

目 录

第一章	绪论.....	1
第二章	指令系统.....	6
第三章	运算方法和运算器.....	15
第四章	控制器.....	50
第五章	存储器.....	55
第六章	输入输出控制.....	79
第七章	计算机模块结构与互连.....	94

第一章 绪论

例 1.1 解释下列术语:硬件,软件,固件,CPU,主机,主频,运算速度,指令。

解 硬件:组成计算机的电子元器件和机电装置的统称。

软件:是计算机程序及运用数据处理系统所必需的手续、规则和文件的总称。软件由程序和文档组成。

固件:将程序固化在 ROM 中组成的部件。固件是一种具有软件特性的硬件,既具有

硬件的快速性,又具有软件的灵活性。

CPU:即中央处理器(Central Processing Unit),包括运算器和控制器两部分。由于运算器、控制器在逻辑关系上联系紧密,将两者统称为 CPU。

主机:由 CPU、内存储器及相应接口组成。

主频:时钟 CP(Clock Pulse)的工作频率。

运算速度:是计算机工作能力和工作效率的主要表征,它取决于在给定的时间内处理器所能处理的数据量以及处理器的时钟频率。

指令:是计算机设计者赋予计算机实现某种基本操作的命令,由操作码和地址码组成。本书中的指令即指机器指令。第二章定义机器指令为计算机设计者赋予计算机实现某种基本操作的命令。

例 1.2 什么是存储程序原理?按此原理,计算机应具备哪些功能?

解 存储程序原理的基本思想是:计算机要自动完成解题任务,必须将事先设计好、用以描述计算机解题过程的程序(程序即指令的有序集合),和数据一样,采用二进制形式存储在机器内部,计算机在工作时自动高速地从机器中逐条取出指令并加以执行。

按照存储程序控制原理,计算机必须具有五大功能:(1)数据传送功能;(2)数据存储功能;(3)数据处理功能;(4)操作控制功能;(5)操作判断功能。

例 1.3 存储程序计算机包含哪几个部分?它们的功能各是什么?

解 存储程序计算机包含:运算器、控制器、存储器、输入设备、输出设备及将它们连结为有机整体的互连结构。它们的功能分别是:

- 运算器:进行数据处理,即执行算术运算和逻辑运算。
- 控制器:计算机的管理机构和指挥中心,它协调计算机的各部件自动地工作。
- 存储器:存放数据和程序。
- 输入设备:将数据和程序送入计算机。
- 输出设备:将计算处理的结果转化为人或其他设备所能识别或接收的信息形式。
- 互连结构:数据交换。

例 1.4 控制器的任务是什么？它主要由哪些部件组成？简述各组成部分的功能。

解 控制器的任务是协调计算机的各部件自动地工作。其工作实质上就是解释程序，它每次从存储器中读取一条指令，经过分析译码，产生一系列的控制信号，发往各个部件以控制它们的操作。连续不断、有条不紊地继续上述动作，即所谓执行程序。

控制器的主要组成部分及其功能分别为：

- 指令控制部件：由程序计数器 PC、指令寄存器 IR 和指令译码器 ID 组成。PC 又叫指令计数器，给出程序中指令在存储器中的单元地址。IR 保存当前正在执行的指令代码，决定指令的操作性质及参与操作的操作数地址等。ID 也叫操作码译码器，它将指令的操作码转换为相应的控制电位信号，指示各部件做什么操作。
- 地址形成部件：包括地址寄存器 AR、变址寄存器 XR 和地址计算部件。其功能主要是依据指令的寻址方式和指令的地址码部分生成实际的操作数地址。
- 定时部件：亦称时序部件，由时钟 CP 和时序信号产生器 TSG 组成。CP 是协调计算机各部件进行操作的同步时钟；TSG 的功能是按时间顺序，周而复始地发出若干节拍信号和脉冲。定时部件就是根据机器的时钟脉冲，发出全机所需的定时节拍信号和脉冲。各部件在不同的节拍信号控制下依次进行工作。
- 微操作控制部件：微操作控制部件根据指令控制部件给出的指令译码电位信号（进行什么操作）、时序部件给出的节拍信号和脉冲（指令执行到哪一步）和运算器 Flag 提供的状态信息，产生计算机指令系统中各种指令所需的各种微操作控制信号。然后再将这些控制信号发送给运算器、存储器、输入输出设备以及控制器本身。

例 1.5 何谓程序计数器？它有何作用？

解 程序计数器 PC 又叫指令计数器。它给出程序中指令在存储器中的单元地址。它兼有指令地址寄存器和计数器的功能。

取指令时，PC 指向要取指令的地址。当一条指令执行完毕时，PC 作为指令地址寄存器，其内容已变成下一条指令的地址。控制器依据 PC 的内容从存储器取出指令送到 IR 后，PC 将自动加 1，指向下一条指令的地址。注意，这里假定一个存储单元存储一条指令。若非顺序执行，只要将 PC 内容作相应改变，就可按新的序列顺序执行。

例 1.6 试述内存储器的组成和各组成部分的功能。

解 内存储器由存储体、选址系统、读写系统和存储时序控制线路构成。各部分的功能分别为：

- 存储体：存储体被分成许多存储单元，每个存储单元存放多位二进制信息，通过存储器地址可寻址存储单元。存储单元存放信息的位数通常是字节的整数倍。
- 选址系统：由存储地址寄存器 MAR、地址译码器和地址驱动器三部分组成。I/O 或 CPU 访存时，先将访存地址送 MAR，经地址译码器找到被访问的存储单元，才能由地址驱动器驱动该存储单元以实现读或写。
- 读写系统：包括存储缓冲寄存器 MBR 和读写线路。读出时，控制器发出读控制信号，借助于读出线路将由选址系统确定的存储单元内容读出送 MBR，以供 I/O 或 CPU 使用。写入时，先将要写入的数据送 MBR，控制器发出写控制信号，借助于写入线路，将 MBR 内容写入由选址系统确定的存储单元。

- 存储时序控制线路:由控制触发器,各种门电路和延迟线路等组成。接收来自 I/O 或 CPU 的启动、读写、清除等命令,产生一系列控制内存存储器完成读写等操作的信号。

例 1.7 存储器是怎样编址的? 128K 字、512K 字和 1M 字容量的存储器的十六进制编址范围如何表示?

解 存储单元按顺序编号,每个存储单元对应一个编号,此编号称为存储单元地址,简称地址。地址与存储单元是一一对应的,每个存储单元只有一个地址。

128K 字容量的存储器十六进制编址范围:00000H ~ 1FFFFH;

512K 字容量的存储器十六进制编址范围:00000H ~ 7FFFFH;

1M 字容量的存储器十六进制编址范围:00000H ~ FFFFFH。

需要注意的是,此时每个存储单元存放的数据位为一个字。

例 1.8 试说明计算机硬件的主要性能参数。

解 衡量计算机硬件性能的主要参数包括:

- 主频:主频或时钟周期是计算机的主要性能指标之一,它在很大程度上决定了计算机的运行速度。CPU 的工作节拍是由时钟 CP 控制的,时钟不断产生固定频率的时钟脉冲,这个时钟的频率就是 CPU 的主频。主频越高,CPU 的工作节拍就越快,运算速度就越高。主频通常用 1s 内处理器所能发出电子脉冲数来表示,单位一般为兆赫兹(MHz)或者吉赫兹(GHz)。

• 运算速度:运算速度是计算机工作能力和工作效率的主要表征,它取决于在给定的时间内处理器所能处理的数据量以及处理器的时钟频率。运算速度通常用每秒执行指令的条数来表示,其计量单位为 MIPS 或者 MFLOPS。

• 运算精度:运算精度通常用计算机能直接处理的二进制信息位数来衡量。这个位数一般和 CPU 中存储数据寄存器的位数相同,一般位数越多,精度越高。

• 主存存储容量:主存储器用来存储数据和程序,直接与 CPU 交换信息。主存的容量越大,可存储的数据和程序就越多,处理问题的能力也就越强;而且与外存储器的信息交换次数越少,系统的效率就越高。以字为单位的计算机常用字数乘以字长表示主存容量。

• 主存存储周期:将信息存入主存,称为写入;将信息从主存中取出,称为读出;对主存的读写,简称访存。对主存连续两次访存所允许的最短时间间隔,叫主存存取周期。存取周期既是表征主存性能的基本参数,也是反映计算机整机性能的重要参数。显然,存取周期愈小,表明从主存存取信息的时间愈短,计算机系统性能愈高。

需要说明的是,一台计算机硬件性能的高低好坏,考虑的因素应是多方面的,如系统结构、硬件组成、外设配置、吞吐率和响应时间,还有可靠性、可用性、可维性、完整性和安全性等等。不能片面强调某一项指标,应综合全面考虑。

例 1.9 何谓基本字长? 它对计算机的哪些部件有何影响?

解 参与运算操作数的基本位数称为基本字长。基本字长在一定情况下标志着计算精度。基本字长决定着寄存器、加法器、数据总线等的位数,直接影响着硬件的造价。

例 1.10 根据表 1.4 中的程序,结合图 1.7 中的计算机总体框图说明计算机的工作过程,并计算执行完该程序共访问多少次内存储器。

解 计算机开始执行程序,即开始了指令执行的过程。从 00A00H 存储单元取出指令 10800B01H 放入 IR,PC 内容加 1 变为 00A01H,IR 的内容经 ID 译码发现是取数指令,于是在执行指令阶段,将存储单元 00B01H 中的数 a 读到 R₈ 寄存器中;接着又进入取指令阶段,从 00A01H 存储单元取出指令 08800B00H 放入 IR,PC 内容加 1 变为 00A02H,IR 的内容经 ID 译码发现是乘法指令,于是在执行指令阶段,从 00B00H 存储单元取出乘数,将它与 R₈ 中的被乘数 a 进行乘法运算,乘积存入 R₈;接着又从 00A02H 存储单元取出指令 02800B02H 放入 IR,PC 内容加 1 变为 00A03H,IR 的内容经 ID 译码发现是加法指令,执行加法指令;如此执行程序中的每条指令,直至从 00A08H 存储单元取出停机指令并执行。停机指令使 TSG 不再发出节拍信号,计算机也停止了指令执行过程,该程序执行完毕。

表 1.4 计算 y 的机器语言程序

主存地址	指令或数据			说明	
00A00	10	8	00B01	取数: R ₈ ← a	
00A01	08	8	00B00	乘法: R ₈ ← a × x	
00A02	02	8	00B02	加法: R ₈ ← a × x + b	
00A03	08	8	00B00	乘法: R ₈ ← (a × x + b) × x	
00A04	02	8	00B03	加法: R ₈ ← (a × x + b) × x + c	
00A05	08	8	00B00	乘法: R ₈ ← ((a × x + b) × x + c) × x	
00A06	02	8	00B04	加法: R ₈ ← ((a × x + b) × x + c) × x + d	
00A07	14	8	00B05	存数: (00B05) ← R ₈ (y 的值)	
00A08	FF	0	00000	停机	
00B00	x			初始数据区	
00B01	a				
00B02	b				
00B03	c				
00B04	d				
00B05	y			结果数据区	

从表 1.3 知,除停机指令外,每种指令的取指和执行均需要访问存储器一次。因此,该程序共访问 17 次内存存储器。

例 1.11 计算机一般有哪两种分类方法?各分为哪些类型的计算机?

解 计算机按其用途即应用特点可分为通用计算机和专用计算机。

按计算机的规模将计算机分为巨型机、大型机、小型机、微型机、单片机等类型。

例 1.12 计算机一般可运用于哪些方面?对每一种运用类型各举两个实例加以说明。

解 计算机可运用于如下方面:

- 科学计算;例:计算化学,计算经济学(数值计算)。
- 数据处理;例:情报检索,图像处理,数据库处理。
- 实时控制;例:航天航空过程控制,工业生产过程控制,现代化武器系统控制。
- 辅助设计;例:飞机设计,基建工程绘图,机械电子设计。
- 智能模拟;例:自动程序设计,模拟训练系统,智能决策系统。

第二章 指令系统

例 2.1 解释下列术语:机器指令,寻址方式,定点数据类型,规格化浮点数,机器零,上溢,下溢,字符串,堆栈,向量,堆栈型指令,累加器指令,通用寄存器型指令,有效地址,形式地址,CISC,RISC。

解 机器指令:计算机设计者赋予计算机实现某种基本操作的命令。

寻址方式:又称为寻址技术,是确定操作数地址的方式。

定点数据类型:是各种数据中最简单、最基本的一种数据表示类型。定点数据中小数点的位置是固定的,它只能出现在数据左边的符号位后或者整个数据的最右端。定点数据可分为整数和小数两种。

规格化浮点数:为了提高运算精度,要使尾数的有效数字尽可能占满已有的位数,同时也使计算机实现浮点运算时有一个统一固定的标准形式,可将浮点数表示为规格化的形式。规格化对尾数 M 提出限制要求: $1/2 \leq |M| < 1$ 。浮点数的规格化形式称为规格化浮点数。

机器零:当尾数 $M=0$ 时,对所有 E 值均有 $N=0 \times R^E = 0$ 。当 $E \leq -2^n$, 并且 $M \neq 0$ 时, 所表示的数据 N 小于机器所能表示的最小数,一般以 $N=0$ 来处理,通常称之为“机器零”。“机器零”的标准格式是 $M=0, E=-2^n$, 即尾数为 0, 阶码为最小值。

上溢:指数据的绝对值太大,超出了数据类型的表示能力范围。

下溢:指数据的绝对值太小,使得数据无法有效地表示。

字符串:被处理的信息是一组字符的序列。

堆栈:堆栈实际上是一种数据有序表,某个时刻其中只有一个数据能够被访问,这个数据就是栈顶,通常需要一个专用的寄存器来指明,这个寄存器称为堆栈指针。对堆栈数据的处理依据后进先出的原则。

向量:向量数据是一组元素的有序集合,描述向量数据的参量有向量起始地址、向量长度和向量间距。

堆栈型指令:指令系统的一种,CPU 中操作数地址是隐含的,即在栈顶。

累加器型指令:指令系统的一种,CPU 中有一个操作数地址是隐含的,即累加器。

通用寄存器型指令:指令系统的一种,CPU 中操作数全显示给出,或者为寄存器地址,或者为主存地址。

有效地址:Effective Address,简写为 EA,表示能直接访问操作数的地址。

形式地址:Formal Address,又称为逻辑地址(Logic Address),通常指指令字中给出的地址。

CISC:Complex Instruction Set Computer,复杂指令集计算机。

RISC:Reduced Instruction Set Computer,精简指令集计算机。

例 2.2 什么是数据表示?试分析数据表示与数据结构研究内容的区别。

解 数据表示指的是能由计算机硬件直接识别的数据类型。数据表示研究的内容包

括数据的类型、取值范围、精度和误差。

数据结构则是在数据表示的基础上研究如何让计算机处理硬件不能够直接识别的数据类型,这需要软件的配合。

例 2.3 试述计算机发展中的数据表示的演变。

解 数据表示是一个变化和发展的领域。早期计算机,由于硬件成本昂贵,只支持简单的定点数据类型,机器也相应地只有定点运算和逻辑运算指令。定点数的表示范围小,使用很不方便。随着计算机在数值计算领域中应用需求的发展,计算机中很快又增加了浮点数据表示,相应地增加了浮点运算指令,机器也设置了浮点运算部件和相应的控制线路。随着计算机在商业和事务处理领域获得了广泛应用,增加十进制数据表示,可以使计算机的效率大幅度提高。计算机硬件可以直接识别十进制数,需要增加十进制运算指令,其硬件也要支持十进制数据的各种操作。随着计算机硬件成本的进一步降低以及计算机系统软件的发展和应用领域需求的扩大,许多计算机又增设了各种新的数据表示,如字符串、堆栈等,而有些大型机、巨型机还设置了向量。这些数据表示提高了计算机的性能,同时也提高了软件设计和生产的效率。

例 2.4 把下列各数表示成 16 位二进制补码定点整数:378, -2, 2045, 16381。

解 假定采用 1 位符号位表示,

$$[378]_{\text{补}} = 0000\ 0001\ 0111\ 1010$$

$$[-2]_{\text{补}} = 1111\ 1111\ 1111\ 1110$$

$$[2045]_{\text{补}} = 0000\ 0111\ 1111\ 1101$$

$$[16381]_{\text{补}} = 0011\ 1111\ 1111\ 1101$$

例 2.5 根据下列给定的数码寄存器位数,确定定点表示整数的范围:8 位,12 位,16 位,24 位,32 位,64 位。

解 数据的表示范围与数据是否采用符号有关,而当采用符号时还与采用的码制有关。例如 8 位数码寄存器位数:无符号整数表示范围 $0 \sim 2^8 - 1$;如果是带符号的数据表示,采用原码和反码的表示范围为: $-(2^7 - 1) \sim (2^7 - 1)$,采用补码和移码的表示范围为: $-2^7 \sim (2^7 - 1)$ 。整数表示范围如表 2.5 所示。

表 2.5 定点整数表示范围

位数	无符号数表示范围	带符号数表示范围	
		原码、反码	补码、移码
8 位	$0 \sim 2^8 - 1$	$-(2^7 - 1) \sim (2^7 - 1)$	$-2^7 \sim (2^7 - 1)$
12 位	$0 \sim 2^{12} - 1$	$-(2^{11} - 1) \sim (2^{11} - 1)$	$-2^{11} \sim (2^{11} - 1)$
16 位	$0 \sim 2^{16} - 1$	$-(2^{15} - 1) \sim (2^{15} - 1)$	$-2^{15} \sim (2^{15} - 1)$
24 位	$0 \sim 2^{24} - 1$	$-(2^{23} - 1) \sim (2^{23} - 1)$	$-2^{23} \sim (2^{23} - 1)$
32 位	$0 \sim 2^{32} - 1$	$-(2^{31} - 1) \sim (2^{31} - 1)$	$-2^{31} \sim (2^{31} - 1)$
64 位	$0 \sim 2^{64} - 1$	$-(2^{63} - 1) \sim (2^{63} - 1)$	$-2^{63} \sim (2^{63} - 1)$

例 2.6 20 位数码寄存器(一位符号位)能表示二进制定点整数的范围多大? 若用

BCD 码表示十进制定点整数,其数值范围多大?

解 用 20 位数码寄存器(1 位符号位)表示二进制定点整数的表示范围:原码、反码为 $-(2^{19} - 1) \sim (2^{19} - 1)$;补码为 $-2^{19} \sim (2^{19} - 1)$ 。

用 BCD 码表示十进制定点整数:无符号表示为 0 ~ 99999,有符号表示为 -9999 ~ 9999。

例 2.7 32 位浮点二进制数,8 位(含一位符号位)为用补码表示的阶码,24 位(含一位符号位)为补码表示的规格化尾数,试指出它所表示的最大正数与最小正数的数据格式。

解 假定采用如图 2.9 所示的浮点数据格式。

符号位	阶码 E(8 位)	尾数 M(23 位)
-----	-----------	------------

图 2.9 浮点数据的一般格式

该数据格式能够表示最大正数和最小正数的形式分别为:

最大正数: 0 0111 1111 111 1111 1111 1111 1111 1111

最小正数: 0 1000 0000 100 0000 0000 0000 0000 0000

例 2.8 最常用的指令格式有哪几种?

解 根据指令长度可分为定长指令格式和可变长指令格式;根据操作码长度可分为定长操作码指令格式和可变长操作码指令格式;根据地址码的个数可分为零地址指令格式、一地址指令格式、二地址指令格式和三地址指令格式等。

例 2.9 根据给定的真值,求其移码: + 011011011, - 11001101, - 00010001, + 00011101。

解 一个数的移码是该数补码的反符号。假定采用 1 位符号位,10 位数据字长,则它们的移码分别为:

$$[+011011011]_{\text{移}} = 1011011011$$

$$[-11001101]_{\text{移}} = 0100110011$$

$$[-00010001]_{\text{移}} = 0111101111$$

$$[+00011101]_{\text{移}} = 1000011101$$

例 2.10 如果向量寄存器由 128 个单元组成,要处理向量长度为 L,怎样分段处理该向量数据?

解 可以使用 $\lceil L/128 \rceil$ 个向量寄存器表示一个向量。

例 2.11 根据给定的压缩向量 A 及压缩位向量 A,还原稀疏向量 A。

A 的压缩向量

A ₁
A ₃
A ₅
A ₈

A 的压缩位向量

1	0	1	0	1	0	0	1
---	---	---	---	---	---	---	---

解

A ₁
0
A ₃
0
A ₅
0
0
A ₈

例 2.12 计算机指令应包含哪些最基本的成分？各有何作用？

解 计算机指令最基本的成分称为指令三要素：

- 操作码(Operation Code),指明进行什么操作,如加减等,常用 opcode 或 OP 标记;
- 源操作数地址(Source Operand Reference),指出操作数在何处,常用 S 标记;
- 目的操作数地址(Result Operand Reference):指出指令操作结果送往何处,常用 R 标记。

例 2.13 试比较寄存器—寄存器型指令与存储器—存储器型指令的优缺点。

解 寄存器—寄存器(R - R)型指令系统中,运算指令中不含存储器型操作数。机器在执行这类指令的过程中,只对寄存器中的操作数进行操作,从寄存器中取操作数,结果也放到寄存器中,不需要访存,因此速度很快。但是,由于所有数据来源于存储器并且最终要存放到存储器中,故寄存器—寄存器型指令的执行需要使用访存指令来从存储器中存取操作数。

存储器—存储器(M - M)型指令系统中,部分运算指令的操作数都存放在存储器中。显然两个以上的操作数存放在存储器中,使得指令的执行需要 4 次以上的访存(1 次取指、2 次读操作数、1 次写操作数),这导致指令执行速度缓慢。这种指令可以不使用寄存器,从这个角度看,CPU 的实现代价相对较低。

例 2.14 选择指令格式应考虑哪些因素？简述其理由。

解 选择指令格式应考虑的因素有：指令字长、操作码长度、地址码个数等。

指令字长即一条指令中包含的二进制代码的位数。指令长度的选择有长指令和短指令,从访问存储器和指令复杂性的角度来看,短指令比长指令好。短指令能够节省存储空间,减少访问存储器次数,执行速度快。但由于指令中包含的信息少,因而指令功能较弱。相反,长指令的功能比较强,便于程序设计,但长指令又可能造成利用率的浪费。对于指令长度的选择,还可选择定长指令格式和可变长指令格式。定长指令格式具有结构简单,便于实现等优点。变长指令格式结构灵活,能充分利用指令长度,但指令的控制复杂。

关于操作码的选择有定长操作码和可变长度操作码两种选择,操作码位数越多,所能表示的操作种类就越多。固定长度操作码指令格式中操作码的长度固定,且集中放在指令字的一个字段中,其余部分全部用于地址码。这种格式的优点是规则,有利于简化硬件译码逻辑,减少指令译码时间。可变长度操作码又称为扩展操作码,操作码的长度允许有几种不同的选择,对地址数少的指令允许操作码长些,对地址数多的指令,操作码就短些。这种指令格式可以缩短指令的长度,减少程序的总位数以及增加指令字所能表示的操作

信息。当然,此时操作码译码复杂,在硬件设计和实现上要困难些。

关于地址码个数的选择有零地址、一地址、二地址和三地址等指令格式。当指令长度一定时,地址码与操作码是相互制约的。同样,当地址位数确定后,地址个数与每个地址的位数也是相互制约的。地址数越少,指令的功能就越基本、越简单,CPU的复杂性也就越低,而且指令的长度也越短;但另一方面,程序中的指令条数也就越多,这通常会增加程序量和复杂度。实际上,地址个数的选择在很大程度上依赖于指令系统的结构。一般来说,堆栈结构的指令系统主要采用零地址指令,累加器结构的指令系统主要采用一地址指令。当代的计算机大多既有二地址指令又有三地址指令。地址个数的选择还与地址的类型有关。对于固定的字长来说,地址越短,其个数就可以越多。寄存器的地址比存储器的地址短得多,所以,操作数在寄存器中的指令宜采用三地址结构。三地址指令因不会破坏源操作数,使用比较简单,便于代码生成。目前的计算机中已普遍采用了操作数在寄存器中的三地址指令。

例 2.15 利用扩展操作码法,构成 40 条指令系统。要求,OP = 4,13 条;OP = 7,6 条;OP = 9,7 条;OP = 12,14 条。

解 假定指令字长为 16 位,本题的答案有多种,下面给出两种仅供参考,分别如表 2.6 和表 2.7 所示。

表 2.6 扩展操作码方法 1

4 位操作码 13 条指令	0000	x x x	x x	x x x	x x x x
	0001	x x x	x x	x x x	x x x x

	1011	x x x	x x	x x x	x x x x
	1100	x x x	x x	x x x	x x x x
7 位操作码 6 条指令	1101	000	x x	x x x	x x x x
	1101	001	x x	x x x	x x x x

	1101	100	x x	x x x	x x x x
	1101	101	x x	x x x	x x x x
9 位操作码 7 条指令	1101	110	00	x x x	x x x x

	1101	110	11	x x x	x x x x
	1101	111	00	x x x	x x x x

	1101	111	10	x x x	x x x x
12 位操作码 14 条指令	1101	111	11	000	x x x x

	1101	111	11	111	x x x x
	1110	000	00	000	x x x x

	1110	000	00	101	x x x x

表 2.7 扩展操作码方法 2

4位操作码 13条指令	0000	x x x x	x x x x	x x x x
	0001	x x x x	x x x x	x x x x

	1011	x x x x	x x x x	x x x x
	1100	x x x x	x x x x	x x x x
7位操作码 6条指令	1101	000x	x x x x	x x x x
	1101	001x	x x x x	x x x x

	1101	100x	x x x x	x x x x
	1101	101x	x x x x	x x x x
9位操作码 7条指令	1110	0000	0 x x x	x x x x
	1110	0000	1 x x x	x x x x

	1110	0010	1 x x x	x x x x
	1110	0011	0 x x x	x x x x
12位操作码 14条指令	1111	0000	0000	x x x x
	1111	0000	0001	x x x x

	1111	0000	1100	x x x x
	1111	0000	1101	x x x x

例 2.16 试述各种寻址方式的特点。

表 2.4 基本寻址方式的比较

寻址方式	物理地址形成规则	主要优点	主要缺点
立即数寻址	操作数 = A	无需访问存储器	操作数范围受限
直接寻址	EA = A	简单	寻址空间受限
间接寻址	EA = (A)	寻址空间大	多次访问主存
寄存器寻址	EA = R	无需访问存储器	寻址空间受限
寄存器间接寻址	EA = (R)	寻址空间大	多访问主存一次
偏移寻址	EA = (R) + A	灵活	复杂
堆栈寻址	EA = (SP)地址隐含	缩短指令字长	应用范围有限

例 2.17 如何利用变址寻址方式,进行 1000 个数在内存中的移动?

解 假定内存储器中 1000 个数据的起始地址为 SDATA, 移动目的地的起始地址为 DDATA, 每个数据 2 个字节。数据移动的 80×86 指令如下:

```
SDATA DW 1,2,...,1000;    定义 1000 个数据  
DDATA DW 1000 DUP(?);  定义存放 1000 个数据的空间  
.....  
MOV SI,0;                变址寄存器置初值为 0  
MOV CX,1000;              设置循环次数  
GOON: MOV AX,SDATA[SI]  
        MOV DDATA[SI],AX  
        ADD SI,2  
        LOOP GOON
```

例 2.18 试比较变址寻址与基址寻址方式异同点。它们有什么优缺点? 为什么说它们都是一种复合寻址?

解 基址寻址, 实际上是相对于基址寄存器 BR(Base Register)的偏移寻址。基址寄存器中存放基址。所谓基地址(Base Address), 就是当前程序访存操作的一个参考基准地址。指令中的地址字段给出的是相对于基地址的偏移量 A, 操作数的有效地址是把基址寄存器的内容加上指令字中的形式地址得到, 即 $EA = (BR) + A$ 。不同的 A 值, 指出了以(BR)为基址的一个数据块中的不同单元。

变址寻址是相对于变址寄存器 X(Index Register)的偏移寻址。其有效地址是把变址寄存器的内容加上指令字中的形式地址得到, 即 $EA = (X) + A$ 。程序是通过改变变址寄存器 X 的内容来访问主存的。变址寻址的一个重要用途就是, 通过有规律地改变 X 的内容, 遍历从基准地址 A 开始的一片单元, 从而可以通过循环执行同一条指令实现对一批数据的处理。

两者相同之处: 在形式上以及计算操作数的有效地址的方法上, 变址寻址和基址寻址是相似的。指令不做任何修改, 可以通过修改寄存器的内容实现访问单元地址的偏移。它们的优点是提高了寻址的灵活性, 扩大了寻址范围, 但同时增加了实现硬件的复杂性——除了指令译码的复杂性增加, 往往还需要专门硬件支持。

两者不同之处: 基址寻址 $EA = (BR) + A$, 基址寄存器 BR 的内容是不变的, 形式地址 A 是相对该基准地址的偏移量, 由于操作码的限制, 基址寻址中的偏移量位数较短。基址寻址主要是解决程序逻辑空间与存储器物理空间的无关性问题。而变址寻址 $EA = (X) + A$ 。形式地址 A 给出的是一个存储器地址基准, 其内容是不变的, 变址寄存器 X 中存放的是相对于该基准地址的偏移量, 变址寻址中的偏移量位数相对较大。变址寻址主要是为了可以编写出高效地访问一片存储空间的程序。

基址寻址和变址寻址的优点是提高了寻址的灵活性, 扩大了寻址范围, 但同时增加了实现的硬件复杂性——除了指令译码的复杂性增加, 往往还需要专门硬件支持, 例如专用寄存器、地址加法器等。

两种或者两种以上的寻址方式相结合而形成的寻址方式称为复合寻址方式。基址寻址和变址寻址方式同属于偏移寻址, 而偏移寻址组合了直接寻址和寄存器间接寻址两种方式, 故它们都是复合寻址。

例 2.19 试述常用指令类型有哪些?

解 据第一章知,存储程序计算机都应当具有数据传送、数据存储、数据处理、操作控制和操作判断这五大功能。因此基本的指令系统中应包含以下几大类型的指令:数据传送、算术逻辑运算、数据转换、I/O、程序控制和系统控制类指令等。

例 2.20 完善的指令系统应满足哪些基本要求?

解 完善的指令系统应满足的基本要求有:

- 完备性:指令系统的完备性是指在一个有限可用的存储空间,对于任何可解的问题,编制计算程序时,指令系统所提供的指令足够使用。这是一个原则性要求,很难确定一个完备性的标准。完备性要求指令系统丰富、功能齐全、使用方便。

- 有效性:有效性是指利用该指令系统编写的程序能够高效率地运行。高效率主要表现在程序占据存储空间小、执行速度快。强调有效性,一直是计算机系统实际的重要原则之一,也是传统的 CISC 的出发点。

- 规整性:规整性包括指令系统的对称性、匀齐性、指令格式和数据格式的一致性。

- 对称性:在指令系统中所有的寄存器和存储器单元都可同等对待,所有的指令都可使用各种寻址方式。

- 匀齐性:一种操作性质的指令可以支持各种数据类型,如算术运算指令可支持字节、字、双字整数的运算,十进制数运算和单、双精度浮点数运算等。

- 指令格式和数据格式的一致性:指令长度和数据长度有一定的关系,以方便处理和存取。

- 兼容性:系列机各种机之间具有相同的基本结构和共同的指令系统,因而指令系统是兼容的,所以各机种上软件可以通用。

例 2.21 RISC 和 CISC 指令系统各有何特点?

解 RISC 指令系统具有以下主要特点:

- 指令功能简单,指令条数少;

- 采用定长、简单的指令格式,典型的为 4 个字节;

- 寻址方式简单,数量少,不采用存储器间接寻址技术;

- 只有 Load/Store 指令能够访问主存,在一条指令中,操作数访存寻址不会超过一次;

- 运算类指令多采用三地址寄存器寻址方式,不直接访存;

- 设置大量的寄存器,指令操作大多都在寄存器之间进行;

- 对于有浮点处理部件的计算机,使用大量的浮点寄存器;

- 一个周期出现一个指令执行结果,但是其性能的发挥极度依赖于编译器的优化。

CISC 结构追求的目标是强化指令功能,减少程序的指令条数,以达到提高性能的目的。增强 CISC 指令系统功能主要是从以下几个方面着手的:

- 面向目标程序增强指令功能。主要方法包括:提高运算型指令功能、提高传送指令功能以及增加程序控制指令功能等。

- 面向高级语言和编译程序改进指令系统。主要方法包括:增加对高级语言和编译系统支持的指令功能或者设计直接支持高级语言命令的指令系统。

- 面向操作系统的优化实现改进指令系统。可以通过设置支持系统工作状态和访问方式转移的指令、支持进程转移的指令、支持进程同步和互斥的指令等措施,达到优化实现操作系统的目地。

相对 RISC 指令系统,CISC 指令系统存在以下问题:

- 复杂指令不能有效地得到利用;
- 复杂指令系统会降低整个机器的执行速度;
- 复杂指令带来了计算机组成及实现上的复杂性,不便于用 VLSI 实现;
- 指令系统设计时间长,且由于系统复杂,可能包含更多的设计错误。

第三章 运算方法与运算器

例 3.1 解释下列术语:逻辑运算,算术运算,逻辑移位,算术移位,半加器,全加器,串行进位,并行进位,进位链,并行进位加法器,进位产生函数,进位传递函数,通用函数发生器,存储进位加法器,对阶,规格化,左规,右规,舍入,尾数调整。

解 逻辑运算:就是对逻辑数进行的运算。其主要特点是:寄存器中的数据按位进行操作,即各位同时进行指定的操作,每位均按二值布尔规则运算,各位之间没有相关关系,无进位和溢出,运算比较简单。

算术运算:运算中,每一位的运算结果与其余各位存在关联,存在进位或者溢出,运算相对比较复杂。另外,由于采用的数据表示和码制不同,导致算术运算的运算规则和方法亦不同。

逻辑移位:是指对寄存器中整组数据进行的移位。逻辑移位按照移位的方向又分为逻辑右移和逻辑左移。

算术移位:是指寄存器带符号数的移位。执行算术移位时,寄存器中的数是带符号的算术运算数,可以是定点数也可以是浮点数的尾数或者阶码。移位后,数的符号位保持不变。

半加器:一位二进制加法单元有 3 个输入量,操作数 A_i 和 B_i ,低位来的进位信号 C_{i-1} 。如果只考虑 2 个输入量相加,不考虑低位进位信号,则这种加法单元被称为半加器 HA(Half Adder)。

全加器:一位二进制加法单元有 3 个输入量,操作数 A_i 和 B_i ,低位来的进位信号 C_{i-1} 。具有全部 3 个输入的加法单元被称为全加器 FA(Full Adder)。

串行进位:两数进行相加(减)时,加法器逐级地形成各位进位,每一级进位直接依赖于前一级进位,通常称行波进位(Ripple Carry)。

并行进位:又叫做同时进位或先行进位(Carry Look Ahead)。在串行进位的基础上,为了提高速度,采用并行进位逻辑并行地形成各位进位,使得两数相加时各位可并行进行。

进位链:由于进位是由低位向高位逐级传递,进位的逻辑结构形似链条,故常称进位传递逻辑为进位链(Carry Link)。

并行进位加法器:具有并行进位链采用并行进位的加法器。由于使用并行进位链同时形成了各位的低位进位,故 n 位全加器可以一步实现 n 位数据相加。实际使用时,由于高位的进位形成逻辑涉及输入变量过多,将受到器件扇入系数的限制。因此在位数较多的加法器中,常采用分级、分组的进位链结构,即将所有参与运算的位分成适当位数的组,组内并行,组间或串行或并行。

进位产生函数:进位公式 $C_i = A_i B_i + (A_i + B_i)C_{i-1} = G_i + P_i C_{i-1}$;其中 $G_i = A_i B_i$,称为第 i 位的进位产生函数(Carry Generate Function)或进位生成条件,或称为本地进位或绝对进位。

进位传递函数:进位公式 $C_i = A_i B_i + (A_i + B_i)C_{i-1} = G_i + P_i C_{i-1}$;其中 $P_i = A_i + B_i$,称为第 i 位的进位传递函数(Carry Propagate Function)或进位传递条件, $P_i C_{i-1}$ 则称为传递进位和条件进位。

通用函数发生器:将若干位全加器、并行进位链及输入选择门集成于一块芯片上,即构成多功能算术逻辑运算单元 ALU。ALU 因可产生多种输出函数,故又可称之为通用函数发生器。

存储进位加法器:存储进位加法器 CSA(Carry Save Adders)与常规的带进位传递的加法器 CPA(Carry Propagate Adders)相比,它的基本思想是在同一年级加法器中将进位信息暂时保留,留待下一级加法器或以后级进行处理。

对阶:浮点数运算中,使两数的阶码相等的过程,叫做对阶。

规格化:尾数不是规格化浮点数,必须将尾数移位,使之规格化,并相应调整阶码,这一过程叫做规格化。规格化浮点数的尾数必须满足 $1/2 \leq |M| < 1$ 。

左规:若浮点运算结果的两个尾符相同,且与尾数小数点后第一位相同,则需左规,即尾数左移。每左移一位,阶码减 1,直至尾数小数点后第一位与尾符不同时为止。

右规:若浮点运算结果的两位符号位不等(即为 $01. \times \times \cdots \times$ 或 $10. \times \times \cdots \times$ 的形式),则需右规,即将尾数右移一位,阶码加 1。显然,右规最多右移一位。

舍入:在浮点运算的对阶或右规时,尾数要右移,这样,尾数的末 1 位或几位可能因超出机器的允许位数而被丢掉,从而造成一定误差。为了减小误差,通常要进行舍入处理。把为减小误差而进行的舍入处理,简称为舍入。

尾数调整:将被除数尾数 $|M_A|$ 调整为小于除数的尾数 $|M_B|$,即经过调整后被除数的尾数为 $|M'_A|$,应使 $|M'_A| < |M_B|$ 。

例 3.2 已知 $A = 1001, B = 0101$,试计算:

- (1) $C \leftarrow A \wedge B$
- (2) $C \leftarrow A \vee B$

(3) $C \leftarrow A \oplus B$

(4) $C \leftarrow A \odot B$

解 (1)

1001	A 的值
Λ	B 的值
<hr/>	
0101	
<hr/>	
0001	

故 $C = 0001$;

(2) $C = 1101$;

(3) $C = 1100$;

(4) $C = 0011$ 。

例 3.3 按操作性质,移位有几种类型? 各有何特点?

