**计算机组成原理 第七次作业**

**22920212204392 黄勖**

**7.1 解释下列名词。**

1. 流水线技术：流水线技术是一种计算机处理器设计方法，通过将指令和数据的处理过程划分为多个阶段，并使不同阶段的处理单元同时执行不同的指令，从而实现并行处理，提高处理器的效率。
2. 指令流水线：指令流水线是一种流水线技术的应用，它将处理一条指令所需的多个步骤划分为多个阶段，使得每个阶段可以并行执行不同的指令。这样，当一条指令执行完毕后，下一条指令就可以进入流水线的下一个阶段，从而实现指令级并行。
3. 运算流水线：运算流水线是指将一个算术或逻辑运算过程划分为多个阶段，并通过并行执行这些阶段来加速运算的方法。例如，乘法运算可以划分为寄存器读取、乘法运算、结果写入等多个阶段，每个阶段由一个处理单元负责执行。
4. 流水寄存器：流水寄存器是用于存储流水线中不同阶段之间的数据的寄存器。它们用于在不同阶段之间传递数据，确保数据的正确流动。
5. 流水时空图：流水时空图是用于描述流水线中指令执行和数据传输的时间和空间关系的图形表示。它可以显示指令在各个阶段的执行时间和各个数据在流水线中的传输情况，帮助设计和分析流水线系统。
6. 数据冲突：数据冲突是指在流水线中由于指令的数据相关性引起的问题。当一条指令需要使用另一条指令尚未产生的数据时，就会发生数据冲突，需要进行适当的处理，以确保数据的正确性。
7. 结构冲突：结构冲突是指在流水线中由于硬件资源的限制而引起的问题。例如，如果多条指令需要同时访问同一个硬件资源（如寄存器或内存），就会发生结构冲突，需要进行调度或冲突解决。
8. 控制相关：控制相关是指在流水线中由于指令的跳转或分支导致的问题。当一条指令的执行结果决定了下一条指令的跳转地址时，就会出现控制相关，需要进行相应的处理，以保证程序的正确执行。
9. 先写后读冲突：先写后读冲突是指在流水线中，一个指令在写入数据到寄存器后，后续的指令又需要读取该寄存器中的数据，但此时数据还未写入完成，导致读取到错误的数据。
10. 先读后写冲突：先读后写冲突是指在流水线中，一个指令在读取数据之后，后续的指令需要写入该数据，但此时读取指令尚未完成，导致写入的数据被覆盖或错误。
11. 写后写冲突：写后写冲突是指在流水线中，两个或多个指令在相同的寄存器位置上写入数据，导致数据的冲突和覆盖。
12. 气泡：气泡是在流水线中为了解决数据冲突或控制相关等问题而插入的空指令或延迟指令。气泡的插入可以使得流水线各个阶段保持同步，避免数据冲突。
13. 重定向：重定向是一种处理控制相关的技术，在流水线中发生分支或跳转时，通过改变指令流水线的执行顺序，将流水线的执行转移到正确的分支路径上，以避免错误的指令执行。
14. 延迟槽：延迟槽是指在流水线中延迟执行分支或跳转指令后的几条指令位置。延迟槽可以用来填充分支延迟期间的指令执行空闲，提高流水线的利用率。
15. 动态分支预测：动态分支预测是一种通过分析程序运行时的分支历史信息来预测分支方向的技术。它根据过去的分支行为来预测当前分支的结果，以减少分支带来的流水线中断和延迟。
16. 超标量技术：超标量技术是指在流水线中同时发射多条指令并行执行的技术。它通过增加处理器的功能单元和资源来实现多指令的同时执行，提高处理器的并行度和性能。
17. 超流水线技术：超流水线技术是指在流水线中将指令执行划分为更多的阶段，以进一步增加流水线的并行度和指令执行速度。超流水线技术可以提高处理器的频率和吞吐量，但也增加了数据冲突和控制相关的问题。
18. 动态多发射技术：动态多发射技术是一种超标量技术，通过在运行时动态判断和调度多条指令的并行执行，以最大程度地利用处理器的资源和性能。
19. 静态多发射技术：静态多发射技术是一种超标量技术，通过在编译时或静态阶段确定指令的并行性，并生成多个并行执行的指令流，以提高处理器的性能。
20. 同步中断：同步中断是指在程序执行过程中的明确指令中断请求，由程序自身发起并控制的中断操作。在同步中断发生时，处理器会根据指令的执行状态和优先级进行相应的中断处理。
21. 异步中断：异步中断是指在程序执行过程中由外部事件触发的中断请求，与程序的执行无关。异步中断通常由外部设备或事件发出，处理器在接收到中断请求后会暂停当前执行的指令，并跳转到中断处理程序执行相应的操作。

