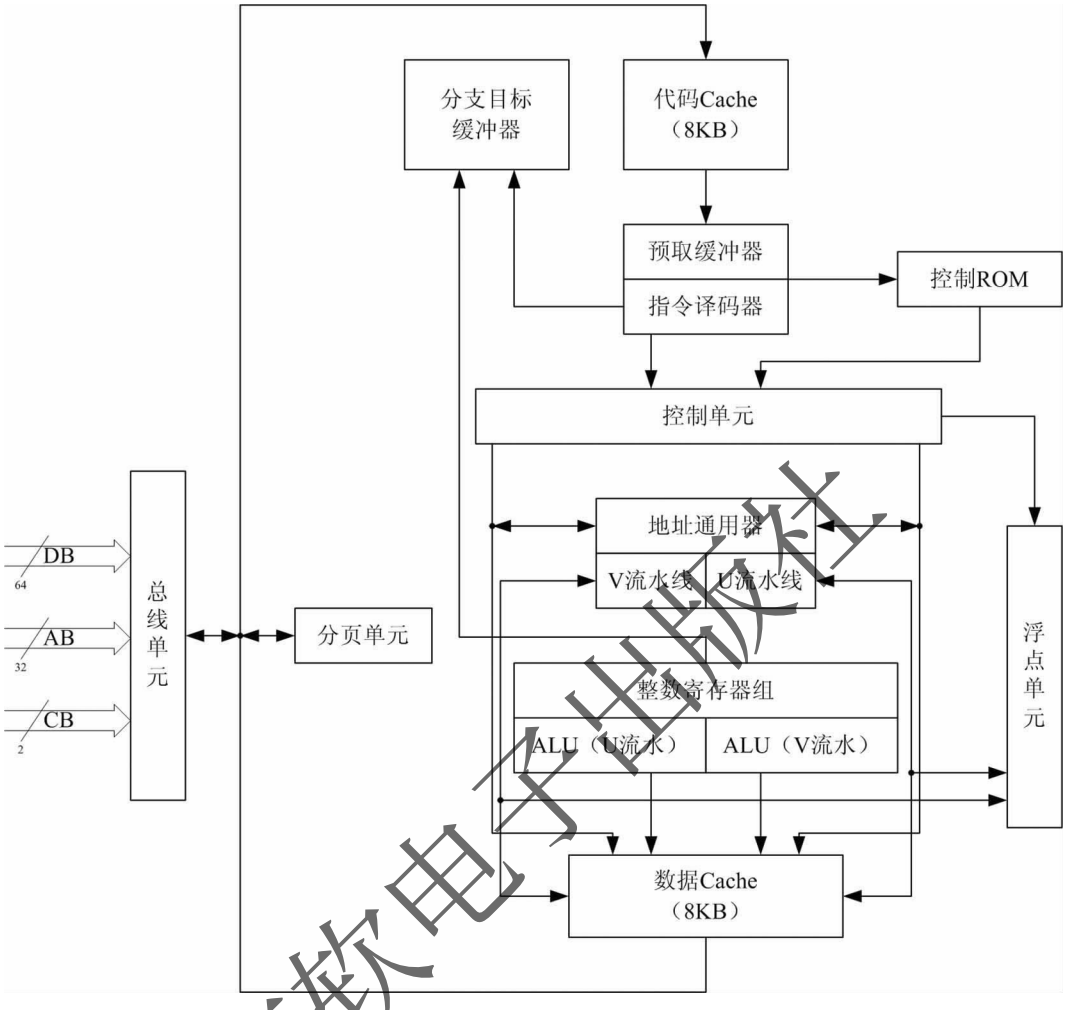


图 6.25　Pentium CPU 的结构

① 超标量流水线。超标量流水线是 Pentium 系统结构的核心，采用 U 和 V 两条并行的指令流水线的超标量流水线(Super－Scalar)设计。每条流水线都拥有自己的算术逻辑单元、地址生成电路和数据 Cache 接口，使 Pentium 在单个时钟周期内执行两条简单指令，比相同频率的 486DX CPU 性能提高一倍。

② 分离的指令 Cache 和数据 Cache。Pentium 内部有两个独立的 8KB 的超高速缓冲存储器，即 8KB 的代码 Cache 和 8KB 的数据 Cache。其中一个只保存指令，另一个只保存数据。分立的指令和数据 Cache 是对 Pentium 超标量结构的有力支持。它不仅使指令预取和数据读写能无冲突地同时完成，而且可同时与 U，V 两条流水线分别交换数据，使 Pentium 的性能大大超过了 80486 微处理器。

③ 新型的分页模式。Pentium 外部地址总线宽度是 36 位，但一般使用 32 位宽，故物理地址空间为 4096MB(4GB)，虚拟地址空间为 64TB。分页模式除与 80486 相同并支持 4KB 页面之外，还支持 2MB 和 4MB 页面。其中，2MB 页面的分页模式必须使用 36 位地址总线。

④ 数据总线带宽增加。Pentium 内部的主要寄存器是 32 位宽的，它是一个 32 位微处理器。但它通向存储器的外部数据总线宽度为 64 位，总线每次可同时传输 8 个字节。因此，在一个总线周期内将数据传输量增加了一倍。

⑤ 高性能的浮点运算部件。Pentium 指令系统对微码进行了重新设计，对其微代码的算法也做了重大改进，所需时钟周期大为减少。尤其是它的片内浮点运算器在 80486 的基础上进行了改进，采用 8 级流水，一个时钟周期内可完成一个甚至两个浮点操作。对于常用的浮点加法、乘法和浮点数装入操作，速度比 80486 快 10 倍，大大减少了指令执行时所需的时钟周期。

⑥ 固化常用指令。在 Pentium 处理器中，常用指令如 MOV，ALU，INC，DEC，PUSH，POP，JMP，CALL(NEAR)，SHIFT，TEST，ROT，NOT 和 NOP 等改用硬件实现，不再使用微码操作，使指令的运行得到进一步加快。

⑦ 采用动态转移预测技术。Pentium 采用动态转移预测技术，以减少由于过程相关性引起的流水线性能损失。它提供的转移目标缓冲器 BTB 是个小容量的 Cache。当一条指令导致程序转移时，BTB 记住这条指令及其转移目标地址。以后遇到这条转移指令时，BTB 会依据前面转移发生的历史来预测这次是转移取还是顺序取。若预测为转移取则将记录的转移目标地址立即送出。

