存器和字计数器的修改、识别总线地址、指定传送类型(输入或输出)以及通知设备已经被授予一个 DMA 周期(DACK)等。

(5) 中断机构

当字计数器溢出(全"0")时,表示一批数据交换完毕,由"溢出信号"通过中断机构向 CPU 提出中断请求,请求 CPU 作 DMA 操作的后处理。必须注意,这里的中断与 5.5 节介绍的 I/O 中断的技术相同,但中断的目的不同,前面是为了数据的输入或输出,而这里是为了报告一批数据传送结束。它们是 I/O 系统中不同的中断事件。

(6) 设备地址寄存器(DAR)

DAR 存放 I/O 设备的设备码或表示设备信息存储区的寻址信息,如磁盘数据所在的区号、盘面号和柱面号。具体内容取决于设备的数据格式和地址的编址方式。

5.6.3 DMA 的工作过程

1. DMA 传送过程

DMA 的数据传送过程分为预处理、数据传送和后处理 3 个阶段。

(1) 预处理

在 DMA 接口开始工作之前,CPU 必须给它预置如下信息。

- 给 DMA 控制逻辑指明数据传送方向是输入(写主存)还是输出(读主存)。
- 向 DMA 设备地址寄存器送入设备号,并启动设备。
- 向 DMA 主存地址寄存器送入交换数据的主存起始地址。
- 对字计数器赋予交换数据的个数。

上述工作由 CPU 执行几条输入输出指令完成,即程序的初始化阶段。这些工作完成后,CPU 继续执行原来的程序,如图 5.48(a)所示。

当 I/O 设备准备好发送的数据(输入)或上次接收的数据已经处理完毕(输出)时,它便通过 DMA 接口向 CPU 提出占用总线的申请,若有多个 DMA 同时申请,则按轻重缓急由硬件排队判优逻辑决定优先等。待 I/O 设备得到主存总线的控制权后,数据的传送便由该 DMA 接口进行管理。

(2) 数据传送

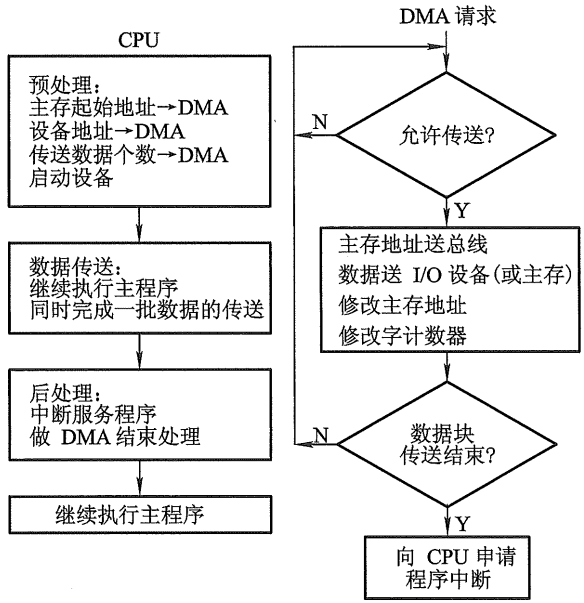
DMA 方式是以数据块为单位传送的,以周期挪用的 DMA 方式为例,其数据传送的流程如图 5.48(b)所示。

结合图 5.47,以数据输入为例,具体操作如下。

① 当设备准备好一个字时,发出选通信号,将该字读到 DMA 的数据缓冲寄存器(BR)中,表示数据缓冲寄存器"满"(如果 I/O 设备是面向字符的,则一次读入一个字节,组装成一个字)。

② 与此同时设备向 DMA 接口发请求(DREQ)。

③ DMA 接口向 CPU 申请总线控制权(HRQ)。



(a) DMA 传送过程　　(b) 数据传送阶段的细化

图 5.48　DMA 传送过程示意图

④ CPU 发回 HLDA 信号,表示允许将总线控制权交给 DMA 接口。

⑤ 将 DMA 主存地址寄存器中的主存地址送地址总线,并命令存储器写。

⑥ 通知设备已被授予一个 DMA 周期(DACK),并为交换下一个字做准备。

⑦ 将 DMA 数据缓冲寄存器的内容送数据总线。

⑧ 主存将数据总线上的信息写至地址总线指定的存储单元中。

⑨ 修改主存地址和字计数值。

⑩ 判断数据块是否传送结束,若未结束,则继续传送;若已结束,(字计数器溢出),则向 CPU 申请程序中断,标志数据块传送结束。

若为输出数据,则应完成以下操作:

① 当 DMA 数据缓冲寄存器已将输出数据送至 I/O 设备后,表示数据缓冲寄存器已“空”。

② 设备向 DMA 接口发请求(DREQ)。

③ DMA 接口向 CPU 申请总线控制权(HRQ)。

④ CPU 发回 HLDA 信号,表示允许将总线控制权交给 DMA 接口使用。

⑤ 将 DMA 主存地址寄存器中的主存地址送地址总线,并命令存储器读。

⑥ 通知设备已被授予一个 DMA 周期(DACK),并为交换下一个字做准备。

⑦ 主存将相应地址单元的内容通过数据总线读入 DMA 的数据缓冲寄存器中。

⑧ 将 DMA 数据缓冲寄存器的内容送到输出设备,若为字符设备,则需将其拆成字符输出。

⑨ 修改主存地址和字计数值。

⑩ 判断数据块是否已传送完毕，若未完毕，继续传送；若已传送完毕，则向 CPU 申请程序中断。

(3) 后处理

当 DMA 的中断请求得到响应后，CPU 停止原程序的执行，转去执行中断服务程序，做一些 DMA 的结束工作，如图 5.48(a)的后处理部分。这包括校验送入主存的数据是否正确；决定是否继续用 DMA 传送其他数据块，若继续传送，则又要对 DMA 接口进行初始化，若不需要传送，则停止外设；测试在传送过程中是否发生错误，若出错，则转错误诊断及处理错误程序。

例 5.3　一个 DMA 接口可采用周期窃取方式把字符传送到存储器，它支持的最大批量为 400 个字节。若存取周期为 100 ns，每处理一次中断需 5 μs，现有的字符设备的传输率为 9 600 bps。假设字符之间的传输是无间隙的，若忽略预处理所需的时间，试问采用 DMA 方式每秒因数据传输需占用处理器多少时间？如果完全采用中断方式，又需占用处理器多少时间？

解：根据字符设备的传输率为 9 600 bps，则每秒能传输

$$9\ 600/8 = 1\ 200\ \text{B}(1\ 200\ \text{个字符})$$

若采用 DMA 方式，传送 1 200 个字符共需 1 200 个存取周期，考虑到每传 400 个字符需中断处理一次，因此 DMA 方式每秒因数据传输占用处理器的时间是

$$0.1\ \mu\text{s} \times 1\ 200 + 5\ \mu\text{s} \times (1\ 200 / 400) = 135\ \mu\text{s}$$

若采用中断方式，每传送一个字符要申请一次中断请求，每秒因数据传输占用处理器的时间是

$$5\ \mu\text{s} \times 1\ 200 = 6\ 000\ \mu\text{s}$$

例 5.4　假设磁盘采用 DMA 方式与主机交换信息，其传输速率为 2 MBps，而且 DMA 的预处理需 1 000 个时钟周期，DMA 完成传送后处理中断需 500 个时钟周期。如果平均传输的数据长度为 4 KB，试问在硬盘工作时，50 MHz 的处理器需用多少时间比率进行 DMA 辅助操作(预处理和后处理)？

解：DMA 传送过程包括预处理、数据传送和后处理 3 个阶段。传送 4 KB 的数据长度需

$$(4\ \text{KB})/(2\ \text{MBps}) = 0.002\ \text{s}$$

如果磁盘不断进行传输，每秒所需 DMA 辅助操作的时钟周期数为

$$(1\ 000 + 500)/0.002 = 750\ 000$$

故 DMA 辅助操作占用 CPU 的时间比率为

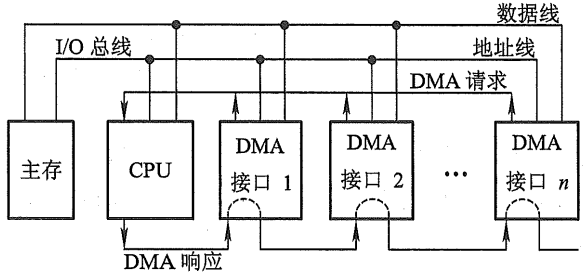
$$[750\ 000 / (50 \times 10^6)] \times 100\% = 1.5\%$$

