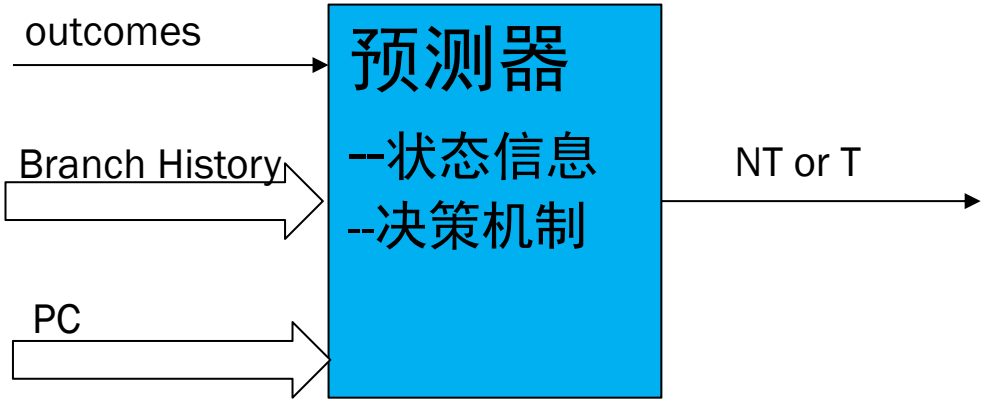
计算机体系结构

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

Review

- **基于BHT表的预测器:**
  - Basic 2-bit predictor:
  - Global predictor:
    - 每个分支对应多个m-bit预测器
    - 最近n次的分支转移的每一种情况分别对应其中一个预测器
  - Local predictor:
    - 每个分支对应多个m-bit预测器
    - 该分支最近n次分支转移的每一种情况分别对应其中一个预测器
  - Tournament predictor:
    - 从多种预测器的预测结果中选择合适的预测结果。
    - 例如：两级全局预测器与两级局部预测器
- **优化取指令的带宽**
  - **基于BTB的分支预测器**
  - **Return Address Stack**
  - **集成的独立的取指部件**

BHT预测器的基本结构及输入输出



- **根据转移历史(和PC)来选择预测器**
- **由预测器的状态决定预测值(输出)**
- **根据实际结果(outcomes)更新预测器的状态信息**

5.4 分支预测技术

控制相关对性能的影响

基于BHT的分支预测

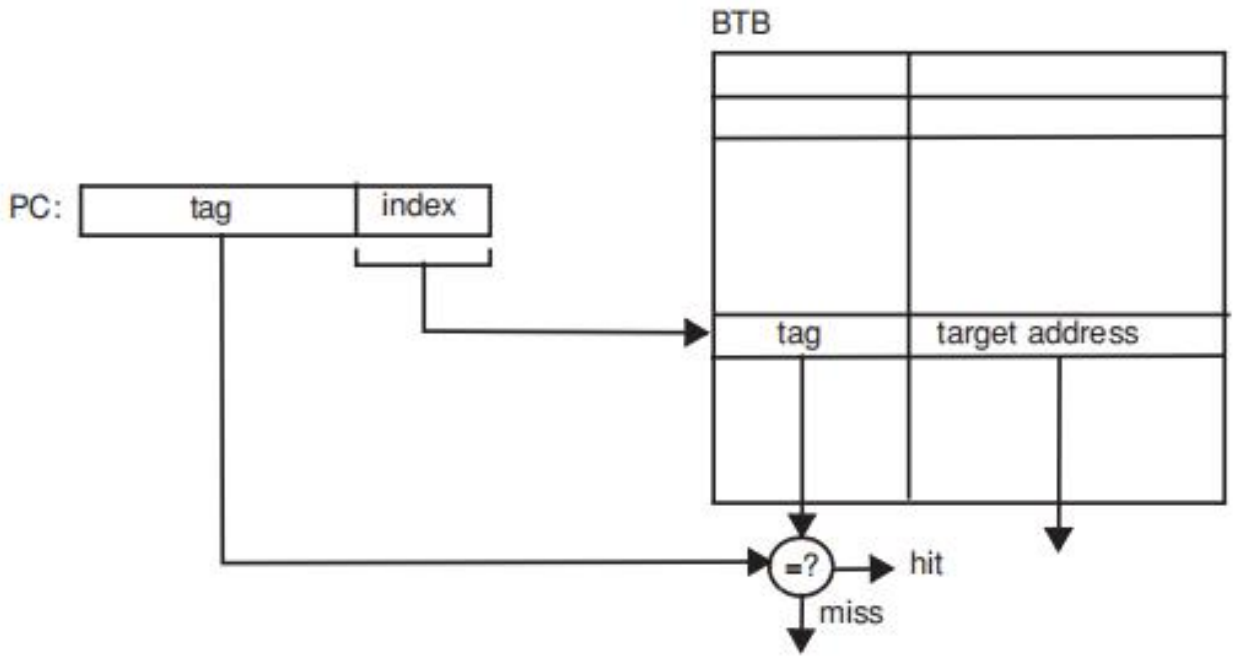
1、基本2-bit预测器
2、关联预测器（两级预测器）
3、组合预测器

基于BTB的分支预测

1、分支目标缓冲区
2、Return Address预测器

Branch Target Buffer (BTB)

- **BTB 小容量的Cache**
- **分支指令的地址作为BTB的索引，以得到分支预测地址**
  - 必须检测分支指令的地址是否匹配，以免用错误的分支地址
  - 从表中得到预测地址
  - 分支方向确定后，更新预测的PC