Pentium 的超标量流水线有两个指令预取缓冲器，当前总使用其中一个。当在指令译码(DI)段译出一条转移指令时立即检索 BTB。若预测为顺序取则继续以当前预取缓冲器取指令。若预测为转移取则立即冻结当前预取缓冲器，启动另一个预取缓冲器，从给出的转移目标地址处开始取分支程序的指令序列，这就保证了流水线的指令预取步骤永远不会空置。而且当预测转移取不正确时，正确路径的指令已在指令预取缓冲器中，因而减小了因预测错误而蒙受的性能损失。除了上述特点外，Pentium 在数据完整性、容错性和节电等方面也采取了一些特殊的设计方法。

6.6 相关知识介绍

6.6.1 CPU 性能参数指标

Pentium 采用动态转移预测技术，以减少由于过程相关性引起的流水线性能损失。它提供的转移目标缓冲器 BTB 是个小容量的 Cache。当一条指令导致程序转移时，BTB 记忆住这条指令及其转移目标地址。以后遇到这条转移指令时，BTB 会依据前面转移发生的历史来预测这次是转移取还是顺序取。若预测为转移取则将记录的转移目标地址立即送出。

(1)主频、外频、倍频及三者间的关系

主频＝外频×倍频

主频是 CPU 的标称运行频率，外频是系统总线的工作频率，倍频是外频与主频相差的倍数。

(2)缓存与 CPU 性能的关系

①Cache(缓存)分为一级缓存 L1 Cache 和二级缓存 L2 Cache。CPU 硬件由 CPU 内核、

L1、L2 两级高速缓存以及相应的辅助电路组成。

②CPU 内核的工作速度(即主频)与两级缓存的容量和速度是影响 CPU 性能的重要因素,理论上 L2 Cache 时钟频率越高,容量越大,CPU 的性能越高。

(3)前端总线频率与 CPU 性能的关系

①CPU 与北桥芯片间的通道被称为前端总线。

②其工作频率(即前端总线频率,或称为 FSB)越高,则数据传输率越高。

(4)核心工作电压与 CPU 性能的关系

①CPU 的核心电压通常在 1.5V 至 1.75V 间。

②提高核心电压可以提高 CPU 工作频率,其实质是通过提升核心电压来提高 CPU 的信号强度,这是超频时常采用的一种办法,但这样会使 CPU 发热量增加,因此一般超频时主要考虑超外频或倍频。

(5)CPU 的制造工艺与 CPU 性能的关系

①CPU 的制造工艺实际是指 CPU 在加工中核心电路的线宽,通常采用微米来度量。

②目前主流 CPU 的制造工艺已达到了 0.09μm,线宽越小,则可以提高集成度,使 CPU 的体积更小,耗电更少,性能更高。

6.6.2　CPU 产品标识

在 CPU 的宣传和测试软件中,常常可以看到 0686h 的字样,它代表 CPU 的 ID(identify,鉴别号码),通过这些数字,可以看出 CPU 的各种参数。例:0686h 可以划分为 0、6、8、6,其含义各是 CPU 类型、系列、型号、步进,那么 0680h = OEM 产品、P6 系列、coppermine 内核,cC0 步进。

(1)CPUID

CPU Type(类型)

“类型”标识英特尔微处理器是用于由用户(最终用户)安装,还是由专业个人计算机系统集成商(OEM)、服务公司或制造商安装。“1”标识微处 理器是用于由用户安装的(例如英特尔 OverDrive(R)处理器之类的升级)。“0”标识微处理器是用于由专业个人计算机系统集成商、服务公司或制造 商安装的。处理器类型还表示出该 CPU 可配置成单处理器、双处理器或是英特尔 OverDrive 处理器系统。

CPU Type 含义

0 桌面系统、OEM

1 移动式系统、升级芯片

Family(系列)

此分类标识英特尔微处理器的品牌以及属于第几代产品。例如,当今的 P6 系列(第六代)英特尔微处理器包括英特尔赛扬(TM)、奔腾(R) II、奔腾 II Xeon(TM)、奔腾 III 和奔腾 III Xeon 处理器。5 系列(第五代)包括奔腾处理器和采用 MMX(TM)技术的奔腾处理器。当前的英特尔奔腾 4 处理器的系列值为“F”(十六进制)。

CPU Family 含义

1 8086 和 80186 级芯片

2 286 级芯片

3 386 级芯片

4 486 级芯片(SX、DX、DX2、DX4)

5 P5 级芯片(经典奔腾和多能奔腾)

6 P6 级芯片(包括赛扬、奔腾 2/3、高能奔腾)

F 奔腾 4

Model(型号)

“型号”编号可以让英特尔识别微处理器的制造技术以及属于第几代设计(例如型号 4)。型号与系列通常是相互配合使用的,用以确定您的计算机中所安装的处理器是属于处理器系列中的哪一种特定类型。在与英特尔联系时,此信息通常用以识别特定的处理器。

CPU Model(P6 级)含义

1 Pentium Pro(高能奔腾)

2 Pentium Pro(高能奔腾)

3 Klamath(奔腾 2)

4 Deschutes(奔腾 2)

5 Covington(赛扬)

6 Mendocino(赛扬 A)

7 Katmai(奔腾 3)

8 Coppermine(奔腾 3)

Stepping ID(步进)

也叫分级鉴别产品数据转换规范,“步进”编号标识生产英特尔微处理器的设计或制造版本数据(例如步进 4)。步进用于标识一次“修订”。通过使用 唯一的步进,可以有效地控制和跟踪所做的更改。步进还可以让最终用户更具体地识别其系统所安装的处理器版本。在尝试确定微处理器的内部设计或制造特性时,英特尔可能会需要使用此分类数据。

KatmaiStepping 含义

2 kB0 步进

3 kC0 步进

CoppermineStepping 含义

1 cA2 步进

3 cB0 步进

6 cC0 步进

(2)高速缓存信息

高速缓存(Cache)是一种用于储存频繁使用的指令和数据的速度非常快的存储器。高速缓存信息可能会包含二级高速缓存大小以及一级数据和指令 高速缓存大小,缓存容量自然是越大越好。要注意的是,某些处理器二级缓存容量虽大,却以 CPU 内核一半或更低的速度运行,

性能不及全速的二级缓存。

英特尔(R)奔腾(R) 4 处理器一级高速缓存由两部分组成,一部分是储存数据字节的数据高速缓存,另一部分是储存解码的指令的执行跟踪高速缓存。执行跟踪高速缓存的大小以微操作数(micro－ops 或 mops)计量。

为用于 MP(多处理器)而设计的英特尔(R) Xeon(TM)处理器包含一个 3 级高速缓存。与 2 级高速缓存相比,此高速缓存提供更大的指令和数据高速存储空间,而且操作性能比主存储器更高。