解 移位的类型按移位性质可分为逻辑移位(Logic Shift)、循环移位(Circular Shift)和算术移位(Arithmetic Shift)。

逻辑移位是指对寄存器中整组数据进行的移位。移位时寄存器末端触发器补入 0。逻辑移位运算主要用于数据处理中字的装配、拼组与拆散等操作，也可用于程序控制中的状态位和特殊信息的调用。当移位寄存器的末端触发器与其他寄存器有移位通路时，逻辑移位运算还可实现信息的串行传送。

循环移位是指寄存器两端触发器有移位通路，形成闭合的移位环路。在移位时，寄存器内信息便在这个环路内循环流动。循环移位主要用于寄存器移位时信息仍须保留的情况。

算术移位是指寄存器带符号数的移位。执行算术移位时，寄存器中的数是带符号的算术运算数，运算的结果通常会引起数值的变化：右移一位，相当于带符号的数除以 2（乘以 1/2）；左移一位，相当于带符号的数乘以 2。

例 3.4 已知 A 和 B，求 $[A + B]_{\text{补}}$ 和 $[A - B]_{\text{补}}$ 。

- (1) $A = 0.1011$, $B = -0.1110$
- (2) $A = -0.1101$, $B = -0.1010$
- (3) $A = 0.1101$, $B = 0.0001$
- (4) $A = 0.1110$, $B = 0.0010$

解 (1) $[A]_{\text{补}} = 00.1011$, $[B]_{\text{补}} = 11.0010$, $[-B]_{\text{补}} = 00.1110$, $[A + B]_{\text{补}} = 11.1101$; $[A - B]_{\text{补}} = 01.1001$, 表示正溢出。其演算过程及结果如算式 3.1 所示。

$\begin{array}{r} 00.1011 \\ +) \quad 11.0010 \\ \hline 11.1101 \end{array}$	$\begin{array}{r} 00.1011 \\ +) \quad 11.1110 \\ \hline 01.1001 \end{array}$
$[A + B]_{\text{补}} = 11.1101$	$[A - B]_{\text{补}} \text{ 结果正溢出}$

算式 3.1

(2) $[A]_{\text{补}} = 11.0011$, $[B]_{\text{补}} = 11.0110$, $[-B]_{\text{补}} = 00.1010$, $[A + B]_{\text{补}} = 10.1001$, 表示负溢出; $[A - B]_{\text{补}} = 11.1101$ 。

(3) $[A]_{\text{补}} = 00.1101$, $[B]_{\text{补}} = 00.0001$, $[-B]_{\text{补}} = 11.1111$, $[A + B]_{\text{补}} = 00.1110$; $[A - B]_{\text{补}} = 00.1100$ 。

(4) $[A]_{\text{补}} = 00.1110$, $[B]_{\text{补}} = 00.0010$, $[-B]_{\text{补}} = 11.1110$, $[A + B]_{\text{补}} = 01.0000$, 表示正溢出; $[A - B]_{\text{补}} = 00.1100$ 。

例 3.5 已知 A 和 B，求 $[A + B]_{\text{移}}$ 和 $[A - B]_{\text{移}}$ 。

- (1) $A = 0.1011$, $B = -0.0010$
- (2) $A = -0.1101$, $B = -0.1010$
- (3) $A = -0.1001$, $B = 0.1101$
- (4) $A = 0.1101$, $B = 0.1011$

解 (1) $[A]_{\text{移}} = 01.1011$, $[B]_{\text{移}} = 11.1110$, $[-B]_{\text{移}} = 00.0010$, $[A + B]_{\text{移}} = 01.1001$; $[A - B]_{\text{移}} = 01.1101$ 。其演算过程及结果如算式 3.2 所示。

$$\begin{array}{r}
 & 00.1011 \\
 +) & 11.1110 \\
 \hline
 & 01.1001 \\
 [A+B]_B = 01.1001
 \end{array}
 \quad
 \begin{array}{r}
 & 01.1011 \\
 +) & 00.0010 \\
 \hline
 & 01.1101 \\
 [A-B]_B = 01.1101
 \end{array}$$

算式 3.2

(2) $[A]_B = 00.0011$, $[B]_B = 11.0110$, $[-B]_B = 00.1010$, $[A+B]_B = 11.1001$, 表示负溢出; $[A-B]_B = 00.1101$ 。

(3) $[A]_B = 00.0111$, $[B]_B = 00.1101$, $[-B]_B = 11.0011$, $[A+B]_B = 01.0100$; $[A-B]_B = 11.1010$, 表示负溢出;

(4) $[A]_B = 01.1101$, $[B]_B = 00.1011$, $[-B]_B = 11.0101$, $[A+B]_B = 10.1000$, 表示正溢出; $[A-B]_B = 01.0010$ 。

例 3.6 设计一个 9 位的先行进位加法器, 要求每 3 位为一组, 采用两级先行进位线路。

解 将参加运算的数据位分别用 A_i 和 B_i 表示, 每位进位用 C_i 表示, 最低位进位为 C_0 。每位的进位产生函数为 G_i , 每位的进位传递函数为 P_i 。将 9 位数据分成 3 组, 每组 3 位。分组后, 小组的进位产生函数分别为 G_I 、 G_{II} 和 G_{III} ; 小组的进位传递函数分别为 P_I 、 P_{II} 和 P_{III} 。采用两级先行进位线路时:

第一级, 小组内并行进位链。第一小组内的进位逻辑为:

$$\begin{aligned}
 C_1 &= G_1 + P_1 C_0 \\
 C_2 &= G_2 + P_2 G_1 + P_2 P_1 C_0 \\
 C_3 &= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0 = G_I + P_I C_0
 \end{aligned}$$

其中, $G_I = G_3 + P_3 G_2 + P_3 P_2 G_1$, $P_I = P_3 P_2 P_1$ 。

第二小组内的进位逻辑为:

$$\begin{aligned}
 C_4 &= G_4 + P_4 C_I \\
 C_5 &= G_5 + P_5 G_4 + P_5 P_4 C_I \\
 C_6 &= G_6 + P_6 G_5 + P_6 P_5 G_4 + P_6 P_5 P_4 C_I = G_{II} + P_{II} C_I
 \end{aligned}$$

这里 C_I 就是第一组产生的组间进位 C_3 。

第三小组内的进位逻辑为:

$$\begin{aligned}
 C_7 &= G_7 + P_7 C_{II} \\
 C_8 &= G_8 + P_8 G_7 + P_8 P_7 C_{II} \\
 C_9 &= G_9 + P_9 G_8 + P_9 P_8 G_7 + P_9 P_8 P_7 C_{II} = G_{III} + P_{III} C_{II}
 \end{aligned}$$

这里 C_{II} 就是第二组产生的组间进位 C_6 。

第二级, 小组间并行进位链。小组间的并行进位链表达式:

$$\begin{aligned}
 C_I &= G_I + P_I C_0 \\
 C_{II} &= G_{II} + P_{II} G_I + P_{II} P_I C_0 \\
 C_{III} &= G_{III} + P_{III} G_{II} + P_{III} P_{II} G_I + P_{III} P_{II} P_I C_0
 \end{aligned}$$

需要说明的是,如果只需要构成 9 位并行加法电路, C_{III} 可以不必由组间并行进位链产生,这是因为 $C_{III} = C_9$ 。这样可以减少并行进位产生逻辑的复杂性。

有了以上表达式后,就可以设计出类似 SN74181 和 SN74182 功能的逻辑线路 CA 和 CB,其中 CA 完成 3 位数据的并行加法并可以产生小组的进位产生函数与进位传递函数,CB 根据 3 组 CA 产生的进位产生函数和进位传递函数生成组间并行进位。具体的运算过程为:

- (1)各组 CA 根据本组参与运算的数据 A_i 和 B_i ,形成本组的进位产生函数 G_i 、 G_{II} 、 G_{III} 和进位传递函数 P_i 、 P_{II} 、 P_{III} ;
- (2)CA 根据 G_i 、 G_{II} 、 G_{III} 、 P_i 、 P_{II} 、 P_{III} 生成组间并行进位 C_i 、 C_{II} 、 C_{III} ;
- (3)各组 CA 根据 A_i 和 B_i 以及组间并行进位 C_i 、 C_{II} 以及 C_0 共同进行组内的并行加法。

例 3.7 采用 SN74181 ALU 和 SN74182 器件,构成一个三级的 64 位先行进位的 ALU。

解 构成一个两级 16 位并行进位 ALU,如图 3.7 所示。

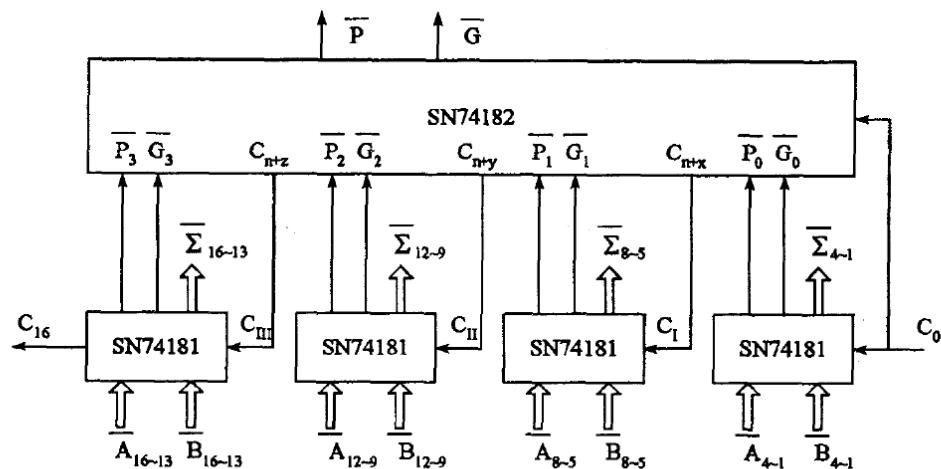


图 3.7 两级 16 位并行进位 ALU

在此基础上,将 4 个这样的 16 位并行进位 ALU 拼接在一起,加上更高一级的组间并行进位逻辑,就可以构造出 64 位三级并行进位 ALU。第一级为一个 SN74182,构成 16 位 ALU 间的并行进位逻辑;第二级为 4 个 SN74182,构成 16 位 ALU 内部的先行进位逻辑;第三级为 16 个 SN74181,完成 64 位数据的加法。三级 ALU 中的第一级和第二级的结构如图 3.8 所示。

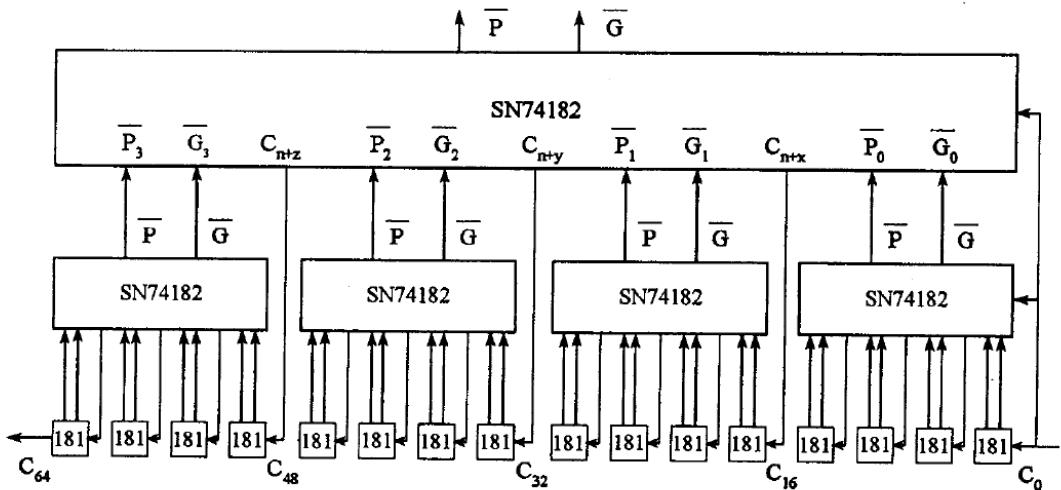


图 3.8 三级 64 位并行进位 ALU

64 位并行进位 ALU 的工作过程如下：

- (1) 各个 SN74181 根据本组参与运算的数据, 形成 4 位数据的进位产生函数和进位传递函数;
- (2) 第 2 级 SN74182 根据这些进位产生函数和进位传递函数值, 形成 16 位数据的进位产生函数与进位传递函数;
- (3) 第 1 级 SN74182 根据第 2 级 SN74182 形成的进位产生函数和进位传递函数值; 形成 C_{16} 、 C_{32} 和 C_{48} ; 同时第 2 级 SN74182 形成 16 位并行加法所需的组间进位;
- (4) 16 组 SN74181 同时完成 64 位数据的并行加法。

例 3.8 已知 A 和 B, 用 Booth 乘法求 $[A \times B]_b$ 和 $[B \times A]_b$ 。

(1) 已知 $A = 0.11011$, $B = -0.11110$, 求 $[A \times B]_b = ?$ 和 $[B \times A]_b = ?$

解 $[A]_b = 00.11011$, $[B]_b = 1.00010$, $[-A]_b = 11.00101$ 。运算过程如算式 3.3 所示。

故 $[A \times B]_b = 1.0011010110$ 。

另外, $[B]_b = 11.00010$, $[A]_b = 0.11011$, $[-B]_b = 00.11110$ 。运算过程如算式 3.4 所示。

故 $[B \times A]_b = 1.0011010110$ 。

部分积							
[P ₀] _#	00. 00000						B _{n+1}
[P ₁] _#	$\rightarrow 1$ 00. 00000						0
	$+[-A]_{\#}$ 11. 00101						0
	11. 00101						0
[P ₂] _#	$\rightarrow 1$ 11. 10010						1
	$+[A]_{\#}$ 00. 11011						1
	00. 01101						0
[P ₃] _#	$\rightarrow 1$ 00. 00110						0
[P ₄] _#	$\rightarrow 1$ 00. 00011						0
[P ₅] _#	$\rightarrow 1$ 00. 00001						0
	$+[-A]_{\#}$ 11. 00101						0
[P ₆] _#	11. 00110						1

算式 3.3

部分积							
[P ₀] _#	00. 00000						A _{n+1}
	$+[-B]_{\#}$ 00. 11110						0
	00. 11110						0
[P ₁] _#	$\rightarrow 1$ 00. 01111						1
[P ₂] _#	$\rightarrow 1$ 00. 00111						1
	$+[B]_{\#}$ 11. 00010						1
	11. 01001						1
[P ₃] _#	$\rightarrow 1$ 11. 10100						0
	$+[-B]_{\#}$ 00. 11110						0
	00. 10010						0
[P ₄] _#	$\rightarrow 1$ 00. 01001						1
[P ₅] _#	$\rightarrow 1$ 00. 00100						1
	$+[B]_{\#}$ 11. 00010						1
[P ₆] _#	11. 00110						0

算式 3.4

(2) 已知 $A = -0.1011$, $B = -0.0101$, 求 $[A \times B]_{\text{补}} = ?$ 和 $[B \times A]_{\text{补}} = ?$

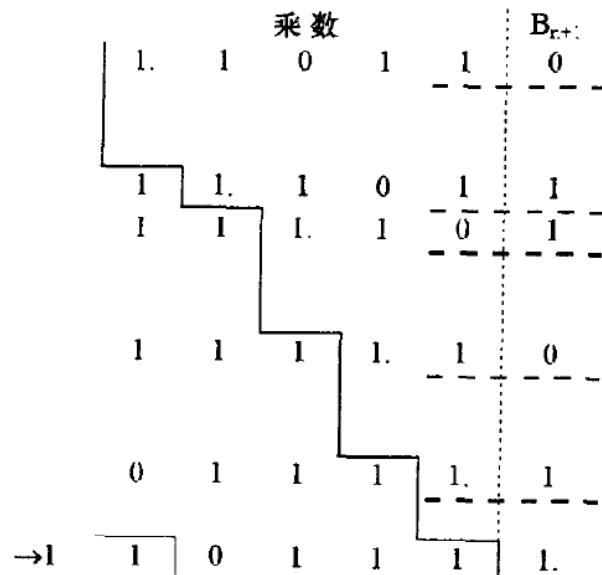
解 $[A]_{\text{补}} = 11.0101$, $[B]_{\text{补}} = 1.1011$, $[-A]_{\text{补}} = 00.1011$ 。运算过程如算式 3.5 所示。

$$\begin{array}{r}
 \text{部分积} \\
 [P_0]_{\text{补}} \quad 00.0000 \\
 +[-A]_{\text{补}} \quad 00.1011 \\
 \hline
 00.1011
 \end{array}$$

$$\begin{array}{r}
 [P_1]_{\text{补}} \quad \rightarrow 1 \quad 00.0101 \\
 [P_2]_{\text{补}} \quad \rightarrow 1 \quad 00.0010 \\
 +[A]_{\text{补}} \quad 11.0101 \\
 \hline
 11.0111
 \end{array}$$

$$\begin{array}{r}
 [P_3]_{\text{补}} \quad \rightarrow 1 \quad 11.1011 \\
 +[-A]_{\text{补}} \quad 00.1011 \\
 \hline
 00.0110
 \end{array}$$

$$\begin{array}{r}
 [P_4]_{\text{补}} \quad \rightarrow 1 \quad 00.0011 \\
 [P_5]_{\text{补}} \quad 00.0011
 \end{array}$$



算式 3.5

故 $[A \times B]_{\text{补}} = 0.00110111$ 。

另外, $[B]_{\text{补}} = 11.1011$, $[A]_{\text{补}} = 1.0101$, $[-B]_{\text{补}} = 00.0101$ 。运算过程如算式 3.6 所示。

故 $[B \times A]_{\text{补}} = 0.00110111$ 。

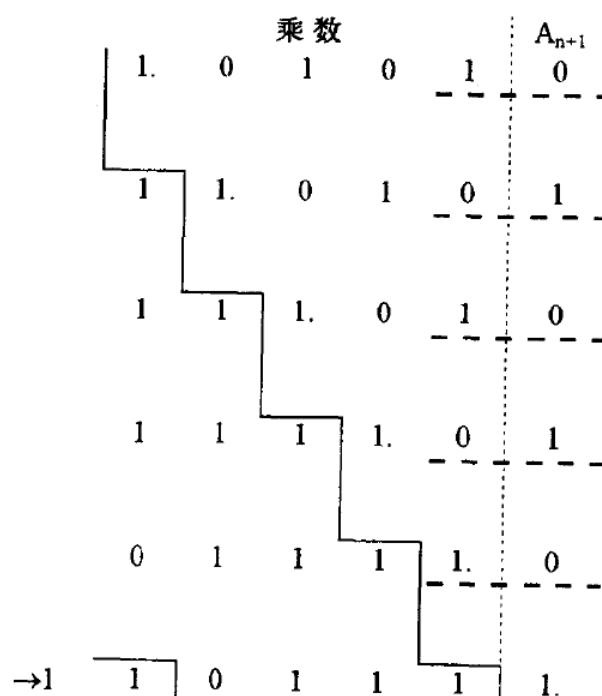
$$\begin{array}{r}
 \text{部分积} \\
 [P_0]_{\text{补}} \quad 00.0000 \\
 +[-B]_{\text{补}} \quad 00.0101 \\
 \hline
 00.0101
 \end{array}$$

$$\begin{array}{r}
 [P_1]_{\text{补}} \quad \rightarrow 1 \quad 00.0010 \\
 +[B]_{\text{补}} \quad 11.1011 \\
 \hline
 11.1101
 \end{array}$$

$$\begin{array}{r}
 [P_2]_{\text{补}} \quad \rightarrow 1 \quad 11.1110 \\
 +[-B]_{\text{补}} \quad 00.0101 \\
 \hline
 00.0011
 \end{array}$$

$$\begin{array}{r}
 [P_3]_{\text{补}} \quad \rightarrow 1 \quad 00.0001 \\
 +[B]_{\text{补}} \quad 11.1011 \\
 \hline
 11.1100
 \end{array}$$

$$\begin{array}{r}
 [P_4]_{\text{补}} \quad \rightarrow 1 \quad 11.1110 \\
 +[-B]_{\text{补}} \quad 00.0101 \\
 \hline
 00.0011
 \end{array}$$



算式 3.6

(3) 已知 $A = -1$, $B = -1$, 求 $[A \times B]_{\text{补}} = ?$ 和 $[B \times A]_{\text{补}} = ?$

解 如果使用小数运算, 假定尾数位数为 3 位, 则 $[A]_{\text{补}} = 11.000$, $[B]_{\text{补}} = 1.000$, $[-A]_{\text{补}} = 01.000$ 。运算过程如算式 3.7 所示。

		部分积	乘 数			B_{n+1}
$[P_0]_{\text{补}}$		00. 000	1.	0	0	0
$[P_1]_{\text{补}}$	$\rightarrow 1$	00. 000	0	1.	0	0
$[P_2]_{\text{补}}$	$\rightarrow 1$	00. 000	0	0	1.	0
$[P_3]_{\text{补}}$	$\rightarrow 1$	00. 000	0	0	0	1.
$+[-A]_{\text{补}}$		01. 000				
$[P_4]_{\text{补}}$		01. 000	→1	0	0	0
						1.

算式 3.7

故 $[A \times B]_{\text{补}} = 01.0000000(+1)$, 表示溢出。注意, 此时是补码小数乘法唯一一种溢出的情况。同理 $[B \times A]_{\text{补}} = 01.0000000(+1)$, 也表示溢出。

另外, 如果使用整数运算, 假定符号位后位数为 3 位, 则 $[A]_{\text{补}} = [-001]_{\text{补}} = 11111$, $[B]_{\text{补}} = 1111$, $[-A]_{\text{补}} = 00001$ 。运算过程如算式 3.8 所示。

		部分积	乘 数			B_{n+1}
$[P_0]_{\text{补}}$		00000	1	1	1	0
$+[-A]_{\text{补}}$		00001				
		00001				
$[P_1]_{\text{补}}$	$\rightarrow 1$	00000	1	1	1	1
$[P_2]_{\text{补}}$	$\rightarrow 1$	00000	0	1	1	1
$[P_3]_{\text{补}}$	$\rightarrow 1$	00000	0	0	1	1
$[P_4]_{\text{补}}$		00000	→1	0	1	1

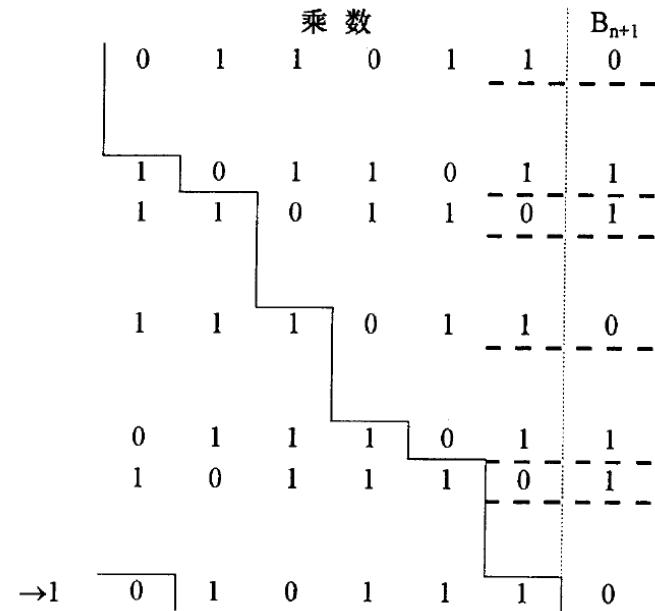
算式 3.8

故: $[A \times B]_{\text{补}} = 0000001(+1)$ 。注意, 此时对于补码定点乘法并没有溢出。同理, $[B \times A]_{\text{补}} = 0000001(+1)$ 。

(4) 已知 $A = -01011$, $B = 11011$, 求 $[A \times B]_{\text{补}} = ?$ 和 $[B \times A]_{\text{补}} = ?$

解 $[A]_{\text{补}} = 1110101$, $[B]_{\text{补}} = 011011$, $[-A]_{\text{补}} = 0001011$ 。运算过程如算式 3.9 所示。

	部分积
$[P_0]_{\text{补}}$	0000000
	$+[-A]_{\text{补}}$
	0001011
	<u>0001011</u>
$[P_1]_{\text{补}}$	$\rightarrow 1$ 0000101
$[P_2]_{\text{补}}$	$\rightarrow 1$ 0000010
	$+[A]_{\text{补}}$
	1110101
	<u>1110111</u>
$[P_3]_{\text{补}}$	$\rightarrow 1$ 1111011
	$+[-A]_{\text{补}}$
	0001011
	<u>0000110</u>
$[P_4]_{\text{补}}$	$\rightarrow 1$ 0000011
$[P_5]_{\text{补}}$	$\rightarrow 1$ 0000001
	$+[A]_{\text{补}}$
	1110101
$[P_6]_{\text{补}}$	<u>1110110</u>

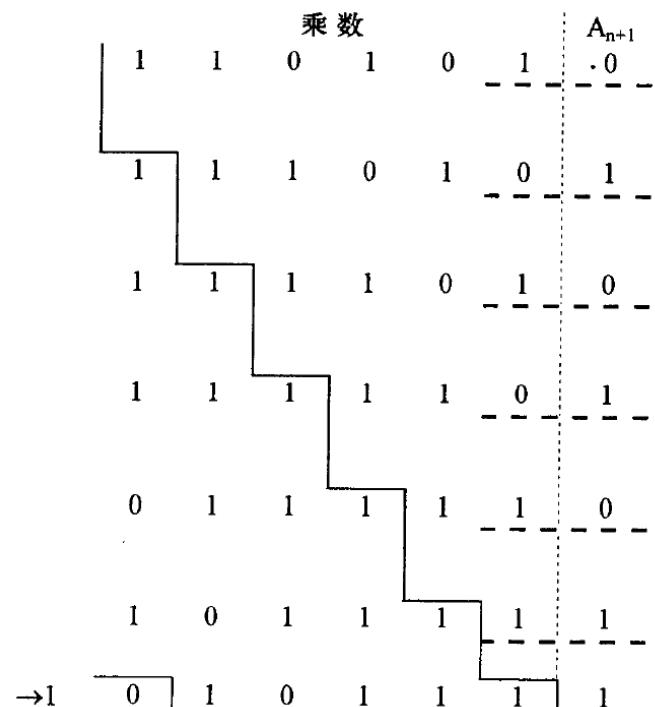


算式 3.9

故 $[A \times B]_{\text{补}} = 11011010111(-0100101001)$ 。

另外, $[B]_{\text{补}} = 0011011$, $[A]_{\text{补}} = 110101$, $[-B]_{\text{补}} = 1100101$ 。运算过程如算式 3.10 所示。

	部分积
$[P_0]_{\text{补}}$	0000000
	$+[-B]_{\text{补}}$
	1100101
	<u>1100101</u>
$[P_1]_{\text{补}}$	$\rightarrow 1$ 1110010
	$+[B]_{\text{补}}$
	0011011
	<u>0001101</u>
$[P_2]_{\text{补}}$	$\rightarrow 1$ 0000110
	$+[-B]_{\text{补}}$
	1100101
	<u>1101011</u>
$[P_3]_{\text{补}}$	$\rightarrow 1$ 1110101
	$+[B]_{\text{补}}$
	0011011
	<u>0010000</u>
$[P_4]_{\text{补}}$	$\rightarrow 1$ 0001000
	$+[-B]_{\text{补}}$
	1100101
	<u>1101101</u>
$[P_5]_{\text{补}}$	$\rightarrow 1$ 1110110
$[P_6]_{\text{补}}$	1110110



算式 3.10

故 $[B \times A]_{\text{补}} = 11011010111(-0100101001)$ 。

例 3.9 已知 A 和 B, 用补码两位乘法求 $[A \times B]_{\text{补}}$ 和 $[B \times A]_{\text{补}}$ 。

(1) 已知 $A = 0.10110$, $B = -0.00011$, 求 $[A \times B]_{\text{补}} = ?$ 和 $[B \times A]_{\text{补}} = ?$

解 $[A]_{\text{补}} = 000.10110$; $[-A]_{\text{补}} = 111.01010$; $[2A]_{\text{补}} = 001.01100$; $[-2A]_{\text{补}} = 110.10100$ 。因 $n=5$ 为奇数, 所以 $[B]_{\text{补}}$ 需 1 位符号位: $[B]_{\text{补}} = 1.11101$ 。运算过程如算式 3.11 所示。

		部分积				乘数				B_{n+1}
$[P_0]_{\text{补}}$		000.	00000			1.	1	1	0	0
	$+ [A]_{\text{补}}$	000.	10110			1	0			
		000.	10110							
$[P_1]_{\text{补}}$	$\rightarrow 2$	000.	00101			1.	1	1	1	0
	$+ [-A]_{\text{补}}$	111.	01010							
		111.	01111							
$[P_2]_{\text{补}}$	$\rightarrow 2$	111.	11011			1	1	0	1	1
$[P_3]_{\text{补}}$	$\rightarrow 1$	111.	11101	$\rightarrow 2$		0	1	1	1	1

算式 3.11

故 $[A \times B]_{\text{补}} = 1.1110111110$ 。

另外, $[B]_{\text{补}} = 111.11101$; $[-B]_{\text{补}} = 000.00011$; $[2B]_{\text{补}} = 111.11010$; $[-2B]_{\text{补}} = 000.00110$ 。因 $n=5$ 为奇数, 所以 $[A]_{\text{补}}$ 需 1 位符号位, $[A]_{\text{补}} = 0.10110$ 。运算过程如算式 3.12 所示。

		部分积				乘数				A_{n+1}
$[P_0]_{\text{补}}$		000.	00000			0.	1	0	1	0
	$+ [-2B]_{\text{补}}$	000.	00110							
		000.	00110							
$[P_1]_{\text{补}}$	$\rightarrow 2$	000.	00001			1	0	0.	1	1
	$+ [2B]_{\text{补}}$	111.	11010							
		111.	11011							
$[P_2]_{\text{补}}$	$\rightarrow 2$	111.	11110			1	1	1	0	0
	$+ [B]_{\text{补}}$	111.	11101							
		111.	11011							
$[P_3]_{\text{补}}$	$\rightarrow 1$	111.	11101	$\rightarrow 2$		1	1	1	1	0
										0.

算式 3.12

故 $[B \times A]_{\text{补}} = 1.1110111110$ 。

(2) 已知 $A = -0.011010$, $B = -0.011101$, 求 $[A \times B]_{\text{补}} = ?$ 和 $[B \times A]_{\text{补}} = ?$

解 $[A]_{\text{补}} = 111.100110$; $[-A]_{\text{补}} = 000.011010$; $[2A]_{\text{补}} = 111.001100$; $[-2A]_{\text{补}} = 000.110100$ 。因 $n=6$ 为偶数, 所以 $[B]_{\text{补}}$ 需 2 位符号位: $[B]_{\text{补}} = 11.100011$ 。运算过程如算

式 3.13 所示。

		部分积						
[P ₀] _#		000. 000000						B _{n+1}
	+[-A] _#	000. 011010						0
		000. 011010						
[P ₁] _#	→2	000. 000110						
	+[A] _#	111. 100110						1
		111. 101100						
[P ₂] _#	→2	111. 111011						
	+[-2A] _#	000. 110100						0
		000. 101111						
[P ₃] _#	→2	000. 001011						
[P ₄] _#		000. 001011						
	→2	1 1 1 1 0 0 . 1 1 0 1						

算式 3.13

故 $[A \times B]_{\text{补}} = 0.001011110010$ 。

另外, $[B]_{\text{补}} = 111.100011$; $[-B]_{\text{补}} = 000.011101$; $[2B]_{\text{补}} = 111.000110$; $[-2B]_{\text{补}} = 000.111010$ 。因 $n=6$ 为偶数, 所以 $[A]_{\text{补}}$ 需 2 位符号位: $[A]_{\text{补}} = 11.100110$ 。运算过程如算式 3.14 所示。

		部分积						
[P ₀] _#		000. 000000						A _{n+1}
	+[-2B] _#	000. 111010						0
		000. 111010						
[P ₁] _#	→2	000. 001110						
	+[2B] _#	111. 000110						1
		111. 010100						
[P ₂] _#	→2	111. 110101						
	+[-2B] _#	000. 111010						0
		000. 101111						
[P ₃] _#	→2	000. 001011						
[P ₄] _#		000. 001011						
	→2	1 1 1 1 0 0 . 1 1 0 1						

算式 3.14

故 $[B \times A]_{\text{补}} = 0.001011110010$ 。

(3) 已知 $A = -1$, $B = -0.11010$, 求 $[A \times B]_{\text{补}} = ?$ 和 $[B \times A]_{\text{补}} = ?$

解 $[A]_{\text{补}} = 111.00000$; $[-A]_{\text{补}} = 001.00000$; $[2A]_{\text{补}} = 110.00000$; $[-2A]_{\text{补}} = 010.00000$ 。因 $n=5$ 为奇数, 所以 $[B]_{\text{补}}$ 需 1 位符号位: $[B]_{\text{补}} = 1.00110$ 。运算过程如算式 3.15 所示。

	部分积				乘数	B_{n+1}
$[P_0]_*$	000.00000				1. 0 0 1	-1 0 0
	+[-2A]_*	010.00000				0
		010.00000			0 0 1	-1
$[P_1]_*$	$\rightarrow 2$	000.10000			0 0 0	-1 0 0
	+[2A]_*	110.00000				
		110.10000			0 0 0	-1 0 0
$[P_2]_*$	$\rightarrow 2$	111.10100				
	+[-2A]_*	010.00000			0 0 0	-1 0 0
		001.10100				
$[P_3]_*$	$\rightarrow 1$	000.11010	$\rightarrow 2$	0	0 0 0	1.

算式 3.15

故 $[A \times B]_*$ = 0.1101000000。

另外, $[B]_*$ = 111.00110; $[-B]_*$ = 000.11010; $[2B]_*$ = 110.01100; $[-2B]_*$ = 001.10100。因 $n=5$ 为奇数, 所以 $[A]_*$ 需 1 位符号位: $[A]_*$ = 1.00000。运算过程如算式

3.16 所示。

	部分积				乘数	A_{n+1}
$[P_0]_*$	000.00000				1. 0 0 0	0 0 0
$[P_1]_*$	$\rightarrow 2$	000.00000			0 0 1	0 0 0
$[P_2]_*$	$\rightarrow 2$	000.00000			0 0 0	1. 0 0
	+[-2B]_*	001.10100				
		001.10100			0 0 0	
$[P_3]_*$	$\rightarrow 1$	000.11010	$\rightarrow 2$	0	0 0 0	1.

算式 3.16

故 $[B \times A]_*$ = 0.1101000000。

(4) 已知 $A = 01011$, $B = -11011$, 求 $[A \times B]_*$ = ? 和 $[B \times A]_*$ = ?

解 $[A]_*$ = 00001011; $[-A]_*$ = 11110101; $[2A]_*$ = 00010110; $[-2A]_*$ = 11101010。

因 $n=5$ 为奇数, 所以 $[B]_*$ 需 1 位符号位: $[B]_*$ = 100101。运算过程如算式 3.17 所示。

	部分积				乘数	B_{n+1}
$[P_0]_*$	00000000				1. 0 0 1	-0 1 0
	+[A]_*	00001011				0
		00001011			1 0 0 1	-0 1 0
$[P_1]_*$	$\rightarrow 2$	00000010			0 1 1 1	-1 0 0
	+[A]_*	00001011				
		00001101			0 1 1 1	-1 0 0
$[P_2]_*$	$\rightarrow 2$	00000011				
	+[-2A]_*	11101010			0 1 1 1	-1 0 0
		11101101				
$[P_3]_*$	$\rightarrow 1$	11110110	$\rightarrow 2$	0	1 0 1	1

算式 3.17

故 $[A \times B]_{\text{补}} = 11011010111 (+ 1011010111)$ 。

另外, $[B]_{\text{补}} = 11100101$; $[-B]_{\text{补}} = 00011011$; $[2B]_{\text{补}} = 11001010$; $[-2B]_{\text{补}} = 00110110$ 。

因 $n=5$ 为奇数, 所以 $[A]_{\text{补}}$ 需 1 位符号位: $[A]_{\text{补}} = 001011$ 。运算过程如算式 3.18 所示。

	部分积				乘数			A_{n+1}
$[P_0]_{\text{补}}$	0	0	1	0	1	1	0	-
	$+[-B]_{\text{补}}$	0	0	1	0	0	-	-
	0	0	1	0	1	0	1	-
$[P_1]_{\text{补}}$	$\rightarrow 2$	0	0	0	1	0	1	-
	$+[-B]_{\text{补}}$	0	0	0	1	0	1	-
	0	0	1	0	1	0	1	-
$[P_2]_{\text{补}}$	$\rightarrow 2$	0	0	0	0	0	1	-
	$+[B]_{\text{补}}$	1	1	0	1	1	0	-
	1	1	0	1	1	1	0	-
$[P_3]_{\text{补}}$	$\rightarrow 1$	1	1	1	0	1	1	0
	$\rightarrow 2$	0	1	0	1	1	1	..

故: $[B \times A]_{\text{补}} = 11011010111 (+ 1011010111)$ 。

例 3.10 补码乘法是否会溢出? 若溢出, 何时溢出?

