南京大学
NANJING UNIVERSITY

5-8章习题课

殷亚凤
智能软件与工程学院
苏州校区南雍楼东区225
yafeng@nju.edu.cn，https://yafengnju.github.io/

习题课

- **第5章 中央处理器**
- 第6章 指令流水线
- 第7章 存储器层次结构
- 第8章 系统互连及输入输出组织

(感谢计算机系袁春风老师提供的许多习题答案！)

3. 右图给出了某 CPU 内部结构的一部分，MAR 和 MDR 直接连到存储器总线(图中省略)。在 CPU 内部总线 A 和 B 之间的所有数据传送都需经过算术逻辑部件 ALU。ALU 的部分控制信号及其功能如下：

MOVa：F=A；　　MOVb：F=B；

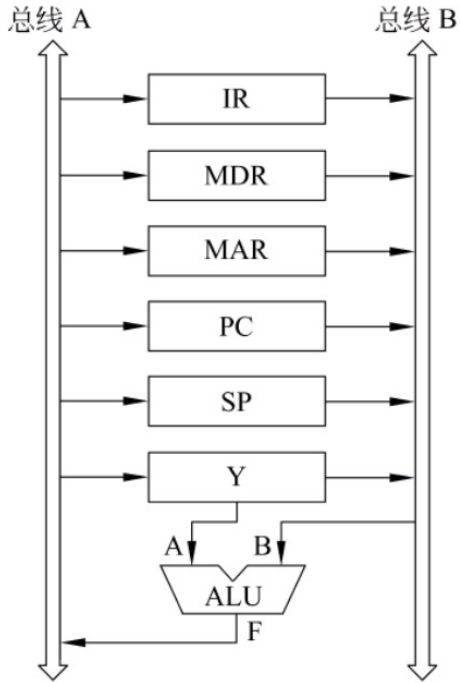
a+1：F=A+1；　　b+1：F=B+1；

a−1：F=A−1；　　b−1：F=B−1。

其中 A 和 B 是 ALU 的输入，F 是 ALU 的输出。假定该 CPU 的指令系统中调用指令 CALL 占两个字，第一个字是操作码，第二个字给出子程序的起始地址，返回地址保存在主存的栈中，用 SP (栈指示器)指向栈顶，存储器按字编址，每次按同步方式从主存读取一个字。要求：

(1) 说明 CALL 指令的功能。

(2) 写出读取并执行 CALL 指令所要求的控制信号序列(提示：当前指令地址已在 PC 中)。



参考答案：

（1）过程调用

（2）方法1：

采用同步方式

1. 取指令操作码：PCout, MOVb, MARin
   Read, b+1, PCin
   MDRout, MOVb, IRin
2. 取子程序首址：PCout, MOVb, MARin
   Read, b+1, Yin
   MDRout, MOVb, PCin
3. 保存返址至栈：SPout, MOVb, MARin
   Yout, MOVb, MDRin
   Write, SPout, b-1, SPin

（2）方法2：

1. 取指令操作码：PCout, MOVb, MARin
   Read, b+1, Yin
   MDRout, MOVb, IRin
2. 取子程序首址：Yout, MOVb, MARin
   Read, a+1, Yin　　（用b+1也行）
   MDRout, MOVb, PCin
3. 保存返址至栈：SPout, MOVb, MARin
   Yout, MOVb, MDRin
   Write, SPout, b-1, SPin

第5章 中央处理器

5. 假定图 5.16 所示单周期数据通路对应的控制逻辑发生错误，使得控制信号 RegWr、RegDst、ALUSrc、Branch、MemWr、ExtOp、R-type、MemtoReg 中某一个在任何情况下总是为 0，则该控制信号为 0 时哪些指令不能正确执行？要求分别讨论。

参考答案：

- 若RegWr≡0，则所有需写结果到寄存器的指令（如：R-Type指令、load指令等）都不能正确执行，因为寄存器不发生写操作；
- 若RegDst≡0，则所有R-Type指令都不能正确执行，因为目的寄存器指定错误；
- 若ALUSrc≡0，则带有立即数的指令不能正确执行，因为立即数不能被选择；
- 若Branch≡0，则Branch指令可能出错，因为永远不会发生转移；
- 若MemWr≡0，则Store指令不能正确执行，因为存储器不能写入所需数据；
- 若ExtOp≡0，则需要符号扩展的指令（如Beq、lw/sw等）发生错误。
- 若R-type≡0，则所有R-type指令的执行可能出错
- 若MemtoReg≡0，则Load指令不能正确执行，因为存储器的数据不能写入寄存器。

6. 假定图 5.16 所示单周期数据通路对应的控制逻辑发生错误，使得控制信号 RegWr、RegDst、ALUSrc、Branch、MemWr、ExtOp、R-type、MemtoReg 中某一个在任何情况下总是为 1，则该控制信号为 1 时哪些指令不能正确执行？要求分别讨论。

参考答案：

- 若RegWr≡1，则所有不需写结果到寄存器的指令（如：sw、beq等）都不能正确执行；
- 若RegDst≡1，则lw和ori等指令不能正确执行，因为目的寄存器指定错误；
- 若ALUSrc≡1，则R-type指令不能正确执行，因为第二源操作数（存在寄存器中）无法被选择；
- 若Branch≡1，则非Branch指令可能出错，因为可能会发生不必要的转移；
- 若MemWr≡1，则除Store指令外其他指令都不能正确执行，因为存储器总会写入数据；
- 若ExtOp≡1，则需要零扩展的指令（如ori等）会发生错误。
- 若R-type≡1，则所有非R-type指令的执行可能出错
- 若MemtoReg≡1，则R-type指令、I-type运算指令不能正确执行，因为ALU运算结果不能写入寄存器。

7. 要在 MIPS 指令集中增加一条 swap 指令，可以有两种做法。一种做法是采用伪指令方式（即软件方式），这种情况下，当执行到 swap 指令时，用若干条已有指令构成的指令序列来代替实现；另一种做法是直接改动硬件来实现 swap 指令，这种情况下，当执行到 swap 指令时，则可在 CPU 上直接执行。要求：

（1）写出用伪指令方式实现“swap rs，rt”时的指令序列（提示：伪指令对应的指令序列中不能使用其他额外寄存器，以免破坏这些寄存器的值）。

（2）假定用硬件实现 swap 指令时会使每条指令的执行时间增加 10%，则 swap 指令要在程序中占多大的比例才值得用硬件方式来实现？

参考答案：

（1） swap指令可用以下三条指令实现。

```
xor $rs, $rs, $rt
xor $rt, $rs, $rt
xor $rs, $rs, $rt
```

（2）假定该指令占x%，其他指令占(1-x)%

则用硬件实现该指令时，程序执行时间为原来的1.1*(x+1-x) =1.1 倍

用软件实现该指令时，程序执行时间为原来的3x+1-x =（2x+1）倍

当1.1 < 2x+1 时，硬件实现才有意义

由此可知，x > 5%

8. 假定图 5.25 多周期数据通路对应的控制逻辑发生错误，使得控制信号 PCWr、MemtoReg、IRWr、RegWr、BrWr、MemWr、PCWrCond、R-type 中某一个在任何情况下总是为 0，则该控制信号为 0 时哪些指令不能正确执行？要求分别讨论。

参考答案：

- 若PCWr≡0，则所有指令都不正确，因为无法更新PC
- 若MemtoReg≡0，则所有Load指令执行错误，因为寄存器写入的是ALU输出
- 若IRWr≡0，则所有指令都不能正确执行，因为IR中不能写入指令
- 若RegWr≡0，则所有需要写结果到寄存器的指令（如：R-Type指令、load指令等）都不能正确执行，因为寄存器不发生写操作
- 若BrWr≡0，则Branch指令不能正确执行，因为无法更新转移目标地址；
- 若MemWr≡0，则Store指令不能正确执行，因为存储器不能写入数据
- 若PCWrCond≡ 0，则Branch指令不能正确执行，因为不能写入转移目标地址到PC
- 若R-type≡0，则所有R-type指令的执行可能出错

