计算机组成
(2022秋)

计算机组成课程组
（刘旭东、高小鹏、肖利民、栾钟治、万寒）
北京航空航天大学计算机学院中德所
栾钟治

➢1

习题5——单周期处理器

- 已发布
  - Spoc平台
- 11月18日截止
  - 23:55
- 在sopc提交
  - 电子版，可手写

➢2

第六讲 MIPS处理器设计

➢3

流水线设计的一般方法

- 单周期数据通路和控制信号为基础
  - 先不考虑转发、暂停和分支等
- 考虑转发
  - 增加转发控制单元，处理ALU和MEM转发
- 考虑因Load导致的数据冒险
  - 增加冒险检测单元
- 考虑分支
  - 缩短分支延迟，分支比较前移

➢4

流水线设计的工程化方法

- 集中式译码与分布式译码
- 基础指令集与流水线设计规划
- 无转发数据通路构造方法
- 功能部件控制信号构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

➢5

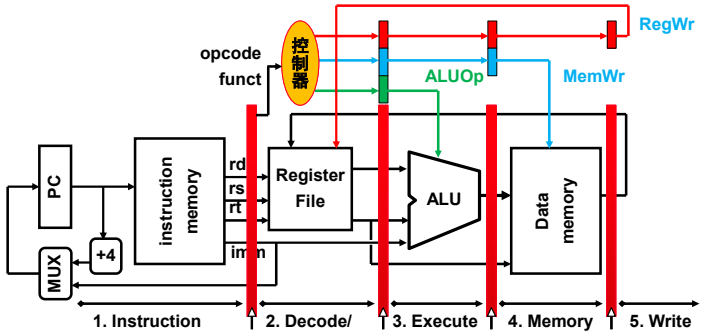
流水线设计的工程化方法

- 集中式译码与分布式译码
- 基础指令集与流水线设计规划
- 无转发数据通路构造方法
- 功能部件控制信号构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

➢6

集中式控制器与分布式控制器

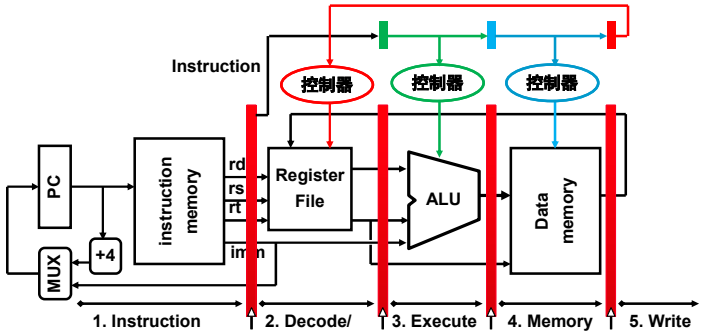
- 集中式控制器
  - 控制器只在ID阶段
  - 控制器产生全部的译码信号
  - 流水所有的译码信号



➢7

集中式控制器与分布式控制器

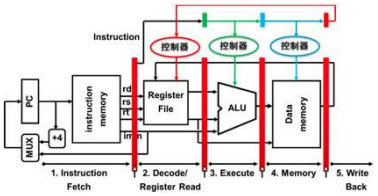
- 分布式控制器
  - 控制器分布在多个流水线阶段
  - 每级控制器只产生该级功能部件相关的译码信号
  - 流水指令

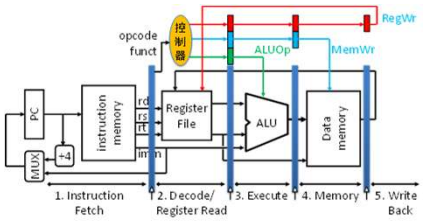


➢8

集中式控制器与分布式控制器

- 资源使用率：集中式控制器
- 结构简洁性：分布式控制器
- 项目维护性：分布式控制器
- 代码可读性：分布式控制器





➢9

流水线设计的工程化方法

- 集中式译码与分布式译码
- 基础指令集与流水线设计规划
- 无转发数据通路构造方法
- 功能部件控制信号构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

➢10

基础指令集与标准流水线

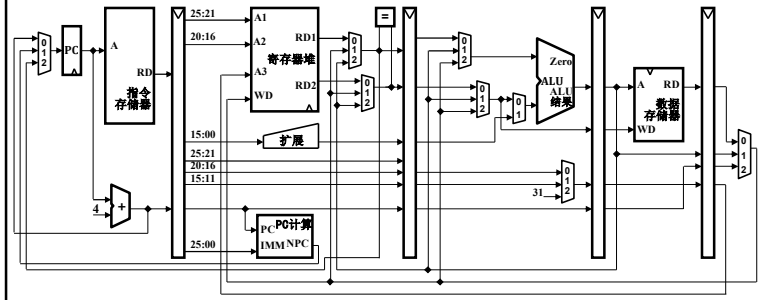
- 指令集
  - lw，sw，addu，subu，ori，lui，beq，j，jal，jalr
- 典型指令；可以支持大多数程序需求
- jal，jalr：涉及2个写入操作，PC写入，RF写入
  - 比较特殊的指令

