CSAPP题目

笔记本:	考研专业课		
创建时间:	2018/11/14 20:06	**更新时间:**	2018/11/28 19:47
作者:	15801850335@163.com		

题目链接:

https://wenku.baidu.com/view/6edb4f2608a1284ac85043e8.html

https://wenku.baidu.com/view/2d490d88cfc789eb172dc8e9.html

https://wenku.baidu.com/view/f50294ea4128915f804d2b160b4e767f5acf8031.html

常用的降低Cache失效率的方法有下面几种:

(1) 增加Cache块大小。增加块大小利用了程序的空间局部性。

(2) 增加Cache的容量。

(3) 提高相联度，降低冲突失效。

(4) 伪相联Cache，降低冲突失效。当对伪相联Cache进行访问时，首先是按与直接映象相同的方式进行访问。如果命中，则从相应的块中取出所访问的数据，送给CPU，访问结束。如果不命中，就将索引字段的最高位取反，然后按照新索引去寻找"伪相联组"中的对应块。如果这一块的标识匹配，则称发生了"伪命中"。否则，就访问下一级存储器。

(5) 硬件预取技术。在处理器提出访问请求前预取指令和数据。

(6) 由编译器控制的预取，硬件预取的替代方法，在编译时加入预取的指令，在数据被用到之前发出预取请求。

(7) 编译器优化，通过对软件的优化来降低失效率。

(8) "牺牲"Cache。在Cache和其下一级存储器的数据通路之间增设一个全相联的小Cache，存放因冲突而被替换出去的那些块。每当发生不命中时，在访问下一级存储器之前，先检查"牺牲"Cache中是否含有所需的块。如果有，就将该块与Cache中某个块做交换，把所需的块从"牺牲"Cache调入Cache。

3. 地址映象方法有哪几种？它们各有什么优缺点？

答:(1)全相联映象：实现查找的机制复杂，代价高，速度慢。Cache 空间的利用率较高，块冲突概率较低，因而Cache 的失效率也低。

(2)直接映象:实现查找的机制简单，速度快。Cache 空间的利用率较低，块冲突概率较高，因而Cache 的失效率也高。

(3)组相联映象:组相联是直接映象和全相联的一种折衷

4 指令的执行可采用顺序执行、重叠执行和流水线三种方式，是分析说明它们的主要区别及优缺点。

答:(1) 指令的顺序执行是指指令与指令之间顺序串行。即上一条指令全部执行完后，才能开始执行下一条指令。

优点:控制简单，节省设备。缺点:执行指令的速度慢，功能部件的利用率低。

(2) 指令的重叠指令是在相邻的指令之间，让第k 条指令与取第 k+1 条指令同时进行。重叠执行不能加快单条指令的执行速度，但在硬件增加不多的情况下，可以加快相邻两条指令以及整段程序的执行速度。与顺序方式相比，功能部件的利用率提高了，控制变复杂了。

(3) 指令的流水执行是把一个指令的执行过程分解为若干个子过程，每个子过程由专门的功能部件来实现。把多个处理过程在时间上错开，依次通过各功能段，每个子过程与其它的子过程并行进行。依靠提高吞吐率来提高系统性能。流水线中各段的时间应尽可能相等。