9. 假定图 5.25 多周期数据通路对应的控制逻辑发生错误，使得控制信号 PCWr、MemtoReg、IRWr、RegWr、BrWr、MemWr、PCWrCond、R-type 中某一个在任何情况下总是为 1，则该控制信号为 1 时哪些指令不能正确执行？要求分别讨论。

参考答案：

- 若PCWr≡1，则程序执行顺序失控，因为每个时钟都会更新PC
- 若MemtoReg≡1，则R型指令、I型计算指令不能正确执行，因为ALU运算结果不能被写入到寄存器中；
- 若IRWr≡1，则所有指令都可能不能正确执行，因为写入IR的可能不是当前指令
- 若RegWr≡1，则所有不需写结果到寄存器的指令（如：sw、beq等）都不能正确执行
- 若BrWr≡1，则非Branch指令不能正确执行，因为PC中的地址出错；
- 若MemWr≡1，则除Store指令外的所有指令都不能正确执行
- 若PCWrCond≡1，则除Branch外的其他指令可能不能正确执行
- 若R-type≡1，则所有非R-type指令的执行可能出错

10. 假定有一条 MIPS 伪指令“bcmp $t1，$t2，$t3”，其功能是实现对两个主存块数据的比较，$t1 和 $t2 中分别存放两个主存块的首地址，$t3 中存放数据块的长度，每个数据占 4 字节，若所有数据都相等，则将 0 置入 $t1；否则，将第一次出现不相等时的地址分别置入 $t1 和 $t2 并结束比较。若 $t4 和 $t5 是两个空闲寄存器，请给出实现该伪指令的指令序列，并说明在类似于图 5.25 所示的多周期数据通路中执行该伪指令时要用多少个时钟周期。

参考答案：

(1) 实现伪指令“bcmp $t1，$t2，$t3”的指令序列如下：

```
         beq    $t3, $zero, done          #若数据块长度为 0,则结束
compare: lw     $t4, 0($t1)               #块 1 的当前数据取到$t4
         lw     $t5, 0($t2)               #块 2 的当前数据取到$t5
         bne    $t4, $t5, done            #$t4 和$t5 的内容不等,则结束
         addi   $t1, $t1, 4               #块 1 中的当前数据指向下一个
         addi   $t2, $t2, 4               #块 2 中的当前数据指向下一个
         addi   $t3, $t3, -1              #比较次数减 1
         bne    $t3, $zero, compare       #若没有全部比较完,则继续比较
         addi   $t1, $zero, 0             #若全部都相等,则将$t1 置 0
done:
```

10. 假定有一条 MIPS 伪指令“bcmp $t1，$t2，$t3”，其功能是实现对两个主存块数据的比较，$t1 和 $t2 中分别存放两个主存块的首地址，$t3 中存放数据块的长度，每个数据占 4 字节，若所有数据都相等，则将 0 置入 $t1；否则，将第一次出现不相等时的地址分别置入 $t1 和 $t2 并结束比较。若 $t4 和 $t5 是两个空闲寄存器，请给出实现该伪指令的指令序列，并说明在类似于图 5.25 所示的多周期数据通路中执行该伪指令时要用多少个时钟周期。

参考答案：

(2) 在类似图5.25所示的多周期数据通路中执行时，上述程序段中用到的beq、lw、bne、addi的时钟周期数分别为3、5、3、4。假设比较的数据块大小为N个字，则上述指令序列中中的循环最多被执行N次，所需的指令数最多为1+N*7+1=7N+2。其中，lw指令为N*2=2N条，时钟周期数为5*2N=10N；分支（beq、bne）指令数为1+2*N=2N+1条，时钟周期数为3*(2N+1)=6N+3；addi指令为1+3*N=3N+1条，时钟周期数为4*(3N+1)=12N+4。所以，总时钟周期数最多为10N+(6N+3)+(12N+4)=28N+7。

13. 对于多周期 CPU 中的异常和中断处理，回答以下问题：

（1）对于除数为 0、溢出、无效指令操作码、无效指令地址、无效数据地址、缺页、访问越权和外部中断，CPU 在哪些指令的哪个时钟周期能分别检测到这些异常或中断？

（2）在检测到某个异常或中断后，CPU 通常要完成哪些工作？简要说明 CPU 如何完成这些工作。

参考答案：

【分析解答】

（1）多周期 CPU 中不同指令执行时可能会发生不同的异常事件，这些异常事件发生在不同的时钟周期内，CPU 在不同的时钟周期中检测不同的异常事件。

① “除数为 0”异常在取数/译码周期进行检测。

② “溢出”异常在 R-型指令和 I-型运算类指令的执行周期进行检测。

③ “无效指令操作码”异常在取数/译码周期进行检测。

④ “无效指令地址”“缺页”和“访问越权”异常在取指令周期检测。

⑤ “无效数据地址”“缺页”和“访问越权”异常在存储器访问周期检测。

⑥ “外部中断”可在每条指令的最后一个周期进行检测。

（2）CPU 检测到某个异常或中断后，要完成的工作是关中断、保护断点和程序状态、识别异常事件（或中断源）并转异常（或中断）处理。CPU 计算断点值并将断点和程序状态字寄存器信息送到栈或特定的寄存器中；异常的识别大多采用软件识别方式，而外部中断则可以采用软件识别或硬件识别方式，然后转到相应的处理程序。

习题课

- 第5章 中央处理器
- **第6章 指令流水线**
- 第7章 存储器层次结构
- 第8章 系统互连及输入输出组织

(感谢计算机系袁春风老师提供的许多习题答案！)

4. 假定某计算机工程师想设计一个新的CPU，一个典型程序的核心模块有一百万条指令，每条指令执行时间为100ps。请问：

（1）在非流水线处理器上执行该程序需要花多长时间？

（2）若新CPU采用20级流水线，执行上述同样的程序，理想情况下，它比非流水线处理器快多少？

（3）实际流水线并不是理想的，流水段之间的数据传送会有额外开销。这些开销是否会影响指令执行时间和指令吞吐率？

参考答案：

（1）非流水线处理器上执行该程序的时间大约为 $100ps \times 10^6 = 100\mu s$。

（2）若在一个20级流水线的处理器上执行，忽略流水段之间的寄存器延时，理想情况下，每个时钟周期为 100/20=5ps，所以，程序执行时间大约为 $5 \times 10^6 = 5\mu s$，因此，大约快100/5=20倍。

（3）流水段之间数据的传递产生的额外开销，使得一条指令的执行时间被延长，即影响了指令执行时间；同时也延长了每个流水段的时间，即影响了指令吞吐率。

5. 假定最复杂的一条指令所用的组合逻辑分成 6 部分，依次为 A～F，其延迟分别为 80ps、30ps、60ps、50ps、70ps、10ps。在这些组合逻辑块之间插入必要的流水段寄存器就可实现相应的指令流水线，寄存器延迟为 20ps。理想情况下，以下各种方式所得到的时钟周期、指令吞吐率和指令执行时间各是多少？应该在哪里插入流水段寄存器？

（1）插入 1 个流水段寄存器，得到一个两级流水线。

（2）插入 2 个流水段寄存器，得到一个三级流水线。

（3）插入 3 个流水段寄存器，得到一个四级流水线。

（4）吞吐量最大的流水线。

参考答案：

（1）两级流水线的平衡点在 C 和 D 之间，其前面一个流水段的组合逻辑延时为 80+30+60=170ps，后面一个流水段的组合逻辑延时为 50+70+10 =130ps。最长功能段延时为 170ps，加上流水段寄存器延时 20ps，因而时钟周期为 190ps，理想情况下，指令吞吐率为每秒钟执行 1/190ps=5.26G 条指令。每条指令在流水线中的执行时间为 2×190=380ps。

（2）两个流水段寄存器分别插在 B 和 C、D 和 E 之间，这样第一个流水段的组合逻辑延时为 80+30=110ps，中间第二段的延时为 60+50=110ps，最后一个段延时为 70+10 =80ps。这样，每个流水段所用时间都按最长延时调整为 110+20=130ps，故时钟周期为 130ps，指令吞吐率为每秒钟执行 1/130ps=7.69G 条指令，每条指令在流水线中的执行时间为 3×130=390ps。