解 补码乘法会溢出。由于参加运算的两个数均为定点小数, 即 $|A| \leq 1, |B| \leq 1$, 故 $|C| = |A| \times |B| \leq 1$ 。对于补码数据表示来说, 当 $A = B = -1$ 时, $C = 1$, 这是唯一一种溢出的情况。

例 3.11 证明补码一位乘法任一部分积的绝对值小于等于 1。

证明 可以采用数学归纳法进行证明。

(1) 乘法进行之初, 部分积 $P_0 \equiv 0$, 其绝对值小于 1, 满足条件; 假定 P_i 是第 i 步产生的部分积, 其绝对值小于等于 1, 即 $|P_i| \leq 1$ 。

(2) 根据补码一位乘法的运算规则可知, 在 $[P_i]_{\text{补}}$ 的基础上, $[P_{i+1}]_{\text{补}}$ 的形成有三种可能: 直接右移一位、加 $[A]_{\text{补}}$ 后右移一位、加 $[-A]_{\text{补}}$ 后右移一位。注意, 右移一位表示除以 2。由于 $|P_i| \leq 1, |A| \leq 1$, 则 $|P_{i+1}| \leq 1$ 。

归纳(1)、(2)可知, 补码一位乘法任一部分积的绝对值小于等于 1。证毕。

实际上, 当运算中采用两位符号位时, 如果某个部分积等于 +1, 则其符号位为 01, 表示发生正溢出。观察算式 3.7 可知, 当 $A = B = -1$ 时, 在补码乘法的最后一步, 由于没有移位出现部分积等于 +1 的情况。

那么, 部分积会不会出现等于 -1 的情况呢? 假如是最后一步产生的部分积具有 11.0…0 的形式, 表示最后的乘积为 -1。根据乘法规则可知, 只有 $(+1) \times (-1)$ 时结果才为 -1, 而 +1 不能参加补码小数的运算。假如是中间某个部分积 $P_i = 11.0…0$, 该部分积是先加然后移位形成的, 因此移位前的和数为 10.0…0, 表示 -2。而 -2 只有通过 $(-1) + (-1)$ 形成, 这表明 $P_{i-1} = 11.0…0$ 。依此类推, 最后推到 $P_0 = 11.0…0$, 这与 $P_0 = 0$ 是矛盾的。因此部分积不会出现等于 -1 的情况。

例 3.12 用流程图分别描述补码一位乘法和补码两位乘法算法流程。

解 补码一位乘法的算法流程如图 3.9 所示。

补码两位乘法的算法流程将因为尾数位数 n 的奇偶而不同,当 n 为奇数时如图 3.10 所示,当 n 为偶数时如图 3.11 所示。

对比图 3.9 和图 3.10 不难发现,两位乘法算法与一位乘法算法流程的主要区别在于 Cnt_x 的初值不同,参加运算的判断位不同,每次运算的选择内容不同,运算后移位操作的位数以及最后一步对部分积的移位操作不同。

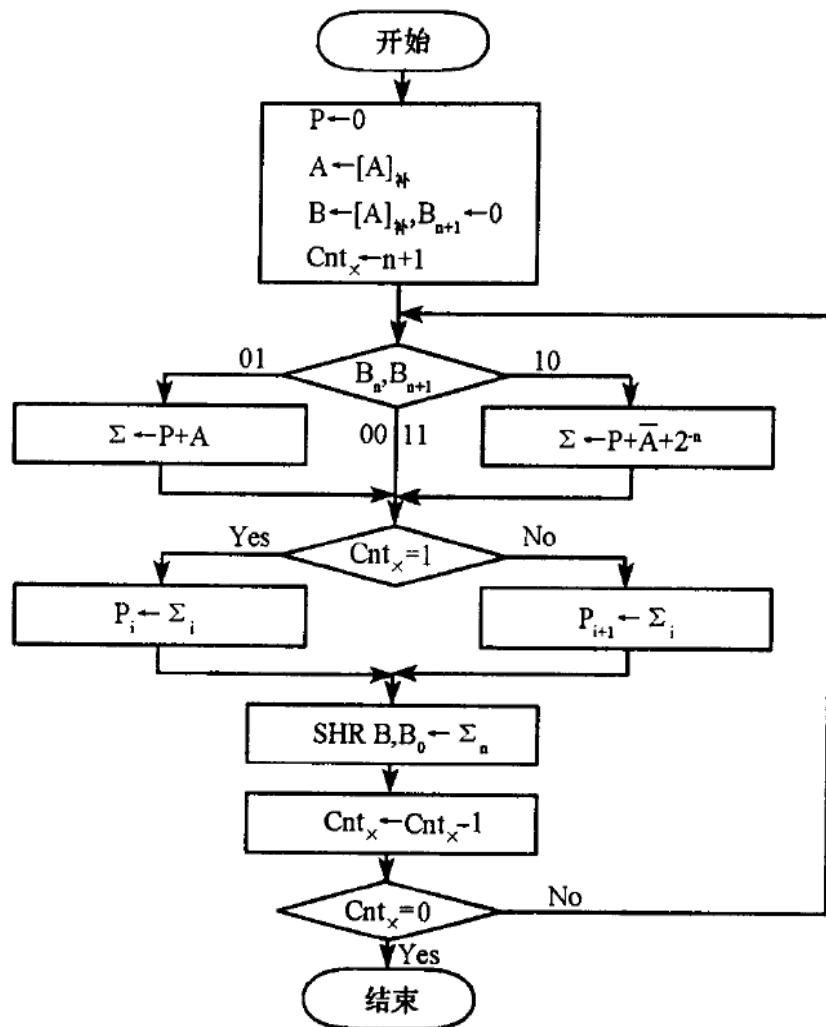


图 3.9 补码一位乘法算法流程图

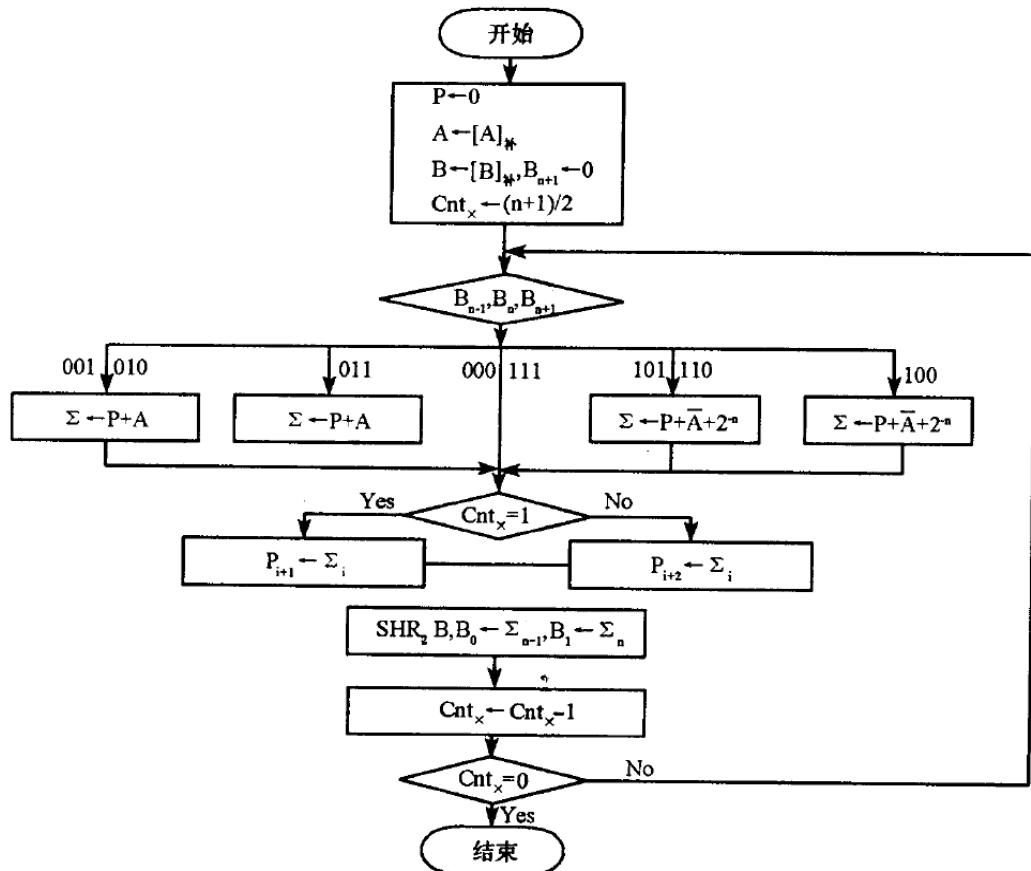


图 3.10 补码两位乘法算法(n 为奇数)流程图

对于两位乘法算法,对比图 3.10 和图 3.11 不难发现,当 n 为奇数和偶数时的主要区别在于 Cnt_x 的初值不同以及最后一步对部分积的移位操作不同。

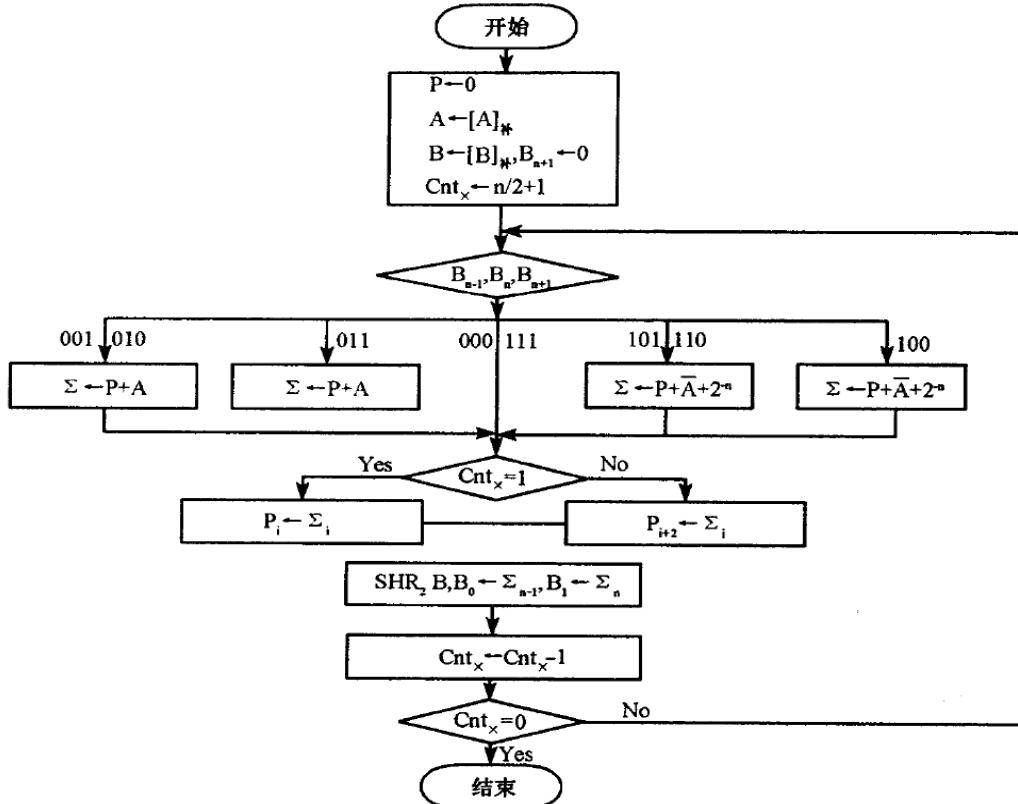


图 3.11 补码两位乘法算法(n 为偶数)流程图

例 3.13 补码两位乘法在工程实践上要注意哪些问题？与补码一位乘法相比要注意什么？

解 在工程实践上，补码两位乘法应该注意许多具体实现上的问题，例如：

(1) 参加运算的判断位为 3 位；每次均是将移位后乘数的最后两位与附加判断位一起判断，为保证运算的正确性，运算前附加位 B_{n+1} 需要被清零。另外，由于尾数位数 n 可能为奇数或者偶数，必须选择适当的乘数符号位，即 n 为奇数时取 1 位符号位， n 为偶数时取两位符号位。

(2) 每次加法完成后，寄存器均需右移两位。右移时，部分积最低两位被移入乘数最高位。最后一步部分积移位的位数需要根据 n 的位数进行判断，即 n 为奇数时右移一位，而 n 为偶数时不右移。

(3) 针对部分积需要增加判溢出电路。乘法过程中符号位出现 001 时表示溢出。需要注意的一段话是“当 $A = B = -1$ 时，乘法结果为 +1，发生溢出，这是补码乘法溢出的唯一一种情况”。当然，也可以在乘法进行前就使用判断电路判“ $A = B = -1$ ”，如果条件满足则不必进行乘法。

两位比较乘法器与一位比较乘法器相比设备量增加不多：运算中参加的判断位为 3

位，故译码器变为 3 位译码； SW_A 为子倍数选择开关，变为 5 选 1； P, SW_P, Σ 的符号位变为 3 位； Σ 可以实现右斜送 2 位和 1 位；计数器初值变小。

例 3.14 以两位乘法为基础，试推导 3 位补码乘法的算法。

解 补码两位乘法是将 Booth 一位乘法两步合并成一步而导出的。一位乘法是将本次乘数位 B_n 与上一步乘数位 B_{n+1} 进行比较，从而决定下一步进行什么运算。

设某一步的部分积为 $[P_i]_{\text{补}}$, 据递推公式可得部分积 $[P_{i+1}]_{\text{补}}$:

$$[P_{i+1}]_{\text{补}} = 2^{-1} \{ [P_i]_{\text{补}} + (B_{n-i+1} - B_{n-i})[A]_{\text{补}} \}$$

有了 $[P_{i+1}]_{\text{补}}$, 又可以根据递推公式得部分积为 $[P_{i+2}]_{\text{补}}$:

$$[P_{i+2}]_{\text{补}} = 2^{-1} \{ [P_{i+1}]_{\text{补}} + (B_{n-i} - B_{n-i-1})[A]_{\text{补}} \}$$

有了 $[P_{i+2}]_{\text{补}}$, 又可以根据递推公式得部分积为 $[P_{i+3}]_{\text{补}}$:

$$[P_{i+3}]_{\text{补}} = 2^{-1} \{ [P_{i+2}]_{\text{补}} + (B_{n-i-1} - B_{n-i-2})[A]_{\text{补}} \}$$

将上面三步乘法合并成一步, 即可从 $[P_i]_{\text{补}}$ 一步求得 $[P_{i+3}]_{\text{补}}$:

$$\begin{aligned}[P_{i+3}]_{\text{补}} &= 2^{-1} \{ [P_{i+2}]_{\text{补}} + (B_{n-i-1} - B_{n-i-2})[A]_{\text{补}} \} \\&= 2^{-2} \{ [P_{i+1}]_{\text{补}} + (B_{n-i} - B_{n-i-1})[A]_{\text{补}} + 2(B_{n-i-1} - B_{n-i-2})[A]_{\text{补}} \} \\&= 2^{-3} \{ [P_i]_{\text{补}} + (B_{n-i+1} - B_{n-i})[A]_{\text{补}} + 2[(B_{n-i} - B_{n-i-1}) \\&\quad + 2(B_{n-i-1} - B_{n-i-2})][A]_{\text{补}} \} \\&= 2^{-3} \{ [P_i]_{\text{补}} + (B_{n-i+1} + B_{n-i} + 2B_{n-i-1} - 4B_{n-i-2})[A]_{\text{补}} \}\end{aligned}$$

从 3 位乘法的角度看, 上式中的 $[P_{i+3}]_{\text{补}}$ 应该是 $[P_{i+1}]_{\text{补}}$, 即有了部分积 $[P_i]_{\text{补}}$, 根据判断位 B_{n-i+1} 、 B_{n-i} 、 B_{n-i-1} 和 B_{n-i-2} 的译码判断值即可得出 $[P_{i+1}]_{\text{补}}$ 。补码 3 位乘法的规则如表 3.6 所示。

表 3.6 补码 3 位比较乘法规则

判断位 B_{n-1} B_{n-2} B_n B_{n+1}				新部分积	操作
0 0 0 0				$[P_{i+3}]_{\text{补}} = 2^{-3} [P_i]_{\text{补}}$	$\rightarrow 3$
0 0 0 1				$[P_{i+3}]_{\text{补}} = 2^{-3} \{ [P_i]_{\text{补}} + [A]_{\text{补}} \}$	$+ [A]_{\text{补}}, \rightarrow 3$
0 0 1 0				$[P_{i+3}]_{\text{补}} = 2^{-3} \{ [P_i]_{\text{补}} + [2A]_{\text{补}} \}$	$+ [2A]_{\text{补}}, \rightarrow 3$
0 0 1 1				$[P_{i+3}]_{\text{补}} = 2^{-3} \{ [P_i]_{\text{补}} + [3A]_{\text{补}} \}$	$+ [3A]_{\text{补}}, \rightarrow 3$
0 1 0 0				$[P_{i+3}]_{\text{补}} = 2^{-3} \{ [P_i]_{\text{补}} + [4A]_{\text{补}} \}$	$+ [4A]_{\text{补}}, \rightarrow 3$
0 1 0 1				$[P_{i+3}]_{\text{补}} = 2^{-3} \{ [P_i]_{\text{补}} + [-4A]_{\text{补}} \}$	$+ [-4A]_{\text{补}}, \rightarrow 3$
1 0 0 0				$[P_{i+3}]_{\text{补}} = 2^{-2} \{ [P_i]_{\text{补}} + [-3A]_{\text{补}} \}$	$+ [-3A]_{\text{补}}, \rightarrow 3$
1 0 0 1				$[P_{i+3}]_{\text{补}} = 2^{-2} \{ [P_i]_{\text{补}} + [-2A]_{\text{补}} \}$	$+ [-2A]_{\text{补}}, \rightarrow 3$
1 0 1 0				$[P_{i+3}]_{\text{补}} = 2^{-2} \{ [P_i]_{\text{补}} + [-A]_{\text{补}} \}$	$+ [-A]_{\text{补}}, \rightarrow 3$
1 1 0 1				$[P_{i+3}]_{\text{补}} = 2^{-3} [P_i]_{\text{补}}$	$\rightarrow 3$

按照补码乘法规则,符号位要参加运算,下面分别讨论被乘数和乘数的符号位问题。

同补码两位乘法,对于补码三位乘法,被乘数 A 需要使用 4 位符号位。由于 $A = -1$ 时, A 的补码是具有意义的, $[4A]_{\text{补}} = 1100.00\cdots 0$, 而 $[-4A]_{\text{补}} = 0100.00\cdots 0$ 。然而,如果使用 3 位符号位, $[-4A]_{\text{补}}$ 将无法表示。同样,为保证在进行加法时部分积的符号位不被破坏,部分积也要采用 4 位符号位。

乘法进行的每一步,都是将乘数的最后三位与附加判断位一起判断,因此乘数符号位的位数与尾数位数 n 相加应该是 3 的倍数。当 n 为 3 的整数倍时,乘数尾数符号位为 3 位,进行 $n/3 + 1$ 步乘法;当 n 被 3 除余 1 时,乘数尾数符号位为 2 位,进行 $(n+2)/3$ 步乘法;当 n 被 3 除余 2 时,乘数尾数符号位为 1 位,进行 $(n+1)/3$ 步乘法。

每进行一步乘法,部分积和乘数均右移 3 位。但是对于最后一位部分积的移位需根据尾数位数 n 分别处理:当 n 为 3 的整数倍时,不移位;当 n 被 3 除余 1 时,右移一位;当 n 被 3 除余 2 时,右移两位。

例 3.15 已知 A 和 B, 用 Booth 除法求 $[C]_{\text{补}}$ 和 $[2^{-n}R_n]_{\text{补}}$, 并恢复余数和修正商。

(1) 已知 $A = 0.10010$, $B = -1$, 求 $[C]_{\text{补}} = ?$ $[2^{-5}R_5]_{\text{补}} = ?$

解 $[A]_{\text{补}} = 00.10010$, $[B]_{\text{补}} = 11.00000$, $B_0 = 1$, $[-B]_{\text{补}} = 01.00000$ 。运算过程如算式 3.19 所示。

余数和被除数				商			
$[R_0]_{\text{补}}$	00. 10010	$A_{01} \neq B_0$	0.	0	0	0	0
$\leftarrow 1$	01. 00100		0.	0	0	0	0
$+ [B]_{\text{补}}$	11. 00000						
$[R_1]_{\text{补}}$	00. 00100	$A_{01} \neq B_0$					
$\leftarrow 1$	00. 01000		0.	0	0	0	0
$+ [B]_{\text{补}}$	11. 00000						
$[R_2]_{\text{补}}$	11. 01000	$A_{01} = B_0$					
$\leftarrow 1$	10. 10000		0.	0	0	0	1
$+ [-B]_{\text{补}}$	01. 00000						
$[R_3]_{\text{补}}$	11. 10000	$A_{01} = B_0$					
$\leftarrow 1$	11. 00000		0.	0	0	1	1
$+ [-B]_{\text{补}}$	01. 00000						
$[R_4]_{\text{补}}$	00. 00000	$A_{01} \neq B_0$					
$\leftarrow 1$	00. 00000		0.	0	0	1	0
$+ [B]_{\text{补}}$	11. 00000						
	11. 00000	$A_{01} = B_0$					
		$\leftarrow 1$	0.	0	1	1	0
		商符变反	1.	0	1	1	1

算式 3.19

由于除尽, 将余数置为全 0, $[2^{-5}R_5]_{\text{补}} = 0.0000000000$ 。

同时由于除尽, $B_0 = 1$, 故需要修正商, $[C]_{\text{补}} = 1.01101 + 0.00001 = 1.01110$ 。

(2) 已知 $A = 0.10101$, $B = 0.11011$, 求 $[C]_{\text{补}} = ?$ $[2^{-5}R_5]_{\text{补}} = ?$

解 $[A]_{\text{补}} = 00.10101$, $[B]_{\text{补}} = 00.11011$, $B_0 = 0$, $[-B]_{\text{补}} = 11.00101$ 。运算过程如算式 3.20 所示。

3.20 所示。

余数和被除数			商						
$[R_0]_{\text{补}}$	00. 10101	$A_{01} = B_0$	0.	0	0	0	0	0	0
$\leftarrow 1$	01. 01010		0.	0	0	0	0	0	1
$+[-B]_{\text{补}}$	11. 00101								
$[R_1]_{\text{补}}$	00. 01111	$A_{01} = B_0$	0.	0	0	0	1	1	
$\leftarrow 1$	00. 11110		0.	0	0	0	1	1	
$+[-B]_{\text{补}}$	11. 00101								
$[R_2]_{\text{补}}$	00. 00011	$A_{01} = B_0$	0.	0	0	1	1	1	
$\leftarrow 1$	00. 00110		0.	0	0	1	1	1	
$+[-B]_{\text{补}}$	11. 00101								
$[R_3]_{\text{补}}$	11. 01011	$A_{01} \neq B_0$	0.	0	1	1	1	0	
$\leftarrow 1$	10. 10110		0.	0	1	1	1	0	
$+[B]_{\text{补}}$	00. 11011								
$[R_4]_{\text{补}}$	11. 10001	$A_{01} \neq B_0$	0.	1	1	1	0	0	
$\leftarrow 1$	11. 00010		0.	1	1	1	0	0	
$+[B]_{\text{补}}$	00. 11011								
$[R_5]_{\text{补}}$	11. 11101	$A_{01} \neq B_0$	1.	1	1	0	0	0	
		$\leftarrow 1$	0.	1	1	0	0	0	
		商符变反							

算式 3.20

由于未除尽, $C_0 = 0$, $A_{01} \neq B_0$, 故需恢复余数, $[R_5]_{\text{补}} = 11.11101 + 00.11011 = 00.11000$ 。
因此, $[2^{-5}R_5]_{\text{补}} = 0.0000011000$ 。

同时由于未除尽, $C_0 = 0$, 故不需要修正商, $[C]_{\text{补}} = 0.11000$ 。

(3) 已知 $A = -0.01011$, $B = -0.11011$, 求 $[C]_{\text{补}} = ?$ $[2^{-5}R_5]_{\text{补}} = ?$

解 $[A]_{\text{补}} = 11.10101$, $[B]_{\text{补}} = 11.00101$, $B_0 = 1$, $[-B]_{\text{补}} = 00.11011$ 。运算过程如算式 3.21 所示。

余数和被除数		商					
$[R_0]_{\text{补}}$	11. 10101	$A_{01} = B_0$	0.	0	0	0	0
$\leftarrow 1$	11. 01010		0.	0	0	0	1
$+[-B]_{\text{补}}$	00. 11011						
$[R_1]_{\text{补}}$	00. 00101	$A_{01} \neq B_0$					
$\leftarrow 1$	00. 01010		0.	0	0	0	1
$+[B]_{\text{补}}$	11. 00101						0
$[R_2]_{\text{补}}$	11. 01111	$A_{01} = B_0$					
$\leftarrow 1$	10. 11110		0.	0	0	1	0
$+[-B]_{\text{补}}$	00. 11011						
$[R_3]_{\text{补}}$	11. 11001	$A_{01} = B_0$					
$\leftarrow 1$	11. 10010		0.	0	1	0	1
$+[-B]_{\text{补}}$	00. 11011						
$[R_4]_{\text{补}}$	00. 01101	$A_{01} \neq B_0$					
$\leftarrow 1$	00. 11010		0.	1	0	1	1
$+[B]_{\text{补}}$	11. 00101						0
$[R_5]_{\text{补}}$	11. 11111	$A_{01} = B_0$					
		$\leftarrow 1$	1.	0	1	1	0
		商符变反	0.	0	1	1	0

算式 3.21

由于未除尽, $C_0 = 0$, $A_{01} = B_0$, 故不需恢复余数, 故 $[2^{-5} R_5]_{\text{补}} = 1.1111111111$ 。

同时由于未除尽, $C_0 = 0$, 故不需要修正商, $[C]_{\text{补}} = 0.01101$ 。

(4) 已知 $A = 0.10100$, $B = -0.10000$, 求 $[C]_{\text{补}} = ?$ $[2^{-5} R_5]_{\text{补}} = ?$

解 $[A]_{\text{补}} = 00.10100$, $[B]_{\text{补}} = 11.10000$, $B_0 = 1$, $[-B]_{\text{补}} = 00.10000$ 。运算过程如算式 3.22 所示。

余数和被除数		商					
$[R_0]_{\text{补}}$	00. 10100	$A_{01} \neq B_0$	0.	0	0	0	0
$\leftarrow 1$	01. 01000		0.	0	0	0	0
$+[B]_{\text{补}}$	11. 10000						
$[R_1]_{\text{补}}$	00. 11000	$A_{01} \neq B_0$					
$\leftarrow 1$	01. 10000		0.	0	0	0	0
$+[B]_{\text{补}}$	11. 10000						
$[R_2]_{\text{补}}$	01. 00000	$A_{01} = B_0$					

算式 3.22

由于 $\sum_{01} \neq \sum_{\infty}$, 表明 $[R_i]$ 发生除法溢出, 将溢出标志触发器置 1, 进行相应的中断处理。

例 3.16 两定点数相除, 如何判断商是否溢出? 若溢出应注意什么(与不溢出的正常除法比较)?

解 定点数相除, 溢出产生的原因是被除数 A 的绝对值大于等于除数 B 的绝对值, 即商 C 的绝对值大于等于 1, 溢出的结果导致商的符号位被商的整数部分修改。

对于除法运算器来说, 两数的比较是通过加减法进行的, 当第一步够减即表示除法溢出。对于 Booth 除法来说, 当某步余数的两符号位异号时, 则除法溢出。

若溢出, 则应该置位中断标志, 表示除法错误, 进行对应的中断处理。

例 3.17 迭代除法建立在什么基础之上? 为何具有生命力?

解 两数相除, 可以将被除数和除数分别看成一个分数的分子和分母。若分子分母分别乘以某一系列的数 r_i ($i = 0, 1, 2, \dots, n$), 使分母迅速收敛于 1, 则分子便以同样的速度收敛于商, 这样就可使除法转换为乘法。为了加快乘法的速度, 可以使用快速乘法来进行。

相对于乘法而言, 除法不可能一次完成多位的除法运算, 除法必须通过规则一步步进行。将除法转换为乘法后, 就可以利用快速乘法来加快运算。只要初始迭代系数 r_0 选得好, 分子将很快收敛于商, 并且具有足够的精度。另外, 使用迭代除法在运算器中就不必设计专门的除法器, 可以利用乘法器来完成除法运算。因此, 迭代除法具有生命力。

例 3.18 初始迭代系数 r_0 原则上应如何确定?

解 若取初始迭代系数 r_0 为除数 B 的近似倒数, 则一个数的近似倒数乘上这个数本身, 其结果必趋近于 1。 r_0 与 $1/B$ 越接近, 将越快收敛于预期精度。最理想的情况就是 $r_0 = 1/B$, 这样经过一次迭代就可以得到商。

例 3.19 迭代除法的执行步骤怎样进行? 以 4 次迭代为例, 图示并叙述之。

解 迭代除法的步骤为:

(1) 用求除数 B 的近似倒数法, 可以得到初始迭代系数 r_0 。

(2) 采用上一次的迭代系数乘以分母并对其求补的办法, 可以得到以后各次的迭代系数 r_i 。按照此方法可得到所有迭代系数 r_0, r_1, \dots, r_{n-1} 和 r_n 。

(3) n 次迭代系数 r_{n-1} 乘上 A_{n-1} , 即得 n 次迭代除法的近似商 $A_n = A_{n-1} \times r_{n-1}$ 。

(4) 经 n 次迭代后, 迭代精度 $(1 - \delta^{2^{n-1}})$ 已足够达到预期要求。

4 次迭代的过程和结果如表 3.7 所示。

表 3.7 迭代过程及结果

迭代次数	B_i	A_i	r_i	备注
1	$B_1 = B_0 \times r_0 = 1 - \delta$	$A_1 = A_0 \times r_0 = A_0 / B_0 (1 - \delta)$	$r_1 = 2 - B_1 = 1 + \delta$	$i = 0, \delta^{2^0} = \delta$
2	$B_2 = B_1 \times r_1 = 1 - \delta^2$	$A_2 = A_1 \times r_1 = A_0 / B_0 (1 - \delta^2)$	$r_2 = 2 - B_2 = 1 + \delta^2$	$i = 1, \delta^{2^1} = \delta^2$
3	$B_3 = B_2 \times r_2 = 1 - \delta^4$	$A_3 = A_2 \times r_2 = A_0 / B_0 (1 - \delta^4)$	$r_3 = 2 - B_3 = 1 + \delta^4$	$i = 2, \delta^{2^2} = \delta^4$
4	$B_4 = B_3 \times r_3 = 1 - \delta^8$	$A_4 = A_3 \times r_3 = A_0 / B_0 (1 - \delta^8)$	$r_4 = 2 - B_4 = 1 + \delta^8$	$i = 3, \delta^{2^3} = \delta^8$

例 3.20 按补码浮点加(减)法的运算步骤,求 $[A \pm B]_{\text{补}} = ?$

(1) 已知 $A = 2^{-001} \times (0.10100)$, $B = 2^{-010} \times (-0.011110)$, 求 $[A \pm B]_{\text{补}} = ?$

解 由于采用变形补码,故阶码和尾数均需要两位符号位,由于 A 的尾数只有 5 位,故要在其后添一位“0”使其与 B 的尾数位数相同:

$$[A]_{\text{补}} = 11111,00.101000; [B]_{\text{补}} = 11110,11.100010;$$

第一步:判 0;两数均不为 0,继续以下运算。

第二步:对阶; $[\Delta_E]_{\text{补}} = [E_A]_{\text{补}} - [E_B]_{\text{补}} = 00001$,表示 A 的阶码较大。对阶时小阶向大阶看齐,对 B 进行调整: $[B]_{\text{补}} = 11111,11.110001$ 。

第三步:尾数加减; $[M_A + M_B]_{\text{补}} = [M_A]_{\text{补}} + [M_B]_{\text{补}} = 00.011001$,需左移一位进行规格化; $[M_A - M_B]_{\text{补}} = [M_A]_{\text{补}} + [-M_B]_{\text{补}} = 00.110111$,不需规格化;

第四步:规格化;

$$[A + B]_{\text{补}} = 11111,00.011001 = 11110,00.110010;$$

$$[A - B]_{\text{补}} = 11111,00.110111.$$

(2) 已知 $A = 2^{-010} \times (-0.1010)$, $B = 2^{-010} \times (-0.01111)$, 求 $[A \pm B]_{\text{补}} = ?$

解 由于采用变形补码,故阶码和尾数均需要两位符号位,由于 A 的尾数只有 4 位,故要在其后添两位“0”使其与 B 的尾数位数相同:

$$[A]_{\text{补}} = 11110,11.011000; [B]_{\text{补}} = 11110,11.100001;$$

第一步:判 0;两数均不为 0,继续以下运算。

第二步:对阶; $[\Delta_E]_{\text{补}} = [E_A]_{\text{补}} - [E_B]_{\text{补}} = 0$,表示两数阶码已经相等,不需要调整。

第三步:尾数加减; $[M_A + M_B]_{\text{补}} = [M_A]_{\text{补}} + [M_B]_{\text{补}} = 10.111001$,需右移一位进行规格化; $[M_A - M_B]_{\text{补}} = [M_A]_{\text{补}} + [-M_B]_{\text{补}} = 11.110111$,需左移两位进行规格化;

第四步:规格化;

$$[A + B]_{\text{补}} = 11110,10.111001 = 11111,11.011101; \text{采用 } 0 \text{ 舍 } 1 \text{ 入法舍入。}$$

$$[A - B]_{\text{补}} = 11110,11.110111 = 11100,11.011100.$$

(3) 已知 $A = 2^{-011} \times (-0.0100)$, $B = 2^{-010} \times (0.010110)$, 求 $[A \pm B]_{\text{补}} = ?$

解 由于采用变形补码,故阶码和尾数均需要两位符号位,由于 A 的尾数只有 4 位,故要在其后添两位“0”使其与 B 的尾数位数相同:

$$[A]_{\text{补}} = 11101,11.110000; [B]_{\text{补}} = 11110,00.010110;$$

第一步:判 0;两数均不为 0,继续以下运算。

第二步:对阶; $[\Delta_E]_{\text{补}} = [E_A]_{\text{补}} - [E_B]_{\text{补}} = 11111$,表示 B 的阶码较大。对阶时小阶向大阶看齐,对 A 进行调整: $[A]_{\text{补}} = 11110,11.111000$ 。

第三步:尾数加减; $[M_A + M_B]_{\text{补}} = [M_A]_{\text{补}} + [M_B]_{\text{补}} = 00.001110$,需左移两位进行规格化; $[M_A - M_B]_{\text{补}} = [M_A]_{\text{补}} + [-M_B]_{\text{补}} = 11.100010$,需左移一位进行规格化;

第四步:规格化;

$$[A + B]_{\text{补}} = 11110,00.001110 = 11100,00.111000.$$

$$[A - B]_{\text{补}} = 11110,11.100010 = 11101,11.000100.$$

例 3.21 浮点加(减)法为什么要对阶? 又为什么要小阶向大阶看齐? 何时需要舍入处理? 何时判溢出? 为什么?

解 浮点数阶码不同时表示尾数位的权重不同,不能直接作加(减)法,故浮点加(减)法需要对阶。

小阶向大阶看齐就是将小阶数的尾数右移,增加阶码直至与大阶数的阶码相等,这样就不会影响到小数点标志位;否则,对齐阶码后,大阶数的尾数需左移,导致参加运算的数据不是定点小数。

在浮点加(减)法的对阶或右规时,尾数要右移,这样,尾数的末 1 位或几位可能因超出机器的允许位数而被丢掉,从而造成一定的误差。为了减小误差,通常要进行舍入处理。把为减小误差而进行的舍入处理,简称为舍入。

两数进行加(减)运算,运算结果的尾数需要右规,且结果的阶码很大,右规后使得阶码超过能够表示的最大正阶码,此时发生上溢出。

两数进行加(减)运算,运算结果的尾数需要左规,且结果的阶码很小,左规后使得阶码超过能够表示的最小负阶码,此时发生下溢出。

例 3.22 按规格化的补码浮点乘法运算步骤,求 $[A \times B]_b$ 。

(1) 已知 $A = 2^{011} \times (0.110100)$, $B = 2^{100} \times (-0.100100)$, 按规格化的补码浮点乘法运算步骤,求 $[A \times B]_b$ (设阶码为 3 位,尾数为 6 位,均不包括符号位)。

解 进行补码运算前,将阶码和尾数均采用两位符号位,将 A 和 B 表示成补码形式:
 $[A]_b = 00011, 00.110100$; $[B]_b = 00100, 11.011100$ 。

第一步:判零并置结果数符;两数均不为 0,补码浮点运算时符号位参加运算,乘积的符号运算得出。

第二步:阶码相加;乘积 C 的阶码 $[E_C]_b = [E_A]_b + [E_B]_b = 00011 + 00100 = 00111$ 。

第三步:尾数相乘;乘积 C 的尾数 $[M_C]_b = [M_A]_b \times [M_B]_b$ 。

$[M_A]_b = 00.110100$, $[M_B]_b = 1.011100$, $[-M_A]_b = 11.001100$ 。运算过程如算式 3.23 所示。

	部分积						乘数				B_{n+1}
$[P_0]_b$	00.	000000		1.	0	1	1	1	0	0	0
$[P_1]_b$	$\rightarrow 1$	00.000000		0	1.	0	1	1	1	0	0
$[P_2]_b$	$\rightarrow 1$	00.000000		0	0	1.	0	1	1	1	0
	$+[-M_A]_b$	11.001100									
		11.001100									
$[P_3]_b$	$\rightarrow 1$	11.100110		0	0	0	1.	0	1	1	1
$[P_4]_b$	$\rightarrow 1$	11.110011		0	0	0	0	1.	0	1	1
$[P_5]_b$	$\rightarrow 1$	11.111001		1	0	0	0	0	1.	0	1
	$+[M_A]_b$	00.110100									
		00.101101									
$[P_6]_b$	$\rightarrow 1$	00.010110		1	1	0	0	0	0	1.	0
	$+[-M_A]_b$	11.001100									
$[P_7]_b$		11.100010									
	$\rightarrow 1$	0	1	1	0	0	0	0	0	1.	