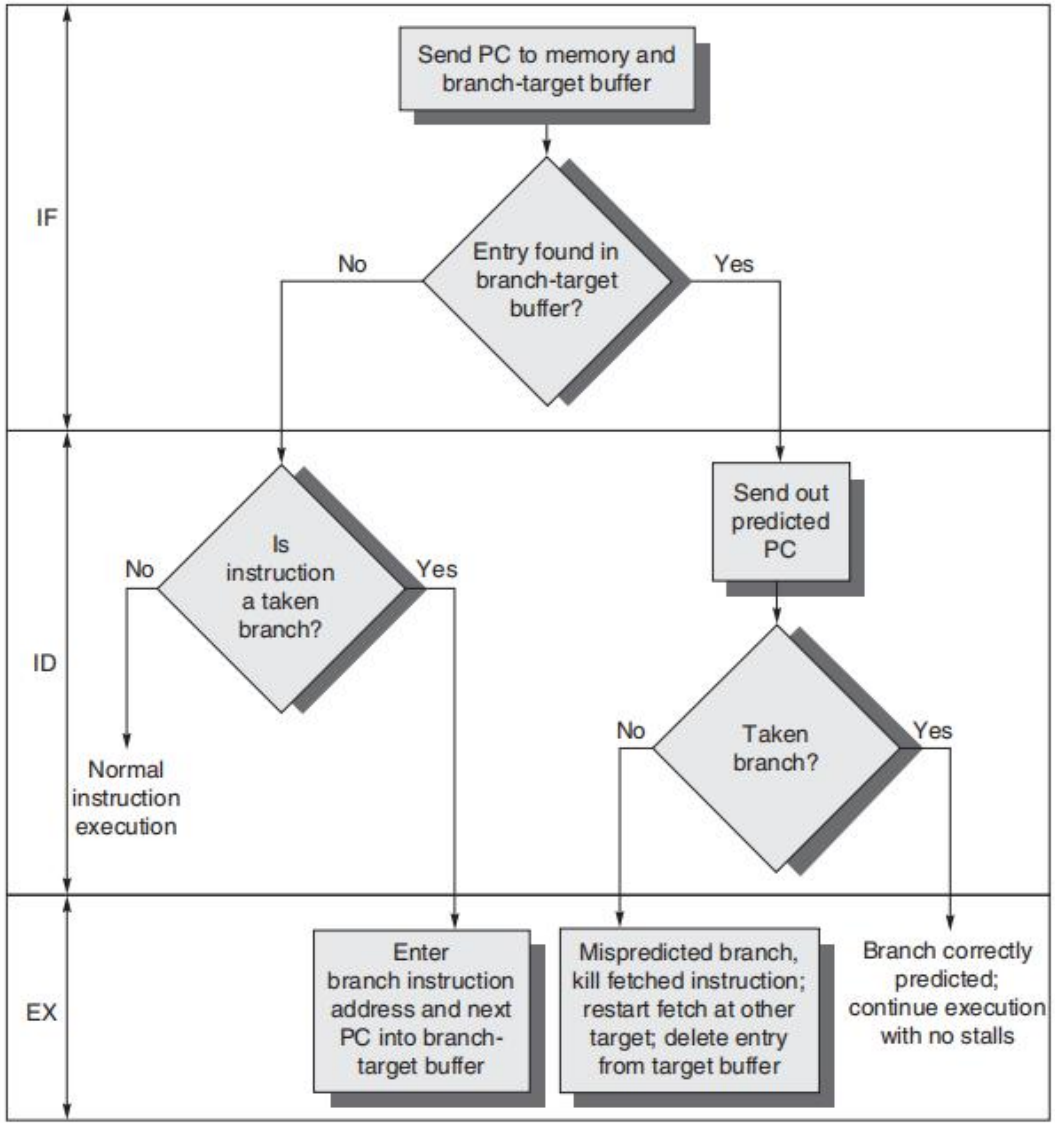


Figure 3.22 The steps involved in handling an instruction with a branch-target buffer.

例如：基本模型

- 简单的五段流水
- ID段 确认 是否 可以 跳转

- BTB预测器 分支目标缓存的换入换出

Instruction in buffer	Prediction	Actual branch	Penalty cycles
yes	taken	taken	0
yes	taken	not taken	2
no		taken	2
no		not taken	0

Figure 2.24 Penalties for all possible combinations of whether the branch is in the buffer and what it actually does, assuming we store only taken branches in the buffer. There is no branch penalty if everything is correctly predicted and the branch is found in the target buffer. If the branch is not correctly predicted, the penalty is equal to 1 clock cycle to update the buffer with the correct information (during which an instruction cannot be fetched) and 1 clock cycle, if needed, to restart fetching the next correct instruction for the branch. If the branch is not found and taken, a 2-cycle penalty is encountered, during which time the buffer is updated.

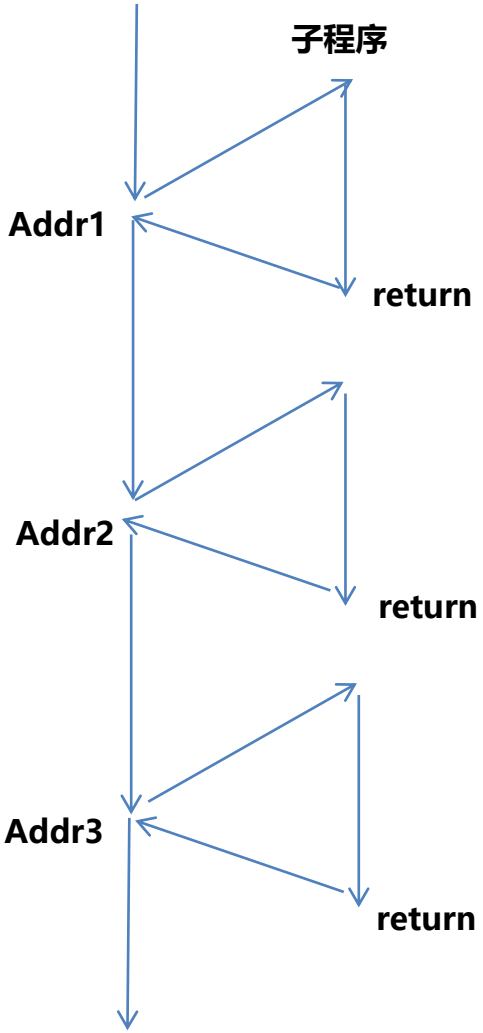
Branch Penalty：如果在BTB中命中，并且预测正确，则Penalty 为0，其他情况则Penalty为2

假设不同情况下预测错误的代价如上图，请基于如下条件确定采用BTB分支预测预测错误的总开销。

- 对于在BTB中命中的分支指令，分支预测的准确率（精度）为90%
- 对于分支预测转移成功的指令在BTB中的命中率为 90%

Return Address Predictors

- **投机执行面临的挑战：预测间接跳转**
  - 运行时才能确定分支目标地址
- **多数间接跳转来源于Procedure Return**
  - 采用BTB时，对于过程返回的预测精度较低
  - SPEC CPU95测试，这类分支预测的准确性不到60%
- **使用一个小的缓存(栈）存放 Return Address**
  - 过程调用时将返回地址压入该栈
  - 过程返回时通过弹栈操作获得转移地址



