一、 名词解释

1、ISA

指令集体系结构（Instruction Set Architecture），定义了处理器执行的基本操作集合，包括支持的指令集、寄存器、内存寻址模式等软硬件接口规范。

2、RISC和CISC

RISC（**精简指令集**，Reduced Instruction Set Computer）：**指令少且简单**，单周期执行，强调流水线效率（如ARM、RISC-V）。

CISC（**复杂指令集**，Complex Instruction Set Computer）：**指令多且复杂**，单指令可完成多步操作，硬件设计复杂（如x86）。

3、SIMD和MIMD

SIMD（**单指令多数据，Single** Instruction Multiple Data）：一条指令同时处理多个数据（如GPU向量运算）。

MIMD（**多指令多数据，Multiple** Instruction Multiple Data）：多个处理器独立执行不同指令处理不同数据（如多核CPU）。

4、Amdahl定律

加快某部件性能能够取得的系统加速比，受限于该部件的执行时间在系统总执行时间中所占的比例。

当某一部分的加速比为 S_p（不一定是并行化，比如优化某个模块），Amdahl定律也可以表示为：

$$S_{\text{overall}} = \frac{1}{(1-f) + \frac{f}{S_p}}$$

其中：

- f：可加速部分所占比例（例如 0.6 表示60%的任务可以被加速）
- S_p：该部分的加速比（如原来需要1秒，现在只需0.25秒，则 $S_p = 4$）
- S_{overall}：整个系统的加速比

5、平均CPI

Cycles Per Instruction：平均每条指令需要的时钟周期数

//MIPS=时钟频率（MHz）÷CPI，意为每秒百万条指令

6、时间重叠、资源重复、资源共享

时间重叠：**流水线技术**，通过时间并行提升效率。

资源重复：**复制硬件资源**（如多处理单元）支持并行。

资源共享：多个任务**分时复用**同一硬件（如多线程）。

7、加速比，吞吐率，效率

加速比是衡量使用并行处理后性能提升的程度，**原时间比改进时间**（一般大于1）；
吞吐率是指单位时间内完成的工作量（一般以指令数表示），如指令/秒；
效率是比较有效运行时间与总运行时间之比（在时空图中指有效面积÷大矩形）。

8、多功能流水线

流水线的各段可以进行不同连接，执行多种不同类型操作

9、静态和动态流水线

静态流水线：功能段**固定配置**，各段只能按照同一种功能的连接方式进行工作。
动态流水线：功能段可**动态分配**，能够按照不同方式连接，同时执行不同任务。

10、名相关、数据相关、控制相关

名相关：指令**使用相同寄存器名**但无数据依赖。
反相关：i读的名与j写的名相同；
输出相关：i与j写相同的名；
数据相关：指令间存在数据依赖，即**一条指令的结果是另一条指令的操作数**（如ADD R1, R2 → SUB R3, R1），并且数据依赖具有传递性。
控制相关：由**分支指令**引起的相关，影响后续指令执行路径。

11、结构冲突、数据冲突、控制冲突

结构冲突：**硬件资源争用**（如内存访问冲突）引起的流水线停顿。
数据冲突：**数据依赖**需要用到前面指令的结果导致流水线阻塞。
控制冲突：**分支预测错误**导致流水线清空。

11、定向技术

允许在流水线中提前将计算结果**直接传递给后续需要此结果的指令**，避免写后读（RAW）冲突引起的停顿。

12、寄存器换名技术

通过**重命名冲突的寄存器**来消除名相关，从而提高指令级并行度。

13、指令的静态调度和动态调度

静态调度是**在编译时**确定指令执行顺序（如循环展开），把相关的指令隔开；
动态调度则是**在运行时**依靠专用硬件来调整指令执行顺序（如Tomasulo算法）。

14、Tomasulo算法

一种指令动态调度算法，通过**保留站、寄存器换名和结果转发**解决数据冲突，支持乱序执行。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冲突的可能性减小到最少。