参考答案：

（3）三个流水段寄存器分别插在 A 和 B、C 和 D、D 和 E 之间，这样第一个流水段的组合逻辑延时为 80ps，第二段延时为 30+60=90ps，第三段延时为 50ps，最后一段延时为 70+10 =80ps。这样，每个流水段都以最长延时调整为 90+20=110ps，故时钟周期为 110ps，指令吞吐率为每秒钟执行 1/110ps=9.09G 条指令，每条指令在流水线中的执行时间为 4×110=440ps。

（4）因为各功能部件对应的组合逻辑中最长延时为 80ps，所以，流水线的时钟周期肯定比 80ps+20ps=100ps 长。为了达到最大吞吐率，时钟周期应该尽量短，因此，最合理的划分方案应该按照每个时钟周期为 100ps 来进行。根据每个功能部件所用时间可知，流水线至少按 5 段来划分，分别把流水线寄存器插入在 A 和 B、B 和 C、C 和 D、D 和 E 之间，这样各段的组合逻辑延时为 80ps、30ps、60ps，50ps 和 80ps。其中，最后一个延时 80ps 是 E 和 F 两个阶段的时间相加而得到的。这样时钟周期为 100ps，指令吞吐率为每秒钟执行 1/100ps=10G 条指令，每个指令的执行时间为 5×100=500ps。

通过对上述 4 种情况进行分析，可以得出以下结论：划分的流水段多，时钟周期就变短，指令执行吞吐率就变高，而相应的额外开销（即插入的流水段寄存器的延时）也变大，使得一条指令的执行时间变长。

第6章 指令流水线

6. 以下指令序列中，哪些指令对之间发生数据相关？假定采用“取指、译码/取数、执行、访存、写回”5段流水线方式，如果不用“转发”技术，需要在发生数据相关的指令前加入几条 nop 指令才能使这段程序避免数据冒险？如果采用“转发”是否可以完全解决数据冒险？不行的话，需要在发生数据相关的指令前加入几条 nop 指令才能使这段 MIPS 程序不发生数据冒险？

```
addu    $s3, $s1, $s0
addu    $t2, $s3, $s3
lw      $t1, 0($t2)
add     $t3, $t1, $t2
```

参考答案：

（1）发生数据相关的是：第 1 和第 2 条指令之间关于$s3，第 2 和第 3 条指令之间关于$t2，第 2 和第 4 条指令之间关于$t2，以及第 3 和第 4 条指令之间关于$t1。

（2）不进行“转发”处理的话，需要分别在第 2、3、4 条指令前加 2 条 nop 指令才能避免数据冒险，共加了 6 条 nop 指令。nop 指令增加的百分比为 6/4=150%，即平均 1 条指令加 1.5 条 nop 指令。该指令序列执行所需的时钟周期数为(5−1)+(4+6)=14。

（3）通过“转发”可以避免第 1 和第 2、第 2 和第 3、第 2 和第 4 条指令之间的数据相关；但第 3 和第 4 条指令之间是 load-use 数据相关，因此，无法用“转发”消除冒险，而需在第 4 条指令前加入 1 条 nop 指令。

7. 假定以下 MIPS 指令序列在图 6.18 所示的流水线数据通路中执行：

```
addu    $s3, $s1, $s0
subu    $t2, $s3, $s3
lw      $t1, 0($t2)
add     $t3, $t1, $t2
add     $t1, $s4, $s5
```

请问：

(1) 上述指令序列中，哪些指令的哪个寄存器需要转发？转发到何处？

(2) 上述指令序列中，是否存在 load-use 数据冒险？

(3) 第 5 周期结束时，各指令执行状态是什么？哪些寄存器的数据正被读出？哪些寄存器将被写入？

参考答案：

- （1）第1条指令的寄存器$s3的内容需要转发，即Ex/Mem流水段寄存器的内容转发到ALU输入端；第2条指令的寄存器$t2的内容需要转发，即Ex/Mem流水段寄存器的内容转发到ALU输入端；第3条指令的寄存器$t1的内容需要转发，即Mem/Wr流水段寄存器的内容转发到ALU输入端
- （2）第3条、第4条指令之间存在load-use数据冒险

参考答案：

- （3）第5周期结束时：

第1条指令，在“写回（Wr）”阶段，寄存器$s3将被写入；
第2条指令，在“访存（Mem）”阶段，执行空操作，Ex/Mem流水段寄存器中的结果送入ALU输入端；
第3条指令，在“执行（Ex）”阶段，ALU执行add操作，进行地址运算，ALU的输出结果将被写入流水段寄存器Ex/Mem中；

第 4 条指令在“译码/取数（ID）”阶段，寄存器 t1和t2 的内容正被读出；在 load-use 冒险检测单元中，因为流水段寄存器 IF/ID 中源操作数寄存器 Rs 为 t1，流水段寄存器 ID/Ex 中目的操作数寄存器 Rt 也为 t1，同时，因为上条指令是 lw，故流水段寄存器 ID/Ex 中的 MemRead 控制信号为 1，所以在该阶段检测到 load-use 冒险条件满足，此时，需要进行 load-use 冒险处理，在流水线中插入一个“气泡”，将指令的执行阻塞一个时钟周期。包括以下三个步骤：①将流水段寄存器 ID/Ex 中的控制信号全部清 0，以保证第 4 条指令被阻塞一个时钟周期执行；②将流水段寄存器 IF/ID 中的指令维持不变，以保证第 4 条指令重新译码后执行；③将 PC 的值维持不变，以保证根据 PC 的值重新取出第 5 条指令。

第 5 条指令在“IF”阶段，指令正被读出，将要送到流水段寄存器 IF/ID 的输入端。因为之前发生了 load-use 数据冒险，所以该指令将在随后的第 6 个时钟周期内重新被读出。

第6章 指令流水线

9. 在一个带转发的 5 段流水线中执行以下 MIPS 程序段，怎样调整指令序列使其性能达到最好？

```
lw    $2, 100($6)
add   $2, $2, $3
lw    $3, 200($7)
add   $6, $4, $7
sub   $3, $4, $6
lw    $2, 300($8)
beq   $2, $8, Loop
```

参考答案：

因为采用“转发”技术，所以，只要对 load-use 数据冒险进行指令序列调整。从上述指令序列来看，第 1 和第 2 条指令、第 6 和第 7 条指令之间存在 load-use 数据冒险，所以，可将与第 2 和第 3 条指令无关的第 4 条指令插入第 2 条指令之前；将无关的第 5 条指令插入第 7 条指令之前。调整顺序后的指令序列如下（粗体部分为变换了位置的指令）。

```
1       lw    $2, 100($6)
4       add   $6, $4, $7
2       add   $2, $2, $3
3       lw    $3, 200($7)
6       lw    $2, 300($8)
5       sub   $3, $4, $6
7       beq   $2, $8, Loop
```

10. 在一个采用“取指、译码/取数、执行、访存、写回”的5段流水线中，若检测结果是否为“0”和将转移目标地址(Btarg和Jtarg)送PC的操作在执行阶段进行，则分支延迟损失时间片(即分支延迟槽)为多少？在带转发的5段流水线中，对于以下MIPS指令序列，哪些指令执行时会发生流水线阻塞？各需要阻塞几个时钟周期？

```
Loop:  add    $t1, $s3, $s3
       add    $t1, $t1, $t1
       add    $t1, $t1, $s6
       lw     $t0, 0($t1)
       bne    $t0, $s5, Exit
       add    $s3, $s3, $s4
       j      Loop
Exit:
```

参考答案：

（1）若检测结果是否为“0”和将转移目标地址送PC的操作在执行阶段进行，那么分支延迟损失时间片为2；

（2）在带转发的5段流水线中，第5条指令所需的操作数$t0是load-use冒险，不能用“转发”解决，需要在第5条指令前加一条nop指令，或者通过硬件将第5条指令的执行阻塞1个时钟周期。当“bne”指令更新PC的操作在“执行（Ex）”阶段进行，流水线将被阻塞2个时钟周期，需要在bne指令后加2条nop指令消除控制冒险。当“jloop”指令更新PC的操作在“执行（Ex）”阶段进行，流水线将被阻塞2个时钟周期，需要在jump指令后加2条nop指令消除控制冒险。

