计算机系统结构

第七讲流水线冲突（2）

冯丹

武汉光电国家研究中心

华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

1.控制冲突

- 起因
  - 执行分支指令的结果有两种
    - 分支成功：PC值改变为分支转移的目标地址。在条件判定和转移地址计算都完成后，才改变PC值。
    - 不成功或者失败：PC的值保持正常递增，指向顺序的下一条指令。
- 控制冲突
  - 分支延迟：分支指令引起的延迟
- 最简单的处理方法
  - “冻结”或者“排空”流水线
  - 前述5段流水线中，改变PC值是在MEM段进行的。给流水线带来了3个时钟周期的延迟

2.控制冲突实例

简单处理分支指令：分支成功的情况

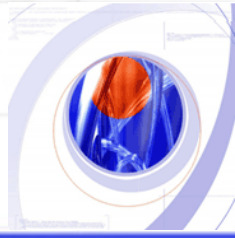
分支指令	IF	ID	EX	MEM	WB					
分支目标指令		IF	stall	stall	IF	ID	EX	MEM	WB	
分支目标指令+1						IF	ID	EX	MEM	WB
分支目标指令+2							IF	ID	EX	MEM
分支目标指令+3								IF	ID	EX

3.控制冲突的重要性及基本解决方案

- 分支指令在目标代码中出现的频度
  - 每3～4条指令就有一条是分支指令。
    - 如果：分支指令出现的频度是30%，且流水线理想 CPI＝1
    - 那么：流水线的实际 CPI ＝ 1.9
- 基本解决方案
  - 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。
  - 以下的讨论中，我们假设：这两步工作被提前到ID段完成，即分支指令是在ID段的末尾执行完成，所带来的分支延迟为一个时钟周期。
  - 改进后，上例中的实际CPI=1.3

4.进一步改进——基于编译器的软件方法

- 共同点
  - 对分支的处理方法在程序的执行过程中始终是不变的，是静态的。
  - 要么总是预测分支成功，要么总是预测分支失败基本解决方案。
- 分类
  - 预测分支失败
  - 预测分支成功
  - 延迟分支



5-1.预测分支失败

- 方法
  - 允许分支指令后的指令继续在流水线中流动，就好象什么都没发生似的；
  - 若确定分支失败，将分支指令看作是一条普通指令，流水线正常流动；
  - 若确定分支成功，流水线就把在分支指令之后取出的所有指令转化为空操作，并按分支目地重新取指令执行。

5-2.预测分支失败实例

– 两个例子，预测正确与预测错误

DLX流水线对分支的处理过程

分支指令 i(失败)	IF	ID	EX	MEM	WB				
指令 i+1		IF	ID	EX	MEM	WB			
指令 i+2			IF	ID	EX	MEM	WB		
指令 i+3				IF	ID	EX	MEM	WB	
指令 i+4					IF	ID	EX	MEM	WB

分支指令 i(成功)	IF	ID	EX	MEM	WB				
指令 i+1		IF	ID	idle	idle	idle			
分支目标 j			IF	ID	EX	MEM	WB		
分支目标 j+1				IF	ID	EX	MEM	WB	
分支目标 j+2					IF	ID	EX	MEM	WB

要点

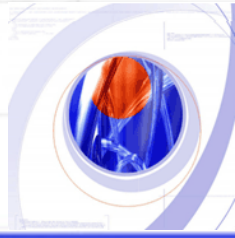
分支结果出来之前不能改变处理机的状态，以便一旦猜错时，处理机能够回退到原先的状态。

效果

预测正确时，能够减少一个时钟周期的延迟

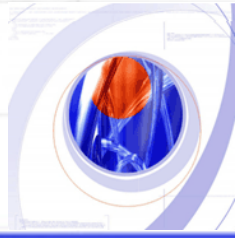
6.预测分支成功

- 方法
  - 假设分支转移成功，并从分支目标地址处取指令执行。
- 要点
  - 起作用的前题：先知道分支目标地址，后知道分支是否成功。
- 效果
  - 前述5段流水线中，这种方法没有任何好处。



7-1.延迟分支（1）

- 方法
  - 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，都要按顺序执行延迟槽中的指令。
- 效果
  - 无论分支成功还是失败都能够减少（掩盖）一个时钟周期的延迟



7-2.延迟分支实例

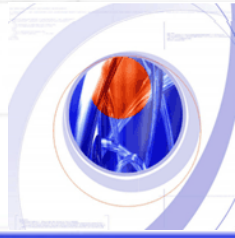
– 在下面的两个例子中，无论是分支成功，还是分支失败，都能起效

分支失败	分支指令i	IF	ID	EX	MEM	WB				
	延迟槽指令 i+1		IF	ID	EX	MEM	WB			
	指令 i+2			IF	ID	EX	MEM	WB		
	指令 i+3				IF	ID	EX	MEM	WB	
	指令 i+4					IF	ID	EX	MEM	WB

分支成功	分支指令i	IF	ID	EX	MEM	WB				
	延迟槽指令 i+1		IF	ID	EX	MEM	WB			
	分支目标指令j			IF	ID	EX	MEM	WB		
	分支目标指令j+1				IF	ID	EX	MEM	WB	
	分支目标指令j+2					IF	ID	EX	MEM	WB