算式 3.23

故 $[M_C]_{\text{补}} = [M_A \times M_B]_{\text{补}} = 1.100010110000$ 。

第四步：规格化与舍入；

$[M_C]_{\text{补}}$ 需左移一位，阶码减 1。采用 0 舍 1 入法， $[C]_{\text{补}} = 00110, 11.000110$ 。

(2) 已知 $A = 2^{-011} \times (-0.000110)$, $B = 2^{101} \times (-0.011100)$, 按规格化的补码浮点乘法运算步骤，求 $[A \times B]_{\text{补}}$ (设阶码为 3 位，尾数为 6 位，均不包括符号位)。

解 进行补码运算前，将阶码和尾数均采用两位符号位，将 A 和 B 表示成补码形式，注意使尾数满足规格化浮点数据的条件： $[A]_{\text{补}} = 11010, 11.010000$; $[B]_{\text{补}} = 00100, 11.001000$ 。

第一步：判零并置结果数符；两数均不为 0，补码浮点运算时符号位参加运算，乘积的符号运算得出。

第二步：阶码相加；乘积 C 的阶码 $[E_C]_{\text{补}} = [E_A]_{\text{补}} + [E_B]_{\text{补}} = 11010 + 00100 = 11110$ 。

第三步：尾数相乘；乘积 C 的尾数 $[M_C]_{\text{补}} = [M_A]_{\text{补}} \times [M_B]_{\text{补}}$ 。

$[M_A]_{\text{补}} = 11.010000$, $[M_B]_{\text{补}} = 1.001000$, $[-M_A]_{\text{补}} = 00.110000$ 。运算过程如算式 3.24 所示。

	部分积	乘数						B_{n+1}
$[P_0]_{\text{补}}$	00. 000000	1.	0	0	1	0	0	0 0
$[P_1]_{\text{补}}$	→1 00. 000000	0	1.	0	0	1	0	0 0
$[P_2]_{\text{补}}$	→1 00. 000000	0	0	1.	0	0	1	0 0
$[P_3]_{\text{补}}$	→1 00. 000000	0	0	0	1.	0	0	1 0
	<u>$+[-M_A]_{\text{补}}$</u>	00.	110000					
$[P_4]_{\text{补}}$	→1 00. 011000	0	0	0	0	1.	0	0 1
	<u>$+[M_A]_{\text{补}}$</u>	11.	010000					
$[P_5]_{\text{补}}$	→1 11. 110100	0	0	0	0	0	1.	0 0
$[P_6]_{\text{补}}$	→1 11. 111010	0	0	0	0	0	0	1. 0
	<u>$+[-M_A]_{\text{补}}$</u>	00.	110000					
$[P_7]_{\text{补}}$	00. 101010							

故 $[M_C]_补 = [M_A \times M_B]_补 = 0.101010000000$ 。

第四步：规格化与舍入；

$[M_C]_补$ 不需规格化。采用 0 舍 1 入法， $[C]_补 = 11110, 0.101010$ 。

例 3.23 浮点乘法应注意哪些问题？何时判下溢和上溢？为什么？

解 浮点乘法是将两个规格化的浮点数相乘。规格化浮点乘法运算可分为四步：判零并置结果数符，阶码相加，尾数相乘，规格化、判溢出与舍入。在浮点乘法过程中应注意判溢出、结果规格化与舍入等问题。

规格化时判下溢，既不会扩大溢出范围，也不会错判成上溢了。下溢条件为：

$$R_{ES} \cdot (\overline{\sum}_{ES} \sum_{ED} + \sum_{ES} \overline{\sum}_{ED}) = R_{ES} \cdot (\sum_{ES} \oplus \sum_{ED})$$

为了不扩大上溢范围，应在规格化时判上溢。上溢条件为：

$$\bar{R}_{ES} \cdot (\overline{\sum}_{ES} \sum_{ED} + \sum_{ES} \overline{\sum}_{ED}) = \bar{R}_{ES} \cdot (\sum_{ES} \oplus \sum_{ED})$$

实际上，只会出现 $\bar{R}_{ES} \sum_{ES} \overline{\sum}_{ED}$ 的情况，而不会出现 $\bar{R}_{ES} \sum_{ES} \sum_{ED}$ 的情况，只是为使判上

溢和判下溢统一用 $(\sum_{ES} \oplus \sum_{ED})$ 逻辑而已。 $\bar{R}_{ES} \sum_{ES} \overline{\sum}_{ED}$ 不会出现，并不影响逻辑的正确性。

结果规格化时，因而乘积的尾数的绝对值必大于等于 $1/4$ ，所以乘法的左规，最多只需左规 1 位。由于补码 $[-1]_补$ 是有意义的，当两数尾数都为 $[-1]_补$ 时，尾数相乘后为 $01.0\cdots 0$ （即 $+1$ ），尾符出现 01 的情况，因此右规时也只能是 1 位。

乘法结果在规格化的时候并不会产生舍入问题。如不取双倍字长的乘积，则有舍入问题。通常采用的舍入法，是 0 舍 1 入法。

例 3.24 试述补码浮点乘法的运算步骤？若尾数用原码表示，阶码用移码表示，运算步骤上有什么区别？

解 补码乘法步骤：判零并置结果数符，阶码相加，尾数相乘，规格化、判溢出与舍入。

若尾数用原码表示，阶码用移码表示，运算步骤上的区别主要来自不同的码制导致的运算步骤的不同。

补码运算判 0 后并不置结果数符，因为补码的符号位是参加运算的。如果尾数用原码表示，则判 0 后需置结果数符，因为原码运算时符号位单独处理。

若阶码用移码表示，则运算过程中判溢出的条件会发生变化。

判是否为规格化浮点数的条件也发生了变化，对于原码来说，只要小数点后一位为 1 则为规格化的数。由于 $[-1]_原$ 是没有意义的，故乘法结果不会出现类似补码乘法那样需要右规的情况。

例 3.25 按规格化补码浮点除法规则，求 $[A \div B]_补$ 。

(1) 已知 $A = 2^{-2} \times (13/32)$, $B = 2^3 \times (15/16)$, 按规格化补码浮点除法规则，求 $[A \div B]_补$ （设阶码为 3 位，尾数为 6 位，均不包括符号位）。

解 首先取两位符号位，将 A 和 B 表示成规格化浮点数的补码表示形式：

$$[A]_{\text{补}} = 11101,00.110100; [B]_{\text{补}} = 00011,00.111100。$$

第一步: 判零并置商符; 两数均不为 0, 补码浮点运算时符号位参加运算, 商的符号运算得出。

第二步: 尾数调整; 由于 $|M_A| < |M_B|$, 故不需要调整尾数。

第三步: 阶码相减; 商 C 的阶码 $[E_C]_{\text{补}} = [E_A]_{\text{补}} + [-E_B]_{\text{补}} = 11101 + 11101 = 11010$ 。

第四步: 尾数相除。

$[M_A]_{\text{补}} = 00.110100, [M_B]_{\text{补}} = 00.111100, B_0 = 0, [-M_B]_{\text{补}} = 11.000100$ 。运算过程如算式 3.25 所示。

		余数和被除数		商				
$[R_0]_{\text{补}}$	00.110100	$A_{01} = B_0$	0.	0	0	0	0	0
$\leftarrow 1$	01.101000		0.	0	0	0	0	0
$+[-M_B]_{\text{补}}$	11.000100							1
$[R_1]_{\text{补}}$	00.101100	$A_{01} = B_0$						
$\leftarrow 1$	01.011000		0.	0	0	0	0	1
$+[-M_B]_{\text{补}}$	11.000100							1
$[R_2]_{\text{补}}$	00.011100	$A_{01} = B_0$						
$\leftarrow 1$	00.111000		0.	0	0	0	1	1
$+[-M_B]_{\text{补}}$	11.000100							1
$[R_3]_{\text{补}}$	11.111100	$A_{01} \neq B_0$						
$\leftarrow 1$	11.111000		0.	0	0	1	1	1
$+[-M_B]_{\text{补}}$	00.111100							0
$[R_4]_{\text{补}}$	00.110100	$A_{01} = B_0$						
$\leftarrow 1$	01.101000		0.	0	1	1	1	0
$+[-M_B]_{\text{补}}$	11.000100							1
$[R_5]_{\text{补}}$	00.101100	$A_{01} = B_0$						
$\leftarrow 1$	01.011000		0.	1	1	1	0	1
$+[-M_B]_{\text{补}}$	11.000100							1
$[R_6]_{\text{补}}$	00.011100	$A_{01} = B_0$						
$\leftarrow 1$	1.		1	1	0	1	1	1
商符变反		0.	1	1	0	1	1	1

算式 3.25

由于未除尽, $C_0 = 0, A_{01} = B_0$, 故不需恢复余数, $[2^{-6}R_6]_{\text{补}} = 0.011100 \times 2^{-6}$ 。

同时由于未除尽, $C_0 = 0$, 故不需要修正商, $[M_C]_{\text{补}} = 0.110111$ 。

(2) 已知 $A = 2^3 \times (-11/16)$, $B = 2^5 \times (15/16)$, 按规格化补码浮点除法规则, 求 $[A \div B]_{\text{补}}$ (设阶码为 3 位, 尾数为 6 位, 均不包括符号位)。

解 首先取两位符号位, 将 A 和 B 表示成规格化浮点数的补码表示形式:

$$[A]_{\text{补}} = 00011,11.010100; [B]_{\text{补}} = 00101,00.111100。$$

第一步: 判零并置商符; 两数均不为 0, 补码浮点运算时符号位参加运算, 商的符号运

算得出。

第二步:尾数调整;由于 $|M_A| < |M_B|$,故不需要调整尾数。

第三步:阶码相减;商 C 的阶码 $[E_C]_B = [E_A]_B + [-E_B]_B = 00011 + 11011 = 11110$ 。

第四步:尾数相除。

$[M_A]_B = 11.010100, [M_B]_B = 00.111100, B_0 = 0, [-M_B]_B = 11.000100$ 。运算过程如算式 3.26 所示。

	余数和被除数		商
$[R_0]_B$	11. 010100	$A_{01} \neq B_0$	0. 0 0 0 0 0 0 0 0
$\leftarrow 1$	10. 101000		0. 0 0 0 0 0 0 0 0
$+ [M_B]_B$	00. 111100		
$[R_1]_B$	11. 100100	$A_{01} \neq B_0$	0. 0 0 0 0 0 0 0 0
$\leftarrow 1$	11. 001000		0. 0 0 0 0 0 0 0 0
$+ [M_B]_B$	00. 111100		
$[R_2]_B$	00. 000100	$A_{01} = B_0$	0. 0 0 0 0 0 0 0 1
$\leftarrow 1$	00. 001000		0. 0 0 0 0 0 0 0 1
$+ [-M_B]_B$	11. 000100		
$[R_3]_B$	11. 001100	$A_{01} \neq B_0$	0. 0 0 0 0 0 1 0 0
$\leftarrow 1$	10. 011000		0. 0 0 0 0 0 1 0 0
$+ [M_B]_B$	00. 111100		
$[R_4]_B$	11. 010100	$A_{01} \neq B_0$	0. 0 0 0 1 0 0 0 0
$\leftarrow 1$	10. 101000		0. 0 0 0 1 0 0 0 0
$+ [M_B]_B$	00. 111100		
$[R_5]_B$	11. 100100	$A_{01} \neq B_0$	0. 0 0 1 0 0 0 0 0
$\leftarrow 1$	11. 001000		0. 0 0 1 0 0 0 0 0
$+ [M_B]_B$	00. 111100		
$[R_6]_B$	00. 000100	$A_{01} = B_0$	0. 0 1 0 0 0 0 0 1
		$\leftarrow 1$	1. 0 1 0 0 0 0 0 1
		商符变反	

算式 3.26

由于未除尽, $C_0 = 1, A_{01} = B_0$, 故需恢复余数, $[R_6]_B = 00.000100 + 11.000100 = 11.001000$ 。故 $[2^{-6}R_6]_B = 1.001000 \times 2^{-6}$ 。

同时由于未除尽, $C_0 = 1$, 故需要修正商, $[M_C]_B = 1.010001 + 0.000001 = 1.010010$ 。

(3) 已知 $A = 2^{-6} \times (-1), B = 2^{-7} \times (1/2)$, 按规格化补码浮点除法规则, 求 $[A \div B]_B$ (设阶码为 3 位, 尾数为 6 位, 均不包括符号位)。

解 首先取两位符号位, 将 A 和 B 表示成规格化浮点数的补码表示形式:

$$[A]_B = 11010, 11.000000; [B]_B = 11001, 00.100000.$$

第一步: 判零并置商符; 两数均不为 0, 补码浮点运算时符号位参加运算, 商的符号运算得出。

第二步：尾数调整；由于 $|M_A| > |M_B|$ ，故需要调整尾数。将 A 调整为 $2^{-4} \times (-1/4)$ ，此时， $|M_A| < |M_B|$ ， $[A]_b = 11100, 11.110000$ 。

第三步：阶码相减；商 C 的阶码 $[E_C]_b = [E_A]_b + [-E_B]_b = 11100 + 00111 = 00011$ 。

第四步：尾数相除。

$[M_A]_b = 11.110000, [M_B]_b = 00.100000, B_0 = 0, [-M_B]_b = 11.100000$ 。运算过程如算式 3.27 所示。

余数和被除数				商				
$[R_0]_b$	11. 110000	$A_{01} \neq B_0$	0.	0	0	0	0	0
$\leftarrow 1$	11. 100000		0.	0	0	0	0	0
$+[-M_B]_b$	00. 100000							
$[R_1]_b$	00. 000000	$A_{01} = B_0$						
$\leftarrow 1$	00. 000000		0.	0	0	0	0	1
$+[-M_B]_b$	11. 100000							
$[R_2]_b$	11. 100000	$A_{01} \neq B_0$						
$\leftarrow 1$	11. 000000		0.	0	0	0	0	1
$+[-M_B]_b$	00. 100000							
$[R_3]_b$	11. 100000	$A_{01} \neq B_0$						
$\leftarrow 1$	11. 000000		0.	0	0	0	1	0
$+[-M_B]_b$	00. 100000							
$[R_4]_b$	11. 100000	$A_{01} \neq B_0$						
$\leftarrow 1$	11. 000000		0.	0	0	1	0	0
$+[-M_B]_b$	00. 100000							
$[R_5]_b$	11. 100000	$A_{01} \neq B_0$						
$\leftarrow 1$	11. 000000		0.	0	1	0	0	0
$+[-M_B]_b$	00. 100000							
$[R_6]_b$	11. 100000	$A_{01} \neq B_0$						
	$\leftarrow 1$	0.	1	0	0	0	0	0
	商符变反	1.	1	0	0	0	0	0

算式 3.27

由于第一步除尽，故需余数 $[2^{-6} R_6]_b = 0$ 。

同时由于除尽， $B_0 = 0$ ，故不需要修正商， $[M_C]_b = 1.100000$ 。

需要说明的是，为了方便机器判断是否为规格化数，通常不将 $1.10\cdots 0(-0.5)$ 认为是规格化的数，而把 $1.0\cdots 0(-1)$ 当成规格化的数。因此，需要对商进行左规，即 $[M_C]_b = 1.000000; [E_C]_b = 00010$ 。

(4) 已知 $A = 2^{-7} \times (-1)$, $B = 2^1 \times (-1)$, 按规格化补码浮点除法规则，求 $[A \div B]_b$ (设阶码为 3 位，尾数为 6 位，均不包括符号位)。

解 首先取两位符号位，将 A 和 B 表示成规格化浮点数的补码表示形式：

$$[A]_b = 11001, 11.000000; [B]_b = 00001, 11.000000.$$

第一步:判零并置商符;两数均不为0,补码浮点运算时符号位参加运算,商的符号运算得出。

第二步:尾数调整;由于 $|M_A| = |M_B|$,故需要调整尾数。将A调整为 $2^{-6} \times (-1/2)$,此时, $|M_A| < |M_B|$, $[A]_{\text{补}} = 11010.11.100000$ 。

第三步:阶码相减;商C的阶码 $[E_C]_{\text{补}} = [E_A]_{\text{补}} + [-E_B]_{\text{补}} = 11010 + 11111 = 11001$ 。

第四步:尾数相除。

$[M_A]_{\text{补}} = 11.100000$, $[M_B]_{\text{补}} = 11.000000$, $B_0 = 1$, $[-M_B]_{\text{补}} = 01.000000$ 。运算过程如算式3.28所示。

余数的被除数			商							
$[R_0]_{\text{补}}$	11. 100000	$A_{01} = B_0$	0.	0	0	0	0	0	0	0
$\leftarrow 1$	11. 000000		0.	0	0	0	0	0	0	1
$+[-M_B]_{\text{补}}$	01. 000000									
$[R_1]_{\text{补}}$	00. 000000	$A_{01} \neq B_0$								
$\leftarrow 1$	00. 000000		0.	0	0	0	0	1	0	0
$+[M_B]_{\text{补}}$	11. 000000									
$[R_2]_{\text{补}}$	11. 000000	$A_{01} = B_0$								
$\leftarrow 1$	10. 000000		0.	0	0	0	1	0	1	1
$+[-M_B]_{\text{补}}$	01. 000000									
$[R_3]_{\text{补}}$	11. 000000	$A_{01} = B_0$								
$\leftarrow 1$	10. 000000		0.	0	0	1	0	1	1	1
$+[-M_B]_{\text{补}}$	01. 000000									
$[R_4]_{\text{补}}$	11. 000000	$A_{01} = B_0$								
$\leftarrow 1$	10. 000000		0.	0	1	0	1	1	1	1
$+[-M_B]_{\text{补}}$	01. 000000									
$[R_5]_{\text{补}}$	11. 000000	$A_{01} = B_0$								
$\leftarrow 1$	10. 000000		0.	1	0	1	1	1	1	1
$+[-M_B]_{\text{补}}$	01. 000000									
$[R_6]_{\text{补}}$	11. 000000	$A_{01} = B_0$								
商符变反										
	1.	0	1	1	1	1	1	1	1	
	0.	0	1	1	1	1	1	1	1	

算式 3.28

由于除尽,故需余数 $[2^{-6}R_6]_{\text{补}} = 0$ 。

同时由于除尽, $B_0 = 1$,故需要修正商, $[M_C]_{\text{补}} = 0.011111 + 0.000001 = 0.100000$ 。

例 3.26 试述浮点除法的运算步骤。为何除法进行前要经过预置和尾数调整两个步骤?

解 规格化浮点除法运算可分为四步:判零并置商符,尾数调整,阶码相减,尾数相除。通常又把判断除数是否为零,被除数是否为零(当除数不为零时)的过程叫做预置。

预置的目的是为了保证除法的简单性和合法性:若被除数为零而除数不为零则无需

运算；若除数为零则除法为非法，应置非法操作标志进行中断处理。

尾数调整是为了使定点除法运算规则对浮点尾数的除法也适用，即确保商符不破坏。同时，尾数调整还有一个好处，就是可以确保商为规格化的浮点数。

例 3.27 用流程图分别描述浮点加减运算、浮点乘法运算和浮点除法运算流程。

解 规格化浮点加(减)运算可分为四步：判零，对阶，求和与规格化。浮点加(减)运算的流程图如图 3.12 所示。

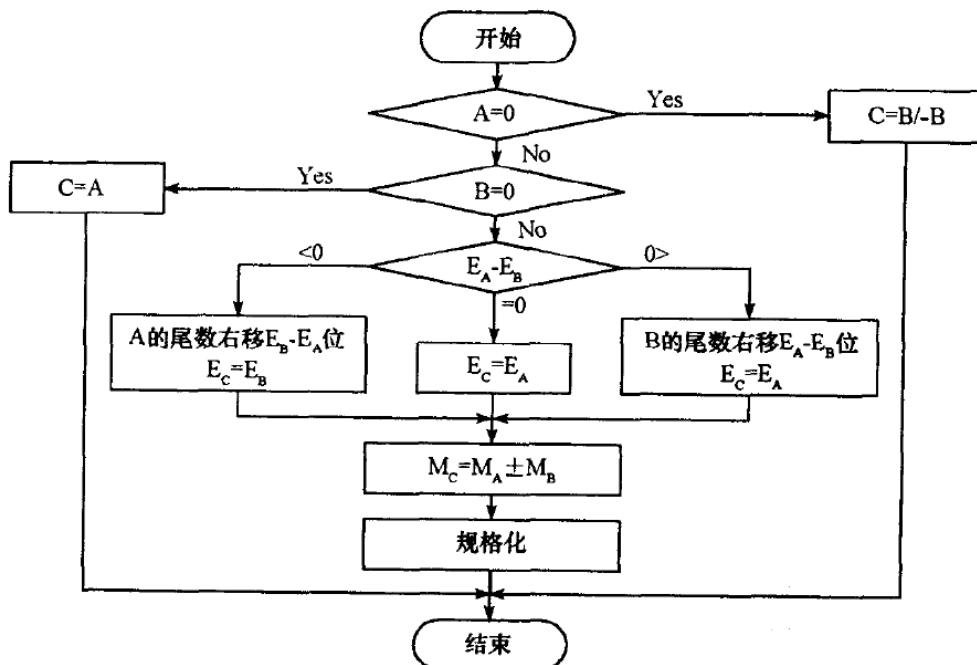


图 3.12 浮点加减法运算流程图

规格化浮点乘法运算可分为四步：判零并置结果数符，阶码相加，尾数相乘，规格化、判溢出与舍入。浮点乘法运算的流程图如图 3.13 所示。

规格化浮点除法运算可分为四步：判零并置商符，尾数调整，阶码相减，尾数相除。浮点除法运算的流程图如图 3.14 所示。

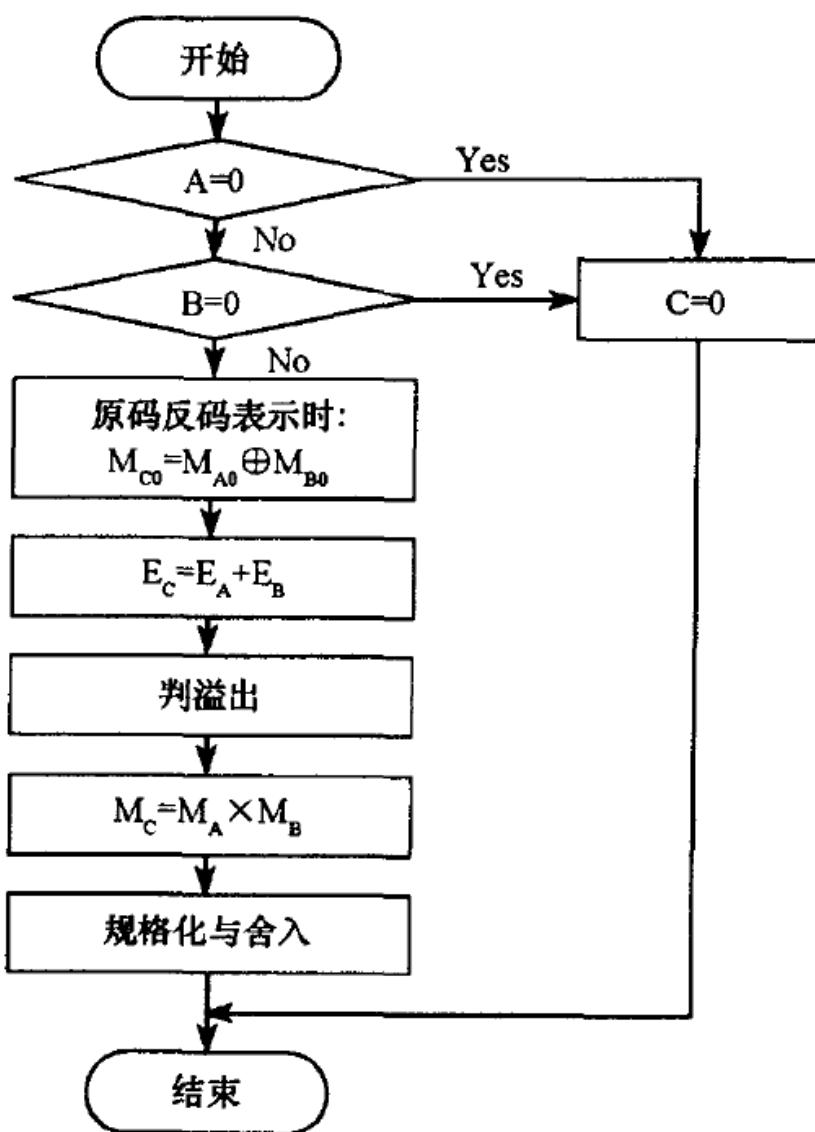


图 3.13 浮点乘法运算流程图

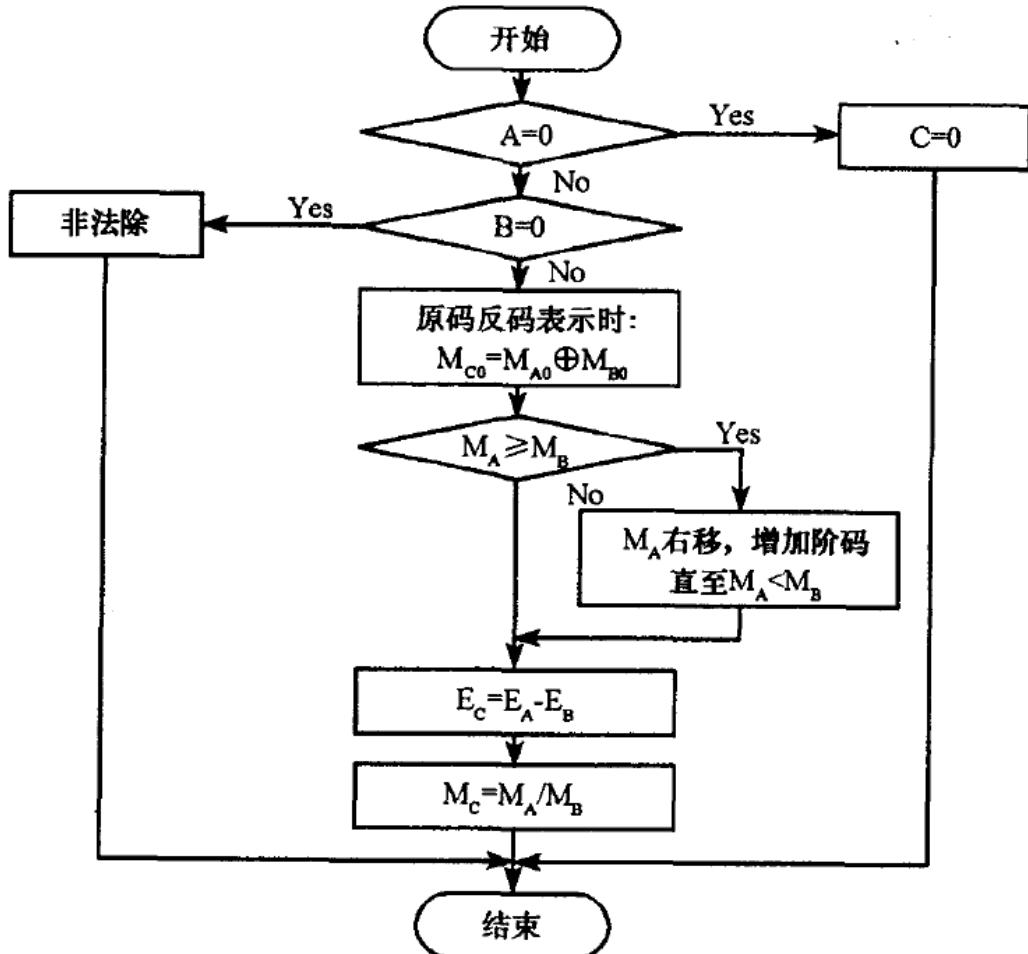


图 3.14 浮点除法运算流程图

例 3.28 乘法和除法浮点运算均可由软件实现,软硬件实现各有何优缺点?

解 浮点乘法运算和除法运算可以分解为定点阶码的加减运算和定点尾数的乘法和除法运算,而定点乘法运算和除法运算均可以分解为加减运算和移位运算。因此,浮点乘法和除法运算可以使用软件的方法将其分解为阶码的加减运算、尾数的加减运算和移位运算。

硬件实现方法的优缺点:计算速度比较快,有较好的系统性能,但硬件实现不够灵活,整机代价较高。

软件实现方法的优缺点:实现较灵活,整机代价较小,但计算速度较慢,系统性能不高。

例 3.29 试就数的范围、精度、程序设计、硬件设计等几方面对定点运算和浮点运算进行比较,说明它们的优缺点。

解 定点数的表示范围比浮点数的表示要窄得多,还要有选择、修改和变更比例因素等问题。在要求表示的数据范围较大时往往采用浮点运算表示。

由于浮点数据表示分为尾数和阶码两部分,运算时需分别对它们进行处理,所以浮点运算比较复杂,完成浮点运算的时间较定点运算的时间长,浮点数的格式也较定点数复杂。

从精度上来看,浮点数一般比定点数的精度要高。

从降低代价和复杂度考虑,计算机浮点运算功能可通过软件实现,目前由于硬件工业的发展,大部分的计算机处理器都具有浮点运算功能。

在浮点运算程序设计时,如果计算机具有浮点数据表示,则浮点运算指令可直接执行;如果计算机只具有定点数据表示,则浮点运算指令需要使用软件方法,通过多条定点运算指令模拟实现。

例 3.30 试比较单总线、双总线、三总线结构运算器的优缺点。

解 单总线结构的运算器在同一时间内,只能有一个操作数放在总线上。这种结构的操作速度较慢,但控制比较简单。

双总线结构的运算器可以并行输入两个操作数,速度较快,但控制需分成两步并需设置输入缓冲器。

三总线结构的运算器的输入输出分别与三个总线连接,运算可在一步完成,操作速度较快,但控制比前两种方式都复杂。

第四章 控制器

例 4.1 解释下列术语:指令流,数据流,控制流,控制寄存器,微操作,节拍,中央控制,局部控制,微命令,微指令,微指令周期。

解 指令流:计算机执行的指令序列称为指令流。

数据流:根据指令要求依次访问的数据序列称为数据流。

控制流:由控制器发出的控制信号序列称为控制流。

控制寄存器:组成控制器组成部分的一些基本寄存器。

微操作:相对指令完成的功能而言,一个部件能够完成的基本操作,称为微操作。

节拍:在同步控制方式下,根据系统时钟信号,以系统时钟周期为基本单位,将指令周期划分为若干个相等的时间段,每个时间段称为一个节拍。

中央控制:是一种适合于计算机中所有指令的处理方式或时序分配方式,使所有指令都在统一的时序下进行处理,所有指令的指令周期都是相同的。

局部控制:每条指令都有独立的处理方式或时序分配方式,由指令启动各自的时序进行处理,每一条指令的指令周期取决于它的微操作序列长度。

微命令:构成控制信号序列的最小单位。

微指令:一组实现一定操作功能的微命令的组合。通常用二进制编码表示。

微指令周期:是指从控制存储器读一条微指令并执行完相应的微操作所需的时间。

例 4.2 试述控制器基本功能。

解 简单地说,控制器的基本功能为在时空上对数据流和指令流实施正确的控制。

空间上,由控制器形成受控部件的控制信号;时间上,控制器控制各种动作的执行顺序。

例 4.3 控制器有哪几种控制方式?各有何特点?

解 控制器通常分为异步控制方式和同步控制方式两种实现方法。

异步控制中,由于各部件之间没有统一的时钟,故微操作信号的宽度不确定,根据需

要来确定。这使得设计比较复杂,消耗器材多,系统调试难度大,可靠性不易保证。

同步控制中,微操作与机器时钟信号同步,使得控制简单,信号宽度固定,但必须保证最耗时的微操作能够完成。同步控制方式的设计简单,节省器材,便于调试,可靠性好。

例 4.4 试述指令周期、时钟周期、存储周期三者的关系。

解 指令周期指从取指令开始到指令执行完成所需要的时间。时钟周期就是机器产生的相邻两个时钟脉冲间的间隔时间,是机器主频的倒数。存储周期指的是存储器系统的工作周期,即两次存储器访问的最长时间间隔。它们三者的关系是:指令周期由多个时钟周期构成,并且通常是存储周期的整数倍。

例 4.5 试述节拍、节拍电位、节拍电位周期、时钟周期和指令周期的关系。

解 在同步控制方式下,根据系统时钟信号,以系统时钟周期为基本单位,将指令周期划分为若干个相等的时间段,每个时间段称为一个节拍。节拍一般用具有一定宽度的电位信号表示,称为节拍电位。在计算机系统中,节拍电位通常是具有周期性的,这个周期称为节拍电位周期。时钟周期是主频的倒数,时钟周期与节拍电位宽度相同。指令周期指从取指令开始到指令执行完成所需要的时间,指令周期和节拍电位周期相同。

例 4.6 指令周期内节拍划分的原则是什么?

解 以时钟周期为单位,将指令周期划分为若干相等的节拍。除了访存微操作以外,一般的微操作需要在一个节拍中完成。

例 4.7 实例计算机中为什么要使用 \bar{T}_3 信号?

解 实例计算机在第 4 拍使用“ $IF \cdot \bar{T}_3$ ”而不是“ $IF \cdot T_3$ ”。如果第 4 拍使用“ $IF \cdot T_3$ ”,则运算器不能正常工作。原因是“ $IF \cdot T_3$ ”节拍上升沿将执行“清 IF”和“置 EX”微操作,使 IF 触发器变成 0 状态,EX 触发器变成 1 状态。此时节拍信号“ $IF \cdot T_3$ ”就不能继续维持高电平,故不是一个完整的节拍,安排在第 4 拍中的微操作就可能不能完成而出现错误。另外,由于 EX 变成高电平,将使得“ $EX \cdot T_3$ ”有效,本来安排在第 8 拍中的微操作就可能提前执行。因此,需要在第 4 拍结束时,使用 $IF \cdot \bar{T}_3$ 的上升沿(对应 T_3 的下降沿)完成“清 IF”和“置 EX”微操作。同理,在第 8 拍结束时,使用 $EX \cdot \bar{T}_3$ 的上升沿(对应 T_3 的下降沿)完成“清 EX”和“置 IF”微操作。注意,“ $IF \cdot \bar{T}_3$ ”仍然算做第 4 个节拍,而“ $EX \cdot \bar{T}_3$ ”仍然算做第 8 个节拍。

例 4.8 结合实例计算机,试述局部控制和中央控制是如何相互转换的?

解 在混合控制中,为实现中央控制和局部控制的转换,需要设置一个移位触发器 SR(Shift Register)作为标志。SR 被置位,表示指令的执行进入局部控制时序。当指令字中 IR_1 与 IR_0 均为 0,或者移位计数器 SC 计数结束时,SR 被复位,表示指令的执行从局部控制时序回到中央控制时序。中央控制与局部控制转换的原理如图 4.10 所示。

当需要进入局部控制时,由中央微操作控制部件发出控制信号加到门 3 的输入端, ID 给出移位指令的译码信号也加到门 3 的输入端,门 3 的输出置 0 中央控制工作触发器 CC,停止中央控制;置 1 局部控制工作触发器 LC,启动局部控制。此时,时钟信号停止送往中央控制线路并转向送到局部控制线路 LTSG。由 LTSG 发出局部节拍信号 T_{L_i} 。 T_{L_i} 送到局部微操作控制部件,在 ID 控制下产生移位信号。当移位完成时,由局部微操作控制

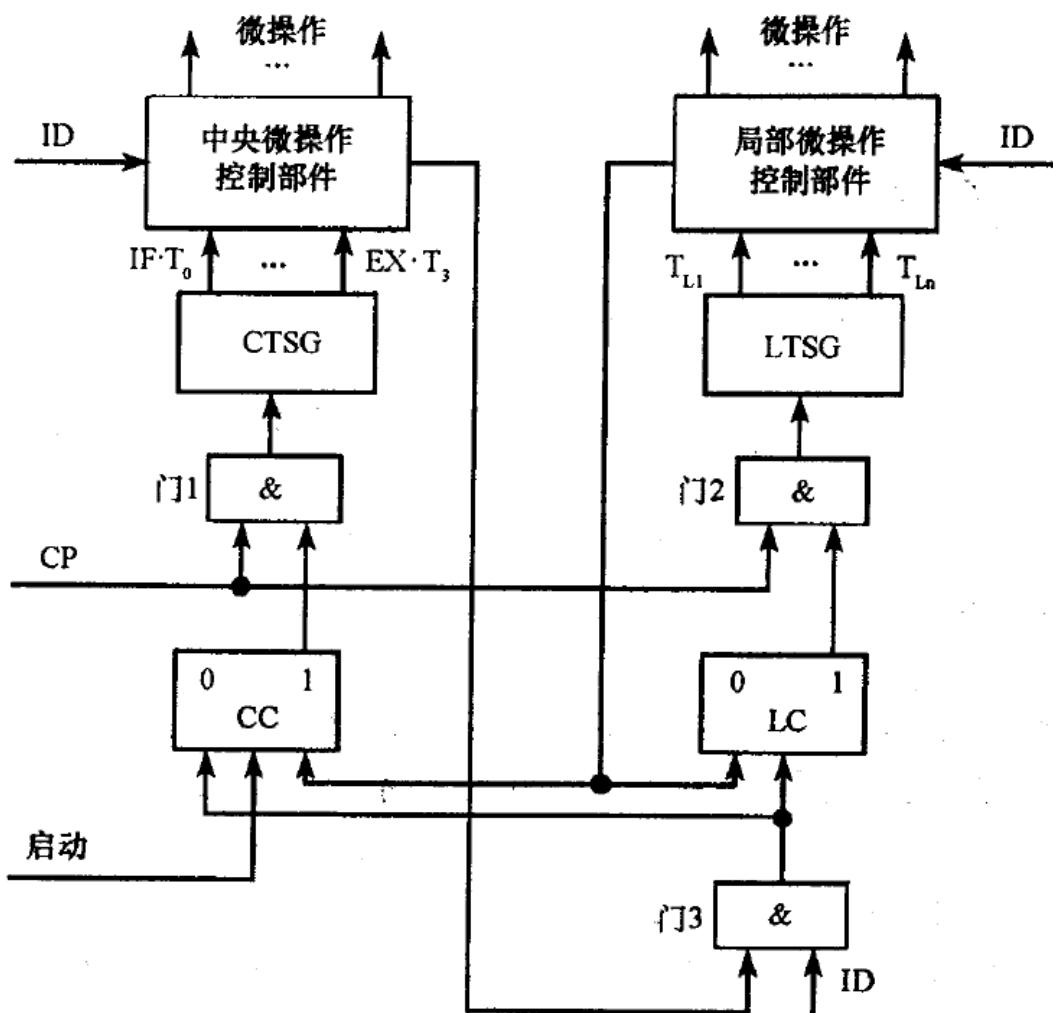


图 4.10 中央控制与局部控制转换

部件发出控制信号,置 0 触发器 LC,停止局部控制;置 1 触发器 CC,重新启动中央控制。

例 4.9 设计组合逻辑控制器的步骤有哪些?

