一、名词解释（6题，每题5分）

1、**ISA(**Instruction Set Architecture**)**

指令集体系结构，是在最底层把硬件结构抽象出来供软件编程控制的，指令级解决了最基本的软件兼容性问题。

**2**、**多功能流水线**

流水线的各段可以进行不同的连接，以实现不同的功能。

**3**、资源冲突

因功能部件资源满足不了指令重叠执行的要求时产生的冲突。

**4**、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。

**5**、控制冲突

由于分支指令或跳转指令的执行导致流水线需要确定下一条指令的地址而引起的冲突，也称为分支冲突。

**6**、**Amdahl**定律

当对一个系统中的某个部件进行改进后，所能获得的整个系统性能的提高，受限于该部件的执行时间占总执行时间的百分比。

**7**、加速比，吞吐率，效率

加速比：并行系统与单处理器系统执行同一任务的时间比。

吞吐率：单位时间内处理任务的数量。

效率：实际加速比与理想加速比的比值，反映了系统资源利用率。

**8**、直接映像

一种缓存映射技术，每个主存块固定映射到一个特定的缓存块中，利用主存地址的某些位来决定缓存位置。

全相联、直接映像、组相联 (Fully Associative, Direct Mapped, Set Associative)  
1. \*\*全相联\*\*：缓存块可以映射到任意缓存位置。  
2. \*\*直接映像\*\*：缓存块映射到固定位置。  
3. \*\*组相联\*\*：缓存被划分为多个组，每组内全相联。

**9**、静态和动态流水线

静态流水线：流水线各级的任务在设计时已固定。

动态流水线：通过硬件和软件控制动态调度任务，能更好地适应变化。

**静态流水线：指在同一时间内，多功能流水线中的各段只能按同一种功能的连接方式工作的流水线。当流水线要切换到另一种功能时，必须等前面的任务都流出流水线之后，才能改变连接。**

**动态流水线：指在同一时间内，多功能流水线中的各段可以按照不同的方式连接，同时执行多种功能的流水线。它允许在某些段正在实现某种运算时，另一些段却在实现另一种运算。**

**10**、分支历史表BHT（BHT, Branch History Table）

一种用于分支预测的表格，记录分支指令的历史行为，以提高预测准确性。也被称为分支预测缓冲器，用于记录分支指令最近一次或几次的执行情况。

11、寄存器更名技术

通过为寄存器分配新名字，消除写后读（WAR）和写后写（WAW）数据冲突，提高指令级并行度。**当一条指令写一个结果寄存器时不直接写到这个结果寄存器，而是先写到一个中间寄存器过渡一下，当这条指令提交的时候再写到结果寄存器中，消除指令之间的寄存器读后写相关（WAR），和写后写相关（WAW）。**

**12**、定向技术

在某条指令产生计算结果之前，其他指令并不立即需要计算结果，可以将该结果从其产生的地方直接送到需要它的地方，避免停顿。

**13**、**ROB**技术（**re-order-buffer** ）

重新排序缓冲器，用于乱序执行的处理器中，确保指令按程序顺序提交，解决数据相关性和控制依赖性问题。**前瞻执行：解决控制相关的方法，它对分支指令的结果进行猜测，然后按这个猜测结果继续取指、流出和执行后续的指令。只是指令执行的结果不是写回到寄存器或存储器，而是放到一个称为ROB的缓冲器中。等到相应的指令得到“确认”(即确实是应该执行的)后，才将结果写入寄存器或存储器。**

**14**、BTB分支目标缓冲器（Branch Target Buffer）

**将分支成功的分支指令的地址和它的分支目标地址都放到一个缓冲区中保存起来，缓冲区以分支指令的地址作为标识。**

**15**、超标量机

**一种多指令流出技术。它在每个时钟周期流出的指令条数不固定，依代码的具体情况而定，但有个上限。**

**16**、向量机

**在流水线处理机中，设置向量数据表示和相应的向量指令，称为向量处理机。**

**17**、时间局部性和空间局部性

**时间局部性是指程序即将用到的信息很可能就是目前正在使用的信息；**

**空间局部性是指程序即将用到的信息很可能与目前正在使用的信息在空间上相邻或临近。**

**18**、时间重叠、资源重复、资源共享

时间重叠：指多个操作在时间上重叠执行，提高并行度。

资源重复：指为提高性能而复制多个资源单元。

资源共享：指多个操作共享同一个资源单元。

**19**、平均访存时间

**CPU 访问存储器的平均时间，它反映了存储器的性能和速度。**

**20**、强制失效、容量失效、冲突失效

**强制性失效：当第一次访问一个块时,该块不在Cache中,需要从下一级存储器中调入Cache,这就是强制性失效。**

**容量失效：如果程序在执行时，所需要的块不能全部调入Cache中，则当某些块被替换后又重新被访问，就会产生失效，这种失效就称作容量失效。**

**冲突失效：在组相联或直接映象Cache中，若太多的块映象到同一组（块）中，则会出现该组中某个块被别的块替换（即使别的组或块有空闲位置)，然后又被重新访问的情况。**

**21**、全相联、直接映像、组相联

**全相联映象：主存中的任一块可以被放置到Cache中任意一个地方；**

**直接映象：主存中的每一块只能被放置到Cache中唯一的一个地方；**

**组相联映象：主存中的每一块可以放置到Cache中唯一的一组中任何一个地方(Cache分成若干组,每组由若干块构成)。**

22、‘牺牲’CACHE、踪迹CACHE

‘牺牲’CACHE：用于缓存因冲突失效而被替换的数据块，减少缓存失效。

踪迹CACHE：缓存一段指令序列，提高指令取出的效率。

23、TLB（Translation Lookaside Buffer）

地址转换后备缓冲器，用于加速虚拟地址到物理地址的转换，减小地址转换延迟。**一个专用高速存储器，用于存放近期经常使用的页表项，其内容是页表部分内容的一个副本。**