**7.2选择题**

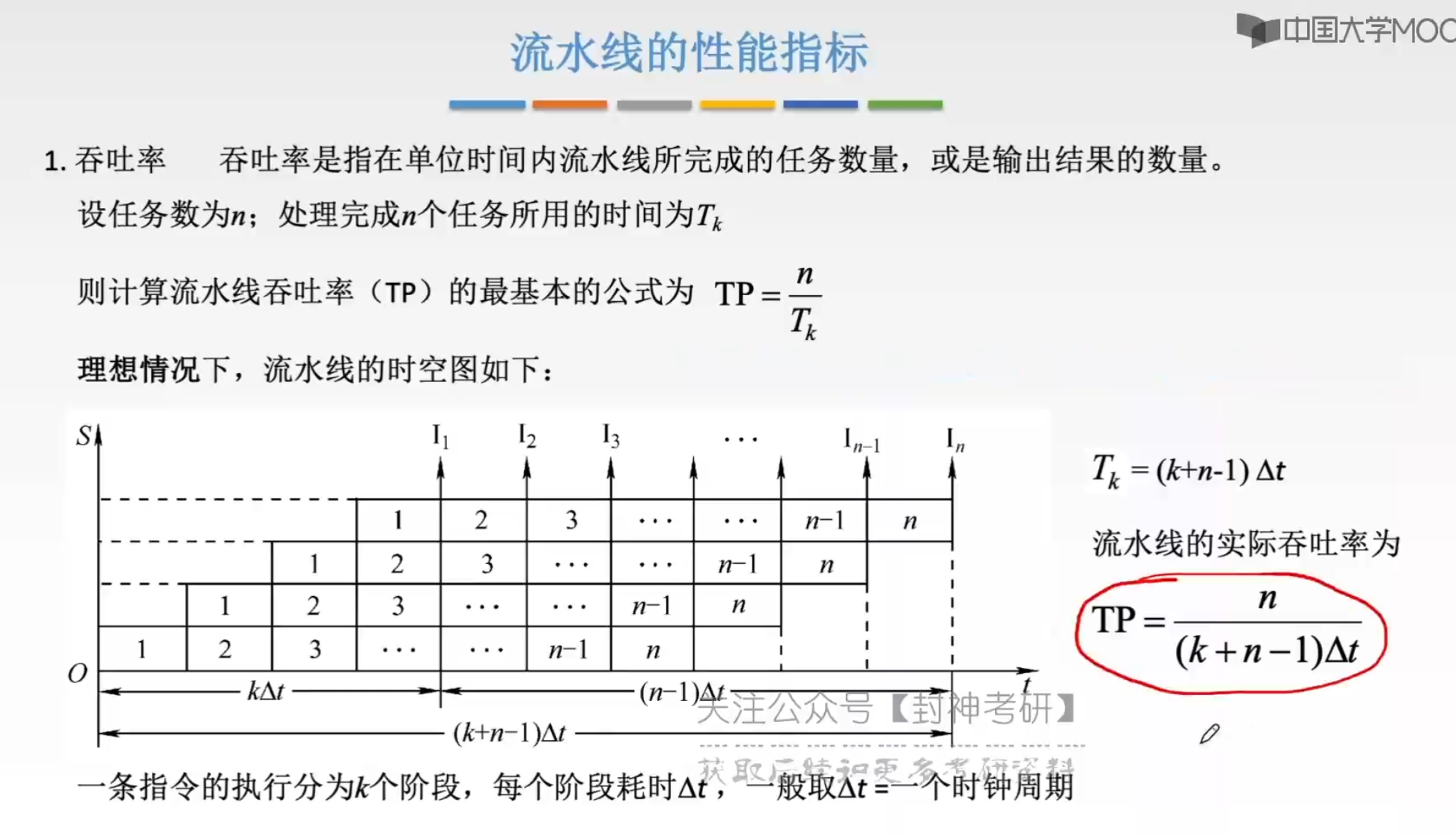
**( 1 )[2013]某 CPU 主频为 1.03GHz，采用 4 级指令流水线，每个流水段的执行需要 1 个时钟周期。假定 CPU 执行了 100 条指令，在其执行过程中，没有发生任何流水线阻塞，此时流水线的吞吐率为（ C ）。**