(3)封装

此外,包装盒上面也有显示装有处理器的物理包装类型。可能的包装类型有:

S. E. C. C. (Single Edge Contact Cartridge)/S. E. C. C. 2－单边接触插盒是一种矩形塑料包装盒,通过边缘指状联接安装于系统主板。典型的 S. E. C. C. /S. E. C. C. 2 插盒有一个矩形的风扇和/或热散热装置,装在插盒的一侧。

S. E. P. P. (Single Edge Processor Package)－这是一种单边处理器包装,通过边缘指状联接安装于系统主板。典型的 S. E. P. P. 包装有一个矩形的风扇和/或热散热装置装在其一侧。

FC－PGA(Flip－Chip Pin Grid Array)－翻转芯片插针网阵包装的外观是一个绿色材料制成的薄方片,其一侧有金针阵列伸出。这些插针插入系统主板上的一个插座。FC－PGA 处理器常 常因被装在其上面的方型风扇和/或热散热装置挡住而不被注意到。FC－PGA 内核的保护层很脆弱,过于用力的散热器安装方式会导致其损坏。

PPGA(Plastic Pin Grid Array)－塑料插针网阵包装的外观是一个塑料材料制成的薄方片,其一侧有金针阵列伸出。这些插针插入系统主板上的一个插座。PPGA 处理器常常因被装在其上面的风扇和/或热散热装置挡住而不被注意到。

MMC2－这是一个专门用于移动式处理器的移动模块包装。

MM－这是一个专门用于移动式处理器的移动模块包装。

uPGA(micor Pin Grid Array)/BGA(Ball Grid Array)－这是一个用于移动式处理器的微针网阵或球状网阵。

OLGA(Organic Land Grid Array)－用于移动式处理器的有机平面网阵包。

OOI－用于间插包上的 OLGA。间插包将 OLGA 包的精细间距垫转换为一个针场。

上述方法适用于下列 CPU:

英特尔奔腾(R)处理器

采用 MMX(TM)技术的英特尔奔腾(R)处理器

英特尔奔腾(R) OverDrive(R)处理器

采用 MMX(TM)技术的英特尔奔腾(R) OverDrive(R)处理器

英特尔奔腾(R) Pro 处理器

英特尔奔腾(R) II OverDrive(R) for 奔腾(R) Pro 处理器

英特尔奔腾(R) II 处理器

英特尔奔腾(R) II Xeon(TM)处理器

英特尔赛扬(TM)处理器

英特尔奔腾(R) III 处理器
英特尔奔腾(R) III Xeon(TM)处理器
英特尔奔腾(R) 4 处理器
英特尔(R) Xeon(TM)处理器
移动式英特尔奔腾(R)处理器
移动式英特尔奔腾(R) II 处理器
移动式英特尔奔腾(R) III 处理器
移动式英特尔赛扬(TM)处理器

6.6.3 多核技术

多内核是指在一枚处理器中集成两个或多个完整的计算引擎(内核)。多核技术的开发源于工程师们认识到,仅提高单核芯片的速度会产生过多热量且无法带来相应的性能改善,先前的处理器产品就是如此。他们认识到,在先前产品中以那种速率,处理器产生的热量很快会超过太阳表面。即便是没有热量问题,其性价比也令人难以接受,速度稍快的处理器价格要高很多。

现在的多核 CPU 主要分原生多核和封装多核。原生多核指的是真正意义上的多核,最早是 AMD 公司提出的,每个核心之间都是完全独立的,都拥有自己的前端总线,不会造成冲突,即使在高负载状况下,每个核心都能保证自己的性能不受太大的影响,通俗的说,原生多核的抗压能力强,但是需要先进的工艺,每扩展一个核心都需要很多的研发时间。

封装多核是只把多个核心直接封装在一起,比如 Intel 早期的 PD 双核系列,就是把两个单核直接封装在一起,但两核心只能共同拥有一条前端总线,在两个核心满载时,两个核心会争抢前端总线,导致性能大幅度下降,所以早期的 PD 被扣上了"高频低能"的帽子,要提高封装多核的性能,在多任务的高压下尽量减少性能损失,只能不断的扩大前端总线的总体大小,来弥补多核心争抢资源带来的性能损失,但这样做只能在一定程度上弥补性能的不足,和原生的比起来还是差了很多,而且成本比较高,但是优点在于多核心的发展要比原生快的多。

6.6.4 CPU6 个主要寄存器

在 CPU 中至少要有六类寄存器。这些寄存器用来暂存一个计算机字。根据需要,可以扩充其数目。下面详细介绍这些寄存器的功能与结构。

(1)数据缓冲寄存器(MDR)

数据缓冲寄存器用来暂时存放由内存储器读出的一条指令或一个数据字;反之,当向内存存入一条指令或一个数据字时,也暂时将它们存放在数据缓冲寄存器中。

缓冲寄存器的作用是 :

可以作为 CPU 和内存、外部设备之间信息传送的中转站;

可以补偿 CPU 和内存、外围设备之间在操作速度上的差别;

在单累加器结构的运算器中,数据缓冲寄存器还可兼作为操作数寄存器。

(2)指令寄存器(IR)

指令寄存器用来保存当前正在执行的一条指令。当执行一条指令时,先把它从内存取到缓冲寄存器中,然后再传送至指令寄存器。指令划分为操作码和地址码字段,由二进制数字组成。为了执行任何给定的指令,必须对操作码进行测试,以便识别所要求的操作。指令译码器就是做这项工作的。指令寄存器中操作码字段的输出就是指令译码器的输入。操作码一经译码后,即可向操作控制器发出具体操作的特定信号。

(3)程序计数器(PC)

为了保证程序能够连续地执行下去,CPU 必须具有某些手段来确定下一条指令的地址。而程序计数器正是起到这种作用,所以通常又称为指令计数器。在程序开始执行前,必须将它的起始地址,即程序的一条指令所在的内存单元地址送入 PC,因此 PC 的内容即是从内存提取的第一条指令的地址。当执行指令 时,CPU 将自动修改 PC 的内容,以便使其保持的总是将要执行的下一条指令的地址。由于大多数指令都是按顺序来执行的,所以修改的过程通常只是简单的对 PC 加 1。

但是,当遇到转移指令如 JMP 指令时,那么后继指令的地址(即 PC 的内容)必须从指令的地址段取得。在这种情况下,下一条从内存取出的指令将由转移指令来规定,而不是像通常一样按顺序来取得。因此程序计数器的结构应当是具有寄存信息和计数两种功能的结构。

(4)地址寄存器(MDR)