11. 假设数据通路中各主要功能部件的操作时间是：存储单元为 200ps；ALU 和加法器为 100ps；寄存器堆读口或写口为 50ps。程序中指令的组成比例为：取数为 25%、存数为 10%、ALU 为 52%、分支为 11%、跳转为 2%。假设控制单元和传输线路等延迟都忽略不计，则以下实现方式中哪个更快？快多少？

（1）单周期方式。每条指令在一个固定长度的时钟周期内完成。

（2）多周期方式。时钟周期取存储单元操作时间的一半，每类指令时钟数是：取数为 7、存数为 6、ALU 为 5、分支为 4、跳转为 4。

（3）流水线方式。时钟周期取存储操作时间的一半，采用“取指 1、取指 2、取数/译码、执行、存取 1、存取 2、写回”7 段流水线；没有结构冒险；数据冒险采用“转发”技术处理；load 指令与后续各指令之间存在依赖关系的概率分别 1/2，1/4，1/8，…；分支延迟损失时间片为 2，预测准确率为 75%；不考虑异常、中断和访问缺失引起的流水线冒险。

参考答案：

（1）单周期方式下，时钟周期为 200+50+100+200+50=600ps，故一条指令的执行时间为 600ps。

（2）多周期方式下，CPI=0.25×7+0.10×6+0.52×5+0.11×4+0.02×4＝5.47，存储器操作变为在两个时钟周期内完成后，多周期数据通路的时钟周期为 100ps，故平均一条指令的执行时间为 100×5.47=547ps。

参考答案：

（3）流水线方式下，存储器操作变为在两个时钟周期内完成后，其流水线包含了 7 个阶段。

对于分支指令，若预测正确，则不需额外时钟周期，故只需 1 个时钟周期；若预测错误，则因为分支延迟损失时间片为 2，所以应该将错误预取的 2 条指令冲刷掉，额外多用了 2 个时钟周期，因此，预测错误时共需 3 个时钟周期，故分支指令的 CPI=0.25×3+0.75×1=1.5。

对于 load 指令，因为一个存储操作占用两个时钟周期，所以随后第 1 条指令则需 3 个（其中阻塞 2 个）时钟周期；随后第二条指令需 2 个（其中阻塞 1 个）时钟周期，以后的指令都不需要阻塞，故 CPI=1/2×3+1/4×2+2/8×1=2.25。

对于 ALU 指令，随后的数据相关指令都可通过转发解决，故 CPI=1。

对于 store 指令，不会发生数据冒险，故 CPI=1。

对于 jump 指令，最快也要在译码阶段才能确定转移地址，因此需阻塞 2 个时钟周期，加上本身一个时钟周期，共 3 个时钟周期，故 CPI=3。

因此综合 CPI=0.25×2.25+0.10×1+0.52×1+0.11×1.5+0.02×3=1.41。所以，平均一条指令的执行时间为 1.41×100=141ps。

由上述分析可知，流水线处理器的指令执行速度最快，是单周期的 600/141≈4.26 倍，是多周期的 547/141≈3.844 倍。

第6章 指令流水线

12. 有一段程序的核心模块中有 5 条分支指令，该模块将会被执行成千上万次，在其中一次执行过程中，5 条分支指令的实际执行情况如下（T：taken；N：not taken）。

分支指令 1：T-T-T。

分支指令 2：N-N-N-N。

分支指令 3：T-N-T-N-T-N。

分支指令 4：T-T-T-N-T。

分支指令 5：T-T-N-T-T-N-T。

假定各个分支指令在每次模块执行过程中实际执行情况都一样，并且动态预测时每个分支指令都有自己的预测表项，每次执行该模块时的初始预测位都相同。请分析并给出以下几种预测方案的预测准确率。

（1）静态预测，总是预测转移（taken）。

（2）静态预测，总是预测不转移（not taken）。

（3）一位动态预测，初始预测转移（taken）。

（4）二位动态预测，初始预测弱转移（taken）。

参考答案：

预测准确率=预测正确次数/总预测次数×100%。以下 R 表示正确预测次数，W 表示错误预测次数。

（1）B1: R-3, W-0； B2: R-0, W-4； B3: R-3, W-3； B4: R-4, W-1； B5: R-5, W-2。预测准确率 60%。

（2）B1: R-0, W-3； B2: R-4, W-0； B3: R-3, W-3； B4: R-1, W-4； B5: R-2, W-5。预测准确率 40%。

参考答案：

（3）根据主教材图 6.21 所示的状态转换图，可以得到各分支点的预测情况如下：

B1: R-3, W-0； B2: R-3, W-1； B3: R-1, W-5； B4: R-3, W-2； B5: R-3, W-4。预测准确率 52%。

具体每次的预测结果如图 6.10 所示。

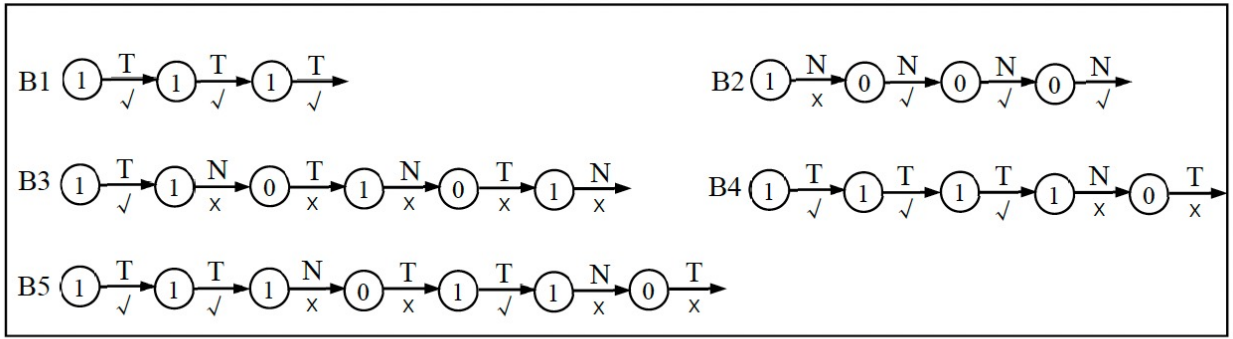


图 6.10 一位动态预测（初始预测为转移）的预测情况

参考答案：

（4）根据主教材图 6.23 所示的状态转换图，可以得到各分支点的预测情况如下：

B1: R-3, W-0； B2: R-3, W-1； B3: R-3, W-3； B4: R-4, W-1； B5: R-5, W-2。预测准确率 72%。

具体每次的预测结果如图 6.11 所示。

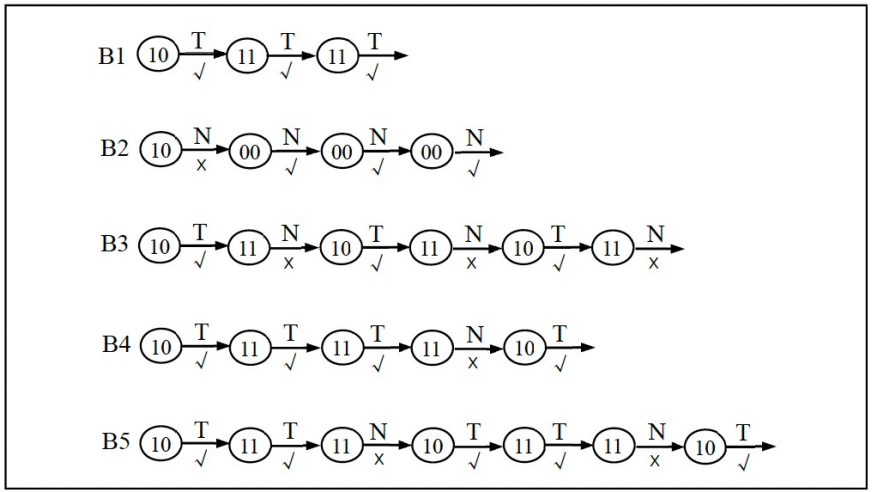


图 6.11 两位动态预测（初始预测为弱转移）的预测情况

习题课

- 第5章 中央处理器
- 第6章 指令流水线
- **第7章 存储器层次结构**
- 第8章 系统互连及输入输出组织

(感谢计算机系袁春风老师提供的许多习题答案！)