15、分支历史表BHT

Branch History Table：用于记录分支指令的**历史行为**，帮助进行分支预测。

16、分支目标缓冲器BTB

Branch Target Buffer：存储分支成功的分支指令的**目标地址**，预测时直接跳转，加速分支预测过程。

17、ROB技术

Re-order Buffer：用于**在乱序执行中记录指令**的结果，一条指令在执行完毕之后不会立刻写回，而是先在Buffer中等待，待确认后再写回，确保指令按程序顺序提交结果。

18、超标量处理机

每个时钟周期发射多条指令，依赖多功能单元和动态调度实现并行。

19、时间并行和空间并行

时间并行：**流水线**技术，时间上重叠执行。

空间并行：复制**硬件资源**（如多核）并行处理。

20、循环展开技术

编译器**将循环体复制多次**，减少分支次数，提升指令级并行度。

21、向量处理机

在流水线处理机中设置向量数据表示和处理相应的向量指令，称为向量处理机。

22、链接技术

有先写后读相关的两条指令，在不出现部件功能冲突和源向量冲突的情况下，可以把功能部件链接起来进行流水线处理，以达到加快处理的目的。

23、时间局部性和空间局部性

时间局部性：**近期**访问的数据可能再次被访问。

空间局部性：访问某地址后，其**邻近**地址可能被访问。

24、平均访存时间

AMAT=Hit Time+Miss Rate×Miss Penalty

平均访存时间 = 命中时间 + 失效率 × 失效开销

从发出访存请求到得到所需数据的平均延迟时间，综合衡量缓存性能。

25、强制性不命中、容量不命中、冲突不命中

强制性不命中（Compulsory）：首次访问数据必然缺失。

容量不命中（Capacity）：缓存容量不足导致数据被替换。

冲突不命中（Conflict）：组相联中因映射冲突被替换。

26、直接映像、全相联映像、组相联映像

直接映像：每块固定映射到唯一缓存行（易冲突）。

全相联：数据可存入任意行（成本高）。

组相联：折中方案，缓存分为组，每组多行（如4路组相联）。

27、Victim（“牺牲”） Cache

位于主缓存之后的小容量全相联缓存，保存最近被主缓存替换的数据，减少冲突不命中。

28、伪相联映像

先按直接映射访问，若不命中则尝试同组另一行（类似组相联），平衡速度和冲突率。

29、非阻塞CACHE技术

支持缓存缺失时继续处理其他请求而**不必等待缓存未命中解决**，隐藏访存延迟。

31、TLB

Translation Look-aside Buffer，**缓存虚拟地址到物理地址的转换结果**，TLB的作用在于存储最近使用的虚拟地址到物理地址的映射关系，使得频繁访问的页面不需要每次都通过查询页表来进行地址转换，从而加快虚拟地址到物理地址转换速度。