**A.0.25×10^9 条指令/秒**

**B.0.97×10^9 条指令/秒**

**C.1.0×10^9 条指令/秒**

**D.1.03 ×10^9 条指令/秒**



100/(103/1.03GHz)

**( 2 )[2009]某计算机的指令流水线由 4 个功能段组成，指令流经各功能段的时间(忽略各功能段之间的缓存时间)分别为 90ns、80ns、70ns 和 60ns，则该计算机的 CPU 时钟周期至少是（ A ）。**

**A.90ns B.80ns c.70ns D.60ns**

**( 3 ) [2018]若某计算机最复杂指令的执行需要完成 5 个子功能，分别由功能部件 A～E 实现，各功能部件所需时间分别为 80ps、50ps、50ps、70ps 和 50ps，采用流水线方式执行指令，流水段寄存器延迟时间为 20ps，则 CPU 时钟周期至少为（ D ）。**

**A.60ps B.70ps c. 80ps D.100ps**

流水线中的每个阶段需要等待一个时钟周期的时间，所以每个子功能的总执行时间等于该子功能的执行时间加上流水段寄存器的延迟时间。因此，各功能部件的总执行时间为：

* 功能部件 A 的总执行时间为 80ps + 20ps = 100ps。
* 功能部件 B 的总执行时间为 50ps + 20ps = 70ps。
* 功能部件 C 的总执行时间为 50ps + 20ps = 70ps。
* 功能部件 D 的总执行时间为 70ps + 20ps = 90ps。
* 功能部件 E 的总执行时间为 50ps + 20ps = 70ps。

由于流水线中的各个子功能可以并行执行，所以流水线的总执行时间等于所有子功能中执行时间最长的那个子功能的总执行时间。在这种情况下，功能部件 A 的总执行时间最长，为 100ps。

CPU 时钟周期至少等于流水线中最长子功能的总执行时间，即 100ps。

**(4)[2016]在无转发机制的 5 段基本流水线中，下列指令序列存在数据冲突的指令对是（ B ）。**

**I1: ADD Rl,R2,R3; (R2)+(R3)→R1**

**I2: ADD R5,R2,R4; (R2)+(R4)→R5**

**I3: ADD R4,R5,R3; (R5)+(R3)→R4**

**I4: ADD R5,R2,R6; (R2)+(R6)→R5**

**A.I1 和 I2 B.I2 和 I3 C.I2 和 I4 D.I3 和 I4**

**( 5 )[2019]在采用“取指、译码/取数、执行、访存、写回”5 段流水线的处理器中,执行如下指令序列,其中 s0、s1、s2、s3 和 t2 表示寄存器编号。**

**I1 : add s2,s1, s0 // R[s2]<-R[s1]+R[s0]**

**I2: load s3,0(t2) // R[s3]<-M[R[t2]+0]**

**I3: add s2,s2,s3 // R[s2]<-R[s2]+R[s3]**

**l4: store s2,0(t2) //M[R[t2]+0]<-R[s2]**

**下列指令对中，不存在数据冒险的是（ C ）。**

**A.I1 和 13 B.I2 和 I3 C.I2 和 I4 D.I3 和 I4**

要确定是否存在数据冒险（data hazard），我们需要检查指令之间的数据依赖关系。数据冒险分为三种类型：读后写（RAW），写后读（WAR）和写后写（WAW）。

现在我们逐个检查选项中的指令对：

A. I1和I3：在I3中，使用了I1中计算的结果。存在数据依赖关系（RAW依赖），因此存在数据冒险。

B. I2和I3：在I3中，使用了I2中加载的数据。存在数据依赖关系（RAW依赖），因此存在数据冒险。

C. I2和I4：在I4中，没有使用I2中的任何结果。不存在数据依赖关系，因此不存在数据冒险。

D. I3和I4：在I4中，使用了I3中计算的结果。存在数据依赖关系（RAW依赖），因此存在数据冒险。

综上所述，选项C中的指令对I2和I4不存在数据冒险。

**(6)[2010]下列选项中，不会引起指令流水线阻塞的是（ A ）。**

**A.数据旁路(转发) B.数据相关 C．条件转移 D.资源冲突**

引起流水线阻塞的三种相关：

结构相关 又称 资源相关：不同指令同时征用同一功能部件

数据相关：后继指令需要前面的运算结果

控制相关：转移指令

数据旁路技术（即是重定向技术）：其主要思想是不必待某条指令的执行结果送回到寄存器，再从寄存器中取出该结果，作为下一条指令的源操作数，而是直接将执行结果送到其他指令所需要的地方，这样可以使流水线不发生停顿。