24、平均CPI （Average Cycles Per Instruction）

每条指令执行的平均时钟周期数，是衡量处理器性能的重要指标。**平均指令周期数，表示执行某个程序的指令平均需要的时钟周期数，它反映了CPU的执行效率。**

25、SIMD和MIMD

SIMD：单指令多数据，单条指令对多组数据进行并行处理。

MIMD：多指令多数据，不同处理器执行不同指令流，对不同数据进行处理。

26、乱序执行顺序确认（Out-of-order Execution and In-order Commit）

处理器可以乱序执行指令以提高性能，但必须按程序顺序提交结果，以确保程序正确性。

27、非阻塞CACHE技术

允许处理器在等待数据缓存访问时继续执行其他指令，提高缓存的利用率和处理器性能。非阻塞缓存技术允许在处理一个缓存缺失的同时继续处理其他缓存请求的技术。非阻塞缓存技术允许在处理缓存失效的同时，CPU仍然可以进行其他命中访问，从而极大提高CPU性能。

28、循环展开

优化编译技术，通过展开循环减少分支指令和循环开销，提高指令并行度。

29、RISC和CISC  (Reduced Instruction Set Computing and Complex Instruction Set Computing)

1. \*\*RISC\*\*：精简指令集计算，指令简单且执行速度快。  
2. \*\*CISC\*\*：复杂指令集计算，指令复杂且功能强大。

30、数组合并技术、内外循环交换技术、循环融合技术

1. \*\*数组合并技术\*\*：通过将独立数组合并为复合数组，提高空间局部性，减少缓存失效次数。  
2. \*\*内外循环交换技术\*\*：在含有嵌套循环的程序中，按照存储器中数据的存储顺序重新排列访问顺序，提高空间局部性，减少缓存失效。  
3. \*\*循环融合技术\*\*：将独立的程序段融合为单一循环，改进时间局部性，减少缓存失效次数。

二、简答题（4题，每题8分）

0、计算机体系结构，从顶层上来讲为什么会有这门课

计算系统结构的学习目标只有一个，就是在各种部件上尽可能提高性能，进而提高整机性能。

1、对相关性的处理通常有两种方式有两种方式：静态方式和动态；说明这两种方式，并分别举出2种实现方式。

**静态方式是指由编译器在编译时对指令进行调度和优化，以减少或消除相关性造成的停顿。动态方式是指由硬件在运行时对指令进行调度和重排序，以允许乱序执行和流出。**

**静态方式的实现方法有：**

1. **循环展开：将循环体中的多条指令复制多次，以减少循环次数和分支开销，同时增加指令间的并行度。**
2. **寄存器换名：将指令中使用相同寄存器的操作数替换为不同的寄存器，以消除名相关。**
3. **指令调度：将相关指令之间的距离加大到不小于指令执行延迟的时钟数，以避免数据相关造成的停顿。**

**动态方式的实现方法有：**

1. **记分牌技术：使用一个记分牌来记录指令的状态和功能部件的状态，以判断指令何时可以流出、读操作数、执行和写结果。**
2. **Tomasulo算法：使用寄存器重命名技术和保留站技术来解决数据相关和名相关，使用公共数据总线技术来实现操作数的旁路传送。**

2、Tomasulo采取了什么方法避免名相关和数据相关。

通过寄存器换名和保留站消除了WAW和WAR冲突

记录和检测指令相关，操作数一旦就绪就立即执行，把发生RAW冲突的可能性减少到最小；

通过寄存器换名来消除WAR冲突和WAW冲突。

**寄存器换名**，可以避免名相关。

**公共数据总线**，解决数据相关。指令之间不需要直接交互，而是通过CDB来传递数据，这种机制可以避免数据冲突，因为指令可以通过总线 接受到所需的数据而不必等待之前的指令完成。

**保留站**：为每个功能部件设置一个或多个缓冲区，用于存储已流出但未执行的指令，从而允许乱序流出和乱序执行。

3、给出一段有相关性的指令，分析相关性、请重新设计指令顺序（编译器方式），消除相关性

add r3,r1,r2

add r4,r3,r2

add r4,r1,r5

r3数据相关，r4名相关

add r3,r1,r2

add r4,r3,r2

add T,r1,r5

**4**、计算机系统结构、计算机组成和计算机实现的概念与关系。

计算机系统结构：计算机系统的软硬件界面，即程序设计者能看到的计算机的基本属性。

计算机组成：计算机系统结构的逻辑实现，包括物理机器级中数据流、控制流组成和逻辑设计等

计算机实现：计算机组成的物理实现，包括处理机、主存等部件的物理结构和连接等。

计算机系统结构：程序设计者所看到的计算机的基本属性，即概念性结构与功能特性。

计算机组成：指的是计算机系统结构的逻辑实现，包含物理机器级中的数据流和控制流的组成以及逻辑设计等。它着眼于物理机器级内各事件的排序方式与控制方式、各部件的功能以及各部件之间的联系。

计算机实现：指的是计算机组成的物理实现,包括处理机和主存等部件的物理结构，器件的集成度和速度,模块、插件、底板的划分与连接,信号传输,电源、冷却及整机装配技术等。它着眼于器件技术和微组装技术。

