(3) 关中断

CPU 进入中断周期,意味着 CPU 响应了某个中断源的请求,为了确保 CPU 响应后所需做的一系列操作不至于又受到新的中断请求的干扰,在中断周期内必须自动关中断,以禁止 CPU 再次响应新的中断请求。图 8.30 是 CPU 自动关中断的示意图。图中允许中断触发器 EINT 和中断标记触发器 INT 可选用标准的 R-S 触发器。当进入中断周期时,INT 为"1"状态,触发器原端输出有一个正跳变,经反相后产生一个负跳变,使 EINT 置"0",即关中断。

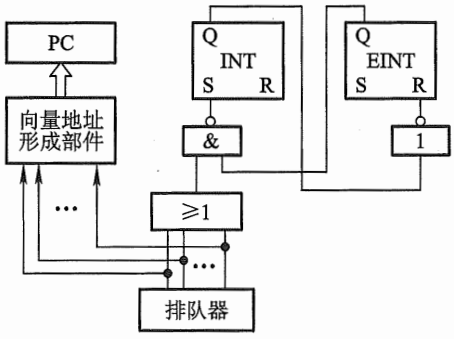

图 8.30　硬件关中断示意图

上述保护断点、寻找入口地址和关中断这些操作都是在中断周期内由一条中断隐指令完成的。所谓中断隐指令,即在机器指令系统中没有的指令,它是 CPU 在中断周期内由硬件自动完成的一条指令。

8.4.5　保护现场和恢复现场

保护现场应该包括保护程序断点和保护 CPU 内部各寄存器内容的现场两个方面。程序断点的现场由中断隐指令完成,各寄存器内的现场可在中断服务程序中由用户(或系统)用机器指令编程实现,参见 5.5.5 节及图 5.43。

恢复现场是指在中断返回前,必须将寄存器的内容恢复到中断处理前的状态,这部分工作也由中断服务程序完成,如图 5.43 所示。

8.4.6　中断屏蔽技术

中断屏蔽技术主要用于多重中断。

1. 多重中断的概念

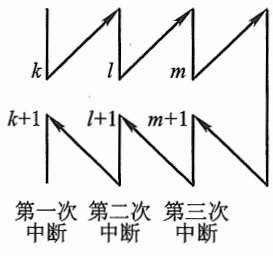

图 8.31　多重中断示意图

当 CPU 正在执行某个中断服务程序时,另一个中断源又提出了新的中断请求,而 CPU 又响应了这个新的请求,暂时停止正在运行的服务程序,转去执行新的中断服务程序,这称为多重中断,又称中断嵌套,如图 8.31 所示。如果 CPU 对新的请求不予响应,待执行完当前的服务程序后再响应,即为单重中断。中断系统若要具有处理多重中断的功能,必须具备各项条件。

2. 实现多重中断的条件

① 提前设置"开中断"指令。

由上述分析可知,CPU 进入中断周期后,由中断隐指令自动将 EINT 置“0”,即关中断,这就意味着 CPU 在执行中断服务程序中禁止响应新的中断请求。CPU 若想再次响应中断请求,必须开中断,这一任务通常由中断服务程序中的开中断指令实现。由于开中断指令设置的位置不同,决定了 CPU 能否实现多重中断。由图 5.43 可见,多重中断“开中断”指令的位置前于单重中断,从而保证了多重中断允许出现中断嵌套。

② 优先级别高的中断源有权中断优先级别低的中断源。

在满足①的前提下,只有优先级别更高的中断源请求才可以中断比其级别低的中断服务程序,反之则不然。例如,有 A、B、C、D 4 个中断源,其优先级按 A→B→C→D 由高向低次序排列。在 CPU 执行主程序期间,同时出现了 B 和 C 的中断请求,由于 B 级别高于 C,故首先执行 B 的服务程序。当 B 级中断服务程序执行完返回主程序后,由于 C 请求未撤销,故 CPU 又再去执行 C 级的中断服务程序。若此时又出现了 D 请求,因为 D 级别低于 C,故 CPU 不响应,当 C 级中断服务程序执行完返回主程序后再去执行 D 级的服务程序。若此时又出现了 A 请求,因 A 级别高于 D,故 CPU 暂停对 D 级中断服务程序的执行,转去执行 A 级中断服务程序,等 A 级中断服务程序执行完后,再去执行 D 级中断服务程序。上述的中断处理示意图如图 8.32 所示。