解 (1)明确设计环境,包括指令系统、主存的存储周期、时钟周期、部件的结构等。

(2)分析指令流程,根据指令功能要求和各部件间的数据通路,为各种指令的微操作序列选定适当的节拍信号。

(3)编制指令操作时间表,不断安排、调整与完善微操作的时序、控制器的结构和数据通路的设置。

(4)形成微操作表,便于清晰、有条理地进行微操作信号的综合。

(5)进行微操作的逻辑综合,形成每个微操作对应的逻辑表达式以及微操作控制部件的逻辑图。

(6)产生各个微操作的逻辑线路。

例 4.10 试述机器指令与微指令、主存与控存、程序与微程序、指令周期与微指令周期的异同。

解 机器指令是提供给机器指令用户编程的基本单位,表示机器能够完成的一项最基本的操作。微指令则是为了实现机器指令操作的一系列微命令的组合。一条机器指令对应若干条微指令组成的微程序。

主存存放的是系统程序和用户程序,容量很大。控存中存放的是对应于机器指令系统的微程序,实现机器指令系统,容量有限。

程序是用户为完成某项任务编制的,由机器指令构成,并放在主存里。微程序是用于描述机器指令的,一般是预先编制好,存放在控存 CM 中,不允许用户修改。

指令周期是指从取指令开始到指令执行完成所需要的时间。微指令周期通常是指从控制存储器读一条微指令并执行完相应的微操作所需的时间。微指令周期和指令周期之间的差别是,微指令周期的时间一般是固定的,指令周期往往是可以变化的。

例 4.11 微指令编码有哪几种常用方法?它们各有何特点?

解 微指令的编码方法有:直接控制编码、最短字长编码、分段直接编码、分段间接编码和常数源字段等。

直接控制编码的特点是:控制简单直观,操作并行性最好,可以提高速度,但指令字太长,CM 的容量过大且指令字利用效率低。

最短字长编码的特点是:指令字长短,但执行效率低。

分段直接编码的特点是:吸收了直接控制和最短字长编码方法的优点,既能缩短字长,又有较高并行性,执行速度快。

分段间接编码的特点是:一个控制字段的某些微操作需要另外一个解释字段来解释才能确定,一个解释字段要同时解释多个控制字段,才能有效地缩短字长。

常数源字段 E 的特点是:可以减少微指令字长和增加微指令的灵活性。

例 4.12 怎样确定多分支转移、微程序循环、微子程序的后继微地址?

解 对于顺序—转移型的后继微地址的形成,可以通过设置转移控制字段 BCF 和转移地址字段 BAF 的值来实现控制。BAF 用于给出转移用微地址,BCF 用于规定是顺序执行还是转移。顺序执行时,后继微指令地址由 μ PC 加上一个增量来产生。无论对于多分支转移、微程序循环和微子程序,条件成立时的后继微地址由 μ PC 和 BAF 组合实现。这些不同的操作由不同的转移控制字段 BCF 来标志。

对于断定型的后继微地址的形成,可以通过设置测试控制字段 TCF 来实现,不同的测试字段对应不同的测试地址 TF。对于多分支转移、微程序循环和微子程序,条件成立时的后继微地址可以由测试控制字段 TCF 产生的测试地址 TF 得到。

例 4.13 试述顺序—转移型和断定型微地址的确定方法。

解 顺序—转移型的后继微地址的形成分顺序执行和转移执行两种情况,顺序执行时后继微指令地址由 μ PC 加一个增量(通常为 1)产生,转移时后继微指令地址由微指令的转移地址字段 BAF 指示的转移地址来源给出。转移地址的来源有以下三种:由 BAF 确定的地址,机器指令所对应的微程序的人口地址,微子程序人口地址和返回地址。

断定型的后继微地址可由微程序设计者直接指定,或者由微程序设计者指定的测试判别字段产生。微指令中非测试字段 HF 直接生成微地址码中的非测试地址 HF,测试控制字段 TCF 指出产生测试地址的测试条件,根据测试结果形成微地址码中的测试地址 TF。

例 4.14 试述水平型微指令格式与垂直型微指令格式特点，并比较其优缺点。

解 采用水平型微指令格式，微指令的微指令字较长，微指令中的微操作有高度的并行性，并且微指令译码简单。水平型微指令格式的优点是：并行操作能力强，效率高；编制的微程序比较短；微程序的执行速度比较快；控制存储器的纵向容量小，灵活性强。水平型微指令格式的缺点是：微指令字比较长；水平型微指令与机器指令差别很大，进行微程序设计困难；源微程序编译成水平型微指令的编码比较复杂。

采用垂直型微指令格式，微指令的微指令字短，微指令的并行微操作能力有限，并且微指令译码比较复杂。垂直型微指令格式的优点是：便于用户编制微程序；编制的微程序规整、直观，便于实现设计的自动化；微指令较短使控制存储器的横向空量少。垂直型微指令格式的缺点是：微程序较长；要求控制存储器的纵向容量大；微程序执行速度慢；如果数据通路具有多种并行操作能力，则不能充分利用这种并行能力。

例 4.15 试述组合逻辑控制器和微程序控制器的优缺点。

解 组合逻辑控制器的优点是：速度快、效率高，使用广泛。组合逻辑控制器的缺点是：系统复杂时，设计复杂性较大，控制电路的设计质量难以保证；复杂控制器的设计难以形式化，设计效率低。

微程序控制器的优点是：规整、灵活、可维护性好，基本设计思想简单，在廉价的和 16 位以下的微处理器中广泛使用，是可编程控制器的核心部件。微程序控制器的缺点是：速度慢，几乎所有指令处理的速度都一样。

第五章 存储器

例 5.1 解释下列术语:存储位元(Memory Cell),存储单元(Memory Location),挥发性存储器,非挥发性存储器, RAM, ROM, CAM, 存储周期, 主存存取时间, DRAM, SRAM, 刷新周期, 刷新操作周期。

解 (1)存储位元(Memory Cell):记录一位二进制信息的存储介质区域或存储元器件。

(2)存储单元(Memory Location):存放一个机器字或一个字节且具有唯一地址的存储场所。

(3)挥发性存储器:断电后信息丢失的存储器。如 RAM。

(4)非挥发性存储器:断电后信息不丢失的存储器。如 ROM。

(5)RAM:随机存取存储器(Random Access Memory)。所谓随机存取指的是对存储器的任何存储单元都可随时存取且存取所需时间都是相同的,与存储单元所处的物理位置无关。

(6)ROM:只读存储器(Read Only Memory)。即只能随机读出、不能随机写入信息的随机存储器。ROM 中的信息是在工作前事先写入的,一旦写入便可长期保存。

(7)CAM:按内容寻址存储器(Content Addressed Memory)亦称相联存储器。不是依地址而是依所存信息的全部或部分内容进行寻址的存储器。CAM 除了有与 RAM 一样的随机读写和保持功能外,关键是还具有与所有地址同时比较的功能,所以查询速度快。

(8)存储周期:亦称存取周期、访问周期、读写周期。连续两次启动同一存储器进行存取操作所需的最短时间间隔。

(9)主存存取时间:从启动一次存储器操作到完成该操作所需时间。

(10)DRAM:动态随机存取存储器(Dynamic Random Access Memory)。

(11)SRAM:静态随机存取存储器(Static Random Access Memory)。

(12)刷新周期: T_{rc} (refresh cycle Time):亦称刷新间隔时间 T_{rf} 。对同一存储位元连续两次刷新,仍能保证鉴别出原存信息的最大允许间隔时间。

(13)刷新操作周期: T_{rec} (refresh operating cycle Time)。刷新一行存储位元即一次刷新操作所需的时间叫刷新操作周期。通常它与存储周期时间相同,即 $t_{rc} = T_{rec}$ 。

例 5.2 存储器有哪几种分类方法？它们是怎样分类的？

解 存储器目前有按在计算机中的作用、按存储介质、按存储方式和按信息的可保存性四种分类方法。

(1)按在计算机中的作用，存储器可分为：高速暂存存储器；高速缓冲存储器；主存储器；辅助存储器，亦称外存储器；其他功能的存储器，如 CPU 中的表格存储器和存储微程序的控制存储器。

(2)按存储介质，存储器可分为：半导体存储器，如 RAM、ROM；磁表面存储器，如磁盘、磁带；光盘存储器；铁电存储器。存储器按存储介质分类，又称按存储机理分类，因为存储介质不同，存储机理也不同。

(3)按存储方式，存储器可分为：随机存取存储器 RAM；按内容寻址存储器 CAM；只读存储器 ROM；顺序存取存储器 SAM，如磁带、只读光盘 VCD 等；直接存取存储器 DAM，如磁盘、可擦写光盘等。

(4)按信息的可保存性，存储器可分为：挥发性存储器和非挥发性存储器，非挥发性存储器又称永久性存储器；破坏性读出存储器和非破坏性读出存储器。半导体 RAM、CAM 为挥发性存储器，半导体 ROM、磁盘、磁带、光盘等为非挥发性存储器。DRAM 为破坏性读出存储器，而 BiRAM、SRAM、ROM、磁盘、磁带、光盘等为非破坏性读出存储器。

例 5.3 何谓存储介质或记录介质？作为存储记忆的元器件应有哪些特征？

解 称能用来记忆二进制代码“1”和“0”的物质或元器件为存储介质或记录介质。作为存储记忆的元器件应仅有两种稳定的物理状态，两种稳定状态又容易相互转换，且能方便地随时检测出它属于哪种稳定状态。

例 5.4 内存储器有哪两种选址方法？它们是怎样实现选址的？

解 内存储器的存储矩阵有线选法二维存储矩阵和重合法三维存储矩阵两种构成方法。前者一维用于选址，一维用于读写；后者二维用于选址，一维用于读写。

假定存储器需 K 位地址，那么存储器可读写的存储单元为 2^K 个。

(1)线选法二维存储矩阵是通过如下方法实现选址的：读写时，提供给存储器的 K 位地址码，经地址译码器可译出 2^K 根驱动线，每线对应一个存储单元。对于具体的 K 位地址，其选中一根且仅选中一根驱动线，该驱动线连接相应存储单元的所有位元，从而实现对选中存储单元的读写。

(2)重合法三维存储矩阵选址的实现是：读写时，提供给存储器的 K 位地址码被分成高位(X 方向)、低位(Y 方向)两部分，当 K 为偶数时，每部分为 $K/2$ 位，分别译码，X、Y 方向分别译出 $\sqrt{2^K}$ 根驱动线；当 K 为奇数时，X 方向地址为 $(K+1)/2$ 位，译出 $\sqrt{2^{K+1}}$ 根驱动线。Y 方向地址为 $(K-1)/2$ 位，译出 $\sqrt{2^{K-1}}$ 根驱动线。对于具体的 K 位地址，在 X 和 Y 方向各选中一根驱动线，两驱动线交叉处的存储位元才构成所选地址的存储单元。

例 5.5 分别绘出 4 字 \times 4 位的线选法和重合法存储矩阵示意图。

(1)4 字 \times 4 位的线选法存储矩阵示意见图 5.24。

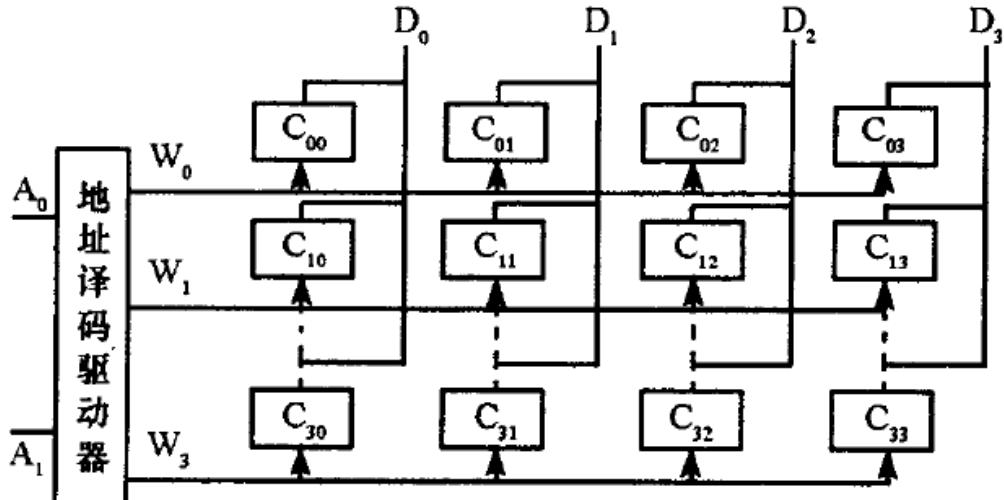


图 5.24 线选法 4 字 × 4 位二维存储矩阵示意图

(2) 4 字 × 4 位的重合法存储矩阵示意见图 5.25。

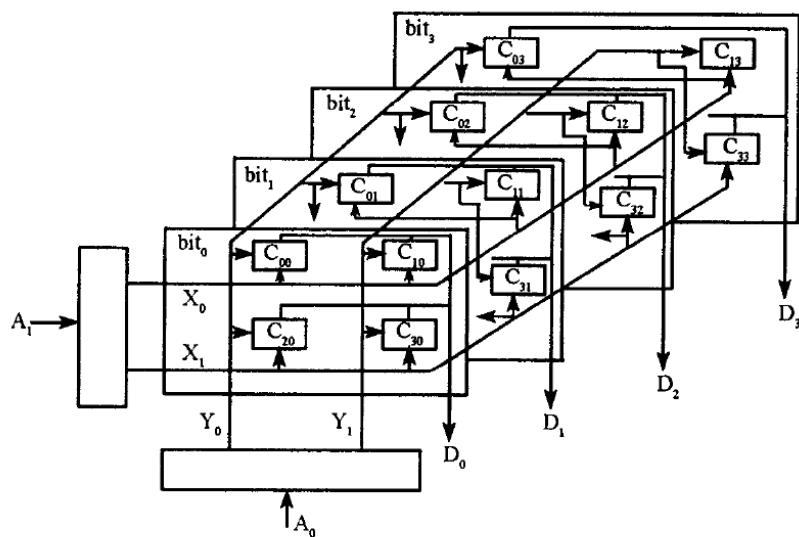


图 5.25 4 字 × 4 位重合法存储矩阵

例 5.6 说明如下存储芯片引脚的含义： A_2 、 D_7 、 Q_5 、 O_4 、 I_0 、 DQ_3 、 IO_6 、 GND 、 NC 、 V_{CC} 、 V_{PP} 、 \overline{CS} 、 \overline{WE} 、 \overline{CE} 、 \overline{OE} 、 \overline{G} 、 \overline{W} 、 \overline{F} 、 RAS 、 CAS 、 \overline{PGM} 。

解	A_2 : 第 2 位地址引脚	D_7 : 第 7 位数据输入引脚	Q_5 : 第 5 位数据输出引脚
	O_4 : 第 4 位数据输出引脚	I_0 : 第 0 位数据输入引脚	DQ_3 : 第 3 位双向数据引脚
	IO_6 : 第 6 位双向数据引脚	GND : 接地引脚	NC : 不连接或不用引脚
	V_{CC} 、 V_{PP} : 电源引脚	\overline{CS} : 片选引脚	\overline{CE} : 片使能引脚
	\overline{OE} 、 \overline{G} : 输出使能引脚	\overline{WE} : 读使能引脚	\overline{F} : DRAM 刷新引脚
	RAS : 行地址选通引脚	\overline{CAS} : 列地址选通引脚	\overline{PGM} : 编程引脚

例 5.7 试从速度、集成度、功耗、应用场合、存储芯片引脚等方面说明 BiRAM、SRAM 和 DRAM 的特点。

解 (1)BiRAM 速度最快,集成度低,位功耗大;用于小容量快速存储器,如 Cache;其存储芯片地址引脚不复用,数据引脚也不复用。如字位结构为 $4K \times 4$ 位 BiRAM 存储芯片,地址引脚有 $\log_2(4K) = 12$ 个;数据引脚有 8 个,4 个数据输入引脚,4 个数据输出引脚。

(2)SRAM 速度比 DRAM 快,比 BiRAM 慢;集成度比 DRAM 低,比 BiRAM 高;位功耗比 DRAM 大,比 BiRAM 小;慢速计算机用它作 Cache,快速计算机用它作主存;其存储芯片地址引脚不复用,地址引脚个数用公式 $\lceil \log_2 \text{存储单元个数} \rceil$ 计算;当字位结构为多位结构时,存储芯片数据引脚必须复用。当字位结构为一位结构时,不能复用;若复用,会使芯片结构更复杂。例如, $64K \times 4$ 的 SRAM 存储芯片地址引脚为 16 个,数据引脚 4 个且为双向引脚,此时必须增加一个输出使能端引脚 \overline{OE} 或 \overline{G} ;而 $64K \times 1$ 的 SRAM 存储芯片地址引脚为 16 个,数据引脚为 2 个,一个数据输入引脚,一个数据输出引脚。

(3)DRAM 速度最慢,集成度最高,功耗最小;在计算机中作主存;DRAM 存储芯片地址引脚复用,此时必须增加行、列地址选通两个控制引脚 \overline{RAS} 、 \overline{CAS} ;当字位结构为多位结构时,数据引脚必须复用,此时必须增加一个输出使能端引脚 \overline{OE} 或 \overline{G} 。当字位结构为一位结构时,不必复用;若复用,会使芯片结构更复杂。例如, $4M \times 4$ 位的 DRAM 存储芯片地址引脚为 11 个,数据引脚 4 个且为双向引脚;而 $4M \times 1$ 的 DRAM 存储芯片地址引脚为 11 个,数据引脚为 2 个,一个数据输入引脚,一个数据输出引脚。

例 5.8 某 DRAM 存储芯片,其字位结构为 $1M \times 1$ 位,试问其地址、数据引脚各是多少个?应设置哪些控制引脚?

解 对于 DRAM 存储芯片,地址引脚复用;当字位结构为一位结构时,数据引脚不能复用,所以:

- 地址引脚个数: $\lceil \log_2 \text{存储单元个数} \rceil / 2 = \lceil \log_2 1M \rceil / 2 = 10$ 。
- 数据引脚个数:2 个,一个作数据输入,一个作数据输出用。
- 设置控制引脚: \overline{WE} 、 \overline{RAS} 、 \overline{CAS} 、 \overline{CS} (也可不设此引脚由 \overline{RAS} 作片选)。

例 5.9 某 SRAM 存储芯片,其字位结构为 $512K \times 8$ 位,试问其地址、数据引脚各是多少个?应设置哪些控制引脚?

解 对于 SRAM 存储芯片,当字位结构为多位结构时,数据引脚应复用。而地址引脚不复用。所以:

- 地址引脚个数为: $\lceil \log_2 \text{存储单元个数} \rceil = \lceil \log_2 512K \rceil = 19$;
- 数据引脚个数为:8;

- 设置控制引脚: \overline{WE} 、 \overline{OE} (或 \overline{G})、 \overline{CS} 。

例 5.10 试述 CAM 基本结构组成、应用场合及优缺点。

解 CAM 存储器,亦称为相联存储器。

(1) 基本结构组成:

- 存储矩阵 MM:信息的物理驻在地,由 CAM 存储位元组成。
- 输入寄存器 IR:比较时,存放关键字,亦称比较数、比较字。写入时,存放要写信息。
- 屏蔽寄存器 MR:比较时,屏蔽不参与比较的位;写入时,屏蔽不写入的位。
- 标志寄存器 FR:存放符合与否和其他所需标志。

- 地址选择器 AS: 它是一个比较复杂的开关网络,由译码器和编码器构成,输入来自 AR 和 FR。若由 AR 提供地址,译码器工作是写入初始数据或读出数据的地址;否则,由 FR 控制编码器工作,是比较符合后确定的地址。

- 地址寄存器 AR: 为初始数据写入或读出提供地址。
- 读出寄存器 RR: 亦称缓冲寄存器,接收读出的信息。

(2) 优缺点:

- 优点:利用 CAM 的屏蔽寄存器,可对存储器所有存储单元的所有位或部分位同时进行比较或改写,使得对存储器查询速度非常快且灵活方便。
- 缺点:CAM 存储位元、存储器结构都比较复杂,造价高,功耗也比较大,使得单片容量和整个存储器容量都比较小。

(3) 应用场合:

- CAM 主要用于快速检索的场合。如 Cache - 主存层次地址映像、变换和虚拟存储器地址映像、变换中使用的快表。
- 网络之间的桥接通信。每个网络可能有成千上万个工作站点,要实现它们的通信,首先要搜索到站点使它们桥接起来,用 RAM 速度很慢,目前大都使用 CAM。
- 人工智能计算机和实时专家系统,要求可以进行大于、小于、等于、是否处于给定的上下界范围以及求最大值、最小值等各种类型的逻辑检索功能。为达到快速、灵活的目的,通常使用 CAM。

例 5.11 试说明 RAM、ROM 和 CAM 的异同点。

解 相同点:RAM、ROM 和 CAM 均为随机存储器,读出时均以存储单元为单位,都可作内存。不同之处在于:

RAM:既能随机读又能随机写的存储器,属于挥发性存储器,断电后信息就不复存在。这里的随机指的是读写所需时间都是相同的,与存储单元所处的物理位置无关。

ROM:只能随机读不能随机写的存储器。信息是在工作前写入的,信息一旦写入后,便可长期保存,属于非挥发性存储器,断电后信息也存在。ROM 除作内存外,还用来存储微程序、固定表格和常数、字库等。

CAM:按比较数的部分内容或全部内容与存储器的所有存储单元同时比较寻址,然后再随机读写的存储器。查询信息的速度非常快。

例 5.12 试述熔断丝型 PROM 和肖特基二极管型 PROM 的工作原理。

解 (1)全“1”熔断丝型 PROM 的工作原理:熔断丝型 PROM 的存储位元由晶体三极管 T 串接一个熔断丝组成。

- 写入信息:写“1”时,不通入电流,熔断丝保持接通状态,写入“1”;写“0”时,通入足够大的电流,熔断熔断丝使之开路,写入“0”。

- 读出信息:加电压,管 T 若通导,有电流流过存储位元,读出“1”,管 T 若不通导,无电流流过存储位元,读出“0”。注意,读写电流之比的经验数据为 1:10,确保写“0”时,烧断熔断丝;读出时不会烧断熔断丝,保持位元所存信息。

(2)全“0”肖特基二极管型 PROM 的工作原理:肖特基二极管型的存储位元由晶体三极管 T 反相串接一个肖特基二极管组成。

- 写入信息:写“1”时,采用足够高的电压,击穿肖特基二极管,使之短路,写入“1”;写“0”时,不加电压,肖特基二极管保持原态,写入“0”。

- 读出信息:加较低电压,T 通导,有电流流过存储位元,读出“1”;加较低电压,T 不通导,无电流流过存储位元,读出“0”。

例 5.13 试从速度、集成度、擦除与编程方法、应用等方面说明 MROM、PROM、EPROM 和 EEPROM 的异同点。

解 相同点:都为随机读,不可随机写的存储器。不同之处在于:

MROM:是固定掩模型 ROM。生产制造时就将信息写入其中,一经形成产品,就不可再修改的存储器。有双极和 MOS 两种类型产品,读出速度快、集成度较低、不能擦除和编程。用于存储固定模式的信息,如字库、监控或游戏程序、固定表格等。

PROM:可修改(编程)一次的 ROM。产品出厂时所存信息不是全“1”,就是全“0”。对所存信息仅可修改一次,一经修改就不可再改变。仅有双极型产品,在 ROM 中读出速度最快、集成度最低。按地址并按位编程且仅能编程一次,用于存储微程序。

EPROM:紫外线擦除电可编程的 ROM。产品出厂时所存信息不是全“1”,就是全“0”。可在脱机情况下多次整片擦除、按地址以存储单元为单位电编程。仅有 MOS 型产品,读出速度较快,单管存储位元集成度较高。需留擦除窗口,封装麻烦。用于存储固定程序,半固定的参数和数据。

EEPROM:电擦除电可编程的 ROM。产品出厂时所存信息不是全“1”,就是全“0”。可在线多次电擦除和编程、可按地址以存储单元为单位电擦除和编程。仅有 MOS 型产品,读出速度较快、双管存储位元集成度较 EPROM 低。不需留擦除窗口,封装较 EPROM 容易。使用场合同 EPROM。

例 5.14 内存储器设计一般应分成哪三个阶段?各阶段都进行哪些工作?

解 详见本章“设计内存的方法”知识点。

例 5.15 用 BiRAM 存储芯片 MCM10146 和 MECL10K 系列门电路组成容量为 $8K \times 48$ 位的 Cache,假定 MECL10K 系列每个门电路输出只能带 8 个负载端。要求:

- (1)计算要多少片 MCM10146;
- (2)画出构成 Cache 的存储矩阵示意图;
- (3)分别计算驱动 A_i 、 D_m 、 \overline{CS} 、 \overline{WE} 所需门数。

解 (1)所需 $1K \times 1$ 位的 BiRAM 存储芯片 MCM10146 的片数:

$$\lceil M/m \rceil \lceil N/n \rceil = \lceil 8K/1K \rceil \lceil 48/1 \rceil = 384(\text{片})$$

(2) Cache 的存储矩阵示意见图 5.26。

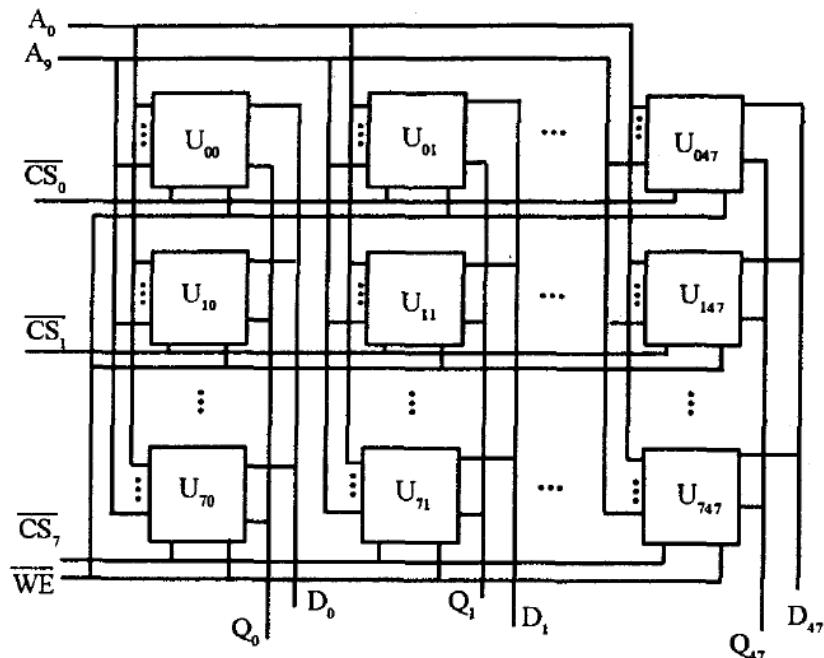


图 5.26 Cache 存储矩阵示意图

(3) 计算所需驱动门数

① A_i 所需门数:

- 每位地址驱动的负载 $L_A : L_A = \lceil M/m \rceil \lceil N/n \rceil f_A = \lceil 8K/1K \rceil \lceil 48/1 \rceil f_A = 384 f_A$ 。
- 每位地址需要的分载门数 $N_A : L_A / 8f_A = 384 f_A / 8 f_A = 48$; 因所需门数超过 8, 单个门驱动不了, 故应当再加 1 级门驱动, 即 $48/8 = 6$ 。所以, $N_A = 48 + 6 = 54$ (门)。

② D_{in} 所需门数:

- 数据输入驱动的负载 $L_{DI} : L_{DI} = \lceil M/m \rceil f_{DI} = 8f_{DI}$ 。
- 数据输入所需的分载门数 $N_{DI} : L_{DI} / 8f_{DI} = 8f_{DI} / 8f_{DI} = 8/8 = 1$, 注意: 当分载门数不大于 1 时, 取分载门数为零, 即不需分载, 直接由相应 MBR 的触发器输出端驱动, 即 $N_{DI} = 0$ 。

③ \overline{WE} 所需门数: 同 A_i 。

④ \overline{CS} 所需门数:

- 每个行片选驱动的负载 $L_{CSi} : L_{CSi} = \lceil N/n \rceil f_{cs} = 48f_{cs}$
- 每个行片选需分载门数为 $N_{CSi} : N_{CSi} = L_{CSi} / 8f_{cs} = 48f_{cs} / 8f_{cs} = 48/8 = 6$ 。该存储器有 8 个行片选, 所以, 片选驱动共需分载门数为 $N_{CS} : N_{CS} = 8N_{CSi} = 8 \times 6 = 48$ (门)。

例 5.16 已知某存储器存储容量为 4MB, 当分别按字节、按 16 位字长、32 位字长和 64 位字长编址时, 地址线(地址码位数)分别为多少根(位)? 数据线分别为多少根? 寻址范围分别为多大?

解

编址方式	字节	16位字长	32位字长	64位字长
地址线数	22	21	20	19
数据线数	8	16	32	64
寻址范围	0~4M-1	0~2M-1	0~1M-1	0~0.5M-1

注意：对存储器而言，地址线是不能复用的，数据线则是必须复用的。

例 5.17 已知某 32 位字长的计算机，主存采用字编址的半导体存储器，其地址线为 20 根，使用 $32K \times 8$ 位的 SRAM 存储芯片组成该机最大的存储空间。试问该存储空间的存储容量为多少字节？需多少 SRAM 存储芯片？哪几位地址作为字扩展去控制 CS？

解 (1) 存储器的存储容量：

因存储器存储地址(存储单元)个数为： $2^{20} = 1M$ ，每个地址存储的信息位数(字长)为： $32b = 4B$ ，而存储容量 $C = \text{存储位数}/\text{每个地址} \times \text{存储地址个数}$ 。故有：

$$C = 1M \times 4B = 4MB$$

(2) 所需存储芯片数：

$$\lceil M/m \rceil \lceil N/n \rceil = \lceil 1M/32K \rceil \lceil 32/8 \rceil = 32 \times 4 = 128(\text{片})$$

(3) 哪几位地址码作为字扩展去控制片选 \bar{CS}

因该存储器地址线为 20 根，即有 20 位地址码。假定为： $A_{19} \sim A_0$ ；而存储芯片 $32K \times 8$ 位所需的地址位数为： $\log_2(32K) = 15$ ，即存储器的低 15 位地址线($A_{14} \sim A_0$)直接与存储芯片的地址引脚连接。而存储器的高 5 位地址码($A_{19} A_{18} A_{17} A_{16} A_{15}$)经译码后有 32 种状态输出分别与 32 行芯片连接。

例 5.18 用 $4K \times 4$ 位的 SRAM 存储芯片构成具有 16 根地址线，字长 16 位的存储器。试回答如下问题：

(1) 画出存储芯片引脚示意图；

(2) 该存储器的存储容量为多少 KB？构成该存储器需多少存储芯片？

(3) 画出片选译码和存储矩阵示意图。

解 (1) 存储芯片引脚示意如图 5.27。

(2) 因字长 16 位、地址线 16 根，所以

- 存储器容量：

$$C = 2^{16} \times 16 \text{ 位} = 64K \times 2B = 128KB$$

- 所需存储芯片数：

$$\lceil M/m \rceil \lceil N/n \rceil = \lceil 64K/4K \rceil \lceil 16/4 \rceil = 16 \times 4 = 64(\text{片})$$

(3) 片选译码和存储矩阵示意如图 5.28。

例 5.19 已知 SRAM 存储芯片引脚示意如图 5.29 所示。其中， DQ_i ：数据输出输入端； $A_0 \sim A_{10}$ ：地址输入端； \bar{CS} ：片选端； \bar{G} ：输出输入控制； \bar{WE} ：读写控制端。试回答：

(1) 依地址和数据端的数量计算存储芯片容量。

(2) 试用该芯片构成 $16K \times 32$ 位的存储器：

① 计算需用多少片 SRAM 存储芯片？

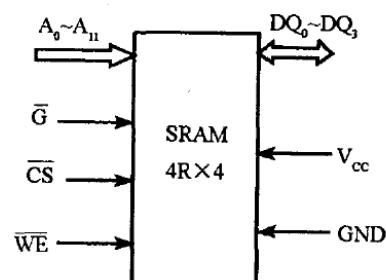


图 5.27 $4K \times 4$ 的 SRAM 引脚示意图

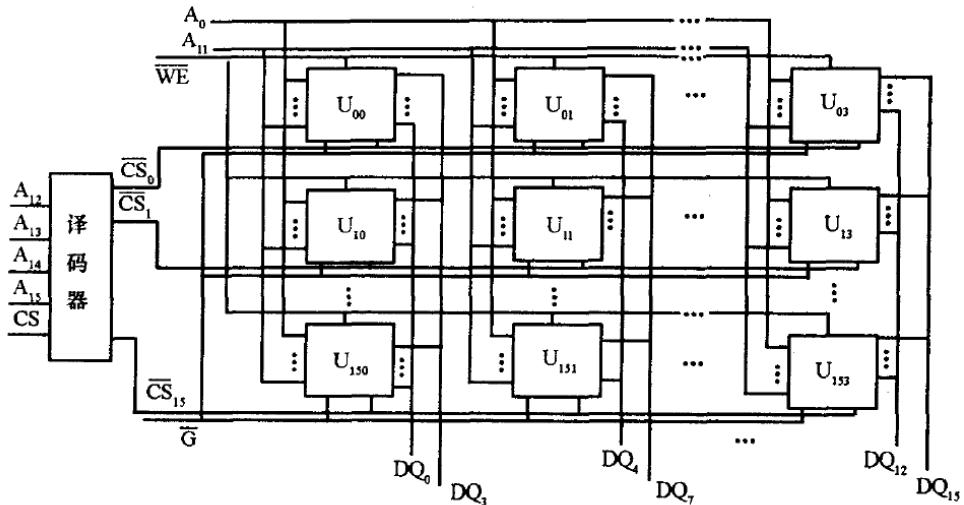


图 5.28 片选译码与存储矩阵示意图

②存储器需多少位地址码？片选应用哪几位地址码译码产生？

③假定一个门电路能驱动 8 个 \overline{WE} 负载，求该存储器的 \overline{WE} 所需驱动门数。

解 (1) 因 SRAM 数据引脚复用，地址引脚不复用，而存储芯片有 11 个地址引脚。所以存储芯片包含 2^{11} 个地址，即 2^{11} 个存储单元，而每个存储单元存储 4 位，故存储芯片容量为： $C = 2^{11} \times 4b = 2K \times 4$ 位。

(2) 用该芯片构成 $16K \times 32$ 位的存储器：

① 需存储芯片数：

$$[M/m][N/n] = [16K/2K][32/11] = 8 \times 3 = 24 \text{ (片)}$$

② 存储器需地址码位数：

$$\log_2(16K) = 14 \text{ (位)}$$

片选译码需要哪几位地址：

$$\log_2([16K/2K]) = \log_2 8 = 3 \text{ (存储器的高 3 位)}$$

③ 存储器的 \overline{WE} 端所需驱动门数： $24 f_{WE}/8 f_{WE} = 3 \text{ (门)}$

例 5.20 用 $64K \times 1$ 位的 DRAM 存储芯片，构成以字节编址的 $4M \times 8$ 位主存，并将它分装在存储容量为 $1M \times 8$ 位的存储板上。试回答如下问题：

(1) 需多少存储芯片？分装在几块板上？

(2) 主存所需地址位数？板选、片选所需地址位数？画出板选、片选逻辑示意图。

(3) 计算 \overline{WE} 负载端数？若每个门电路能驱动 8 个 \overline{WE} 负载，需多少个门电路？

(4) 每块存储板至少需多少引脚？若为 SRAM 芯片，每块板至少需多少引脚？

解 假定存储器地址码为 $A_{21} \sim A_0$ ，且 A_{21} 为最高位地址。

(1) 所需芯片数： $[M/m][N/n] = [4M/64K][8/1] = 64 \times 8 = 512 \text{ (片)}$

分装的存储板数： $[4M/1M] \cdot [8/8] = 4 \text{ (块)}$

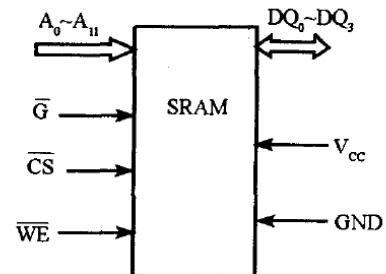


图 5.29 SRAM 存储芯片引脚

(2) 主存地址码位数: $\log_2(4M) = \log_2 2^2 = 22$ (位)