**5**、论述**RISC**与的**CISC**技术；讨论**RISC**从哪些方面提高了指令的执行效率，并举例说明。

CISC：采用复杂指令集，指令数目多，强调指令的功能性，通过增强指令功能提高性能。

RISC：采用精简指令集，有以下特点

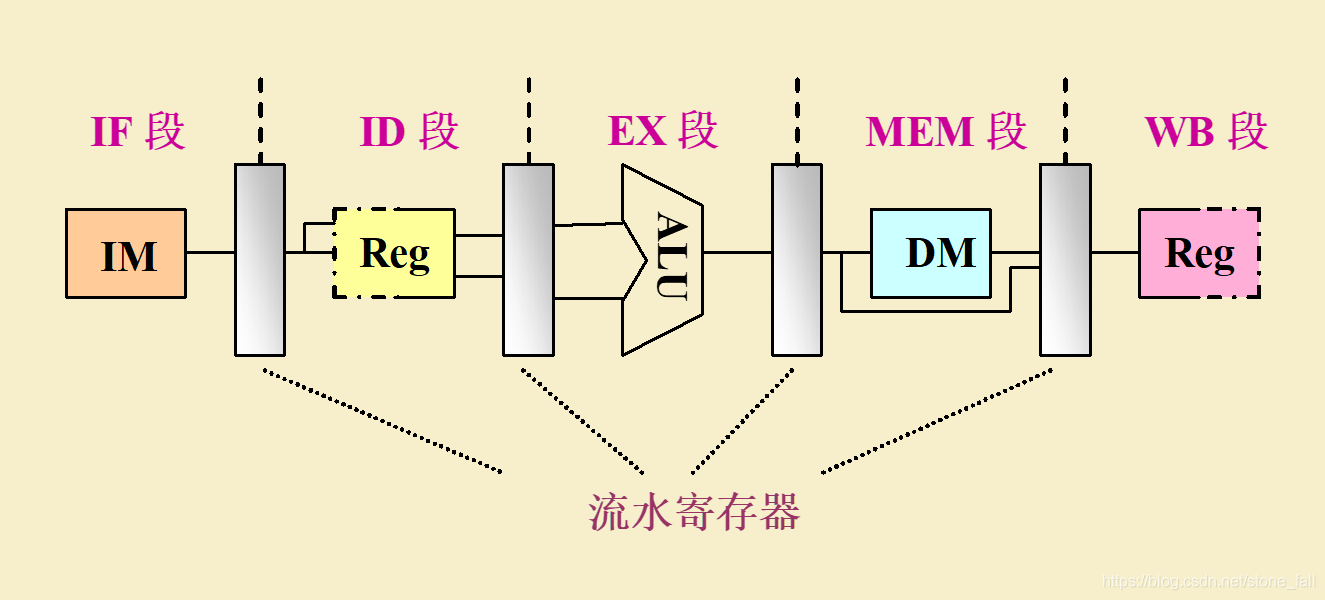
精简的指令集，仅保留使用频率高的指令

大部分指令硬连逻辑直接译码

大量使用通用寄存器

上述方面极大减小了CPI，提高了性能，如RISC1性能比当时先进商业处理器快了3倍

**6**、简要画出经典**5**段流水流水线的数据通路图；说明**load**和**store**在每个周期的表现。



**Load/store：**

**IF：根据从主存取指**

**ID：译码，读寄存器**

**EX：计算访存地址**

**MEM：访存(读/写)**

**WB：读出数据写入寄存器（load）**

**7**、流水线冲突有哪三种？请简述每种流水线冲突。

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和其他会改变PC值的指令所引起的冲突。

**8**、如果某计算机系统有**3** 个部件可以同时改进，则这**3** 个部件经改进后达到的加速比分别为：***S*1=30**, ***S*2=20**, ***S*3=10**。

如果部件**1** 和部件**2** 改进前的执行时间占整个系统执行时间的比例都为**30%**，那么，部件**3** 改进前的执行时间占整个系统执行时间的比例为多少，才能使**3** 个部件都改进后的整个系统的加速比***Sn*** 达到**10**？

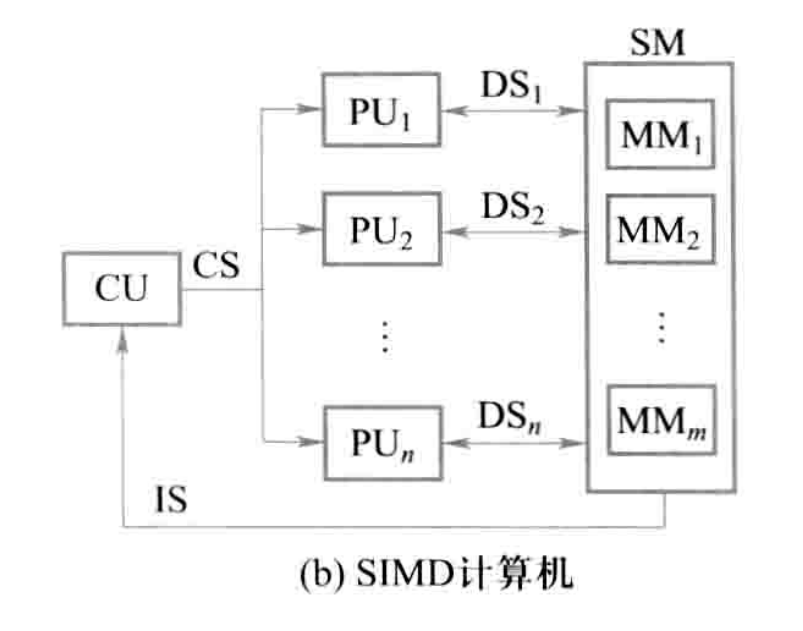
如果部件**1** 和部件**2** 改进前的执行时间占整个系统执行时间的比例都为**30%**，那么，部件**3** 改进前的执行时间占整个系统执行时间的比例为多少，才能使**3** 个部件都改进后的整个系统的加速比***Sn*** 达到**10**？

**10=1/(0.3\*(1/30)+0.3\*(1/20)+k\*(1/10)+(1-0.6-k))**

**k=0.361**

1. **GPU**采用了哪种处理器设计方式作为原型，请简述并画出这种处理器的体系结构原理图。

SIMD 单指令多线程。



**10**、名相关和数据相关会产生写读冲突、读写冲突、写写冲突。简述这三种冲突，并举例说明是如何造成了。

1) RAW (写读冲突) :

原程序要求对同一单元进行先写后读的操作，可能因为非按序执行成为先读后

写,造成出错。（或者DLX经典五级流水线RAW会导致后一条指令读到的寄存器前一条指令还未写入）。

2) WAR (读写冲突)

原程序要求对同一单元进行先读后写的操作，可能因为非按序执行成为先写

后读，造成出错。

3) WAW (写写冲突)

原程序中如果两条指令都要对同一单元进行写数操作，可能因为非按序执行

的原因，改变了两条指令写入的次序。

**11**、层次化存储系统存在的理论依据是什么？简要阐述这个依据中的原理。

**程序的局部性原理。程序对存储器的访问是相对集中的。空间：之后访问的数据与当前访问的数据相邻/相近。时间：正在被访问的数据可能也是之后会访问的。**

**12**、写出平均访存时间的公式，从公式的三个变量出发，分别举出一个优化（减少）平均访存时间的技术方案。

平均访存时间=命中时间+失效率\*失效开销

减少命中时间：设计容量小、结构简单的Cache，采用虚拟Cache。

减少失效率：提高Cache的容量和相联度。

减少失效开销：采用多级Cache，采用非阻塞Cache技术

**13**、**CACHE**的地址映像规则有三种：全相联、直接映像与组相联。阐述这三种规则，并用图示法说明三种规则的有缺点。

**全相联：主存中任意一块可映射到cache任意一块。块冲突率低，空间利用率高/复杂度和成本高**

**直接印象：主存的一块唯一映射到cache一块。硬件简单，速度快/块冲突率高**

**组相联：主存一块映射到唯一一组的任意块。冲突率比直接映射大大降低，实现比全相联容易，是折中**

①全相联映象方式

主存储器中的任意一块可以映象到Cache中的任意一块上。.

