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

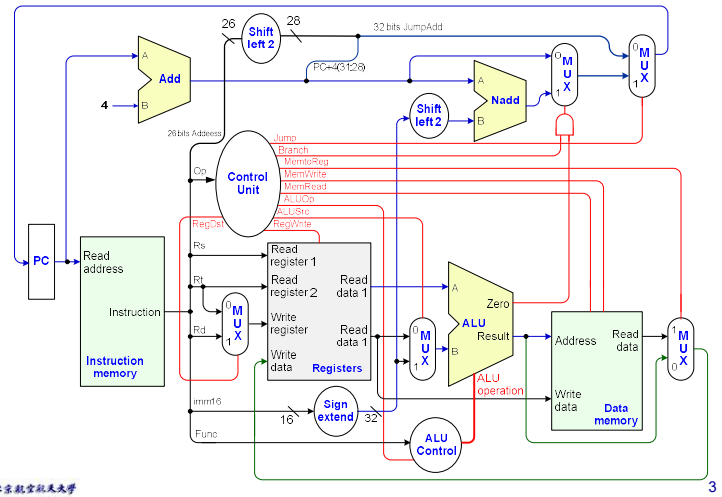
1

习题4——汇编语言

- 已发布
  - Spoc平台
- 10月28日截止
  - 23:55
- 在sopc提交
  - 电子版，可手写

2

回顾：单周期控制器设计（包含跳转指令的数据通路）



3

回顾：单周期数据通路性能分析

- 指令周期（指令执行时间）不同类型的指令具有不同的指令周期

数据通路各部分以及各类指令的执行时间

Instruction class	Instruction memory	Register read	ALU operation	Data memory	Register write	Total
R-type	200	50	100	0	50	400 ps
Load word	200	50	100	200	50	600 ps
Store word	200	50	100	200		550 ps
Branch	200	50	100	0		350 ps
Jump	200					200 ps

1. 固定时钟周期
   - 指令平均周期 = 600ps
2. 可变时钟周期
   - 假设指令在程序中出现的频率，平均指令执行时间
     600*25%+550*10%+400*45%+350*15%+200*5% = 447.5ps

—若采用可变时钟周期，时间性能比单周期更高；
—但控制比单周期要复杂、困难，得不偿失。
—改进方法：改变每种指令类型所用的时钟数，即采用多周期实现

4

回顾：最基本的指令处理引擎

- 每条指令花费一个时钟周期来执行
- 只用组合逻辑来实现指令的执行
  - *没有中间的、程序员不可见的状态更新*

A = 时钟周期开始时的体系结构状态（程序员可见）

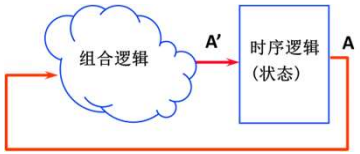
↓

在一个时钟周期内处理指令

↓

A' = 时钟周期结束时的体系结构状态(程序员可见)

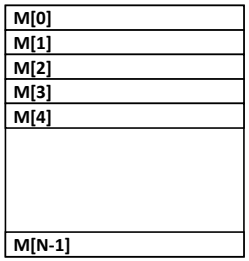
- 单周期机器



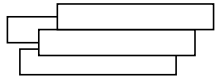
时钟周期长度
组合逻辑中的关键路径

5

回顾：程序员可见(体系结构)的状态



内存
用地址索引的存储位置的阵列



寄存器
- 在ISA中会给寄存器命名(相当于地址)
- 通用和专用寄存器

程序计数器
当前指令的内存地址

➢指令和程序指定如何转换程序员可见的状态值

6

回顾：单周期 vs. 多周期

- 单周期的机器
  - 每条指令执行需要一个时钟周期
  - 所有状态的更新在指令执行结束的时刻完成
  - 劣势：最慢的指令决定时钟周期的长度 → 时钟周期时间长
  - 指令处理周期的所有阶段都在*一个机器时钟周期*中完成
  - 数据信号操作的同时产生控制信号（在同一个时钟周期内起作用）
- 多周期的机器
  - 指令处理分到多个周期/阶段中完成
  - 指令执行过程中可以更新状态
  - 但是体系结构状态的更新只能在指令执行结束的时刻完成
  - 与单周期相比的“优势”：最慢的 “阶段” 决定时钟周期长度
  - 指令处理周期的所有阶段可以在*多个机器时钟周期*中完成
  - 实际上，每个阶段都可以在多个时钟周期中完成
  - 下一个周期需要的控制信号可以在前一个周期就产生