Return Address Buffer entries

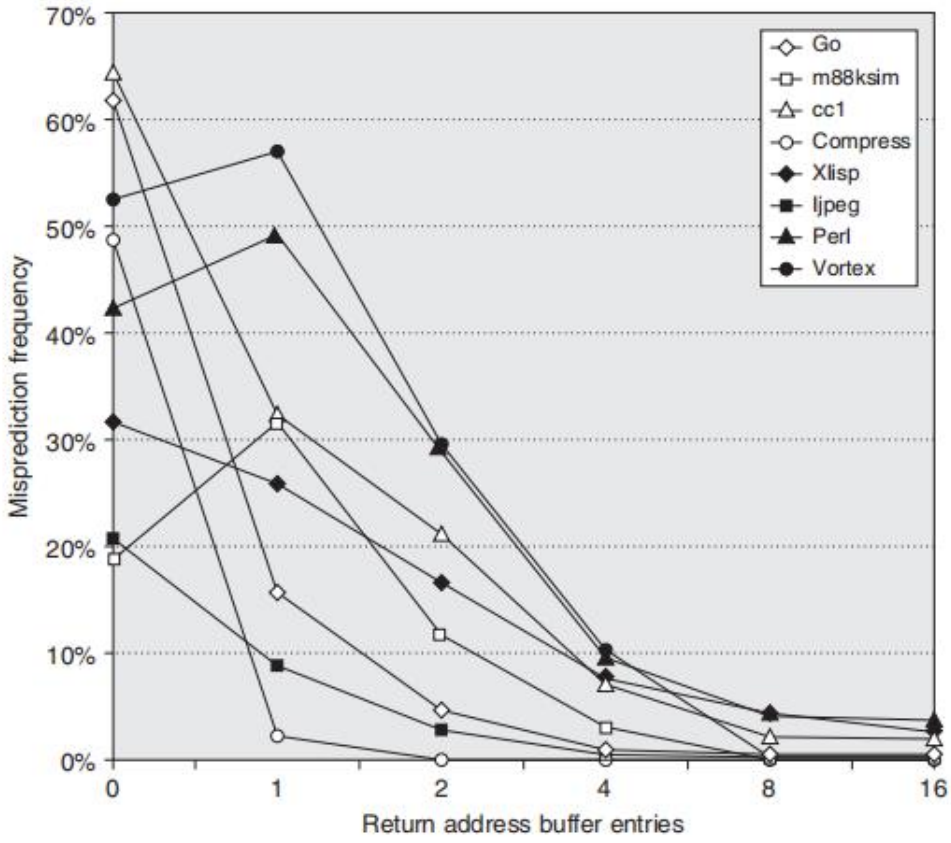


Figure 3.24 Prediction accuracy for a return address buffer operated as a stack on a number of SPEC CPU95 benchmarks. The accuracy is the fraction of return addresses predicted correctly. A buffer of 0 entries implies that the standard branch prediction is used. Since call depths are typically not large, with some exceptions, a modest buffer works well. These data come from Skadron et al. [1999] and use a fix-up mechanism to prevent corruption of the cached return addresses.

- **返回栈（Return Address Buffer)中表项数（entries)与预测精度的关系**

Instruction Fetch Unit

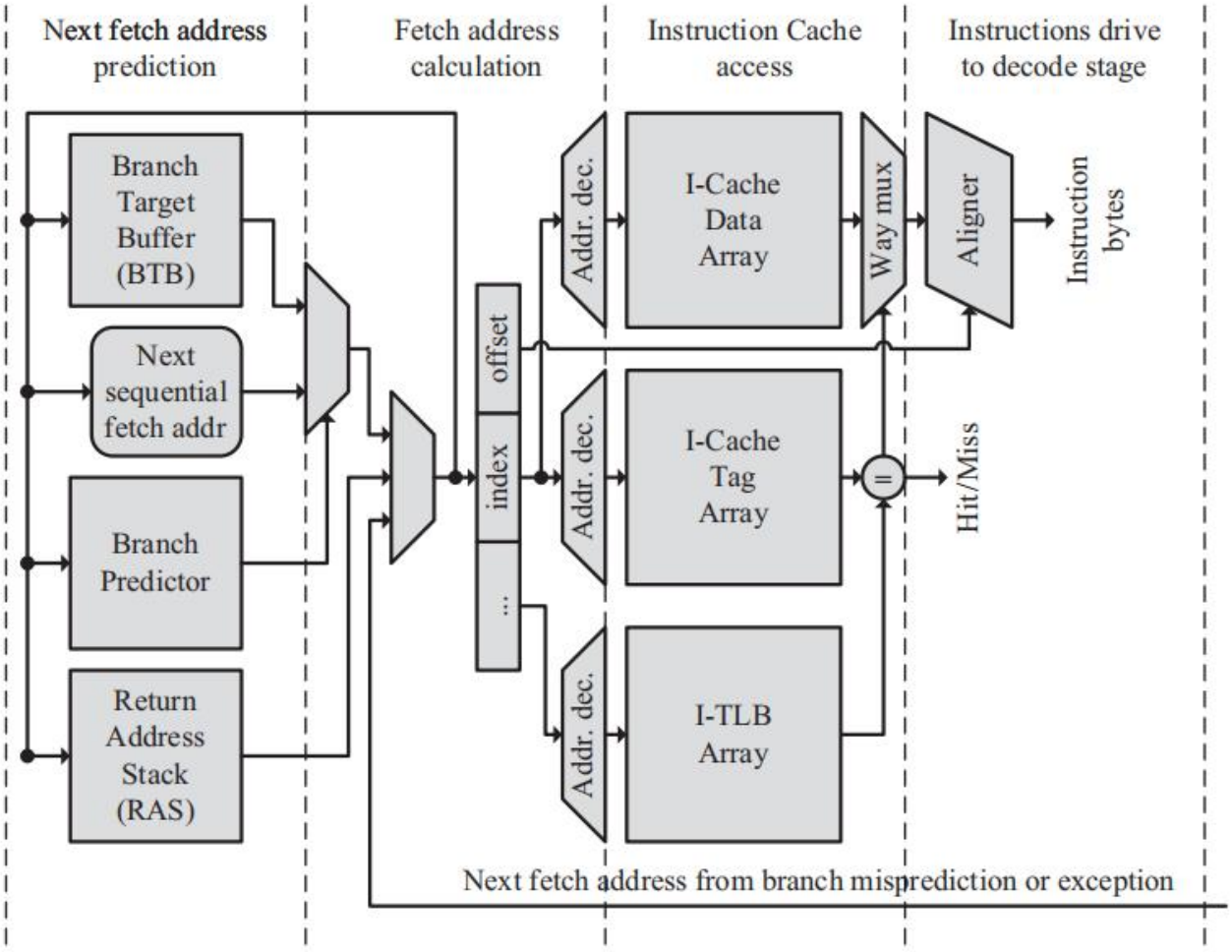


FIGURE 3.1: Example fetch pipeline.

第5章 指令级并行

5.1 指令级并行的基本概念及静态指令流调度

ILP及挑战性问题

软件方法挖掘指令集并行

基本块内的指令集并行

5.2硬件方法挖掘指令级并行(4学时)