二、简答题

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。

计算机系统结构：定义了**程序员看到的计算机属性**，即概念性的结构和功能特性。它包括指令集架构（ISA）、数据类型、寻址模式、寄存器组织等。

计算机组成：描述了计算机各个部分**如何连接在一起工作**以实现特定的系统结构。这涉及到控制单元的设计、数据通路的组织、存储系统的层次结构等。

计算机实现：指的是实际构建计算机所使用的**具体技术**，如电路设计、逻辑设计、物理封装等。

三者之间的关系是逐层递进的，计算机系统结构决定了计算机的功能需求，计算机组成则是为满足这些需求而进行的具体设计，最后计算机实现将这些设计转化为具体的硬件产品。

2、论述RISC与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。

二者区别：

RISC强调**简化指令集**，减少指令种类，每条指令**执行时间较短**，通常在一个时钟周期内完成。

CISC则通过复杂的指令集来**减少程序中指令的数量**，但单条指令执行时间较长。

RISC提高效率的方式：

简化指令集：使指令可以在一个时钟周期内执行完毕。

固定长度的指令格式：便于指令解码和流水线处理。

大量使用寄存器：减少了内存访问次数，提高了速度。

Load/Store结构：只有load和store指令可以访问内存，其他操作都在寄存器间进行。

举例说明：

在RISC架构中，通过**流水线技术**，不同的指令阶段（取指、译码、执行等）可以并行处理，从而显著提高指令吞吐量。

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

4、Tomasulo分别采取了什么方法避免RAW冲突、WAR冲突和WAW冲突？

避免RAW（Read After Write）冲突：通过**保留站**（Reservation Stations）和**公共数据总线**（Common Data Bus），允许在产生结果的同时将其广播给所有需要该值的保留站或寄存器。

避免WAR（Write After Read）冲突：通过**寄存器换名**技术，确保读操作总是获取到正确的值，即使后续有写操作也要等到读操作完成后才更新寄存器。

避免WAW（Write After Write）冲突：同样借助**寄存器换名**技术，确保最终写入寄存器的是最新的正确值，而不是过期的数据。通过这种方式，即使是乱序执行也能保证正确的程序语义。

5、多指令流出技术中，超标量与超流水线的核心差异是什么？

超标量（Superscalar）

核心思想：在单个时钟周期内**同时发出多条指令**。这意味着处理器内部有多个执行单元，并且可以在同一时刻处理不同的指令。

实现方式：通过增加硬件资源（如多个算术逻辑单元ALU、浮点运算单元FPU等），使得能够并行处理多个独立的指令（空间换时间）。这要求处理器具备复杂的调度机制来识别可以并行执行的指令以及解决任何数据或控制依赖性。

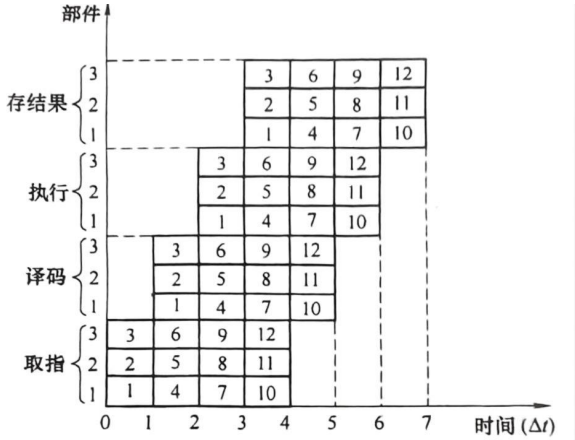


图 5-31　度 $m=3$ 的超标量处理机时空图

超流水线（Superpipelining）

核心思想：将传统的几段流水线**进一步细分**，以减少每段流水线所需的时间，从而允许**更高的时钟频率**。这样即使每个时钟周期只发射一条指令，也能因较高的时钟频率而加快整体处理速度。

实现方式：通过对现有流水线阶段进行**更细粒度的划分**，例如将原本5段流水线扩展为10段甚至更多。这样做减少了每个阶段的工作量，允许更快地完成每个阶段的任务，进而支持更高的时钟频率。