LW
SW
ADDU
SUBU
ORI
LUI
BEQ
J
JAL
JALR

➢11

基础指令集与标准流水线

- 流水线：以性能为目标的标准流水线
  - 数据冒险：转发、暂停
  - 控制冒险：分支比较前移、转发、暂停



➢12

三控制器架构

- 功能部件控制器：就是书中的控制器
  - 译码指令，控制各个功能部件
  - 属于功能性设计范畴：即与指令的功能相关，与性能无关
    - 无论单周期还是流水线，设计思路相同
- 暂停控制器
  - 将IF/ID指令与前序指令（位于后序流水段）分析，决定是否暂停
  - 属于性能设计范畴
- 转发控制器
  - 分析各级指令的相关性，决定如何转发
  - 属于性能设计范畴
- 三控制器架构特点
  - 结构清晰，易于理解
  - 暂停控制器、转发控制器：独立，相互不干扰

流水线功能部件

- 延用单周期数据通路功能部件
- 按流水段分类，便于理解和记忆
- RF在2个阶段均被使用
  - 译码/读操作数阶段；结果回写寄存器阶段

阶段	部件	输入	输出	描述
取指令	PC	D	Q	程序计数器
	ADD4	PC，+4	PC4	完成PC+4
	IM	A	D	指令存储器
译码/读操作数	RF	A1，A2，A3，WD	RD1，RD2	寄存器堆
	EXT	I16	IMM32	立即数扩展
	NPC	PC，I26	NextPC	为B类/J计算下条地址
	CMP	D1，D2	Result	比较2个数
计算	ALU	A，B	ALU	算数/逻辑运算
访存	DM	A，WD	RD	数据存储器
回写	RF			

流水线寄存器

- 需要设置4级流水线寄存器
  - 5级流水线的最后一级寄存器为RF
- 标记X：代表对应流水级需要设置相应寄存器
  - IR：4个流水级均需要
  - AO：仅M级和W级需要

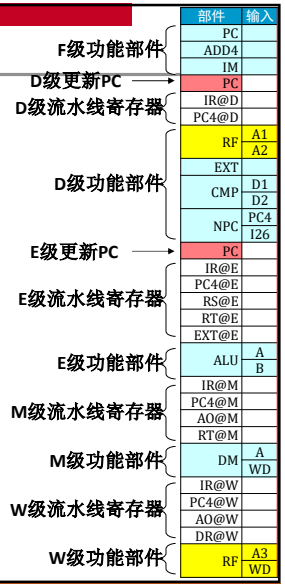
名称	功能	D级 IF/ID	E级 ID/EX	M级 EX/MEM	W级 MEM/WB
IR	传递指令	X	X	X	X
PC4	下一条指令地址	X	X	X	X
RS	RF的RS值(RD1输出)		X		
RT	RF的RT值(RD2输出)		X	X	
EXT	扩展后的32位立即数		X		
AO	ALU计算结果			X	X
DR	DM读出结果				X

流水线设计的工程化方法

- 集中式译码与分布式译码
- 基础指令集与流水线设计规划
- 无转发数据通路构造方法
- 功能部件控制信号构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

流水线数据通路构造表格

- 每级由寄存器和功能部件组成
  - 按流水线5个阶段划分
- X@Y：代表Y阶段的X寄存器
  - IR@W：W级的IR
- PC：出现在3个阶段
  - F级：取指令
  - D级：保存PC+4
  - E级：保存B/J/JAL/JALR的值
- RF：出现在2个阶段
  - D阶段：准备操作数
  - W阶段：回写结果



部件	输入
PC	
ADD4	
IM	
PC	
IR@D	
PC4@D	
RF	A1
	A2
EXT	
CMP	D1
	D2
NPC	PC4
	I26
PC	
IR@E	
PC4@E	
RS@E	
RT@E	
EXT@E	
ALU	A
	B
IR@M	
PC4@M	
AO@M	
RT@M	
DM	A
	WD
IR@W	
PC4@W	
AO@W	
DR@W	
RF	A3
	WD

S1：LW的数据通路

- ❖根据RTL描述建立各级流水线寄存器、功能部件间连接关系
  - ➢LW：5级
- ❖IR必填
  - ➢采用分布式译码
- ❖指令不涉及的不需要填：如PC4
- ❖X[y]：代表X部件的y域
- ❖IR@D[i16]：D级IR的16位立即数

部件	输入	LW
PC		
ADD4		PC
IM		PC
PC		ADD4
IR@D		IM
PC4@D		
RF	A1	IR@D[rs]
	A2	
EXT		IR@D[i16]
NPC	PC4	
	I26	
PC		
IR@E		IR@D
PC4@E		
RS@E		RF.RD1
RT@E		
EXT@E		EXT
ALU	A	RS@E
	B	EXT@E
IR@M		IR@E
PC4@M		
AO@M		ALU
RT@M		
DM	A	AO@M
	WD	
IR@W		IR@M
PC4@W		
AO@W		
DR@W		DM
RF	A3	IR@W[rt]
	WD	DR@W