7

回顾：初步的性能分析

- 指令执行时间
  - {CPI} x {clock cycle time}
- 程序执行时间
  - 所有指令的[{CPI} x {clock cycle time}]之和
  - {指令数} x {平均 CPI} x {clock cycle time}
- 单周期微体系结构的性能
  - CPI = 1
  - Clock cycle time 长
- 多周期微体系结构的性能
  - CPI = 每条指令不同
    - 平均 CPI → 希望能很小
  - Clock cycle time 短

现在，我们有两个独立的自由度可以优化

8

第六讲 MIPS处理器设计

一．处理器设计概述
二．MIPS模型机
三．MIPS单周期处理器设计
1. 单周期数据通路设计
2. 单周期控制器设计
3. 单周期性能分析
四．MIPS多周期处理器设计
1. 多周期数据通路设计
2. 多周期控制器设计
3. 多周期性能分析
五．MIPS流水线处理器设计

4.1 MIPS 多周期数据通路设计

❖为什么不使用单周期实现方式

➢单周期设计中，时钟周期对所有指令等长。而时钟周期由计算机中可能的最长路径决定，一般为取数指令。但某些指令类型本来可以在更短时间内完成。

❖多周期方案

➢将指令执行分解为多个步骤，每一步骤一个时钟周期，则指令执行周期为多个时钟周期，不同指令的指令周期包含时钟周期数不一样。

➢优点：

- 提高性能：不同指令的执行占用不同的时钟周期数；
- 降低成本：一个功能单元可以在一条指令执行过程中使用多次，只要是在不同周期中（这种共享可减少所需的硬件数量）。

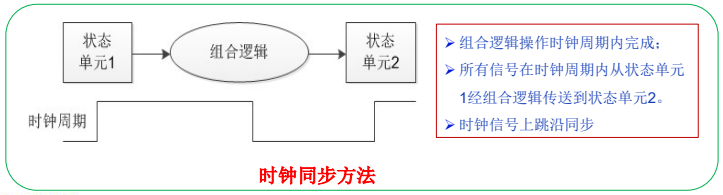
4.1 MIPS多周期数据通路设计

❖多周期数据通路设计总体考虑

➢普林斯顿结构：指令和数据使用同一个存储器；

➢一个ALU： R型指令算术逻辑运算、取指令后形成PC+4新值、及Beq指令转向地址计算（PC+4+Signext(imm16)<<2），都在ALU中完成；

➢时钟同步方法：一个时钟周期内信号总是从一个状态单元经过组合逻辑处理后传送到另一个状态单元。所以指令每一步的执行总是从前一个状态单元接收输入，经过功能单元处理，在下一个时钟周期触发沿将结果写入下一个状态单元，因此数据通路中需要增加了一个或多个寄存器以保存指令各执行步骤形成的结果（输出值），以便在指令的后续时钟周期内继续使用。



时钟同步方法

4.1 MIPS多周期数据通路设计

1. R型指令多周期分析

Op (31-26)	Rs (25-21)	Rt (20-16)	Rd (15-11)	Shamt (10-6)	Func (5-0)

◆ *R[rd] ← R[rs] op R[rt]*

步骤	RTL描述	执行部件	备注
取指令	IR ← M[PC] PC ← PC + 4	存储器 ALU	增加一个寄存器保存指令：IR
读寄存器	A ← R[IR[25:21]] B ← R[IR[20:16]]	寄存器堆	增加两个寄存器保存读取的数据R[rs]和R[rt]：A、B
计算	ALUOut ← A *op* B	ALU	增加一个寄存器保存ALU结果：ALUOut
写寄存器	R[IR[15:11]] ← ALUOut	寄存器堆	完成寄存器数据写入

4.1 MIPS多周期数据通路设计

2. Lw指令多周期分析

Op (31-26)	Rs (25-21)	Rt (20-16)	16 bit Address or Immediate (15-0)

- R[rt] ← M[*R[rs] + signext(imm16)*]

步骤	RTL描述	执行部件	备注
取指令	IR ← M[PC] PC ← PC + 4	存储器 ALU	指令寄存器IR用于保存读取的指令
读寄存器	A ← R[IR[25:21]]	寄存器堆	寄存器A保存从读取的数据R[rs]
计算地址	ALUOut ← A + *signext(IR[15:0])*	ALU	ALUOut保存计算得到的内存地址
访问存储器	DR ← M[ALUOut]	存储器	增加一个寄存器保存数据：DR
写寄存器	R[IR[20:16]] ← DR	寄存器堆	完成寄存器数据写入

