计算机体系结构

周学海
xhzhou@ustc.edu.cn
0551-63492149
中国科学技术大学

Review

- **流水线技术并不能提高单个任务的执行效率，它可以提高整个系统的吞吐率**
  - 多个任务同时执行，但使用不同的资源
- **流水线性能分析：吞吐率、加速比、效率**
  - 流水线中的瓶颈——最慢的那一段
  - 其潜在的加速比 = 流水线的级数
  - 流水段所需时间不均衡将降低加速比
  - 流水线存在装入时间和排空时间，使得加速比降低
- **由于存在相关问题，会导致流水线停顿**
  - 结构相关、数据相关和控制相关

Review 相关的种类

- **相关会影响流水线性能**
- **结构相关: 由于争用资源而引起的**
  - 解决办法: 等待　增加（或拆分）资源
- **数据相关：两条指令访问相同的数据而引起的**
  - 类型：
    - RAW
    - WAR，　WAW
  - 解决办法：
    - 硬件：定向（前递/前推）技术（forwarding)
    - 软件：指令级调度
- **控制相关：由于控制类指令引起的**
  - 减少性能损失的基本方法：
    - 等待　(通过修改数据通路减少stall)
    - 分支预测
    - 延迟槽
- **陷阱与中断**

review：流水线性能分析

$$TP_{\max} = \frac{1}{\max\{\Delta t_i\}}$$

$$TP = \frac{n}{\sum_{i=1}^{k} \Delta t_i + (n-1)\Delta t_j}$$

$$S = \frac{n \cdot \sum_{i=1}^{k} \Delta t_i}{\sum_{i=1}^{k} \Delta t_i + (n-1)\Delta t_j}$$

$$E = \frac{n \cdot \sum_{i=1}^{k} \Delta t_j}{k \cdot [\sum_{i=1}^{k} \Delta t_i + (n-1)\Delta t_j]}$$

$$E = TP \cdot \frac{\sum_{i=1}^{k} \Delta t_i}{k}$$

吞吐率、加速比、效率之间的关系

流水线技术应用的难度何在？ ：相关问题

相关的类型：

结构相关，控制相关，以及

数据相关（RAW, WAR, WAW）

流水线的加速比计算

$$CPI_{pipelined} = \text{Ideal CPI} + \text{Average Stall cycles per Inst}$$

$$\text{Speedup} = \frac{\text{Ideal CPI} \times \text{Pipeline depth}}{\text{Ideal CPI} + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{unpipelined}}{\text{Cycle Time}_{pipelined}}$$

For simple RISC pipeline, CPI = 1:

$$\text{Speedup} = \frac{\text{Pipeline depth}}{1 + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{unpipelined}}{\text{Cycle Time}_{pipelined}}$$

结构相关对性能的影响

- **例如: 如果每条指令平均访存1.3 次，而每个时钟周期只能访存一次，那么**
  - 在其他资源100%利用的前提下，平均 CPI ≥ 1.3

例如： Dual-port vs. Single-port

- **机器A: Dual ported memory（“Harvard Architecture”）**
- **机器B: Single ported memory**
- **存在结构相关的机器B的时钟频率是机器A的时钟频率的1.05倍**
- **Ideal CPI = 1**
- **在机器B中load指令会引起结构相关，所执行的指令中Loads指令占 40%**

Average instruction time = CPI * Clock cycle time

无结构相关的机器A：

Average Instruction time = Clock cycle time

存在结构相关的机器B：

Average Instruction time = (1+0.4*1) * clock cycle time /1.05

= 1.3 * clock cycle time

评估减少分支策略的效果

$$\text{Pipeline speedup} = \frac{\text{Pipeline depth}}{1 + \text{Branch frequency} \times \text{Branch penalty}}$$

Scheduling scheme	*Branch penalty*	*CPI*	*speedup v. unpipelined*	*speedup v. stall*
Stall pipeline	3	1.42	3.5	1.0
Predict taken	1	1.14	4.4	1.26
Predict not taken	1	1.09	4.5	1.29
Delayed branch	0.5	1.07	4.6	1.31

1.14 = 1 + 1*14%*100%

1.09 = 1+1*14%*65%

1.07 = 1+ 0.5*14%

Conditional & Unconditional = 14%, 65% change PC

Chapter3 基本流水线

- **3.1 基本流水线**
  - 流水线的基本概念
  - 流水线中的相关
  - 流水线的性能分析
- **3.2 基本流水线的扩展**
  - 异常处理
  - 多周期操作处理

3.2 基本流水线的扩展

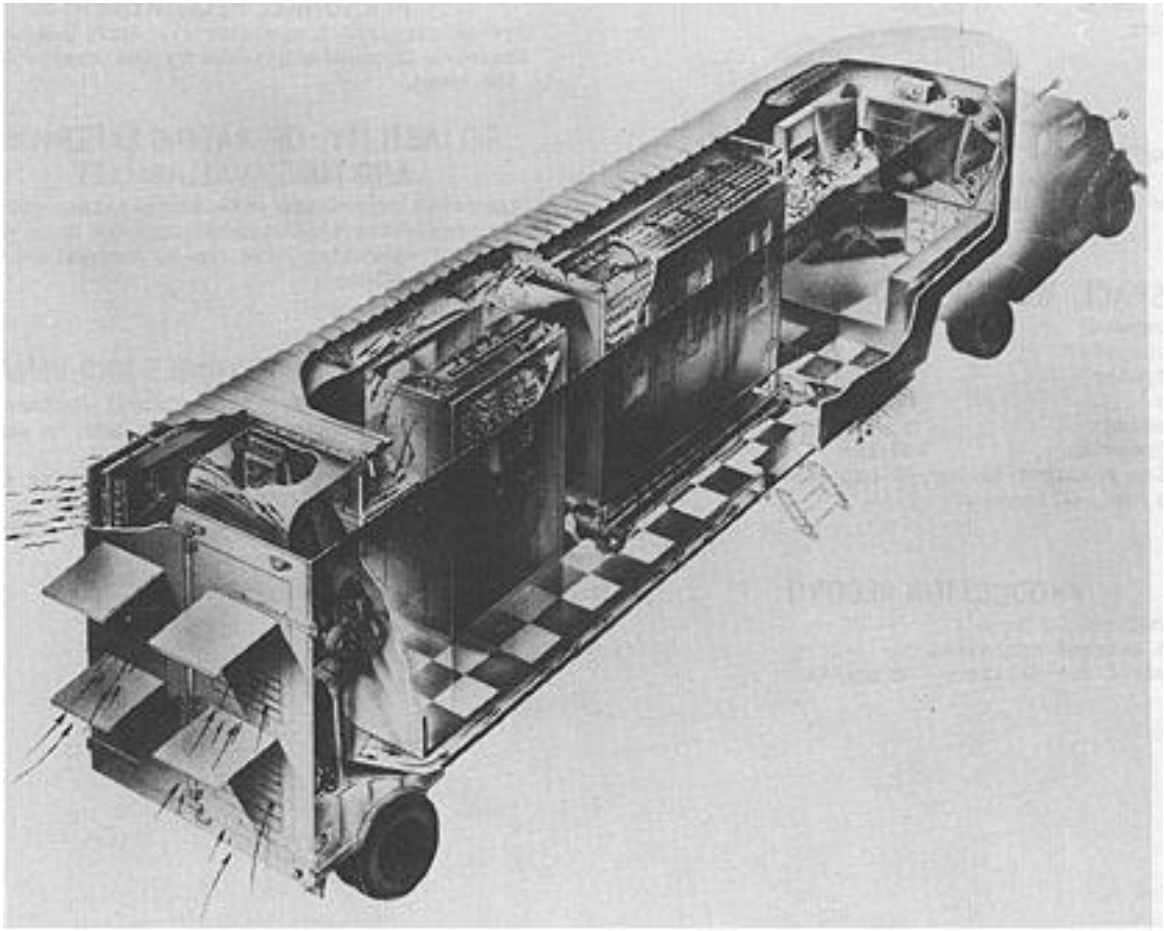
Exception、Interrupt和Trap

- **广义上的异常机制：处理器在执行指令流的过程中遇到异常事件而中止执行当前程序，转而去处理异常事件**
- **异常事件：内部异常事件和外部请求事件**
- **根据异常事件的来源可分为：**
  - 内部异常（Exception）：处理器内部事件或程序执行过程中的事件引起的异常
    - 例如：硬件故障、Page fault、arithmetic underflow等
  - 中断（Interrupt）：处理器外部请求事件引起的异常
- **根据异常事件发生在程序执行中的位置，可分为：**
  - 同步异常：在每次执行时异常事件发生在相同位置。即异常原因可精确定位到某指令
  - 异步异常：在每次执行时异常事件发生在不同位置。如定时器过期、I/O设备中相关事件的信号、硬件故障、电源故障等等，不能够被精确定位到某条指令
- **根据异常处理后是否能够精确地返回到引起异常的指令，可分为：**
  - 精确异常和非精确异常
- **陷阱（Trap）:**因执行指令引起异常（Exception）情况而被迫将控制权移交给管理员环境（监控程序，运行级别的变化）
  - Not all exceptions cause traps (c.f. IEEE 754 floating-point standard)
- **中断响应：响应（处理）外部请求事件**，导致控制权转移到管理员环境
- **陷阱和中断响应通常由同样的流水线机制处理**

