复习提纲

* 考试题型

1. 单选题（单选每小题2分，多选每小题4分，共14分)
2. 名词解释题（本大题共6小题，每小题5分，共30分）
3. 简答题（本大题共4小题，每题6分，共24分 ）
4. 综合题（本大题共4小题，每题8分，共32分）

* 复习内容

**一、名词解释**

1、**ISA**

指令集体系结构(Instruction Set Architecture, ISA)，简称体系结构或系统结构(architecture),它是软件和硬件之间接口的一个完整定义。

1. **MicroArchitecture**

**一个微结构可以看做是ISA指令集体系结构的硬件实现。（且是某种特定ISA的硬件circuitry）**

**3**、资源冲突（第三章）

多条指令进入流水线后，在同一时间征用同一功能部件，从而发生冲突。

1. 数据冲突（第三章）

由于流水线中各指令重叠执行，使得原来对操作数的访问顺序发生变化。从而引起的一种冲突

1. 控制冲突（第三章）

流水线遇到分支指令和其他会改变PC值的指令时所引起的冲突。

通常的解决措施为：

在流水线中尽早判断出分支转移是否成功

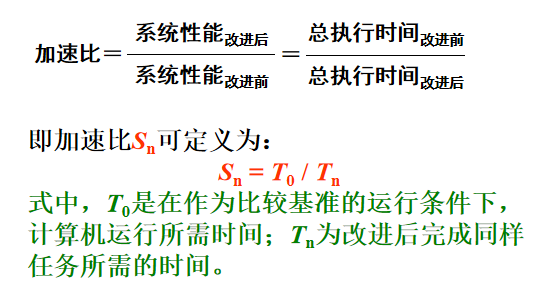
尽早计算出分支目标地址

**6**、**Amdahl**定律（第一章）

加快某部件执行速度所能获得的系统性能加速比，受限于该部件的执行时间占系统中总执行时间的百分比。

1. 加速比（第一章）

阿姆达尔定律定义了由于采用了某种改进方法后系统所能获得的**加速比**大小。



1. 非冯**·**诺依曼计算机

（或许是因为传统的冯诺依曼结构具有局限性

{- 采用存储程序方式，指令和数据不加区别混合存储在同一个存储器中。

- 存储器是按照地址访问的线性编址一维结构，每个单元的位数是固定的

- 指令由操作码和地址组成。

- 通过执行指令直接发出控制信号控制计算机的操作。

- 以运算器为中心，输入输出设备与存储器之间的数据传送都要经ALU

- 数据以二进制表示

}）

指的是脱离了冯诺依曼架构的计算机，例如光子计算机、并行计算机、数据流计算机以及量子计算机等

而非冯诺依曼计算机本质上是采取**串行顺序处理**的工作机制，即即使有关数据已经就绪，必须执行指令序列。而提高该模式处理效率的方式即增强并行处理的能力。

**9**、静态流水线、动态流水线、单功能流水线、多功能流水线（第三章）

静态流水线：当执行某一规定动能的指令全部流出吼，才允许改变部件间连接的流水线；

动态流水线（只能是多功能流水线）：没有静态流水线在时间上的限制，可以在任何时候根据需要改变其连接；

单功能流水线：指一条流水线只能完成一种单一的任务；

多功能流水线：指能够在一个时间段内或不同时间段改变部件之间的连接，从而达到改变其功能的流水线。

1. 锁存技术（第三章）

在流水线的每一个功能部件后面都要有一个**缓冲寄存器，或称为锁存器、闸门寄存器**，它的通是保存本流水段的执行结果。锁存，就是将数据临时存储。使得某部件额值在运算之后可以暂时存储。

**11**、寄存器换名技术

代表性算法：Tomasulu算法

可以消除WAR冲突和WAW冲突

减少冲突的发生

1. 定向技术

类似DMA？ 不借助寄存器直接将数据送到需要的运算器上，

在指令的动态调度中，采用定向传送技术可以在一定程度上解决数据冲突问题

**主要思想：**

发生数据冲突时，后面的指令并不是立即就用到前一条指令的计算结果。如果计算结果能够从其产生的地方直接送到需要它的地方，就可以避免暂停。当定向硬件检测到前面某条指令的结果寄存器就是当前指令的源寄存器时，控制逻辑会将前面那条指令的结果直接从其产生的地方定向到当前指令所需的位置。

1. **ROB**技术（**re-order-buffer** ）

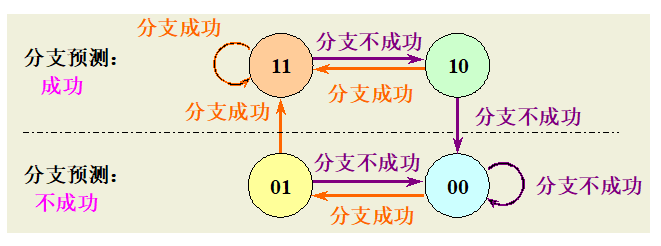
**在基于硬件的前瞻执行中，对分支指令的结果进行猜测，并假设这个猜测结果总是对的，然后按这个猜测结果继续取、流出和执行后续的指令。只是执行指令的结果不是写回到寄存器或存储器，而是放到一个称为ROB的缓冲器中。**

**其作用是等到相应的指令得到 确认（commit）（即确实是应该执行的）之后，才将结果最终写入寄存器或存储器。**

**14**、分支历史表BHT

Branch History Table；是动态分支预测技术中的一个定义，又称为预测分支缓冲器。通常可以用BHT来记录分支指令最近一次或几次的执行情况，并据此进行预测。