2 分别从执行程序的角度和处理数据的角度来说明计算机系统中并行性等级从低到高可分为哪几级？

答：答：从处理数据的角度来看，并行性等级从低到高可分为：

（1）字串位串：每次只对一个字的一位进行处理。这是最基本的串行处理方式，不存在并行性；

（2）字串位并：同时对一个字的全部位进行处理，不同字之间是串行的。已开始出现并行性；

（3）字并位串：同时对许多字的同一位（称为位片）进行处理。这种方式具有较高的并行性；

（4）全并行：同时对许多字的全部位或部分位进行处理。这是最高一级的并行。

从执行程序的角度来看，并行性等级从低到高可分为：

（1）指令内部并行：单条指令中各微操作之间的并行；

（2）指令级并行：并行执行两条或两条以上的指令；

（3）线程级并行：并行执行两个或两个以上的线程，通常是以一个进程内派生的多个线程为调度单位；

（4）任务级或过程级并行：并行执行两个或两个以上的过程或任务(程序段)，以子程序或进程为调度单元；

（5）作业或程序级并行：并行执行两个或两个以上的作业或程序。

3 试说明计算机系统结构、计算机组成与计算机实现之间的相互关系。

答：计算机系统结构、计算机组成、计算机实现互不相同，但又相互影响。

（1）计算机的系统结构相同，但可采用不同的组成。如IBM370 系列有 115、125、135、158、168 等高档的多种型号机器。从汇编语言、机器语言程序设计者看到的概念性结构相同，均是由中央处理存，通道、设备控制器，外设4级构成。其中，中央处理机都有相同的机器指令和汇编指令系统，只的分析、执行在低档机上采用顺序进行，在高档机上采用重叠、流水或其它并行处理方式。 （2）相同的组成可有多种不同的实现。如主存器件可用双极型的，也可用 MOS 型的；可用 VLSI 单片多片小规模集成电路组搭。

（3）计算机的系统结构不同，会使采用的组成技术不同，反之组成也会影响结构。如为实现A:=B+可采用面向寄存器的系统结构，也可采用面向主存的三地址寻址方式的系统结构。要提高运行速度加与相乘并行，为此这两种结构在组成上都要求设置独立的加法器和乘法器。但对面向寄存器的系要求寄存器能同时被访问，而对面向主存的三地址寻址方式的系统结构并无此要求，倒是要求能同个访存操作数地址和能同时访存。

4 计算机系统设计中经常使用的 4 个定量原理是什么？并说出它们的含义。

答：（1）以经常性事件为重点。在计算机系统的设计中，对经常发生的情况，赋予它优先的处理权和资源使用权，以得到更多的总体上的改进。（2）Amdahl 定律。加快某部件执行速度所获得的系统性能加速比，受限于该部件在系统中所占的重要性。（3）CPU 性能公式。执行一个程序所需的 CPU 时间 = IC × CPI ×时钟周期时间。（4）程序的局部性原理。程序在执行时所访问地址的分布不是随机的，而是相对地簇聚

5. 简述流水线技术的特点。

答：流水技术有以下特点：

（1） 流水线把一个处理过程分解为若干个子过程，每个子过程由一个专门的功能部件来实现。因此，流水线实际上是把一个大的处理功能部件分解为多个独立的功能部件，并依靠它们的并行工作来提高吞吐率。

（2） 流水线中各段的时间应尽可能相等，否则将引起流水线堵塞和断流。

（3） 流水线每一个功能部件的前面都要有一个缓冲寄存器，称为流水寄存器。

（4） 流水技术适合于大量重复的时序过程，只有在输入端不断地提供任务，才能充分发挥流水线的效率。

（5） 流水线需要有通过时间和排空时间。在这两个时间段中，流水线都不是满负荷工作。

6. 试说明存储层次设计要解决的四个问题。

答：映像规则、查找算法、替换算法、写策略

1.4 计算机系统设计中经常使用的 4 个定量原理是什么？并说出它们的含义。

答：（1）以经常性事件为重点。在计算机系统的设计中，对经常发生的情况，赋予它优先的处理权和资源使用权，以得到更多的总体上的改进。（2）Amdahl 定律。加快某部件执行速度所获得的系统性能加速比，受限于该部件在系统中所占的重要性。（3）CPU 性能公式。执行一个程序所需的 CPU 时间 $= IC \times CPI \times$时钟周期时间。（4）程序的局部性原理。程序在执行时所访问地址的分布不是随机的，而是相对地簇聚。