异常处理的发展历史

- **第一个带有陷阱（traps）的系统是Univac-I （1951年)**
  - 算术运算溢出有两种选择
    - 1.在地址0处触发一个两条指令的修复例程的执行 或者
    - 2.根据程序员的选择，使计算机停止
  - 在后来的Univac 1103, 1955, 增加了外部中断机制
    - 用于实时采集风洞数据
- **第一个带有I/O 中断的系统是DYSEAC（1954)**
  - 带有两个程序计数器，并且I/O信号引起这两个PC间的切换
  - 它也是第一个带有DMA (direct memory access by I/O device)的系统
  - 同时，也是第一台移动计算机 (两台半挂牵引车, 12 tons + 8 tons)

DYSEAC, first mobile computer!

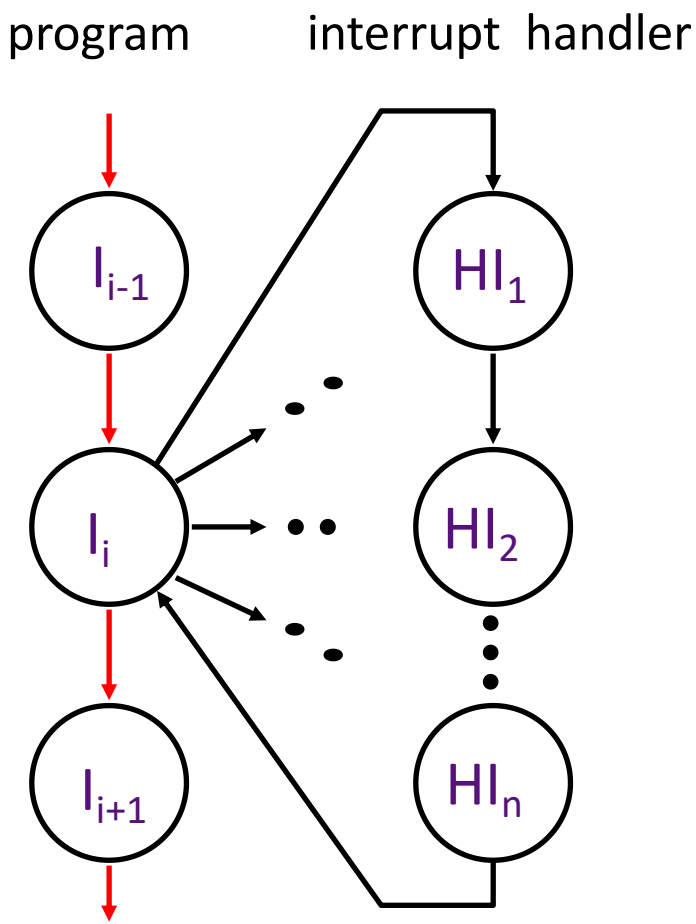


- Carried in two tractor trailers, 12 tons + 8 tons
- Built for US Army Signal Corps

[Courtesy Mark Smotherman]

异步中断: 改变正常的控制流

- I/O设备通过发出中断请求信号来请求处理
- 当处理器决定响应该中断时
  - 停止当前指令(Ii)的执行，执行完当前指令前面的指令（执行完 Ii-1）(精确中断)
  - 将Ii指令的PC值保存到专门寄存器(EPC)中
  - 关中断并将控制转移到以监控模式运行的指定的中断处理程序



需要由另一个（系统）程序处理的外部或内部事件。从程序的角度来看，事件通常是意外的或罕见的。

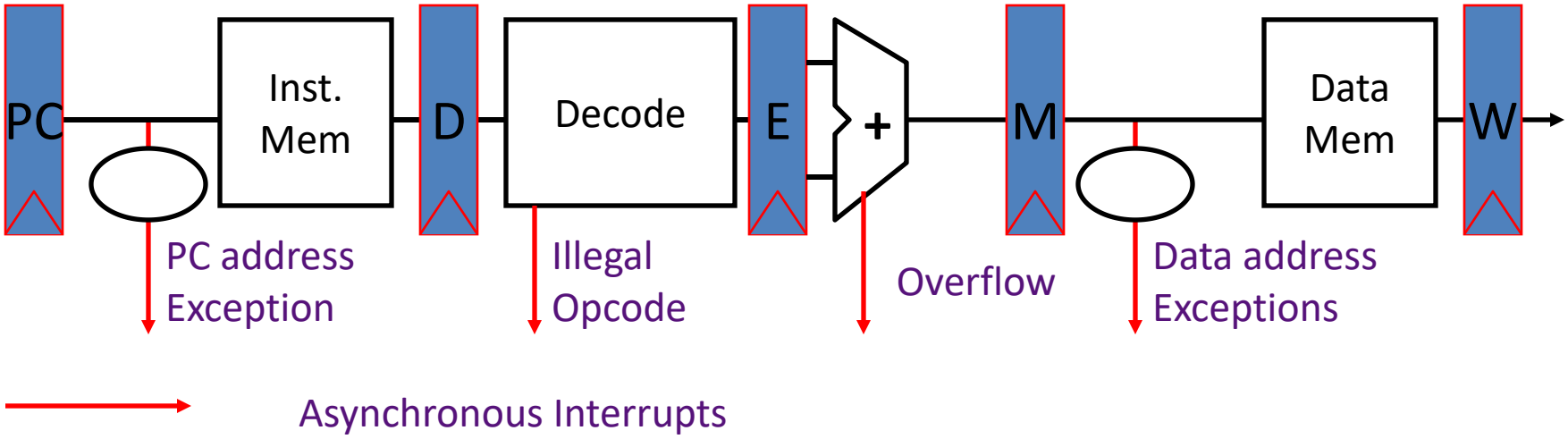
Interrupt Handler

- **允许嵌套中断时，在开中断之前需要保存EPC ⇒**
  - 需要执行指令将EPC 存入 GPRs（或堆栈）
  - 至少在保存EPC之前，需要一种方法来暂时屏蔽进一步的中断
- **需要读取记录中断原因的状态寄存器**
- **使用专门的间接跳转指令ERET (return-from-environment) 返回**
  - 开中断
  - 将处理器恢复到用户模式
  - 恢复硬件状态和控制状态