S1：全部指令的数据通路

部件	输入	LW	SW	ADDU	SUBU	ORI	BEQ	J	JAL	JALR
PC										
ADD4		PC	PC	PC	PC	PC	PC	PC	PC	PC
IM		PC	PC	PC	PC	PC	PC	PC	PC	PC
PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4
IR@D		IM	IM	IM	IM	IM	IM	IM	IM	IM
PC4@D							ADD4	ADD4	ADD4	ADD4
RF	A1	IR@D[rs]	IR@D[rs]	IR@D[rs]	IR@D[rs]	IR@D[rs]	IR@D[rs]	IR@D[rs]		IR@D[rs]
	A2			IR@D[rt]	IR@D[rt]		IR@D[rt]	IR@D[rt]		
EXT		IR@D[i16]	IR@D[i16]			IR@D[i16]				
CMP	D1						RF.RD1			
	D2						RF.RD2			
NPC	PC4						PC4@D	PC4@D	PC4@D	
	I26						IR@D[i16]	IR@D[i26]	IR@D[i26]	
PC							NPC	NPC	NPC	RF.RD1
IR@E		IR@D	IR@D	IR@D	IR@D	IR@D			IR@D	IR@D
PC4@E									PC4@D	PC4@D
RS@E		RF.RD1	RF.RD1	RF.RD1	RF.RD1	RF.RD1				
RT@E			RF.RD2	RF.RD2	RF.RD2					
EXT@E		EXT	EXT			EXT				
ALU	A	RS@E	RS@E	RS@E	RS@E	RS@E				
	B	EXT@E	EXT@E	RT@E	RT@E	EXT@E				
IR@M		IR@E	IR@E	IR@E	IR@E	IR@E			IR@E	IR@E
PC4@M									PC4@E	PC4@E
AO@M		ALU	ALU	ALU	ALU	ALU				
RT@M			RT@E							
DM	A	AO@M	AO@M							
	WD		RT@M							
IR@W		IR@M		IR@M	IR@M	IR@M			IR@M	IR@M
PC4@W									PC4@M	PC4@M
AO@W				AO@M	AO@M	AO@M				
DR@W		DM								
RF	A3	IR@W[rt]		IR@W[rd]	IR@W[rd]	IR@W[rt]			0x1F	IR@W[rd]
	WD	DR@W		AO@W	AO@W	AO@W			PC4@W	PC4@W

S2：综合全部指令的数据通路

- ❖水平方向归并
  - ➢去除冗余输入来源
- ❖在每个输入来源个数大于1的输入端前增加1个MUX
  - ➢注意：同时需要产生相应的控制信号
- ❖特例：NPC的i16和i26归并为i26

部件	输入	输入来源			MUX	控制
PC						
ADD4		PC				
IM		PC				
PC		ADD4	NPC	RF.RD1	M1	PCSel
IR@D		IM				
PC4@D		ADD4				
RF	A1	IR@D[rs]				
	A2	IR@D[rt]				
EXT		IR@D[i16]				
CMP	D1	RF.RD1				
	D2	RF.RD2				
NPC	PC4	PC4@D				
	I26	IR@D[i26]				
IR@E		IR@D				
PC4@E		PC4@D				
RS@E		RF.RD1				
RT@E		RF.RD2				
EXT@E		EXT				
ALU	A	RS@E				
	B	EXT@E	RT@E		M2	BSel
IR@M		IR@E				
PC4@M		PC4@E				
AO@M		ALU				
RT@M		RT@E				
DM	A	AO@M				
	WD	RT@M				
IR@W		IR@M				
PC4@W		PC4@M				
AO@W		AO@M				
DR@W		DM				
RF	A3	IR@W[rt]	IR@W[rd]	0x1F	M3	WRSel
	WD	DR@W	AO@W	PC4@W	M4	WDSel

流水线设计的工程化方法

- 集中式译码与分布式译码
- 基础指令集与流水线设计规划
- 无转发数据通路构造方法
- 功能部件控制信号构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

功能部件控制信号构造方法

- 控制信号产生基本原理：与单周期相同
- 分歧点：集中式译码？分布式译码？
  - 集中式：
    - 与单周期控制器设计完全相同
    - 流水控制信号
  - 分布式：多个小控制器
    - 每个小控制器的设计思路与单周期相同
    - 流水指令

流水线设计的工程化方法

- 集中式译码与分布式译码
- 基础指令集与流水线设计规划
- 无转发数据通路构造方法
- 功能部件控制信号构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

数据冒险：需求与供给能否匹配？

- 需求者：需要引用reg值的组件
  - 由于reg值最终被某个组件使用，因此那个组件才是需求者
  - 例如：所有运算类指令的需求在E级的ALU
  - 例如：j指令不需要读取任何GPR，因此j指令没有需求
- 供给者：保存有reg新结果的流水线寄存器
  - 例如：所有运算类指令的供给者是EX/MEM、MEM/WB
  - 例如：load类指令的供给者是MEM/WB
- 数据冒险可以转化为：需求与供给的匹配
  - 无法匹配：暂停
  - 可以匹配：转发

需求者的最晚时间模型

- T_{use}(time-to-use)：指令进入IF/ID寄存器后，其后的某个功能部件再经过多少cycle就必须要使用相应的寄存器值
  - 特点1：是读取操作数的时间上限
  - 特点2：同一条指令可以有2个不同的T_{use}
  - 例如，R型计算类指令的T_{use}为1
    - rs/rt值：最晚被ID/EX寄存器驱动
  - 例如，store类指令的T_{use}分别为1和2
    - rs值：最晚被ID/EX寄存器驱动
    - rt值：最晚被EX/MEM寄存器驱动