4.1 MIPS多周期数据通路设计

3. Sw指令多周期分析

Op (31-26)	Rs (25-21)	Rt (20-16)	16 bit Address or Immediate (15-0)

- M[*R[rs] + signext(imm16)*] ← R[rt]

步骤	RTL描述	执行部件	备注
取指令	IR ← M[PC] PC ← PC + 4	存储器 ALU	指令寄存器IR用于保存读取的指令
读寄存器	A ← R[IR[25:21]]	寄存器堆	寄存器A保存从读取的数据R[rs]
计算地址	ALUOut ← A + *signext(IR[15:0])*	ALU	ALUOut保存计算得到的内存地址
访问存储器	M[ALUOut] ← R[IR[20:16]]	存储器	把数据R[rt]写入存储器

4.1 MIPS多周期数据通路设计

4. Beq指令多周期分析

Op (31-26)	Rs (25-21)	Rt (20-16)	16 bit Address or Immediate (15-0)

- If ((*R[rs] - R[rt])==0*) then PC ← *PC + 4 + signext(imm16)<<2*
else PC ← *PC + 4*
- 问题：ALU 要完成 *PC +4*，*PC + 4 + signext(imm16)<<2* 和 *R[rs] - R[rt]* 三次运算操作，需要在三个不同时钟周期内实现，**如何合理安排？**

步骤	RTL描述	执行部件	备注
取指令	IR ← M[PC] PC ← *PC + 4*	存储器 ALU	第一周期后 PC+4 完成，PC为新值
读寄存器 /计算转向地址	A ← R[IR[25:21]] B ← R[[IR[20:16]] ALUout ← *PC + Signext[IR[15:0]]<<2*	寄存器堆 ALU	ALU计算转向目标地址保存于ALUOut
完成转移	if (A – B==0) then PC ← ALUOut	ALU	ALU减法结果不写入ALUOut，PC修改是一条件操作

4.1 MIPS多周期数据通路设计

❖指令数据通路综合（R型指令、Lw、Sw、Beq）

步骤	R型指令	Lw指令	Sw指令	Beq指令
取指令	IR ← M[PC] PC ← PC + 4 ← ALU完成加法结果送PC			
读寄存器/译码	A ← R[IR[25:21]] B ← R[IR[20:16]] ALUOut ← *PC + Signext[IR[15:0]]<<2*			
计算	ALUOut ← A op B	ALUOut ← A + Signext(IR[15:0])		If (A-B==0) then PC ← ALUout
R型完成/访问内存	R[IR[15:11]] ← ALUOut	DR ← M[ALUOut]	M[ALUout]←B	
写寄存器		R[IR[20:16]] ←DR		

4.1 MIPS多周期数据通路设计

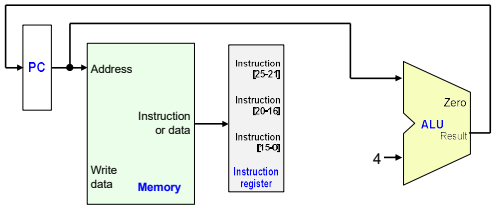
❖指令数据通路综合（R型指令、Lw、Sw、Beq）

步骤	R型指令	Lw指令	Sw指令	Beq指令
取指令	IR ← M[PC] PC ← PC + 4 P0			
读寄存器/译码	A ← R[IR[25:21]] B ← R[IR[20:16]] ALUOut ← PC + Signext[IR[15:0]]<<2 P1			
计算	P6 ALUOut ← A op B	ALUOut ← A + Signext(IR[15:0]) P2		If (A-B==0) then PC ← ALUout P8
R型完成/访问内存	P7 R[IR[15:11]] ← ALUOut	P3 DR ← M[ALUOut]	P5 M[ALUOut]← B	
写寄存器		P4 R[IR[20:16]] ←DR		

17

4.1 MIPS多周期数据通路设计

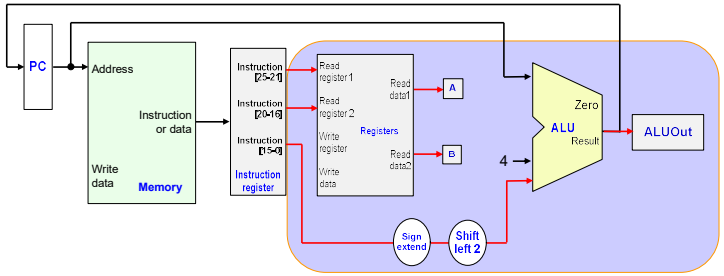
1. 多周期数据通路构建（P0）
   - IR ← M[PC]
   - PC ← PC + 4