Synchronous Trap

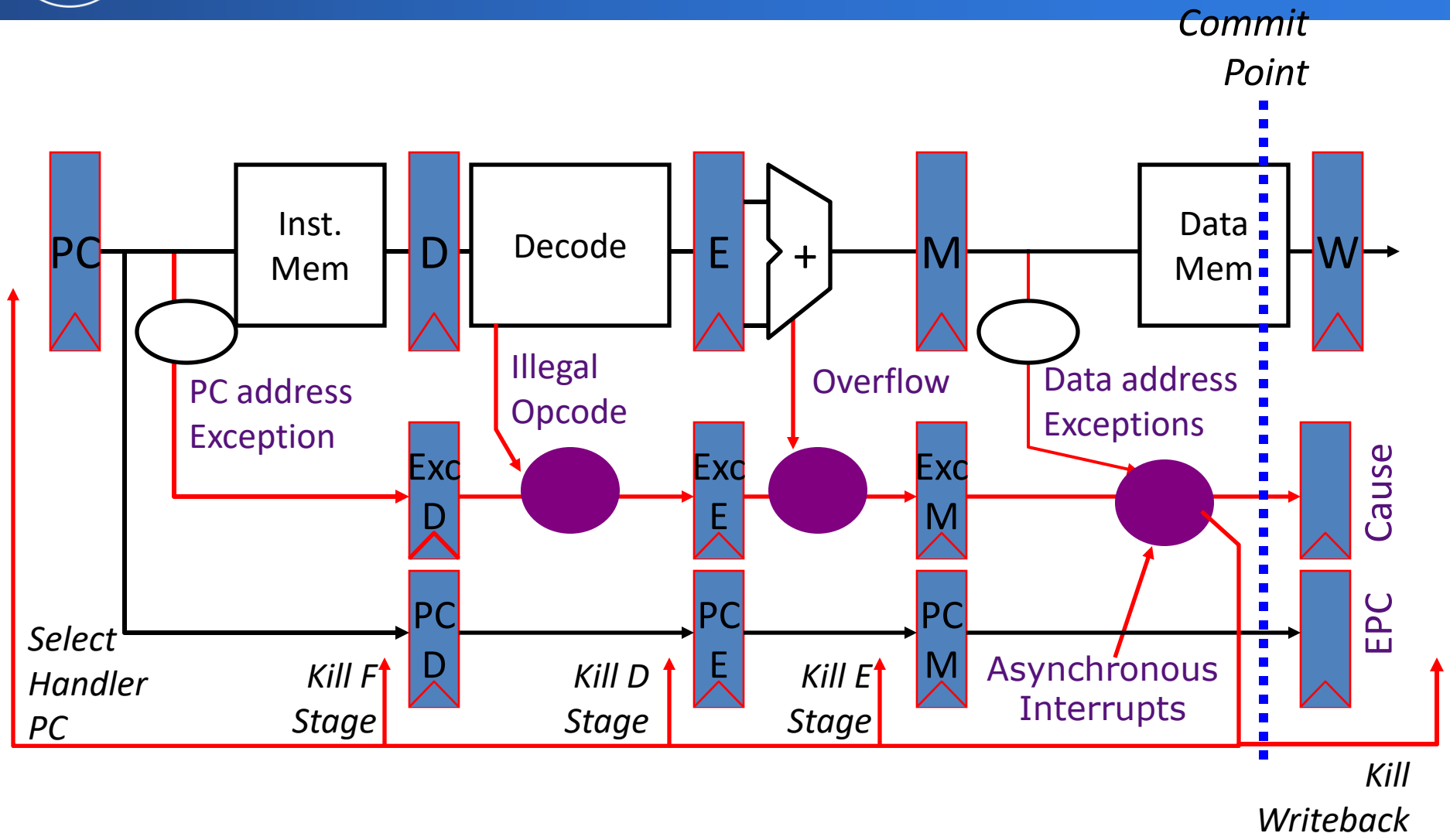
- **同步陷阱是由特定指令执行的异常引起的**
- **通常，该指令无法完成，需要在处理异常之后重新启动**
  - 需要撤消一个或多个部分执行的指令的效果
- **在系统调用(陷阱)的情况下，该特殊的跳转指令被认为已经完成**
  - 一种特殊的跳转指令，涉及到切换到特权模式的操作

Exception Handling 5-Stage Pipeline



- 如何处理不同流水段的多个并发异常?
- 如何以及在哪里处理外部异步中断?

Recap: Exception Handling 5-Stage Pipeline



Exception Handling 5-Stage Pipeline

- 在流水线中将异常标志保留到提交阶段

- **在提交阶段并入异步中断请求 (抑制其他中断)**
- **针对某一给定的指令，早期流水阶段中的异常抑制该指令的后续异常**
- **如果提交时检测到异常:**
  - 更新异常原因及EPC寄存器
  - 终止所有流水段,
  - 将异常处理程序的地址送入PC寄存器，以跳转到处理程序中执行

异常的推测

- **预测机制**
  - 异常总是比较少的，所以简单地预测为没有异常 通常是大概率事件
- **检查预测机制**
  - 在指令执行的最后阶段进行异常检测
  - 采用专门的硬件用于检测各种类型的异常
- **恢复执行机制**
  - 仅在提交阶段改变机器状态，因此可以在发生异常后丢弃部分执行的指令
  - 刷新流水线后启动异常处理程序
- **定向路径机制允许后续的指令使用没有提交的指令结果**

3.2 基本流水线的扩展

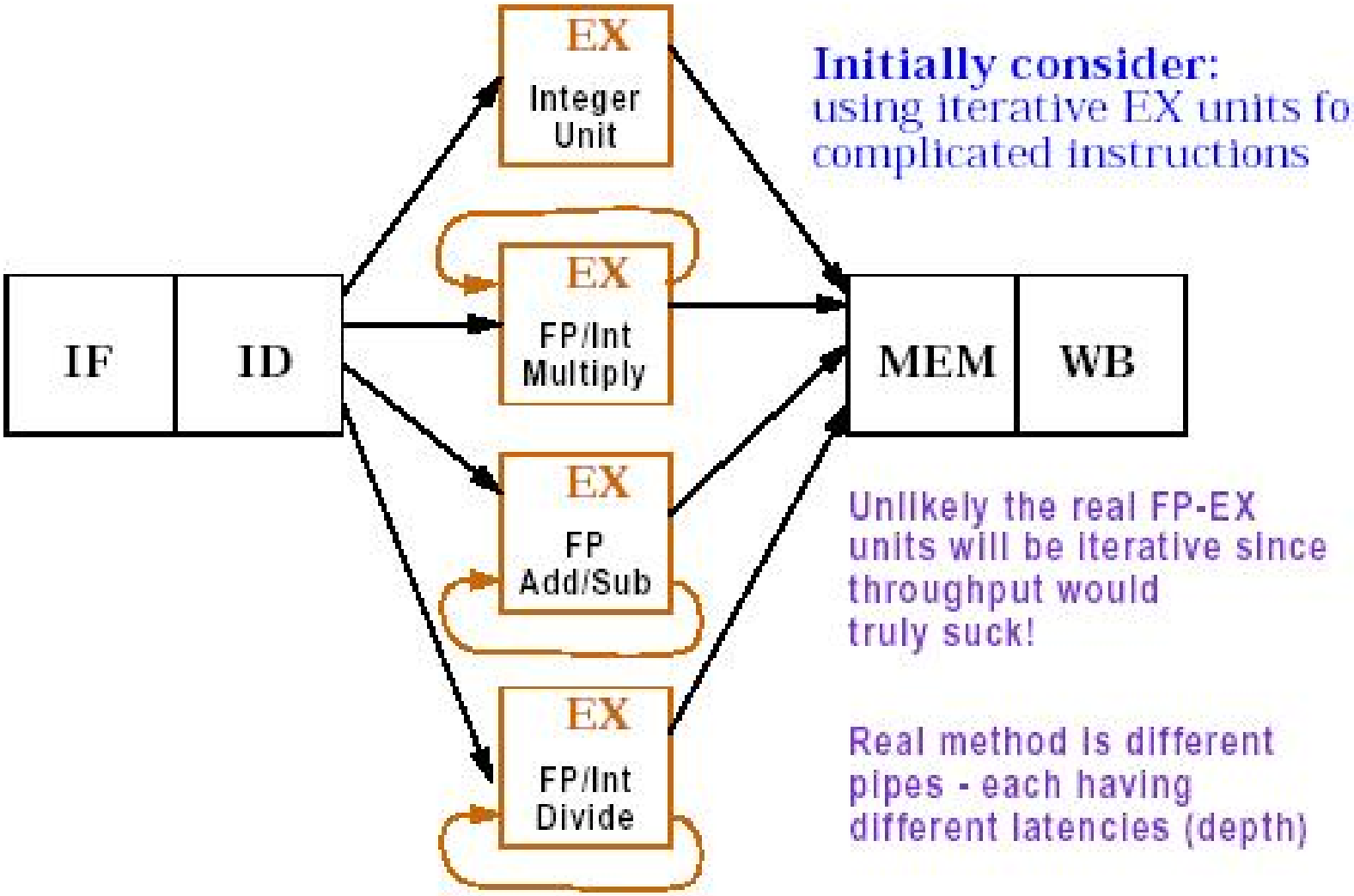
多周期操作的处理

- **问题**
  - 浮点操作在1～2个cycles完成是不现实的，一般要花费较长时间
  - 在MIPS中如何处理?
- **在1到2个cycles时间内完成的处理方法**
  - 采用较慢的时钟源，或
  - 在FP部件中延迟其EX段
- **现假设FP指令与整数指令采用相同的流水线，那么**
  - EX 段需要循环多次来完成FP操作，循环次数取决于操作类型
  - 有多个FP功能部件，如果发射出的指令导致结构或数据相关，需暂停