供给者的最早时间模型

- T_{new}(time-to-new)：位于ID/EX及其后各流水段的指令，再经过多少周期能够产生要写入寄存器的结果
  - 特点1：动态值，随着指令的流动，该值在不断减小，直至0
  - 特点2：一条指令可以有多个不同的T_{new}
  - 例如，R型计算类指令的T_{new}为1或0
    - 1：指令位于ID/EX，ALU正在计算。
    - 0：指令位于EX/MEM和MEM/WB
  - 例如，load类指令的T_{new}为2，1，0
    - 2：指令位于ID/EX，尚未读取存储器。
    - 1：指令位于EX/MEM，正在读取存储器
    - 0：指令位于MEM/WB，包含了结果

数据冒险的策略分析

- $T_{new} = 0$：表明结果已经产生
  - 指令位于MEM/WB：那么虽然结果尚未最终写入RF，但RF设计使得W结果可以被正确的读出，因此无需任何操作
  - 指令位于其他位置：通过转发解决数据相关
- $T_{new} \neq 0$：表明结果尚未产生
  - $T_{new} > T_{use}$：不可能及时供给数据，只能暂停流水线
  - $T_{new} \leq T_{use}$：由于结果产生时间短于读取时间，因此当结果产生后可以通过转发解决数据冒险
- 暂停：$T_{new} > T_{use}$
- 转发：$T_{new} = 0$&指令不在MEM/WB 或 $T_{new} \leq T_{use}$

数据冒险的策略分析

- 暂停：由于在IF/ID就能决定是否需要暂停，因此分析量少
  - 只需将指令的T_{use}与各级的T_{new}进行对比即可决定是否需要暂停
- 转发：由于在ID级、EX级、MEM级均涉及操作数读取，因此分析量大
  - 需要将各级指令与其后的各级指令进行对比
- 思路：先解决暂停，再解决转发
  - 先易后难
  - 去除暂停部分后，有助于减少转发的分析量

流水线设计的工程化方法

- 集中式译码与分布式译码
- 形式建模综合方法概述
- 基础指令集与流水线设计规划
- 无转发数据通路构造方法
- 功能部件控制信号构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

构造T_{use}表和T_{new}表

ADD
SUB
andi
ori
LW
SW
BEQ

- 示例指令集
  - add，sub：cal_r类，即R型计算类指令
  - andi，ori：cal_i类，即I型计算类指令
  - beq：b_type类
  - lw：ld类
  - sw：st类
- 会产生结果的指令：cal_r类，cal_i类，load类
- 用指令分类可以大幅度简化分析工作量

```
cal_r = add  + sub + or   + …
cal_i = addi + ori + andi + …
ld    = lw   + lb  + lh   + …
st    = sw   + sb  + …
```


构造Tuse表和Tnew表

- Tuse表：以指令位于IF/ID来分析
  - 流水线在指令被存储在IF/ID后就决定是否需要暂停
- Tnew表：只需分析处于ID/EX和EX/MEM这2种情况
  - IF/ID：无任何结果
  - MEM/WB：如果结果到达该阶段，则通过RF设计可以消除数据冒险

IF/ID当前指令		
指令类型	源寄存器	Tuse
beq	rs/rt	0
cal_r	rs/rt	1
cal_i	rs	1
load	rs	1
store	rs	1
store	rt	2

ID/EX (Tnew)			EX/MEM (Tnew)			MEM/WB (Tnew)		
cal_r 1/rd	cal_i 1/rt	load 2/rt	cal_r 0/rd	cal_i 0/rt	load 1/rt	cal_r 0/rd	cal_i 0/rt	load 0/rt

构造阻塞矩阵

- 凡是$T_{new} > T_{use}$的指令序列，都需要阻塞
- 示例
  - 序列1 cal_r – beq：由于cal_r需要1个cycle后才能得到结果，而beq现在就需要读取寄存器，因此只能暂停
  - 序列2 load – store：store要读取的rs将在1个cycle后必须使用，而位于ID/EX的load必须经过2个cycle后才能读出DM的数据，因此只能暂停

IF/ID当前指令			ID/EX (T_{new})			EX/MEM (T_{new})
指令类型	源寄存器	T_{use}	cal_r 1/rd	cal_i 1/rt	load 2/rt	load 1/rt
beq	rs/rt	0	暂停	暂停	暂停	暂停
cal_r	rs/rt	1			暂停	
cal_i	rs	1			暂停	
load	rs	1			暂停	
store	rs	1			暂停	

暂停控制信号

❖建立分类指令的暂停条件

```
stall_beq = beq_D & (cal_r_E & ((IR_D.rs==IR_E.rd) |
                                (IR_D.rt==IR_E.rd)) |
          。。。
stall_cal_r = cal_r_D & load_E & ((IR_D.rs==IR_E.rt) |
                                  (IR_D.rt==IR_E.rt) )
```

❖建立最终的暂停条件

```
stall = stall_beq + …
```

❑ 建立控制信号

IF/ID当前指令			ID/EX (T_{new})			EX/MEM (T_{new})
指令类型	源寄存器	T_{use}	cal_r 1/rd	cal_i 1/rt	*load* 2/*rt*	load 1/rt
beq	*rs/rt*	0	暂停	暂停	暂停	暂停
cal_r	rs/rt	1			*暂停*	
cal_i	rs	1			暂停	
load	rs	1			暂停	
store	rs	1			暂停	

暂停控制信号

❖执行动作：

➢①冻结IF/ID：后续指令继续被保存

➢②清除ID/EX：指令全为0，等价于插入NOP

➢③禁止PC：防止PC继续计数，PC应保持为PC+4

```
IR_D.en  = !stall
IR_E.clr =  stall
PC.en    = !stall
```