一位是最简单的预测表，通常两位即可达到不错的性能，可以提高预测的准确度



1. 超标量机

- 把一个时钟周期内能够同时**发射多条指令**的处理机称为超标量处理机

- 超标量处理机最基本的要求是要有两套或两套以上完整的指令执行部件

- 为了能够在一个时钟周期内同时发射多条指令，超标量处理机必须有两条或两条以上能够同时工作的指令流水线。

可以理解为实际上实现的是空间并行

1. 向量机（第四章）

把向量数据表示与流水线结合其阿里，就构成了向量流水处理机，简称为向量流水机或向量处理机。

- 向量处理机的处理对象是向量元素。

**17**、时间局部性和空间局部性

- 时间局部性

- 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

- 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接近

1. 数组合并技术、内外循环交换技术、循环融合技术（第五章）

在编译器优化阶段，为了降低Cache的不命中率。

考虑到数据对存储位置的限制比指令少，更便于优化。通过对数据的重新组织，使得一块数据被从Cache替换出去之前，能最大限度地被利用（即访问次数最多）

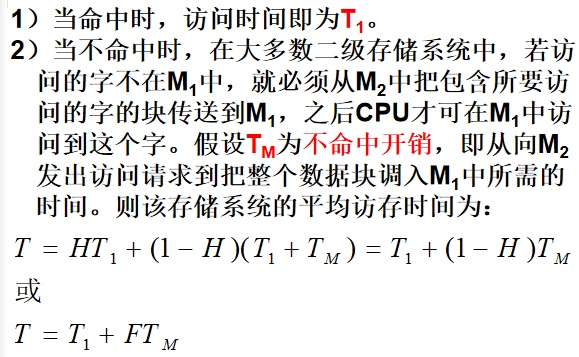
而数组合并，内外循环交换，循环融合技术就是具体的数据重组方式。

1. 平均访存时间（第五章）

这属于存储系统的性能指标（通常来说有三个主要指标，即速度T、容量S、和价格C）

而访问时间T，也被称为平均访存时间或等效访问时间等

一般分两种情况来考虑CPU的一次访存



1. 强制失效、容量失效、冲突失效（第五章）

为了最终降低Cache的不命中率，要先明确三种类型的不命中（3C）

- 强制性不命中 Compulsory miss

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（也可称为**冷启动失效**或**首次访问失效**）

- 容量不命中 Capacity miss

如果**程序执行时所需要的的块不能全部调入Cache中**，则当某些块被替换后，若又重新被访问就会发生失效。这种失败称为容量失效。

- 冲突不命中 Conflict miss

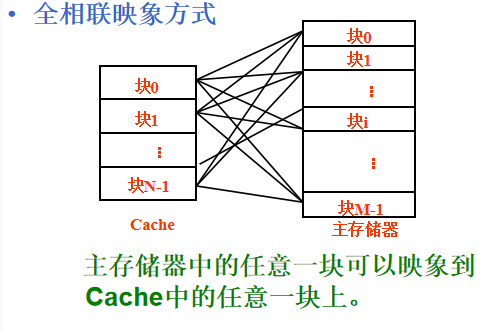
在组相连或直接映像Cache中，若太多的块映像到统一组中，则会出现该组中某个块被别的块替换，然后又被重新访问的情况，发生了冲突失效。

1. 全相联映像、直接映像、组相联映像（第五章）

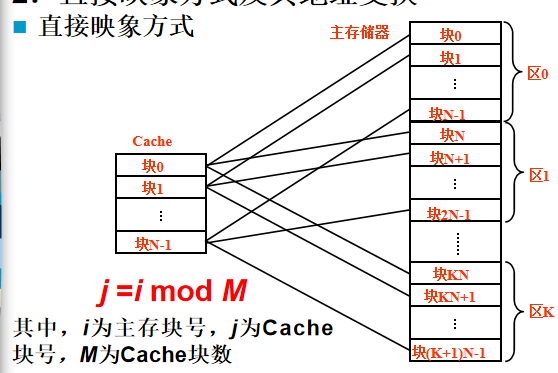
首先明确地址映像：

把主存储器地址空间映像到Cache地址空间

1. 全相连映像及变换

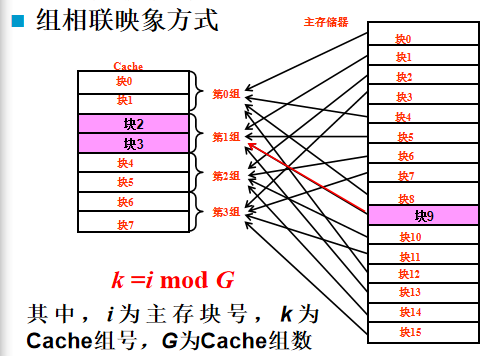


1. 直接映像及其地址变换



特点是对主存储器进行了分区，这样显得更有序，而不是全相连中的任何主存块都能映射到随机的Cache块上

1. 组相连，结合了全相连和直接相连的优势



1. 非阻塞**CACHE**技术

为了减小Cache的不命中开销，采用非阻塞Cache技术

即Cache不命中时仍允许CPU进行其他的命中访问。即允许不命中下命中。（存储器必须能处理多个不命中）

**23**、**TLB（第五章 5.7 虚拟存储器）**

快表，方便查询时直接读取存储内容

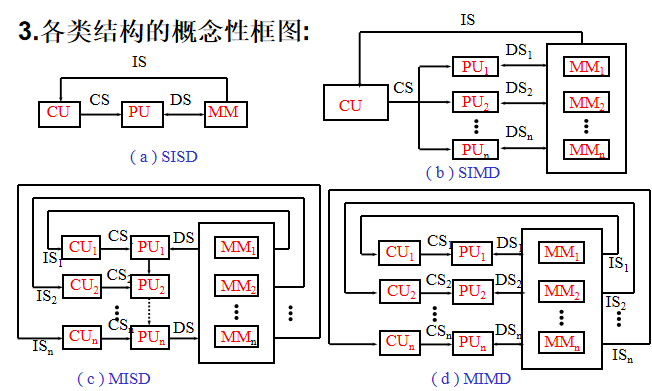
用快速硬件构成一个小容量的页表，这个小容量的页表被称为块表，可以叫TLB translation lookaside buffer。与块表相对应，存放在主存储器中的页表称为慢标。慢表是一个全表，而块表只是慢表中的一小部分。