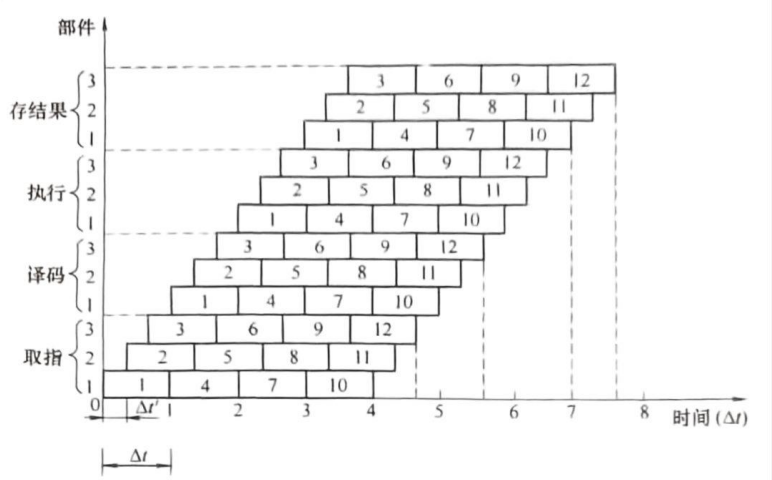


图 5-33　每 Δt′流出一条指令且度 $m=3$ 的超流水线处理机时空图

//**超长指令字（VLIW, Very Long Instruction Word）**

VLIW是一种计算机处理器架构设计，它通过将多个操作组合成一个指令包来提高指令级并行性,**用一条长指令来实现多个操作的并行执行**。时空图特点是取指译码都普通，但是执行时可以垂直地执行多个指令。

与超标量处理器不同，VLIW处理器在编译时由编译器负责指令的调度和依赖关系的处理，而不是在运行时由硬件处理。

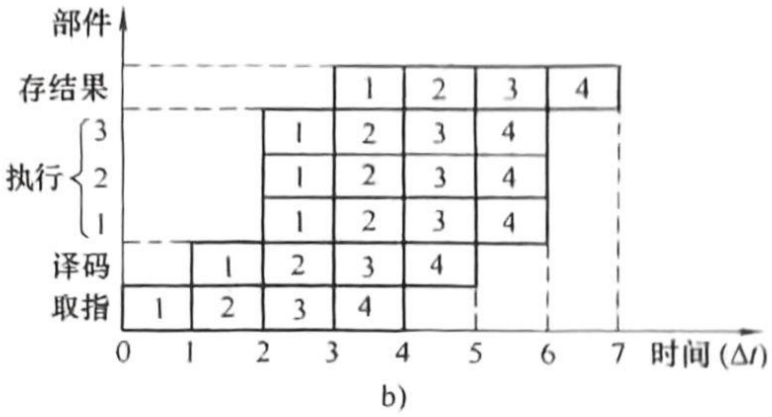
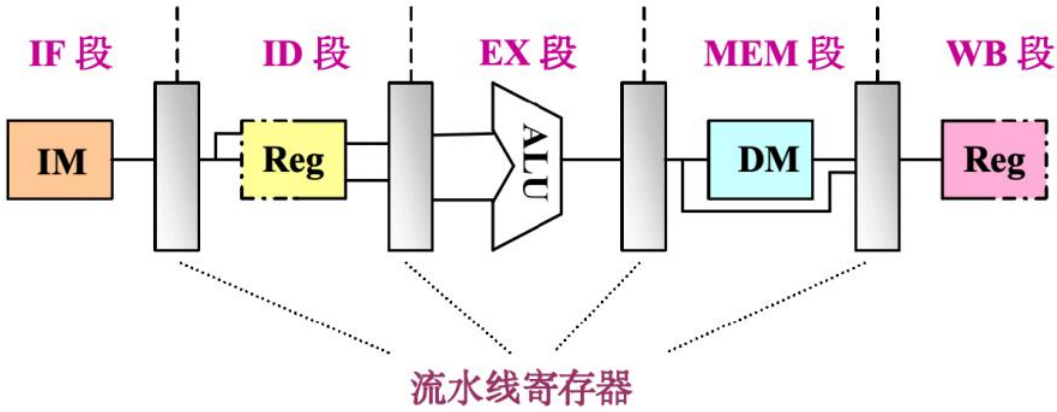