1.5 分别从执行程序的角度和处理数据的角度来看，计算机系统中并行性等级从低到高可分为哪几级？

答：从处理数据的角度来看，并行性等级从低到高可分为：

（1）字串位串：每次只对一个字的一位进行处理。这是最基本的串行处理方式，不存在并行性；

（2）字串位并：同时对一个字的全部位进行处理，不同字之间是串行的。已开始出现并行性；

（3）字并位串：同时对许多字的同一位（称为位片）进行处理。这种方式具有较高的并行性；

（4）全并行：同时对许多字的全部位或部分位进行处理。这是最高一级的并行。

从执行程序的角度来看，并行性等级从低到高可分为：

（1）指令内部并行：单条指令中各微操作之间的并行；

（2）指令级并行：并行执行两条或两条以上的指令；

（3）线程级并行：并行执行两个或两个以上的线程，通常是以一个进程内派生的多个线程为调度单位；

（4）任务级或过程级并行：并行执行两个或两个以上的过程或任务（程序段），以子程序或进程为调度单元；

（5）作业或程序级并行：并行执行两个或两个以上的作业或程序。

吞吐率：在单位时间内流水线所完成的任务数量或输出结果的数量。

流水线的加速比：使用顺序处理方式处理一批任务所用的时间与按流水处理方式处理同一批任务所用的时间之比。

流水线的效率：即流水线设备的利用率，它是指流水线中的设备实际使用时间与整个运行时

间的比值。

结构冲突：因硬件资源满足不了指令重叠执行的要求而发生的冲突。

数据冲突：当指令在流水线中重叠执行时，因需要用到前面指令的执行结果而发生的冲突。

控制冲突：流水线遇到分支指令或其它会改变 PC 值的指令所引起的冲突。

3.2 指令的执行可采用顺序执行、重叠执行和流水线三种方式，它们的主要区别是什么？各有何优缺点。

答：（1）指令的顺序执行是指指令与指令之间顺序串行。即上一条指令全部执行完后，才能开始执行下一条指令。

优点：控制简单，节省设备。缺点：执行指令的速度慢，功能部件的利用率低。

（2）指令的重叠指令是在相邻的指令之间，让第 k 条指令与取第 k+l 条指令同时进行。重叠执行不能加快单条指令的执行速度，但在硬件增加不多的情况下，可以加快相邻两条指令以及整段程序的执行速度。与顺序方式相比，功能部件的利用率提高了，控制变复杂了。

（3）指令的流水执行是把一个指令的执行过程分解为若干个子过程，每个子过程由专门的功能部件来实现。把多个处理过程在时间上错开，依次通过各功能段，每个子过程与其它的子过程并行进行。依靠提高吞吐率来提高系统性能。流水线中各段的时间应尽可能相等

3.5 简述流水线技术的特点。

答：流水技术有以下特点：

（1） 流水线把一个处理过程分解为若干个子过程，每个子过程由一个专门的功能部件来实现。因此，流水线实际上是把一个大的处理功能部件分解为多个独立的功能部件，并依靠它们的并行工作来提高吞吐率。

（2） 流水线中各段的时间应尽可能相等，否则将引起流水线堵塞和断流。

（3） 流水线每一个功能部件的前面都要有一个缓冲寄存器，称为流水寄存器。

（4） 流水技术适合于大量重复的时序过程，只有在输入端不断地提供任务，才能充分发挥流水线的效率。

（5） 流水线需要有通过时间和排空时间。在这两个时间段中，流水线都不是满负荷工作。

2.2 区别不同指令集结构的主要因素是什么？根据这个主要因素可将指令集结构分为哪类？

答：区别不同指令集结构的主要因素是CPU中用来存储操作数的存储单元。据此可将指令系统结构分为堆栈结构、累加器结构和通用寄存器结构。

1.2 某台主频为 400MHz 的计算机执行标准测试程序，程序中指令类型、执行数量和平均时钟周期数如下：

指令类型	指令执行数量	平均时钟周期数
整数	45000	1
数据传送	75000	2
浮点	8000	4
分支	1500	2

求该计算机的有效 CPI、MIPS 和程序执行时间。

1. CPI =(45000*1+75000*2+8000*4+1500*2)/129500=1.776

2. MIPS 速率=f/CPI=400/1776=225.23MIPS

3. 程序执行时间= (45000*1+75000*2+8000*4+1500*2)/400=0.000575s

1、某台主频为 400MHz 的计算机执行标准测试程序，程序中指令类型、执行数量和平均时钟周期数如下：

指令类型	指令执行数量	平均时钟周期数
整数	45000	1
数据传送	75000	2
浮点	8000	4
分支	1500	2

求该计算机的有效 CPI、MIPS 和程序执行时间。（10 分）

解：（1）CPI =(45000×1 + 75000×2 + 8000×4 + 1500×2)/ 129500 = 1.776

（2）MIPS 速率 = f/ CPI = 400/1.776 = 225.225MIPS

（3）程序执行时间= (45000×1 + 75000×2 + 8000×4 + 1500×2) / (400*1000000)

2、地址映象方法有哪几种？它们各有什么优缺点？降低 Cache 失效率有哪几种方法？简述其基本思想（15 分）

答：(1) 全相联映象。实现查找的机制复杂，代价高，速度慢。Cache 空间的利用率较高，块冲突概率较低，因而 Cache 的失效率也低。(2) 直接映象。实现查找的机制简单，速度快。Cache 空间的利用率较低，块冲突概率较高，因而 Cache 的失效率也高。(3) 组相联映象。组相联是直接映象和全相联的一种折衷。