**(7)[2011]下列给出的指令系统特点中，有利于实现指令流水线的是（ D ）。**

**l．指令格式规整且长度一致 Ⅱ．指令和数据按边界对齐存放**

**Ⅲ．只有 Load/Store 指令才能对操作数进行存储访问**

**A.仅 I、ll B.仅Ⅱ、Ⅲ C.仅Ⅰ、Ⅲ D. I、ll、Ⅲ**

Ⅰ. 指令格式规整且长度一致：规整的指令格式可以使流水线中的各个阶段的处理逻辑更加简单和统一，减少了指令译码和执行的复杂度。长度一致的指令可以使每个流水线阶段的处理时间相等，避免了流水线阻塞和冲突。

Ⅱ. 指令和数据按边界对齐存放：按边界对齐存放指令和数据可以简化内存访问和加载操作，减少了访存阶段的复杂性。边界对齐存放还可以提高内存访问的效率，减少指令和数据的读取时间。

Ⅲ. 只有 Load/Store 指令才能对操作数进行存储访问：这种特点通常与精简指令集（RISC）体系结构相关。Load/Store 指令的特点是只能从内存中加载数据到寄存器或将寄存器中的数据存储回内存，而其他指令只能在寄存器之间进行操作。这种特点简化了指令流水线的设计，减少了数据冒险和相关性问题。

综上所述，选项 D 中的特点 I、II 和 III 都有利于实现指令流水线。

**(8)[2017]下列关于指令流水线数据通路的叙述中,错误的是（ A ）。**

**A.包含生成控制信号的控制部件**

**B.包含算术逻辑运算部件（ALU)**

**C.包含通用寄存器组和取指部件**

**D.由组合逻辑电路和时序逻辑电路组合而成**

指令流水线数据通路通常由多个部件组成，包括取指部件、译码部件、执行部件、访存部件和写回部件等。这些部件负责执行指令流水线的各个阶段。

选项 B 提到了算术逻辑运算部件（ALU），这是指令流水线中执行部件的一部分，用于执行算术和逻辑运算。

选项 C 提到了通用寄存器组和取指部件，这是指令流水线中的一部分，用于存储和读取数据。

选项 D 提到了组合逻辑电路和时序逻辑电路，这是指令流水线中各个部件的基本构成，用于实现不同的功能和时序控制。

然而，选项 A 提到了包含生成控制信号的控制部件。CPU由数据通路和控制部件构成，因此数据通路不包含控制部件。因此，选项 A 是错误的叙述。

**(9) [2017]下列关于超标量流水线特性的叙述中,正确的是（ C ）。**

**I.能缩短流水线功能段的处理时间**

**Ⅱ.能在一个时钟周期内同时发射多条指令**

**Ⅲ．能结合动态调度技术提高指令执行并行性**

**A.仅Ⅱ B.仅 I、Ⅲ c.仅Ⅱ、Ⅲ D.I、Ⅱ和Ⅲ**

超标量流水线是一种流水线处理器的设计技术，具有以下特点：

Ⅱ. 能在一个时钟周期内同时发射多条指令：超标量流水线具有多个功能单元和执行单元，可以在一个时钟周期内并行执行多条指令。这样可以提高指令的执行速度和效率。

Ⅲ. 能结合动态调度技术提高指令执行并行性：超标量流水线通过动态调度技术，可以在运行时根据指令之间的依赖关系和资源可用性进行指令调度，以提高指令执行的并行性和效率。动态调度技术可以根据运行时的情况选择合适的指令并行执行，充分利用处理器资源。

Ⅰ. 能缩短流水线功能段的处理时间：这个叙述是错误的。超标量流水线并不能直接缩短流水线功能段的处理时间，它通过并行执行多条指令来提高整体的执行效率，但并不直接缩短功能段的处理时间。

综上所述，正确的叙述是 C.仅Ⅱ、Ⅲ。

**(10)[2020]下列给出的处理器类型中，理想情况下 CPI 为 1 的是（ B ）。**

**I.单周期 CPU II．多周期 CPU Ⅲ．基本流水线 CPU Ⅳ.超标量流水线 CPU**

**A.I 和Ⅱ B.I 和Ⅲ C.I、Ⅲ、Ⅳ D. Ⅲ、Ⅳ**

理想情况下 CPI（Cycles Per Instruction）为1表示每条指令都能在一个时钟周期内完成执行。根据给出的处理器类型，我们可以分析各个处理器类型的特点：