5. 假定用 8K×8 位的 EPROM 芯片组成 32K×16 位的只读存储器，要求回答以下问题。

（1）数据寄存器最少应有多少位？

（2）地址寄存器最少应有多少位？

（3）共需多少个 EPROM 芯片？

参考答案：

（1）数据寄存器最少是16位

（2）32K=2^{15}，所以地址寄存器最少是15位，其中高2位用来生成片选信号，后13位表示片内地址

（3）共需要的芯片数为（32K/8K）×（16/8）=8

7. 假定一个存储器系统支持四体交叉存取，某程序执行过程中访问地址序列为 3，9，17，2，51，37，13，4，8，41，67，10，则哪些地址访问会发生体冲突？

参考答案：

对于 4 体交叉访问的存储系统，理想情况下，每隔 1/4 周期可读写一个数据，假定这个时间为Δt。每个存储模块内的地址分布如下。

模块 0：0、4、8、12、16 … …

模块 1：1、5、9、13、17 …37 …41……

模块 2：2、6、10、14、18 … …

模块 3：3、7、11、15、19…51…67……

很显然，如果相邻 4 次访问中给定的访存地址出现在同一个模块内，就会发生访存冲突。所以 17 和 9、37 和 17、13 和 37、8 和 4 会发生冲突。41 和 13 也在同一个模块内且访问间隔小于 4 个Δt，但是，由于访问第 8 单元发生冲突而使其访问延迟三个Δt 进行，从而使得 41 号单元也延迟三个Δt 访问，因而，其访问不会和 13 号单元的访问发生冲突。

8. 假定一个程序重复完成将磁盘上一个 4KB 的数据块读出，进行相应处理后，写回到磁盘的另外一个数据区。各数据块内信息在磁盘上连续存放，并随机地置于磁盘的一个磁道上。磁盘转速为 7200RPM，平均寻道时间为 10ms，磁盘最大数据传输率为 40MB/s，磁盘控制器的开销为 2ms，没有其他程序使用磁盘和处理器，并且磁盘读写操作和磁盘数据的处理时间不重叠。若程序对磁盘数据的处理需要 20000 个时钟周期，处理器时钟频率为 500MHz，则该程序完成一次数据块“读出-处理-写回”操作所需的时间为多少？每秒钟可以完成多少次这样的数据块操作？

参考答案：

磁盘旋转一圈的时间为 1000/(7200/60)≈8.33ms，故平均旋转等待时间约为 8.33/2≈4.17ms。因为数据块内信息连续，所以，一个数据块的传输时间为 $4\times2^{10}/(40\times10^{6})$=0.1024ms，因而数据块的平均读取或写回时间约为 2ms+10ms+4.17ms+0.1024ms≈16.27ms。数据块的处理时间为 20000/500M=0.04ms。因为数据块随机存放在某个磁道上，所以，每个数据块的“读出-处理-写回”操作时间都是相同的，其整个时间约为 16.27ms×2+0.04ms=32.58ms。所以每秒钟可以完成这样的数据块操作次数是 1000ms/32.58ms≈30 次。

11. 假定某计算机主存地址空间大小为 1GB，按字节编址，cache 的数据区（即不包括标记、有效位等存储区）有 64KB，块大小为 128 字节，采用直接映射和直写（write through）方式。请问：

（1）主存地址如何划分？要求说明每个字段的含义、位数和在主存地址中的位置。

（2）cache 的总容量为多少位？

参考答案：

（1）cache 共有 64KB/128B = 512 行，直接映射方式下，cache 行号占 9 位；由于每个主存块大小为 128B，按字节编址，故块内地址为 7 位；主存地址空间大小为 1GB，所以主存地址位数为 30 位。主存地址中标记有 30–9–7=14 位。综上所述，主存地址共有以下三个字段：高 14 位为标记，中间 9 位为行索引，低 7 位为块内地址。

（2）因为直接映射不考虑替换算法，所以cache行中没有用于替换的控制位；因为采用直写方式，所以，cache行中也没有修改位。每个cache行中包含1位有效位、14位标记位和128B的数据，因此，cache总容量为512×(1+14+128×8)b= 519.5Kb。

12. 假定某计算机的 cache 共 16 行，开始为空，块大小为 1 个字，采用直接映射方式，按字编址。CPU 执行某程序时，依次访问以下地址序列：2，3，11，16，21，13，64，48，19，11，3，22，4，27，6 和 11。

（1）说明每次访问是命中还是缺失，试计算访问上述地址序列的命中率。

（2）若 cache 数据区容量不变，而块大小改为 4 个字，则上述地址序列的命中情况又如何？

参考答案：

（1） cache 采用直接映射，每行存放一个字，因此共 16 行；每个主存块对应 1 个字，所以主存块号=字号。得到映射公式为 cache 行号=字号 mod 16。程序开始执行时 cache 为空，所以每个单元第一次访问总是缺失（miss）。CPU 访问给定地址序列的过程如下（每个数字对“x-y”的含义为“x 是访问的主存地址，y 是对应的 cache 行号”。hit 表示命中，miss 表示缺失，miss/replace 表示缺失并替换）：2-2: miss； 3-3: miss； 11- 11: miss； 16- 0: miss； 21-5: miss；13-13: miss；64-0: miss/replace；48-0: miss/replace；19-3: miss/replace；11-11: hit；3-3: miss/replace；22-6: miss；4-4: miss；27-11: miss/replace；6-6: miss/replace；11-11: miss/replace。只有一次命中！命中率为 1/16=6.25%。

参考答案：

（2）数据块大小改为 4 个字，cache 最多能存放 16 个字的数据，因此 cache 共 4 行；每个主存块对应 4 个字，所以，主存块号= [字号/4]。得到映射公式为：cache 行号=主存块号 mod 4。CPU 访问给定地址序列的过程如下（每个数字对“x-y-z”的含义为“x 是访问的主存地址，y 是对应的主存块号，z 是对应的 cache 行号”。hit 表示命中，miss 表示缺失，miss/replace 表示缺失并替换）：2-0-0: miss；3-0-0: hit；11-2-2: miss；16-4-0: miss/replace；21-5-1: miss；13-3-3: miss；64-16-0: miss/replace；48-12-0: miss/replace；19-4-0: miss/replace；11-2-2: hit；3-0-0: miss/replace；22-5-1: hit；4-1-1: miss/replace；27-6-2: miss/replace；6-1-1: hit；11-2-2: miss/replace。共命中 4 次，命中率为 4/16=25%。数据块变大后，命中率提高了，其原因在于块变大后空间局部性得到更大发挥。

第7章 存储器层次结构

13. 假定数组元素在主存按从左到右的下标优先顺序存放。试改变下列函数中循环的顺序，使得其数组元素的访问与排列顺序一致，并说明为什么修改后的程序比原来的程序执行时间更短。

```
int sum_array(int a[N][N][N])
{
    int i, j, k, sum=0;
    for(i=0; i<N; i++)
        for(j=0; j<N; j++)
            for(k=0; k<N; k++) sum+=a[k][i][j];
    return sum;
}
```

参考答案：

数组元素的访问顺序和排列顺序一致的程序如下：

```
int sum_array ( int a[N][N][N])
{
      int i, j, k, sum=0;
      for (k=0; k < N; k++)
            for (i=0; i < N; i++)
                  for (j=0; j < N; j++)
                        sum+=a[k][i][j];
      return sum;
}
```

当被访问的数组元素不在 cache 时，则将该数组元素所在的一个主存块全部装入 cache，因为访问顺序和排列顺序一致，所以，随后访问的若干个数组元素都和该数组元素在同一个主存块中，因而也都能在 cache 中命中。因此，修改后的程序，其数组访问的空间局部性比原程序更好，命中率更高，使得执行时间更短。

16. 以下是对矩阵进行转置的程序段：

```
typedef int array[4][4];
void transpose(array dst, array src)
{
    int i, j;
    for(i=0; i<4; i++)
        for(j=0; j<4; j++) dst[j][i]=src[i][j];
}
```