1. 平均**CPI**

平均每条指令的时钟周期数，降低CPI有利于提高处理机的执行效率

1. **SIMD**

按照flynn分类法（三个重要概念。指令流，数据流，多倍性）



**25**、乱序发射/乱序执行/顺序完成

这是在处理多流出流水线的调度问题时会遇到的问题，为了保证同一时钟周期内可以发出多条指令。 也对应着不同的前瞻方式，他们的实现都通过借用定向技术来确保实现。

1. 指令动态调度

属于指令调度的一种，动态调度即在程序的执行过程中，依靠专门硬件对代码进行调度，减少数据相关导致的停顿。

几个重要思想，为了允许乱序执行，5端流水线的译码阶段再分出两个阶段，ID取指令译码改为流出、读操作数两个步骤

（与之相反的静态调度大多是依靠编译器对代码进行优化。）

27、LRU算法

一种页面替换算法，最久未使用替换算法，来提高命中率。

**二、简答题**

1. 对流水线的**冲突处理**通常有两种方式：静态调度方式和动态调度方式；说明这两种方式，并分别举出其实现方式。（第三章）

静态调度：

- **依靠编译器对代码进行静态调度**，以减少相关和冲突

- 它不是在程序的执行过程中，而是在编译期间进行代码调度和优化

- **通过把相关的指令拉开距离来减少课程产生的停顿**

**例如预测分支成功，预测分支失败，延迟转移技术，都是通过编译器来实现的**

动态调度：

-在程序执行过程中，**依靠专门的硬件对代码进行调度**，减少数据相关导致的停顿

例如把流水线的译码阶段分为两个阶段，流出和读操作数阶段。

采用tomasulo算法，对寄存器进行换名，以减少WAR和WAW冲突

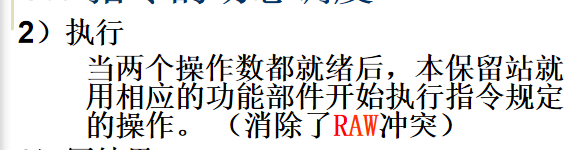
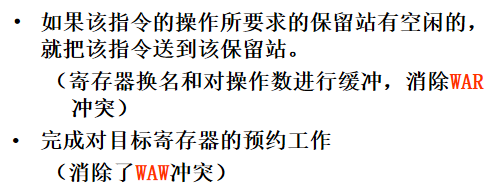
1. tomasulo采取了什么方法避免三种数据冲突。

首先明确三种数据冲突为RAW，WAR，WAW

Tomasulo核心思想：

- **记录和检测指令的相关**，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**



1. 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

X = a+ b;

Y = x + c;

Z = y + d;

1 2 具有数据相关WAR

2 3同样具有数据相关

改为：

X = a + b

Y = c + d

Z = x + y

1. 计算机系统结构、计算机组成和计算机实现的概念与关系。

广义的系统结构定义：指令集结构、组成、硬件；

计算机系统结构概念的实质：确定计算机系统中软、硬件的界面，界面之上是软件实现的功能，界面之下是硬件和固件实现的功能；

**计算机组成**：计算机系统结构的逻辑实现。即根据计算机系统结构所指定的功能，从逻辑上完成对计算机的设计。（这里包括各部件的逻辑实现，部件之间的互相连接以及物理机器级中的数据流和控制流的组成以及逻辑设计等）

**计算机实现**：即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，器件的集成度和速度，模块、插件、底板的划分与连接，信号传输等。

相互之间的联系：

同一种系统结构可以有不同组成上来实现

一种组成可以有多种物理实现

1. **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的思想：由于软硬件的局限，尤其是存储器。人们普遍认为计算机的指令越丰富越好。

CISC指令系统的问题：

- 2 8 定律。指令系统中20%的指令是经常被使用的，所占比例约为全部程序的80%。而剩下的80%指令很少被用到，只站程序的20%，这样大大增加了控制器的设计难度。

- 编译难度的增加和译码时间的延长。单纯通过增加指令的复杂性已不再能提高程序的执行速度了。

- VLSI技术进步提供的机遇

RISC的特点（精简指令集）

- 精简指令系统及其结构，仅保留使用频率高的指令。

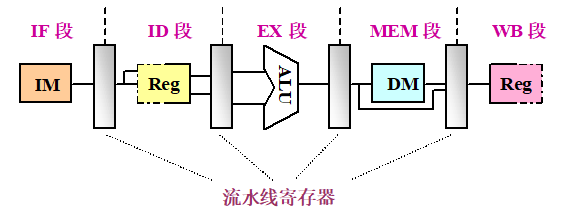
- 不设微程序控制器，由硬连逻辑直接译码。

- 使用大量的通用寄存器

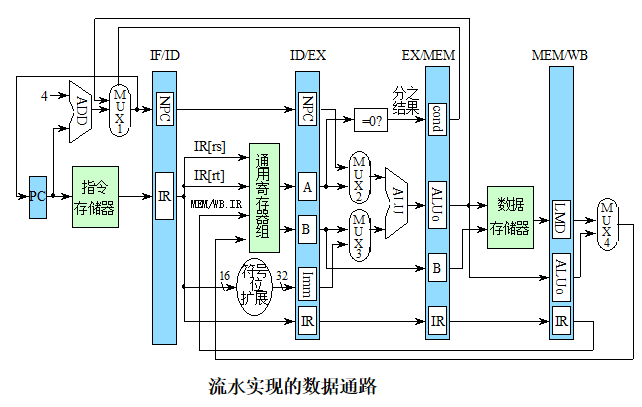
由于精简指令系统一班是由频率高的简单指令组成。易于实现优化编译和流水线技术。

RISC的运算速度要比CISC快约3倍。其中的关键是CPI的下降

1. 简要画出**DLX**多周期（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。







1. **流水线冲突有哪三种？请简述每种流水线冲突。**

结构冲突（资源冲突）

数据冲突（顺序执行时都需要对某流动的数据 进行读写操作，由于实际部件对数据获取的位置不同可能会出问题）

控制冲突（条件分支）

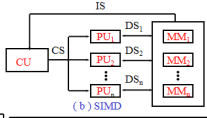
**8**、**如果某计算机系统有3 个部件可以同时改进，则这3 个部件经改进后达到的加速比分别为：S1=30,S2=20,S3=10。**