图 5-32　VLIW 处理机

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指令

取指（IF）：从程序计数器（PC）中获取当前指令的地址，并从指令存储器中读取对应的指令。

译码（ID）：解析指令的操作码、寄存器操作数以及立即数等信息。计算出需要访问的内存地址（基址寄存器内容加上偏移量），同时准备将数据写入的目标寄存器。

执行（EX）：如果是涉及地址计算的指令，这个阶段会完成地址计算。对于load指令而言，主要是计算出实际的内存地址。

访存（MEM）：根据上一阶段计算出来的内存地址，从数据存储器（或缓存）中读取数据。这是load指令的关键步骤，即从内存加载数据到寄存器。

写回（WB）：将访存阶段从内存中读取的数据写回到目标寄存器文件中指定的寄存器。

Store指令

取指（IF）：与load指令相同，从程序计数器（PC）中获取当前指令的地址，并从指令存储器中读取对应的指令。

译码（ID）：同样地，解析指令的操作码、寄存器操作数以及立即数等信息。计算出需要访问的内存地址（基址寄存器内容加上偏移量），并准备好要存储的数据（源寄存器的内容）。

执行（EX）：进行地址计算。对于store指令，这一阶段主要负责计算出最终的存储地址。

访存（MEM）：根据上一阶段计算出来的内存地址，将数据存储器（或缓存）中对应地址的数据更新为源寄存器的内容。这是store指令的核心步骤，即将寄存器中的数据写入到内存。

写回（WB）：对于store指令，此阶段实际上并不执行任何操作，因为不需要将数据写回到寄存器文件中。因此，在一些实现中，可能省略掉这一步骤或者作为一个空操作。

7、流水线冲突有哪三种？请简述每种流水线冲突，并针对每一种冲突提供一种解决办法。

资源冲突：当两个或多个指令在同一时钟周期内**尝试访问相同的硬件资源**（如ALU）时发生。

解决办法：**增加硬件资源**，例如添加额外的ALU。

数据冲突（相关）：由于指令之间的**依赖关系**导致的数据问题。分为读-写（RAW）、写-读（WAR）、写-写（WAW）三种情况。

解决办法：插入**气泡**（nop操作）或使用**转发**技术（forwarding），提前将计算结果传递给后续指令。

控制冲突：由**分支预测错误或跳转指令**引起的冲突，导致流水线中的后续指令无效。

解决办法：采用**动态分支预测**、延迟分支或分支目标缓冲（BTB）来减少误预测率。

8、计算机系统中，实现并行性主要有哪三大途径？请分别举例说明。

时间重叠：通过流水线实现，在一个任务完成前开始下一个任务的部分处理。

例子：CPU流水线执行指令。

资源重复：增加硬件资源以允许更多的操作同时进行。

例子：多核处理器。

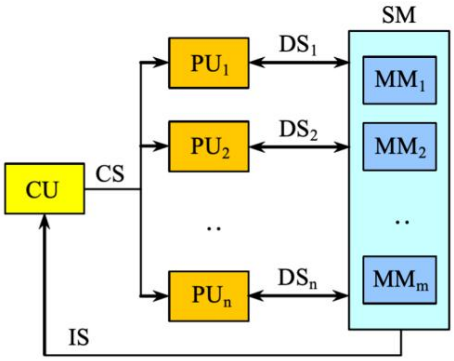
资源共享：允许多个进程或线程共享同一组硬件资源。

例子：多线程技术，如Intel的超线程技术。

9、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。

GPU采用了**SIMD（单指令流多数据流）**的设计方式作为其原型

核心思想：一个控制器发出单一指令应用于多个处理单元，每个处理单元操作不同的数据集。



10、名相关和数据相关会产生RAW冲突、WAR冲突和WAW冲突。简述这三种冲突，并举例说明是如何造成的。

RAW（Read After Write）写后读：后一条指令需要先于它的指令写入的结果。

示例：指令A向寄存器R1写入值，随后指令B从R1读取值。

```
I1: ADD R1, R2, R3   ; 写R1
I2: SUB R4, R1, R5   ; 读R1（需等I1完成）
```

WAR（Write After Read）读后写：在读取某个位置之后写入该位置。

示例：指令A从内存地址M读取，随后指令B写入到M。

```
I1: MUL R1, R2, R3   ; 读R2
I2: ADD R2, R4, R5   ; 写R2（需等I1读完）
```