18

4.1 MIPS多周期数据通路设计

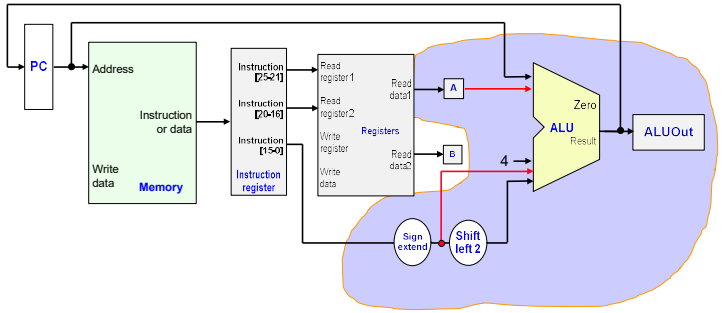
2. 多周期数据通路构建（P1）
   - A ← R[IR[25:21]]， B ← R[IR[20:16]]
   - ALUOut ← *PC + Signext[IR[15:0]]<<2*



19

4.1 MIPS多周期数据通路设计

3. 多周期数据通路构建（P2）
   - ALUOut ← *A + Signext[IR[15:0]]*

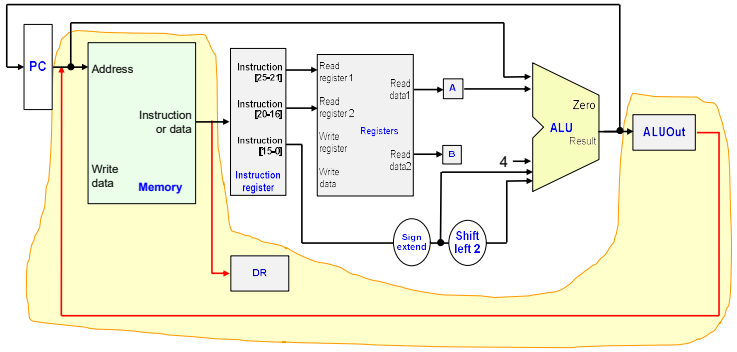


20

4.1 MIPS多周期数据通路设计

4. 多周期数据通路构建（P3）

- DR ← M[ALUOut]