假设该段程序运行的计算机中 sizeof(int)＝4，且只有一级 cache，其中 L1 数据 cache 的数据区大小为 32B，采用直接映射、回写方式，块大小为 16B，初始为空。数组 dst 从地址 0000 C000H 开始存放，数组 src 从地址 0000 C040H 开始存放。填写下表，说明对数组元素 src[row][col]和 dst[row][col]的访问是命中(hit)还是缺失(miss)。若将 L1 数据 cache 的数据区容量改为 128B，请重新填写表中内容。

	src 数组				dst 数组			
	col＝0	col＝1	col＝2	col＝3	col＝0	col＝1	col＝2	col＝3
row＝0	miss				miss			
row＝1								
row＝2								
row＝3								

参考答案：

从程序来看，数组访问过程如下：

src[0] [0]、dst[0] [0]、src[0] [1]、dst[1] [0]、src[0] [2]、dst[2] [0]、src[0] [3]、dst[3] [0]

src[1] [0]、dst[0] [1]、src[1] [1]、dst[1] [1]、src[1] [2]、dst[2] [1]、src[1] [3]、dst[3] [1]

src[2] [0]、dst[0] [2]、src[2] [1]、dst[1] [2]、src[2] [2]、dst[2] [2]、src[2] [3]、dst[3] [2]

src[3] [0]、dst[0] [3]、src[3] [1]、dst[1] [3]、src[3] [2]、dst[2] [3]、src[3] [3]、dst[3] [3]

因为块大小为 16B，每个数组元素有 4 个字节，所以 4 个数组元素占一个主存块，因此每次总是调入 4 个数组元素到 cache 的一行中。

当数据区容量为 32B 时，L1 data cache 中共有 2 行。因为地址 0000 C000H 和 0000 C040H 的最低 5 位都是 0，所以，数组元素 dst[0][i]、dst[2][i] 、src[0][i]、src[2][i] (i=0～3)都映射到 cache 第 0 行，数组元素 dst[1][i]、dst[3][i] 、src[1][i]、src[3][i] (i=0～3)都映射到 cache 第 1 行。因此，从上述访问过程来看，src[0][0]所在的主存块（即存放 src[0][i] (i=0～3)中 4 个数组元素的主存块）刚调入 cache，dst[0][0]所在主存块又把它替换掉了。

当数据区容量为 128B 时，L1 data cache 中共有 8 行。数组元素 dst[0][i]、dst[1][i] 、dst[2][i]、dst[3][i]、src[0][i]、src[1][i]、src[2][i]、src[3][i] (i=0～3) 分别映射到 cache 第 0、1、2、3、4、5、6、7 行。因此，不会发生数组元素的替换。每次总是第一个数组元素不命中，后面三个数组元素都命中。

参考答案：

表 7.2 和表 7.3 给出了 cache 数据容量分别为 32B 和 128B 时数组 src 和 dst 的每个元素的命中情况。

表 7.2　题 27 中的 src 数组和 dst 数组在 cache 数据容量分别为 32B 时的命中情况

	src 数组				dst 数组			
32B	col=0	col=1	col=2	col=3	col=0	col=1	col=2	col=3
row=0	0/miss	0/miss	0/hit	0/miss	0/miss	0/miss	0/miss	0/miss
row=1	1/miss	1/hit	1/miss	1/hit	1/miss	1/miss	1/miss	1/miss
row=2	0/miss	0/miss	0/hit	0/miss	0/miss	0/miss	0/miss	0/miss
row=3	1/miss	1/hit	1/miss	1/hit	1/miss	1/miss	1/miss	1/miss

表 7.3　题 27 中的 src 数组和 dst 数组在 cache 数据容量分别为 128B 时的命中情况

	src 数组				dst 数组			
128B	col=0	col=1	col=2	col=3	col=0	col=1	col=2	col=3
row=0	4/miss	4/hit	4/hit	4/hit	0/miss	0/hit	0/hit	0/hit
row=1	5/miss	5/hit	5/hit	5/hit	1/miss	1/hit	1/hit	1/hit
row=2	6/miss	6/hit	6/hit	6/hit	2/miss	2/hit	2/hit	2/hit
row=3	7/miss	7/hit	7/hit	7/hit	3/miss	3/hit	3/hit	3/hit

18. 假设某计算机的主存地址空间大小为 64MB，采用字节编址方式。其 cache 数据区容量为 4KB，采用 4 路组相联映射方式、LRU 替换算法和回写(write back)策略，块大小为 64B。请问：

(1) 主存地址字段如何划分？要求说明每个字段的含义、位数和在主存地址中的位置。

(2) 该 cache 的总容量有多少位？

(3) 假设 cache 初始为空，CPU 依次从 0 号地址单元顺序访问到 4344 号单元，重复按此序列共访问 16 次。若 cache 命中时间为 1 个时钟周期，缺失损失为 10 个时钟周期，则 CPU 访存的平均时间为多少时钟周期？

参考答案：

(1) cache的行数为4KB/64B=64；因为采用4路组相联，所以每组有4行，共16组。主存地址空间大小为 64MB，按字节编址，所以主存地址 26 位，其中低6位为块内地址，中间4位为组号（组索引） 高16 位为标记。

(2) 因为采用写回策略，所以cache每行中要有一个修改位（dirty bit）；因为每组有4行，所以每行有两位LRU位。此外，每行 有16位 标记、1位有效位和64B数据，共有64行，故总容量为 64×(1+2+16+1+64×8)=34048 位。

参考答案：

（3）因为块大小为64字节，CPU总共访问了4345个单元，因为4345/64＝67.89，所以第0～4344单元应该对应前68块（第0～67块），也即CPU访问过程是对主存的前68块连续访问16次。图7.3给出了访问过程中主存块和cache行之间的映射关系。图中列方向是cache的16个组，行方向是每组的4行。

	第0行	第1行	第2行	第3行
第0组	0/64/48	16/0/64	32/16	48/32
第1组	1/65/49	17/1/65	33/17	49/33
第2组	2/66/50	18/2/66	34/18	50/34
第3组	3/67/51	19/3/67	35/19	51/35
第4组	4	20	36	52
…	…	…	…	…
…	…	…	…	…
第15组	15	31	47	63

图7.3　题21中cache组和主存块之间的映射

参考答案：

主存的第0～15块分别对应cache的第0～15组，可以放在对应组的任意一行中，在此假定按顺序存放在第0行；主存的第16～31块也分别对应cache的第0～15组，放到第1行中；同理，主存的第32～47块分别放到cache的第0～15组的第2行中；主存的第48～63块分别放到cache的第0～15组的第3行中。这样，访问主存的第0～63块都是没有冲突的，每块都是第一次在cache中没有找到，然后把这一块调到cache对应组的某一行中，这样该块后面的每次访问都能在cache中找到。因此，每一块只有第一单元未命中，其余63个单元都命中。主存的第64～67块分别对应cache的第0～3组，此时，这4组的4个行都已经被主存块占满，所以这四组的每一组都要选择一个主存块从cache中淘汰出来。因为采用LRU算法，所以，将最近最少用的第0～3块分别从第0～3组的第0行中替换出来。再把第64～67块分别放到cache第0～3组的第0行中，每块也都是第一次在cache中没有找到，调入后，每次都能在cache中找到。

综上所述，第一次循环中，每一块都只有第一单元未命中，其余都命中。

以后的15次循环中，因为cache第4～15组的48行中的主存块一直没有被替换过，所以只有68−48=20个行中对应主存块的第一个单元未命中，其余都命中。

总访问次数为4345×16=69520，其中，未命中次数为68+15×20=368。

命中率p为(69520-368)/69520≈99.47%。

平均访存时间约为$t_a=t_c+(1-p)t_m=1+(1-0.9947)*10=1.053$个时钟周期

第7章 存储器层次结构

22. 假定有 3 个处理器，分别带有以下不同的 cache：

cache 1：采用直接映射方式，块大小为 1 个字，指令和数据的缺失率分别为 4%和 6%。

cache 2：采用直接映射方式，块大小为 4 个字，指令和数据的缺失率分别为 2%和 4%。

cache 3：采用 2 路组相联映射方式，块大小为 4 个字，指令和数据的缺失率分别为 2%和 3%。

在这些处理器上运行同一个程序，其中有一半是访存指令，在 3 个处理器上测得该程序的 CPI 都为 2.0。已知处理器 1 和 2 的时钟周期都为 420ps，处理器 3 的时钟周期为 450ps。若缺失损失为（块大小＋6）个时钟周期，请问：哪个处理器因 cache 缺失而引起的额外开销最大？哪个处理器执行速度最快？