流水线设计的工程化方法

❖集中式译码与分布式译码

❖基础指令集与流水线设计规划

❖无转发数据通路构造方法

❖功能部件控制信号构造方法

❖数据冒险的一般性分析方法

❖暂停机制生成方法

❖转发机制生成方法

❖控制冒险处理机制

转发机制生成方法

❖S1：根据Tuse和Tnew构造每个转发MUX

❖S2：构造每个转发MUX的控制信号表达式

根据Tuse和Tnew构造每个转发MUX

- 按照指令分类，梳理指令在各级流水线的rs或rt读需求
- 每个读需求对应1个转发MUX
- 转发MUX的输入0：必然是本级流水线寄存器
  - 对于IF/ID级来说，输入0则来自是RF的输出
- 【建议】命名应遵循一定的规则

流水级	源寄存器	涉及指令
IR@D	rs	beq
	rt	beq
IR@E	rs	cal_r, cal_i, ld, st
	rt	cal_r, st
IR@M	rt	st

MFRSD	ForwardRSD	RF.RD1
MFRTD	ForwardRTD	RF.RD2
MFRSE	ForwardRSE	RS@E
MFRTE	ForwardRTE	RT@E
MFRTM	ForwardRTM	RT@M
转发MUX	控制信号	输入0

➢37

根据Tuse和Tnew构造每个转发MUX

- 用Tnew中剔除非0后的表项，来分析转发MUX的后续输入
  - 注意：并非有N个0项就有N个后续输入

ID/EX (Tnew)			EX/MEM (Tnew)			MEM/WB (Tnew)		
cal_r 1/rd	cal_i 1/rt	load 2/rt	cal_r 0/rd	cal_i 0/rt	load 1/rt	cal_r 0/rd	cal_i 0/rt	load 0/rt

流水级	源寄存器	涉及指令
IF/ID	rs	beq
	rt	beq
ID/EX	rs	cal_r, cal_i, ld, st
	rt	cal_r, st
EX/MEM	rt	st

MFRSD	ForwardRSD	RF.RD1
MFRTD	ForwardRTD	RF.RD2
MFRSE	ForwardRSE	RS@E
MFRTE	ForwardRTE	RT@E
MFRTM	ForwardRTM	RT@M
转发MUX	控制信号	输入0

EX/MEM (Tnew)		MEM/WB (Tnew)		
cal_r 0/rd	cal_i 0/rt	cal_r 0/rd	cal_i 0/rt	load 0/rt

➢38

根据Tuse和Tnew构造每个转发MUX

- 构造每个转发MUX的后续输入
- 示例：MFRSD
  - EX/MEM：cal_r和cal_i指令都是计算类，结果必然由ALU产生，因此均填入AO。即代表MFRSD的输入来自EX/MEM中的AO寄存器
    - AO：代表ALUOut
  - MEM/WB：由于这是最后一级，即所有指令的结果都通过M4(MUX)回写，因此均填入M4。

流水级	源寄存器	涉及指令				EX/MEM (Tnew)		MEM/WB (Tnew)		
						cal_r 0/rd	cal_i 0/rt	cal_r 0/rd	cal_i 0/rt	load 0/rt
IF/ID	rs	beq	MFRSD	ForwardRSD	RF.RD1	AO	AO	M4	M4	M4
	rt	beq	MFRTD	ForwardRTD	RF.RD2					
ID/EX	rs	cal_r, cal_i, ld, st	MFRSE	ForwardRSE	RS@E					
	rt	cal_r, st	MFRTE	ForwardRTE	RT@E					
EX/MEM	rt	st	MFRTM	ForwardRTM	RT@M					
			转发MUX	控制信号	输入0					

➢39

根据Tuse和Tnew构造每个转发MUX

- 根据前例，可以构造出全部的转发MUX
  - 当store类指令位于EX/MEM时，不可能再有同级的指令了
  - 因此有2项空白
- 构造更大指令集时，需求项及供给项可能均需要调整
  - 但由于MIPS的指令功能到格式映射的相对统一，因此调整不会剧烈
  - 再次从一个侧面反映出MIPS指令集设计的水平！

流水级	源寄存器	涉及指令				EX/MEM (Tnew)		MEM/WB (Tnew)		
						cal_r 0/rd	cal_i 0/rt	cal_r 0/rd	cal_i 0/rt	ld 0/rt
IF/ID	rs	beq	MFRSD	ForwardRSD	RF.RD1	AO	AO	M4	M4	M4
	rt	beq	MFRTD	ForwardRTD	RF.RD2	AO	AO	M4	M4	M4
ID/EX	rs	cal_r, cal_i, ld, st	MFRSE	ForwardRSE	RS@E	AO	AO	M4	M4	M4
	rt	cal_r, st	MFRTE	ForwardRTE	RT@E	AO	AO	M4	M4	M4
EX/MEM	rt	st	MFRTM	ForwardRTM	RT@M			M4	M4	M4
			转发MUX	控制信号	输入0					

➢40

根据Tuse和Tnew构造每个转发MUX

- 对于MFRSD来说，其最终有效输入为3个
  - 输入0～RF.RD1；输入1～AO；输入2～M4
- 实现转发MUX时，需要剔除每级中的重复项
- 在表格中保留重复项的目的：有利于建立后续的控制信号方程