**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？**

**百分之百**

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。**

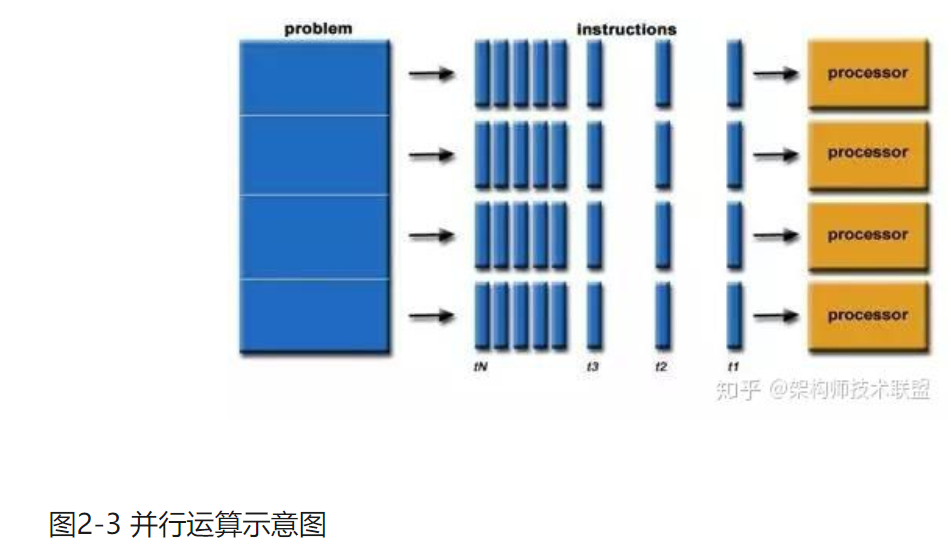
原型是并行处理机，即SIMD计算机



单指令流，多数据流是其重要特点。

现代的GPU对图像和图形处理是十分高效率的，这是因为GPU被设计为很高的**并行架构**这样使得比通用处理器CPU在大的数据块并行处理算法上更具有优势。

因此CPU和GPU架构差异很大，CPU功能模块很多，能适应复杂运算环境；GPU构成则相对简单，目前**流处理器和显存控制器**占据了绝大部分晶体管。

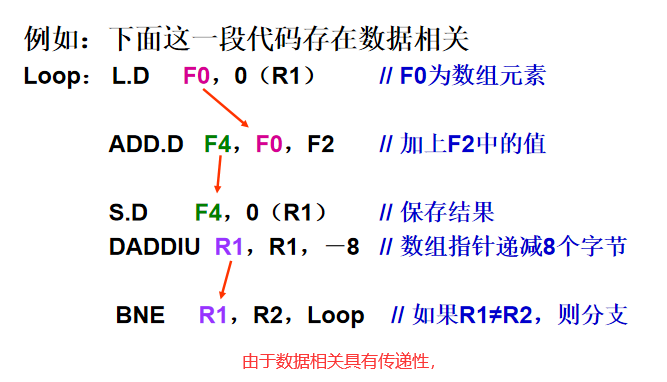


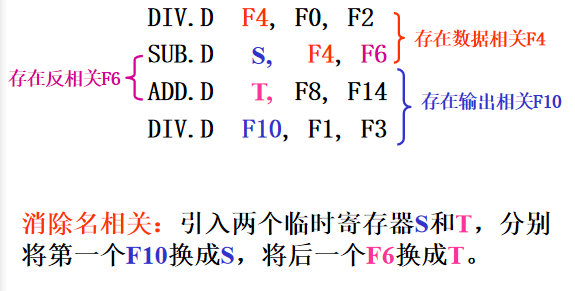
1. **名相关和数据相关会产生写读冲突、读写冲突、写写冲突。简述这三种冲突，并举例说明是如何造成的。**



**\_**

数据之间存在前后传送的发生





1. 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。

局部性原理的存在，且不同层次的主要功能不同，且为了实现不同功能，其硬件材质也有区别。

若不分层，整体化的存储系统性能提升缓慢且有限。

**从大量的统计中得到的一个规律是，程序中对于存储空间90%的访问局限于存储空间的10%的区域中，而另外10%的访问则分布在存储空间其余90%的区域中。**

**而访存的局部性包括两个方面**

**- 时间局部性：如果一个存储项被访问，则该存储项很可能被再次访问**

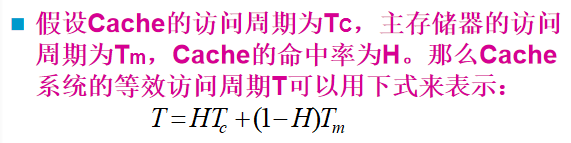
**- 空间局部性：如果一个存储项被访问，则该项及其相邻项可能很快被一起访问。**

为了解决这个问题，我们就将存储系统层次化，在时间局部问题上，把经常用的内容放入M1（快速的），把相邻的放入M1。

现在最常见的层次化系统即为 cache-主存层次，和主存-辅存层次。与之相对应的即Cache存储系统（Cache和主存储器构成）和虚拟存储系统（主存储器和磁盘存储器））

1. 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。（第五章）

平均访存时间 = 命中时间 + 不命中率x不命中开销



优化硬件架构，降低Cache访问周期时间消耗

优化硬件架构，减少主存储器的访问周期时间消耗。

**减少不命中率：**

增加Cache块的大小（适当增加）

提高相连度

增加Cache容量

Cache预期：在用到某信息块之前就将其预取进Cache或缓冲器中

编译器优化

**降低不命中开销**：

让读不命中优先于写

写缓冲合并

采用两级Cache

请求字处理技术

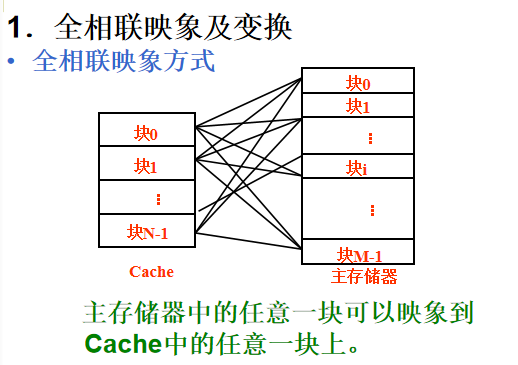
非阻塞Cache技术

**减少命中时间**：

使用容量小、结构简单的Cache：硬件越简单，速度就越快。应使Cache足够小，以便可以与CPU放在同一块芯片上。

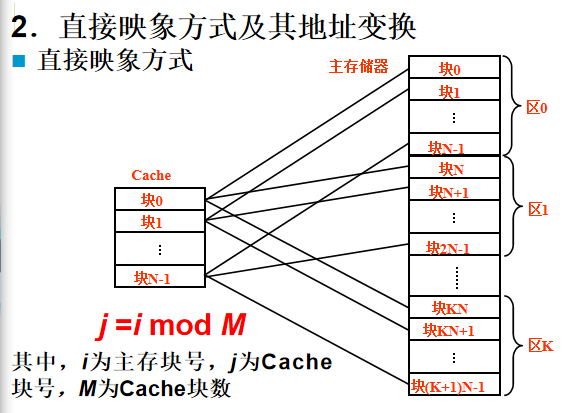
虚拟Cache：将Cache、主存储器、磁盘存储器组织成物理地址Cache存储系统