5.2-1 指令流动态调度方法之一：Scoreboard

5.2-2 指令流动态调度方法之二：Tomasulo

5.3 分支预测方法

5.4 基于硬件的推测执行：3.6

5.5-1 存储器访问冲突消解

5.5-2 多发射技术

5.6 多线程技术

5.4 推断执行

支持推断执行的Tomasulo

1. 带有ROB的机器结构
2. 四阶段算法描述

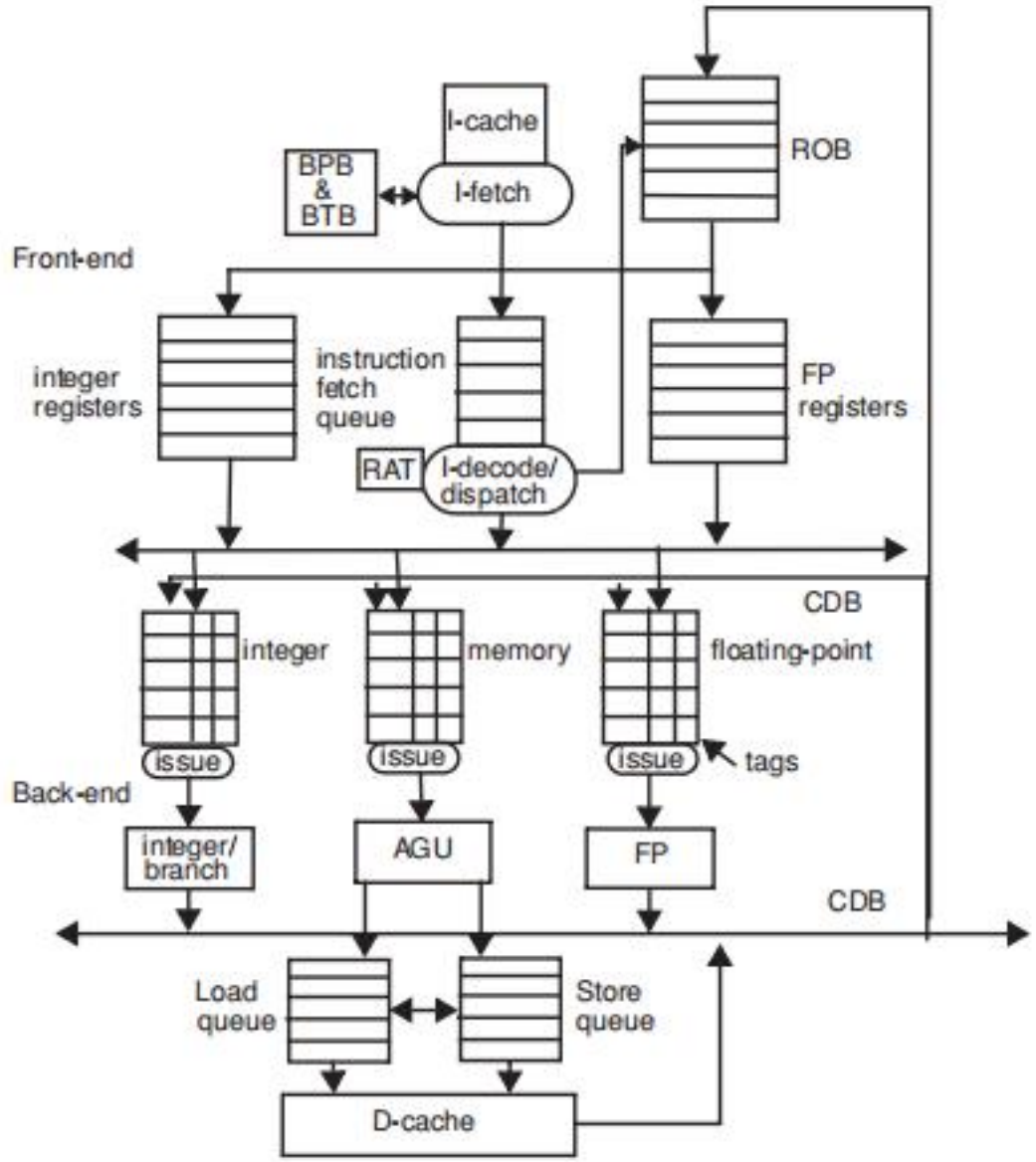
代码执行示例

1. 简单代码示例
2. 推断执行示例

Tomasulo小结

1. ROB的作用
2. 动态内存歧义消除

一种支持推断执行的机器结构

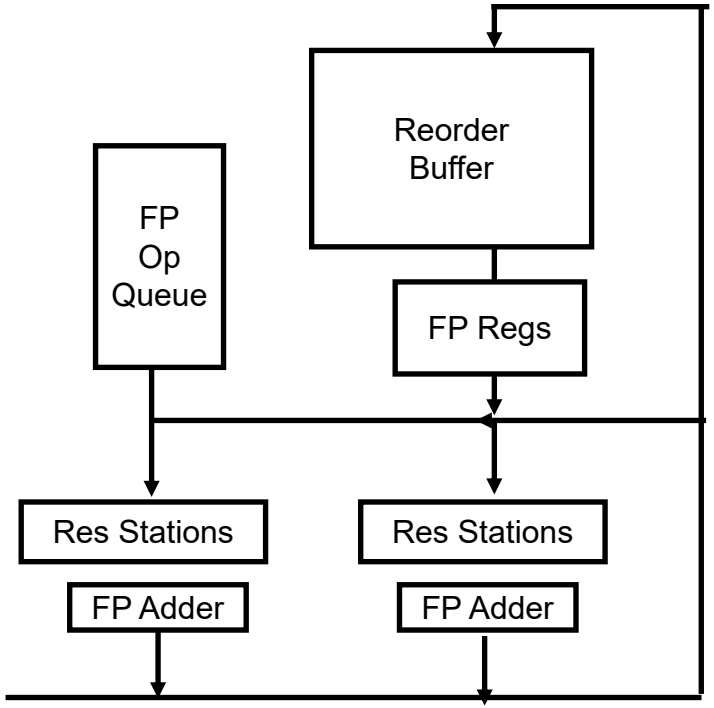


支持硬件推断执行和精确异常处理的Tomasulo算法所依赖的机器结构

1. **带有Reorder Buffer**（ROB）
2. **带有BPB和BTB**，具有快速解决控制相关的能力

硬件支持推断执行以及精确异常

- **支持推断执行的条件：具有“恢复”能力**
- **硬件缓存没有提交的指令结果: reorder buffer (ROB)**
  - 3 个域: 指令类型,目的地址, 值
  - Reorder buffer 可以作为操作数源 => 就像有更多的寄存器（与RS类似）
  - 当指令执行阶段完成后，用ROB的编号代替RS中的值
  - 增加指令提交阶段（Commit）
  - ROB提供执行完成阶段和提交阶段的操作数
  - 一旦结果提交，结果就写入寄存器
  - 在预测错误时，容易恢复推断执行的指令，或发生异常时，容易恢复状态



支持推断执行的 Tomasulo 算法的四阶段

- **1. Issue—get instruction from FP Op Queue**
  - 如果RS和ROB有空闲单元就发射指令。如果寄存器或ROB中源操作数可用，就将其发送到RS，目的地址的ROB编号也发送给RS
- **2. Execution—operate on operands (EX)**
  - 当操作数就绪后，开始执行。如果没有就绪，监测CDB，检查RAW相关
- **3. Write result—finish execution (WB)**
  - 将运算结果通过CDB传送给所有等待结果的FU以及ROB单元，标识RS可用
- **4. Commit—update register with reorder result**
  - 按ROB表中顺序，如果结果已有，就更新寄存器（或存储器），并将该指令从ROB表中删除
  - 预测错误或有异常（中断）时，刷新ROB
  - P191 Figure 3.14 (英文版)，P141 Figure 3-9 (中文版)
- **执行过程中需要检测CDB冲突**