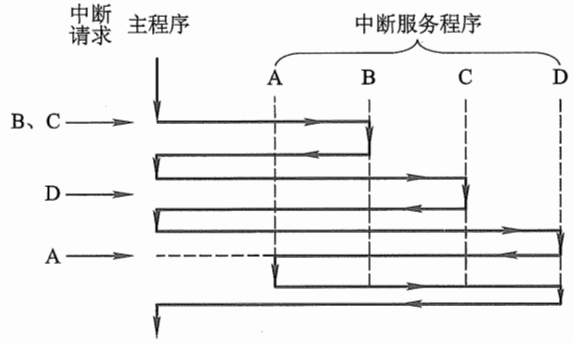

图 8.32　多重中断处理示意图

为了保证级别低的中断源不干扰比其级别高的中断源的中断处理过程,保证上述②的实施,可采用屏蔽技术。

3. 屏蔽技术

(1) 屏蔽触发器与屏蔽字

图 5.37 示出了程序中断接口电路中完成触发器 D、中断请求触发器 INTR 和屏蔽触发器 MASK 三者之间的关系。当该中断源被屏蔽时(MASK = 1),此时即使 D = 1,中断查询信号到来时刻只能将 INTR 置“0”,CPU 接收不到该中断源的中断请求,即它被屏蔽。若该中断源未被屏蔽(MASK = 0),当设备工作已完成时(D = 1),中断查询信号则将 INTR 置“1”,表示该中断源向 CPU 发出中断请求,该信号送至排队器进行优先级判断。

如果排队器集中设在 CPU 内，加上屏蔽条件，就可组成具有屏蔽功能的排队器，如图 8.33 所示。

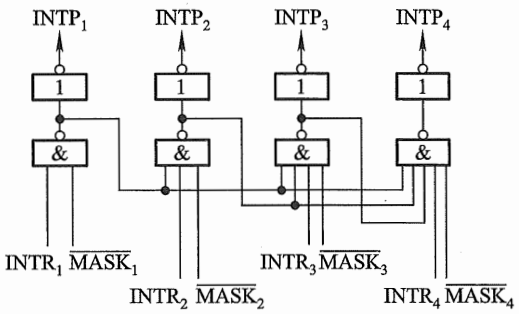

图 8.33　具有屏蔽功能的排队器

显然，对应每个中断请求触发器就有一个屏蔽触发器，将所有屏蔽触发器组合在一起，便构成一个屏蔽寄存器，屏蔽寄存器的内容称为屏蔽字。屏蔽字与中断源的优先级别是一一对应的，如表 8.7 所示。

表 8.7　中断优先级与屏蔽字的关系

优先级	屏蔽字
1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3	0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4	0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
5	0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
6	0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
⋮	⋮
15	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
16	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

表 8.7 是对应 16 个中断源的屏蔽字，每个屏蔽字由左向右排序为第 1，2，3…，共 16 位。不难发现，每个中断源对应的屏蔽字是不同的。1 级中断源的屏蔽字是 16 个 1；2 级中断源的屏蔽字是从第 2 位开始共 15 个 1；3 级中断源的屏蔽字是从第 3 位开始共 14 个 1……第 16 级中断源的屏蔽字只有第 16 位为 1，其余各位为 0。