片内地址码位数: $\log_2(64K) = \log_2 2^{16} = 16$ (位)

片选用地址码位数: $\log_2\lceil(1M/64K)\rceil = \log_2 16 = 4$ (位)

板选用地址码位数: $\log_2\lceil(4M/1M)\rceil = \log_2 4 = 2$ (位)

板选、片选译码示意如图 5.34。

(3) \overline{WE} 的负载数、所需驱动门数

因每个存储芯片有一个负载 \overline{WE} , 存储器 512 片分装在四块板上, 每块板上有 128 个存储芯片, 即有 128 个 \overline{WE} 端。所以每块板所需驱动门数为:

$128/8 = 16$, 因 $16 > 8$, 单个门驱动不了, 需再加一级门, $16/8 = 2$; 共需 $16 + 2 = 18$ (个)。

四块板共需驱动门数为: $18 \times 4 = 72$ (个)。

注意:之所以先计算每块板所需门数,原因是不同的存储板不能共用门。所以不能用 $2/8 = 64, 64/8 = 8$, 共 72 来计算。此时虽然也是 72 个门,具体在每块板上如何连接未体出来。若计算门数时,负载端数若不能被 8 整除,就会出现两块板共用一个门的情况,是不容许的。当然此题不会出现这种情况。

(4) 存储板引脚个数

①DRAM 存储板引脚个数

- 数据引脚: 8 个(数据输入输出复用)
- 地址引脚: 8 个(地址引脚复用, 两位地址使用一个引脚)
- 作片选译码的引脚: $4(A_{19} A_{18} A_{17} A_{16}) + 2$ (板选和 CS, 见图 5.30)

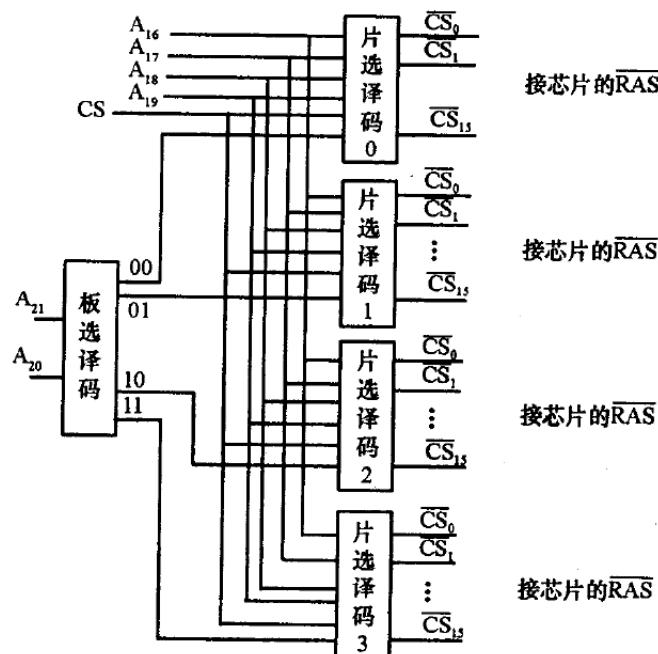


图 5.30 板选、片选译码示意图

- 电源引脚: 2 个(V_{cc} 、GND)

- 控制引脚:3个(\overline{OE} 、 \overline{WE} 、 \overline{CAS})

共计: $8+8+6+2+3=27$ (个)

注意: \overline{RAS} 由片选译码输出给出,板选译码和地址复用逻辑在四块存储板之外。

②SRAM 存储板引脚个数

- 数据引脚:8个(数据输入输出复用)
- 地址引脚 16 个(地址引脚不允许复用)
- 作片选的引脚: $4(A_{19} A_{18} A_{17} A_{16}) + 2$ (板选和 CS, 见图 5.30)
- 电源引脚:2个(V_{cc} 、GND)
- 控制引脚:2个(\overline{OE} 、 \overline{WE})

共计: $8+16+6+2+2=34$ (个)

例 5.21 某计算机字节编址,单总线结构。地址线 16 根,从高位至低位依次为 $A_{15} \sim A_0$; 双向数据总线 8 根 $DQ_7 \sim DQ_0$; 与主存有关的控制线为允许访存线 \overline{MREQ} , 低电平有效; 读写控制线 R/\overline{W} , 高电平为读, 低电平为写。现提供: $2K \times 8$ 位、 $4K \times 8$ 位和 $8K \times 8$ 位 3 种 ROM 芯片; $64K \times 1$ 位、 $2K \times 8$ 位、 $4K \times 8$ 位和 $8K \times 8$ 位 4 种 SRAM 芯片; 74LS138 3:8 译码器和 74LS139 2:4 译码器和各种门电路。译码器示意如图 5.26, 功能说明见知识点的例 3。试按上述地址空间分配设计两个主存:

1. 主存一的地址空间分配:

- ① $9000H \sim A7FFH$ 为系统程序区, 用 ROM;
- ② $A800H \sim AFFFH$ 为系统工作区, 用 RAM;
- ③ $B000H \sim BFFFH$ 为用户区, 用 RAM。

2. 主存二的地址空间分配:

- ① 最大 8K 地址的存储空间为系统程序区, 用 ROM;
- ② 与之相邻的 4K 地址空间为系统工作区, 用 RAM;
- ③ $A000H \sim BFFFH$ 为用户区, 用 RAM。

要求:选用合适的芯片完成相应设计并说明所需芯片种类及片数;画出片选和存储矩阵逻辑图。

分析:本题既用到 RAM 存储芯片,又用到 ROM 存储芯片,且所用芯片容量不一定都一样。对于此类存储器的设计,首先要对所配地址空间进行分析,其次是依地址的分析选择合适的存储芯片,最后选择译码器和门电路,这里地址空间的分析是基础。所谓选择合适的存储芯片指的是,完成设计所需存储芯片个数和种类要少,同时要使得译码逻辑简单且容易实现,所需译码器和门电路个数和种类也要少。

解 1 主存一的地址空间分配分析:

①地址空间分配: $9000H \sim A7FFH$ (ROM)

1001 0000 0000 0000	$\}$	$36K \sim 40K - 1$	ROM_1	$4K \times 8$ 位	ROM	1 片
1001 1111 1111 1111	$\}$	$40K \sim 42K - 1$	ROM_2	$2K \times 8$ 位	ROM	1 片

②地址空间分配: $A800H \sim AFFFH$ (RAM)

$\begin{array}{ccccccc} 1010 & 1000 & 0000 & 0000 \end{array} \} 42K \sim 44K - 1$ RAM₁ 2K × 8 位 RAM 1 片
 $\begin{array}{ccccccc} 1010 & 1111 & 1111 & 1111 \end{array}$

③地址空间分配: B000H ~ BFFFH (RAM)

$\begin{array}{ccccccc} 1011 & 0000 & 0000 & 0000 \end{array} \} 44K \sim 48K - 1$ RAM₂ 4K × 8 位 RAM 1 片
 $\begin{array}{ccccccc} 1011 & 0111 & 1111 & 1111 \end{array}$

通过对地址空间分配的地址码分析可知, A₁₅ 为“1”是所有地址空间必需的, 因此片选译码必须有其参与控制。凡是 4KB 容量的存储芯片, 除 A₁₅ 外, 还需用 A₁₄ A₁₃ A₁₂ 3 位地址给予控制。对于 2KB 容量的存储芯片除用 A₁₅ A₁₄ A₁₃ A₁₂ 控制之外, A₁₁ 也应参与其中。

依地址空间分配及其分析, 设计该主存需使用 4K × 8 位、2K × 8 位的 ROM 各一片, 2K × 8 位、4K × 8 位的 RAM 各一片, 构成存储矩阵; 使用一片 74LS138 和几个门电路构成片选译码电路。片选和存储矩阵逻辑如图 5.31 所示。

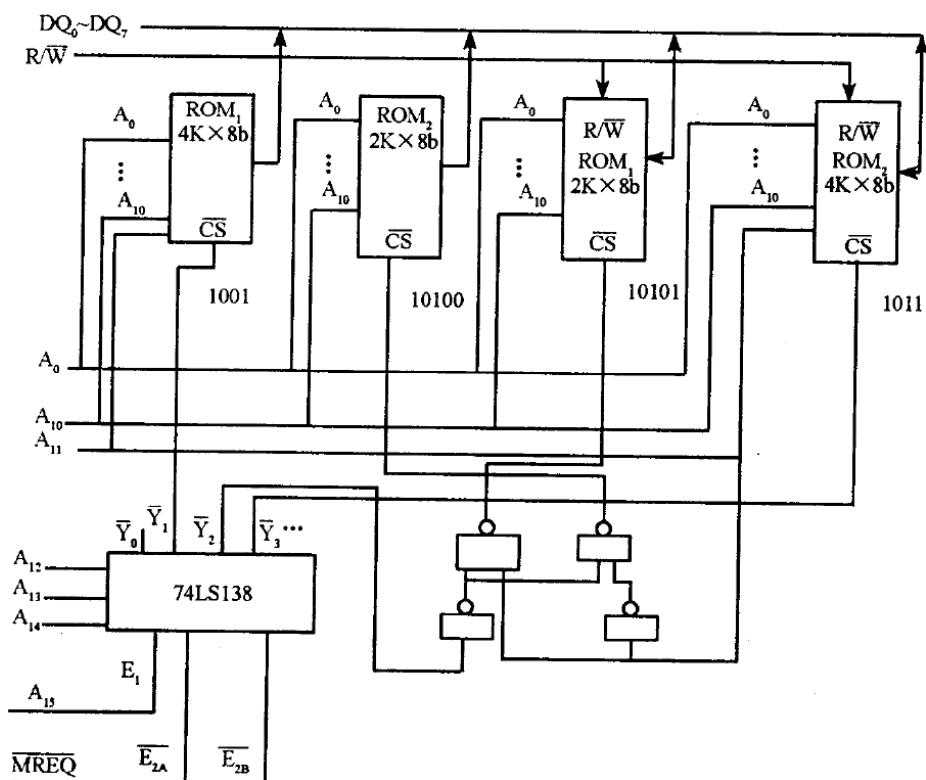


图 5.31 主存一的片选译码与存储矩阵示意图

解 2 主存二的地址空间分配分析:

①最大地址的 8K 空间为系统程序区, 用 ROM, 地址区域为: E000H ~ FFFFH。

$\begin{array}{ccccccc} 1110 & 0000 & 0000 & 0000 \end{array} \} 56K \sim 64K - 1$ ROM₁ 用 8K × 8 位的 ROM 一片
 $\begin{array}{ccccccc} 1111 & 1111 & 1111 & 1111 \end{array}$

②与①相邻的 4K 地址空间为系统工作区, 用 RAM, 地址区域为: D000H ~ DFFFH。

$1101\ 0000\ 0000\ 0000$ } 52K ~ 56K - 1 RAM₁ 用 $4K \times 8$ 位的 RAM 一片
 $1101\ 1111\ 1111\ 1111$

③ A000H ~ BFFFH, 为用户区, 用 RAM, 地址区域为 (40K ~ 48K - 1)。

$1010\ 0000\ 0000\ 0000$ } 40K ~ 48K - 1 RAM₂ 用 $8K \times 8$ 位的 RAM 一片
 $1011\ 1111\ 1111\ 1111$

通过对地址空间分配的地址码分析推知, $A_{15} A_{14} A_{13}$ 可作为 8KB 存储芯片的片选译码控制。对于 4KB 存储芯片要用 $A_{15} A_{14} A_{13}$ 和 A_{12} 共同控制。

依地址空间分配及其分析, 设计该主存需使用 $8K \times 8$ 位的 ROM, $8K \times 8$ 位和 $4K \times 8$ 位的 RAM 各 1 片构成。片选译码电路使用 74LS138 一片和两门电路构成, 见图 5.32。

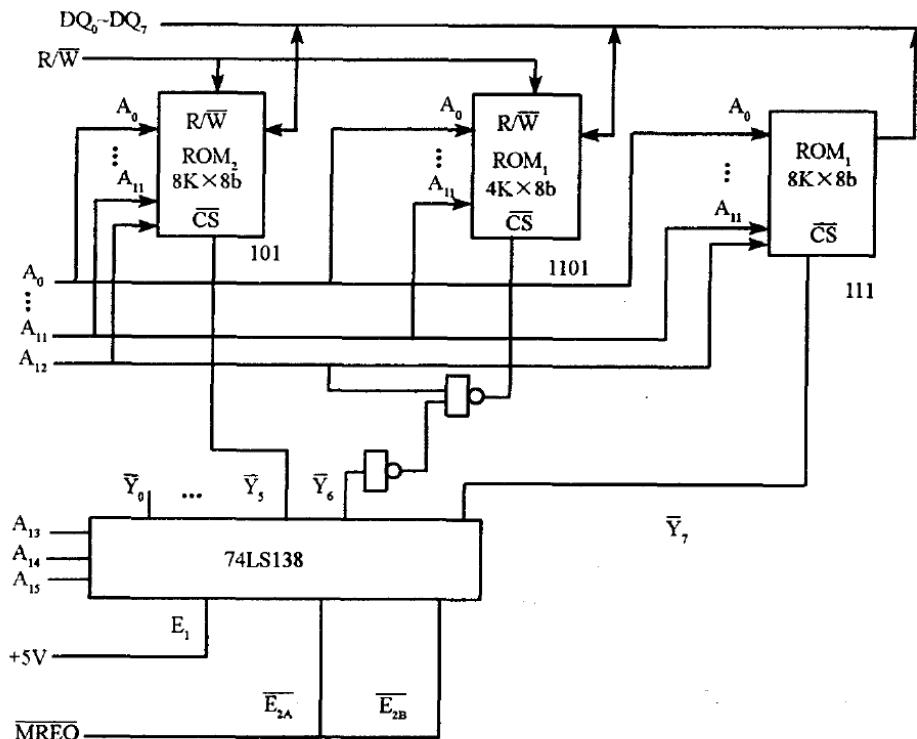


图 5.32 主存二的片选译码与存储矩阵示意图

例 5.22 何谓 DRAM 刷新? DRAM 为什么要刷新? DRAM 刷新的要求是什么?

(1) 何谓 DRAM 刷新: 按一定的时间间隔为 DRAM 存储位元中记忆电容充电的过程。

(2) DRAM 为什么要刷新: DRAM 存储位元是靠电容电荷存储效应记忆信息的。通常用电容上电荷的有、无表示“1”和“0”。信息写入后, 随着时间的推移, 电容上电荷的泄漏将会丢失所有信息。为了保证所存信息的正确性, 必须按一定的时间间隔为电容充电, 使电容电荷恢复到泄漏前的状态。

(3) DRAM 刷新的要求: 刷新的要求亦称刷新的原则, 刷新有三个要求。一是定时刷新, 即严格按刷新周期规定的时间刷新; 二是刷新优先于访存但不能打断访存, 即当提出刷新要求时若正在访存, 需等待本次访存结束, 才能响应刷新; 三是刷新期间不允许访存, 即刷新和访存是互斥的。设计刷新逻辑线路时必须遵循这三个原则。

例 5.23 按刷新操作周期分配方式分类, 有哪几种刷新方法? 各自优缺点怎样?

(4) 按刷新操作周期分配方式分类,可分为集中式刷新、分散式刷新和透明式刷新三种方法。

现以 MCM511000A 芯片为例说明,其刷新周期 T_r 为 8ms, 刷新操作周期 T_{rc} 为 160ns, 刷新周期数为 512, 即 512 个 T_{rc} 完成整个芯片 1M 位元的刷新。

①集中式刷新

集中式刷新亦称批刷新,就是从刷新周期 T_r 中抽出最后 512 个访存周期作为刷新操作周期集中进行刷新。注意,访存周期和刷新操作周期时间相同。

集中式刷新优点是控制逻辑简单,设计容易实现;缺点是效率不高,如对于 MCM511000A 芯片构成的存储器,每 8ms 中就有 512 个访存周期即 $160\text{ns} \times 512 = 81.92\mu\text{s}$ 的时间,不允许访存,CPU 要处于等待状态,影响了计算机的工作效率。

②分散式刷新

分散式刷新是把集中到一起的不允许访存的刷新时间分散开,即把刷新周期 T_r (8ms) 分成刷新周期数(512)等份,在每等份的最后一个访存周期作为刷新操作周期,以完成一行存储位元的刷新。

分散式刷新的优点是计算机的工作效率高。因每次只刷新一行存储位元,仅占用一个访存周期,刷新可能插在计算机执行与存储器无关的操作时间内进行,即有时不影响访存。缺点是控制逻辑复杂,设计不易实现。

③透明式刷新

透明式刷新方式的基本思想是设一个系统访存周期,它是存储器实际访存周期的两倍,并令系统访存周期的前半周作为存储器访存周期,后半周用作存储器刷新操作周期。这样,每当系统访问一次存储器,就自动顺序刷新存储芯片中的一行存储位元。

透明式刷新优点是控制简单、设计容易,不需增加多少器材;缺点是存储器效率仅为 50%,只能用于低速计算机系统。

5.24——5.49

例 6.2 磁记录材料和磁头材料各有何特点?

解 磁记录材料是用来记录二进制信息的,其应具有硬磁特性。即在外磁化场撤消后仍保持磁化状态,有较高的剩磁值,适当的矫顽力和良好的矩形特性,以利于记录和将记录的信息读出。

磁头材料是制作磁头用的,应具有软磁特性。即有电流作用时,产生的饱和磁感应强度 B_m 大,以便产生强磁场,可靠的写入信息;对外磁化场的导磁率高,以利于读出信息;当外磁化场撤消后,剩磁小,以便保存磁记录介质已记录的信息。

例 6.3 什么是矩磁材料?什么是软磁材料?各有何特点?在磁记录中,各应用于什么场合?为什么?

解 矩磁材料,即具有矩形磁滞回线的磁性材料。其具有硬磁特性。即在外磁化场撤消后仍保持原磁化状态,有较高的剩磁值,适当的矫顽力和良好的矩形特性。

软磁材料,即具有软磁特性的磁性材料。这种材料,当有电流作用时,饱和磁感应强度 B_m 大,当有外磁化场时导磁率很高,而当外磁化场撤消后,几乎无剩磁,随即回到未磁化状态。

在磁记录装置中,矩磁材料因其具有的硬磁特性,适用于制作磁记录材料。软磁材料因其具有的软磁特性,适用于制作磁头材料。

例 6.4 磁头有几种分类方法?它们是怎样分类的?

解 磁头目前有按磁头的使用功能、按磁头的工作方式、按磁头组合结构和按磁头工作原理四种分类方法。

(1) 按照磁头的使用功能分类

- ①写头或读头:只有单一写入功能的磁头或只能读出不能写入的磁头。
- ②读写磁头:既能读出又能写入的读写磁头,其仅有一个工作间隙,读写分时使用。
- ③读写组合磁头:具有双工作间隙的写后读磁头,写后马上读以检查写入是否正确。
- ④抹头:专用于抹去信息的磁头。
- ⑤伺服头:专门用来读写磁道位置信息的磁头。

(2) 按照磁头的工作方式分类

- ①接触式磁头:读写时,磁头与介质是接触的。
- ②浮动式磁头:读写时,磁头与磁介质永远保持固定的很小间隙。
- ③固定磁头:每个磁道均设置一个磁头,读写时不必移动磁头找寻磁道。
- ④移动磁头:一个磁面仅一个磁头,读写时需先移动磁头找到磁道,而后才能读写。

(3) 按磁头组合结构分类

- ①单磁头:一个磁头组件仅包含一个磁头。
- ②组合磁头:一个磁头组件包含多个磁头,以便完成特定功能。

(4) 按磁头工作原理分类

- ①感应式磁头:依电磁感应原理实现读写,磁头内部有电磁感应用的线圈。
- ②磁阻或巨磁阻磁头:依磁阻变化实现读出,磁头内部有磁阻元件,其只能用来读出信息,写入信息必须使用感应式磁头。

例 6.5 磁头工作间隙有何作用？间隙宽度以及磁头和介质表面之间的距离对记录性能有何影响？为什么？

解 由磁记录原理可知，磁头工作间隙是感应性磁头实现磁表面存储器读写的关键结构，没有它就不能实现读写。

工作间隙宽度决定了磁化位元的大小。工作间隙宽度窄，磁化位元小；工作间隙宽度宽，磁化位元大，所以工作间隙宽度将影响记录介质的记录密度。磁头和介质表面之间的距离不能大，距离大的缺点，一是写入和读出困难，距离太大，将会读不出信息；二是即使能写入和读出信息，也会因存储位元所占区域太大，使得记录密度太小。磁头和介质表面之间的距离越小越好，距离小的优点是记录密度高，读写可靠。距离小的缺点是，由于介质表面不平或偏摆，会造成磁头或介质的损坏。为此，有的磁表面存储器使用浮动式磁头，如硬磁盘。有的磁表面存储器使用接触式磁头，这时磁头和介质是接触的，如磁带。

例 6.6 从磁头和磁介质考虑，如何提高磁表面存储器的记录密度？

解 从磁头的角度考虑，减小磁头工作间隙，使存储区域即磁化位元变小；使用单极型磁头，磁化位元由水平变为垂直，使磁化位元更小；使用 MR、GMR 作为读磁头，使得读出信息时，磁头读出信号的大小和磁头与磁介质运动的相对速度无关。

从磁记录介质考虑，磁层要尽量薄；尽量选择高记录密度的磁介质。

从磁头和磁记录介质双方考虑，两者的间距要尽量小，甚至可以采用两者接触的方式。

例 6.7 硬磁盘驱动器如何分类？各有何特点？

解 硬盘驱动器 HDD 分为固定磁头 HDD、移动头固定盘组 HDD 和移动头可换盘组 HDD 三类。各类 HDD 特点分述如下

(1) 固定磁头 HDD 特点：每道都设置一个磁头，工作时磁头可上、下移动，转稳后贴近磁面，无寻道时间，只需由电子线路译码找磁头，故存取时间短($t_s = t_w + t_{sw}$)；磁头数量多、造价高且使磁道密度降低、存储容量减小。此类 HDD 只用于特殊场合。

(2) 移动头固定盘组 HDD 的特点：盘组安装在主轴上，不可更换。一个磁面有一或两个磁头，盘组所有磁面的磁头安装在一个磁头架上。定位机构驱动磁头架寻找磁道，因此可同时定位到盘组所有磁面编号相同的磁道上。道密度最高、存储容量大，因需找道时间，存取时间长($t_s = t_s + t_w + t_{sw}$)，目前计算机广泛使用这类 HDD。

(3) 移动头可换盘组 HDD 的特点：在结构上与移动头固定盘 HDD 唯一不同的是盘组可装卸，相同型号盘组可互换，因此具有脱机存储能力，存储容量是无限的。但制造公差要求严格，限制了可靠性；与相同尺寸固定盘组比较，道密度不能做得很密，故单个盘组存储容量小。

例 6.8 何谓温彻斯特技术？采用该技术制造磁盘有何好处？

解 温彻斯特技术主要包括三个方面的内容。一是将磁头、盘组、定位机构和主轴电机密封在一个盘盒内构成头盘组件 HAD；二是采用小尺寸小浮力的浮动磁头和接触启停方式，并在介质上涂覆润滑剂以防损害磁面；三是将写放大器、读出前置放大器和磁头选择开关等集成化并安置在磁头臂上。采用该技术制造磁盘好处是：

(1) 因构成密封的头盘组件,可大大减少灰尘污染。降低磁头浮动高度,又可减小磁盘尺寸。若为可换式,整体装卸可消除影响磁头定位精度的机械变动因素。

(2) 因采用小尺寸小浮力的浮动磁头和接触启停方式,并在介质上涂覆润滑剂,既消除了磁头加载对盘面可能造成的冲击,又可减少头面间隙而提高记录密度。

(3) 将放大器等集成化并安置在磁头臂上,改善了读写信号的高频传输特性。

例 6.9 移动磁头固定盘驱动器组成及各组成部分的功能。

解 移动头固定盘组 HDD 由盘片与主轴驱动系统,磁头定位系统和数据控制系统三部分组成。它们的功能如下:

(1) 盘片与主轴驱动系统的功能:使盘组尽快转稳、转准,以便对盘组进行读写。

(2) 磁头定位系统的功能:快速寻找并跟踪磁道;启动后或中途找道出错时,磁头能迅速准确地回到零道;停机、掉电磁头能迅速退出盘面或进入启停区,以保护磁头和磁面。

(3) 数据控制系统的功能:控制和实现数据的写入和读出。

例 6.10 磁盘中磁头定位驱动系统有何功能?对其性能有何要求?

解 磁头定位驱动系统的功能:快速寻找并跟踪磁道;启动后或中途找道出错时,磁头能迅速准确地回到零道;停机、掉电时磁头能迅速退出盘面或进入启停区,以保护磁头和磁面。

要求定位性能速度快、精度高。速度快意味着找道时间短,从而实现快速存取,以满足计算机系统的要求。精度高即寻找到的磁道与实际磁道的误差小,这样既能保证可靠读写,又有利于提高道密度。

例 6.11 磁盘存储器地址由哪几部分构成?各部分的意义是什么?

解 主机和磁盘是以扇区为单位成批地交换信息的,且磁盘地址指向的是起始扇区。所以磁盘存储器地址的一般形式为:台号·柱面·盘面·扇区。

台号:指出是计算机系统的哪个磁盘驱动器,若系统仅配一台磁盘,则台号可省略。

柱面:指出本次交换的柱面地址,即磁道号。其在交换信息过程中始终不变。

盘面:指出本次交换的盘面地址,即磁头号。

扇区:指出本次交换的起始扇区地址,亦称起始扇段号。

例 6.12 磁盘的存储容量由哪些因素确定?分析提高数据传输率和缩小存取时间的途径。

解 磁盘的容量由盘组中的盘面数,每个盘面的磁道数和磁道的容量所决定。而盘面的磁道数由道密度决定,磁道的容量由位密度决定。所以,也可以说磁盘的容量是由盘组中的盘面数以及磁盘的磁道密度和位密度决定的。

数据传输率是单位时间磁盘传送数据的位数或字节数。通常用磁盘转动一圈所读写的数据量与转动一周所用时间之比求得,即用公式 $f = D_b \pi D_{max} / t = C_m / t$ 计算。因此,提高数据传输率可以通过增加道容量即提高位密度和提高磁盘转速来实现。

磁盘的存取时间主要由磁头的找道时间和磁头等待所需的扇区到达磁头位置时间决定。因此,缩小存取时间可以通过减小磁头的找道时间即加速磁头的移动速度和缩小等待时间即提高磁盘转速实现。

例 6.13 磁盘和内存交换信息时,为什么一般不跨柱面交换?

解 为提高磁盘的数据传输性能,磁盘的一次读写操作,最多能交换一个柱面的信息量。原因是对于移动头磁盘驱动器,磁头定位系统一次定位操作,定位的磁道集合正好是一个柱面。若信息交换量超过一个柱面,或逻辑上作为一个整体的信息跨柱面存放,则需磁头定位系统多次定位,这势必使得磁头在不同柱面间频繁移动,从而增加读写信息的寻道时间,这必将导致磁盘的数据传输性能大大下降。所以磁盘和内存交换信息时,为了提高数据传输速度,一般不跨柱面交换。

例 6.14 有一单片盘驱动器,每面可记录 3000 道,道容量为 1250000 位,转速为 7200RPM,求其平均等待时间、数据传输率和存储容量。

已知: $m = 2$, $n = 3000$, $C_m = 1250000b$, $V_{转} = 7200RPM$ 。求:(1) 平均等待时间 \bar{t}_w ; (2) 数据传输率 f ; (3) 非格式化存储容量 C_n 。

解 由已知得: $t = t_{max} = 60/7200 = 1/120s$

$$(1) \bar{t}_w = (t_{min} + t_{max})/2 = (0 + 1/120)/2 = 1/240s \approx 4.16ms$$

$$(2) f = C_m/t = 1250000 \times 120/8 = 1.875 \times 10^7 B/s = 17.88MB/s$$

$$(3) C_n = ftm n = C_m m n = 1250000 \times 2 \times 3000/8 = 9.375 \times 10^8 B \approx 894.07MB$$

例 6.15 磁盘机的盘组有 20 个记录数据的盘面,盘直径为 12inch,其中半径 3.5inch 区域用于记录信息,假定盘外侧空白区域为 0.3inch。其记录密度为 2500TPI 和 127000bpi,磁盘转速为 5400RPM,假定 $\pi = 3$,计算盘组容量和数据传输率。

已知: 记录区域半径为 3.5inch, 盘面外侧空白 0.3inch。 $m = 20$, $D_b = 127000bpi$, $D_t = 2500TPI$, $V_{转} = 5400RPM$ 。求:(1) 非格式化盘组容量 C_n ; (2) 数据传输率 f 。

解 由已知可求得:

$$D_{min} = 12 - 2 \times (3.5 + 0.3) = 4.4$$

$$t = 60/5400 = 1/90s$$

$$n = 2500 \times 3.5 + 1 = 8751$$

(1) 非格式化盘组容量 C_n

$$\begin{aligned} C_n &= D_b \pi D_{min} m n = 127000 \times 3 \times 4.4 \times 20 \times 8751/8 \\ &= 36675441000B \approx 34.157GB \end{aligned}$$

(2) 数据传输率 f

$$f = D_b \pi D_{min} / t = 127000 \times 3 \times 4.4 \times 90/8 = 18859500B/s \approx 17.99MB/s$$

例 6.16 磁盘机的盘组由 6 个盘片组成,其专设 1 个伺服面,其他盘面为记录数据的盘面。转速为 7200RPM,有效记录区域的外直径为 13.3cm,内直径为 6.3cm,记录密度为 6640bpm,磁道道距 P_t 为 0.01mm。假定 $\pi = 3$,试计算:

(1) 计算盘组存储容量。

(2) 数据传输率为每秒多少字节?

(3) 若主机字长 32 位,主存存取周期为 50ns,主存以 DMA 方式和盘组交换信息,试计算磁盘信息交换所用存取周期数与主存正常工作周期数之比。

已知: 盘组有 6 个盘片,专设 1 个盘面作伺服面。盘片外直径 $D_{max} = 13.3cm$, 盘片内直径 $D_{min} = 6.3cm$, $P_t = 0.01mm$, $D_b = 6640bpm$, $V_{转} = 7200RPM$ 。求:(1) 非格式化盘组容量 C_n ; (2) 数据传输率 f ; (3) 磁盘工作周期数与主存工作周期数之比。

解 由已知可求得:

$m = 6 \times 2 - 1 = 11$; 一个盘片两个盘面, 伺服面不记录数据所以减“1”。

$$n = (13.3 - 6.3)/2 \times 10/0.01 + 1 = 3501;$$

$$t = 60/7200 = 1/120\text{s}.$$

$$(1) C_n = D_b \pi D_{\min} m n$$

$$= 6640 \times 3 \times 6.3 \times 10 \times 11 \times 3501/8 = 6041220500\text{B} \approx 5.626\text{GB}$$

$$(2) f = D_b \pi D_{\min} / t = 6640 \times 3 \times 6.3 \times 10 \times 120/8 = 18824400\text{B/s} \approx 17.95\text{MB/s}$$

(3) 磁盘工作周期数与主存工作周期数之比。

$$\text{主存工作周期数/s: } 1/50\text{ns} = 2 \times 10^7$$

$$\text{磁盘工作周期数/s: } 1/(4/18824400) \approx 4.71 \times 10^6 \text{ (4/18824400 为传送 32b 所用时间)}$$

$$\text{磁盘工作周期数/主存工作周期数} \approx 4.71 \times 10^6 / 2 \times 10^7 \approx 471/2000$$

例 6.17 磁盘机的盘组由 11 个盘片组成, 其中专设 1 个盘面为伺服面, 其他盘面为记录数据的盘面。盘存储区域内直径为 14.3cm, 外直径为 30.3cm, 道密度为 80TPM, 每道分为 96 个扇区, 每个扇区 4K 字节, 磁盘转速为 3600RPM。试计算:

(1) 盘组容量 C_f 是多少字节?

(2) 平均等待时间是多少?

(3) 数据传输率是多少字节/s?

已知: 盘组有 11 个盘片, 专设 1 个盘面作伺服面。盘片外直径 $D_{\max} = 30.3\text{cm}$, 盘片内直径 $D_{\min} = 14.3\text{cm}$, $S_t = 96$, $D_t = 80\text{TPM}$, $B_s = 4\text{KB}$, $V_{\text{转}} = 3600\text{RPM}$ 。求: (1) 格式化盘组容量 C_f ; (2) 平均等待时间 t_w ; (3) 格式化数据传输率 f_f 。

解 由已知得:

$m = 11 \times 2 - 1 = 21$; 一个盘片两个盘面, 伺服面不记录数据所以减“1”。

$$n = (30.3 - 14.3)/2 \times 10 \times 80 + 1 = 6401;$$

$$t = t_{\max} = 60/3600 = 1/60\text{s}.$$

$$(1) C_f = B_s S_t m n = 4\text{KB} \times 96 \times 21 \times 6401 = 51617664\text{KB} \approx 49.226\text{GB}$$

$$(2) \bar{t}_w = (t_{\min} + t_{\max})/2 = (0 + 1/60)/2 = 1/120\text{s}$$

$$(3) f_f = C_f/t = B_s S_t/t = 4\text{KB} \times 96 \times 60 = 23040\text{KB/s} \approx 22.5\text{MB/s}$$

例 6.18 磁盘机的盘组由 9 个盘片组成, 其中专设 1 个盘面为伺服面, 其他盘面为记录数据的盘面。盘存储区域内直径为 6.1cm, 外直径为 12.9cm, 道密度为 180TPM, 位密度为 5000bpm, 平均寻道时间为 12ms, 磁盘转速为 7200RPM。假定 $\pi = 3$, 试计算:

(1) 盘组容量 C_n 是多少字节?

(2) 数据传输率是多少字节/s?

(3) 从任一磁道读取 20000 个字节数据的平均存取时间是多少?

已知: 盘组有 9 个盘片, 专设 1 个盘面作伺服面。盘片外直径 $D_{\max} = 12.9\text{cm}$, 盘片内直径 $D_{\min} = 6.1\text{cm}$, $D_b = 5000\text{bpm}$, $D_t = 180\text{TPM}$, $\bar{t}_s = 12\text{ms}$, $V_{\text{转}} = 7200\text{RPM}$ 。求: (1) 非格式化盘组容量 C_n ; (2) 数据传输率 f ; (3) 读 20000B 的平均存取时间 \bar{t}_a 。

解 由已知得:

$m = 9 \times 2 - 1 = 17$; 一个盘片两个盘面, 伺服面不记录数据所以减“1”。

$$n = (12.9 - 6.1)/2 \times 10 \times 180 + 1 = 6121;$$

$$t = t_{\max} = 60/7200 = 1/120\text{s}.$$

$$(1) C_n = D_b \pi D_{\min} m n$$

$$= 5000 \times 3 \times 6.1 \times 10 \times 17 \times 6121/8 = 11901519000\text{B} \approx 11.084\text{GB}$$

$$(2) f = D_b \pi D_{\min} / t = C_m / t \\ = 5000 \times 3 \times 6.1 \times 10 \times 120 / 8 = 13725000 \text{B/s} \approx 13.089 \text{MB/s}$$

$$(3) \bar{t}_a = \bar{t}_s + \bar{t}_w + \bar{t}_{rw} \\ = 12 + 1/240 \times 1000 + 20000/13725000 \times 1000 = 12 + 25/6 + 1.45719 \\ \approx 17.62 \text{ms}$$

例 6.19 磁盘机的盘组由 4 个盘片组成, 其中专设 1 个盘面为伺服面, 其他盘面为记录数据的盘面。盘存储区域内直径为 4.3cm, 外直径为 8.9cm, 道密度为 100TPM, 位密度为 5000bpm, 磁盘转速为 7200RPM。假定 $\pi = 3$ 。试计算:

- (1) 数据盘面数和柱面数;
- (2) 盘组容量 C_n 是多少字节?
- (3) 数据传输率是多少字节/s?
- (4) 假定系统配备上述磁盘机 12 台, 每个磁道分成 64 个扇区, 试为该磁盘系统设计一个地址方案。

已知: 盘组有 4 个盘片, 专设 1 个盘面作伺服面; 盘片外直径 $D_{\max} = 8.9 \text{cm}$, 盘片内直径 $D_{\min} = 4.3 \text{cm}$; $D_b = 5000 \text{bpm}$, $D_t = 100 \text{TPM}$; $V_{\text{转}} = 7200 \text{RPM}$, 系统配置 12 台磁盘。求:(1) 数据盘面数 m 和柱面数 n ; (2) 盘组容量 C_n ; (3) 数据传输率 f ; (4) 系统配备磁盘机 12 台, 为磁盘系统设计一个地址方案。

解 由已知得: $t = t_{\text{max}} = 60/7200 = 1/120 \text{s}$

(1) $m = 4 \times 2 - 1 = 7$; 一个盘片两个盘面, 伺服面不记录数据所以减“1”。

$$n = (8.9 - 4.3)/2 \times 10 \times 100 + 1 = 2301$$

$$(2) C_n = D_b \pi D_{\min} m n$$

$$= 5000 \times 3 \times 4.3 \times 10 \times 7 \times 2301 / 8 = 1298626800 \text{B} \approx 1.2094 \text{GB}$$

$$(3) f = D_b \pi D_{\min} / t$$

$$= 5000 \times 3 \times 4.3 \times 10 \times 120 / 8 = 9675000 \text{B/s} \approx 9.23 \text{MB/s}$$

(4) 磁盘地址方案: 依已知系统配备磁盘机 12 台, 一个磁道划分为 64 个扇区以及计算得出的磁盘具有 2301 个柱面, 7 个盘面。可计算台号、柱面、盘面和扇区四个部分地址的位数:

台号位数: $\lceil \log_2 12 \rceil = 4$ 位; 柱面位数: $\lceil \log_2 2300 \rceil = 12$ 位;

盘面位数: $\lceil \log_2 7 \rceil = 3$ 位; 扇区位数: $\lceil \log_2 64 \rceil = 6$ 位

磁盘地址方案见图 6.13。

台号	柱面号	盘面号	扇区号
24…21	20…9	8…6	5…0

图 6.13 磁盘系统地址方案示意图

例 6.20 有一双面软盘, 每面有 77 道, 转速为 360RPM, 对任何磁道均可以 250kb/s 的速度读写, 每道分成 16 个扇区。试计算:

- (1) 该盘最大容量为多少字节?