2. DMA 接口与系统的连接方式

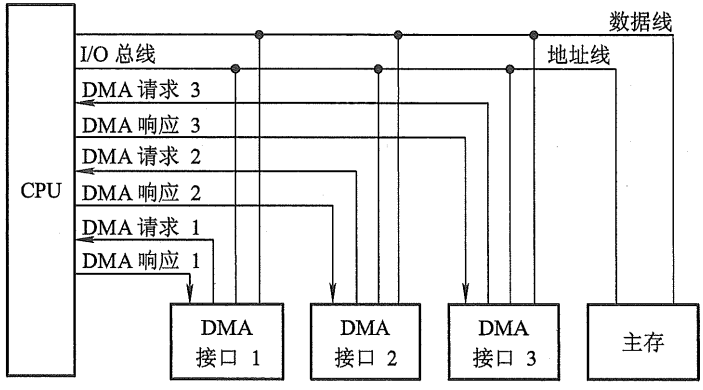
DMA 接口与系统的连接方式有两种，如图 5.49 所示。

图 5.49(a)为具有公共请求线的 DMA 请求方式，若干个 DMA 接口通过一条公用的 DMA 请求线向 CPU 申请总线控制权。CPU 发出响应信号，用链式查询方式通过 DMA 接口，首先选中的设备获得总线控制权，即可占用总线与主存传送信息。

图 5.49(b)是独立的 DMA 请求方式,每一个 DMA 接口各有一对独立的 DMA 请求线和 DMA 响应线,它由 CPU 的优先级判别机构裁决首先响应哪个请求,并在响应线上发出响应信号,被获得响应信号的 DMA 接口便可控制总线与主存传送数据。



(a) 具有公共请求线的 DMA 请求



(b) 独立的 DMA 请求

图 5.49　DMA 接口与系统的连接方式

3. DMA 小结

与程序中断方式相比,DMA 方式有如下特点。

① 从数据传送看,程序中断方式靠程序传送,DMA 方式靠硬件传送。

② 从 CPU 响应时间看,程序中断方式是在一条指令执行结束时响应,而 DMA 方式可在指令周期内的任一存取周期结束时响应。

③ 程序中断方式有处理异常事件的能力,DMA 方式没有这种能力,主要用于大批数据的传送,如硬盘存取、图像处理、高速数据采集系统等,可提高数据吞吐量。

④ 程序中断方式需要中断现行程序,故需保护现场;DMA 方式不中断现行程序,无须保护现场。

⑤ DMA 的优先级比程序中断的优先级高。

5.6.4　DMA 接口的类型

现代集成电路制造技术已将 DMA 接口制成芯片，通常有选择型和多路型两类。

1. 选择型 DMA 接口

这种类型的 DMA 接口的基本组成如图 5.47 所示，它的主要特点是在物理上可连接多个设备，在逻辑上只允许连接一个设备，即在某一段时间内，DMA 接口只能为一个设备服务，关键是在预处理时将所选设备的设备号送入设备地址寄存器。图 5.50 是选择型 DMA 接口的逻辑框图。选择型 DMA 接口特别适用于数据传输率很高的设备。