1. **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优缺点。



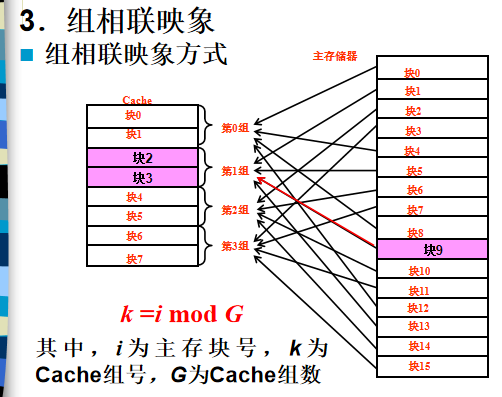
优点：块冲突小，控制简单，Cache利用率高

缺点：查找速度慢（因为没有顺序，需要把整个Cache扫描一遍才知道是否预存）



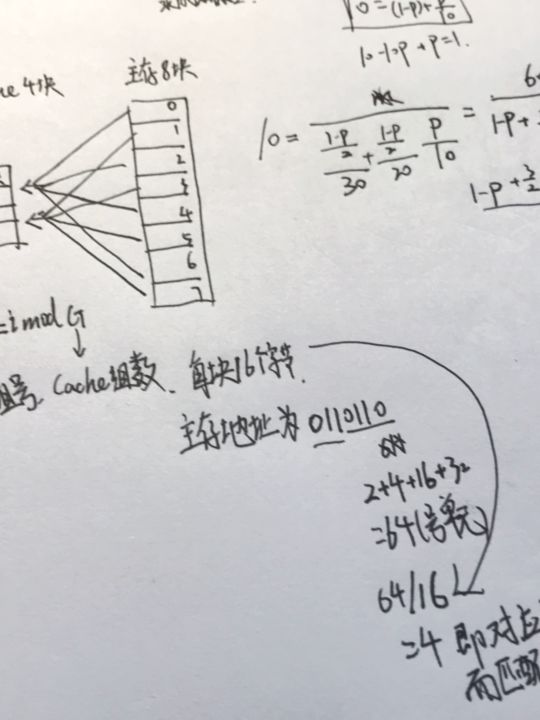
优点：硬件实现简答，不需要**相联存储器**，并且只需要比较区号，速度较快

缺点：块的冲突率较高



优点：块的冲突率大大降低，块的利用率提高，并且查找速度比全相连方式快

1. 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，每块大小为16个字节。某程序运行时，要访存主存地址（二进制）为0110110的字节，则访问Cache的哪一块？

对应主存的3号块，cache的1号组

1. **解决流水线瓶颈问题有哪些方法，应用场合和效果有何异同？**
2. 细分瓶颈段
3. 重复设置瓶颈段

前者用于工作可细分的情况，后者用于工作不可细分的情况。

二者效果一样

1. 简述通过软件（编译器）来减少分支延迟的3种**静态方法**及它们的共同特点。

通常有三种通过软件来减少分支延迟的方法

1. 预测分支失败
   1. 若分支失败，则把其视为一条普通的指令，其后的指令继续顺序执行即可。
2. 预测分支成功
   1. 假设分支转移成功，并且从分支目标地址处取指令执行
   2. 起作用的前提：需要知道分支目标地址，后知道分支是否成功
3. 延迟转移技术

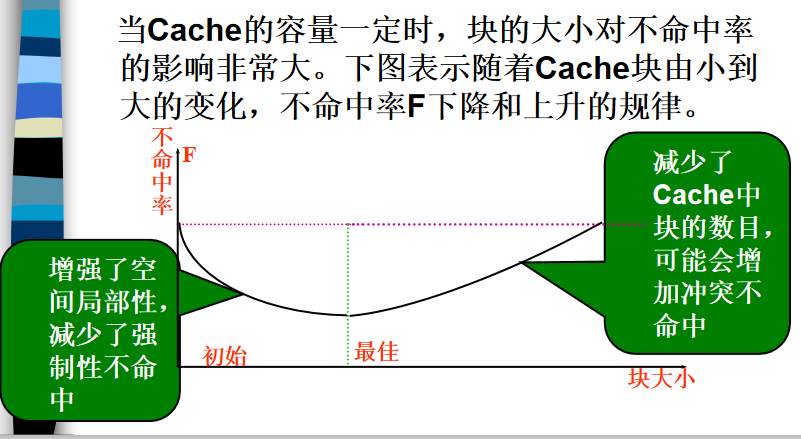
特点：都需要分支目标地址，这样才能排空后再取，或者方便跳转

**标准答案**：

1. 预测分支失败：沿失败的分支继续处理指令，就好像执行了一条普通指令。当确定分支是失败时，说明预测正确，流水线正常流动。当确定分支是成功时，流水线就把在分支指令之后取出的指令转化为空操作，并按分支目标地址重新取指令执行。
2. 预测分支成功：当流水线ID端（第二阶段，指令译码/读寄存器）检测到分之后，一单计算出了分支目标地址就开始从该目标地址取指令执行。
3. 延迟分支：主要思想是从逻辑上“延长”分支指令的执行时间。把延迟分支看成是由原来的分支指令和若干个延迟槽构成。不管分支是否成功，都要按顺序先执行延迟槽中的指令。

三种方法的共同特点：他们对分支的处理方法在整个程序的执行过程中始终是不变的。他们要么总是预测分支成功，要么总是预测分支失败。

17、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?