优点:块冲突小，Cache的利用率高。

缺点: 硬件开销大，因为必须使用完整的地址比较来查找数据块

②直接映象方式

主存中的每一个块只能被放置到Cache中唯一的一个位置。

优点: 实现简单，因为只需要对地址进行简单的运算即可确定缓存行。

缺点:块的冲突率较高。

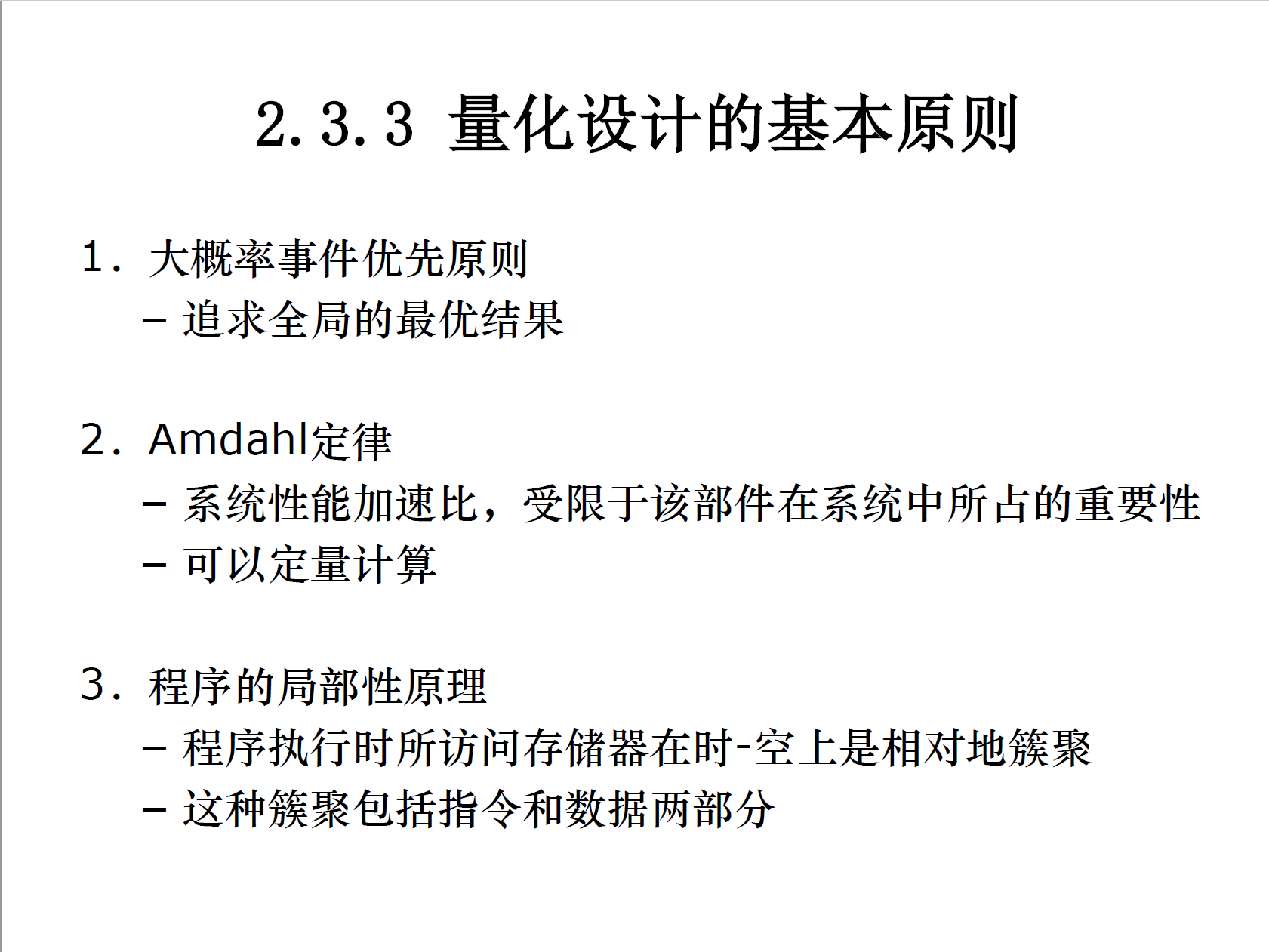
③Cache被等分为若干组，每组由若干个块组成。主存中的每一块可以被放置到Cache中唯一的一个组中的任何一个位置。

组相联映象方式

优点: 块分配更灵活，缓存利用率高。映射速度较快，因为只需比较组号和索引值即可确定缓存行。

缺点：硬件复杂度稍高；可能存在冲突

1. 计算机系统设计中经常使用的设计原则和定量原理是什么？请说出他们的含义？



1. 在降低Cache失效率的方法中，对于给定的Cache容量，当块大小增加时，失效率开始是下降，后来反而上升了。解释Cache失效率为什么出现这样的变化?