常用的降低 Cache 失效率的方法有下面几种：

（1） 增加 Cache 块大小。增加块大小利用了程序的空间局部性。

（2） 增加 Cache 的容量。

（3） 提高相联度，降低冲突失效。

（4） 硬件预取技术。在处理器提出访问请求前预取指令和数据。

（5） 由编译器控制的预取，硬件预取的替代方法，在编译时加入预取的指令，在数据被用到之前发出预取请求。

（6） 编译器优化，通过对软件的优化来降低失效率。

（7） “牺牲”Cache。在 Cache 和其下一级存储器的数据通路之间增设一个全相联的小 Cache，存放因冲突而被替换出去的那些块。每当发生不命中时，在访问下一级存储器之前，先检查“牺牲”Cache 中是否含有所需的块。如果有，就将该块与 Cache 中某个块做交换，把所需的块从“牺牲”Cache 调入 Cache。

1.3 计算机系统中有三个部件可以改进，这三个部件的部件加速比为：

部件加速比 $_1$=30； 部件加速比 $_2$=20； 部件加速比 $_3$=10

• 如果部件 1 和部件 2 的可改进比例均为 30%，那么当部件 3 的可改进比例为多少时，系统加速比才可以达到 10？

• 如果三个部件的可改进比例分别为 30%、30%和 20%，三个部件同时改进，那么系统中不可加速部分的执行时间在总执行时间中占的比例是多少？

1. 在多个部件可改进情况下Amdahl 定理的扩展:

$$T_e = T_o\left[(1-f_e)+\frac{f_e}{S_e}\right]$$

$$S = \frac{1}{(1-f_e)+\frac{f_e}{S_e}}$$

$$S = \frac{1}{(1-\sum_i f_i)+\sum_i \frac{f_i}{S_i}}$$

式中，fi 为可加速部件 i 在未优化系统中所占的比例；Si 是部件 i 的加速比。

$$S = \left\{[1-(f_1+f_2+f_3)]+\frac{f_1}{S_1}+\frac{f_2}{S_2}+\frac{f_3}{S_3}\right\}^{-1}$$

$$10 = \left\{[1-(0.3+0.3+f_3)]+\frac{0.3}{30}+\frac{0.3}{20}+\frac{f_3}{30}\right\}^{-1}$$

$$f_3 = \frac{65}{180} = 0.36$$

2

$$p = \frac{[1-(0.3+0.3+0.2)]T}{\frac{0.3T}{30}+\frac{0.3T}{20}+\frac{0.2T}{10}+0.2T}$$

$$= \frac{0.2}{\frac{0.3}{30}+\frac{0.3}{20}+\frac{0.2}{10}+0.2}$$

$$= \frac{0.2}{\frac{0.6}{60}+\frac{0.9}{60}+\frac{1.2}{60}+\frac{12}{60}}$$

$$= \frac{12}{14.7} = 0.82$$

1、假设某应用程序中有 4 类操作，通过改进，各操作获得不同的性能提高。具体数据如下表所示：

操作类型	程序中的数量（百万条指令）	改进前的执行时间（周期）	改进后的执行时间（周期）
操作 1	10	2	1
操作 2	30	20	15
操作 3	35	10	3
操作 4	15	4	1

（1）改进后，各类操作的加速比分别是多少？

（2）各类操作单独改进后，程序获得的加速比分别是多少？

（3）4 类操作均改进后，整个程序的加速比是多少？

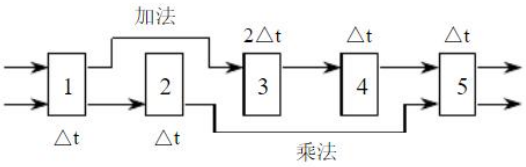
1、解：根据 Amdahl 定律 $S_n = \dfrac{1}{(1-Fe)+\dfrac{Fe}{Se}}$ 可得

操作类型	各类操作的指令条数在程序中所占的比例 F_i	各类操作的加速比 S_i	各类操作单独改进后，程序获得的加速比
操作 1	11.1%	2	1.06
操作 2	33.3%	1.33	1.09
操作 3	38.9%	3.33	1.37
操作 4	16.7%	4	1.14