对MIPS的扩充

- **四个功能部件**
- **Integer 部件处理**
  - Loads, Store, Integer ALU操作和Branch
- **FP/Integer 乘法部件**
  - 处理浮点数和整数乘法
- **FP加法器**
  - 处理FP加，减和类型转换
- **FP/Integer除法部件**
  - 处理浮点数和整数除法
- **这些功能部件未流水化**

扩展的MIPS流水线

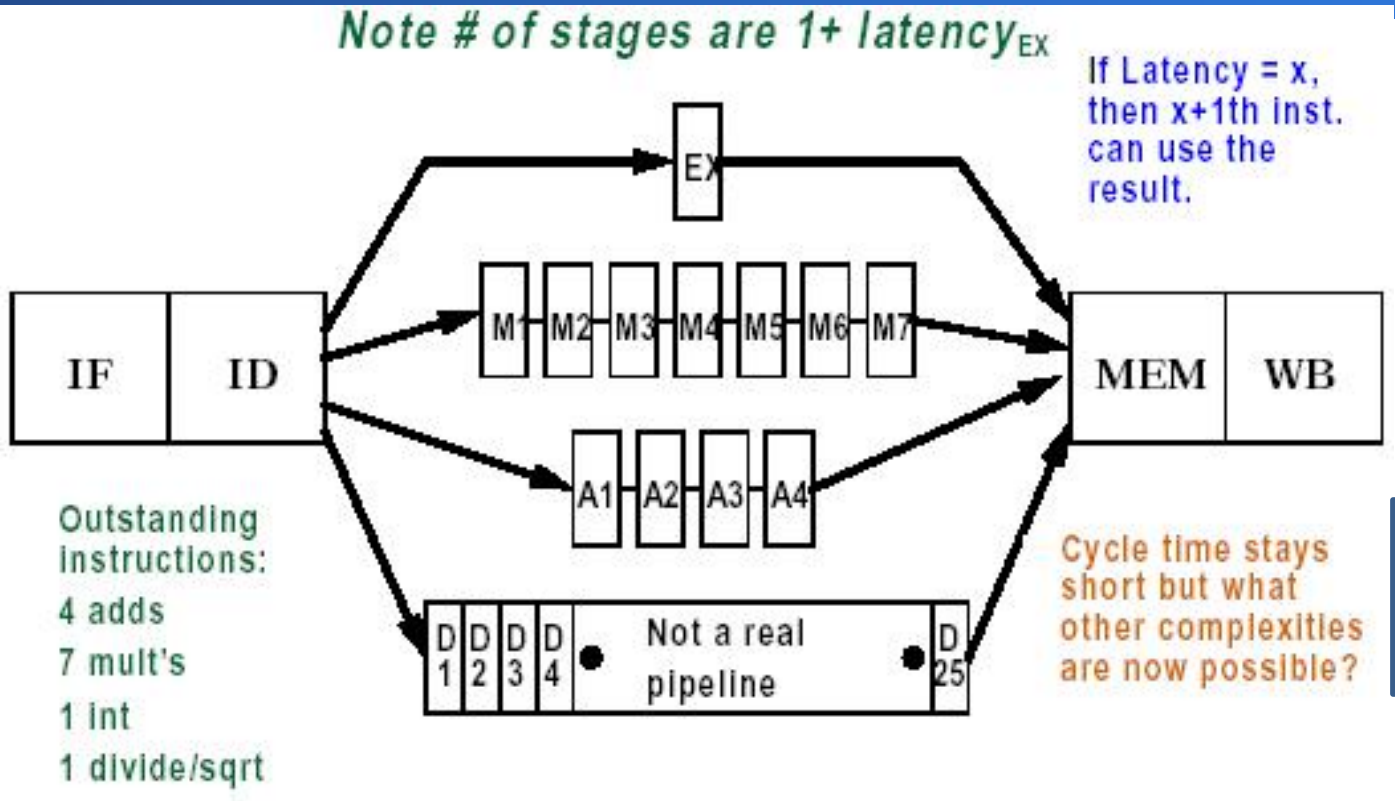

EX
Integer Unit
EX
FP/Int Multiply
EX
FP Add/Sub
EX
FP/Int Divide
IF
ID
MEM
WB
Initially consider:
using iterative EX units for complicated instructions
Unlikely the real FP-EX units will be iterative since throughput would truly suck!
Real method is different pipes - each having different latencies (depth)

Latency & Repeat Interval

- **延时(Latency)**
  - 定义1：完成某一操作所需的cycle数
  - **定义2：使用当前指令所产生结果的指令与当前指令间的最小间隔周期数**
- **循环间隔（Repeat/Initiation interval)**
  - 发射相同类型的操作所需的间隔周期数
- **EX部件流水化的新的MIPS**

Function Unit	Latency	Repeat Interval
Integer ALU	**0**	**1**
Data Memory (Integer and FP loads(**1 less for store latency**))	**1**	**1**
FP Add	**3**	**1**
FP multiply	**6**	**1**
FP Divide (also integer divide and FP sqrt)	**24**	**25**

将部分执行部件流水化后的MIPS流水线



部件流水化后，执行阶段周期数不同产生的新问题?

Function Unit	Latency	Repeat Interval
Integer ALU	**0**	**1**
Data Memory (Integer and FP loads(**1 less for store latency**))	**1**	**1**
FP Add	**3**	**1**
FP multiply	**6**	**1**
FP Divide (also integer divide and FP sqrt)	**24**	**25**

新的相关和定向问题

- **结构冲突增多**
  - 非流水的Divide部件，使得EX段增长24个cycles
- **EX段不同周期数**
  - 在一个周期内可能有多个寄存器写操作
  - 可能指令**乱序完成**（乱序到达WB段）有可能存在WAW
- **由于在ID段读，还不会有 WAR 相关**
- **乱序完成导致异常处理复杂**
- **由于指令的延迟加大导致RAW相关的stall数增多**
- **需要付出更多的代价来增加定向路径**

顺序发射、顺序执行、乱序完成

新的结构相关

Instruction	1	2	3	4	5	6	7	8	9	10	11
MULTD F0, F4, F6	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM	WB
...		IF	ID	EX	MEM	WB					
...			IF	ID	EX	MEM	WB				
ADDD F2, F4, F6				IF	ID	A1	A2	A3	A4	MEM	WB
...					IF	ID	EX	MEM	WB		
...						IF	ID	EX	MEM	WB	
LD F8, 0(R2)							IF	ID	EX	MEM	WB

- **纵向检查指令所使用的资源**
  - 第10个cycle，三条指令同时进入MEM，但由于MULTD和ADDD在MEM段没有实际动作，这种情况没有关系
  - 第11个cycle，三条指令同时进入WB段，存在结构相关

解决方法

- **Option 1**
  - 在ID段跟踪写端口的使用情况，以便能暂停该指令的发射
  - 一旦发现冲突，暂停当前指令的发射
- **Option 2**
  - 在进入MEM或WB段时，暂停冲突的指令，让有较长延时的指令先做。这里假设较长延时的指令，可能会更容易引起其他RAW相关，从而导致更多的stalls

关于数据相关

Instruction		Clock cycle number															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
L.D	F4,0(R2)	IF	ID	EX	MEM	WB												
MUL.D	F0,F4,F6		IF	ID	stall	M1	M2	M3	M4	M5	M6	M7	MEM	WB				
ADD.D	F2,F0,F8			IF	stall	ID	stall	stall	stall	stall	stall	stall	A1	A2	A3	A4	MEM	WB
S.D	F2,0(R2)					IF	stall	stall	stall	stall	stall	stall	ID	EX	stall	stall	stall	MEM