地址寄存器用来保存当前 CPU 所访问的内存单元的地址。由于在内存和 CPU 之间存在着操作速度上的差别,所以必须使用地址寄存器来保持地址信息,直到内存的读/写操作完成为止 。

当 CPU 和内存进行信息交换,即 CPU 向内存存/取数据时,或者 CPU 从内存中读出指令时,都要使用地址寄存器和数据缓冲寄存器。同样,如果我们把外围设备的设备地址作为像内存的地址单元那样来看待,那么,当 CPU 和外围设备交换信息时,我们同样使用 地址寄存器和数据缓冲寄存器。

地址寄存器的结构和数据缓冲寄存器、指令寄存器一样,通常使用单纯的寄存器结构。信息的存入一般采用电位—脉冲方式,即电位输入端对应数据信息位,脉冲输入端对应控制信号,在控制信号作用下,瞬时地将信息打入寄存器。

(5)累加寄存器(AC)

累加寄存器 AC 通常简称为累加器,它是一个通用寄存器。其功能是:当运算器的算术逻辑单元 ALU 执行算术或逻辑运算时,为 ALU 提供一个工作区。累加寄存器暂时存放 ALU 运算的结果信息。显然,运算器中至少要有一个累加寄存器。

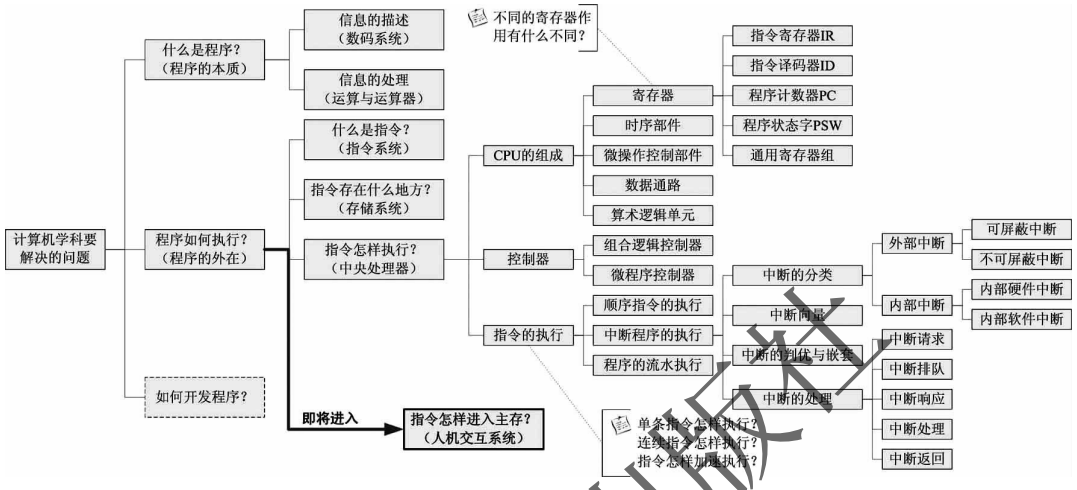
目前 CPU 中的累加寄存器,多达 16 个,32 个,甚至更多。当使用多个累加器时,就变成通用寄存器堆结构,其中任何一个可存放源操作数,也可存放结果操作数。在这种情况下,需要在指令格式中对寄存器号加以编址。

(6)状态条件寄存器(PSW)

状态条件寄存器保存由算术指令和逻辑指令运行或测试的结果建立的各种条件码内容,如运算结果进位标志(C),运算结果溢出标志(V),运算结果为零标 志(Z),运算结果为负标志(N)

等等。这些标志位通常分别由 1 位触发器保存。

除此之外,状态条件寄存器还保存中断和系统工作状态等信息,以便使 CPU 和系统能及时了解机器运行状态和程序运行状态。因此,状态条件寄存器是一个由各种状态条件标志拼凑而成的寄存器。



课程逻辑图

实践环节设计

项目 12:算术运算及对标志位的影响(UP(6/4))

一、能力要求

(1)计算机基础知识:掌握关于指令代码、机器代码、指令周期、机器周期、时钟周期等基础知识。(重要)

(2)系统的显现和交互作用:理解 CPU 各部件协同工作时的时序关系。(重要)

(3)分析问题:分析执行算术运算指令的具体过程。(重要)

(4)估计与定性分析:分析一条指令在不同时钟周期的具体执行。(中等)

(5)查询印刷资料和电子文献:查询指令机器码及时序图。(重要)

(6)书面交流能力:规范地撰写项目报告。(重要)

二、项目构思

(1)掌握时钟周期、机器周期与指令周期的关系;

(2)掌握标志位的作用;

(3)掌握机器指令的编写与执行过程;

(4)掌握算术运算指令的执行过程。

三、项目设计

使用虚拟实验仪,将操作数放在 2000H 和 2001H 两个单元中,编程进行以下的算术运算,并记录标志位的状态。

(1)41H＋3BH,结果放在 2002H 单元中;
(2)41H＋5AH,结果放在 2003H 单元中;
(3)AFH＋7EH,结果放在 2004H 单元中;
(4)E3H＋1DH,结果放在 2005H 单元中;
(5)41H－3BH,结果放在 2006H 单元中;
(6)3BH－41H,结果放在 2007H 单元中;
(7)2FH－2FH,结果放在 2008H 单元中。

四、项目实施

(1)写出指令代码段,例如 41H＋3BH:

```
MOV A，2000
ADD A，2001
MOV 2002，A
```

(2)写出指令的机器代码,例如上面指令的机器代码为:

A0　00　20　04　0120　A2　02　20

(3)在虚拟实验仪上输入指令的机器代码。

五、项目运行

(1)运行每一段指令,按表 6.7 格式在项目报告中记录结果。

表 6.7　　项目运行结果

题目	指令	机器代码	结果存放内存单元	运算结果	S 标志位	C 标志位	Z 标志位
41H＋3BH							
41H＋5AH							
AFH＋7EH							
E3H＋1DH							
41H－3BH							
3BH－41H							
2FH－2FH							

(2)回答以下问题:

项目实施(2)中所示的机器代码中,哪些是操作数,哪些是操作码?

操作码是在指令周期的第几个机器周期的第几个时钟周期的时钟的上升沿还是下降沿被送入指令寄存器的?