WAW（Write After Write）写后写：当两个或更多的指令试图连续地写入同一个数据项，第二个写操作不能发生在第一个写操作完成之前。

示例：指令A和B都试图写入寄存器R1，且B应在A之前完成。

```
I1: LD R1, [addr]    ; 写R1
I2: ADD R1, R2, R3   ; 写R1（需保证I1先写）
```

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。

层次化存储系统的存在基于**局部性原理**，包括**时间局部性**和**空间局部性**。时间局部性指的是如果一个信息项被访问，那么它可能很快会被再次访问；空间局部性意味着如果一个信息项被访问，那么与之相邻的信息项也可能很快被访问。因此，通过构建多层次的存储系统（如缓存、主存、磁盘），可以利用局部性原理提高数据访问效率，从而加速程序运行速度。

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。

AMAT（平均访存时间Average Memory Access Time）

$$=\text{Hit Time}+\text{Miss Rate}\times\text{Miss Penalty}$$

命中时间（Hit Time）

优化技术：使用更小、更快的Cache，如将一级Cache设计为直接映像。

示例：采用直接映像方式可减少地址比较时间。

不命中率（Miss Rate）

优化技术：增加块大小或使用组相联映射。

示例：使用更大的Cache块可以利用空间局部性，降低不命中率。

不命中开销（Miss Penalty）

优化技术：引入多级Cache（L1+L2+L3）。

示例：L1 Cache不命中时访问速度较快的L2 Cache，而不是主存。

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优缺点。

（1）全相联映像（Fully Associative）

原理：每个主存块可以放到Cache中的**任意位置**。

优点：冲突最少，命中率高。

缺点：硬件复杂度高，成本大，查找慢。

（2）直接映像（Direct Mapped）

原理：每个主存块只能放在Cache中一个**固定位置**。

优点：实现简单，速度快。

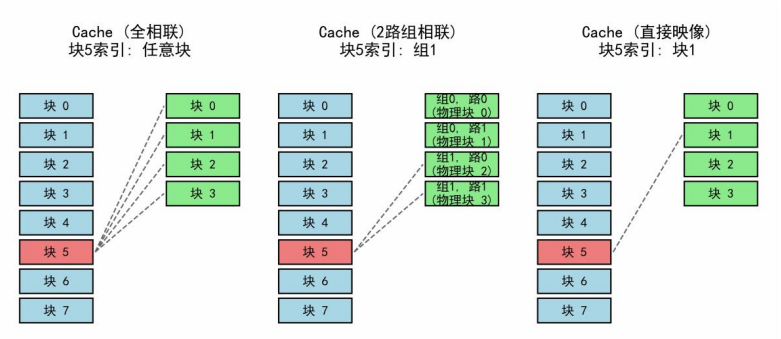
缺点：容易发生冲突不命中。

（3）组相联映像（Set Associative）

原理：将Cache分成若干组，每组包含多个块。主存块只能放入**某一组内的任意位置**。

优点：折中方案，兼顾命中率与实现复杂度。

缺点：比直接映像复杂，但比全相联简单。



14、计算机系统设计中经常使用的四个设计原则和定量原理是什么？请说出它们的含义？

加速经常性事件：优化高频操作比优化低频操作更有效（如优化L1 Cache访问）

Amdahl定律：系统加速比受可改进部分占比限制

CPU性能公式：CPU时间=指令数×CPI×时钟周期，三要素共同决定性能

局部性原理：时间局部性（重复访问）+空间局部性（访问相邻数据），指导存储层次设计

15、在降低Cache不命中的方法中，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?