在初期达到最佳状态前，通过增加块大小，增强了空间局部性，减少了强制不命中。

在块大小超过最佳范围的时候，减少了Cache中的块的数量，可能会增加冲突不命中。

1. 简要说明提高计算机系统**并行性**的3 种技术途径，并各举一例

#### 时间重叠（流水线技术）

引入时间因素，让多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部分，以加快硬件周转而赢得速度。

#### 资源重复（超标量处理机、阵列处理机）

引入空间因素，以数量取胜。通过重复设置硬件资源，大幅度地提高计算机系统的性能，资源重复是实现并行性中的同时性

#### 资源共享（多道程序、分时系统、计算机网络、分布处理系统）

是一种软件方法，它使得多个任务按一定时间顺序轮流使用同一套硬件设备。

在不同的级别采用并行技术。如：微操作级、指令级、线程级、进程级、任务级等

*（提高计算机性能的两个重要方法*

*- 缩短执行每条指令所需的平均周期数（CPI），如RISC技术*

*- 调高处理机在执行指令中的并行度，即同一时刻中处理机内同时运行多条指令。*

*）*

增加硬件，即提供多条合适的流水线

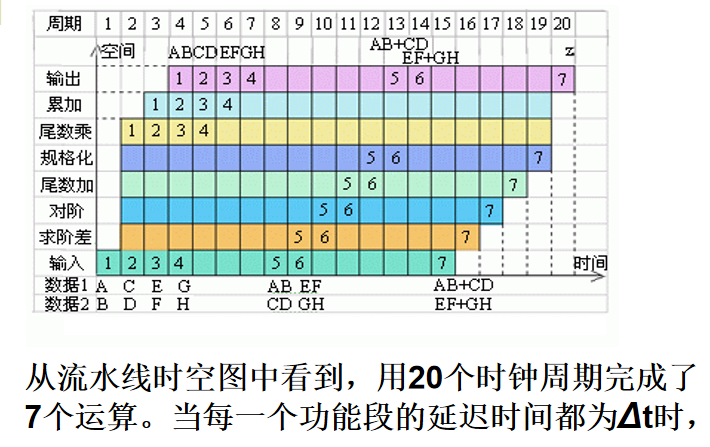
**三.综合题**

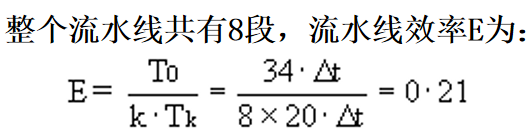
**1**、计算机运行以下指令：

线性多功能静态流水线，输入任务是不连续的情况，画出该计算的时空图,并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 （第三章）