S.D 多延迟一个cycle，以消解与ADD.D的冲突

新的冲突源

- **GPR与FPR间的数据传送造成的数据相关**
  - MOVI2FP and MOVFP2I instructions
- **如果在ID段进行相关检测，指令发射前须做如下检测：**
  - 结构相关
    - 循环间隔检测
    - 确定寄存器写端口是否可用
  - RAW相关
    - 列表所有待写的目的寄存器
    - 不发射以待写寄存器做为源寄存器的指令，插入latency个stall
  - WAW相关
    - 仍然使用上述待写寄存器列表
    - 不发射那些目的寄存器与待写寄存器列表中的指令有WAW冲突的指令，延迟1个cycle发射。

精确异常与长流水线

- **例如**
  - DIVF F0,F2,F4
  - ADDF F10,F10,F8
  - SUBF F12,F12,F14
- **ADDF 和SUBF都在DIVF前完成**
- **如果DIVF导致异常，会如何?**
  - 非精确异常
- **Ideas???**

精确异常与非精确异常

- **精确异常**
  - 如果流水线可以控制使得引起异常的指令前序指令都执行完，故障后的指令可以重新执行，则称该流水线支持精确异常
  - 按照指令的逻辑序处理异常
  - 理想情况，引起故障的指令没有改变机器的状态
  - 要正确的处理这类异常请求，必须保证故障指令不产生副作用
- **在有些机器上，浮点数异常**
  - 流水线段数多，在发现故障前，故障点后的指令就已经写了结果，在这种情况下，必须有办法处理。
  - 很多高性能计算机，Alpha 21164，MIPS R10000等支持精确中断，但精确模式要慢10+倍，一般用在代码调试时，很多系统要求精确异常模式，如IEEE FP标准处理程序，虚拟存储器等。
- **精确异常对整数流水线而言，不是太难实现**
  - 指令执行的中途改变机器的状态
  - 例如IA-32 的自动增量寻址模式

MIPS中的异常

- **IF**
  - page fault, misaligned address, memory protection violation
- **ID**
  - undefined or illegal opcode
- **EX**
  - arithmetic exception
- **MEM**
  - page fault, misaligned address, memory protection violation
- **WB**
  - none

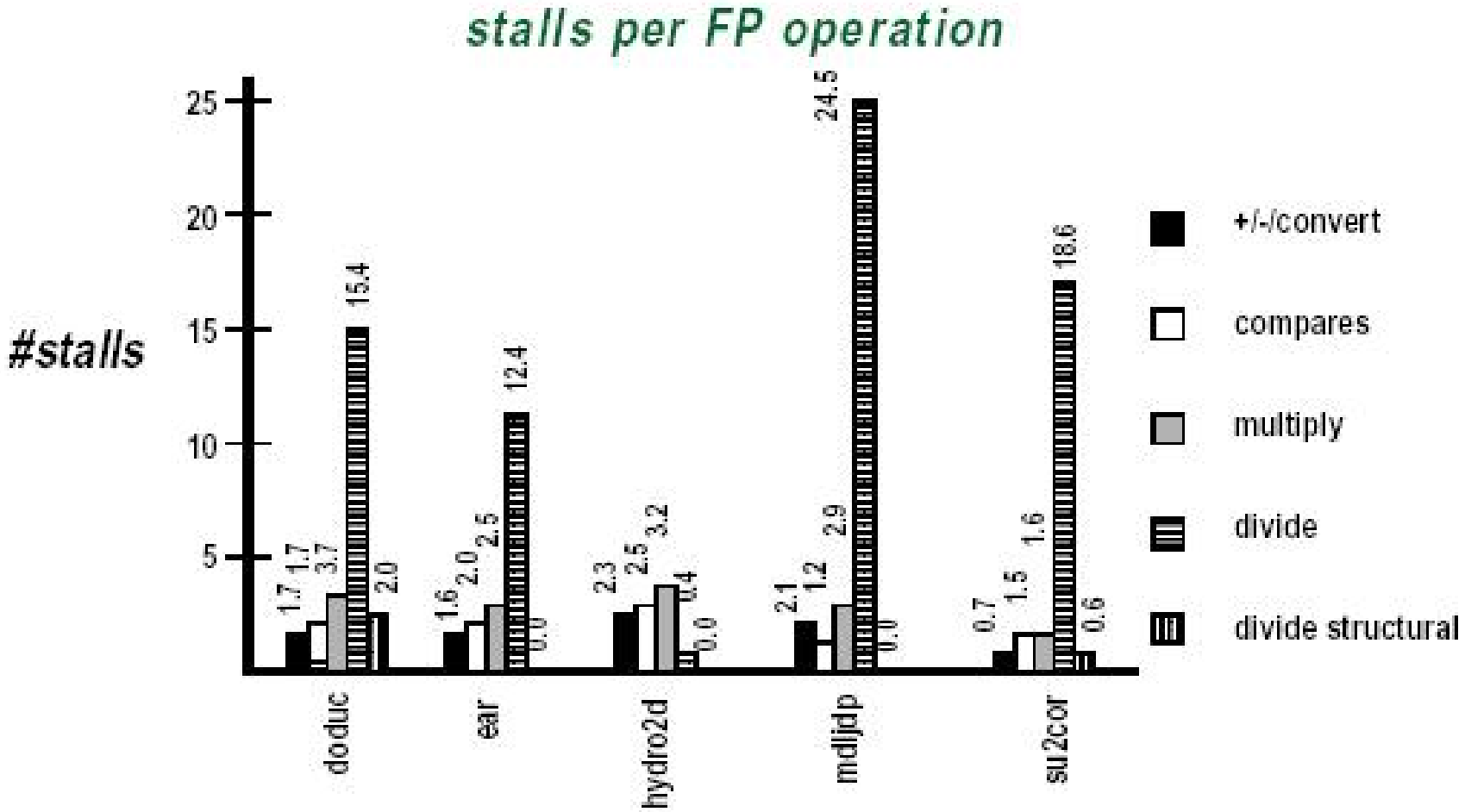
处理异常的4种可能的办法

- **方法1：忽略这种问题，当非精确处理**
  - 原来的supercomputer的方法
  - 但现代计算机对IEEE 浮点标准的异常处理，虚拟存储的异常处理要求必须是精确异常。
- **方法2：缓存操作结果，直到早期发射的指令执行完。**
  - 当指令运行时间较长时，Buffer区较大
  - Future file（Power PC620 MIPS R10000)
    - 缓存执行结果，按指令序确认
  - history file（CYBER 180/990)
    - 尽快确认
    - 缓存区存放原来的操作数，如果异常发生，回卷到合适的状态