Issue

Status	Wait until	Action or bookkeeping
Issue all instructions	Reservation station (r) and ROB (b) both available	`if (RegisterStat[rs].Busy)/*in-flight instr. writes rs*/ {h ← RegisterStat[rs].Reorder; if (ROB[h].Ready)/* Instr completed already */ {RS[r].Vj ← ROB[h].Value; RS[r].Qj ← 0;} else {RS[r].Qj ← h;} /* wait for instruction */ } else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0;}; RS[r].Busy ← yes; RS[r].Dest ← b; ROB[b].Instruction ← opcode; ROB[b].Dest ← rd;ROB[b].Ready ← no;`
FP operations and stores		`if (RegisterStat[rt].Busy) /*in-flight instr writes rt*/ {h ← RegisterStat[rt].Reorder; if (ROB[h].Ready)/* Instr completed already */ {RS[r].Vk ← ROB[h].Value; RS[r].Qk ← 0;} else {RS[r].Qk ← h;} /* wait for instruction */ } else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0;};`
FP operations		`RegisterStat[rd].Reorder ← b; RegisterStat[rd].Busy ← yes; ROB[b].Dest ← rd;`
Loads		`RS[r].A ← imm; RegisterStat[rt].Reorder ← b; RegisterStat[rt].Busy ← yes; ROB[b].Dest ← rt;`
Stores		`RS[r].A ← imm;`

rs：FP操作指令源操作数寄存器，Load/store指令的基址寄存器

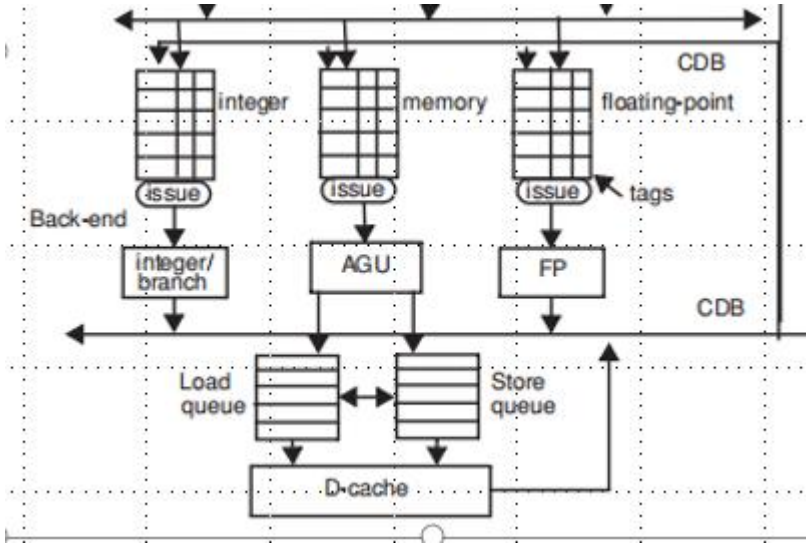
rt：FP指令的源操作数寄存器，store操作的待写入的寄存器，load操作的目的寄存器

h：ROB中当前指令所依赖的指令对应的ROB编号；

b：当前指令对应的ROB编号；r:当前指令对应的保留站编号

Execute

Execute FP op	(RS[r].Qj == 0) and (RS[r].Qk == 0)	Compute results—operands are in Vj and Vk
Load step 1	(RS[r].Qj == 0) and there are no stores earlier in the queue	`RS[r].A ← RS[r].Vj + RS[r].A;`
Load step 2	Load step 1 done and all stores earlier in ROB have different address	Read from Mem[RS[r].A]
Store	(RS[r].Qj == 0) and store at queue head	`ROB[h].Address ← RS[r].Vj + RS[r].A;`



Write result & Commit

Write result all but store	Execution done at *r* and CDB available	b ← RS[r].Dest; RS[r].Busy ← no; ∀x(if (RS[x].Qj==b) {RS[x].Vj ← result; RS[x].Qj ← 0}); ∀x(if (RS[x].Qk==b) {RS[x].Vk ← result; RS[x].Qk ← 0}); ROB[b].Value ← result; ROB[b].Ready ← yes;
Store	Execution done at *r* and (RS[r].Qk == 0)	ROB[h].Value ← RS[r].Vk;

Commit	Instruction is at the head of the ROB (entry h) and ROB[h].ready == yes	d ← ROB[h].Dest; /* register dest, if exists */
		if (ROB[h].Instruction==Branch)
		{if (branch is mispredicted)
		{clear ROB[h], RegisterStat; fetch branch dest;};}
		else if (ROB[h].Instruction==Store)
		{Mem[ROB[h].Destination] ← ROB[h].Value;}
		else /* put the result in the register destination */
		{Regs[d] ← ROB[h].Value;};
		ROB[h].Busy ← no; /* free up ROB entry */
		/* free up dest register if no one else writing it */
		if (RegisterStat[d].Reorder==h) {RegisterStat[d].Busy ← no;};

5.4 推断执行

支持推断执行的Tomasulo

1. 带有ROB的机器结构
2. 四阶段算法描述

代码执行示例

1. 简单代码示例
2. 推断执行示例

Tomasulo小结

1. ROB的作用
2. 动态内存歧义消除

例如：

```
LD      F6, 34(R2)
LD      F2, 45(R3)
MULT    F0, F2, F4
SUBD    F8, F6, F2
DIVD    F10, F0, F6
ADDD    F6, F8, F2
```

假设：执行阶段的周期数

LD： 1 cycles MULT: 10 cycles

SUBD/ADDD: 2cycles DIVD: 40 cycles

Tomasulo With Reorder Buffer-Cycle 0

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
0	Mult2	No						

Reservation Station

LD F6, 34(R2)
LD F2, 45(R3)
MULT F0, F2, F4
SUBD F8, F6, F2
DIVD F10, F0, F6
ADDD F6, F8, F2

Entry	Busy	Instruction	State	Destination	Value
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

	Busy	Address
Load1		
Load2		
Load3		

Reorder Buffer

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
0	Reorder#									
	Busy	No	No	No	No	No	No	No		No

假设：执行阶段的周期数

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Tomasulo With Reorder Buffer-Cycle 1

LD： 1 cycles MULT: 10 cycles SUBD/ADDD: 2cycles DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
0	Mult2	No						

Reservation Station

LD F6, 34(R2)
LD F2, 45(R3)
MULT F0, F2, F4
SUBD F8, F6, F2
DIVD F10, F0, F6
ADDD F6, F8, F2

	Entry	Busy	Instruction	State	Destination	Value
Head	1	Yes	LD F6,34(R2)	Issue	F6	
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					