4 类操作均改进后，整个程序的加速比：

$$S_n = \frac{1}{(1-\sum F_i)+\sum \dfrac{F_i}{S_i}} \approx 2.16$$

2、有一条静态多功能流水线由 5 段组成，加法用 1、3、4、5 段，乘法用 1、2、5 段，第 3 段的时间为 2△t，其余各段的时间均为△t，而且流水线的输出可以直接返回输入端或暂存于相应的流水寄存器中。现要在该流水线上计算 $\prod_{i=1}^{4}(A_i+B_i)$，画出其时空图，并计算其吞吐率、加速比和效率。

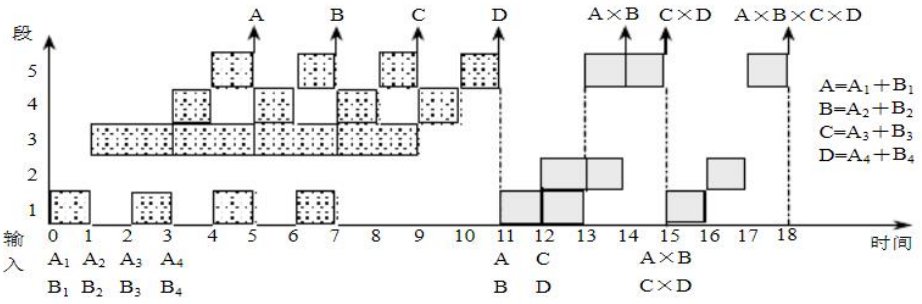


2、解:（1）会发生流水线阻塞情况。 **1 分**

Instr.1	stage1	stage2	stage3	stage3	stage4						
instr.2		stage1	stage2	stall	stage3	stage3	stage4				
instr.3			stage1	stall	stage2	stall	stage3	stage3	stage4		
instr.4				stall	stage1	stall	stage2	stall	stage3	stage3	stage4

（2）解：首先，应选择适合于流水线工作的算法。对于本题，应先计算 A_1+B_1、A_2+B_2、A_3+B_3 和 A_4+B_4；再计算$(A_1+B_1)\times(A_2+B_2)$和$(A_3+B_3)\times(A_4+B_4)$；然后求总的结果。

其次，画出完成该计算的时空图，如图所示，图中阴影部分表示该段在工作。



由图可见，它在 18 个△t 时间中，给出了 7 个结果。所以吞吐率为：

$$TP = \frac{7}{18\Delta t}$$

如果不用流水线，由于一次求积需 3△t，一次求和需 5△t，则产生上述 7 个结果共需（4×5+3×3）△t =29△t。所以加速比为：

$$S = \frac{29\Delta t}{18\Delta t} = 1.61$$

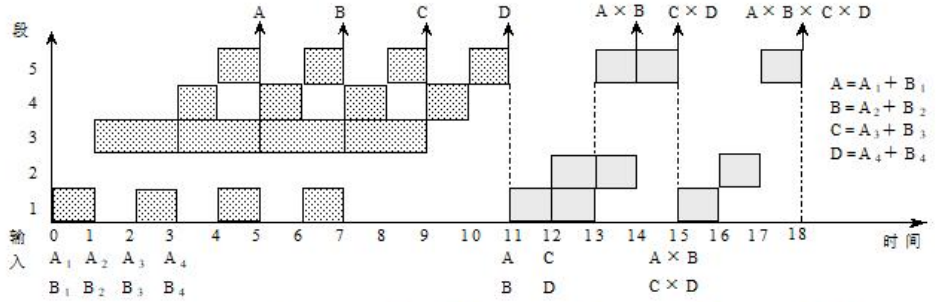
该流水线的效率可由阴影区的面积和 5 个段总时空区的面积的比值求得：

$$E = \frac{4\times5+3\times3}{5\times18} = 0.322$$

解:

首先，应选择适合于流水线工作的算法。对于本题，应先计算 $A_1 + B_1$、$A_2 + B_2$、$A_3 + B_3$ 和 $A_4 + B_4$；再计算$(A_1 + B_1) \times (A_2 + B_2)$和$(A_3 + B_3) \times (A_4 + B_4)$；然后求总的结果。