解：





吞吐率为 7 / 20 即20个单位时间内共运行了7条指令

加速比为(4x4+2\*6+1\*6）/20 = 1.5即线性模式和流水线模式消耗时间的比值

**2**、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：



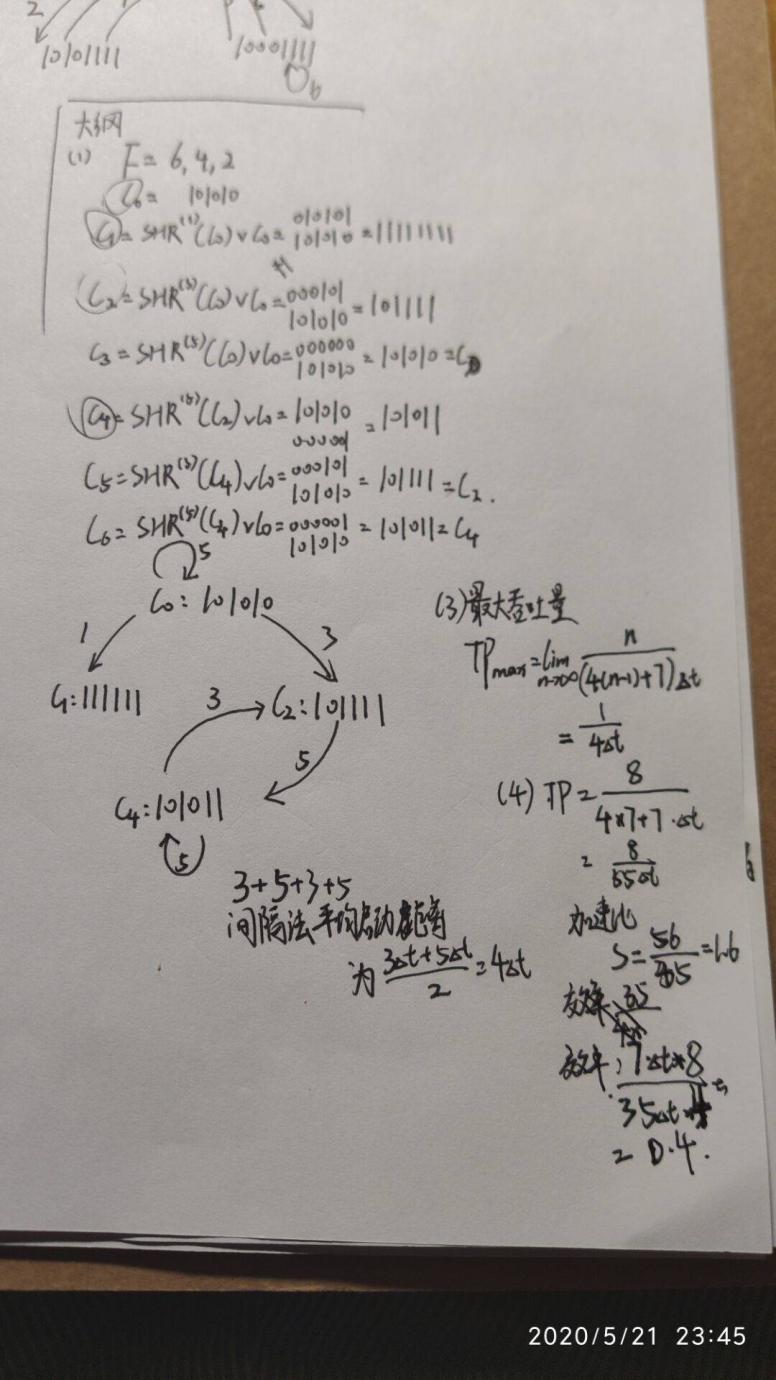
（**1**）写出禁止向量和冲突向量

（**2**）画出调度状态图

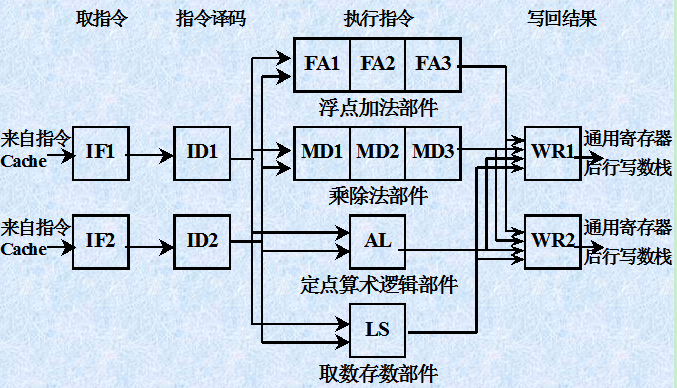
（**3**）求出最大吞吐量

（**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



**3**、超标量机的相关性问题以及调度（第三章）



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

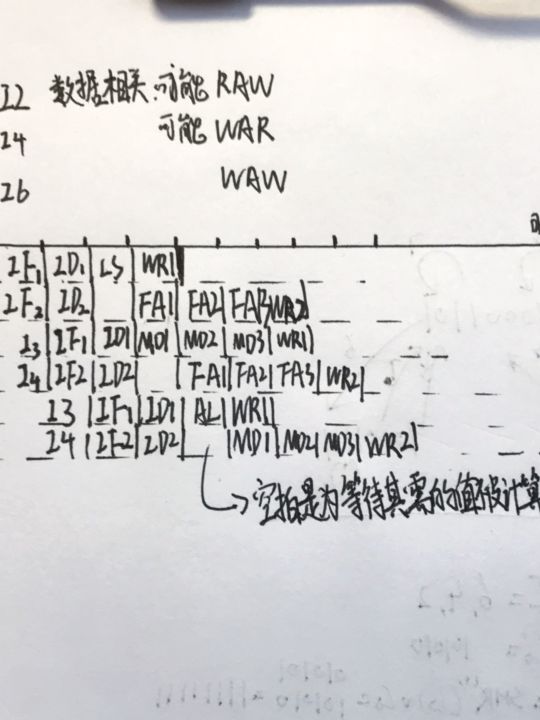
I6：FMUL R6, R7 ；R6←(R6)×(R7)

（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。)

解：



注意之前给出的结构图，留意每个寄存器在哪一拍被送入数据，计算得到数据。

**4**、**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. 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。

- 流水线静态调度，是通过编译器优化的方式调整指令的执行顺序，减少冲突和延迟，避免空拍。

- 通过编译器优化降低Cache不命中率，即在编译的时候，对程序中的指令和数据进行重新组织，数组合并，内外循环交换

- 选用合适的页面替换算法，只需要改变算法即可，可以根据当时那一批指令的特点来及时的调整算法以改善性能。

1. 举三个例子说明系统中采用**硬件来提高性能**的方法和效果。

- 超标量处理机通过重复设置多个取指令部件，多个译码，执行和写会部件，使其同时工作来提高指令的执行速度，以增加硬件资源为代价来换取处理机性能。

- 超流水线处理机只增加少量硬件，通过各部分硬件的充分重叠工作来提高处理机性能

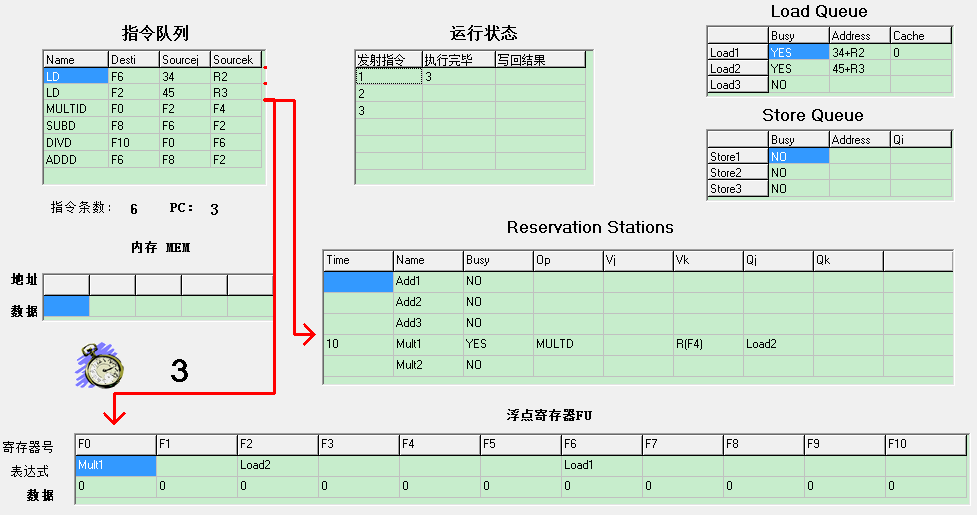
- 虚拟存储器中引入快表，使查询速度加快。

**7、**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：

1. 写出tomasulo算法的核心思想。
2. 写出第4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。