Cache失效率的变化与块大小的增加有直接关系，其原因如下：

1. \*\*初期下降\*\*：当块大小增加时，每个块能存储更多的相邻数据。这有助于利用程序的空间局部性特性，使得在下一次访问相邻数据时，能命中Cache，从而降低失效率。

2. \*\*随后上升\*\*：当块大小继续增加时，会导致以下问题：

- \*\*更少的块数量\*\*：固定容量的Cache在块大小增加后，能存储的块数量减少。这会增加冲突失效率（conflict misses），因为多个数据块可能映射到相同的Cache行。

- \*\*空间浪费\*\*：如果程序访问的局部性较差，大块的数据可能会包含大量无用数据，增加无效Cache访问，从而增加失效率。

Cache容量一定，块大小增加，增强了空间局部性，减少了强制不命中；但减少了块数量，增加了冲突不命中。在块比较小的时候，第一种作用超过了第二种作用；当等到块变大的时候，第二种作用又超过了第一种作用。

1. 写出三种降低CACHE不命中率的方法并举例说明；写出三种减少CACHE命中时间的方法并举例说明；写出三种降低失效开销的方法并举例说明。

\*\*降低CACHE不命中率的方法\*\*：

1. \*\*增加Cache容量\*\*：增大Cache的总大小，可以容纳更多数据块，减少不命中率。

- 示例：从64KB增加到128KB的Cache。

2. \*\*提高关联度（Associativity）\*\*：增加相联度，可以减少冲突失效率。

- 示例：将直接映射Cache改为4路组相联Cache。

3. \*\*更好的替换策略\*\*：采用先进的替换算法（如LRU、LFU）来减少不命中率。

- 示例：使用LRU（Least Recently Used）替换策略代替随机替换策略。

\*\*减少CACHE命中时间的方法\*\*：

1. \*\*降低Cache层级\*\*：减少访问更低层Cache的时间。

- 示例：将L2 Cache从CPU核心中移至更近的L1 Cache。

2. \*\*优化Cache访问路径\*\*：简化Cache控制器的路径，减少访问延迟。

- 示例：减少Cache控制器中地址译码的延迟。

3. \*\*使用更快的缓存技术\*\*：采用更高速的存储技术。

- 示例：从SRAM（静态随机存取存储器）切换到更高性能的SRAM技术。

\*\*降低失效开销的方法\*\*：

1. \*\*写策略优化\*\*：采用Write-back而非Write-through来减少写失效时的数据传输开销。

- 示例：使用Write-back策略，将数据写回主存仅在块被替换时进行。

2. \*\*预取机制（Prefetching）\*\*：提前将可能访问的数据块加载到Cache中，减少失效时的数据加载时间。

- 示例：根据访问模式进行指令和数据预取。

3. \*\*减少缓存抖动\*\*：通过减少多个任务在同一块Cache上的竞争，降低失效开销。

- 示例：采用软件或硬件方法减少任务在同一Cache行的竞争。

1. 流水线的额外开销包括哪两种？讨论其对流水线性能的影响和解决方法。

1. \*\*流水线寄存器开销\*\*：每个阶段需要增加寄存器来保存中间结果，这些寄存器会增加硬件成本和延迟。

- \*\*性能影响\*\*：寄存器增加了每个阶段的延迟，从而影响流水线的最大工作频率。

- \*\*解决方法\*\*：优化寄存器设计，使用更高速的寄存器技术，并优化流水线深度以平衡性能和开销。

2. \*\*流水线阻塞（Stall）开销\*\*：当某个阶段的操作需要等待前一阶段完成时，会导致流水线暂停，产生阻塞开销。

- \*\*性能影响\*\*：阻塞会导致吞吐量下降，流水线不能达到理论上的最大性能。

- \*\*解决方法\*\*：采用数据前递（forwarding）和分支预测等技术减少阻塞频率，提高流水线的实际性能。

1. 解决流水线的瓶颈有哪两种常见方法，举例说明并比较其效果。

1. \*\*增加流水线深度\*\*：

- \*\*方法\*\*：将一个阶段分成更小的子阶段，使每个子阶段的工作量更少，从而提高流水线频率。

- \*\*效果\*\*：可以提高整体频率和吞吐量，但可能增加流水线寄存器开销和复杂性。

2. \*\*超标量（Superscalar）执行\*\*：

- \*\*方法\*\*：在同一周期内发射多个指令，通过并行执行多个指令提高指令吞吐量。

- \*\*效果\*\*：可以显著提高执行效率，但需要更复杂的调度和依赖解决机制。

\*\*比较\*\*：

- \*\*流水线深度增加\*\*：适用于提升单线程性能，但在过多阶段时可能遇到寄存器开销和依赖问题。

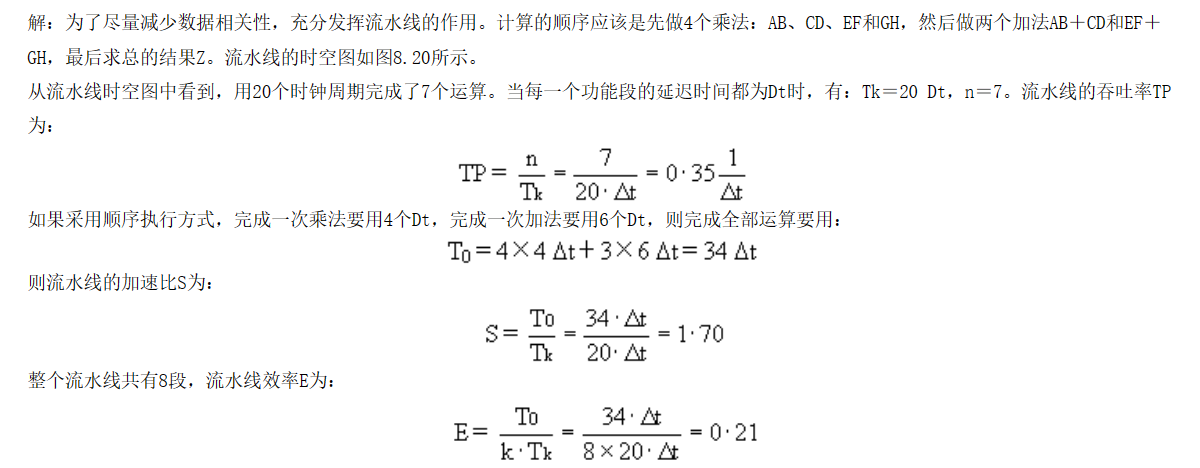
- \*\*超标量执行\*\*：适用于提升整体吞吐量，但实现复杂性和资源需求较高。