初期下降原因：

更大的块能更好地利用空间局部性，一次**加载更多相邻数据**，后续访问命中概率增加。

后期上升原因：

冲突增加：较大的块减少了Cache中可容纳的**总块数**，导致更多的冲突不命中。

污染效应：不必要的数据被加载进Cache，**挤出其他有用数据**。

填充浪费：加载了不会被访问的数据区域，造成**资源浪费**。

16、写出三种降低CACHE不命中率的方法并举例说明。

增加块大小（Block Size）

示例：从 32 字节增加到 64 字节，利用空间局部性。

使用预取（Prefetching）

示例：在访问当前块时，自动预取下一个块进入 Cache。

增加相联度

示例：由直接映像改为 2 路组相联，减少冲突不命中。

17、多级Cache设计如何减少不命中开销？

原理：层级过滤

L1 Cache速度快但容量小，用于快速响应大多数访问；

L2/L3 Cache容量大但速度较慢，用于处理L1不命中的请求；

不命中时先访问L2，再访问主存，从而减少平均等待时间。

效果：

减少了对主存的访问次数；

缓解了L1 Cache容量小带来的不命中压力。

18、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。

流水寄存器延迟

影响：流水寄存器用于在不同流水线阶段之间传递数据。每个流水寄存器都会引入一定的延迟，这是因为信号需要时间从一个阶段稳定地传送到下一个阶段。这种延迟会增加每个周期的时间，从而降低流水线的最大工作频率，影响整个系统的吞吐量。

解决方法：可以选用具有较低传播延迟的高性能锁存器作为流水寄存器减少这种延迟

时钟偏移开销

影响：时钟偏移是指由于时钟信号到达流水线中不同组件的时间差异。如果时钟偏移过大，可能导致某些操作过早或过晚执行，破坏了流水线的操作顺序，甚至导致错误结果。严重的时钟偏移限制了流水线能够安全运行的最大频率。

解决方法：通过同步设计确保所有相关电路接收到来自同一时钟源的信号，并且采用平衡的时钟树设计来最小化时钟偏移。

19、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。

细分瓶颈段（Pipeline Stage Division）

定义与原理： 细分瓶颈段指的是将流水线中耗时较长的一个阶段进一步拆分为多个较短的子阶段，以平衡各阶段的时间开销，从而提高流水线的整体吞吐量。

优点：

提高了流水线的平衡性，使得每个阶段的处理时间更加均匀。

增加了流水线深度，理论上可以在单位时间内处理更多的指令。

缺点：

增加了流水线的控制复杂度，可能引入额外的延迟。

每增加一段都会带来额外的寄存器开销，用于存储中间结果。

设置重复瓶颈段（Pipeline Stage Duplication）

定义与原理： 设置重复瓶颈段是指在同一级流水线上复制多个相同的瓶颈功能单元，使多条指令可以同时通过该瓶颈段的不同实例，从而减少瓶颈段造成的等待时间。

优点：

直接增加了关键路径上的资源，减少了瓶颈段导致的停顿。

不改变流水线的基本架构，减少了因重新设计带来的风险。

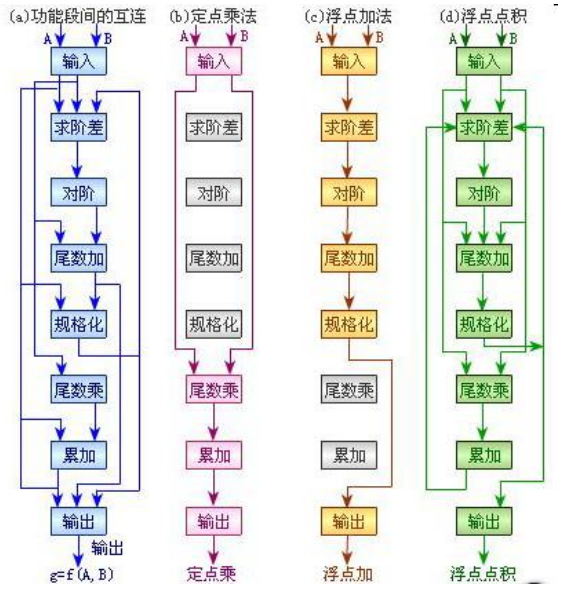
缺点：

需要更多的硬件资源，增加了成本和功耗。

只对特定类型的瓶颈有效，且扩展性有限（如需处理更多指令，则需要更多重复的功能单元）

三、综合题

1、用TI－ASC计算机的多功能动态流水线计算两个向量的点积：Z＝AB＋CD＋EF＋GH。要求：（1）请列出运算顺序安排。（2）画出对应的流水线时空图。（3）计算流水线的实际吞吐率、加速比和效率。