- (2) 平均等待时间是多少?
- (3) 每扇区最大字节数是多少?
- (4) 若位密度为 3200bpi, 磁道最内圈直径是多少?

已知: 盘面数 $m = 2$, 柱面数 $n = 77$, $V_{\text{转}} = 360\text{RPM}$, 数据传输率 $f = 250\text{kb/s}$, $D_b = 3200\text{bpi}$, $S_t = 16$ 。求:(1) 该盘最大容量 C_n ; (2) 平均等待时间 t_w ; (3) 每扇区最大字节数 B_s ; (4) 磁道最内圈直径 D_{min} 。

解 由已知 $V_{\text{转}} = 360\text{RPM}$ 可得 $t = 60/360 = 1/6\text{s}$ 。

$$\begin{aligned} (1) C_n &= f t m n \\ &= 250\text{kb} \times 1/6 \times 2 \times 77/8 = 802.08\text{KB} \end{aligned}$$

(2) 因平均等待时间为最大等待时间 $t_{w\max}$ 即 t 的一半, 所以有

$$t_w = t/2 = 1/6/2 = 1/12\text{s} \approx 83\text{ ms}.$$

$$(3) B_s = f t / 16 = 250\text{kb} \times 1/6/16 = 2.6\text{ kb}.$$

(4) 因为 $f t = D_b \pi D_{\text{min}}$, 所以有

$$D_{\text{min}} = f t / D_b \pi = 250\text{kb} \times 1/6 / (3.14 \times 3200) \approx 4.25\text{inch}.$$

例 6.21 现代磁盘传输率高、容量很大、可靠性高, 试述现代磁盘采用的新技术。

解 详见本章第 6 个知识点“磁盘存储器”的第 5 个问题。

例 6.22 按存储介质光盘分为哪几类? 各类有何特点?

解 按记录介质光盘分为形变型、相变型和磁光型三种。

(1) 形变型光盘: 通过记录层表面形状发生变化记录信息, 如凹凸、发泡与否等。特点是一经写入, 就不能再修改。原因是形状改变了, 不能再复原。

(2) 相变型光盘 PCD(Phase Change Disc): 通过记录层发生相变记录信息。有两类相变介质。一类是不可逆相变介质, 只能写一次, 不能擦除。用于制造追记型光盘; 另一类是可逆相变介质, 可擦写多次, 用于制造可擦型光盘。

(3) 磁光型光盘 MOD(Magneto Optical Disc): 通过磁记录层的两种剩磁化状态记录信息。特点是磁效应写入, 光磁效应用读出; 可多次擦写。

例 6.23 按存取方式光盘分为哪几类? 各类有何特点?

解 按存取方式光盘可分为只读型、可追记型和可擦写型三种。

(1) 只读光盘: CD-ROM、VCD 和 DVD-ROM 均属此类。

特点: 出厂时信息已写好, 只能顺序读出, 不能写入和擦除; 光道为由里向外的一条螺旋线, 段(扇区)长度、位密度均相同; 工作时线速度相同, 角速度时刻变化, 为恒线速盘; 一般采用形变型介质, 盘径为 4.75inch。

(2) 追记型光盘: WORM 和 CD-R 属此类。

特点: 用户只能写一次, 一经写入, 便不能更改; 光道可为螺旋线亦可为同心圆。采用不可逆相变介质或形变型介质。WORM 盘径为 12inch 或 5.25inch; CD-R 为 4.75inch。

(3) 可擦写型光盘(Rewritable): MOD 和可逆 PCD 属此类。

特点: 可读、可擦除、可多次写入; 光道为同心圆; 恒角速盘, 线速度不同; 采用可逆相变介质或磁光型介质。

例 6.24 形变型光盘、相变型光盘和磁光盘各是怎样写入和读出信息的?

解 (1) 形变型光盘读写信息原理

写入:高功率强激光束照射要写入区域,使记录层溶化、蒸发,形成凹坑存储位元。

读出:使用低功率弱激光束,照射盘面。有凹坑处,因粗糙吸收光多,故反射率低;无凹坑处,因光滑吸收光少,反射率高。从而区别信息“0”和“1”。注意,读出时的弱激光不改变盘面形态。

(2) 相变型光盘读写信息原理

① 可写一次型光盘读写信息原理

写入:高功率激光照射要写入区域,晶态变为非晶态,冷却后维持非晶态,无激光照射处维持原来的晶态。

读出:低功率激光束照射盘面,不改变记录薄层状态。晶态,反射光强;非晶态,反射光弱。依反射光强、弱判断读出“1”和“0”信息。

② 可重写型读写信息原理

擦除:中功率激光束照射盘面 50ms 以上,使非晶态恢复为晶态。

写入:高功率激光束照射要写入区域,使晶态变为非晶态。冷却后维持非晶态,无激光照射处维持原来的晶态。

读出:低功率激光束照射盘面,不改变盘面状态,依反射光强、弱判断读出“1”和“0”信息。

(3) 磁光盘读写信息原理

擦除:写前,强激光照射光道所有记录信息处,在稳定磁场 H_w 作用下,使磁层都处于“0”磁化状态。

写入:强激光束照射写“1”处,温度升高,磁头产生与擦除时大小相等,方向相反的磁场。使介质磁化翻转,写入“1”。激光未照射处,仍保持“0”磁化状态。

读出:低功率激光经起偏器变成偏振光,偏振光照射在不同磁化方向的介质上,发生克尔磁光效应,即反射光偏振面发生不同方向的偏转,通过检偏器取出某种磁化方向的偏振光,从而读出“1”或“0”。

例 6.25 回答如下问题:

(1) 如下存储装置中,主存储器、Cache、通用寄存器、硬磁盘、软磁盘、光盘和磁带都可用来存储信息,试按存取时间由快到慢排序。

(2) 如下存储装置中,主存储器、Cache、CPU 通用寄存器、固定头磁盘、移动头固定盘组磁盘和磁带都可用来存储信息,试按存取容量由小到大排序。

解 (1) 通用寄存器、Cache、主存储器、硬磁盘、光盘、软磁盘、磁带。

(2) CPU 通用寄存器、Cache、主存储器、固定头磁盘、移动头固定盘组磁盘、磁带。

例 6.26 如下存储装置分别属于哪种存取方式的存储器:DRAM 存储器、汉字库、Cache、CPU 通用寄存器、固定头磁盘、游戏卡、移动头磁盘、控制存储器 CM、磁光盘、软磁盘、VCD 光盘和磁带。

解 DRAM 存储器、Cache、CPU 通用寄存器为随机存取存储器;固定头磁盘、移动头磁盘、磁光盘、软磁盘为直接存取存储器;汉字库、游戏卡、控制存储器 CM 为只读存储器;

VCD 光盘和磁带为顺序存取存储器。

例 6.27 多体低位交叉编址并行主存系统有何特征？同单体多字并行主存系统比较，有何优点？

解 多体低位交叉编址并行主存系统有如下特征：每个体都有自己的地址寄存器 MAR，地址译码器和数据缓冲寄存器 MBR，都自成体系能独立读写；多个体在存控的控制下，实现重叠交叉并行存取。

同单体多字并行主存系统比较优点是，所有体可同时读写，主存数据传输率高；因可读写其中的 1 个体，2 个体，……，n 个体，所以读写灵活；因该主存连续的系统地址以 n 为模指向不同的存储体，因此 n 越大，访存冲突越小。所以共享主存的计算机系统都采用低位交叉编址并行主存系统。

例 6.28 提高存储器速度可采取哪些措施？试简要说明。

解 提高存储器速度通常从存储技术本身和存储器结构两个方面入手。

从存储技术本身提高内存速度可采用高速电子元器件、线选法构造存储矩阵和采用新型存储芯片。而对于外存则应采用高密度的记录介质、高密度的记录磁头和光头以及高转速电机和高速度高精度的磁、光道定位部件。

从存储结构方面提高内存存储器速度可采用双端口、多端口存储器，单体多字并行主存和多体交叉编址并行主存系统。提高外存储器速度可采用“磁盘 Cache”和构造磁盘子系统即廉价冗余磁盘阵列 RAID。

例 6.29 为什么要把存储系统分成若干存储层次？主要有哪两个层次？它们各自解决什么问题？

现代计算机要求其存储系统速度快、容量大和位价格低，而在物理的实现上，又存在着“速度越快，位价格越高；容量越大，速度越慢；只有容量做得大，位价格才能降低”的矛盾。为达到要求、解决矛盾，唯一可行方法就是采用多种存储技术构成具有层次结构的存储系统。

现代计算机的存储系统主要有 Cache、主存和辅存两个存储层次。前者解决存储系统的速度问题，后者解决存储系统的容量问题。

例 6.30 现代计算机存储系统为什么采用层次结构？试说明 Cache、主存和辅存两个层次在功能上、技术上和实现方法上有何不同？

对存储器的三个要求是速度快、容量大和位价格低。而在实现技术中发现：速度越快，位价格就越高；容量越大，速度就越慢；只有容量做得大，位价格才能降低。因此，要实现“容量大、价格低”的要求，应采用能提供大容量的存储技术，如磁表面存储器和光盘存储器等；欲满足“速度快”的要求，则应采用昂贵且容量小的快速存储技术，如半导体存储器。很显然，仅采用一种存储技术，存储器的设计就会陷入困境，因为对存储器的三个要求是相互矛盾的。

众所周知，计算机是通过执行程序来完成一切工作的，而程序的执行又符合局部性规律，即程序局部性原理。也就是说，在一个较短的时间间隔内，程序执行涉及到的指令和

数据的存放地址是簇集的,而不是随机分布的。

正是基于上述两个现实,所以现代计算机组成层次结构的存储系统。即使用双极型半导体器件构造小容量高速缓存 Cache、使用 MOS 型半导体器件构造较大容量的主存、使用光和磁存储材料构造大容量或海量的磁表面存储器和光盘存储器并把它们有机地组合起来,构成层次结构的存储系统。

在功能上,Cache、主存层次解决存储器速度慢的问题;而主存、辅存层次则解决容量小的问题。

在技术上,Cache、主存层次完全使用硬件,因此信息交换很快。对所有程序员都是透明的;而主存、辅存层次则主要运用软件,所用硬件很少,所以信息交换慢。仅对应用程序员是透明的。

在实现上,Cache 由双极型或 SRAM(小、微型机用)存储器件构成,通常容量小,速度快;主存由 DRAM 存储器件构造,容量大,速度稍慢;辅存采用价格便宜、容量很大,甚至是无限的、速度很慢的磁盘、光盘或磁带。

第六章 输入输出控制

例 7.1 解释下列术语:接口,标准化 I/O 接口,并行接口,串行接口,I/O 子系统,中断源,中断,中断系统,内中断,外中断,单重重断,多重中断,中断优先权,禁止中断,自愿中断,中断隐指令,中断屏蔽,向量地址,周期挪用,字节多路通道,选择通道,数组多路通道,通道控制字,通道状态字,通道程序。

解 (1) **接口:**两个不同系统的交接部分。如两种硬设备之间的接口是由电子线路构成的一个逻辑部件;而两个程序块之间的接口是一个程序。

(2) **标准化 I/O 接口:**若在结构尺寸、接插连接、信号电平、逻辑电路和传输总线等方面均采用统一格式的 I/O 接口,称之为标准化 I/O 接口。

(3) **并行接口:**和主机、外设都以并行方式传送信息的接口。

(4) **串行接口:**以并行方式和主机、而与外设则每次传送一位的方式传送信息的接口。

(5) **I/O 子系统:**完成 I/O 操作的所有软件和硬件。

(6) **中断源:**引起处理机中断的事件。

(7) **中断:**CPU 暂时中止现行程序,转去执行“处理随机发生的紧急事件或特殊请求”的程序,处理完后自动返回被中止程序继续运行的功能。

(8) **中断系统:**计算机实现中断功能的软硬件总称。

(9) **内中断:**主机内部硬件和软件原因引起的中断。

(10) **外中断:**主机以外的部件引起的中断。

(11) **单重重断:**执行中断服务程序时,不允许再响应新的中断请求。

(12) **多重中断:**执行某个中断服务程序时,还可响应优先级别高的中断请求。

(13) **中断优先权:**中断响应的优先次序。

(14) **禁止中断:**CPU 在某些时刻不允许响应中断,称为禁止中断。一般用硬件实现。

(15) **自愿中断:**在程序中预先安排的由广义指令引起的中断。

(16) **中断隐指令:**响应中断时才由硬件产生以便完成中断响应的各项工,因它像一条指令,但 CPU 指令系统中又无此指令,故称中断隐指令。

(17) **中断屏蔽:**用程序方法有选择性的封锁部分中断,使之不发出中断请求,而允许其余部分中断发出中断请求并得响应,称这种功能为中断屏蔽。

(18) **向量地址:**访问中断向量表所需的地址,亦称中断指针。

(19) **周期挪用:**亦称周期窃取。当 DMA 有请求时,CPU 暂停访存,DMA 占用一个或若干个存储周期,称该过程为周期挪用。

(20) **字节多路通道:**连接控制多台慢速外设,以字节交叉方式传送数据的通道。

(21) **选择通道:**连接控制一台或多台同种高速外设,以成组方式顺序传送数据的通道。

(22) **数组多路通道:**连接控制多台高速外设,以成组交叉方式传送数据的通道。

(23) **通道控制字:**规定通道及其连接的外设执行什么操作和如何执行操作的命令字。

(24) **通道状态字:**反映通道、子通道和外设状态的一组二进制的信息。

(25) **通道程序:**通道命令字的有序集合。

例 7.2 何谓外设？按功能外设是如何分类的？

解 外设：安装在主机周围，与主机进行信息交换，并改变信息形态的装置。

按功能，外设可分为：

(1) 输入设备：如键盘、扫描仪、鼠标等。

(2) 输出设备：如显示器、打印机、绘图机以及缩微胶卷照相装置等。

(3) 外存储器：如磁表面存储器、光盘、快擦写存储器 Flash 构成的“固态盘”。

(4) 模数转换设备：即 A/D 和 D/A 设备。

(5) 网络通信及终端设备：如调制解调器、网络桥接器、智能终端等。

例 7.3 试述外设在计算机系统中的地位与作用？

解 (1) 外设是计算机系统重要的组成部分，相当于计算机的五官四肢。没有外设，计算机不能运转；配备不全或性能不好，计算机效率低。

(2) 人机通信和对话的工具

程序、数据和命令要送入计算机，计算机的运算结果、运行状态要输送出来，都要通过外设来实现，它是人机对话的唯一通道。

(3) 外设是完成数据媒体变换的装置

计算机的控制流与数据流都是用电信号表示的二进制代码，而人却习惯于用字符、图形、图像等表达信息的含义。这种变换只能由外设来完成。

(4) 外设是系统软件及信息的驻在地

操作系统和工具软件以及用户程序在执行前和关机后一般都是存储在外存中。当今计算机系统，外存的有无、性能的优劣是衡量系统性能的重要标志之一。

(5) 外设是计算机推广应用的桥梁

计算机普及与应用促进了外设的发展，而新型外设的出现又拓宽了计算机在不同领域的推广应用。

例 7.4 什么是 I/O 控制？其主要功能是什么？

解 I/O 控制：主机对输入和输出操作进行硬件和软件的控制，称之为 I/O 控制。

I/O 控制的主要功能有：

(1) 控制外设的各种动作，如启动、停止、输纸、定位等；

(2) 组织协调各外设分时享用传送信息的硬件和软件，从而控制主机与外设、外设与外设之间的并行工作；

(3) 平衡外设与主机之间的数据流量，组织数据交换，使单位时间的数据流量不因过大而造成数据丢失，也不因过小而使设备得不到充分利用；

(4) 向主机报告外设的状态，使主机合理安排程序转移；

(5) 对外设产生的数据错误和数据传送中的错误，进行检测与校正。

(6) 组织并实施对外设及 I/O 子系统的检查、诊断和维护。

例 7.5 试说明 I/O 操作和 I/O 设备的特点。

解 (1)I/O 设备的特点

①慢速性:外设是机电、机械设备装置,其速度为 μs 、 ms ,甚至为 s 级;而主机,尤其是 CPU 速度却是 ns 级。

②多样性:外设品种多、其工作原理涉及机械、光电、磁、声、自动控制、通信等多个学科领域多。

③复杂性:外设体积大小悬殊,工作原理相差极大。简单的如鼠标,复杂的如磁盘、光盘、激光打印机。

(2)I/O 操作的特点

①异步性:主机与外设、外设与外设速度差异很大,导致 CPU 对外设的控制、外设对 CPU 的请求都将是随机的、异步的。

②实时性:外设速度慢,但一经启动,则以固定速率工作,要求与主机在规定的时间内完成信息交换。

③I/O 操作的实现与设备的无关性:外设虽多种多样,但 I/O 操作应尽量标准化,使 I/O 操作的实现与设备无关。即能通过简单的命令即可使用外设并完成 I/O 操作,尽可能少地考虑外设的内部细节。

例 7.6 I/O 组织有哪些基本原则?为什么规定这些原则?

解 I/O 组织的基本原则,就是构造 I/O 子系统的基本原则。依据 I/O 操作的异步性、实时性和 I/O 操作的实现与设备的无关性以及 I/O 设备的慢速性、多样性和复杂性特点,为了充分发挥主机快速性的效能和提高主机效率,I/O 组织应遵循下述三个基本原则:

(1)自治控制原则:将输入输出功能尽可能从 CPU 中分离出来,由专门的部件去完成。使 CPU 致力于高速运算处理操作。

(2)分类原则:根据不同性质的外设,特别是工作速度的快慢,分类组织 I/O 控制。

(3)层次结构原则:将 I/O 子系统的功能按不同的层次分布,标准的操作及控制功能放在与主存及 CPU 相连的层次上,即系统级接口中;而非标准的操作及控制功能放在与设备相连的层次上,即设备级接口中。

例 7.7 四级和三级 I/O 子系统各是怎样构成的?各级的功能怎样?各有何特点?

解 (1)四级 I/O 子系统

因由 CPU 中的一部分、I/O 通道、设备控制器和外围设备四部分组成而得名,每部分称为一级。各级的功能分别为:

①CPU 中的一部分:执行启动、查询通道和外设 I/O 指令,并处理来自 I/O 通道的中断请求。

②I/O 通道:四级 I/O 子系统的核心。在 CPU 启动通道后,它代替 CPU 调度、管理和控制连接于该通道的所有外设。

③设备控制器:控制一个或多个外设完成数据读写或其他辅助操作的逻辑部件。不同设备的控制器虽不相同,但均以标准的方式与通道连接。

④外围设备:指 I/O 设备、外存储器和终端设备等。它可以是一台外设,也可以是另一台计算机或终端用户。

四级 I/O 子系统特点:因 I/O 通道代替 CPU 调度、管理和控制连接于该通道的所有外设。CPU 仅在数据传送开始前和传送结束时做些工作,而在数据传送期间,对 CPU 的打扰很少,所以 CPU 的效率得以充分发挥。但其所需硬件较多,还需专门配备通道 I/O 指令。

(2) 三级 I/O 子系统

因由设在 CPU 中的 I/O 逻辑、I/O 接口和外设及其控制器三部分组成而得名,每部分称为一级。各级的功能分别为

①I/O 逻辑:发出启停、查询、使用外设的 I/O 的指令,接收并处理接口请求。

②I/O 接口:将 CPU 启停、查询、使用外设的 I/O 指令送外设;控制数据交换;接收、寄存 I/O 状态和请求并送 CPU。它是三级 I/O 子系统的核心,I/O 接口指的是程序直接控制传送、程序中断控制传送或 DMA 传送接口。

③外设及其控制器:控制一台外设实现 I/O 操作及辅助操作。

三级 I/O 子系统特点:I/O 接口可为程序直接控制传送、程序中断控制传送或 DMA 传送接口。在数据传送过程中,它们对 CPU 打扰多,程序直接控制传送尤甚。但它缩小了

体积,降低了造价,在小、微型计算机系统中常被采用。

例 7.8 I/O 接口有哪些功能?其构成与哪些因素有关?

解 (1) I/O 接口的功能

①寻址功能:从多台外设中选择一台使用。

②数据格式转换和电平变换功能:完成串、并转换和信号电平的变换。

③数据缓存和传送数据的功能:因接口处于主机与外设之间,所以必须具有数据通路并能完成数据传送。因主机、外设速度差异,还应设置数据缓冲器。

④提供外设和接口状态功能:随时提供外设和接口状态,便于主机使用和控制外设。

⑤实现主机与外设的通信和控制功能。

(2) I/O 接口构成与下述因素有关

①I/O 控制的类型:在程序直接控制传送、程序中断控制传送、DMA、I/O 通道和 I/O 处理机五类 I/O 控制传送方式中,控制类型的不同,I/O 接口构成将存在很大差别。

②数据传送宽度:选择串行或并行传送,不但数据缓冲寄存器的结构不同,而且还会影晌相应的控制逻辑。

③通信控制方式:采用同步或异步通信控制方式,将影响 I/O 接口组成。前者在发送端与接收端要有统一的时钟,后者则无需有统一的时钟。前者控制逻辑简单,后者控制逻辑复杂。

④传送信息的种类:传送信息的种类将因外设不同而异,它必将影响 I/O 接口组成。所有外设都须传送设备地址、数据和设备状态及控制信息,而状态及控制信息各设备不一定相同。有的设备还需提供设备内部地址,如磁盘的柱面、盘面和扇区号等。

例 7.9 外设有几种编址方法?各有何特点?

解 外设有独立编址和统一编址两种编址方法。

(1) 独立编址方法

独立编址亦可称为非统一编址。其特点是仅对计算机系统的所有外设编址,即对计算机系统的每个外设给一个唯一的编号,该编号又称为外设地址。借助于这些编号,CPU 通过专设的 I/O 指令组,使用和控制系统中的所有外设。换句话说,I/O 指令中专设外设地址字段指出所用设备。

(2) 统一编址方法

统一编址亦可称为非独立编址。其特点是把 I/O 操作和主存储器的读写同等对待，将外设及其接口中允许主机访问的寄存器和主存的存储单元一样对待，统一编址。这样 CPU 就可用访存指令访问外设的寄存器，不需要专设 I/O 指令组，使输入输出程序设计十分灵活。缺点是外设占据主存的部分地址空间，使之成为专用，非输入输出程序设计不能再使用。加之外设地址和主存地址一样长，使得外设地址选择电路也变得复杂了。

例 7.10 结合程序查询方式接口，说明其工作过程。

解 图 7.6 给出程序查询方式接口，下面依该接口说明其工作过程。传送前，用软件方法设置存取信息的主存始地址和要传送数据的个数，并用 I/O 指令查询外设是否闲置良好，若闲置，则顺序执行以下各步；否则，执行其他程序。

- (1) I/O 指令启动外设即将 SCR 中的“忙闲”触发器置“1”，并将命令码送 SCR 中；
- (2) 通过 I/O 查询指令，了解外设是否准备就绪，即判断 SCR 中的“就绪”是否为“1”；
- (3) 如外设未准备就绪，即 SCR 中的“就绪”触发器为“0”，则踏步等待即重复(2)，否则，往下执行(4)；
- (4) CPU 执行 I/O 指令传送数据，若为输入，CPU 从接口的 DBR 中取走单位数据，若为输出，CPU 再输出一个单位数据至接口的 DBR，同时将接口中的“就绪”标志清“0”，“忙闲”标志置“1”；
- (5) 修改主存地址计数器和交换个数计数器，并判交换个数计数器是否为零；
- (6) 若交换个数计数器不为零，重复(1)~(5)，直至交换个数计数器为零即一批数据交换完毕，结束 I/O 传送，继续执行其他程序。

例 7.11 中断在计算机系统中有何作用？

解 (1) 实现主机和外设的并行工作

中断用于控制 I/O 的数据传送，使得主机、外设，外设和外设有一定程度的并行，从而提高了计算机系统的工作效率。

(2) 处理故障

计算机出现故障时通过发中断请求，启动中断系统工作，调用相应处理程序，将故障的危害降低到最低程度，从而提高了计算机系统的可靠性。

(3) 实现多道程序和分时操作

通过分配每道程序一个固定的时间片，利用时钟定时发中断实现多道程序的分时操作；亦可由各道程序自己产生中断实现程序的切换和分时操作。

(4) 实现实时控制

所谓实时控制，是指在某个事件或现象出现时及时地进行控制。而这些事件是随机出现的，不能预见的。只有通过中断系统，才能达到控制受控对象的目的。

(5) 实现人机联系与通信

计算机运行过程中，需要随机地干预机器，如抽查中间结果，了解工作状态，下达临时命令等。在没有中断系统的计算机中这些功能几乎是无法实现的。而利用中断系统进行人机通信既有效又方便。

例 7.12 何谓中断判优？说明有几种方法？各有何特点？

解 (1)中断判优：通过中断排队，从中选出优先中断源的过程，称之为中断判优。

(2)中断判优通常有硬件判优和软件判优两种方法。

①软件判优是通过执行中断查询程序找到优先中断源，并立即进入相应的中断服务程序。其特点是仅需很少的硬件，但需执行程序所以响应中断的速度慢，只能用于中断源少、速度慢的计算机系统中。

②硬件判优是通过硬件判优线路找到优先中断源，同软件判优比较，其速度快，但费器材。目前广泛采用的有两种。一种为最左判优线路，见图 7.7。它是广泛使用的一种并行中断判优线路，速度快，但费器材。另一种是链型判优线路，见图 7.8。它属于串行中断判优线路，通常用于 I/O 中断的判优，构成链型判优线路的各“环”分布在相应中断接口的逻辑中。这样的构造既给系统扩充外设提供极大方便，又较最左判优省器材。

例 7.13 采用程序中断方式传送的接口应由哪些部分构成？各部分功能是什么？

解 程序中断方式传送的接口见图 7.11。其构成及功能如下：

(1)设备选择器：接收 CPU 发出的设备地址，选择和控制相应外设及接口实现 I/O 操作。

(2)数据缓冲器：寄存输入或输出的单位数据。

(3)状态与控制命令寄存器：寄存两部分内容。一部分存放 CPU 送来的指明接口及外设操作方式的控制命令，如读、写、走纸等。另一部分记录外设及接口的状态，如“忙闲 BUSY”“就绪 DONE”“中断请求 INTR”“中断屏蔽 MASK”以及“故障”等。

(4)中断控制逻辑：实现程序中断控制传送必需的逻辑线路。通常有中断请求信号的产生逻辑、中断屏蔽和中断选优判优逻辑、中断类型码回送逻辑以及面向外设的特殊操作控制逻辑等。

例 7.14 CPU 响应中断的条件是什么？响应中断时应做哪些工作？

解 (1)CPU 响应中断的条件有 4 个，它们分别是：

①有中断请求；

②CPU 的一条指令执行完毕；

③无更紧迫的任务或事件，如无 DMA 传送、掉电中断；

④CPU 允许响应中断，可通过设一个中断允许触发器 INT 实现。

(2)中断响应完成的任务

①暂停现行程序；

②进入中断服务程序。

注意：中断响应的任务，一般通过执行中断隐指令实现。

例 7.15 整个程序中断过程中，哪些工作由硬件完成？哪些工作由软件完成？哪些工作既可由硬件，也可由软件完成？

解 中断过程中需做五个方面的工作即需解决五个主要问题。

(1)接收、记录各中断源产生的中断，并确定可否向 CPU 提出中断请求问题。必须使用硬件解决该问题。

(2)进行中断判优，选取优先中断源的问题。软、硬件均可解决，但软件速度慢，很少采用。

- (3) 实施中断响应的问题。必须通过硬件解决。
- (4) 实现中断处理的问题。必须软件完成,即通过执行中断服务程序实现。
- (5) 返回被中止的程序继续执行的问题。通过软件即在中断服务程序编写中断返回指令实现。

例 7.16 说明向量地址与中断服务程序入口地址的区别与联系。

解 向量地址:访问中断向量表所需地址。

中断向量表:所有中断服务程序入口地址组成的一维表格。通常存放在主存的一片连续的存储空间中。向量地址就指向这一片连续存储空间的某个地址。

中断服务程序入口地址:中断服务程序第一条指令的地址,是向量地址中的内容。换句话说,向量地址是中断服务程序入口地址的地址。

例 7.17 以键盘为例,结合中断接口线路,说明程序中断传送的工作过程。

解 详见本章第七个知识点“程序中断控制传送”中的程序中断控制传送举例。

例 7.18 DMA 工作方式有哪几种?各有何特点?

解 DMA 工作方式有下述三种。

(1) 全串行方式

① 处理方法:外设要传送,发请求信号给 CPU,CPU 暂停访存,一批数据传输完毕后,CPU 再继续访存。

② 特点:控制简单;存储器效率和 CPU 效率均未得到发挥,因为最高速外设单位数据的准备时间再小也大于 T_M 。

(2) 周期挪用方式:亦称周期窃取,简单中断。DMA 大多采用这种方式。

① 处理方法:外设要传送,CPU 暂停一个访存周期给 DMA。外设交换一个单位数据后,CPU 继续进行访存。

② 特点:这种方式 CPU 和外设访存时,有可能冲突,也可能不冲突。所以存储器和 CPU 效率均较高,但控制复杂。

(3) 交替访存方式:亦称透明 DMA

① 处理方法:CPU、DMA 各有自己的主存地址寄存器 MAR 和存储缓冲寄存器 MBR,CPU、DMA 两者交替访存。

② 特点:DMA 对 CPU 无影响,CPU 感觉不到 DMA 的存在,主机也不停止程序运行,效率高;主存的效率降低,且要求主存存储周期小,即主存速度快。

例 7.19 在周期挪用的 DMA 方式中,每交换一个单位数据,外设实现上也要求中断主机一次。这种中断与程序中断有何不同?

解 程序中断控制传送,每交换一个单位数据,在实现上要求中断主机一次。这个中断要求中止现行程序,转去执行一次中断服务程序,完成一个单位数据的交换,而后再返回被中止的程序继续执行。注意,该中断使 CPU 暂停现行程序,转去执行一次中断服务程序。CPU 需要保存返回点、保存现场、中断服务、恢复现场等一系列的工作,占用 CPU 的时间多。

周期挪用 DMA 方式,每交换一个单位数据,在实现上也要求中断主机一次。但这个中断只是要求 CPU 暂停一个访存周期,在这个访存周期中,外设与主存间交换一个单位数据,而后 CPU 继续访存。注意,此中断 CPU 仅暂停一个访存周期,无任何其他额外操作,占用 CPU 的时间少。因此,为与程序中断控制传送的中断相区别,又称周期挪用 DMA 实现交换一个单位数据时的中断为简单中断。

例 7.20 周期挪用 DMA 接口由哪几部分构成？各部分功能是什么？

解 周期挪用 DMA 接口由接口寄存器、中断传送标准接口和 DMA 控制逻辑三部分组成，见图 7.15。组成及各组成部分的功能简述如下：

(1) 接口寄存器

- ① 数据缓冲寄存器 DBR：寄存要传送的单位数据信息
- ② 地址缓冲寄存器 ABR：亦称主存流动地址。寄存主存缓冲区始地址。
- ③ 字计数器 WC：寄存传输长度，即要交换的单位数据个数。
- ④ 状态寄存器 DSR：寄存设备及接口状态。

⑤ 设备控制寄存器 DCR：寄存 CPU 启动接口及设备的有关控制命令，如读、写或寻道等。有的接口将状态和控制两个寄存器合为一个寄存器。

⑥ 设备内部地址寄存器 DAR：寄存设备内部地址。如磁盘的柱面号、盘面号、扇区号。有的计算机将该寄存器作为设备控制器的组成部分。

(2) 中断传送标准接口

构成见图 7.12。功能有二：一是当 WC 产生溢出说明一批数据传送完或传输中出现故障，由它向 CPU 发出中断请求，以便 CPU 完成 DMA 传送的结束处理；二是 DMA 传送前的所有准备工作也要借助于它实现，如：查询、启停外设，设置传输参数等。

(3) DMA 控制逻辑

功能是控制 DMA 的数据传送过程。由单位数据准备就绪、DMA 请求和响应等多个触发器，DMA 优先排队、DMA 向 CPU 发请求以及回送命令码等线路组成。

例 7.21 DMA 接口有几种类型？说明它们的组成及异同点？

解 DMA 接口有四种。它们是经典的周期挪用 DMA 接口（见图 7.15），广义型 DMA 传送接口（见图 7.21），选择型 DMA 传送接口（见图 7.16）和多路型 DMA 传送接口（见图 7.22）。

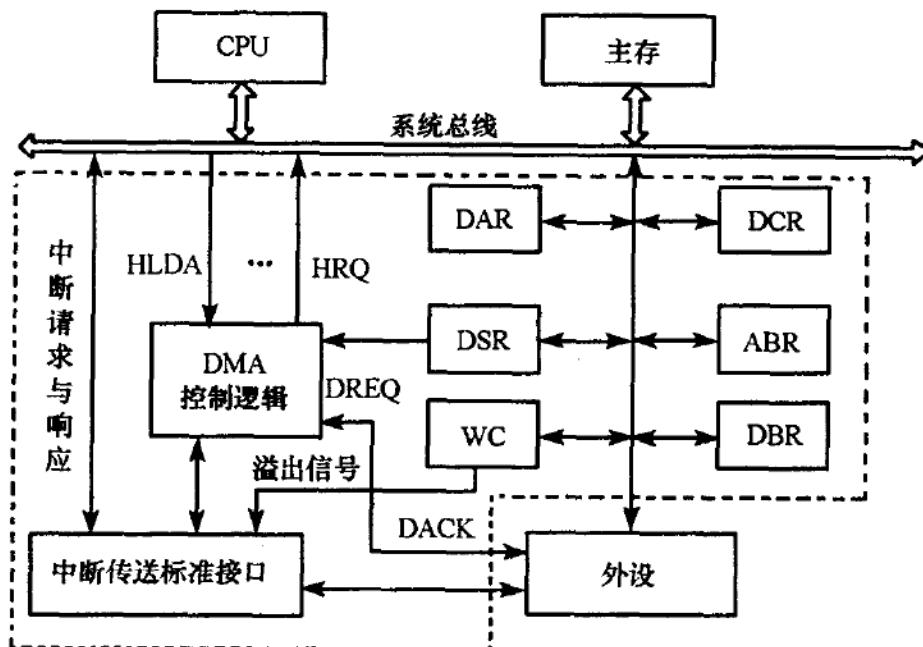


图 7.15 周期挪用 DMA 传送接口基本组成

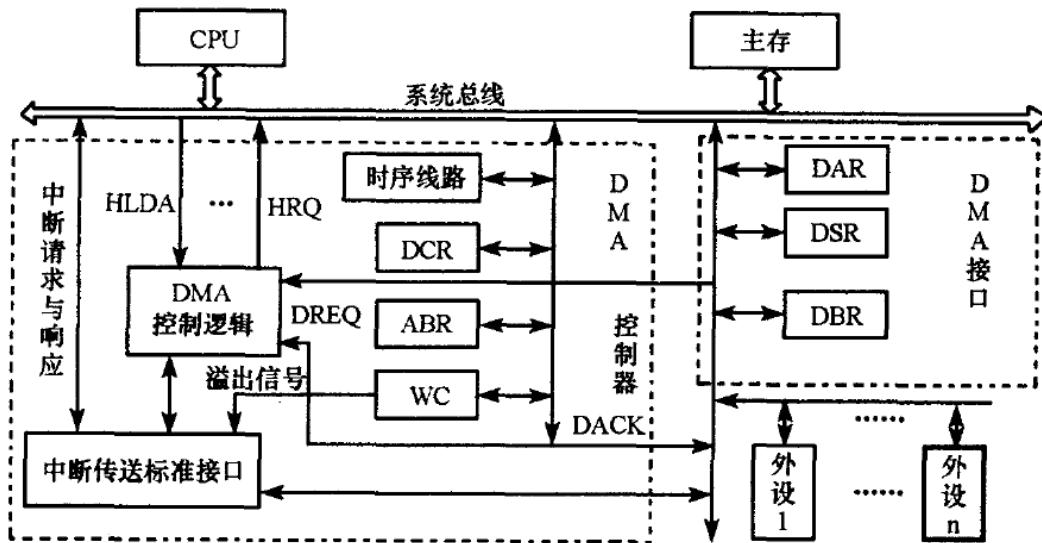


图 7.16 选择型 DMA 传送接口组成

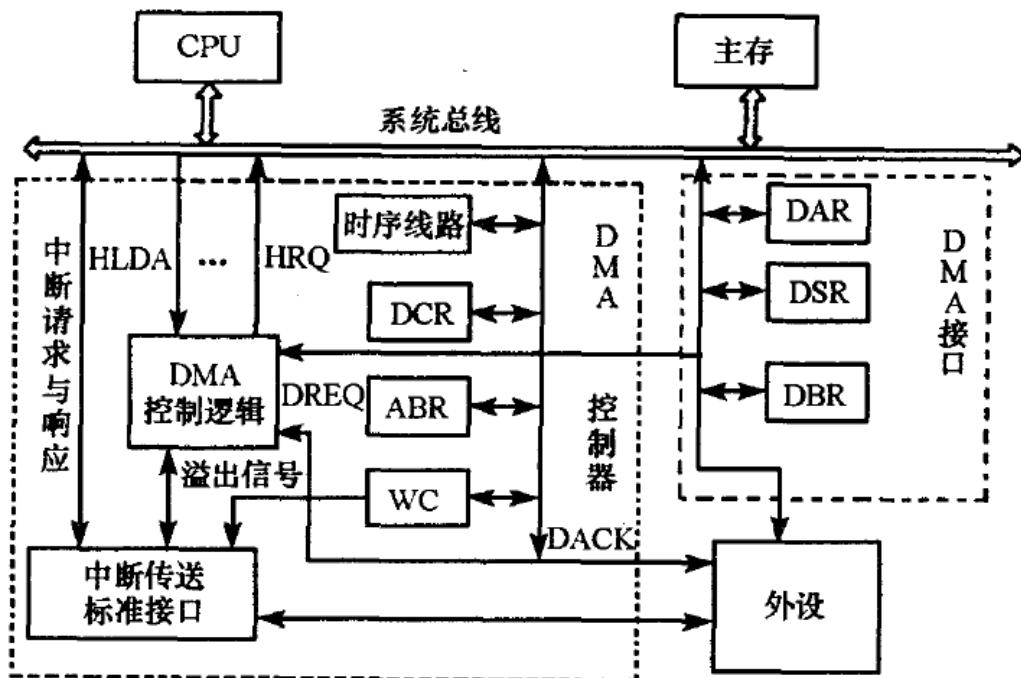


图 7.21 广义型 DMA 传送接口组成示意图

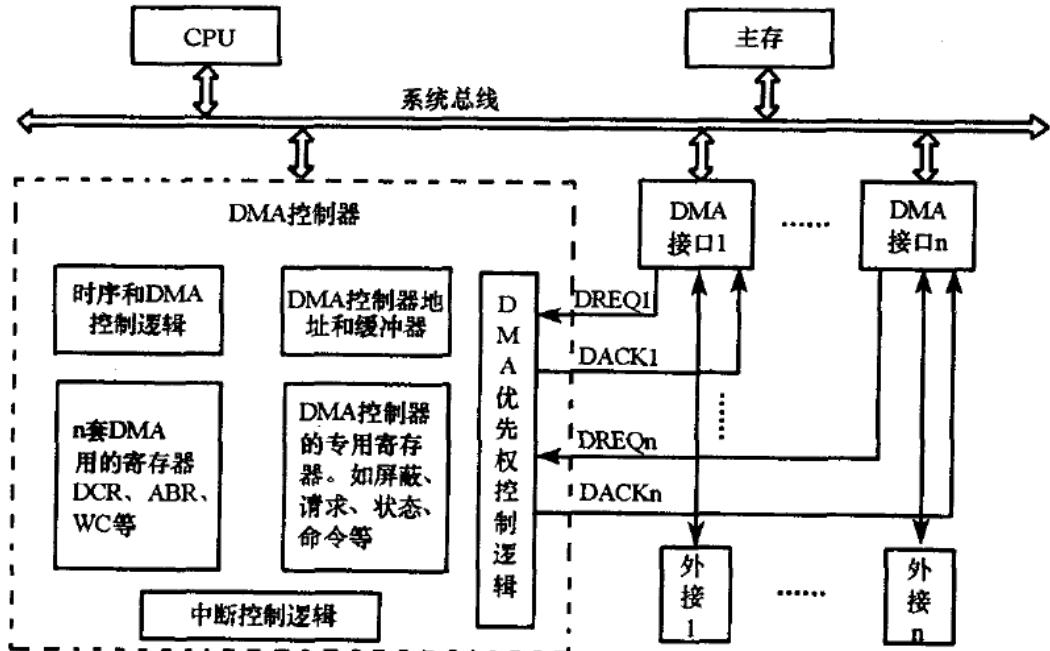


图 7.22 多路型 DMA 传送接口组成示意图

为了设计标准化、模块化和与具体外设尽量无关,现代计算机采用大规模集成电路技术,把经典周期挪用 DMA 传送接口进行修改,分立为 DMA 控制器和 DMA 接口两个部件。称这样的 DMA 接口为广义型 DMA 接口,如图 7.21 所示。