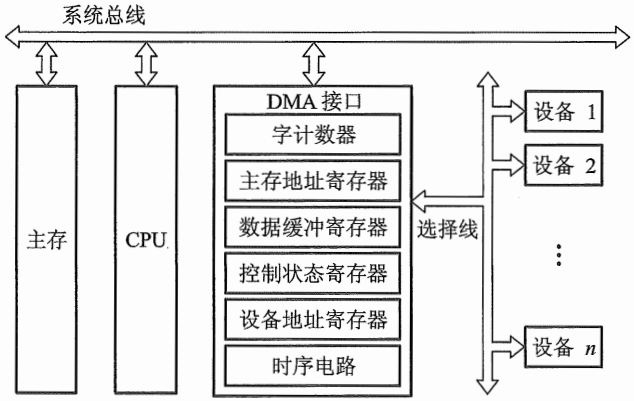


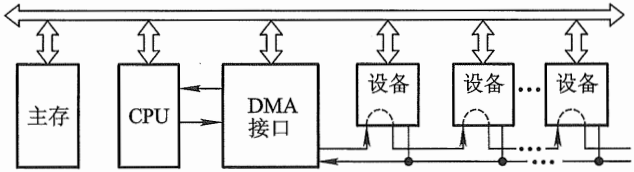
图 5.50　选择型 DMA 接口的逻辑框图

2. 多路型 DMA 接口

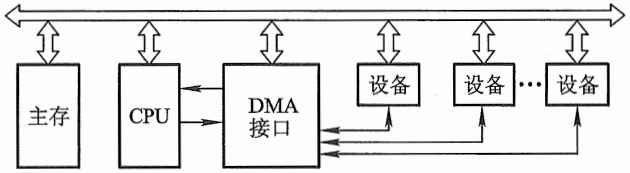
多路型 DMA 接口不仅在物理上可以连接多个设备，而且在逻辑上也允许多个设备同时工作，各个设备采用字节交叉的方式通过 DMA 接口进行数据传送。在多路型 DMA 接口中，为每个与它连接的设备都设置了一套寄存器，分别存放各自的传送参数。图 5.51(a)和(b)分别是链式多路型 DMA 接口和独立请求多路型 DMA 接口的逻辑框图。这类接口特别适合于同时为多个数据传输率不十分高的设备服务。

图 5.52 是多路型 DMA 接口工作原理示意图。图中磁盘、磁带、打印机同时工作。磁盘、磁带、打印机分别每隔 30 μs、45 μs、150 μs 向 DMA 接口发 DMA 请求，磁盘的优先级高于磁带，磁带的优先级高于打印机。

假设 DMA 接口完成一次 DMA 数据传送需 5 μs，由图 5.52 可见，打印机首先发请求，故 DMA 接口首先为打印机服务(T_1)；接着磁盘、磁带同时又有 DMA 请求，DMA 接口按优先级别先响应磁盘请求(T_2)，再响应磁带请求(T_3)，每次 DMA 传送都是一个字节。这样，在 90 多微秒的时间里，DMA 接口为打印机服务一次(T_1)，为磁盘服务 4 次(T_2、T_4、T_6、T_7)，为磁带服务 3 次(T_3、T_5、T_8)。可见 DMA 接口还有很多空闲时间，可再容纳更多的设备。



(a) 链式多路型DMA接口



(b) 独立请求多路型DMA接口

图 5.51　多路型 DMA 接口的逻辑框图

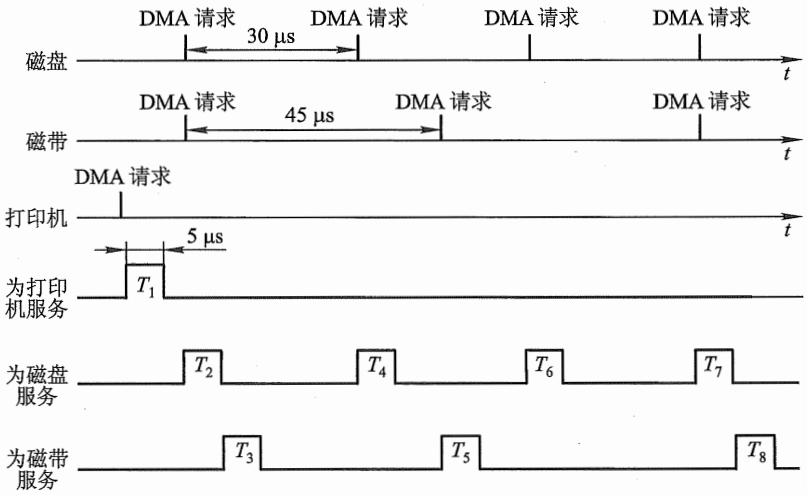


图 5.52　多路型 DMA 接口工作原理示意图

思考题与习题

5.1　I/O 设备有哪些编址方式,各有何特点?

5.2　简要说明 CPU 与 I/O 设备之间传递信息可采用哪几种联络方式,它们分别用于什么场合。

5.3　I/O 设备与主机交换信息时,共有哪几种控制方式? 简述它们的特点。

5.4　试比较程序查询方式、程序中断方式和 DMA 方式对 CPU 工作效率的影响。

5.5　图形显示和图像显示有何区别?