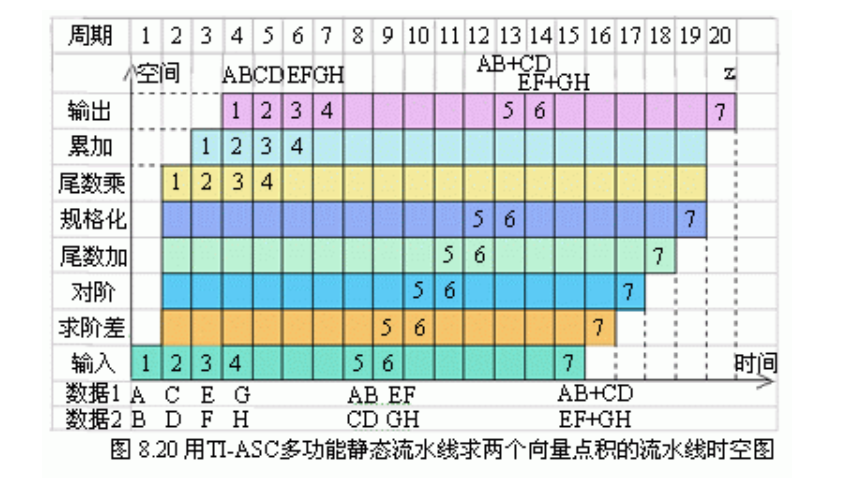
三综合题（**4**题 **38**分，）

**1**、计算机运行以下指令：

线性多功能静态流水线，输入任务是不连续的情况，计算流水线的吞吐率、加速比和效率。 用TI－ASC计算机的多功能静态流水线计算两个向量的点积：Z＝AB＋CD＋EF＋GH







文本

低可信度描述已自动生成

2、某台主频为1000MHz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 | 5 |
| 分支 | 2000 | 2 |

求该计算机的平均CPI、MIPS和程序执行时间(单位：us)。

**CPI：每条指令所耗费的时钟周期**

**MIPS：每秒执行的多少百万条指令数量**

**主频：每秒=1000M个时钟周期 M=106**

**平均CPI = （45000×1 + 75000×2 + 8000×5 + 2000×2）/130000 ≈ 1.838**

**MIPS = f/CPI = 1000/1.838 = 543.933MIPS**

**程序执行时间=（45000×1 + 75000×2 + 8000×5 + 2000×2）/1000M = 239us**

**3**、一条有**4**个流水段的非线性流水线，每一段的延迟时间相等，预约表如下：



（**1**）写出禁止向量和冲突向量

（**2**）画出调度状态图

（**3**）求出最大吞吐量

（**4**）按最优调度连续输入**8**个任务，实际吞吐量，加速比和效率各为多少

1. 写出禁止向量和冲突向量

**禁止向量f = { 6 , 4 , 2 }，初始冲突向量101010（1表示冲突，禁止向量中的数）**

**C1 = SHR(1)（C0）∨C0 =（ 010101 ）∨（101010）=（ 111111 ）**

**C2 = SHR(3)（C0）∨C0 =000101∨101010＝101111**

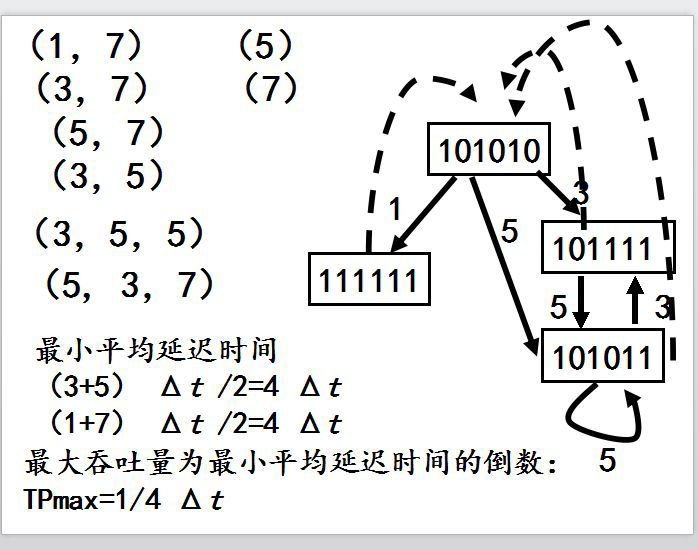
**C3 = SHR(5)（C0）∨C0 =000001∨101010＝101011**

**C4 = SHR(5)（C2）∨C0 =000001∨101010＝101011=c3**

**C5 = SHR(3)（C3）∨C0 =000101∨101010＝101111= c2**

**C6 = SHR(5)（C3）∨C0 =000001∨101010＝101011=c3**

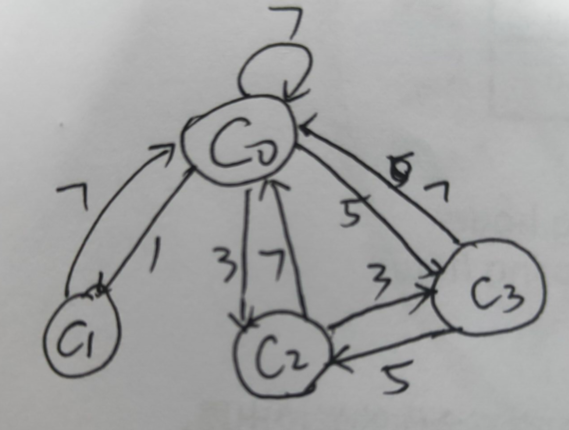
1. 画出调度状态图

****

1. 禁止向量F=(2, 4, 6)。

冲突向量C0=(101010), C1=SHR1(C0)∨C0=(111111), C2=SHR3(C0)∨C0=(101111), C3=SHR5(C0)∨C0=(101011), SHR5(C2)∨C0=(101011)=C3, SHR3(C3)∨C0=(101111)=C2, SHR5(C3)∨C0=(101011)=C3