第3 & 4种方法

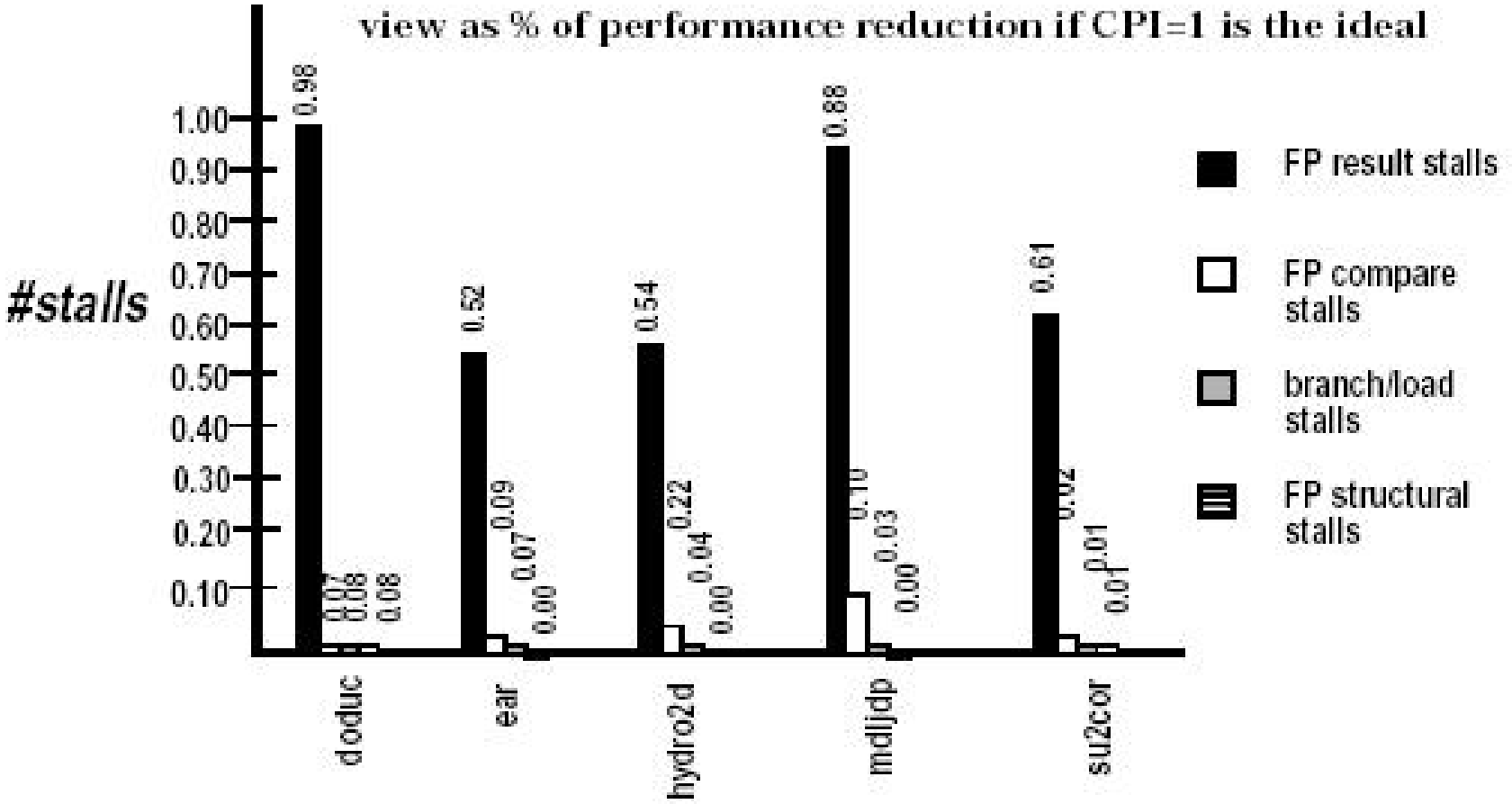
- **方法3：以非精确方式处理，用软件来修正**
  - 为软件修正保存足够的状态
  - 让软件仿真尚未执行完的指令的执行
  - 例如
    - Instruction 1 – A 执行时间较长，引起中断的指令
    - Instruction 2, instruction 3, ….instruction n-1 未执行完的指令
    - Instruction n　已执行完的指令
    - 由于第n条指令已执行完，希望中断返回后从第n+1条指令开始执行，如果我们保存所有的流水线的PC值，那么软件可以仿真Instruction1 到Instruction n-1 的执行
- **方法4：暂停发射，直到确定先前的指令都可无异常的完成，再发射下面的指令。**
  - 在EX段的前期确认（MIPS流水线在前三个周期中）
  - MIPS R2K to R4K 以及Pentium使用这种方法

MIPS流水线的性能



Stalls per FP operation for each major type of FP operation for the SPEC89 FP benchmarks

平均每条指令的stall数



The stalls occurring for the MIPS FP pipeline for five for the SPEC89 FP benchmarks.

3.2 基本流水线的扩展

review：标量流水线

- **流水线提高的是指令带宽（吞吐率），而不是单条指令的执行速度**
- **限制流水线性能发挥的因素：相关（hazard）**
  - 结构相关：需要更多的硬件资源
  - 数据相关：bypassing or forwarding（硬件），编译器调度（软件）
  - 控制相关：尽早检测条件，计算目标地址，延迟转移，预测
- **编译器可降低数据相关和控制相关的开销**
  - Load 延迟槽、Branch 延迟槽、指令流静态调度
- **限制流水线性能发挥的因素：对异常事件的响应**
  - 引起控制流的变化，会冲刷（flush）流水线

review：异常处理

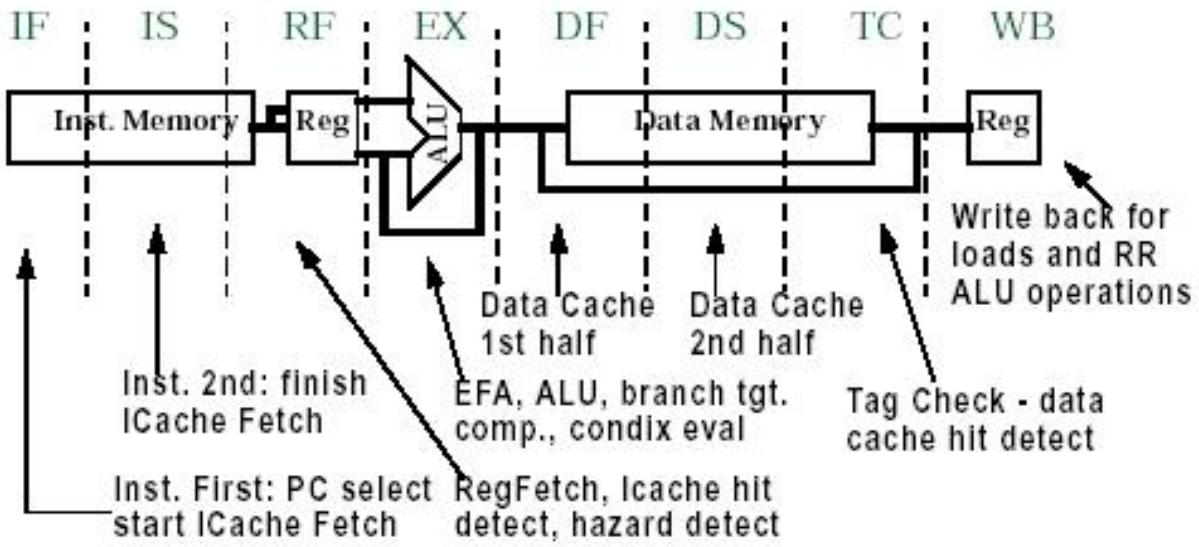
- **广义的异常：异常和中断**
  - 异常：程序运行过程中产生的**内部事件**，称为异常事件，因异常事件将控制权转移到监控程序，称为陷阱（Trap）
  - 中断：响应（处理）程序之外的**外部事件**，导致控制权转移到监控程序
- **同步异常 vs.异步异常**
  - 同步：在每次执行时异常事件发生在相同位置
  - 异步：在每次执行时异常事件可能发生在不同位置
- **精确异常 vs. 非精确异常**
  - 精确：响应异常后，可精确返回引起异常的指令位置
  - 非精确：响应异常后，无法精确返回引起异常的指令位置
- **异常处理：**
  - 异常处理的时机：指令commit阶段 即最后完成阶段
  - 优先级：外部事件引起的异常，内部事件引起的异常（同一条指令按时间顺序）
  - 过程：保存现场、处理、恢复现场

review：支持浮点数操作的MIPS流水线

- **浮点运算使得流水线控制更加复杂**
- **与基本流水线的显著区别**
  - EX段是多周期的，不同操作EX段的周期数不同
  - EX段周期数不同→**乱序完成→WAR相关**
- **增加流水线级数在提高性能的同时，会增加相关产生的可能性**
- **两个重要参数：Latency & Repeat Interval**
- **问题：**
  - 结构相关（增多）；
  - 数据相关、控制相关引起的stall增多；
  - 有新的冲突源产生；定向路径增多；异常处理复杂
- **MIPS R4000 8级整型数流水线**
  - 存储器操作分阶段 – load延迟为2个cycles
  - Branch操作在EX段确定分支方向- 3个cycles的延迟
  - 多个定向源： EX/DF，DF/DS，DS/TC，TC/WB
- **MIPS R4000的浮点数操作**

MIPS R4000

- **实际的 64-bit 机器**
  - 主频100MHz ~200MHz
  - 较深的流水线（级数较多）(有时也称为 superpipelining)