5.6　字符显示器的接口电路中配有缓冲存储器和只读存储器，各有何作用？

5.7　试比较针式打印机、激光打印机和喷墨打印机的特点。

5.8　某计算机的 I/O 设备采用异步串行传送方式传送字符信息。字符信息的格式为 1 位起始位、7 位数据位、1 位检验位和 1 位停止位。若要求每秒钟传送 480 个字符，那么该设备的数据传送速率为多少？

5.9　什么是多媒体技术？简要说明研制多媒体计算机的关键技术。

5.10　什么是 I/O 接口，它与端口有何区别？为什么要设置 I/O 接口？I/O 接口如何分类？

5.11　简述 I/O 接口的功能和基本组成。

5.12　结合程序查询方式的接口电路，说明其工作过程。

5.13　说明中断向量地址和入口地址的区别和联系。

5.14　在什么条件下，I/O 设备可以向 CPU 提出中断请求？

5.15　什么是中断允许触发器？它有何作用？

5.16　在什么条件和什么时间，CPU 可以响应 I/O 的中断请求？

5.17　某系统对输入数据进行取样处理，每抽取一个输入数据，CPU 就要中断处理一次，将取样的数据存至存储器的缓冲区中，该中断处理需 P 秒。此外，缓冲区内每存储 N 个数据，主程序就要将其取出进行处理，这个处理需 Q 秒。试问该系统可以跟踪到每秒多少次中断请求？

5.18　试以键盘设备为例，结合中断接口电路，说明其工作过程。

5.19　在程序中断方式中，磁盘申请中断的优先级高于打印机。当打印机正在进行打印时，磁盘申请中断请求。试问是否要将打印机输出停下来，等磁盘操作结束后，打印机输出才能继续进行？为什么？

5.20　试比较单重中断和多重中断服务程序的处理流程，说明它们不同的原因。

5.21　中断向量通过什么总线送至什么地方？为什么？

5.22　程序查询方式和程序中断方式都是通过“程序”传送数据，两者的区别是什么？

5.23　调用中断服务程序和调用子程序有何区别？

5.24　试分析图 5.33 所示对多个设备的查询流程，说明这种处理方式存在的问题以及如何改进。

5.25　根据以下要求设计一个产生 3 个设备向量地址的电路。

（1）3 个设备的优先级按 A→B→C 降序排列。

（2）A、B、C 的向量地址分别为 110100、010100、000110。

（3）排队器采用链式排队电路。

（4）当 CPU 发来中断响应信号 INTA 时，可将向量地址取至 CPU。

5.26　什么是多重中断？实现多重中断的必要条件是什么？

5.27　DMA 方式有何特点？什么样的 I/O 设备与主机交换信息时采用 DMA 方式，举例说明。

5.28　CPU 对 DMA 请求和中断请求的响应时间是否相同？为什么？

5.29　结合 DMA 接口电路说明其工作过程。

5.30　在 DMA 的工作方式中，CPU 暂停方式和周期挪用方式的数据传送流程有何不同，画图说明。

5.31　假设某设备向 CPU 传送信息的最高频率是 40 000 次/秒，而相应的中断处理程序执行时间为 40 μs，试问该外设是否可用程序中断方式与主机交换信息，为什么？

5.32　设磁盘存储器转速为 3 000 r/min，分 8 个扇区，每扇区存储 1 KB，主存与磁盘存储器数据传送的宽度为 16 位（即每次传送 16 位）。假设一条指令最长执行时间是 25 μs，是否可采用一条指令执行结束时响应 DMA 请求的方案，为什么？若不行，应采取什么方案？