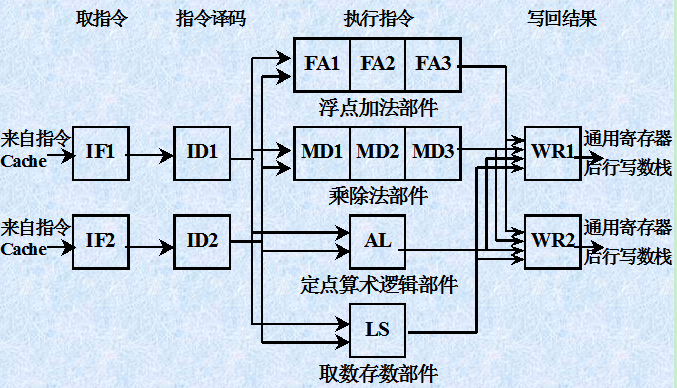


1. 调度策略，平均间隔拍数：
   1. (1, 7), 4
   2. (3, 7), 5
   3. (5, 7), 6
   4. (3, 5, 7), 5
   5. (3, 5), 4
   6. (5, 3, 7), 5
   7. (5, 3), 4
   8. (7), 7

最佳调度策略为(1, 7)或(3, 5)或(5, 3)，最小平均启动距离为4。最大吞吐量为1/(4\*Δt)

1. 以调度策略(1, 7)为例，因为有偶数个任务，所以实际吞吐率为8/(8\*8/2) \*Δt =1/(4\*Δt)

**4**、超标量机的相关性问题以及调度



计算机运行以下指令：

I1：LOAD R1, A ；R1←(A)

I2：FADD R2, R1 ；R2←(R2)＋(R1)

I3：FMUL R3, R4 ；R3←(R3)×(R4)

I4：FADD R4, R5 ；R4←(R4)＋(R5)

I5：DEC R6 ；R6←(R6)－1

I6：FMUL R6, R7 ；R6←(R6)×(R7)

（1）请列出程序代码中可能出现的数据相关及相关类型。

（2）当程序通过下图的双发射超标量机时，请采用顺序发射乱序完成的方式画出指令流水时空图。

(流水线没有使用定向技术。)

1. 数据相关：I2用的R1是I1读取的R1

反相关：I4写入的R4寄存器是I3读取的寄存器

输出相关：I6写入的R6寄存器是I5写入的寄存器

表格

描述已自动生成

**5**举三个例子说明系统中采用软件来提高性能的方法和效果。

编译器优化：编译器可以对源代码进行优化，以减少程序运行时的开销。例如，编译器可以选择更高效的算法、使用更少的寄存器或减少内存访问等方式来优化程序。

并行计算：通过在程序中使用并发技术，可以利用计算机系统中的多个处理器来同时执行不同的任务，从而提高程序的执行效率。例如，OpenMP和MPI是两种常用的并行计算技术。

虚拟化技术：虚拟化技术可以将计算机系统的物理资源划分为多个虚拟资源，并允许多个应用程序在同一台物理机上并发运行。这样可以减少硬件资源的浪费，提高系统的利用率和性能。

**6**、**CACHE**映像算法

有一个Cache存储器，主存有8块(0-7)，Cache有4块(0-3)，采用组相联映像，组内块数为2块。采用LRU（近期最久未使用）替换算法。（12分，（1）题4分，（2）题8分）

(1)指出主存各块与Cache各块之间的映像关系。

(2)某程序运行过程中，访存的主存块地址流为：

2， 3， 4， 1， 0， 7， 5， 3， 6， 1， 5， 2， 3， 7， 1

说明该程序访存对Cache的块位置的使用情况，指出发生块失效且块争用的时刻，计算Cache命中率

1. 主存块0,2,4,6映射到Cache的块0,1中，主存块1,3,5,7映射到Cache的块2,3中

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Cache0 | Cache1 | Cache2 | Cache3 | 事件 | 命中率 |
| 2 | 2 |  |  |  | 失效 | 0 |
| 3 | 2 |  | 3 |  | 失效 | 0 |
| 4 | 2 | 4 | 3 |  | 失效 | 0 |
| 1 | 2 | 4 | 3 | 1 | 失效 | 0 |
| 0 | 0 | 4 | 3 | 1 | 失效且争用 | 0 |
| 7 | 0 | 4 | 7 | 1 | 失效且争用 | 0 |
| 5 | 0 | 4 | 7 | 5 | 失效且争用 | 0 |
| 3 | 0 | 4 | 3 | 5 | 失效且争用 | 0 |
| 6 | 0 | 6 | 3 | 5 | 失效且争用 | 0 |
| 1 | 0 | 6 | 3 | 1 | 失效且争用 | 0 |
| 5 | 0 | 6 | 5 | 1 | 失效且争用 | 0 |
| 2 | 2 | 6 | 5 | 1 | 失效且争用 | 0 |
| 3 | 2 | 6 | 5 | 3 | 失效且争用 | 0 |
| 7 | 2 | 6 | 7 | 3 | 失效且争用 | 0 |
| 1 | 2 | 6 | 7 | 1 | 失效且争用 | 0 |

1. 在一个4段的流水线处理机上需经7拍才能完成一个任务，按给出的预约表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 时间  功能段 | T1 | T2 | T3 | T4 | T5 | T6 |
| S1 | √ |  |  |  |  | √ |
| S2 |  | √ |  |  |  |  |
| S3 |  |  | √ | √ |  |  |
| S4 |  |  |  |  | √ |  |

(1) 写出延迟禁止表F和冲突向量C。　　(2) 画出流水线调度状态图。　　(3) 求出最小平均延迟的调度方案及最大吞吐率。　　(4) 求按此流水调度方案输入8个任务的实际吞吐率。

(1) 禁止向量F=(1, 5)

冲突向量C0=(010001)

SHR2(C0)∨C0=(010101)=C1,

SHR3(C0)∨C0=(010011)=C2,

SHR4(C0) ∨C0=(010001)=C0,

SHR6(C0)=C0,

SHR2(C1)∨C0=(010101)=C1,

SHR4(C1)∨C0=(010001)=C0

SHR6(C1)∨C0=C0

SHR3(C2)∨C0=(010011)=C2

SHR4(C2)∨C0=C0

SHR6(C2)∨C0=C0

(3)