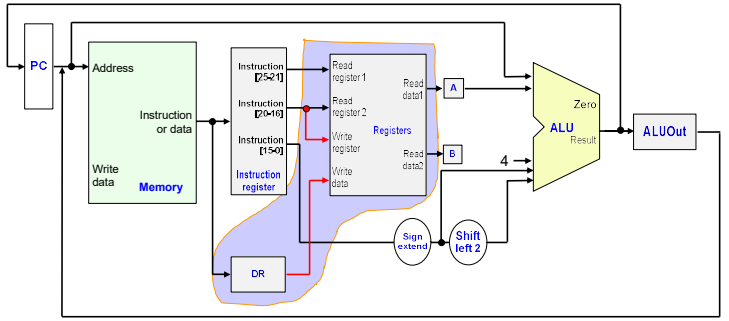


21

4.1 MIPS多周期数据通路设计

5. 多周期数据通路构建（P4）

- R[IR[20:16]] ←DR

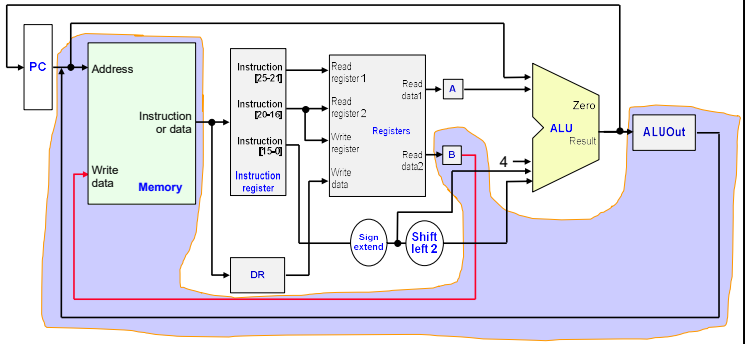


22

4.1 MIPS多周期数据通路设计

6. 多周期数据通路构建（P5）

- M[ALUOut]← B

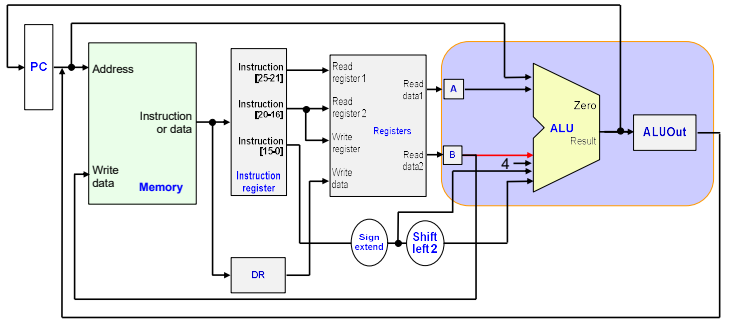


23

4.1 MIPS多周期数据通路设计

7. 多周期数据通路构建（P6）

- AUOut ← A *op* B

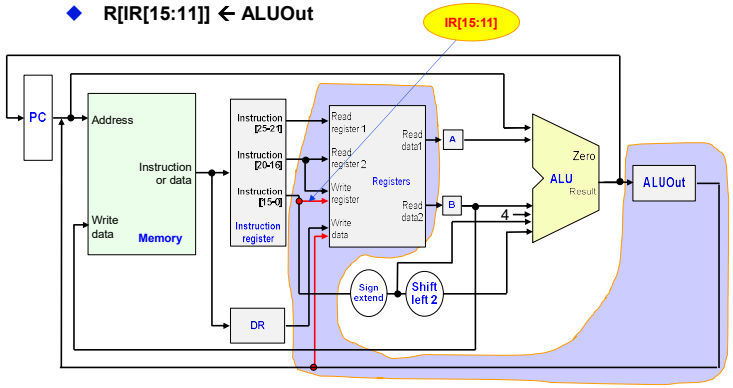


24

4.1 MIPS多周期数据通路设计

8. 多周期数据通路构建（P7）

- R[IR[15:11]] ← ALUOut

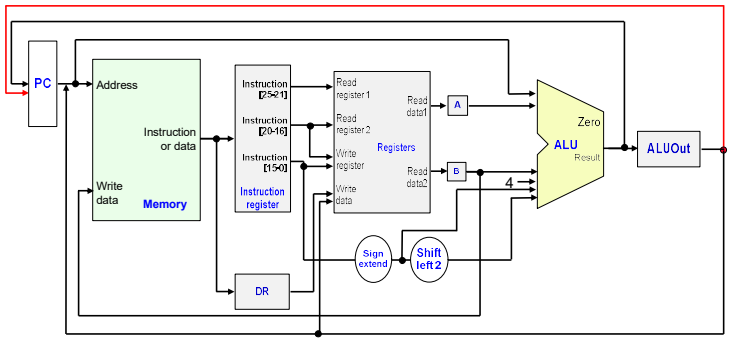


25

4.1 MIPS多周期数据通路设计

9. 多周期数据通路构建（P8）

- If (A-B==0) then PC ← ALUout

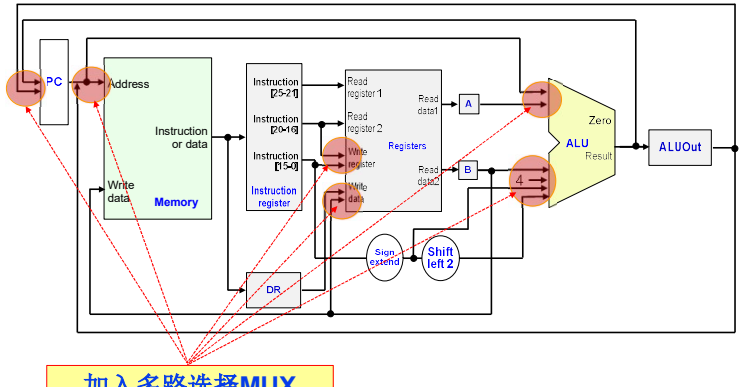


26

4.1 MIPS多周期数据通路设计

10. 多周期数据通路构建

- 存在多个输入源的，需要增加多路选择器MUX

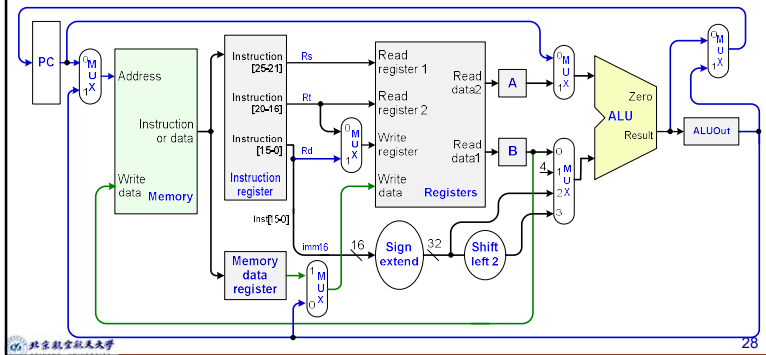