I. 单周期 CPU：在单周期 CPU 中，每条指令都需要一个时钟周期来完成执行，符合要求。

II. 多周期 CPU：在多周期 CPU 中，不同的指令可能需要不同数量的时钟周期来执行，因此 CPI 可能大于1，不符合要求。

III. 基本流水线 CPU：基本流水线 CPU 将指令执行划分为多个流水线阶段，每个阶段在一个时钟周期内完成。理想情况下，基本流水线 CPU 的 CPI 可以为1，每条指令都能在一个时钟周期内完成执行。

IV. 超标量流水线 CPU：超标量流水线 CPU 具有多个功能单元和执行单元，可以在一个时钟周期内同时发射多条指令，提高指令执行的并行性和效率。然而，由于资源竞争和相关性等因素，超标量流水线 CPU 的 CPI 可能大于1。

**7.4简述采用插入气泡方式解决数据冲突的主要过程**

答：  
插入气泡（Bubble Insertion）是一种用于解决数据冲突（Data Hazards）的技术，通常在流水线处理器中使用。该技术通过插入等待周期（Bubble）来处理数据冲突，以确保指令的正确执行顺序和数据的一致性。以下是采用插入气泡方式解决数据冲突的主要过程：

1. 数据冲突的检测：在流水线中，数据冲突通常发生在指令之间，其中一个指令需要使用前一条指令尚未完成的数据。流水线通过检测数据冲突来确定是否需要采取插入气泡的措施。
2. 发送气泡：如果检测到数据冲突，流水线会向冲突指令后的流水线阶段插入一个或多个等待周期，称为气泡。这些气泡会使流水线阶段暂停，并且不会产生任何结果。插入气泡的目的是为了等待前一条指令的数据准备完成。
3. 数据传递：一旦前一条指令的数据准备完成，流水线中的气泡就会被移除，并且数据可以继续流动到下一个阶段。这样，冲突指令就能够获得正确的数据，并继续执行。
4. 控制流程的处理：插入气泡可能会影响指令的控制流程，例如条件分支指令（如if语句）。在这种情况下，需要根据前一条指令的条件结果来判断是否需要采取分支。如果需要插入气泡来等待条件结果，那么需要确保分支指令在正确的条件下执行。
5. 性能影响：插入气泡会导致流水线的停顿和延迟，从而降低了指令的执行速度和流水线的效率。因此，尽量减少数据冲突和插入气泡的发生是优化流水线性能的关键。

总之，插入气泡是一种通过暂停流水线阶段并等待前一条指令的数据准备完成来解决数据冲突的方法。它确保了指令的正确执行顺序和数据的一致性，但也会对流水线的性能产生一定的影响。因此，在设计流水线处理器时，需要综合考虑数据冲突的减少和性能的平衡。

**7.5简述采用重定向方式解决数据冲突的主要冲突**

答：采用重定向（Forwarding）方式解决数据冲突是一种常用的技术，用于在流水线处理器中解决数据相关性导致的冲突。该技术通过将数据从产生冲突的指令直接传递给需要使用该数据的指令，避免了气泡插入和流水线暂停，以提高处理器的性能。以下是采用重定向方式解决数据冲突的主要过程：

1. 数据冲突的检测：在流水线中，数据冲突通常发生在指令之间，其中一个指令需要使用前一条指令尚未完成的数据。流水线会检测数据冲突，以确定是否需要采取重定向的措施。
2. 数据传递：如果检测到数据冲突，流水线会将数据从产生冲突的指令传递给需要使用该数据的指令，而无需等待数据写回到寄存器文档。这种数据传递可以通过在流水线的不同阶段之间设置特殊的数据通路来实现。
3. 数据依赖判断：在进行数据传递之前，需要判断数据冲突的类型。如果是读后写（RAW）冲突，即当前指令需要使用前一条指令的计算结果，那么可以通过重定向将数据直接传递给当前指令。如果是写后写（WAW）或写后读（WAR）冲突，重定向方式可能无法解决，需要采取其他措施，如插入气泡或使用乱序执行技术。
4. 优先级处理：在进行数据传递时，需要根据指令的优先级来决定数据应该传递给哪个指令。通常情况下，优先级较高的指令会优先获取数据，以确保指令执行的正确性。
5. 控制流程的处理：重定向方式还可以用于解决条件分支指令（如if语句）的控制流程问题。如果条件分支指令依赖于前一条指令的计算结果，可以通过重定向方式将结果传递给条件分支指令，以便正确地确定分支的方向。