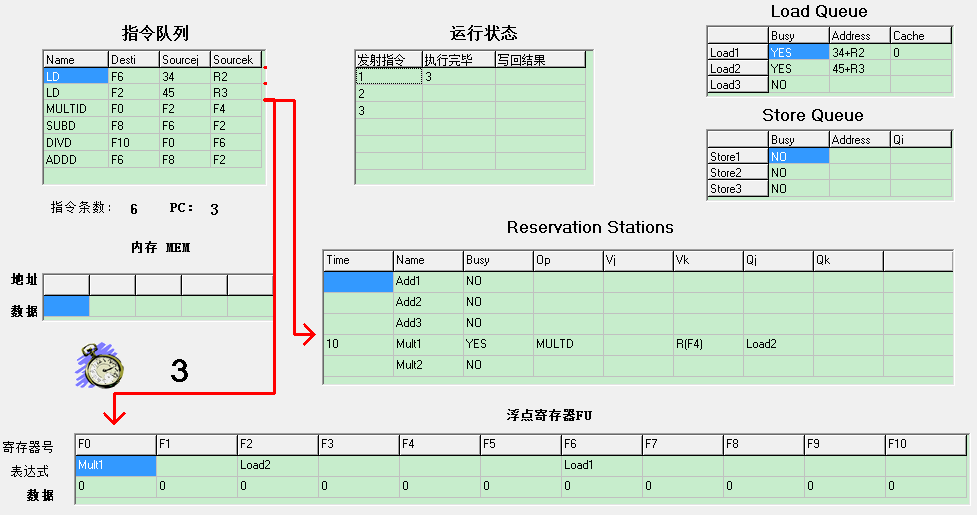
调度方案和平均间隔拍数：

1. (4), 4
2. (6), 6
3. (2, 6), 4
4. (2, 2), 2
5. (2, 4), 3
6. (2, 6), 4
7. (3, 3), 3
8. (3, 4), 3.5
9. (3, 6), 4.5

最小平均延迟的调度方案是(2, 2)。最大吞吐率是1/(2\*Δt)

(4)

实际吞吐率为8/((6+2\*7) \*Δt)=2/(5\*Δt)

8、tomasulo算法的第3个时钟周期的指令状态，保留站状态，和寄存器结果状态如下图所示；

（其中Op表示现在保留站中正在工作的指令,Vj，Vk表示已经准备好的操作数，Qj,Qk表示已发射但未准备好的操作数）。已知load 执行延时2个cycles，add（sub）执行延时2个cycles，mul 执行延时10个cycles，div 执行延时40个cycles。

要求：

写出tomasulo算法的核心思想。

写出第5个时钟周期的指令运行状态，保留站状态，和寄存器结果状态，并说明原因。

①纪录和检测指令相关，操作数一旦就绪就立即执行，把发生RAW冲突的可能性减小到最少；

②通过寄存器换名来消除WAR冲入和WAW冲突。寄存器换名是通过保留站来实现，它保存的等待流出和正在流出指令所需要的操作数。

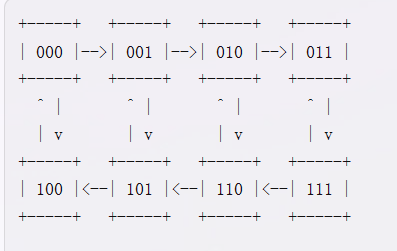
基本思想：只要操作数有效，就将其取到保留站，避免指令流出时才到寄存器中取数据，这就使得即将执行的指令从相应的保留站中取得操作数，而不是从寄存器中。指令的执行结果也是直接送到等待数据的其他保留站中去。因而，对于连续的寄存器写，只有最后一个才真正更新寄存器中的内容。一条指令流出时，存放操作数的寄存器名被换成为对应于该寄存。

9、在流水线分支预测时经常采用历史分支表的方式，采用两位编码历史分支表的状态图如下：当预测状态转换，如从01分支预测不成功转换成11分支预测成功后，需要经过连续两次不成功才能回到不成功预测状态。（如下图），反之亦然。



试着设计一种三位编码的历史分支表，实现当分支预测状态转移后，需要经过4次连续成功预测或不成功预测才能实现状态转移。

* 三位编码可以表示八种状态，其中四种为跳转状态，四种为不跳转状态。
* 跳转状态可以用0开头的编码表示，如000、001、010、011；不跳转状态可以用1开头的编码表示，如100、101、110、111。
* 每个状态的第二位和第三位可以表示该状态的强度，即需要多少次连续的成功或不成功的预测才能改变状态。例如，000表示强跳转状态，需要四次连续的不成功的预测才能变成001；001表示较强跳转状态，需要三次连续的不成功的预测才能变成010；以此类推。
* 当预测成功时，如果当前状态是跳转状态，则第二位和第三位都加一，直到达到最大值11；如果当前状态是不跳转状态，则第二位和第三位都减一，直到达到最小值00。
* 当预测不成功时，如果当前状态是跳转状态，则第二位和第三位都减一，直到达到最小值00；如果当前状态是不跳转状态，则第二位和第三位都加一，直到达到最大值11。
* 当第二位和第三位都为00时，表示该状态已经达到最弱的程度，再次预测不成功时，就要改变第一位，即从跳转状态变成不跳转状态，或者从不跳转状态变成跳转状态。
* 这样设计的三位编码历史分支表可以实现当分支预测状态转移后，需要经过四次连续成功预测或不成功预测才能实现状态转移。



10、考虑考虑某两级cache，第一级为L1，第二级为L2，两级cache的全局不命中率分别是5%和2%，假设L2的命中时间是5个时钟周期，L2的不命中开销是100时钟周期，L1的命中时间是1个时钟周期，平均每条指令访存1.4次，不考虑写操作的影响。求：

（1）计算L2的局部不命中率

（2）计算L1的不命中开销是多少个时钟周期

（3）每次访存的平均访存时间是多少个时钟周期

（4）每次访存的平均停顿时间是多少个时钟周期

（5）每条指令的平均停顿时间是多少个时钟周期