27

4.1 MIPS多周期数据通路实现

❖MIPS的多周期数据通路

- PC：输入端多路选择（2选1）
- 存储器：地址输入端多路选择（2选1）
- ALU：输入A端多路选择（2选1），输入B端多路选择（4选1）
- 寄存器堆：写地址输入端（2选1），写数据输入端（2选1）



28

4.1 MIPS多周期数据通路设计

■ 增加J指令（转移）通路

Op (31-26)	26 bit Address (for Jump Instruction) (25-0)

➢ PC← *PC+4 [31:28] || add26<<2*

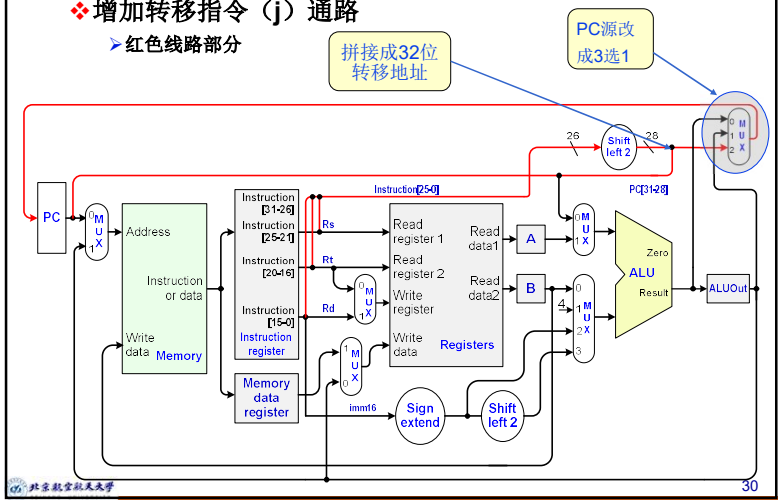
步骤	R型指令	Lw指令	Sw指令	Beq指令	J指令
取指令	IR ← M[PC] PC ← PC + 4				
读寄存器/译码	A ← R[IR[25:21]] B ← R[IR[20:16]] ALUOut ← *PC + Signext[IR[15:0]]<<2*				
计算	ALUOut ← A op B	ALUOut ← A + Signext(IR[15:0])		If (A-B==0) then PC ← ALUout	PC ← *PC[31:28] || IR[25:0]<<2*
R型完成/访问内存	R[IR[15:11]] ←ALUOut	DR ← M[ALUOut]	M[ALUOut] ← B		
写寄存器		R[IR[20:16]] ←DR			