通过采用重定向方式解决数据冲突，可以有效减少流水线的停顿和延迟，提高处理器的性能。然而，重定向可能涉及更复杂的硬件设计和数据通路，并且需要细致处理优先级和控制流程等问题，以确保正确和可靠的执行。

**7.6流水线方式缩短的是指令的执行时间还是程序的执行时间？**

答：流水线方式主要是通过将处理器的工作分为多个阶段并同时处理多条指令，以缩短指令的执行时间，而不是程序的执行时间。

在流水线处理器中，每个指令在流水线的各个阶段中并行执行，使得多条指令可以同时在不同的阶段进行处理。这样可以提高处理器的吞吐量，即单位时间内完成的指令数量。通过流水线方式，指令的执行时间可以大大缩短，从而提高了处理器的性能。

然而，尽管指令的执行时间被缩短了，但整个程序的执行时间不一定会减少。这是因为流水线中的每个阶段都需要一定的时间来完成处理，而指令在流水线中的执行速度受到最慢的阶段所限制。此外，由于数据冲突、分支预测错误等因素的存在，可能会导致流水线的停顿和延迟，进而影响整个程序的执行时间。

因此，流水线方式主要是缩短指令的执行时间，提高处理器的吞吐量和性能，而对于整个程序的执行时间，还需要考虑其他因素的影响。

**7.8如果采用气泡流水线执行下述程序，请给出类似图 7.18 所示的流水线时空图。注意时空图中最后一个时钟周期第 5 条指令进入 ID 段。**

addi $s0,$s0,4

lw $s1,($s0)

add $s2,$s2,$s1

and $s3,$s1,$s2

sub $s4,$s2,$s2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CLKs | 取值IF | 译码ID | 执行EX | 访存MEM | 写回WB |
| **1** | **addi $s0, $s0,4** |  |  |  |  |
| **2** | **lw $s1, ($s0)** | **addi $s0,$s0,4** |  |  |  |
| **3** | **add $s2,$s2,$s1** | **lw $s1, ($s0)** | **addi $s0,$s0,4** |  |  |
| **4** | **add $s2,$s2,$s1** | **lw $s1, ($s0)** | **Bubble** | **addi $s0,$s0,4** |  |
| **5** | **add $s2,$s2,$s1** | **lw $s1, ($s0)** | **Bubble** | **Bubble** | **addi $s0,$s0,4** |
| **6** | **add $s3,$s1,$s2** | **add $s2,$s2,$s1** | **lw $s1, ($s0)** | **Bubble** | **Bubble** |
| **7** | **add $s3,$s1,$s2** | **add $s2,$s2,$s1** | **Bubble** | **lw $s1, ($s0)** | **Bubble** |
| **8** | **sub $s4,$s2,$s2** | **add $s2,$s2,$s1** | **Bubble** | **Bubble** | **lw $s1, ($s0)** |
| **9** | **Next Instr** | **sub $s4,$s2,$s2** | **lw $s1, ($s0)** | **Bubble** | **Bubble** |

**有问题。**

**7.9如果采用重定向流水线执行 7.8 中程序，请给出类似图 7.18 所示的流水线时空图。注意时空图中最后一个时钟周期第 5 条指令进入 ID 段。**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CLKs | 取值IF | 译码ID | 执行EX | 访存MEM | 写回WB |
| **1** | **addi $s0, $s0,4** |  |  |  |  |
| **2** | **lw $s1, ($s0)** | **addi $s0,$s0,4** |  |  |  |
| **3** | **add $s2,$s2,$s1** | **lw $s1, ($s0)** | **addi $s0,$s0,4** |  |  |
| **4** | **add $s3,$s1,$s2** | **add $s2,$s2,$s1** | **lw $s1, ($s0)** | **addi $s0,$s0,4** |  |
| **5** | **add $s3,$s1,$s2** | **add $s2,$s2,$s1** | **Bubble** | **lw $s1, ($s0)** | **addi $s0,$s0,4** |
| **6** | **sub $s4,$s2,$s2** | **add $s3,$s1,$s2** | **add $s2,$s2,$s1** | **Bubble** | **lw $s1, ($s0)** |
| **7** | **Next Instr** | **sub $s4,$s2,$s2** | **add $s3,$s1,$s2** | **add $s2,$s2,$s1** | **Bubble** |