MFRSD	ForwardRSD	RF.RD1	AO	AO	M4	M4	M4
MFRTD	ForwardRTD	RF.RD2	AO	AO	M4	M4	M4
MFRSE	ForwardRSE	RS@E	AO	AO	M4	M4	M4
MFRTE	ForwardRTE	RT@E	AO	AO	M4	M4	M4
MFRTM	ForwardRTM	RT@M			M4	M4	M4
转发MUX	控制信号	输入0					

输入	来源
0	RF.RD1
1	AO@M
2	M4@W

MFRSD	ForwardRSD	RF.RD1	AO@M	M4
MFRTD	ForwardRTD	RF.RD2	AO@M	M4
MFRSE	ForwardRSE	RS@E	AO@M	M4
MFRTE	ForwardRTE	RT@E	AO@M	M4
MFRTM	ForwardRTM	RT@M	M4	
转发MUX	控制信号	输入0	输入1	输入2

➢41

数据通路增加转发MUX

- 遍历数据通路的功能部件，找到所有出现rs和rt的需求点
- 注意ALU.B和RT@M，这两个rt需求是相同的！
  - 这意味着它们应该来自同一个转发MUX

MFRSD	RF.RD1	AO@M	M4
MFRTD	RF.RD2	AO@M	M4
MFRSE	RS@E	AO@M	M4
MFRTE	RT@E	AO@M	M4
MFRTM	RT@M	M4	
转发MUX	输入0	输入1	输入2

部件	输入	输入来源			MUX	控制
PC						
ADD4		PC				
IM		PC				
PC		ADD4	NPC	RF.RD1	M1	PCSel
IR@D		IM				
PC4@D		ADD4				
RF	A1	IR@D[rs]				
	A2	IR@D[rt]				
EXT		IR@D[i16]				
CMP	D1	RF.RD1				
	D2	RF.RD2				
NPC	PC4	PC4@D				
	I26	IR@D[i26]				
IR@E		IR@D				
PC4@E		PC4@D				
RS@E		RF.RD1				
RT@E		RF.RD2				
EXT@E		EXT				
ALU	A	RS@E				
	B	EXT@E	RT@E		M2	BSel
IR@M		IR@E				
PC4@M		PC4@E				
AO@M		ALU				
RT@M		RT@E				
DM	A	AO@M				
	WD	RT@M				
IR@W		IR@M				
PC4@W		PC4@M				
AO@W		AO@M				
DR@W		DM				
RF	A3	IR@W[rt]	IR@W[rd]	0x1F	M3	WRSel
	WD	DR@W	AO@W	PC4@W	M4	WDSel

➢42

数据通路增加转发MUX

- 遍历数据通路的功能部件，找到所有出现rs和rt的需求点
- 将对应的输入替换为转发MUX的输出
  - 注意ALU.B和RT@M，这两个rt需求是相同的，因此应该用同一个转发MUX
  - 注意：对于PC，由于构造转发MUX的示例指令集中没有jal/jalr指令，因此缺乏相应的转发MUX与之对应

MFRSD	RF.RD1	AO@M	M4
MFRTD	RF.RD2	AO@M	M4
MFRSE	RS@E	AO@M	M4
MFRTE	RT@E	AO@M	M4
MFRTM	RT@M	M4	
转发MUX	输入0	输入1	输入2

部件	输入	输入来源			MUX	控制
PC						
ADD4		PC				
IM		PC				
PC		ADD4	NPC	RF.RD1	M1	PCSel
IR@D		IM				
PC4@D		ADD4				
RF	A1	IR@D[rs]				
	A2	IR@D[rt]				
EXT		IR@D[i16]				
CMP	D1	MFRSD				
	D2	MFRTD				
NPC	PC4	PC4@D				
	I26	IR@D[i26]				
IR@E		IR@D				
PC4@E		PC4@D				
RS@E		MFRSD				
RT@E		MFRTD				
EXT@E		EXT				
ALU	A	MFRSE				
	B	EXT@E	MFRTE		M2	BSel
IR@M		IR@E				
PC4@M		PC4@E				
AO@M		ALU				
RT@M		MFRTE				
DM	A	AO@M				
	WD	MFRTM				
IR@W		IR@M				
PC4@W		PC4@M				
AO@W		AO@M				
DR@W		DM				
RF	A3	IR@W[rt]	IR@W[rd]	0x1F	M3	WRSel
	WD	DR@W	AO@W	PC4@W	M4	WDSel

➢43

转发机制生成方法

- S1：根据Tuse和Tnew构造每个转发MUX
- S2：构造每个转发MUX的控制信号表达式

➢44

S2：构造每个转发MUX的控制信号表达式

❖控制信号表达式构造的基本思路

- 精确控制每个转发选择
- 所有非转发的条件都用于选择输入0

输入	来源
0	RF.RD1
1	AO@M
2	M4@W

流水级	源寄存器	涉及指令				EX/MEM (Tnew)		MEM/WB (Tnew)		
						cal_r	cal_i 0/rt	cal_r 0/rd	cal_i 0/rt	ld 0/rt
IF/ID	rs	beq	MFRSD	ForwardRSD	RF.RD1	AO	AO	M4	M4	M4
	rt	beq	MFRTD	ForwardRTD	RF.RD2	AO	AO	M4	M4	M4
ID/EX	rs	cal_r, cal_i, ld, st	MFRSE	ForwardRSE	ID/EX.RS	AO	AO	M4	M4	M4
	rt	cal_r, st	MFRTE	ForwardRTE	ID/EX.RT	AO	AO	M4	M4	M4
EX/MEM	rt	st	MFRTM	ForwardRTM	EX/MEM.RT			M4	M4	M4
			转发MUX	控制信号	输入0					