Reorder Buffer

	Busy	Address
Load1	Yes	34+Regs[R2]
Load2		
Load3		

Cycle 1

	F0	F2	F4	F6	F8	F10	F12	……	F30
Reorder#				#1					
Busy	No	No	No	Yes	No	No	No		No

Tomasulo With Reorder Buffer-Cycle 2

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
0	Mult2	No						

Reservation Station

	Busy	Address
Load1	Yes	34+Regs[R2]
Load2	Yes	45+Regs[R3]
Load3		

LD F6, 34(R2)
LD F2, 45(R3)
MULT F0, F2, F4
SUBD F8, F6, F2
DIVD F10, F0, F6
ADDD F6, F8, F2

	Entry	Busy	Instruction	State	Destination	Value
Head	1	Yes	LD F6,34(R2)	Ex1	F6	
tail	2	Yes	LD F2,45(R3)	Issue	F2	
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	10					

Reorder Buffer

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
2	Reorder#		#2		#1					
	Busy	No	Yes	No	Yes	No	No	No		No

Tomasulo With Reorder Buffer-Cycle 3

LD： 1 cycles MULT: 10 cycles SUBD/ADDD: 2cycles DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	Yes	Mult		Regs[F4]	#2		#3
0	Mult2	No						

Reservation Station

	Busy	Address
Load1	No	
Load2	Yes	45+Regs[R3]
Load3		

LD F6, 34(R2)
LD F2, 45(R3)
MULT F0, F2, F4
SUBD F8, F6, F2
DIVD F10, F0, F6
ADDD F6, F8, F2

	Entry	Busy	Instruction	State	Destination	Value
Head	1	Yes	LD F6,34(R2)	Write	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Ex1	F2	
tail	3	Yes	MULT F0,F2,F4	Issue	F0	
	4					
	5					
	6					
	7					
	8					
	9					
	10					

Reorder Buffer

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
3	Reorder#	#3	#2		#1					
	Busy	Yes	Yes	No	Yes	No	No	No		No

Tomasulo With Reorder Buffer-Cycle 4

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
2	Add1	Yes	SUB	Regs[F6]	Mem[45+regs[R3]]		#2	#4
0	Add2	No						
0	Add3	No						
10	Mult1	Yes	Mult	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	No						

Reservation Station

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
Head	2	Yes	LD F2,45(R3)	Write	F2	Mem[load2]
	3	Yes	MULT F0,F2,F4	Issue	F0	
Tail	4	Yes	SUBD F8,F6,F2	Issue	F8	
	5					
	6					
	7					
	8					
	9					
	10					

Reorder Buffer

LD F6, 34(R2)
LD F2, 45(R3)
MULT F0, F2, F4
SUBD F8, F6, F2
DIVD F10, F0, F6
ADDD F6, F8, F2

Cycle		F0	F2	F4	F6	F8	F10	F12	……	F30
4	Reorder#	#3	#2			#4				
	Busy	Yes	Yes	No	No	Yes	No	No		No

Tomasulo With Reorder Buffer-Cycle 5

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
1	Add1	Yes	SUB	Regs[F6]	Mem[45+regs[R3]]			#4
0	Add2	No						
0	Add3	No						
9	Mult1	Yes	Mult	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Station

LD F6, 34(R2)
LD F2, 45(R3)
MULT F0, F2, F4
SUBD F8, F6, F2
DIVD F10, F0, F6
ADDD F6, F8, F2

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
Head	3	Yes	MULT F0,F2,F4	Ex1	F0	
	4	Yes	SUBD F8,F6,F2	Ex1	F8	
Tail	5	Yes	DIVD F10,F0,F6	Issue	F10	
	6					
	7					
	8					
	9					
	10					

Reorder Buffer

	Busy	Address
Load1	No	
Load2	No	
Load3		

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
5	Reorder#	#3				#4	#5			
	Busy	Yes	No	No	No	Yes	Yes	No		No

LD： 1 cycles MULT: 10 cycles SUBD/ADDD: 2cycles DIVD: 40 cycles

Tomasulo With Reorder Buffer-Cycle 6

LD： 1 cycles MULT: 10 cycles SUBD/ADDD: 2cycles DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	Yes	SUB	Regs[F6]	Mem[45+regs[R3]]			#4
0	Add2	Yes	ADD		Regs[F2]	#4		#6
0	Add3	No						
8	Mult1	Yes	MULT	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Station

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
Head	3	Yes	MULT F0,F2,F4	Ex2	F0	
	4	Yes	SUBD F8,F6,F2	Ex2	F8	
	5	Yes	DIVD F10,F0,F6	Issue	F10	
Tail	6	Yes	ADDD F6,F8,F2	Issue	F6	
	7					
	8					
	9					
	10					

Reorder Buffer

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
6	Reorder#	#3			#6	#4	#5			
	Busy	Yes	No	No	Yes	Yes	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 7

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
2	Add2	Yes	ADD	#4	Regs[F2]			#6
0	Add3	No						
7	Mult1	Yes	MULT	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Station

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
Head	3	Yes	MULT F0,F2,F4	Ex3	F0	
	4	Yes	SUBD F8,F6,F2	Write	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Issue	F10	
Tail	6	Yes	ADDD F6,F8,F2	Issue	F6	
	7					
	8					
	9					
	10					

Reorder Buffer

	Load1	Busy	Address
	Load2	No	
	Load3	No	

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
7	Reorder#	#3			#6	#4	#5			
	Busy	Yes	No	No	Yes	Yes	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 8

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
1	Add2	Yes	ADD	#4	Regs[F2]			#6
0	Add3	No						
6	Mult1	Yes	MULT	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Station

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
Head	3	Yes	MULT F0,F2,F4	Ex4	F0	
	4	Yes	SUBD F8,F6,F2	Write	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Issue	F10	
Tail	6	Yes	ADDD F6,F8,F2	Ex1	F6	
	7					
	8					
	9					
	10					

Reorder Buffer

	Busy	Address
Load1	No	
Load2	No	
Load3		

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
8	Reorder#	#3			#6	#4	#5			
	Busy	Yes	No	No	Yes	Yes	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 9

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	Yes	ADD	#4	Regs[F2]			#6
0	Add3	No						
5	Mult1	Yes	MULT	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Station

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
Head	3	Yes	MULT F0,F2,F4	Ex5	F0	
	4	Yes	SUBD F8,F6,F2	Write	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Issue	F10	
Tail	6	Yes	ADDD F6,F8,F2	Ex2	F6	
	7					
	8					
	9					
	10					

Reorder Buffer

	Load1	Busy	Address
	Load2	No	
	Load3	No	

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
9	Reorder#	#3			#6	#4	#5			
	Busy	Yes	No	No	Yes	Yes	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 10

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
4	Mult1	Yes	MULT	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Station

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
Head	3	Yes	MULT F0,F2,F4	Ex6	F0	
	4	Yes	SUBD F8,F6,F2	Write	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Issue	F10	
Tail	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

	Busy	Address
Load1	No	
Load2	No	
Load3		

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
10	Reorder#	#3			#6	#4	#5			
	Busy	Yes	No	No	Yes	Yes	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 11

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
3	Mult1	Yes	MULT	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Station

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
Head	3	Yes	MULT F0,F2,F4	Ex7	F0	
	4	Yes	SUBD F8,F6,F2	Write	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Issue	F10	
Tail	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
11	Reorder#	#3			#6	#4	#5			
	Busy	Yes	No	No	Yes	Yes	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 12