六、目标检验

学生自查:按照项目设计推导运行结果,并和项目实际运行结果比较,验证结果是否正确。

教师检查:按照标准答案以及学生撰写项目报告情况给出项目成绩。

项目考试:本项目是项目考试内容之一。

项目 13:逻辑运算及对标志位的影响(UP(6/4))

一、能力要求

(1)计算机基础知识:掌握关于指令代码、机器代码、指令周期、机器周期、时钟周期等基础

知识。(重要)

(2)系统的显现和交互作用:理解 CPU 各部件协同工作时的时序关系。(重要)

(3)分析问题:分析执行逻辑运算指令的具体过程。(重要)

(4)估计与定性分析:分析一条指令在不同时钟周期的具体执行。(中等)

(5)查询印刷资料和电子文献:查询指令机器码及时序图。(重要)

(6)书面交流能力:规范地撰写项目报告。(重要)

二、项目构思

掌握逻辑运算指令的执行过程。

掌握标志位的作用。

三、项目设计

使用虚拟实验仪,将操作数放在 2000H 和 2001H 两个单元中,编程进行以下的逻辑运算,并记录标志位的状态。

(1)! 00H,结果放在 2002H 单元中;

(2)! FFH,结果放在 2003H 单元中;

(3)77H&44H,结果放在 2004H 单元中;

(4)5AH&A5H,结果放在 2005H 单元中;

(5)A3H&81H,结果放在 2006H 单元中;

(6)73H|4AH,结果放在 2007H 单元中;

(7)DFH|39H,结果放在 2008H 单元中。

四、项目实施

(1)写出指令代码段;

(2)写出指令的机器代码;

(3)在虚拟实验仪上输入指令的机器代码。

五、项目运行

(1)运行每一段指令,按表 6.8 格式在项目报告中记录结果。

表 6.8　　项目运行结果

题目	指令	机器代码	结果存放内存单元	运算结果	S 标志位	C 标志位	Z 标志位
! 00H							
! FFH							
77H&44H							
5AH&A5H							
A3H&81H							
73H|4AH							
DFH|39H							

(2)回答问题:与、或、非指令的操作码分别在指令执行的第几个机器周期的第几个时钟周期的上升沿还是下降沿被送入指令寄存器 IR?

六、目标检验

学生自查:按照项目设计推导运行结果,并和项目实际运行结果比较,验证结果是否正确。

教师检查：按照标准答案以及学生撰写项目报告情况给出项目成绩。

项目考试：本项目是项目考试内容之一。

项目 14*：算术逻辑综合运算及对标志位的影响(UP(6/4))

一、能力要求

(1)计算机基础知识：掌握关于指令代码、机器代码、指令周期、机器周期、时钟周期等基础知识。(重要)

(2)系统的显现和交互作用：理解 CPU 各部件协同工作时的时序关系。(重要)

(3)解决问题时的妥协、判断和平衡：平衡程序占用存储空间与执行时间之间的关系。(不重要)

(4)分析问题：分析复杂运算与简单运算之间的关系。(中等)

(5)综合和通用化能力：能够转换算法，使用已有的指令解决复杂运算问题。(中等)

(6)查询印刷资料和电子文献：查询指令机器码及时序图。(重要)

(7)书面交流能力：规范地撰写项目报告。(重要)

二、项目构思

(1)学习算术逻辑综合运算的指令代码的编写；

(2)掌握标志位对运算结果的影响。

三、项目设计

使用虚拟实验仪，编程进行以下的算术逻辑运算，并记录标志位的状态。

(1)1FH＋63H＋59H，结果放在 2000H 单元中；

(2)F5H－3BH－9AH，结果放在 2001H 单元中；

(3)1FFFH＋0001H，结果放在 2002H 和 2003H 两个单元中；

(4)1024H－50AFH，结果放在 2004H 和 2005H 单元中；

(5)5AH⊙00H，结果放在 2006H 单元中；

(6)5AH ⊕ FFH，结果放在 2007H 单元中。

四、项目实施

(1)写出指令代码段；

(2)写出指令的机器代码；

(3)在虚拟实验仪上输入指令的机器代码。

五、项目运行

运行每一段指令，按表 6.9 格式在项目报告中记录结果。

表 6.9　　项目运行结果

题目	指令	机器代码	结果存放内存单元	运算结果	S 标志位	C 标志位	Z 标志位
1FH＋63H＋59H							
F5H－3BH－9AH							
1FFFH＋0001H							
1024H－50AFH							
5AH⊙00H							
5AH ⊕ FFH							

学生自查:按照项目设计推导运行结果,并和项目实际运行结果比较,验证结果是否正确。

教师检查:按照标准答案以及学生撰写项目报告情况给出项目成绩。

项目 15:中断系统(一)(UP(6/4))

一、能力要求

(1)计算机基础知识:掌握关于中断的基础知识。(重要)

(2)系统的显现和交互作用:理解 CPU 各部件协同工作时的时序关系。(重要)

(3)验证假设与结论:验证中断处理的过程。(中等)

(4)具有综合和通用化能力:能够自定义中断处理程序。(中等)

(5)发现问题和表述问题:发现中断响应的时刻问题。(重要)

(6)估计与定性分析:估计中断响应的时刻。(中等)

(7)查询印刷资料和电子文献:查询指令机器码。(重要)

(8)书面交流能力:规范地撰写项目报告。(重要)

(9)定义功能,概念和结构:自定义中断功能。(不重要)

二、项目构思

(1)了解中断响应的过程。

(2)掌握中断服务程序的编写方法。

三、项目设计

使用虚拟实验仪,在任意一个主程序的基础上,编写一段中断服务程序,其功能是将累加器 A 中的内容复制到寄存器 B 中。

四、项目实施

(1)写出指令代码段;

(2)写出指令的机器代码;

(3)在虚拟实验仪上 8000H 开始的地址输入指令的机器代码。

五、项目运行

运行程序,观察运行结果,在项目报告中记录程序及指令机器代码,并回答以下问题:

(1)中断服务程序的最后一条指令是什么?

(2)CPU 响应中断需要哪几个过程?

(3)本项目中,收到中断请求后,CPU 是否能够立即响应中断?响应中断需要满足什么样的条件?