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)。

3. 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？

4、 在一台服务器上运行一段I/O密集型基准测试程序，该测试程序共有3,000,000条指令，运行时间为5毫秒，其中CPU时间占20%，I/O时间占80%。为提高性能，将服务器CPU从1.5GHz升级为3GHz，服务器内存和磁盘升级为访问速度更快的内存和固态硬盘，使得I/O访问速度提高至原来的4倍；升级后，该服务器运行同一段基准测试程序的执行时间为多少毫秒？

5、假设某程序中循环代码段 p:“for(int i=0;i<N;i++) sum+=A[i];” 对应的汇编代码如下：

```
I1:  loop: sll R4,R2,2
I2:        add R4,R4,R3
I3:        load R5,0(R4)
I4:        add R1,R1,R5
I5:        add R2,R2,1
I6:        bne R2,R6,loop
```

若采用 “按序发射、按序完成”的5级流水线：IF（取值）、ID（译码及取数）、EXE（执行）、MEM（访存）、WB（写回寄存器），且硬件不采取任何措施，分支指令的执行均引起3个时钟周期的阻塞，则：

（1）以上指令序列中哪些指令的执行会由于数据相关而发生流水线冲突？

（2）哪条指令的执行会发生控制冲突？

（3）为什么指令1的执行不会因为与指令5的数据相关而发生冲突？

6、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：

时间 / 流水段	1	2	3	4	5	6	7
S_1	×						×
S_2		×				×	
S_3			×		×		
S_4				×			

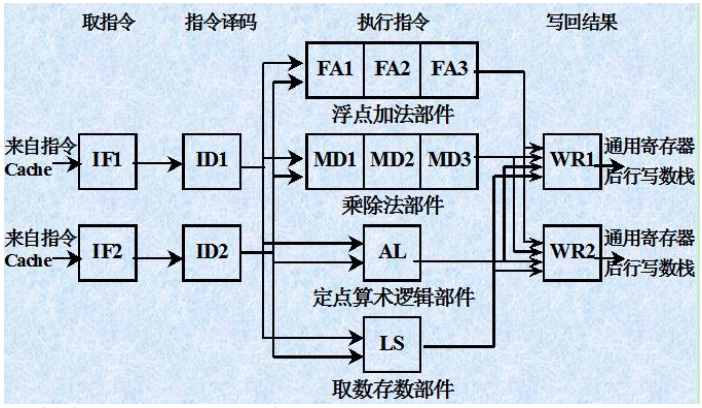
（1）写出禁止向量和冲突向量

（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

7、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。)

8、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。

(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

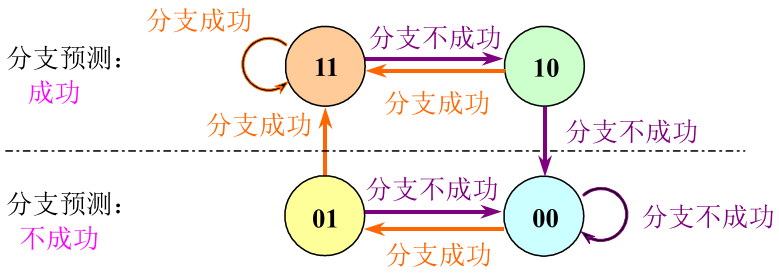
说明该程序访存对Cache的块位置的使用情况，指出发生块失效且块争用的时刻，计算Cache命中率。

9、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。

10、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。

11、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