在中断服务程序中设置适当的屏蔽字，能起到对优先级别不同的中断源的屏蔽作用。例如，1 级中断源的请求已被 CPU 响应，若在其中断服务程序中（通常在开中断指令前）设置一个全“1”的屏蔽字，便可保证在执行 1 级中断服务程序过程中，CPU 不再响应任何一个中断源（包括本级在内）的中断请求，即此刻不能实现多重中断。如果在 4 级中断源的服务程序中设置一个

屏蔽字 0001111111111111，由于第 1~3 位为 0，意味着第 1~3 级的中断源未被屏蔽，因此在开中断指令后，比第 4 级中断源级别更高的 1、2、3 级中断源可以中断 4 级中断源的中断服务程序，实现多重中断。

（2）屏蔽技术可改变优先等级

严格地说，优先级包含响应优先级和处理优先级。响应优先级是指 CPU 响应各中断源请求的优先次序，这种次序往往是硬件线路已设置好的，不便于改动。处理优先级是指 CPU 实际对各中断源请求的处理优先次序。如果不采用屏蔽技术，响应的优先次序就是处理的优先次序。

采用了屏蔽技术后，可以改变 CPU 处理各中断源的优先等级，从而改变 CPU 执行程序的轨迹。例如，A、B、C、D 这 4 个中断源的优先级别按 A→B→C→D 降序排列，根据这一次序，CPU 执行程序的轨迹如图 8.34 所示。当 4 个中断源同时提出请求时，处理次序与响应次序一致。

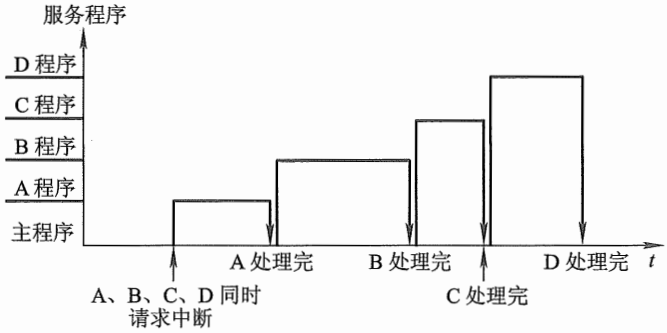

图 8.34　CPU 执行程序的轨迹

在不改变 CPU 响应中断的次序下，通过改变屏蔽字可以改变 CPU 处理中断的次序。例如，将上述 4 个中断源的处理次序改为 A→D→C→B，则每个中断源所对应的屏蔽字发生了变化，如表 8.8 所示。表中原屏蔽字对应 A→B→C→D 的响应顺序，新屏蔽字对应 A→D→C→B 的处理顺序。

表 8.8　中断处理次序与屏蔽字的关系

中断源	原屏蔽字	新屏蔽字
A	1　1　1　1	1　1　1　1
B	0　1　1　1	0　1　0　0
C	0　0　1　1	0　1　1　0
D	0　0　0　1	0　1　1　1

在同样中断请求的情况下，CPU 执行程序的轨迹发生了变化，如图8.35 所示。CPU 在运行程序的过程中，若 A、B、C、D 4 个中断源同时提出请求，按照中断级别的高低，CPU 首先响应并处理 A 中断源的请求，由于 A 的屏蔽字是 1111，屏蔽了所有的中断源，故 A 程序可以全部执行完，然后回到主程序。由于 B、C、D 的中断请求还未响应，而 B 的响应优先级高于其他，所以 CPU 响应 B 的请求，进入 B 的中断服务程序。在 B 的服务程序中，由于设置了新的屏蔽字 0100，即 A、C、D 可打断 B，而 A 程序已执行完，C 的响应优先级又高于 D，于是 CPU 响应 C，进入 C 的服务程序。在 C 的服务程序中，由于设置了新的屏蔽字 0110，即 A、D 可打断 C，由于 A 程序已执行完，于是 CPU 响应 D，执行 D 的服务程序。在 D 的服务程序中，屏蔽字变成 0111，即只有 A 可打断 D，但 A 已处理结束，所以 D 可以一直执行完，然后回到 C 程序。C 程序执行完后，回到 B 程序。B 程序执行完后，回到主程序。至此，A、B、C、D 均处理完毕。