参考答案：

假设所运行的程序共执行 N 条指令，每条访存指令仅读写一次内存数据，则在该程序执行过程中各处理器因 cache 缺失而引起的额外开销和执行时间计算如下。

对于处理器 1，额外开销为 N×(4%+6%×50%)×(1+6) = 0.49 N 个时钟周期，执行程序所需时间为(N×2.0+0.49N)×420ps = 1045.8N ps。

对于处理器 2，额外开销为 N×(2%+4%×50%)×(4+6) = 0.40N 个时钟周期，执行程序所需时间为 (N×2.0+0.40N)×420ps=1008Nps。

对于处理器 3，额外开销为 N×(2%+3%×50%)×(4+6) = 0.35N 个时钟周期，执行程序所需时间为 (N×2.0+0.35N)×450ps=1057.5N ps。

由此可见，处理器 1 的 cache 缺失引起的额外开销最大，处理器 2 的执行速度最快。

25. 假定一个计算机系统中有一个 TLB 和一个 L1 数据 cache。该系统按字节编址，虚拟地址 16 位，物理地址 12 位，页大小为 128B；TLB 采用 4 路组相联方式，共有 16 个页表项；L1 数据 cache 采用直接映射方式，块大小为 4B，共 16 行。在系统运行到某一时刻时，TLB、页表和 L1 数据 cache 中的部分内容如下：

组号	标记	页框号	有效位	标记	页框号	有效位	标记	页框号	有效位	标记	页框号	有效位
0	03	—	0	09	0D	1	00	—	0	07	02	1
1	13	2D	1	02	—	0	04	—	0	0A	—	0
2	02	—	0	08	—	0	06	—	0	03	—	0
3	07	—	0	63	0D	1	0A	34	1	72	—	0

(a) TLB(4 路组相联)：4 组、16 个页表项

虚页号	页框号	有效位
00	08	1
01	03	1
02	14	1
03	02	1
04	—	0
05	16	1
06	—	0
07	07	1
08	13	1
09	17	1
0A	09	1
0B	—	0
0C	19	1
0D	—	0
0E	11	1
0F	0D	1

(b) 部分页表(开始 16 项)

行索引	标记	有效位	字节 3	字节 2	字节 1	字节 0
0	19	1	12	56	C9	AC
1	—	0	—	—	—	—
2	1B	1	03	45	12	CD
3	—	0	—	—	—	—
4	32	1	23	34	C2	2A
5	0D	1	46	67	23	3D
6	—	0	—	—	—	—
7	16	1	12	54	65	DC
8	24	1	23	62	12	3A
9	—	0	—	—	—	—
A	2D	1	43	62	23	C3
B	—	0	—	—	—	—
C	12	1	76	83	21	35
D	16	1	A3	F4	23	11
E	33	1	2D	4A	45	55
F	—	0	—	—	—	—

(c) L1 数据 cache：直接映射，共 16 行，块大小为 4B

请问(假定图中数据都为十六进制形式)：

(1) 虚拟地址中哪几位表示虚拟页号？哪几位表示页内偏移量？虚拟页号中哪几位表示 TLB 标记？哪几位表示 TLB 索引？

(2) 物理地址中哪几位表示物理页号？哪几位表示页内偏移量？

(3) 主存物理地址如何划分成标记字段、行索引字段和块内地址字段？

(4) CPU 从地址 067AH 中取出的值为多少？说明 CPU 读取地址 067AH 中内容的过程。

参考答案：

（1）16 位虚拟地址中低 7 位为页内偏移量，高 9 位为虚页号；虚页号中高 7 位为 TLB 标记，低 2 位为 TLB 组索引。

(2) 12位物理地址中低7位为页内偏移量，高5位为物理页号；
(3) 12位物理地址中，低2位为块内地址，中间4位为cache行索引，高6位为标记。

(4) 地址 067AH=0000 0110 0111 1010B，所以，虚页号为 000001100B，映射到 TLB 的第 0 组，将 0000011B=03H 与 TLB 第 0 组的 4 个标记比较，虽然和其中一个相等，但对应的有效位为 0，其余都不等，所以 TLB 缺失，需要访问主存中的慢表。直接查看 000001100B =00CH 处的页表项，有效位为 1，取出物理页号 19H=11001B，和页内偏移 1111010B 拼接成物理地址 110011 1110 10B。根据中间 4 位 1110 直接找到 cache 第 14 行(即第 E 行)，其有效位为 1，且标记为 33H=110011B，正好等于物理地址高 6 位，故 cache 命中。最后根据物理地址最低两位 10，取出字节 2 中的内容 4AH=01001010B。

习题课

- 第5章 中央处理器
- 第6章 指令流水线
- 第7章 存储器层次结构
- **第8章 系统互连及输入输出组织**

(感谢计算机系袁春风老师提供的许多习题答案！)

7. 某计算机 CPU 主频为 500MHz，所连接的某外设最大数据传输率为 20KB/s，该外设接口中有一个 16 位的数据缓存器，相应的中断服务程序执行时间为 500 个时钟周期，是否可以用中断方式进行该外设的输入输出？假定该外设的最大数据传输率改为 2MB/s，是否可以用中断方式进行该外设的输入输出？

参考答案：

（1）因为该外设接口中有一个 16 位数据缓存器，所以，若用中断方式进行输入/输出的话，可以每 16 位数据进行一次中断请求，因此，中断请求的时间间隔为 $10^6\times2/(20\times10^3)=100\mu s$。 **$10^6\times16/(20\times10^3\times8)=100$微秒**

对应的中断服务程序的执行时间为$(10^6/500\times10^6)\times500=1\mu s$，因为中断响应过程就是执行一条隐指令的过程，所用时间相对于中断处理时间（即执行中断服务程序的时间）而言，几乎可以忽略不计，因而整个中断响应并处理的时间大约 1μs 多一点，远远小于中断请求的间隔时间。因此，可以用中断方式进行该外设的输入输出。

（2）若外设的最大传输率为 2MB/s，则中断请求的时间间隔为 $10^6\times2/(2\times10^6)=1\mu s$。而整个中断响应并处理的时间大约 1μs 多一点，中断请求的间隔时间小于中断响应和处理时间，也即中断处理还未结束就会有该外设新的中断到来，所以不可以用中断方式进行该外设的输入输出。

8. 若某计算机有 5 个中断源 1＃、2＃、3＃、4＃、5＃，中断响应优先级为 1＃＞2＃＞3＃＞4＃＞5＃，而中断处理优先级为 1＃＞4＃＞5＃＞2＃＞3＃。要求：

(1) 设计各中断源对应的中断屏蔽字(假设 0 为屏蔽，1 为开放)。

(2) 若在运行主程序时，同时出现 2＃、4＃中断请求，在处理 2＃中断过程中，又同时出现 1＃、3＃、5＃中断请求，试画出 CPU 的运行过程示意图。

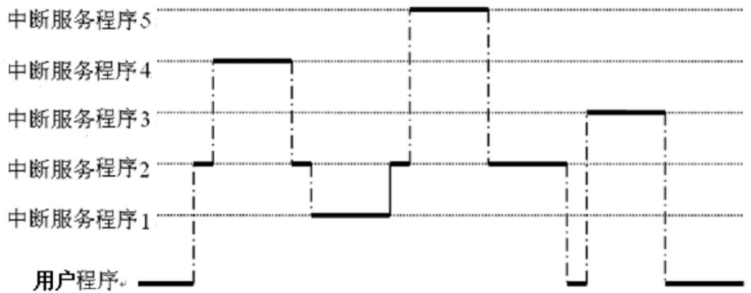
参考答案：

(1)