示例：always语句建模MF_RS_D的控制信号表达式

❑ 宏定义提高可读性和一致性

- `define op 31:26
- `define rs 25:21

输入	来源
0	RF.RD1
1	AO@M
2	M4@W

High 优先级 Low

```
assign ForwardRSD =
    IR_D[`op]==beq & cal_r_M & IR_D[`rs]==IR_M[`rd] ? 1 :
    IR_D[`op]==beq & cal_i_M & IR_D[`rs]==IR_M[`rt] ? 1 :
    IR_D[`op]==beq & cal_r_W & IR_D[`rs]==IR_W[`rd] ? 2 :
    IR_D[`op]==beq & cal_i_W & IR_D[`rs]==IR_W[`rt] ? 2 :
    IR_D[`op]==beq & load_W  & IR_D[`rs]==IR_W[`rt] ? 2 :
                                                      0 ;
```

❖顺序代表优先级

➢多条前序指令写同一个寄存器

流水级	源寄存器	涉及指令				EX/MEM (Tnew)		MEM/WB (Tnew)		
						cal_r 0/rd	cal_i 0/rt	cal_r 0/rd	cal_i 0/rt	ld 0/rt
IF/ID	rs	beq	MFRSD	ForwardRSD	RF.RD1	AO	AO	M4	M4	M4
	rt	beq	MFRTD	ForwardRTD	RF.RD2	AO	AO	M4	M4	M4
			转发MUX	控制信号	输入0					

流水线设计的工程化方法

❖集中式译码与分布式译码
❖形式建模综合方法概述
❖基础指令集与流水线设计规划
❖无转发数据通路构造方法
❖功能部件控制信号构造方法
❖数据冒险的一般性分析方法
❖暂停机制生成方法
❖转发机制生成方法
❖控制冒险处理机制

控制冒险处理机制

❖分歧点1：是否实现延迟槽

➢如果实现，需要注意jal及jalr指令应保存PC+8(或者更多，取决于是否前移)

❖分歧点2：执行是否前移至ID阶段

❖课程要求：实现延迟槽，并且前移至ID阶段

延迟槽 / 前移	是	否
是	硬件无需处理 编译调度指令	B类：有条件清除IF/ID J类：无条件清除IF/ID
否		B类：有条件清除IF/ID、ID/EX J类：无条件清除IF/ID、ID/EX、EX/MEM

Q：JAL、JALR的回写寄存器怎么处理呢？

A：视同普通的回写

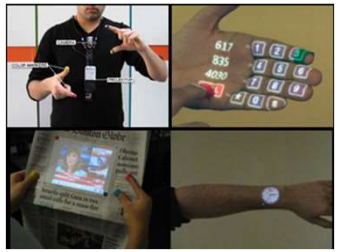
总结

- 流水线设计的复杂性在于对冲突的覆盖性分析
  - 覆盖性分析使得设计与测试均具备了完整的正向设计的理论基础
  - 分析避免了频繁的、无谓的试错
  - 提高开发效率，确保开发正确性
- 教科书中存在的不足
  - 没有覆盖性分析，难以满足大规模指令集的流水线设计与测试需求
  - 没有覆盖性分析，必然遗漏部分数据相关
    - 如lw～sw指令，必须暂停。但事实上可以通过增加转发MUX实现不停顿
    - 如cal～sw指令，未明确指出处理机制
  - RF内部的数据转发语焉不详
    - 内部转发：当读和写同一个寄存器时，读出的数据应该为要写入的数据

中断与异常的处理

输入/输出(I/O)

- 人类与计算机通过I/O交互
- 计算机需要通过I/O获得持久化的存储能力
- 计算机还可以通过I/O展现各种令人惊异的能力



MIT Media Lab
“Sixth Sense”

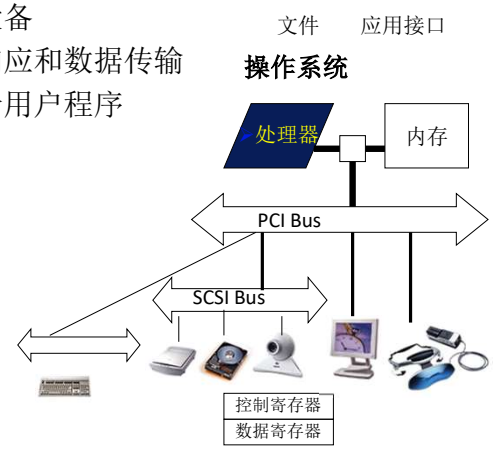
I/O 设备及其速度

- I/O 的速度: 从鼠标到局域网可以跨越7个数量级

设备	行为	交互对象	数据率 (KB/s)
键盘	输入	人类	0.01
鼠标	输入	人类	0.02
音频输出	输出	人类	5.00
软盘	存储	机器	50.00
激光打印机	输出	人类	100.00
磁盘	存储	机器	10,000.00
无线网络	输入或输出	机器	10,000.00
图形化显示	输出	人类	30,000.00
有线局域网	输入或输出	机器	125,000.00

我们需要I/O做什么?