5.33　试从下面 7 个方面比较程序查询、程序中断和 DMA 三种方式的综合性能。

(1) 数据传送依赖软件还是硬件。

(2) 传送数据的基本单位。

(3) 并行性。

(4) 主动性。

(5) 传输速度。

(6) 经济性。

(7) 应用对象。

5.34　解释周期挪用,分析周期挪用可能会出现的几种情况。

5.35　试从 5 个方面比较程序中断方式和 DMA 方式的区别。

附录 5A　ASCII 码

表 5.2 列出的 ASCII 码(American Standard Code for Information Interchange,美国信息交换标准码)是美国信息交换标准委员会制定的 7 位二进制码,共有 128 种字符,其中包括 32 个通用控制字符、10 个十进制数码、52 个英文大写与小写字母、34 个专用符号(如 $、%、+、= 等)。除了 32 个控制字符不打印外,其余 96 个字符全部可以打印。

ASCII 码由 $b_7b_6b_5b_4b_3b_2b_1$ 这 7 个二进制位组成,书写上可用两位十六进制数表示,如“A”可用 41H 表示,“7”可用 37H 表示。为了提高信息传输的可靠性,通常增加一位 b_8 做校验位,这样一个字符就可用 8 位二进制代码表示。

表 5.2　ASCII 码 $b_7b_6b_5b_4b_3b_2b_1$

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
0 0 0 0	NUL	DLE	SP	0	@	P	`	P
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	$	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	'	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[	k	¦

续表

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
1 1 0 0	FF	FS	,	<	L	/	l	|
1 1 0 1	CR	GS	-	=	M	]	m	}
1 1 1 0	SO	RS	.	>	N	↑	n	~
1 1 1 1	SI	VS	/	?	O	-	o	DEL

注：

NUL	空行	VT	纵向制表	SYN	同步空转
SOH	标题开始	FF	改换格式	ETB	信息组传送结束
STX	文件开始	CR	回车	CAN	作废
ETX	文件结束	SO	移出	EM	记录媒体结束
EOT	传送结束	SI	移入	SUB	代替
ENQ	询问	DEL	删除	ESC	脱离
ACK	回答	DC1	设备控制 1	FS	字段分隔
BEL	报警	DC2	设备控制 2	GS	字组分隔
LF	换行	NAK	否定回答		

用 ASCII 码可方便地表示十进制数串。十进制数串在计算机内主要有两种表示形式：非压缩型和压缩型。

(1) 非压缩型

非压缩型的十进制数每一个字符占一个字节，又根据符号位的不同位置，将其分为前分隔式和后嵌入式两种。

前分隔式的符号位占一个字节，并且放在数字位之前，用 2B（即字符“+”的 ASCII 码）表示正号，用 2D（即字符“-”的 ASCII 码）表示负号。每个十进制位均用对应的 ASCII 码表示，例如：

+427　表示为　　2B　34　32　37

-427　表示为　　2D　34　32　37

后嵌入式的符号位不占一个字节，而是将符号嵌入最低一位数字中，其规则是：如果是负数，就将最低位十进制数的 ASCII 码加上 40H；如果是正数则不变。例如：

+427　表示为　　34　32　37

-427　表示为　　34　32　77

可见最低一个字节既表示数值，又表示符号。

用非压缩型表示的十进制数进行算术运算很不方便，因为每个字节占 8 位，只有其低 4 位的值才表示数值，高 4 位值在算术运算时无数值意义，这种表示主要用于非数值计算的有关领域中。

(2) 压缩型

如果采用一个字节存放两个十进制的数位,就成了压缩型的十进制数。这种方式比非压缩型节省了存储空间,又便于完成十进制数的算术运算。压缩型十进制数的每个数位可用数字符 ASCII 码的低 4 位表示,或用 BCD 码表示。

附录 5B　BCD 码

BCD(Binary Coded Decimal)码又称二-十编码,它用 4 位二进制代码表示一位十进制数。最常见的 BCD 码是 8421 码,又称 NBCD(Natural Binary Coded Decimal)码。由于 8421 码每位的权与二进制数完全相同,而 4 位二进制代码共有 16 种组合,因此 1010~1111 这 6 种代码是无效的。NBCD 码与十进制数的对应关系如表 5.3 所示。