中断处理程序	中断屏蔽字				
	第 1 号	第 2 号	第 3 号	第 4 号	第 5 号
第 1 号	1	0	0	0	0
第 2 号	1	1	0	1	1
第 3 号	1	1	1	1	1
第 4 号	1	0	0	1	0
第 5 号	1	0	0	1	1

参考答案：

（2）在运行用户程序时，同时出现中断源 2 和 4，因为用户程序对所有中断源都开放，所以，在中断响应优先级排队电路中，中断源 2 和 4 进行排队判优，根据中断响应优先级 2>4，因此先响应 2 号中断源。在 CPU 执行中断源 2 的中断服务程序过程中，首先保护现场、保护旧屏蔽字、设置新的屏蔽字 11011，然后，在具体中断处理前先开中断。一旦开中断，则马上响应 4 号中断源，因为第 2 号的中断屏蔽字中对第 4 号中断源的屏蔽位是 1，也即对第 4 号中断源是开放的。在第 4 号中断处理结束后，回到第 2 号中断源的中断服务程序执行；在具体处理第 2 号中断过程中，同时发生了第 1、3、5 号中断源请求，因为第 2 号中断对第 1、5 号中断开放，对第 3 号中断屏蔽，所以只有第 1 和第 5 两个中断源进行排队判优，根据中断响应优先级 1>5，所以先响应第 1 号中断源。因为第 1 号中断处理优先级最高，所以在其处理过程中不会响应任何新的中断请求，直到第号级中断处理结束，然后返回第 2 号中断；因为第 2 号中断对第 5 号中断开放，所以在第 2 号中断服务程序中执行一条指令后，又转去执行第 5 号中断服务程序，执行完后回到第 2 号中断，在第 2 号中断服务程序执行过程中，虽然第 3 号中断有请求，但是，因为第 2 号中断对第 3 号中断不开放，所以，第 3 号中断一直得不到响应。直到第 2 号中断处理完回到用户程序，才能响应并处理第 3 号中断。CPU 运行过程如 下图 所示。



9. 假定某计算机字长 16 位，没有 cache，运算器一次定点加法时间等于 100ns，配置的磁盘旋转速度为每分钟 3000 转，每个磁道上记录两个数据块，每一块有 8000 字节，两个数据块之间间隙的越过时间为 2ms，主存存储周期为 500ns，存储器总线宽度为 16 位，总线带宽为 4MB/s。

（1）磁盘读写数据时的最大数据传输率是多少？

（2）当磁盘按最大数据传输率与主机交换数据时，主存存储周期空闲百分比是多少？

（3）直接寻址的“存储器-存储器”SS 型加法指令在无磁盘 I/O 操作打扰时的执行时间为多少？当磁盘 I/O 操作与一连串这种 SS 型加法指令执行同时进行时，则这种 SS 型加法指令的最快和最慢执行时间各是多少？（假定采用多周期处理器方式，CPU 时钟周期等于主存存储周期。）

参考答案：

（1）磁盘旋转一周所需时间为 $60\times10^3/3000=20ms$，单个数据块的传输时间为$(20/2)-2=8ms$，所以最大数据传输率为 $8000/(8/1000)=1MB/s$。

（2）磁盘最大数据传输率为 1MB/s，存储器总线宽度为 16b=2B，故每隔 $10^9\times2/(1\times10^6)=2000ns$ 产生一个 DMA 请求，即每 2000/500=4 个主存周期中有一个被 DMA 挪用，此时，CPU 没有访问主存，因此，4 个主存周期中有 3 个空闲，故主存频带空闲百分比是 75%，如图 8.5 所示。图中箭头处开始的一个主存周期被 DMA 挪用。

图 8.5　无 CPU 访存时主存周期被 DMA 使用的情况

参考答案：

（3）无 I/O 打扰时，执行一条直接寻址的 SS 型加法指令的过程如图 8.6 所示，包括取指令、取源操作数 1、取目操作数（源操作数 2）、执行、写结果，因此执行时间为 5×500ns=2500ns=2.5μs。此时，每个指令周期所包含的 5 个时钟周期中，只有执行阶段不访问主存，所以主存频带空闲百分比是 20%。

图 8.6　无 I/O 打扰时主存周期被 CPU 使用的情况

当磁盘 I/O 操作与一连串这种 SS 型加法指令同时进行时，可能因为 CPU 和 DMA 同时访存而使指令的执行时间被延长。每次 DMA 请求要求挪用一个主存周期来访问主存，同时，CPU 执行指令时也要求访问主存，当两者发生冲突时，DMA 优先级高，CPU 的访存请求被延迟。因为每隔 2000ns 产生一个 DMA 请求，因此每 4 个主存周期必定有一个被 DMA 所挪用。此时，主存周期的占用情况如图 8.7 所示。

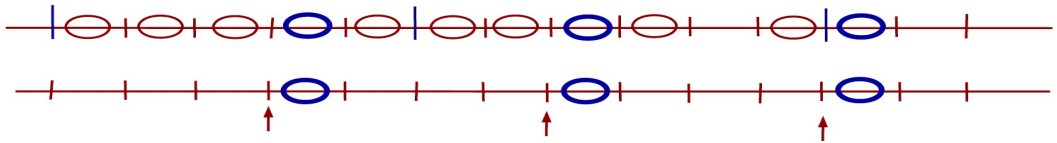


图 8.7　CPU 和 DMA 交替串行访问主存时的情况

由图 8.7 可知，最好的情况是在 SS 型加法指令执行过程中没有访存冲突（如 8.7 中最开始的一个指令周期），此时最快，指令执行时间为 2.5μs；最坏的情况是有一次访存冲突（如图 8.7 中第二个指令周期），此时最慢，指令执行时间为 2.5 +500/1000=3μs。

第8章 系统互联及输入输出组织

10. 假定某计算机所有指令都可用两个总线周期完成，一个总线周期用来取指令，另一个总线周期用来存取数据。总线周期为 250ns，因而每条指令的执行时间为 500ns。若该计算机中配置的磁盘上每个磁道有 16 个 512 字节的扇区，磁盘旋转一圈的时间是 8.192ms，则采用周期挪用法进行 DMA 传送时，总线宽度为 8 位和 16 位的情况下该计算机指令执行速度分别降低了百分之几？

参考答案：

(1) 磁盘的平均数据传输率为 $16\times512B/(8.192/10^3)=1000000B/s=1MB/s$。当总线位宽为 8 位时，DMA 控制器每隔 $1/(1\times10^6)=10^{-6}s=1\mu s$ 申请一次数据传送，在 1μs 期间 CPU 共执行 1/(500/1000)=2 条指令。因此，每两条指令的执行被插入一个总线周期用于一次数据传送，也即平均每条指令延长了 250/2=125ns。因而，计算机执行指令的速度降低了 125/500=25%。

(2) 当总线位宽为 16 位时，DMA 控制器每隔 $2/(1\times10^6)=2\times10^{-6}s=2\mu s$ 申请一次数据传送，在 2μs 期间 CPU 共执行 2/(500/1000)=4 条指令，因此，每 4 条指令的执行被插入一个总线周期用于一次数据传送，也即平均每条指令延长了 250/4=62.5ns。因而，计算机执行指令的速度降低了 62.5/500=12.5%。

第8章 系统互联及输入输出组织

12. 假定采用中断控制 I/O 方式，则以下各项工作是在 4 个 I/O 软件层的哪一层完成的？

(1) 根据逻辑块号计算磁盘物理地址(柱面号、磁头号、扇区号)。

(2) 检查用户是否有权读写文件。

(3) 将二进制整数转换为 ASCII 码以便打印输出。

(4) CPU 向设备控制器写入控制命令(如“启动工作”命令)。

(5) CPU 从设备控制器的数据端口读取数据。

参考答案：

(1) 与设备无关的I/O软件

(2) 与设备无关的I/O软件

(3) 用户空间I/O软件

(4) 设备驱动程序

(5) 设备驱动程序

提问

Q & A

殷亚凤
智能软件与工程学院
苏州校区南雍楼东区225
yafeng@nju.edu.cn，https://yafengnju.github.io/

南京大学 NANJING UNIVERSITY