六、目标检验

学生自查:按照项目设计推导运行结果,并和项目实际运行结果比较,验证结果是否正确。

教师检查:按照标准答案以及学生撰写项目报告情况给出项目成绩。

项目考试:本项目是项目考试内容之一。

项目 16*:中断系统(二)(UP(6/4))

一、能力要求

(1)计算机基础知识:掌握关于中断的基础知识。(重要)

(2)验证假设与结论:验证中断向量的作用。(中等)

(3)具有综合和通用化能力:能够自定义中断处理程序。(中等)

(4)查询印刷资料和电子文献:查询 8086 中断分类表。(重要)

(5)书面交流能力:规范地撰写项目报告。(重要)

(6)定义功能、概念和结构:自定义中断功能。(不重要)

二、项目构思

(1)掌握中断向量的概念。

(2)掌握中断的处理过程。

三、项目设计

使用 DEBUG 调试程序,用户自定义一段中断程序(例如:在屏幕上显示字符'A'),并用这段中断程序来改写 0 号中断。

四、项目实施

(1)查看当前各个寄存器的值,记录 CS:IP 的值;

(2)验证系统提供的 0 号中断的功能;

(3)写出自定义的中断程序代码;

(4)修改中断向量表中的 0 号中断的地址为上述自定义中断程序的起始地址。

五、项目运行

用 P 命令执行程序,查看运行结果。

六、目标检验

学生自查:按照项目设计推导运行结果,并和项目实际运行结果比较,验证结果是否正确。

教师检查:按照标准答案以及学生撰写项目报告情况给出项目成绩。

项目 17☆:流水结构(UP(6/4))

一、能力要求

(1)计算机基础知识:掌握关于指令流水线的基础知识。(重要)

(2)系统的显现和交互作用:显现指令流水工作时系统的整体协调与相互作用。(中等)

(3)分析问题:分析指令流水时对总线的争用以及时序问题。(中等)

(4)验证假设与结论:验证流水线的作用。(不重要)

(5)查询印刷资料和电子文献:查询指令机器码。(重要)

(6)书面交流能力:规范地撰写项目报告。(重要)

二、项目构思

了解流水线操作的基本概念。

三、项目设计

使用虚拟实验仪完成该项目。

四、项目实施

(1)编写一段简单的程序,例如:

```
MOV A, 2000
ADD A, 2001
MOV 2002, A
```

(2)写出指令的机器代码;

(3)在虚拟实验仪上输入指令的机器代码。

五、项目运行

(1)运行该程序,观察运行过程;

(2)按下"流水"按钮,再次运行该程序,观察运行过程;

(3)回答问题:流水结构是如何提高程序的运行速度的?

六、目标检验

学生自查:按照项目设计推导运行结果,并和项目实际运行结果比较,验证结果是否正确。

教师检查:按照标准答案以及学生撰写项目报告情况给出项目成绩。

项目 18*:虚拟模型机的设计(CP(1))

一、能力要求

(1)计算机基础知识:掌握关于计算机各个部件的基础知识。(重要)

(2)系统的显现和交互作用:显现计算机各部件之间的相互作用。(中等)

(3)解决问题时的妥协、判断和平衡:平衡各部件工作时的时序。(中等)

(4)分析问题:分析指令的执行过程。(重要)

(5)验证假设与结论:验证运算器与控制器的工作原理。(中等)

(6)具有综合和通用化能力:综合运用各种软、硬件知识。(中等)

(7)发现问题和表述问题:发现与表述设计中出现的各种问题。(不重要)

(8)估计与定性分析:分析各部件的工作原理。(重要)

(9)查询印刷资料和电子文献:查询计算机各部件以及所使用到的逻辑部件相关技术文档。(重要)

(10)定义功能、概念和结构:定义计算机各部件的结构与功能。(不重要)

(11)设计过程的分段与方法:分步设计与硬件实现。(中等)

(12)硬件制造过程:了解不同的硬件的制造过程。(不重要)

二、项目构思

完成计算机硬件系统各个部件的设计与实现,并构建出完整的模型计算机系统。

三、项目设计

模型计算机的设计是从规划一个简单而基本完备的指令系统开始的,并用这套指令系统开展少量的汇编程序设计。接着设计计算机系统的初步组成方案,运算器可以设计成定点运算器部件,控制器可以设计成组合逻辑的控制部件,内存储器系统可以设计成静态的存储器部件,使其与设计的 CPU 系统结合在一起,再加入串行接口电路以及对中断处理的支持线路,最后通过简单的总线把所有部件连接在一起,就构成完整的计算机硬件系统。

四、项目实施

实施时可以采用两种方案:

(1)硬件与软件结合的方式。这种方式为传统的实施方式,需要提供仿真的汇编和运行支持,硬件的开发、维护成本较高,且不宜扩展,因此,不建议采用这种方式。

(2)纯软件实施的方式。这种方式可以采用软件编程实现,例如本项目集中的虚拟实验仪,就是采用 Flash 实现的。使用纯软件方式实施,开发和维护费用较低,且容易扩展,因此是一种值得推广的方式。

五、项目运行

在模型计算机硬件搭建完成之后,可以进一步在基本软件系统中,配备简单的监控程序(操作系统雏形),比较规范的汇编器程序,并开发一个仿真终端程序,模拟键盘的输入和显示器的输出,最终就得到了一个硬软件组成的基本完整的模型计算机系统。