表 5.3　8421 码与十进制数对照表

十进制数	8421 码	8421 奇校验码	8421 偶校验码
0	0 0 0 0	1 0 0 0 0	0 0 0 0 0
1	0 0 0 1	0 0 0 0 1	1 0 0 0 1
2	0 0 1 0	0 0 0 1 0	1 0 0 1 0
3	0 0 1 1	1 0 0 1 1	0 0 0 1 1
4	0 1 0 0	0 0 1 0 0	1 0 1 0 0
5	0 1 0 1	1 0 1 0 1	0 0 1 0 1
6	0 1 1 0	1 0 1 1 0	0 0 1 1 0
7	0 1 1 1	0 0 1 1 1	1 0 1 1 1
8	1 0 0 0	0 1 0 0 0	1 1 0 0 0
9	1 0 0 1	1 1 0 0 1	0 1 0 0 1

采用 BCD 码所表示的十进制数,再用十六进制数 C 表示“+”号,用十六进制数 D 表示“-”号,而且均放在数字串的最后,就可表示有符号的十进制数。例如:

+427　表示为　　0100　0010　0111　1100

-427　表示为　　0100　0010　0111　1101

当十进制数串为偶数时,在第一个字节的高 4 位补“0”,即

+42　表示为　　0000　0100　0010　1100

-42　表示为　　0000　0100　0010　1101

附录 5C　奇偶校检码

为了校验编码的正确性，在被传送的 n 位代码上增加一位检验位，并使其配置后的 $n+1$ 位代码中“1”的个数为奇数，则称其为奇校验；若配置后“1”的个数为偶数，则称其为偶校验。例如，在十进制数的 8421 码的前面加上一位校验位，组成 5 位代码，若 5 位二进制代码配置结果“1”的个数为奇数，就称为奇校验码；若配置结果“1”的个数为偶数，就称为偶校验码，如表 5.3 所示。对表中奇校验码而言，倘若传送过程中 5 位代码中“1”的个数不为奇数，则表明传送出错，可见奇校验码具有检错能力。同理，偶校验码也具有检错能力。

奇偶校验码通常用于 I/O 设备，例如，键盘输入时使用 ASCII 码，再配一位校验位，组成 8 位的奇偶校验码，正好占一个字节。在传送过程中如果出现一位错，便能检测出来，但由于不知出错位的位置，故无法纠错。此外，一旦传送过程中出现两位错，奇偶性不变，也无法判断是否出错。

第3篇　中央处理器

以上各章节基本上把CPU看作一个“黑匣子”，并且分析了它通过总线与存储器和I/O部件之间的相互关系。本篇将剖析其内部结构，讲述CPU的功能，包括计算机的运算、指令系统、指令流水、时序系统、中断系统及控制单元。除时序系统和控制单元将在第4篇单独讲述外，其余部分均在此篇介绍。

第 6 章　计算机的运算方法

计算机的应用领域极其广泛，但不论其应用在什么地方，信息在机器内部的形式都是一致的，即均为 0 和 1 组成的各种编码。本章主要介绍参与运算的各类数据（包括无符号数和有符号数、定点数和浮点数等），以及它们在计算机中的算术运算方法。使读者进一步认识到计算机在自动解题过程中数据信息的加工处理流程，从而进一步加深对计算机硬件组成及整机工作原理的理解。有关逻辑运算以及计算机中采用的各种进位制均在前修课中介绍过，本章只在附录中给出了各种进位制及其相互转换的关系（可参阅附录 6A）。至于计算机中的字符编码以及校验码，读者可分别参阅本书附录 5A、附录 5B、附录 5C、4.2.6 节和 4.4.6 节等。

6.1　无符号数和有符号数

在计算机中参与运算的数有两大类：无符号数和有符号数。

6.1.1　无符号数

计算机中的数均放在寄存器中，通常称寄存器的位数为机器字长。所谓无符号数，即没有符号的数，在寄存器中的每一位均可用来存放数值。当存放有符号数时，则需留出位置存放符号。因此，在机器字长相同时，无符号数与有符号数所对应的数值范围是不同的。以机器字长为 16 位为例，无符号数的表示范围为 0～65 535，而有符号数的表示范围为 −32 768～+32 767（此数值对应补码表示，详见 6.1.2 节）。

6.1.2　有符号数

1. 机器数与真值

对有符号数而言，符号的“正”“负”机器是无法识别的，但由于“正”“负”恰好是两种截然不同的状态，如果用“0”表示“正”，用“1”表示“负”，这样符号也被数字化了，并且规定将它放在有效数字的前面，即组成了有符号数。

例如，有符号数（小数）：