****

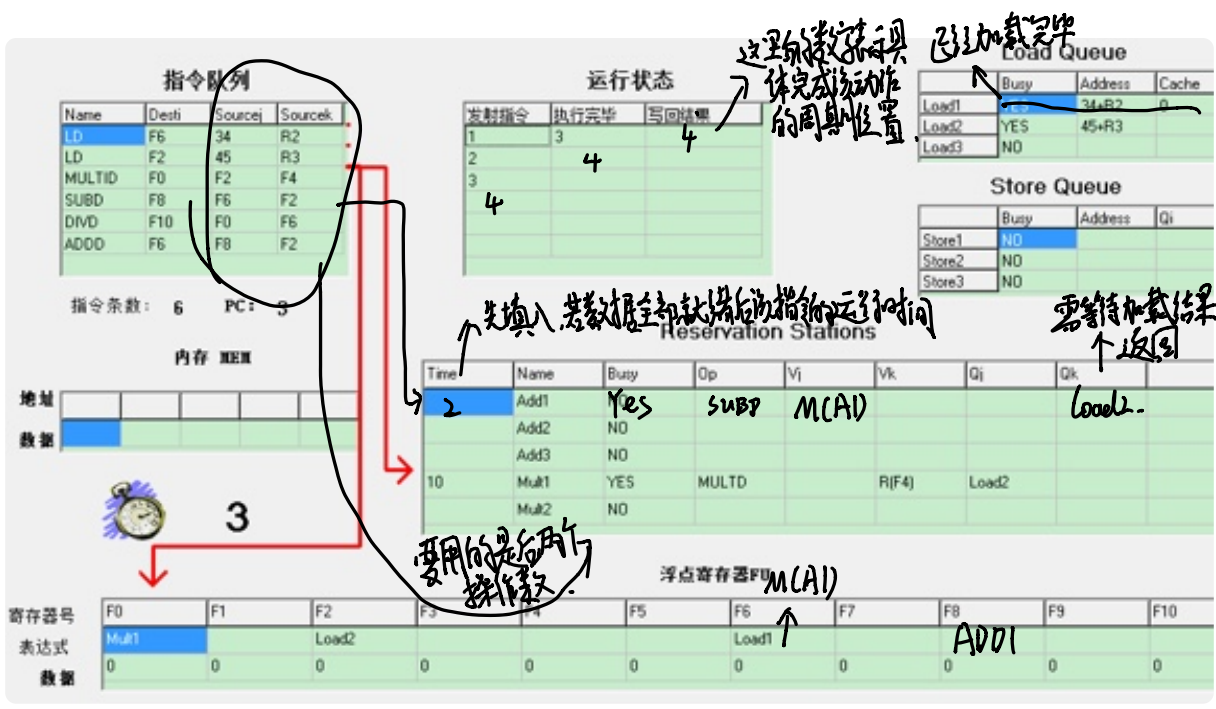
解：

核心思想：

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

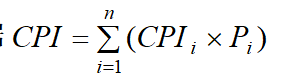




8、某台主频为400MHz的计算机执行标准测试程序，程序中指令类型、执行数量和平均时钟周期数如下：

|  |  |  |
| --- | --- | --- |
| 指令类型 | 指令执行数量 | 平均时钟周期数 |
| 整数 | 45000 概率：45/130 | 1 |
| 数据传送 | 75000 75/130 | 2 |
| 浮点 | 8000 8/130 | 5 |
| 分支 | 2000 2/130 | 2 |

求该计算机的平均CPI、MIPS和程序执行时间(单位：us)。

=1.84

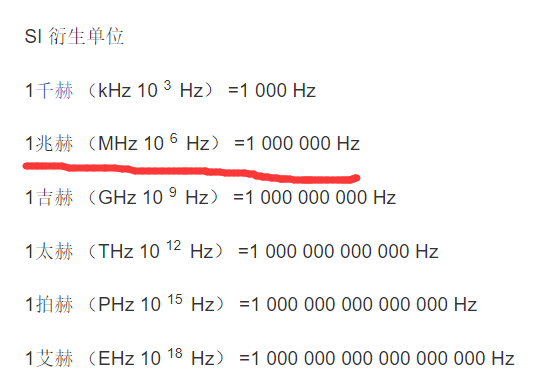
T\_每条指令平均执行时间= CPI x Tc = CPI /f\_c

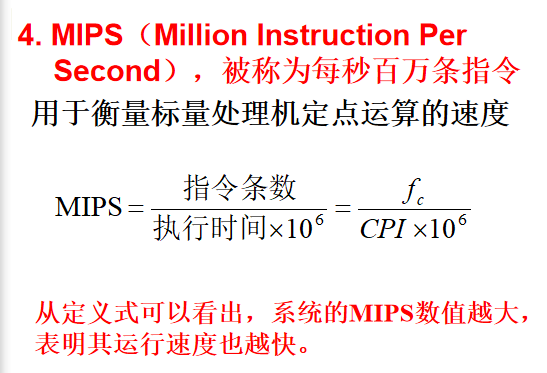
T = 1.84 x (1/400MHz) = 1.84 x 2.5ns = 4.6 ns

MIPS = f\_c / (CPI x 10^6) = 4x10^8 / (1.84\*10^6) = 217.4

（

1Hz = 1/s，即在单位时间内完成振动的次数，单位为赫兹（1赫兹=1次/秒）。





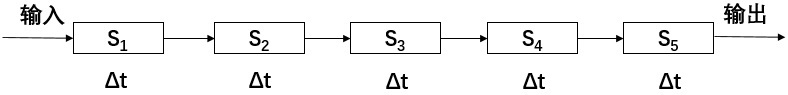
）

9、一台1600MHz计算机执行测试程序，指令类型、执行数量和平均时钟周期数如下



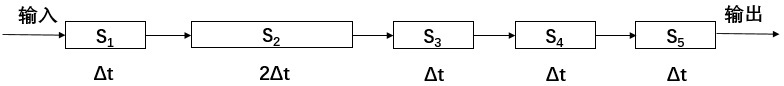
已知平均CPI等于2，则浮点指令的CPI为多少?

解：可计算浮点指令的平均周期数4

10、一条各流水段执行时间均等的5段线性流水线，各段的执行时间均为Δt，其连续执行了10个任务，且不考虑数据与控制冲突，则其实际加速比为多少?（精确到小数点后2位）

解：s = 50 / 14 = 3.57

11、一条各流水段执行时间不完全相等的5段线性流水线，假设其第1、3、4、5段的执行时间为Δt，第2段的执行时间为2Δt，其连续执行了5个任务，且不考虑数据与控制冲突，则其**实际效率**为多少?（精确到小数点后2位）



解： 6x5 / 14 x 5 = 0.43

12、假设分支目标缓冲的命中率为95%，程序中无条件转移指令的比例为5%，没有无条件转移指令的程序的CPI值为1。假设分支目标缓冲中包含分支目标指令，允许无条件转移指令进入分支目标缓冲，则程序的CPI值为多少? 假设原来的CPI=1.2。（精确到小数点后2位）

1.02 (从无条件转移指令带来的延迟x入手求解）

13、假设有一条长流水线，仅仅对条件转移指令使用分支目标缓冲。假设分支预测错误的开销为4个时钟周期，缓冲不命中的开销为3个时钟周期。假设命中率为95%，预测精度为95%，分支频率为10%，没有分支的基本CPI为1。程序执行的CPI为 多少?（精确到小数点后3位）

解：

这种题，如果命中了开销就是0，不命中就要根据比例乘以开销。

10% \* （5% \* 3 + 95% \* 5% \* 4）+ 1 = 1.034

14 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10个时钟周期，L2的不命中开销是200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。问：每条指令的平均停顿时间是多少个时钟周期？

1.4 x 2% x 200