六、目标检验

学生自查:按照项目设计推导运行结果,并和项目实际运行结果比较,验证结果是否正确。

教师检查：按照标准答案以及学生撰写项目报告情况给出项目成绩。

教学效果测评

本单元将通过作业、项目报告、项目考试以及期末考试进行效果测评，其中作业如下。

一、单项选择题

1. 指令周期是指_____。

A. CPU 从主存取出一条指令的时间　　B. CPU 执行一条指令的时间

C. 一个时钟周期时间

D. CPU 从主存取出一条指令加上执行这条指令的时间

2. 堆栈指针 SP 的内容是_____。

A. 栈顶单元内容　　B. 栈顶单元地址

C. 栈底单元内容　　D. 栈底单元地址

3. 在计算机系统中，表征系统运行状态的部件是_____。

A. 程序计数器　　B. 累加计数器　　C. 中断计数器　　D. 程序状态字

4. 微程序控制方式的计算机，把所有的微程序集中存放在一个独立的存储器中，通常将其称为_____。

A. 只读存储器　　B. 随机存储器　　C. 控制存储器　　D. 辅助存储器

5. 用于存放现行指令的寄存器称为_____。

A. 指令寄存器　　B. 指令译码器

C. 程序计数器　　D. 程序状态寄存器

6. 8086CPU 中 SP 是_____。

A. 段寄存器　　B. 基址指针寄存器

C. 指令寄存器　　D. 堆栈指针寄存器

7. PSW 是指_____。

A. 指令寄存器　　B. 指令译码器

C. 程序计数器　　D. 程序状态寄存器

8. _____是向主存写入数据或从主存读出数据的缓冲部件。

A. MDR　　B. MAR　　C. PSW　　D. DS

9. 在 CPU 中跟踪指令后继地址的寄存器是_____。

A. 主存地址寄存器　　B. 程序计数器

C. 指令寄存器　　D. 状态条件寄存器

10. 下列_____情况可能会发生中断请求。

A. 一次 I/O 操作结束　　B. 两个 4 位二进制数相加

C. 两个数相或　　D. CPU 读内存储器

11. 中断向量的地址是_____。

A. 子程序入口地址　　B. 中断服务子程序入口地址

C. 中断服务子程序入口地址的地址　　D. 中断返回地址

12. 在计算机中，存放微指令的控制存储器隶属于_____。

A. 外存　　B. 高速缓存　　C. 内存储器　　D. CPU

13. 同步控制是_____。
A. 只适用于 CPU 控制的方式　　B. 只适用于外围设备控制的方式
C. 由统一时序信号控制的方式　　D. 所有指令执行时间都相同的方式
14. 微程序控制器中,机器指令与微指令的关系是_____。
A. 每一条机器指令由一条微指令来执行
B. 每一条机器指令由一段微指令编写的微程序来解释执行
C. 每一条机器指令组成的程序可由一条微指令来执行
D. 一条微指令由若干条机器指令组成
15. 以硬连线方式构成的控制器也称为_____。
A. 组合逻辑型控制器　　B. 微程序控制器
C. 存储逻辑型控制器　　D. 运算器
16. 一个节拍脉冲持续_____的时间。
A. 一个指令周期　　B. 一个时钟周期
C. 一个 CPU 周期　　D. 一个存取周期
17. 中断处理时要求保留 CPU 现场的原因是_____。
A. 为了返回现场　　B. 为了保证原程序完整
C. 为了中断完毕返回原程序执行
D. 为了中断处理时需要用到 CPU 现场数据
18. 中断处理时,保存现场后要开中断是为了_____。
A. 不准再响应新中断,以保存完整现场　　B. 允许响应高级中断请求
C. 中断嵌套　　D. 提高中断处理速度
19. 中断处理时,保存现场前要关中断是为了_____。
A. 不准再响应新中断,以保存完整现场　　B. 允许响应高级中断请求
C. 中断嵌套　　D. 提高中断处理速度
20. 计算机主频周期是指_____。
A. 指令周期　　B. 时钟周期　　C. CPU 周期　　D. 存取周期
21. 计算机的指令部件包括_____。
A. 控制器、运算器　　B. 指令计数器、指令寄存器、指令译码器
C. 地址寄存器、数据寄存器、接口　　D. 控制存储器、地址寄存器、数据缓冲器
22. 控制存储器用来存放_____。
A. 控制程序　　B. 微程序　　C. 汇编语言程序　　D. 高级语言程序
23. 时序电路的作用是_____。
A. 给出各种时间信号　　B. 给出各种控制信号
C. 给出执行指令的地址信号　　D. 给出计算机中各种时间顺序信号
24. CPU 内通用寄存器的位数取决于_____。
A. 存储器容量　　B. 机器字长　　C. 指令的长度　　D. CPU 的管脚数
25. 在 CPU 中,保存当前正在执行的指令的寄存器为①;保存当前正在执行的指令地址(在某些机器中为下一条要执行的指令地址)的寄存器是②;算术逻辑运算的结果通常放在③或④中_____
A. ① 程序计数器②地址寄存器③数据寄存器④通用寄存器
B. ① 指令寄存器②程序计数器③通用寄存器④累加器

C. ① 程序状态字②数据寄存器③通用寄存器④地址寄存器

D. ① 通用寄存器②累加器③程序计数器④通用寄存器

26. 在计算机中存放当前指令地址的寄存器叫①;在顺序执行指令的情况下(存储器按字节编址,指令字长 32 位),每执行一条指令,使寄存器自动加②;在执行③指令或④操作时,⑤应接收新地址。_____

A. ①指令寄存器②2③转移④顺序⑤地址寄存器

B. ①地址寄存器②4③中断④转移⑤程序计数器

C. ①程序计数器②4③转移④中断⑤程序计数器

D. ①程序计数器②1③中断④顺序⑤地址寄存器

27. 假设微处理器的主频为 50MHz,两个时钟周期组成一个机器周期,平均三个机器周期完成一条指令,则它的机器周期为① _____ ns,平均运算速度近似为② _____ MIPS。