LD： 1 cycles MULT: 10 cycles SUBD/ADDD: 2cycles DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
2	Mult1	Yes	MULT	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Station

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
Head	3	Yes	MULT F0,F2,F4	Ex8	F0	
	4	Yes	SUBD F8,F6,F2	Write	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Issue	F10	
Tail	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

	Busy	Address
Load1	No	
Load2	No	
Load3		

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
12	Reorder#	#3			#6	#4	#5			
	Busy	Yes	No	No	Yes	Yes	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 13

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
1	Mult1	Yes	MULT	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Station

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
Head	3	Yes	MULT F0,F2,F4	Ex9	F0	
	4	Yes	SUBD F8,F6,F2	Write	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Issue	F10	
Tail	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
13	Reorder#	#3			#6	#4	#5			
	Busy	Yes	No	No	Yes	Yes	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 14

LD： 1 cycles MULT: 10 cycles SUBD/ADDD: 2cycles DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	Yes	MULT	Mem[45+Regs[R3]]	Regs[F4]			#3
0	Mult2	Yes	DIV		Regs[F6]	#3		#5

Reservation Station

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
Head	3	Yes	MULT F0,F2,F4	Ex10	F0	
	4	Yes	SUBD F8,F6,F2	Write	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Issue	F10	
Tail	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

	Busy	Address
Load1	No	
Load2	No	
Load3		

Reorder Buffer

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
14	Reorder#	#3			#6	#4	#5			
	Busy	Yes	No	No	Yes	Yes	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 15

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
40	Mult2	Yes	DIV	#2*Regs[F4]	Regs[F6]			#5

Reservation Station

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
Head	3	Yes	MULT F0,F2,F4	Write	F0	#2*F4
	4	Yes	SUBD F8,F6,F2	Write	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Issue	F10	
Tail	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

	Busy	Address
Load1	No	
Load2	No	
Load3		

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
15	Reorder#	#3			#6	#4	#5			
	Busy	Yes	No	No	Yes	Yes	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 16

LD： 1 cycles MULT: 10 cycles SUBD/ADDD: 2cycles DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
39	Mult2	Yes	DIV	#2*Regs[F4]	Regs[F6]			#5

Reservation Station

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
	3	Yes	MULT F0,F2,F4	Commit	F0	#2*F4
Head	4	Yes	SUBD F8,F6,F2	Write	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Ex1	F10	
Tail	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

	Busy	Address
Load1	No	
Load2	No	
Load3		

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
16	Reorder#				#6	#4	#5			
	Busy	No	No	No	Yes	Yes	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 17

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
38	Mult2	Yes	DIV	#2*Regs[F4]	Regs[F6]			#5

Reservation Station

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
	3	Yes	MULT F0,F2,F4	Commit	F0	#2*F4
	4	Yes	SUBD F8,F6,F2	Commit	F8	F6-#2
Head	5	Yes	DIVD F10,F0,F6	Ex2	F10	
Tail	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
17	Reorder#				#6		#5			
	Busy	No	No	No	Yes	No	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 18

LD： 1 cycles　　MULT: 10 cycles　　SUBD/ADDD: 2cycles　　DIVD: 40 cycles

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
37	Mult2	Yes	DIV	#2*Regs[F4]	Regs[F6]			#5

Reservation Station

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
	3	Yes	MULT F0,F2,F4	Commit	F0	#2*F4
	4	Yes	SUBD F8,F6,F2	Commit	F8	F6-#2
Head	5	Yes	DIVD F10,F0,F6	Ex3	F10	
Tail	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
18	Reorder#				#6		#5			
	Busy	No	No	No	Yes	No	Yes	No		No

Continue……37 Cycles

Tomasulo With Reorder Buffer-Cycle 55

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
0	Mult2	Yes	DIV	#2*Regs[F4]	Regs[F6]			#5

Reservation Station

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
	3	Yes	MULT F0,F2,F4	Commit	F0	#2*F4
	4	Yes	SUBD F8,F6,F2	Commit	F8	F6-#2
Head	5	Yes	DIVD F10,F0,F6	Ex40	F10	#3/F6
Tail	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

Cycle		F0	F2	F4	F6	F8	F10	F12	……	F30
54	Reorder#				#6		#5			
	Busy	No	No	No	Yes	No	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 56

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
0	Mult2	No						

Reservation Station

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
	3	Yes	MULT F0,F2,F4	Commit	F0	#2*F4
	4	Yes	SUBD F8,F6,F2	Commit	F8	F6-#2
Head	5	Yes	DIVD F10,F0,F6	Write	F10	#3/F6
Tail	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

	Busy	Address
Load1	No	
Load2	No	
Load3		

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
56	Reorder#				#6		#5			
	Busy	No	No	No	Yes	No	Yes	No		No

Tomasulo With Reorder Buffer-Cycle 57

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
0	Mult2	No						

Reservation Station

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
	3	Yes	MULT F0,F2,F4	Commit	F0	#2*F4
	4	Yes	SUBD F8,F6,F2	Commit	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Commit	F10	#3/F6
Head	6	Yes	ADDD F6,F8,F2	Write	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

	Busy	Address
Load1	No	
Load2	No	
Load3		

Cycle		F0	F2	F4	F6	F8	F10	F12	……	F30
57	Reorder#				#6					
	Busy	No	No	No	Yes	No	No	No		No

Tomasulo With Reorder Buffer-Cycle 58

Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Dest
0	Add1	No						
0	Add2	No						
0	Add3	No						
0	Mult1	No						
0	Mult2	No						

Reservation Station

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Dest.	Value
	1	Yes	LD F6,34(R2)	Commit	F6	Mem[load1]
	2	Yes	LD F2,45(R3)	Commit	F2	Mem[load2]
	3	Yes	MULT F0,F2,F4	Commit	F0	#2*F4
	4	Yes	SUBD F8,F6,F2	Commit	F8	F6-#2
	5	Yes	DIVD F10,F0,F6	Commit	F10	#3/F6
Head	6	Yes	ADDD F6,F8,F2	Commit	F6	#4+F2
	7					
	8					
	9					
	10					

Reorder Buffer

Cycle

		F0	F2	F4	F6	F8	F10	F12	……	F30
58	Reorder#									
	Busy	No	No	No	No	No	No	No		No

Tomasulo With Reorder Buffer-Summary

Instruction	Issue	Exec Comp	WriteBack	Commit
LD F6,34(R2)	1	2	3	4
LD F2,45(R3)	2	3	4	5
MULT F0,F2,F4	3	5~14	15	16
SUBD F8,F6,F2	4	5~6	7	17
DIVD F10,F0,F6	5	16~55	56	57
ADDD F6,F8,F2	6	8~9	10	58

顺序发射、乱序执行、乱序完成、顺序提交

两种Tomasulo算法比较（三阶段vs.四阶段）

Loop	L.S F0, 0(R1)
	L.S F1, 0(R2)
	ADD.S F2, F1, F0
	S.S F2, 0(R1)
	ADDI R1,R1, #4
	ADDI R2,R2, #4
	SUBI R3,R3,#1
	BNEZ R3, Loop