其次，画出完成该计算的时空图，如图所示，图中阴影部分表示该段在工作。



由图可见，它在 18 个△t 时间中，给出了 7 个结果。所以吞吐率为：

$$TP = \frac{7}{18\Delta t}$$

如果不用流水线，由于一次求积需 3△t，一次求和需 5△t，则产生上述 7 个结果共需（4×5+3×3）△t =29△t。所以加速比为：

$$S = \frac{29\Delta t}{18\Delta t} = 1.61$$

该流水线的效率可由阴影区的面积和 5 个段总时空区的面积的比值求得：

$$E = \frac{4\times5+3\times3}{5\times18} = 0.322$$

5.12 假设一台计算机具有以下特性：

（1）95%的访存在 Cache 中命中；

（2）块大小为两个字，且失效时整个块被调入；

（3）CPU 发出访存请求的速率为 10^9 字/s；

（4）25%的访存为写访问；

（5）存储器的最大流量为 10^9 字/s（包括读和写）；

（6）主存每次只能读或写一个字；

（7）在任何时候，Cache 中有 30%的块被修改过；

（8）写失效时，Cache 采用按写分配法。

现欲给该计算机增添一台外设，为此首先想知道主存的频带已用了多少。试对于以下两种情况计算主存频带的平均使用比例。

（1）写直达 Cache；

（2）写回法 Cache。

解：采用按写分配

（1）写直达 cache 访问命中，有两种情况：

读命中，不访问主存；

写命中，更新 cache 和主存，访问主存一次。

访问失效，有两种情况：

读失效，将主存中的块调入 cache 中，访问主存两次；

写失效，将要写的块调入 cache，访问主存两次，再将修改的数据写入 cache 和主存，访问主存一次，共三次。上述分析如下表所示。

访问命中	访问类型	频率	访存次数
Y	读	95%*75%=71.3%	0
Y	写	95%*25%=23.8%	1
N	读	5%*75%=3.8%	2
N	写	5%*25%=1.3%	3

一次访存请求最后真正的平均访存次数=(71.3%*0)+(23.8%*1)+(3.8%*2)+(1.3%*3)＝0.35

已用带宽=$0.35\times10^9/10^9$=35.0%

（2）写回法 cache 访问命中,有两种情况：

读命中，不访问主存；

写命中，不访问主存。采用写回法，只有当修改的 cache 块被换出时，才写入主存；

访问失效,有一个块将被换出，这也有两种情况：

如果被替换的块没有修改过，将主存中的块调入 cache 块中，访问主存两次；

如果被替换的块修改过，则首先将修改的块写入主存，需要访问主存两次；然后将主存中的块调入 cache 块中，需要访问主存两次，共四次访问主存。

访问命中	块为脏	频率	访存次数
Y	N	95%*70%=66.5%	0
Y	Y	95%*30%=28.5%	0
N	N	5%*70%=3.5%	2
N	Y	5%*30%=1.5%	4

所以：

一次访存请求最后真正的平均访存次数=66.5%*0＋28.5%*0+3.5%*2+1.5%*4=0.13

已用带宽＝$0.13\times10^9/10^9$=13%

(b)思考下面的程序，它视图使用一对信号量来实现互斥。

初始时：s=1,t=0

线程 1：P(s)→V(s)→P(t)→V(t)

线程 2：P(s)→V(s)→P(t)→*(t)