① A. 10　　B. 20　　C. 40　　D. 100

② A. 2　　B. 3　　C. 8　　D. 15

28. 关于指令周期,以下叙述正确的是_____。

A. 一个时钟周期由若干个 CPU 周期组成,一个 CPU 周期由若干个指令周期组成

B. 一个 CPU 周期由若干个时钟周期组成,一个时钟周期由若干个指令周期组成

C. 一个 CPU 周期由若干个指令周期组成,一个指令周期由若干个时钟周期组成

D. 一个指令周期由若干个 CPU 周期组成,一个 CPU 周期由若干个时钟周期组成

29. 在使用微程序控制器的计算机系统中,一条机器指令的执行对应着一个_____的执行过程。

A. 微命令　　B. 微指令　　C. 微程序　　D. 微操作

30. 通用寄存器 AX,BX,CX,DX 作为 **16** 位寄存器时可以存放_____,作为 **8** 个 **8** 位寄存器时可以存放_____。

A. 数据,数据和地址　　B. 数据和地址,地址

C. 地址,数据和地址　　D. 数据和地址,数据

二、填空题

1. 在 CPU 中跟踪下一条指令地址的寄存器是_____。

2. SP 为_____。

3. 8088 / 8086CPU 中,_____、_____、_____是 16 位通用寄存器,每个通用寄存器可作两个_____位寄存器使用。

4. 在微程序控制器中,一条机器指令是由若干_____组成的微程序来解释执行的。

5. 微程序顺序控制常用的两种方式是_____方式和_____方式。

6. AX,BX,CX,DX 均为_____位寄存器,也可作为两个_____位寄存器使用。

7. 在控制器中,指令寄存器(IR)的作用是_____。

8. CPU 周期也称为_____,一个 CPU 周期包含若干个_____。

9. 任何一条指令的指令周期至少需要_____个 CPU 周期。

10. 一条机器指令的处理过程,宏观上可分作_____和_____过程。

11. 中央处理器(CPU)的四个主要功能是_____、_____、_____和_____。

12. 控制器最基本的组成部分包括_____、_____和_____。

13. 中央处理器简称_____。

14. 组合逻辑控制器是用_____实现的，优点是_____，缺点是_____，调试、修改和扩充指令困难。

15. 中央处理器包括_____和_____。

16. 实现一条机器指令功能的_____叫微程序。

17. 微程序存放在_____，对其要求是_____。

18. 顺序执行时 PC 的值_____。

19. 在程序循环执行的过程中，控制器控制计算机的运行总是处于_____、分析指令和_____的循环中。

20. 遇到转移和调用指令时，后继指令的地址（即 PC 的内容）是从指令寄存器中的_____取得的。

21. 微程序控制器的核心部件是存储微程序的_____。

22. 在微程序控制中，计算机执行一条指令的过程就是依次执行一个确定的_____的过程。

23. 控制存储器一般由_____构成。

24. 由于数据通路之间的结构关系，微操作可分为_____和_____两种。

25. 任何指令周期的第一步必定是_____周期。

26. CPU 从_____取出一条指令并执行这条指令的时间和称为_____。由于各种指令的操作功能不同，各种指令的指令周期是_____。

27. 控制器在生成各种控制信号时，必须按照一定的_____进行，以便对各种操作实施时间上的控制。

28. 中断周期执行的三件事情_____、_____、_____都是利用_____实现的。

29. 8086 系统中，中断返回时要恢复_____和_____才能返回主程序继续执行。

30. 在程序执行过程中，下一条待执行指令的地址码寄存在_____中。

31. CPU 响应中断时最先完成的两个步骤是_____和_____。

32. 计算机将正在执行的指令存放在_____中。

33. CPU 中，除了操作控制器外，还必须有_____对各种操作实施时间上的控制。

34. 8086CPU 中的寄存器是_____位的。

35. 当 OF＝_____时表示运算有溢出。

36. CPU 对主存储器的两种操作为_____和_____，上述两种操作的信号是通过_____总线发送至主存储器。

37. CPU 与主存储器连接时，采用了三组总线，分别是：_____总线、_____总线和_____总线。

38. 在计算机中存放指令地址的叫 A，在取指令之前，首先把 A 的内容送到 B，然后由 CPU 发出读指令，把指令从 B 所指定的内存单元中取出，送到 CPU 的 C，则 A，B，C 分别是以下选项中的_____、_____和_____。

(1)指令(2)累加器(3)通用寄存器(4)变址寄存器(5)程序计数器(6)状态寄存器(7)内存地址寄存器(8)数据寄存器

39. 计算机执行程序的过程始终遵循着_____、_____和_____的过程。

40. CPU 响应中断后，在执行中断服务程序之前，至少要做_____、_____

和_____。

41. 中断服务程序的最后一条是_____指令。

42. 一个完整的中断处理过程包括下面几个阶段：_____、_____、_____、_____和_____，其中发送向量地址在_____阶段，执行中断服务程序在_____阶段，恢复程序断点和中断前的程序状态字在_____阶段。

三、判断题

1. 节拍产生器是产生控制信号的部件。（　）
2. 流水线操作可以加快程序的执行过程。（　）
3. 程序计数器 PC 的内容自动加 1 是在指令执行完毕后进行。（　）
4. 一个指令周期由若干个机器周期组成。（　）
5. 串行寄存器一般都具有移位功能。（　）
6. 机器的主频越快，机器的速度也就越快。（　）

四、对应题

写出下列中文全称对应的英文缩写或英文缩写对应的中文或英文全称。

1. CPU
2. 电可擦除可编程只读存储器
3. ALU
4. 指令寄存器
5. CB
6. 动态随机读取存储器
7. MDR
8. 指令译码器
9. PSW
10. 程序计数器

五、简答题

1. 请说明指令周期、机器周期、时钟周期之间的关系。
2. 指令和数据均以二进制代码形式放在主存中，请问 CPU 如何区别它们是指令还是数据？
3. 简述 CPU 的主要功能。
4. 比较水平微指令和垂直微指令的优缺点。
5. 举出 CPU 中 6 个主要寄存器的名称及功能。
6. CPU 响应中断应具备哪些条件？
7. 简要说明加法指令 ADD　R0，R1 的执行步骤。
8. 中断与调用子程序有什么区别？
9. 原理性地说明条件相对转移指令的指令格式和执行步骤。
10. 假设一条指令按取指、分析和执行三步解释执行，每步相应的时间分别是 $T_{取}$，$T_{分}$，$T_{执}$，当：

(1) $T_{取}=T_{分}=T_{执}=5ns$ 时，

(2) $T_{取}=5ns$，$T_{分}=6ns$，$T_{执}=7ns$ 时，

A. 分别计算下列几种情况下执行完 100 条指令所需的时间：

(1)顺序方式。

(2)仅(K+1)取指与 K 执行重叠。

(3)仅(K+2)取指、(K+1)分析、K 执行重叠。

B. 计算以上第(3)种方式的装满时间、稳定流水时间和排空时间。

11. 简要说明组合逻辑控制器中的节拍发生器的作用是什么？简述它的运行原理。

12. 中断服务子程序与一般的子程序有什么区别？

13. 何谓中断嵌套？如何实现？中断嵌套要注意哪些问题？

14. 中断向量、中断向量表与中断类型号之间有什么关系？若某中断类型号是 12H，则其中断向量存于何处？

15. 某 8086PC 机中中断向量表的部分内容(均以十六进制表示)如下，请写出该机对应与中断类型 0,5,8,20H 的服务程序入口地址。

```
0000:0000    E8 4E 2E 01 F0 01 70 00
0000:0008    5F F8 00 F0 F0 01 80 00
0000:0010    30 50 00 60 01 30 90 20
0000:0018    40 A5 23 FF 87 E9 00 20
0000:0020    A1 90 60 26 70 90 35 AB
……
0000:0080    C3 2E 12 01 E4 12 2E 01
0000:0088    42 02 34 09 70 02 34 09
```

16. 有 4 个中断源 D_1,D_2,D_3,D_4，它们的中断优先级和中断屏蔽码见表 6.10，表中，“1”表示该中断源被屏蔽，“0”表示该中断源开放。假设从处理机响应中断源的中断服务请求到运行中断服务程序中第一次开中断所用的时间为 1 微秒，其他中断服务时间为 10 微秒。

(1)处理机在 0 时刻开始响应中断请求，这时 4 个中断源都已经申请中断服务，写出处理机开始响应各中断源的中断请求和处理机为各中断源完成中断服务的时刻。

(2)处理机在 0 时刻开始响应中断请求，这时中断源 D3 和 D4 已经申请中断服务，在 6 微秒时中断源 D_1 和 D_2 同时申请中断服务，写出处理机开始响应各中断源的中断请求和处理机为各中断源完成中断服务的时刻。

表 6.10　　中断源的中断优先级与中断屏蔽码

中断源	中断优先级	中断屏蔽码($D_1D_2D_3D_4$)
D_1	1(最高)	1100
D_2	2(第二)	0101
D_3	3(第三)	1010
D_4	4(最低)	1011