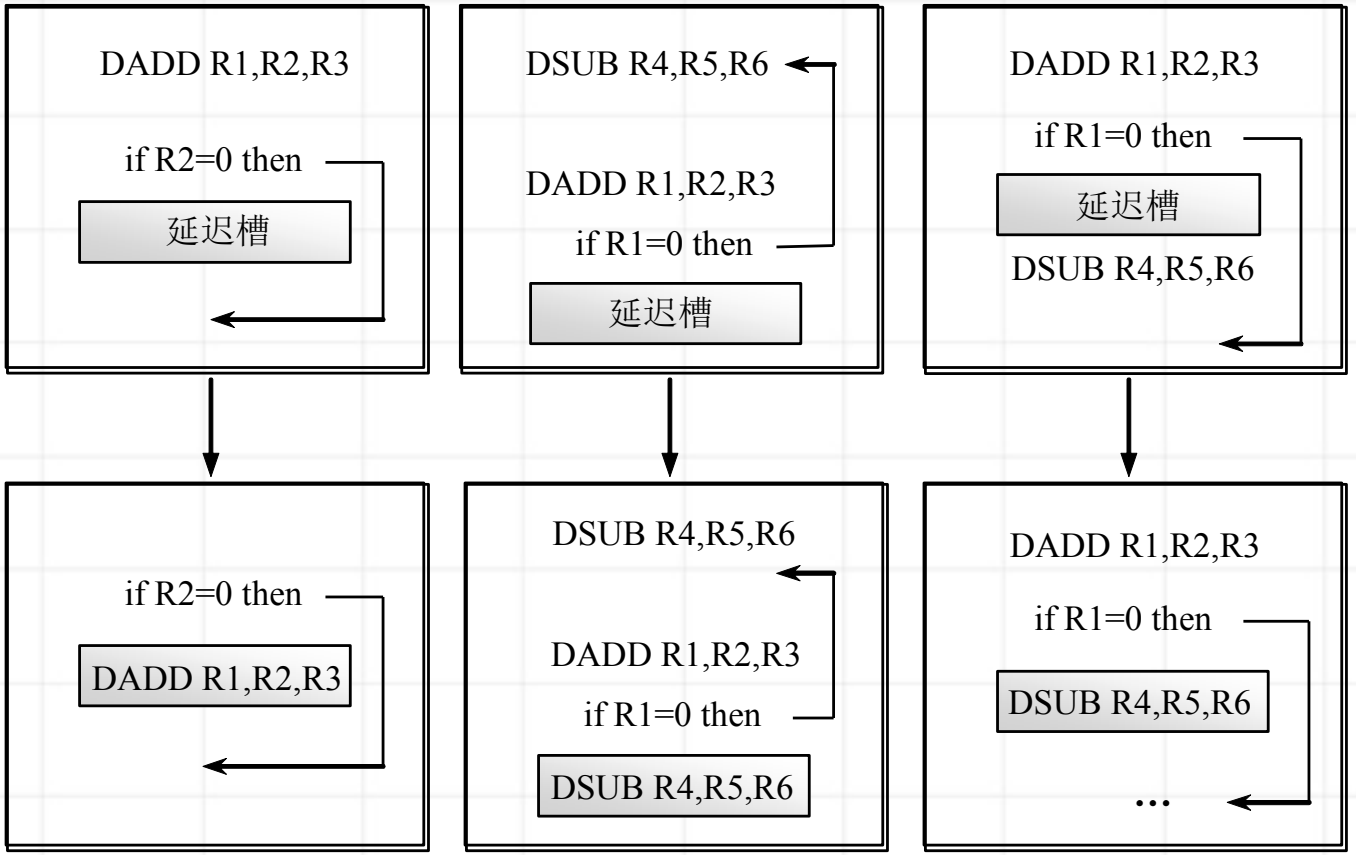
7-3.延迟分支（2）

- 要点
  - 在延迟槽中放入有用的指令，由编译器完成。能否带来好处取决于编译器能否把有用的指令调度到延迟槽中。
- 三个子类
  - 从前调度
  - 从目标处调度
  - 从失败处调度



7-4.调度方法实例

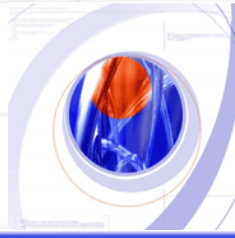
– 三个例子：从前调度，从目标处调度，从失败处调度



(a) 从前调度　(b) 从目标处调度　(c) 从失败处调度

7-5.约束与进一步改进

- 分支延迟受到两个方面的限制
  - 在延迟槽中放入有用的指令，由编译器完成。
  - 能否带来好处取决于编译器能否把有用的指令调度到延迟槽中。
- 进一步改进：分支取消机制（处理预测错误的情况）
  - 当分支的实际执行方向和事先所预测的一样时，执行分支延迟槽中的指令，否则就将分支延迟槽中的指令转化成一个空操作。



7-6.分支取消实例

– 例子：预测错误后的取消与预测正确的情况

分支失败	分支指令i	IF	ID	EX	MEM	WB				
	延迟槽指令 i+1		IF	idle	idle	idle	idle			
	指令 i+2			IF	ID	EX	MEM	WB		
	指令 i+3				IF	ID	EX	MEM	WB	
	指令 i+4					IF	ID	EX	MEM	WB

分支成功	分支指令i	IF	ID	EX	MEM	WB				
	延迟槽指令 i+1		IF	ID	EX	MEM	WB			
	分支目标指令j			IF	ID	EX	MEM	WB		
	分支目标指令j+1				IF	ID	EX	MEM	WB	
	分支目标指令j+2					IF	ID	EX	MEM	WB

预测分支成功的情况下，分支取消机制的执行情况

谢谢大家