(1)画出这个程序的进度图。

(2)它总是死锁吗？请分析原因

(3)如果是，那么对初始信号量的值做那些改变就能消除潜在的死锁呢？

(4)画出得到的无死锁程序的进度条。

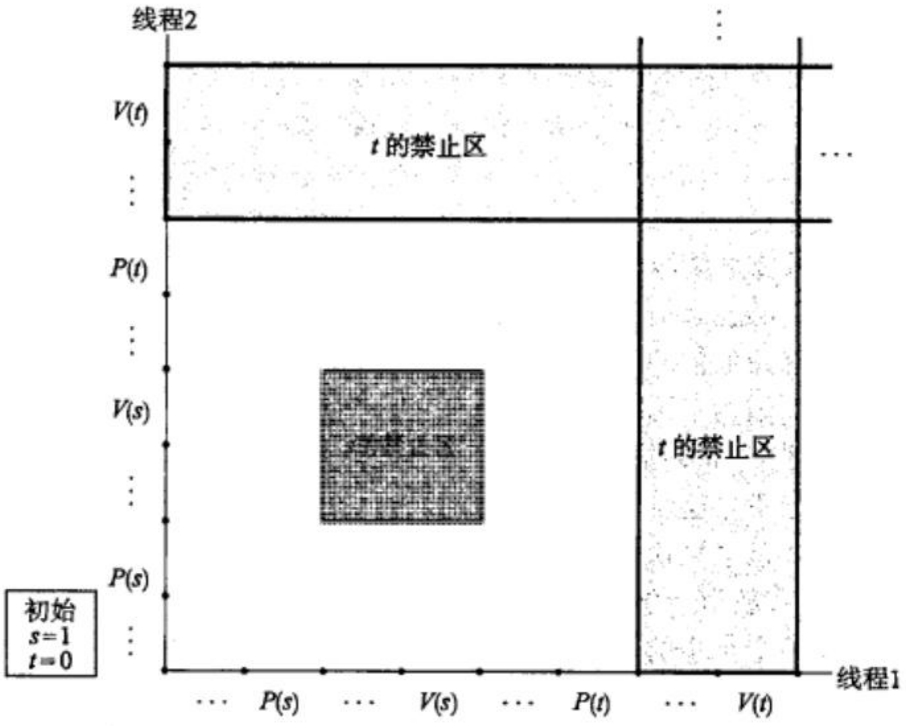


图 12-46　一个有死锁的程序进度图

（2）因为任何可行轨迹最终都将陷入死锁状态，所以这个程序总会发生死锁。

（3）为了消除死锁，可以将变量 t 初始化为 1。

（4）修正后的进度图如下。

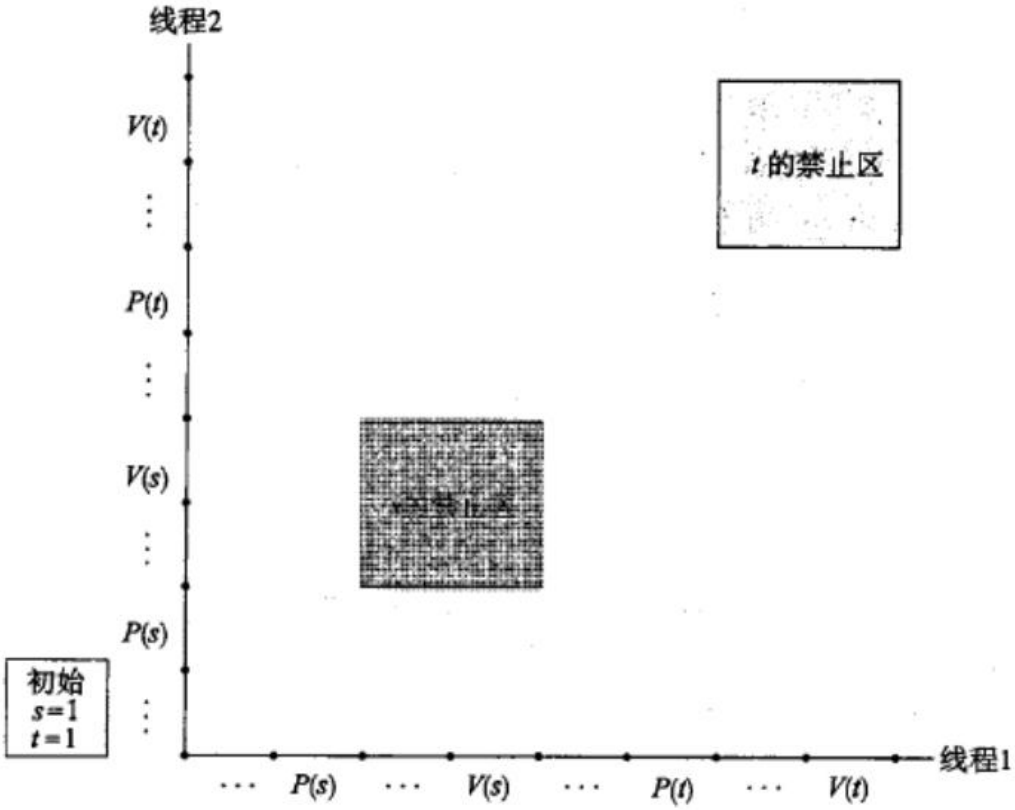


图 12-47　改正后的无死锁的程序的进度图

优化程序性能

原 优化程序性能—《深入理解计算机系统》

2015年10月16日 20:41:39　steelhe　阅读数：790

第一部分：基本策略

1）高级设计：适当的算法和数据结构

2）基本编码原则：使编译器产生高效的代码，理解 编译器 的能力和局限性，消除不必要的内容

·消除连续的函数调用

·消除不必要的存储器引用，要考虑是否为 同一地址

以上两点，也是妨碍编译器优化的主要因素，编译器很难判断以进行优化。

3）低级优化：将一个任务分成多个部分，利用多核和多处理器的并行计算；了解计算机的时序特性，为实现指令集并行，降低不同部分之间的数据相关；利用图形数据流和确认关键路径来确定一个循环需要的时间下界