DMA 控制器是公共逻辑,与具体外设无关,只要以 DMA 方式与主机交换信息,就必须配置 DMA 控制器。DMA 控制器通常由 DCR、ABR 和 WC 以及中断传送标准接口、DMA 控制逻辑和时序电路组成。其功能是控制 DMA 设备传送的全过程。DMA 接口体现外设

的特定要求,与 DMA 连接的外设不同,其接口亦会存在较大差异。DMA 接口一般应包括 DBR、DAR、DSR 和 DMA 请求等逻辑,其功能是在 DMA 控制器的控制下完成数据传送。

广义型 DMA 接口增加时序线路的目的是,不再由 CPU 产生微操作命令信号,而是由 DMA 控制器自身产生,使得 DMA 传送时占用 CPU 的资源更少了。

多路型 DMA 传送接口如图 7.22 所示。其不仅在物理上可以连接多台外设,而且在逻辑上也允许多台外设同时工作,各外设采用字节或字交叉方式进行数据传送。这种类型接口由一个 DMA 控制器和多个 DMA 接口组成,有多少个 DMA 接口,就可连接多少台外设。

例 7.22 以读磁盘为例,结合周期挪用 DMA 接口线路,说明 DMA 传送的工作过程。

解 周期挪用 DMA 的工作过程分为传送初始化、数据传送和结束处理三个阶段。

(1) 传送初始化

①CPU 判磁盘忙闲良故,若忙或者有故障,则执行其他程序;否则向磁盘 DMA 接口寄存器传送下述参数并执行第②步。

内存始地送 ABR;传输长度的补码送 WC;控制命令、工作方式送 DCR;磁盘的柱面号盘面号和扇区号送设备内部地址寄存器 DAR。

②启动磁盘工作,而后继续执行其他程序。

注意:传送初始化由 CPU 的程序实现,占用 CPU 的时间。

(2) 数据传送

①从磁盘按位读出并拼装为一个单位数据后送 DBR,表示准备已就绪。

②磁盘向 DMA 接口发传输请求 DREQ, DMA 接口向 CPU 申请总线控制权 HRQ。

③CPU 发回 HLDA 信号,表示 DMA 已取得总线控制权,磁盘可以挪用一个存储周期传送一个单位数据。CPU 建立一个 DMA 周期,DMA 周期完成④~⑥三步工作。

④ABR 中的内容送主存的 MAR。将 DBR 中的单位数据送主存的 MBR,同时回送写

存储器命令到主存。

⑤启动主存写操作,将 MBR 中的数据写入 MAR 指定存储单元。

⑥修改传输长度即 WC 加 1,给出下一个要传输单位数据的主存地址即 ABR 加 1。同时判断 WC 是否溢出。

⑦若无溢出,检查也无错误,重复执行①~⑥;否则,向 CPU 中断请求。若 WC 溢出,表明一批数据已交换完毕,向 CPU 发出结束的中断请求;若传输有错,向 CPU 发出传输错误的中断请求。

(3) DMA 结束处理

CPU 响应 DMA 的中断请求,暂停现行程序转去执行中断服务程序。中断服务程序从 DMA 接口的 DSR 中取出状态,进行判断,若为传输错误引起的中断,则转错误诊断及处理程序。若正常结束且需继续传送,则再次对磁盘 DMA 接口初始化;若正常结束不需要再传送,则停磁盘。

例 7.23 试述周期挪用 DMA 传送与程序中断传送的区别。

- 程序中断传送主要靠软件实现数据传送,DMA 主要靠硬件实现数据传送。所以程序中断传送速度慢,而 DMA 传送速度快。
- DMA 请求的响应,只挪用一个存储周期,实现一个单位数据的传送。程序中断传送则中止现行程序的运行,转去执行中断服务程序,实现一个单位数据的传送。因此,程序中断传送的效率低。
- 程序中断传送只适用于慢速外设,DMA 适用于高速成组传送的外设。
- 程序中断传送的响应必须在一条指令执行之末,而 DMA 原则上可在 CPU 不访存的任何时刻得到响应。
- 中断的功能强,可处理各种异常或复杂突发事件,而 DMA 只适合于对数据传送的控制。

例 7.24 I/O 通道有几种类型? 它们的结构组成怎样?

①字节多路通道

字节多路通道:连接控制多台慢速外设,以字节交叉方式传送数据的通道。

①字节多路通道

字节多路通道:连接控制多台慢速外设,以字节交叉方式传送数据的通道。

字节多路通道结构组成如图 7.18 所示,其数据传送示意见图 7.19(a)。

- 包含多个子通道(SCH),每个 SCH 连接一个控制器和一台或多台同种设备。
- 通道控制为所有 SCH 共享,数据缓冲为所有 SCH 公用,用来拆卸、装配数据。
- 字符缓冲器、状态与控制寄存器、主存地址计数器、字节计数器各 SCH 分设。

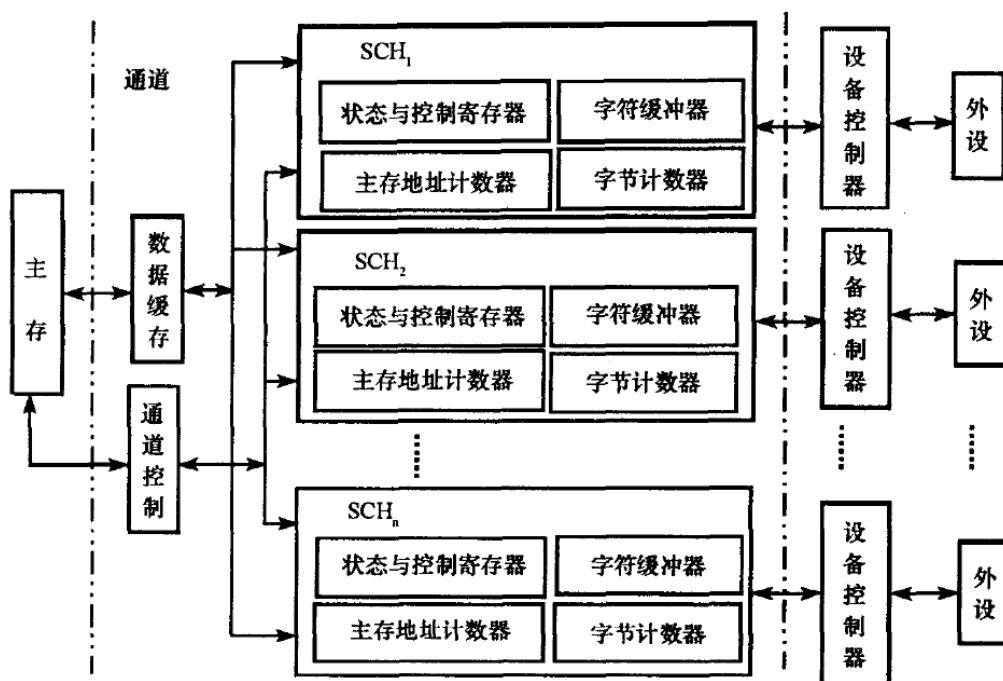
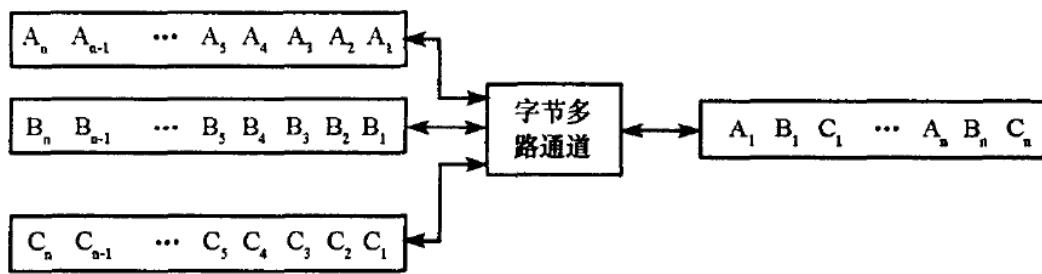


图 7.18 字节多路通道



(a)字节交叉传送

图 7.19 (a)

②选择通道

选择通道:连接控制一台或多台同种高速外设,以成组方式顺序传送数据的通道。

选择通道结构组成如图 7.20 所示。它包括:指出设备读写数据所需的主存地址计数器和记录数据交换长度的字计数器,以及设备内部地址寄存器、数据缓冲寄存器和数据格式变换线路等等。

选择通道每次只能从所连接的设备中选择一台进行数据交换。如图 7.19(b)所示,当传送 C 数据块的设备占用通道时,其他设备均不能交换。只有等 C 传送完整个数据块并释放该通道后,其他设备才可占用通道,并开始交换信息。

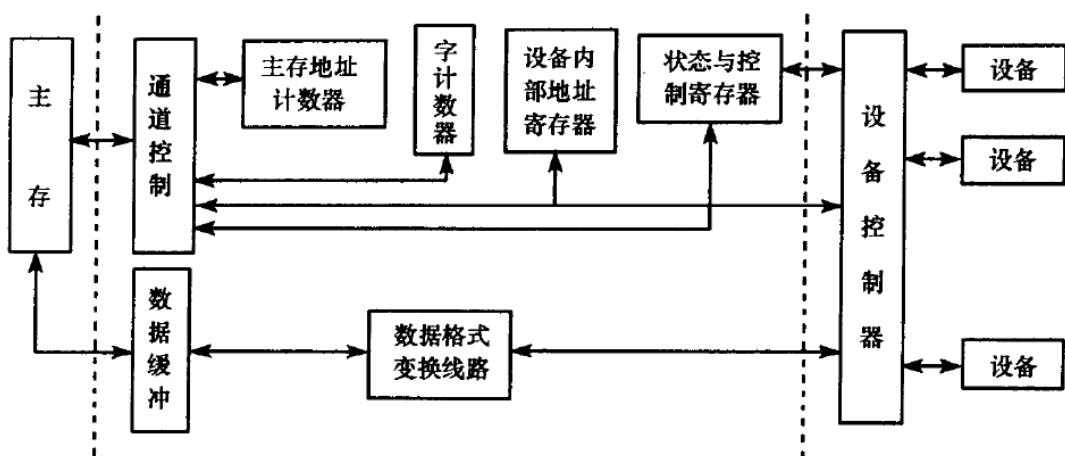
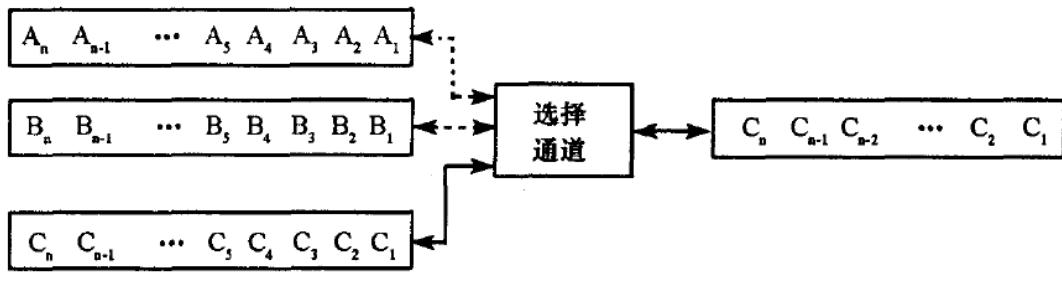


图 7.20 选择通道逻辑结构



③数组多路通道

数组多路通道:连接控制多台高速外设,以成组方式交叉传送数据的通道。

数组多路通道是字节多路通道和选择通道两者结合的产物。目的是充分利用高速成组传送的设备寻找设备内部地址的时间,使得某台设备进行数据传送与其他设备的机电寻址操作重叠。但因控制复杂,随着高速外设寻找设备内部地址的速度加快,有的已达

5ms 以下,利用价值已不很高,故目前很少采用。

例 7.25 试述通道命令字与通道 I/O 指令的异同点。

解 相同点:两者都是计算机的控制信息,控制以通道方式和主机传送数据的外设。

不同点:通道 I/O 指令属于计算机指令系统,是由 CPU 取出、分析和执行的。其操作通常为查询通道、外设状态或启动、停止通道。通道命令字属于通道的指令,由通道取出、分析和执行的。其操作通常为设备的读、写、寻找磁道光道等。

例 7.26 试述通道传送的工作过程。

解 和 DMA 类似,通道传送也分为传送准备、数据传送和结束处理三个阶段。与 DMA 比较,前两个阶段差别较大,结束处理阶段完全相同。

(1) 传送的准备工作

- ① 用户通过访存指令提供传送时所需参数,即提供软控制字;
- ② 用户通过自愿进管指令向 CPU 提出通道 I/O 请求,进入操作系统的设备管理程序;
- ③ 设备管理程序根据软控制字编制通道程序 CHP,并存到主存指定区域;
- ④ CPU 通过了解外设的通道 I/O 指令判用户所需外设是否良闲?若良闲,则执行启动通道和外设 I/O 指令,进入数据传送阶段。

(2) 数据传送

① 通道从主存指定区域取出通道程序 CHP,执行 CHP,进行数据传送。通道程序执行完毕,数据传送结束。

② 数据传送正常结束或传输中出现错误,通道都向 CPU 发中断请求,等待处理。

(3) 结束处理

CPU 响应中断,执行中断服务程序,取出 CSW,若正常结束,执行“正常结束”服务程序进行处理;否则执行“错误处理”服务程序进行处理。

例 7.27 试述通道传送与 DMA 传送的区别。

- DMA 主要靠专用接口硬件实现数据传送;通道则靠执行通道程序 CHP,实现数据传送。
- DMA 初始化要由 CPU 执行多条指令完成,通道则自动取 CHP 执行,无需初始化,原因是 CCW 包含了初始化信息,通道从存储器中取出 CCW,存放到通道相应的寄存器,就相当于对通道传送进行了初始化。
- DMA 一般用来控制高速外设成组传送,通道既可控制高速外设成组传送,也可控制低速外设以字或字节方式交叉传送。

例 7.28 试述通道传送与程序中断传送的区别。

- 程序中断控制传送靠中止 CPU 现行程序,转去执行中断服务程序实现;通道则是通过执行通道程序 CHP 实现。
- 中断服务程序与 CPU 的现行程序是串行工作的;而通道的通道程序 CHP 与 CPU 的现行程序是并行工作的。
- 通道是集中独立的硬件,可连接多台快、慢速外设并行工作;程序中断控制传送只适用于慢速外设,且每个外设都有自己独立的接口和中断服务程序。
- 中断的功能强,可处理各种异常或突发事件;而通道只适合于对数据传送的控制。(程序中断控制传送以 CPU 为中心;通道则和 DMA 一样以主存为中心。)

例 7.29 说明选择通道与选择型 DMA 接口的异同点。

解 相同点:物理上都可连接多台同种高速外设,但在逻辑上只允许连接一台外设。即在某一时间段内,接口只能为一台外设服务。仅当被选中设备一批数据传送完后,才能选择另一台设备,不允许在数据传送过程中切换设备。一批数据传送结束的处理也相同。

不同点:选择通道数据传送前,接口不需 CPU 指令进行初始化。数据传送时,只需执行通道程序 CHP 即可完成数据传送。选择型 DMA 接口则需 CPU 指令进行初始化,还需挪用 CPU 的访存周期,才可完成数据传送。

例 7.30 说明字节多路通道与多路型 DMA 接口的异同点。

解 相同点:在物理上和逻辑上都可连接多台外设,一批数据传送结束的处理也相同。

不同点:字节多路通道数据传送前,接口不需 CPU 指令进行初始化。数据传送时,只需执行通道程序 CHP 即可完成数据传送。多路型 DMA 接口需 CPU 指令进行初始化,还需挪用 CPU 的访存周期,才可完成数据传送。

第七章 计算机模块结构与互联

例 8.1 解释下列术语:互连结构,总线,总线带宽,总线宽度,总线主设备,总线从设备,总线源设备,总线目标设备,总线标准,隐蔽式仲裁。

解 (1)互连结构:连接各种模块的通路的集合称为互连结构。

(2)总线:由传输线、三态门和输入输出缓冲构成的多个功能部件所共享的一组信息传输线。

(3)总线带宽:单位时间内总线上传送的数据量,亦称总线数据传输率或总线频宽。

(4)总线宽度:总线一次能传送的数据位数,即数据线的条数。

(5)总线主设备:能够申请并获得总线使用权的设备。

(6)总线从设备:不能够申请总线使用权的设备。

(7)总线源设备:总线操作过程中,将发送数据的设备称为总线源设备。

(8)总线目标设备:总线操作过程中,将接收数据的设备称为总线源设备。

(9)总线标准:各种设备通过总线连接成一个系统所必须遵循的规范,它为各种设备的互连提供了一种标准界面。

(10)隐蔽式仲裁:获得总线使用权的主设备进行数据传输的同时,总线仲裁电路还可对另一主设备的请求进行仲裁,因仲裁不需要额外的时间,故称这种仲裁方式为隐藏式仲裁。PCI 总线采用的就是隐蔽式仲裁。

例 8.2 从规模上看,计算机的总线可以分为哪几种?按总线信息的传输种类分,有哪几种?

解 按总线规模和涉及的范围,计算机的总线可以分为如下四种:

(1)片内总线:芯片内部设置的总线,用于完成芯片内部各功能单元的信息交换。如

ALU 内部总线属于片内总线。

(2)系统总线:用于计算机内部各大功能部件的信息传输,因这些部件都各自制作在不同插件板上,故亦称板级总线。

(3)设备总线:设备总线又称为 I/O 总线,用于连接各种外部设备。如 PCI 总线。

(4)通信总线:又称外部总线,用于完成计算机间、计算机与其他系统间的信息通信,例如 RS-232 总线。

按总线信息的传输种类,可分为以下三种:

(1)地址总线 AB:指出总线设备编号或存储单元地址的信号线,通常为单向线。

(2)控制总线 CB:用于控制设备或部件对数据线、地址线使用的信号线。总体上是双向线,具体到实际的某条信号线则是单向的。

(3)数据总线 DB:设备或部件进行数据交换的通路,通常为双向线。

例 8.3 为什么说总线宽度是连接到总线上的设备可能获得的最大性能的决定因素之一?为什么说总线宽度又是影响系统性能的关键因素之一?

解 总线宽度是总线一次能传送的数据位数。显然,总线的宽度越大,并行性就越好,一次可传输的信息量就越多。所以说数据总线宽度决定连接到总线上的设备可能获得的最大性能的决定因素之一。

计算机系统性能除与组成计算机的各个部件性能有关外,这些部件互连性能的优劣也至关重要。各个部件性能再好,若互连不好,成为系统的瓶颈,系统性能就会大打折扣。对于以总线作为互连结构的计算机系统,互连性能好的标志是总线带宽即总线数据传输

率高。而与总线带宽密切相关的两个参数是总线宽度和总线时钟频率。在总线时钟频率一定的前提下,总线宽度越宽,总线带宽越大。所以总线宽度也是影响系统性能的关键因素之一。

例 8.4 什么是控制总线? 控制总线在总线通路中有何作用?

解 用于控制设备或部件对数据线、地址线使用的信号线,统称为控制总线。

控制总线主要包括定时和命令信号线两类。定时信号线指出有效的地址和数据出现在总线上的时间;命令信号线标明总线上要完成的操作。常用的命令信号线有总线请求/应答、读/写等。

例 8.5 什么是总线设备? 它们是如何分类的? 各类设备有何特点? 试举例说明。

解 连接到总线上的所有部件,统称为总线设备。

从可否申请总线使用权考虑,总线设备可分为为主设备和从设备。从数据传输方向角度,总线设备可分为源设备和目标设备。从访问总线设备的方法,总线设备可分为存储器设备和 I/O 设备。各类设备的特点和举例说明如下:

(1)总线主设备和从设备

- 总线主设备:能够申请并能获得总线使用权的设备。
- 总线从设备:不具有申请总线使用权的设备。

总线主、从设备的概念是按总线设备的逻辑功能划分的。而连接到总线上的物理设备,有的是总线主设备,如 CPU;有的是总线从设备,如存储器;还可以既是主设备又是从设备,如智能化磁盘控制器。

(2)总线源设备和总线目标设备

- 总线源设备:发送数据的设备。
- 总线目标设备:接收数据的设备。

总线源、目标设备与总线主、从设备之间没有必然的联系。总线主设备可以是总线源设备,也可以是总线目标设备,还可以既不是总线源设备也不是总线目标设备。总线从设备可以是总线源设备,也可以是总线目标设备。例如 DMA 控制器控制向磁盘写入数据。主存为总线源设备,磁盘为总线目标设备,但两者均为总线从设备,而主设备为 DMA 控制器。

(3)存储器设备和 I/O 设备

- 存储器设备:用访问存储器的方法访问的设备称为存储器设备。该类设备需用访存型总线命令来访问。
- I/O 设备:用访问 I/O 的方法访问的设备称为 I/O 设备。该类设备需用 I/O 型总线命令来访问。

例如:DMA 控制器控制从磁盘读出数据到主存。主存是存储器设备而磁盘是 I/O 设备。

例 8.6 单总线结构存在什么缺点？存在的主要问题有哪些？解决此问题的途径是什么？

解 (1)单总线结构存在的缺点

计算机系统的所有部件，如 CPU、内存、显存、显示器、磁盘、键盘等，不管速度差异多大，都以总线设备的形式连接到一条总线上，且都以同等地位分时共享这条总线。

(2)单总线存在的主要问题

①多设备竞争总线使用权，降低了计算机系统的效率。例如，显示器要正常工作，需要不断访问显存；CPU 执行程序，要不停地访问内存。显示器和 CPU 不得不竞争使用总线，这无疑对于 CPU 和显示器效率都会产生影响，对于高速 CPU 影响尤甚。

②多种速度差异很大的设备连接在一条总线上，整个系统性能会降低。速度很快的 CPU 和速度很慢的键盘等以同等地位分时共享总线。如键盘使用总线的速度每分钟最多三百次。慢速设备使用总线，影响快速设备对总线的使用，系统性能会降低。

③扩展设备的能力受限，计算机系统功能也受到限制。因总线驱动能力有限，使得单总线上不能连接很多设备。较多的主设备导致总线控制器的仲裁时间增加，这对计算机的整体性能是不利的。

(3)解决此问题的途径

采用多总线系统，也可分成多级总线。通过多条带宽不同的总线，将设备按速度分类，速度相当的连接到同一条总线上。低速总线作为高速总线的一个设备工作，通过总线桥接器将速度不同的总线连接起来。这样，可解决或缓解多设备竞争总线使用权的矛盾，不同设备使用不同总线的同时，还可通过共享总线进行数据交换；又可解决速度差异很大的设备连接在一条总线上降低系统效率的问题；另外还解决了单总线驱动能力有限导致扩展能力受限的问题。

例 8.7 什么是总线控制器？它的功能是什么？

解 总线系统中，用来管理“总线上设备和设备使用总线的过程”的部件，称为总线控制器。它并不一定是独立的控制器，其组成可以分布到总线的各个部件或者各个设备上。其功能如下：

(1)总线系统资源的管理：连接到总线上的所有部件，如存储空间、I/O 设备、通道、DMA 和中断接口等称为总线资源。对资源进行分配、选择、启动等称为对总线资源的管理。

(2)总线系统的定时：产生总线命令和各种定时信号。

(3)总线的仲裁：确定哪个设备获得总线使用权。

(4)总线的连接：要完成多种总线不同总线协议的转换和多条总线之间的连接。

例 8.8 试用表格分析比较同步总线和异步总线的优缺点。

表 8.2 同步总线和异步总线的优缺点

总线类型	优 点	缺 点
同步总线	1. 总线在统一的总线时钟下操作,设计、使用和调试都比较简单 2. 所有总线信号和命令必须与总线时钟同步,即总线上所有事件都在总线时钟开始处发生	1. 所有设备的总线操作所需总线时钟数都是整数,造成时间的浪费 2. 总线操作周期设计需满足最慢设备使用,导致快速设备不能高效地使用总线
异步总线	1. 慢速设备对快速总线的影响仅限于本次总线操作,不影响下一次快速设备的总线操作,所以效率高 2. 对于设备速度上差异的适应性更强	1. 总线无统一的总线时钟,设计、控制和调试都比较复杂 2. 在整个工作阶段控制信号的时间需要保持,直至一次传输完成

例 8.9 何谓分时多路复用总线? 以常见的读写复用总线为例,说明复用策略下信息传送的过程。

解 如果在总线的一条连接线上定义了多种意义的信号或者连接多个总线设备,在不同的时间段,该线所表示的意义或者所连接的总线设备是不同的,称这样的连接线为分时多路复用总线。

读写复用总线,即一条连接线上定义读和写两种命令信号。对于仅定义两种命令信号的总线,通常用高、低电平分别表示一种命令信号,如用高电平表示读,低电平表示写。当要进行读传输操作时,为高电平,控制总线的读操作;当要进行写传输操作时,为低电平,控制总线的写操作。

例 8.10 以 I/O 设备访问操作为例。说明同步总线操作过程,并画出其时序图。

解 输入设备使用总线进行一次传输操作的时序如图 8.18 所示。图中,传输操作周期被分为 4 个时间段:时间槽 1、时间槽 2、时间槽 3 和时间槽 4。

时间槽 1: 总线开始工作时间。总线发出启动命令 Start, 同时主设备发地址到 Address, 指出输入设备的地址(编号);发输入输出操作命令 IO 和读输入设备的命令 Read。

时间槽 2: 总线从设备操作时间。经过这段时间,将要输入的数据送到 Data。如果速度太慢,要输入的数据送不到 Data,还可附加一个或多个时间槽。

时间槽 3: 总线主设备取数时间。输入设备发出总线应答信号 Acknowledge, 表示输入

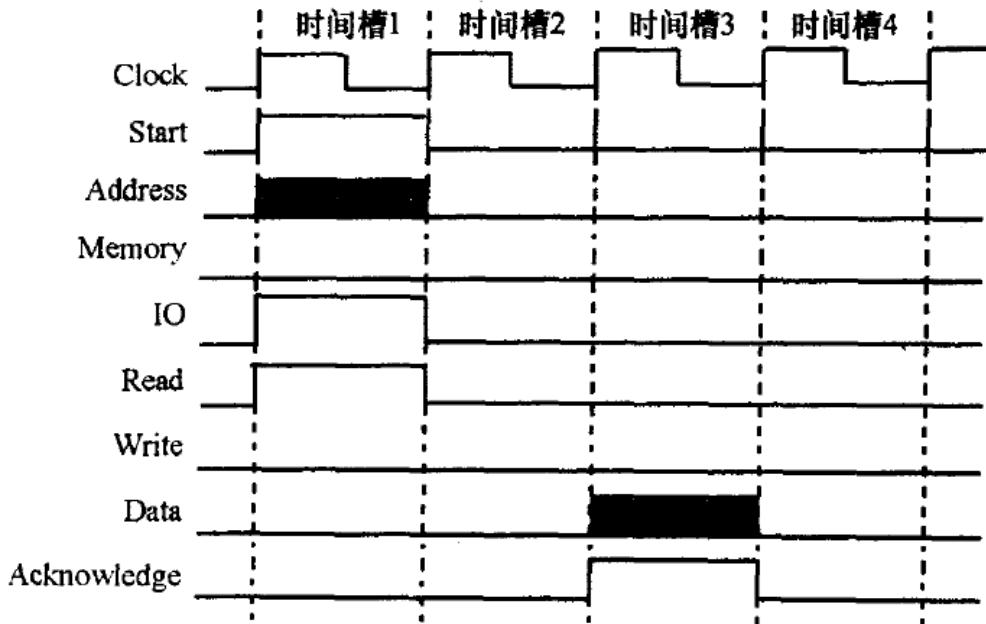


图 8.18 同步总线的输入设备一次传输过程

设备已将数据送到 Data。总线主设备接收到该信号，知道数据总线上已有数据可用，可以取走。该信号同时告知总线控制器，本次总线操作可以结束。

时间槽 4: 总线恢复时间。总线控制器接收到应答信号 Acknowledge，知道本次总线操作结束，收回总线使用权，准备将总线使用权交给下一个总线主设备。如果总线恢复时间足够短，时间槽 4 也可去掉。

例 8.11 何谓集中式仲裁和分布式仲裁？二者各有何优缺点？

解 (1) 集中式仲裁：设置集中式的仲裁电路，它连接所有总线主设备并根据某种策略选中其中的 1 个总线主设备获得总线使用权。这种仲裁的优缺点如下：

①优点：仲裁过程及总线设备接口简单。

②缺点：仲裁电路出现故障，将导致整个系统瘫痪；扩展设备需要对仲裁电路进行大的修改，难度较大。

(2) 分布式仲裁：所有主设备均设置自己的仲裁电路。当主设备发出请求时，各仲裁电路根据一定的策略，共同决定总线使用权。这种仲裁的优缺点如下：

①优点：线路可靠性高，设备扩展灵活，设备接插比较随意。

②缺点：确定总线主设备是否在正常工作，系统需要进行超时判断。由于每个主设备需要在其接口设计仲裁电路，导致设计的复杂性加大。

例 8.12 什么是固定优先权仲裁和动态优先权仲裁？它们各有什么特点？动态优先权算法一般有哪些？

解 (1)固定优先权仲裁:每个总线主设备都具有固定的不可改变的优先级,称这样的仲裁方式为固定优先权仲裁。

这种仲裁的特点是:固定优先权电路结构简单,设计容易;但因一些总线主设备固定具有很高的优先权,而另一些始终优先权很低,所以系统效率不高。

(2)动态优先权仲裁:每个总线主设备的优先权不是固定不变的,而是随时间变化的,称这样的仲裁方式为动态优先权仲裁。

这种仲裁的特点是:比固定优先权仲裁策略更加合理,使用范围更加广泛;实现固定

优先权仲裁的电路仅使用组合逻辑网络,而实现动态优先权的电路通常是一个时序逻辑网络,因此实现要复杂得多。

(3)动态优先权仲裁算法

①简单的轮转优先权算法:假定有 n 个总线主设备并具有 n 级优先权,每个主设备对应一级;,第 n 级优先权最高。每次仲裁后,最低优先权的总线主设备被置为最高优先权,而其他主设备的优先权均减 1。

②相关的轮转优先权算法:假定同上。若获得总线使用权的主设备的优先级为 i ,则优先权比 i 大的总线主设备的优先级保持不变,比 i 小的主设备的优先级加 1,优先权为 i 的主设备被置为最低级。

③随机优先权算法:每次仲裁后,所有总线主设备的优先权将通过一个伪随机数发生器进行重新分配。

④最近最少使用算法:某一时刻最长时间没有获得总线使用权的总线主设备具有最高优先权。

例 8.13 什么是查询式优先权仲裁? 它和其他仲裁方式有何不同? 它具有什么特点?

解 (1)查询式优先权仲裁:又称控制器查询仲裁。当总线控制器发现有使用总线的请求时,控制器将通过查询地址来确定使用总线的主设备。这样的仲裁方式,称为查询式优先权仲裁。

(2)与其他仲裁方式的差别:查询式优先权仲裁采用“请求、查询、应答”方式确定总线使用权。即总线主设备发出使用总线的请求,总线控制器将先产生并发出“查询地址”,然后再由总线仲裁电路发出“应答”决定哪个总线主设备获得总线使用权。而其他仲裁方式仅采用“请求、应答”方式确定总线使用权。即总线主设备发出使用总线的请求,总线仲裁电路根据一定的策略,马上决定哪个总线主设备获得总线的使用权,不必再发出“查询地址”。

(3)查询式优先权仲裁的特点:优先权是通过仲裁电路产生的“查询地址”顺序来确定的,因此优先权算法比较容易控制和修改;因仲裁需要先产生“查询地址”而后查询,最后才应答,即使完全使用硬件实现,速度也会受到限制,因此,仅适用于总线主设备少的情况。

例 8.14 什么是并行仲裁和串行仲裁? 它们各有什么特点?

解 (1)一次只能处理一个总线主设备的总线请求信号的仲裁方式,称为串行仲裁。

串行仲裁的特点:结构简单,可扩展性好,节省器材;速度慢,仅适用于低速外围设备,所以目前很少采用。

(2)一次可以处理多个总线主设备的总线请求信号,并对它们进行仲裁的仲裁方式称为并行仲裁。

并行仲裁的特点:速度快,适用于高速总线设备;结构复杂,使用器材多。多用于内部总线和高速总线的仲裁。

例 8.15 什么是成组数据传送?与基本的数据传送方式相比,它有何特点?成组数据传送时,怎样控制数据传送结束?

解 (1)成组数据传送:获得一次总线控制权,传送一个数据块的数据传送方式,称为成组数据传送。

(2)与基本的数据传送方式相比,其特点是采用成组数据交换,省去了首数据以外其他数据的地址命令处理时间,整个信息交换时间大幅度减少。当每组数据块长度很大时,平均数据传输时间中地址命令处理时间几乎可以忽略不计,这是总线采用成组数据传送方式的最大优越性。

(3)成组数据传送结束的两种控制方法:

①采用固定长度的数据传送,使用计数器来控制数据传送的结束。

②定义一条专用的控制信号线,信号线有效进行成组数据传送,信号线失效,表示一次成组传送使用总线结束。

例 8.16 什么是“读后写”和“写后读”操作?它们各有何特点?

在不同的应用中,对总线的数据交换可能有一些特殊要求,如“读后写”和“写后读”就是两种特殊的数据传送方式。

①“读后写”操作:又称“读出、修改、写”操作,该操作对同一个设备先读出,接着将读出的数据修改,随后再将修改后的数据写入,整个操作是一个整体,中间不能被打断。

“读后写”操作的特点:操作的整个过程不允许被打断,必须一次完成。对于总线操作而言,进行“读、修改、写”的总线设备一旦获得了总线的使用权,就一直占用总线,直到写数据完成为止。

②“写后读”操作:该操作对同一个设备先写入数据,接着再将写入的数据读出,以确定数据是否被正确写入,整个操作是一个整体,中间不能被打断。

“写后读”操作的特点:操作的整个过程不允许被打断,必须一次完成。对于总线操作而言,进行“写后读”的总线设备一旦获得了总线的使用权,就一直占用总线,直到数据读出完成为止。

The End