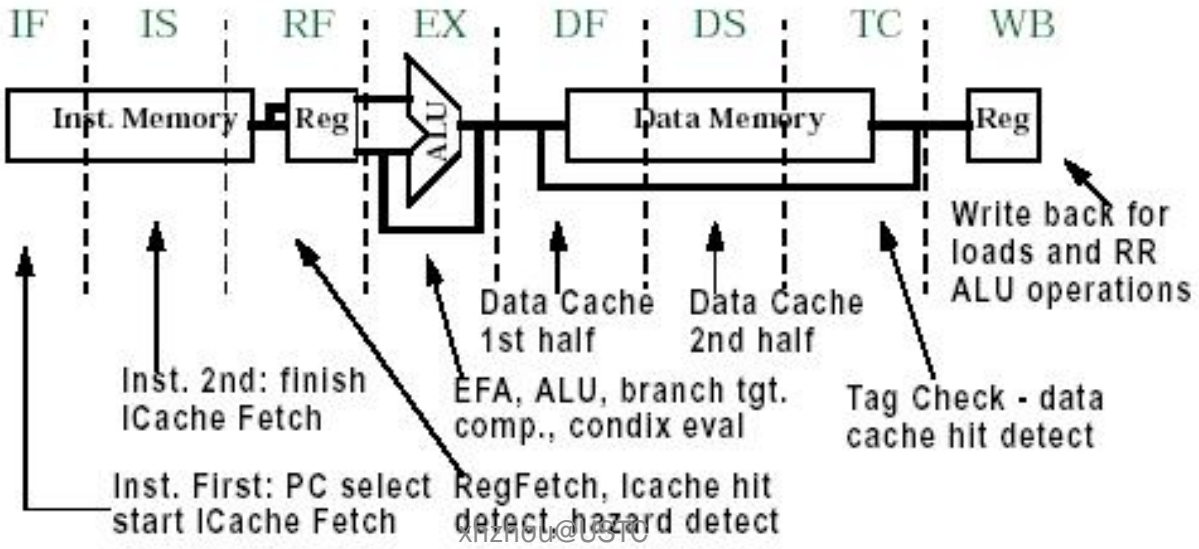


MIPS R4000的8 级整数流水线

- IF–取指阶段前半部分；选择PC值，初始化指令cache的访问
- IS–取指阶段后半部分，主要完成访问指令cache的操作
- RF–指令译码，寄存器读取，相关检测及指令cache命中检测
- EX–执行，包括：计算有效地址，进行ALU操作，计算分支目标地址和检测分支条件
- DF–取数据，访问数据cache的前半部分
- DS–访问数据cache的后半部分
- TC–tag 检测，确定数据cache是否命中
- WB–Load操作和R-R操作的结果写回

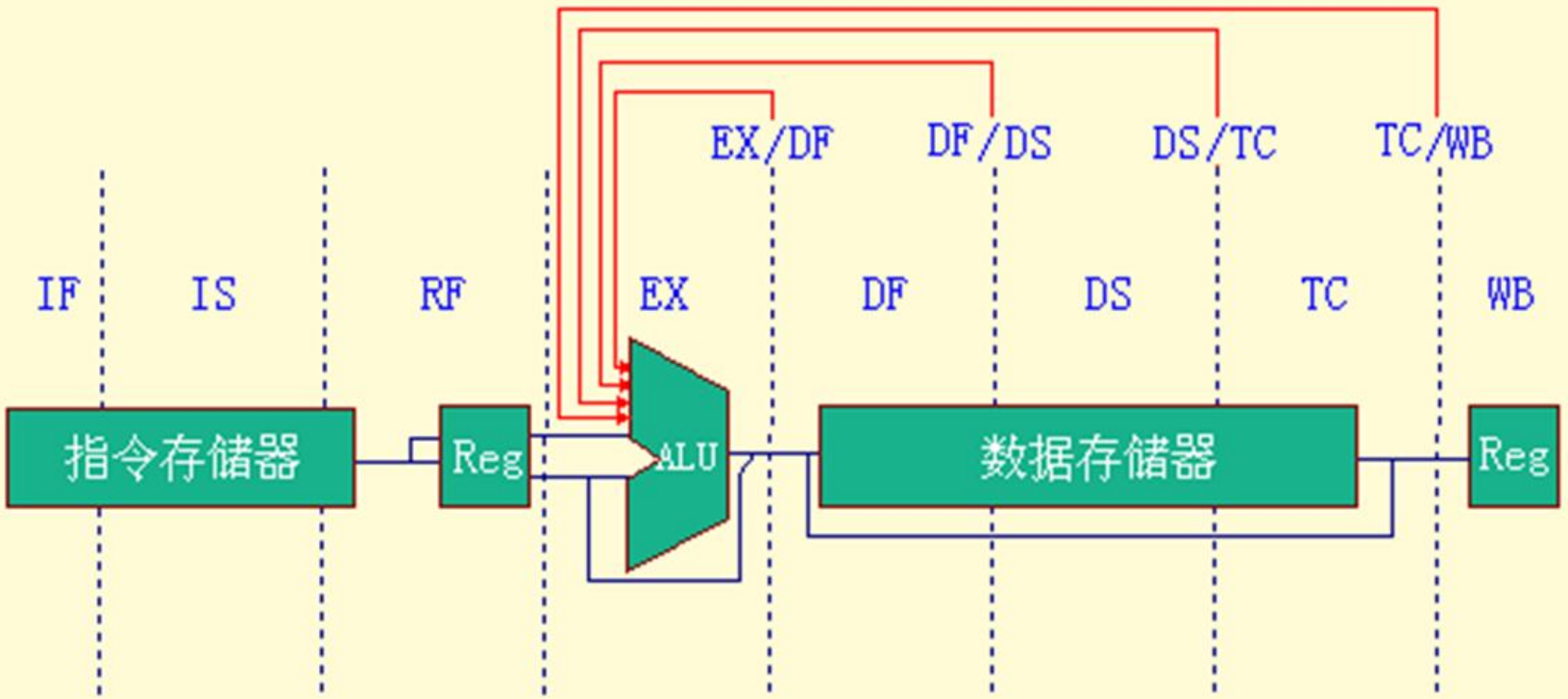
需注意的问题

- **在使用定向技术的情况下，Load 延迟为2个cycles**
  - Load和与其相关的指令间必须有2条指令或两个bubbles
  - 原因：load的结果在DS结束时可用
- **分支延迟3个cycles**
  - 分支与目标指令间需要3条指令或3个bubbles
  - 原因：目标地址在EX段后才能知道
- **R4000的流水线中，到ALU输入端有四个定向源**
  - EX/DF, DF/DS, DS/ TC, TC/WB



ALU输入端的定向源

R4000的流水线中ALU输入有四个定向源



图示

TWO Cycle Load Latency

IF	IS	RF	EX	DF	DS	TC	WB
	IF	IS	RF	EX	DF	DS	TC
		IF	IS	RF	EX	DF	DS
			IF	IS	RF	EX	DF
				IF	IS	RF	EX
					IF	IS	RF
						IF	IS
							IF

THREE Cycle Branch Latency

(conditions evaluated during EX phase)

IF	IS	RF	EX	DF	DS	TC	WB
	IF	IS	RF	EX	DF	DS	TC
		IF	IS	RF	EX	DF	DS
			IF	IS	RF	EX	DF
				IF	IS	RF	EX
					IF	IS	RF
						IF	IS
							IF

Delay slot plus two stalls

Branch likely cancels delay slot if not taken

MIPS R4000 浮点数操作

- **3个功能部件组成：FP Adder, FP Multiplier, FP Divider**
- **在乘/除操作的最后一步要 使用FP Adder**
- **FP操作需要2（negate）-112个（square root）cycles**

- **8 种类型的FP units:**

Stage	Functional unit	Description
A	FP adder	Mantissa ADD stage
D	FP divider	Divide pipeline stage
E	FP multiplier	Exception test stage
M	FP multiplier	First stage of multiplier
N	FP multiplier	Second stage of multiplier
R	FP adder	Rounding stage
S	FP adder	Operand shift stage
U	Unpack FP numbers	

流水段	功能部件	描　　述
A	浮点加法器	尾数加流水段
D	浮点除法器	除法流水段
E	浮点乘法器	例外测试段
M	浮点乘法器	乘法器第一个流水段
N	浮点乘法器	乘法器第二个流水段
R	浮点加法器	舍入段
S	浮点加法器	操作数移位段
U		展开浮点数