采用了屏蔽技术后，在中断服务程序中需设置新的屏蔽字，流程如图 8.36 所示。与第 5 章图 5.43(b)所示的中断服务程序相比，增加了置屏蔽字和恢复屏蔽字两部分内容。而且为了防止在恢复现场过程中又出现新的中断，在恢复现场前又增加了关中断，恢复屏蔽字之后，必须再次开中断。

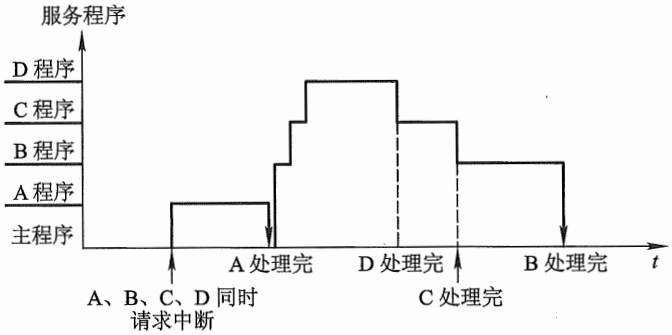

图 8.35　改变中断处理次序后 CPU 执行程序的轨迹

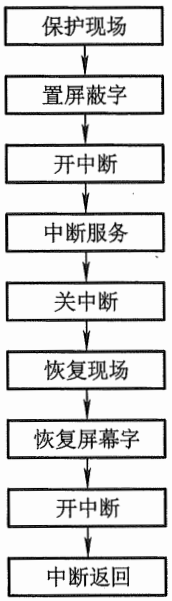

图 8.36　采用屏蔽技术的中断服务程序

例 8.2　设某机有 4 个中断源 1、2、3、4，其硬件排队优先次序按 1→2→3→4 降序排列，各中断源的服务程序中所对应的屏蔽字如表 8.9 所示。

表 8.9　例 8.2 各中断源对应的屏蔽字

中断源	屏蔽字			
	1	2	3	4
1	1	1	0	1
2	0	1	0	0
3	1	1	1	1
4	0	1	0	1

（1）给出上述 4 个中断源的中断处理次序。

（2）若 4 个中断源同时有中断请求，画出 CPU 执行程序的轨迹。

解：（1）根据表 8.9，4 个中断源的处理次序是按 3→1→4→2 降序排列。

（2）当 4 个中断源同时有中断请求时，由于硬件排队的优先次序是 1→2→3→4，故 CPU 先响应 1 的请求，执行 1 的服务程序。由于在该服务程序中设置了屏蔽字 1101，故开中断指令后转去执行 3 的服务程序，且 3 的服务程序执行结束后又回到 1 的服务程序。1 的服务程序结束后，CPU 还有 2、4 两个中断源请求未响应。由于 2 的响应优先级高于 4，故 CPU 先响应 2 的请求，执行 2 的服务程序。在 2 的服务程序中由于设置了屏蔽字 0100，意味着 1、3、4 可中断 2 的服务程序。而 1、3 的请求已处理结束，因此在开中断指令之后转去执行 4 的服务程序，4 的服务程序执行结束后又回到 2 的服务程序的断点处，继续执行 2 的服务程序，直至该程序执行结束。图 8.37 示意了 CPU 执行程序的轨迹。