- 连接多种类型设备
- 设备的控制、响应和数据传输
- 提供相关能力给用户程序



➢53

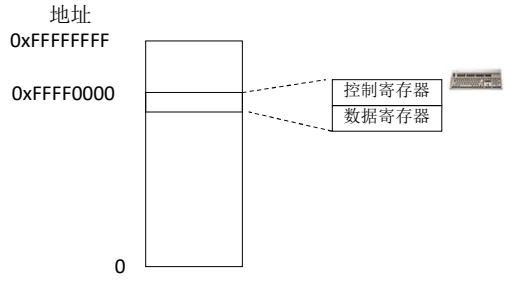
指令集体系结构(ISA)与I/O

- 处理器要为I/O做什么?
  - 输入: 读字节序列
  - 输出: 写字节序列
- 有些处理器有专门的输入输出指令
- 另外的模式(MIPS):
  - 用load实现输入, 用store实现输出（小规模）
  - *Memory Mapped Input/Output*
  - 贡献出一部分地址空间作为输入输出设备的通信通路

➢54

内存映射I/O

- 某确定的地址段并不是常规的内存
- 它们被关联到设备的寄存器



➢55

处理器-I/O的速度不匹配

- 1 GHz 的微处理器每秒可以执行10亿条load或store指令 (4,000,000 KB/s 数据率)
  - I/O设备的数据率范围从 0.01 KB/s到125,000 KB/s
- *输入:* 设备无法及时准备好数据供处理器load
  - 可能还需要等待人类响应
- *输出:* 设备无法及时准备好接收处理器store的数据
- *怎么办?*

➢56

处理器执行动作前先查询状态

- 通向设备的通信通路上通常有2个寄存器:
  - *控制寄存器* 用来确认是否允许读/写 (I/O ready)
  - *数据寄存器* 暂存数据
- 处理器周期性查询（循环）控制寄存器, 等待设备置位 *Ready bit* (0 → 1)
- 接着处理器load(输入)或者store(输出)数据寄存器
  - 重置控制寄存器 Ready bit (1 → 0)
- *这一过程称为"Polling"（轮询）*

I/O 示例 (MIPS中的轮询)

- 输入: 从键盘读入$v0

```
                lui   $t0, 0xffff # ffff0000
Waitloop:       lw    $t1, 0($t0) # control reg
                andi  $t1,$t1,0x1
                beq   $t1,$zero, Waitloop
                lw    $v0, 4($t0) # data reg
```

- 输出: 从$a0写到显示器

```
                lui   $t0, 0xffff # ffff0000
Waitloop:       lw    $t1, 8($t0) # control reg
                andi  $t1,$t1,0x1
                beq   $t1,$zero, Waitloop
                sw    $a0,12($t0) # data reg
```

- "Ready" —处理器的视角!

轮询的开销

- 处理器规格: 1 GHz 时钟频率, 完成一个轮询操作需要400个时钟周期 (轮询程序, 读写设备，返回)
- 轮询对CPU资源的占用:
  - 鼠标: 每秒30次查询，确保不会遗漏用户的动作
  - 软盘: 以2字节为一个单元，50 KB/秒的数据率传输，没有遗漏
  - 硬盘: 以16字节大小的块为单位，16 MB/秒的数据传输率，没有遗漏

轮询的处理器时间占比

- 鼠标轮询:
  - *占用时间:* 30 [轮询/秒] × 400 [时钟周期/轮询] = 12K [时钟周期/秒]
  - *时间百分比:* 1.2×10^4 [时钟周期/秒] / 10^9 [时钟周期/秒] = 0.0012%
  - 鼠标轮询对处理器影响很小
- 磁盘轮询:
  - *频率:* 16 [MB/秒] / 16 [B/轮询] = 2^{20} [轮询/秒]
  - *占用时间:* 2^{20} [轮询/秒] × 400 [时钟周期/轮询] ≈ 419M [时钟周期/秒]
  - *时间百分比:* 4.19×10^8 [时钟周期/秒] / 10^9 [时钟周期/秒] = 41.9%
  - 不可接受!
- 问题: 轮询, 读写较小的块

替代方案?

- 浪费太多处理器时间用于“自旋等待”（spin-waiting）I/O就绪
- 当I/O设备就绪时调用相关的过程
- 方案：使用*异常*机制触发I/O，然后在I/O进行数据传输的时候*中断*程序

➢61

异常和中断

- “突发的”事件需要改变控制流
  - ➢不同的指令集体系结构会使用不同的术语
- *异常*
  - ➢CPU内部产生
    (例如未定义的opcode，溢出，系统调用，TLB 缺失)
- *中断*
  - ➢来自外部I/O控制器
- 需要牺牲性能

➢62

处理异常

- MIPS中异常由系统控制协处理器（CP0）处理
- 保存出问题（或者被中断）的指令的PC内容
  - ➢MIPS： 保存在特殊的寄存器中
    Exception Program Counter (EPC)
- 保存问题的描述
  - ➢MIPS：保存在特殊的寄存器中，*Cause* 寄存器
  - ➢最简单的实现只需要1bit（0 ：未定义的opcode，1：溢出）
- 转跳到异常处理代码*(exception handler code)*，起始地址：0x80000180
- 通知操作系统
  - ➢可以“杀”程序
  - ➢对于 I/O 设备请求或系统调用，通常同时切换到另一个进程
    - 比如当发生 TLB 缺失和页失效时

➢63