29

4.1 MIPS多周期数据通路实现

❖增加转移指令（j）通路

➢红色线路部分



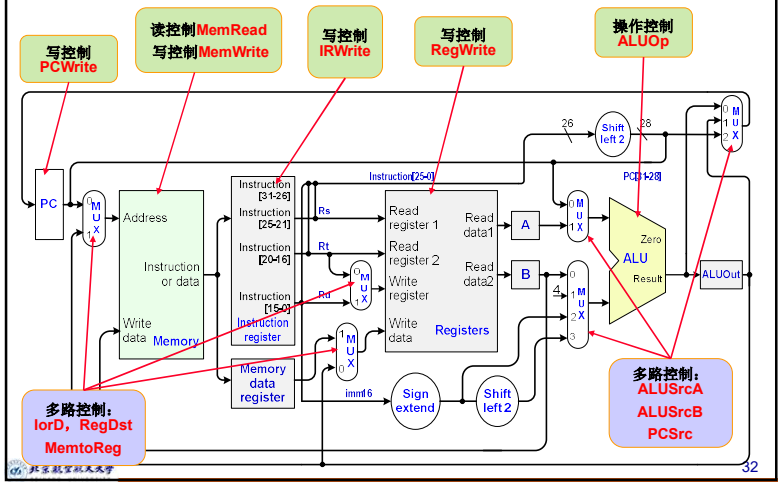
30

第六讲 MIPS处理器设计

31

4.2 多周期控制器设计

❖ 数据通路控制信号设计



32

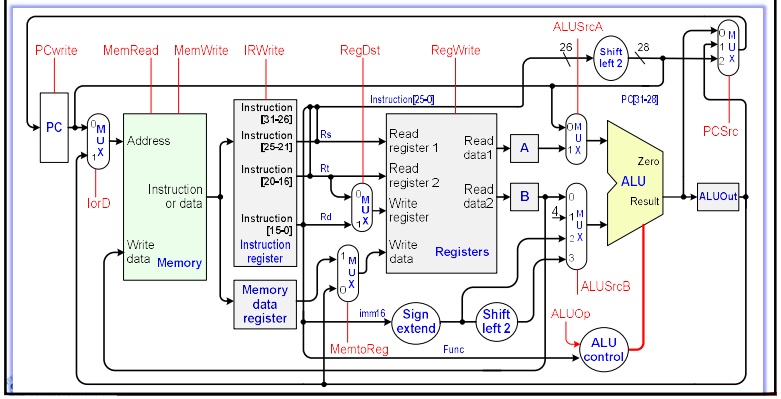
4.2 多周期控制器设计

❖数据通路控制信号

➢ALU独立控制单元ALUcontrol

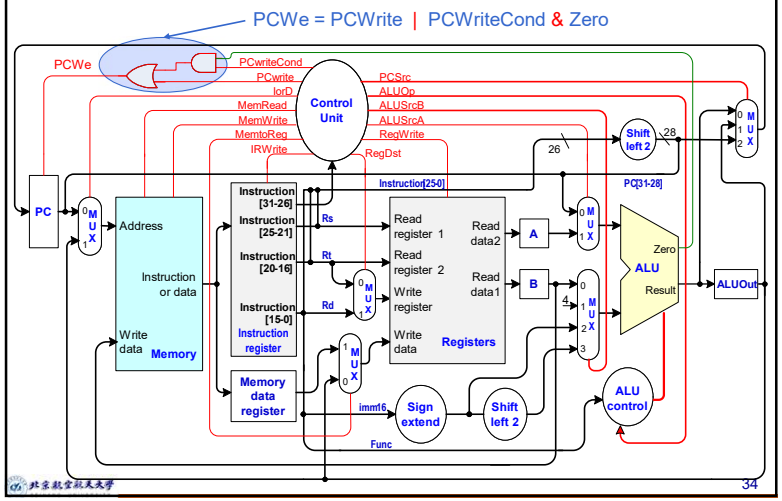
- 输入：主控生成的ALUop，字段Func（指令5:0位）
- 输出：ALU运算控制 ALU operation（4位）

ALUOp指明ALU的运算类型
- 00：访存指令所需加法
- 01：beq指令所需减法
- 10：R型指令功能码决定



33

4.2 多周期控制器设计——完整数据通路与控制信号



34

4.2 多周期控制器设计

多周期通路控制信号

控制信号	失效时作用	有效时作用
RegDst	寄存器堆写入端地址来选择Rt字段	寄存器堆写入端地址选择Rd字段
RegWrite	无	把数据写入寄存器堆中对应寄存器
ALUSrcA	ALU输入A端选择PC	ALU输入A端选择寄存器A
MemRead	无	存储器读数据（输出）
MemWrite	无	存储器写数据（输入）
MemtoReg	寄存器堆写入端数据选择ALUOut	寄存器堆写入端数据选择DR
IorD	存储器地址输入选择PC	存储器地址输入选择ALUOut
IRWrite	无	存储器输出（指令）写入IR
PCWrite	无	PC写入，PC输入源由PCSrc选择
PCWriteCond	无	如ALU的Zero端输出有效，则PC写入，输入源由PCSrc选择（Beq指令）

35

4.2 多周期控制器设计

多周期通路控制信号（续）

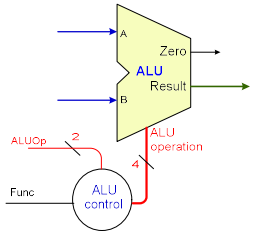
控制信号	取值（二进制）	作用
ALUOp	00	ALU执行加法
	01	ALU执行加法
	10	ALU操作由Func字段（IR[5:0]）决定
ALUSrcB	00	ALU输入端B数据源选择寄存器B
	01	ALU输入端B数据源选择常数4
	10	ALU输入端B选择符号扩展输出（Signext(imm16)）
	11	ALU输入端B选择移位器输出（Signext(imm16)<<2）
PCSrc	00	PC输入源选择ALU输出（取指阶段，PC+4）
	01	PC输入源选择寄存器 ALUOut（Beq指令）
	10	PC输入源选择转移地址 *PC[31:28]* || *IR[25:0]*<<2（j指令）