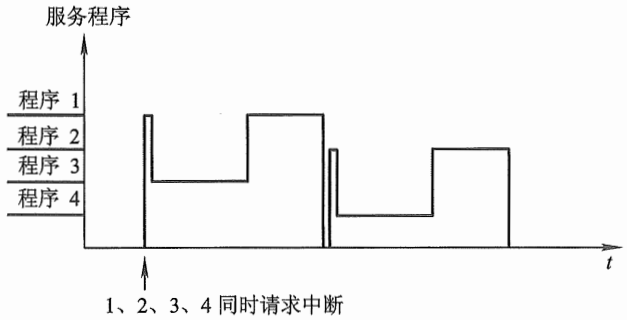

图 8.37　例 8.2 CPU 执行程序的轨迹

（3）屏蔽技术的其他作用

屏蔽技术还能给程序控制带来更大的灵活性。例如，在浮点运算中，当程序员估计到执行某段程序时可能出现“阶上溢”，但又不希望因“阶上溢”而使机器停机，为此可设一屏蔽字，使对应“阶上溢”的屏蔽位为“1”，这样，即使出现“阶上溢”，机器也不停机。

4. 多重中断的断点保护

多重中断时,每次中断出现的断点都必须保存起来,如图 8.31 中共出现了 3 次中断,有 3 个断点 $k+1$、$l+1$、$m+1$ 需保存。中断系统对断点的保存都是在中断周期内由中断隐指令实现的,对用户是透明的。

断点可以保存在堆栈中,由于堆栈先进后出的特点,因此图 8.31 中的 $k+1$ 先进栈,接着是 $l+1$进栈,最后是 $m+1$ 进栈。出栈时,按相反顺序便可准确返回到程序间断处。

断点也可保存在特定的存储单元内,例如约定一律将程序断点存至主存的 0 号地址单元内。由于保存断点是由中断隐指令自动完成的,因此 3 次中断的断点都将存入 0 地址单元,这势必造成前两次存入的断点 $k+1$ 和 $l+1$ 被冲掉。为此,在中断服务程序中的开中断指令之前,必须先将 0 地址单元的内容转存至其他地址单元中,才能真正保存每一个断点。读者可自行练习,画出将程序断点保存到 0 号地址单元的多重中断服务程序流程。

思考题与习题

8.1 CPU 有哪些功能?画出其结构框图并简要说明每个部件的作用。

8.2 什么是指令周期?指令周期是否有一个固定值?为什么?

8.3 画出指令周期的流程图,分别说明图中每个子周期的作用。

8.4 设 CPU 内有这些部件:PC、IR、SP、AC、MAR、MDR 和 CU。

(1) 画出完成间接寻址的取数指令"LDA @X"(将主存某地址单元的内容取至 AC 中)的数据流(从取指令开始)。

(2) 画出中断周期的数据流。

8.5 中断周期前是什么阶段?中断周期后又是什么阶段?在中断周期 CPU 应完成什么操作?

8.6 存储器中有若干数据类型:指令代码、运算数据、堆栈数据、字符代码和 BCD 码,计算机如何识别这些代码?

8.7 什么叫系统的并行性?粗粒度并行和细粒度并行有何区别?

8.8 什么是指令流水?画出指令二级流水和四级流水的示意图,它们中哪一个更能提高处理器速度,为什么?

8.9 当遇到什么情况时流水线将受阻?举例说明。

8.10 举例说明流水线中的几种数据相关。

8.11 今有四级流水线,分别完成取指(IF)、译码并取数(ID)、执行(EX)、写结果(WR)4 个步骤。假设完成各步操作的时间依次为 90 ns、90 ns、60 ns、45 ns。

(1) 流水线的时钟周期应取何值?

(2) 若相邻的指令发生数据相关,那么第 2 条指令安排推迟多少时间才能不发生错误?

(3) 若相邻两指令发生数据相关,为了不推迟第 2 条指令的执行,可采取什么措施?

8.12 在 5 个功能段的指令流水线中,假设每段的执行时间分别是 10 ns、8 ns、10 ns、10 ns 和 7 ns。对于完成 12 条指令的流水线而言,其加速比为多少?该流水线的实际吞吐率为多少?

8.13 为什么说超长指令字比超标量更能提高并行处理能力?