双精度浮点数操作延迟及初始化间隔

浮点指令	延 迟	初始化间隔	使用的流水段
加、减	4	3	U,S+A,A+R,R+S
乘	8	4	U,E+M,M,M,M,N,N+A,R
除	36	35	U,A,R,D^{28},D+A,D+R,D+A,D+R,A,R
求平方根	112	111	U,E,(A+R) ^{108},A,R
取反	2	1	U,S
求绝对值	2	1	U,S
浮点比较	3	2	U,A,R

MIPS FP 流水段

FP Instr	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*…*
Add, Subtract	U	S+A	A+R	R+S					
Multiply	U	E+M	M	M	M	N	N+A	R	
Divide	U	A	R	D^{28}	…				D+A，D+R, D+R, D+A, D+R, A, R
Square root	U	E	$(A+R)^{108}$		…				AR
Negate	U	S							
Absolute value	U	S							
FP compare	U	A	R						

Stages:

M	*First stage of multiplier*
N	*Second stage of multiplier*
R	*Rounding stage*
S	*Operand shift stage*
U	*Unpack FP numbers*

A	***Mantissa ADD stage***
D	***Divide pipeline stage***
E	***Exception test stage***

		Clock cycle												
Operation	Issue/stall	0	1	2	3	4	5	6	7	8	9	10	11	12
Multiply	Issue	U	E+M	M	M	M	N	N+A	R					
Add	Issue		U	S+A	A+R	R+S								
	Issue			U	S+A	A+R	R+S							
	Issue				U	S+A	A+R	R+S						
	Stall					U	S+A	A+R	R+S					
	Stall						U	S+A	A+R	R+S				
	Issue							U	S+A	A+R	R+S			
	Issue								U	S+A	A+R	R+S		

		Clock cycle												
Operation	Issue/stall	0	1	2	3	4	5	6	7	8	9	10	11	12
Add	Issue	U	S+A	A+R	R+S									
Multiply	Issue		U	**E+M**	M	M	M	N	N+A	R				
	Issue			U	M	M	M	M	N	N+A	R			

		Clock cycle											
Operation	**Issue/stall**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
Divide	issued in cycle 0...	D	D	D	D	D	D+A	D+R	D+A	D+R	A	R	
Add	Issue		U	S+A	A+R	R+S							
	Issue			U	S+A	A+R	R+S						
	Stall				U	S+A	A+R	R+S					
	Stall					U	S+A	A+R	R+S				
	Stall						U	S+A	A+R	R+S			
	Stall							U	S+A	A+R	R+S		
	Stall								U	S+A	A+R	R+S	
	Stall									U	S+A	A+R	R+S
	Issue										U	S+A	A+R
	Issue											U	S+A
	Issue												U

		Clock cycle												
Operation	**Issue/stall**	0	1	2	3	4	5	6	7	8	9	10	11	12
Add	Issue	U	S+A	**A+R**	**R+S**									
Divide	Stall		U	**A**	**R**	D	D	D	D	D	D	D	D	D
	Issue			U	A	R	D	D	D	D	D	D	D	D
	Issue				U	A	R	D	D	D	D	D	D	D

R4000性能（1）

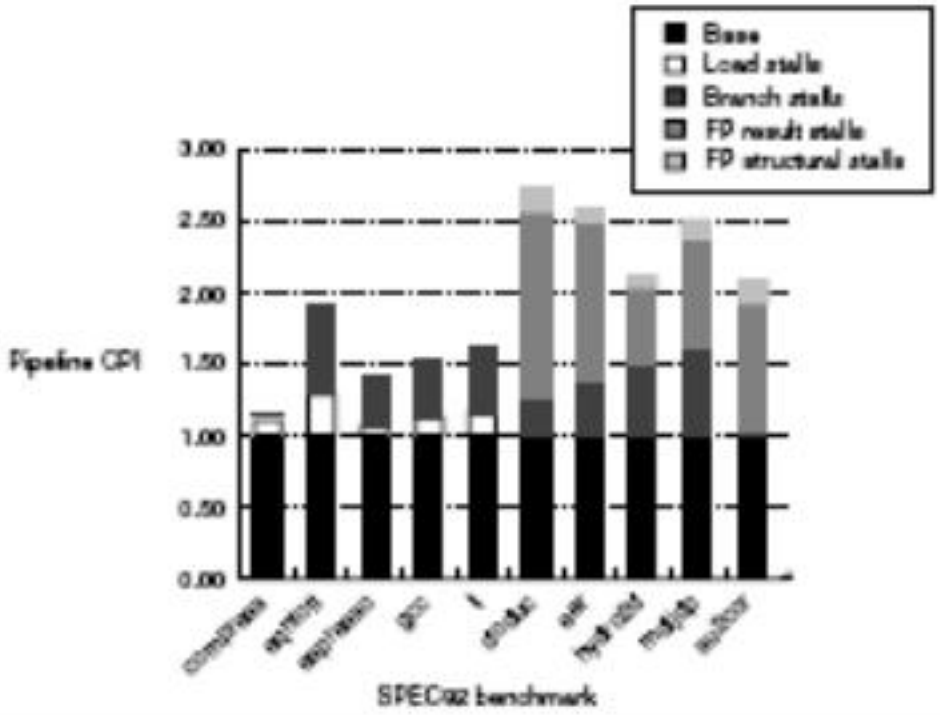


Figure A.48 The pipeline CPI for 10 of the SPEC92 benchmarks, assuming a perfect cache. The pipeline CPI varies from 1.2 to 2.8. The leftmost five programs are integer programs, and branch delays are the major CPI contributor for these. The rightmost five programs are FP, and FP result stalls are the major contributor for these. Figure A.49 shows the numbers used to construct this plot.

R4000 性能（2）

Benchmark	Pipe CPI	Load Stalls	Branch Stalls	FP Res. Stalls	FP Struc. Stalls
compress	1.20	0.14	0.06	0.00	0.00
eqntott	1.88	0.27	0.61	0.00	0.00
espresso	1.42	0.07	0.35	0.00	0.00
gcc	1.56	0.13	0.43	0.00	0.00
li	1.64	0.18	0.46	0.00	0.00
INTEGER AVERAGE	1.54	0.16	0.39	0.00	0.00
doduc	2.84	0.01	0.22	1.39	0.22
mkljdp2	2.66	0.01	0.31	1.20	0.15
ear	2.17	0.00	0.46	0.59	0.12
hydro2d	2.53	0.00	0.62	0.75	0.17
su2cor	2.18	0.02	0.07	0.84	0.26
FP AVERAGE	2.48	0.01	0.33	0.95	0.18
OVERALL AVERAGE	2.00	0.10	0.36	0.46	0.09

基本流水线小结

- **流水线提高的是指令带宽（吞吐率），而不是单条指令的执行速度**
- **相关限制了流水线性能的发挥**
  - 结构相关：更多的硬件资源
  - 数据相关：定向路径，编译器调度
  - 控制相关：尽早检测条件，计算目标地址，延迟转移，预测
- **增加流水线的级数会增加相关产生的可能性**
- **异常，浮点运算使得流水线控制更加复杂**
- **编译技术可降低数据相关和控制相关的开销**
  - Load 延迟l
  - Branch 延迟槽
  - Branch预测

小结

- **影响流水线性能**
  - 结构相关、数据相关
  - 控制相关、异常
- **异常处理**
  - 种类与分类
  - 精确与非精确中断
- **支持浮点数操作的MIPS流水线**
  - Latency & Repeat Interval
  - 问题：结构相关（增多）；数据相关、控制相关引起的stall增多；有新的冲突源产生；定向路径增多；异常处理复杂
  - MIPS R4000 8级流水线
    - 存储器操作分阶段 – load延迟为2个cycles
    - Branch操作在EX段确定分支方向- 3个cycles的延迟
    - 多个定向源： EX/DF，DF/DS，DS/TC，TC/WB
  - MIPS R4000的浮点数操作

Acknowledgements

- **These slides contain material developed and copyright by:**
  - John Kubiatowicz (UCB)
  - Krste Asanovic (UCB)
  - David Patterson (UCB)
  - Chenxi Zhang (Tongji)
- **UCB material derived from course CS152、CS252、CS61C**
- **KFUPM material derived from course COE501、COE502**