·循环展开，代码移动（将循环中不会变的计算部分，移到循环外边，如 for(int i=0;i<strlen(s);i++)中的strlen）

·多个累计变量和重新结合（如数据结合，整形编译器可以自动重新结合，浮点数因为精度的原因，不会）等

·用功能的风格重写条件操作，用条件传递代替条件控制，减小 预测惩罚

优化程序性能的几点：

1）高级设计。选择恰当的数据结构和算法;

2）基本编码原则：消除连续的函数调用;消除不必要的存储器引用;

3）低级优化。循环展开;提高并行性，如多个累积变量，重新结合技术，条件传送代替条件选择。

版本2：消除循环的低效率

并非不使用循环，而是将循环中每次都需要重新计算但实际并不会发生改变的量用常量代替。本例中，这个量是向量v的长度。类似的还有使用strlen(s)求得的字符串长度。

显然，如果这个量在每次循环时都会被改变，是不适用的。

复制代码

```
void combine2(vec_ptr v, data_t *dest)
{
    long int i;
    long int length = vec_length(v);
    *dest = IDENT;
    for (i = 0; i < length; i++) {
        data_t val;
        get_vec_element(v, i, &val);
        *dest = *dest OP val;
    }
}
```

复制代码

版本3：减少过程调用

也即减少函数调用。通过对汇编代码的学习，可以知道函数调用是需要一些额外的开销的，包括建栈、传参，以及返回。通过把访问元素的函数直接扩展到代码中可以避免这个开销。但是这样做牺牲了代码的模块性和抽象性，对这种改变编写文档是一种折衷的补救措施。

版本4：消除不必要的存储器引用

在对于vec_rec结构操作时，其中间结果*dest是存放在存储器中的，每次取取值和更新都需要对存储器进行load或store， 要慢于对寄存器的操作。如果使用寄存器来保存中间结果，可以减少这个开销。

这个中间结果，可以通过register显式声明为寄存器变量。不过下面的代码经过汇编后可以发现acc是存放在寄存器中的，不必显示地声明。

然而，中间变量并非越多越好。原书5.11.1节展示了一个情形，如果同时使用的中间变量数过多，会出现"寄存器溢出"现象，部分中间变量仍然需要通过存储器保存。同理，多于机器支持能力的register声明的变量并不一定全部使用了寄存器来保存。

版本5：循环展开

这个优化措施利用了对CPU数据流的知识，比汇编代码更接近机器底层。简单地说是利用了CPU的并行性，将数据分成不相关的部分并行地处理。版本5~7的更多细节和原理可以参考原书。类似的原理在练习题5.5和5.6中展示了为什么Horner法比一般的多项式求值的运算次数少，反而更慢的原因。

版本6与版本7：提高并行性

和版本5的思想类似，但由于并行化更高，性能更好一些，充分利用了向量中各个元素的不相关性。

补充说明

这个示例中没有提到的改进方法还有：书写适合条件传送实现的代码。下面是原书的两段用于对比的代码，后者更适合条件传送实现。

硬件优化趋势

六、总结

多核处理器思路的出现源自CPU主频和功率等的物理极限。通过阿姆达尔定律可以十分清楚地知道多核处理器的性能受到那些因素的限制，即串行执行比例和交互开销。所以多核处理器的发展趋势是合理划分任务、减少核间通信以及加强程序的并行性。这几个方面都已经有一些成果，但总体上还处于探索之中。

二、优化程序性能	
大纲内容	书上内容
优化编译器的能力和局限性	编译器能完成基本的优化，但仍然是有限的。限制编译器只进行安全的优化，保证优化后的程序和未优化的版本有着一样的行为。也意味着对编码的要求更高。

表示程序性能	CPE每元素的周期数，越小越好。集中精力降低CPE
特定体系结构或应用特性的性能优化	**高级设计**：选择恰当的数据结构和算法； **基本编码原则**： • 消除连续的函数调用；（可能时，将计算移到循环外） • 消除不必要的内存引用；（引入中间变量保存结果） **低级优化**： • 展开循环，降低开销； • 提高指令级并行； • 用功能性风格重写条件操作，使得编译采用条件数据传送
限制因素	寄存器溢出； 分支预测和预测错误惩罚；
确认和消除性能瓶颈	程序剖析； 使用剖析程序来指导优化； 优化通用原则：Amdahl定律；