- **假设：**
  - Load和store部件：计算访存地址 需要 2 cycle；对Cache访问 需要 1个cycle
  - 浮点ADD执行：需要6个cycle
  - Store操作内部分解为两个操作操作：S.S-A 计算访存地址；S.S-D 对Cache访问
  - 其他整型类执行：需要2个cycle

Tomsasulo算法执行示例（无预测）

		Issue	Exe Start	Exe End	Cache	CDB	备注
I1	L.S F0, 0(R1)	1	2	3	（4）	（5）	
I2	L.S F1, 0(R2)	2	3	4	（5）	（6）	
I3	ADD.S F2,F1,F0	3	7	12	---	（13）	等待F1
I4	S.S-A F2, 0(R1)	4	5	6	---	---	
I5	S.S-D F2,0(R1)	5	14	15	（16）	---	等待F2
I6	ADDI R1,R2, #4	6	7	8	---	（9）	
I7	ADDI R2, R2,#4	7	8	9	---	（10）	
I8	SUBI R3, R3, #1	8	9	10	---	（11）	
I9	BNEZ R3, Loop	9	12	13	---	（14）	等待R3的值
I10	L.S F0, 0(R1)	15	16	17	（18）	（19）	等待I9
I11	L.S F1, 0(R2)	16	17	18	（19）	（20）	
I12	ADD.S F2,F1,F0	17	21	26	---	（27）	等待F1

Tomsasulo算法执行示例（有预测）

		Issue	Exe Start	Exe End	Cache	CDB	Commit	备注
I1	L.S F0, 0(R1)	1	2	3	4	（5）	6	
I2	L.S F1, 0(R2)	2	3	4	5	（6）	7	
I3	ADD.S F2,F1,F0	3	7	12	---	（13）	14	等待F1
I4	S.S-A F2, 0(R1)	4	5	6	---	---		
I5	S.S-D F2,0(R1)	5	14	15	16	---	（17）	等待F2
I6	ADDI R1,R2, #4	6	7	8	---	（9）	（18）	
I7	ADDI R2, R2,#4	7	8	9	---	（10）	（19）	
I8	SUBI R3, R3, #1	8	9	10	---	（11）	（20）	
I9	BNEZ R3, Loop	9	14	15	---	（16）	（21）	等待R3的值，若第12拍或第13拍进入EXE段，则WR阶段（CDB争用）分别与I10，I11存在冲突
I10	L.S F0, 0(R1)	10	11	12	13	（14）	（22）	
I11	L.S F1, 0(R2)	11	12	13	14	（15）	（23）	
I12	ADD.S F2,F1,F0	12	16	21	---	（22）	（24）	等待F1

5.4 推断执行

支持推断执行的Tomasulo

1. 带有ROB的机器结构
2. 四阶段算法描述

代码执行示例

1. 简单代码示例
2. 推断执行示例

Tomasulo小结

1. ROB的作用
2. 动态内存歧义消除

使用ROB保持机器的精确状态

- **ROB维持了机器的精确状态，允许投机（推测）执行**
  - 直到确认无异常 然后进入提交阶段
  - 直到确定分支预测正确进入提交阶段
  - 如果有异常或预测错误
    - 刷新ROB、RS和寄存器结果状态表
- **存储器操作使用类似的方法**
  - Memory Ordering Buffer（MOB）
    - Store操作的结果先存放到MOB中，然后提交阶段按存储操作的程序序提交

Memory Disambiguation：处理对存储器引用的数据相关

- **Question: 给定一个指令序列，store，load 这两个操作是否有关？即下列代码是否有相关问题?**

  Eg: st 0(R2),R5

- **.....**

  ld R6,0(R3)

- **我们是否可以较早启动ld?**
  - Store的地址可能会延迟很长时间才能得到.
  - 我们也许想在同一个周期开始这两个操作的执行.
- **两种方法:**
  - No Speculation: 不进行load操作，直到我们确信地址 0(R2) ≠ 0(R3)
  - Speculation: 我们可以假设他们相关还是不相关 (called “dependence speculation”)，如果推测错误通过ROB来修正
- **参考书：Gonzalez, A., et al. (2011). “Processor Microarchitecture: An Implementation Perspective.” Synthesis Lectures on Computer Architecture #12, Morgan & Claypool Publishers**

Memory Disambiguation

TABLE 6.1: Memory disambiguation schemes.

NAME	SPECULATIVE	DESCRIPTION
Total Ordering	No	All memory accesses are processed in order.
Partial Ordering	No	All stores are processed in order, but loads execute out of order as long as all previous stores have computed their address.
Load Ordering Store Ordering	No	Execution between loads and stores is out of order, but all loads execute in order among them, and all stores execute in order among them.
Store Ordering	Yes	Stores execute in order, but loads execute completely out of order.

- **非投机方式的基本原则：当前存储器指令之前的store指令计算存储器地址后，才能执行当前的存储器操作**

Summary-Tomasulo小结 #1/3

- **Reservations stations: 寄存器重命名，缓冲源操作数**
  - 避免寄存器成为瓶颈
  - 避免了Scoreboard中无法解决的 WAR, WAW hazards
  - 允许硬件做循环展开
  - 不限于基本块(快速解决控制相关)
- **Reorder Buffer:**
  - 提供了撤销指令运行的机制
  - 指令以发射序存放在ROB中
  - 指令顺序提交
- **分支预测对提高性能是非常重要的**
  - 推断执行: 在控制相关还没有解决情况下，就开始执行
  - 推断执行利用了ROB撤销指令执行的机制
    - 处理预测错误时，撤销 推测执行的指令
  - 基于BHT的分支预测技术
  - 基于BTB的分支预测技术

Summary-Tomasulo小结 #2/3

- **贡献**
  - Dynamic scheduling
  - Register renaming
  - Load/store disambiguation
- **360/91 后 Pentium II; PowerPC 604; MIPS R10000; HP-PA 8000; Alpha 21264使用这种技术**
- 不足之处：
  - Too many value copy operations
    - Register File →RS→ROB→Register File
  - Too many muxes/busses (CDB)
    - Values are from everywhere to everywhere else!
  - Reservation Stations mix values(data) and tags (control)
    - Slow down max clock frequency

Summary-Tomasulo小结 #3/3

- **存储器访问的冲突消解**
  - 非投机方式的冲突消解
    - Total Ordering
    - Partial Ordering
      - Load指令前的store指令已经完成了地址计算，有可能乱序执行存储器load操作
    - Load Ordering, Store Ordering
      - Load指令前的存储器访问指令已经完成了地址计算，load队头的load操作有可能在store指令之前执行访存操作。
  - 投机方式的执行
    - Store Ordering
    - 假设Load操作与之前未计算出有效地址的store操作无关。
- **问题：给出四种访问方式挖掘并行性的能力排序。**

Acknowledgements

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