36

4.2 多周期控制器设计

❖ALU控制单元

- ➢输入：
  - 指令的Func字段（指令5:0位）
  - 由主控单元生成的 ALUOp
- ➢ALUOp指明ALU的运算类型
  - 00：访存指令所需的加法
  - 01：beq指令所需的减法
  - 10：R型指令功能码字段决定



指令	Func字段	ALUop	ALU运算类型	ALU operation
Lw	XXXXXX	00	加	0010
Sw	XXXXXX	00	加	0010
Beq	XXXXXX	01	减	0110
Add	100 000	10	加	0010
Sub	100 010	10	减	0110
And	100 100	10	与	0000
Or	100 101	10	或	0001

ALU控制单元真值表

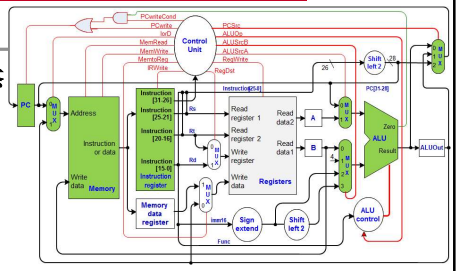
4.2 多周期控制器设计

多周期数据通路汇总

时钟周期	R型指令	Lw指令	Sw指令	Beq指令	J指令
TC1	P0 IR ← M[PC] PC ← PC + 4				
TC2	P1 A ← R[IR[25:21]]， B ← R[IR[20:16]] ALUOut ← PC + Signext[IR[15:0]]<<2				
TC3	P6 ALUOut ← A op B	P2 ALUOut ← A + Signext(IR[15:0])		P8 If (A-B==0) then PC ← ALUout	P9 PC ← PC[31:28] || IR[25:0]<<2
TC4	P7 R[IR[15:11]] ←ALUOut	P3 DR ← M[ALUOut]	P5 M[ALUOut] ← B		
TC5		P4 R[IR[20:16]] ← DR			

4.2 多周期控制器设计

1. 数据通路控制信号状态（P0，所有指令TC1）
   - IR ← M[PC]
   - PC ← PC + 4

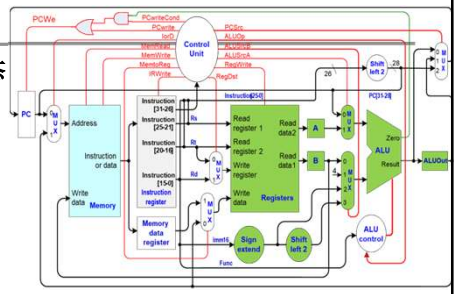


通路部件	存储器地址MUX	存储器	IR	ALU-A端MUX	ALU-B端MUX	ALU	PC输入端MUX	PC
操作	选择PC	读出	写入	选择PC	常数4	加	选择ALU	写入
控制信号状态	IorD=0	MemRead	IRWrite	ALUSrcA = 0	ALUSrcB = 01	ALUOp = 00	PCSrc = 00	PCWrite

状态ST0：
- MemRead有效
- IRWrite有效
- PCWrite有效
- IorD = 0
- ALUSrcA = 0
- ALUSrcB = 01
- ALUOp = 00
- PCSrc = 00

4.2 多周期控制器设计

2. 数据通路控制信号状态（P1，所有指令TC2）
   - A ← R[IR[25:21]]，B ← R[IR[20:16]]
   - ALUOut ← PC + *Signext[IR[15:0]]<<2*



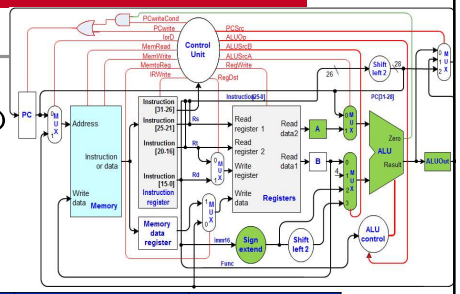
通路部件	寄存器堆	ALU-A端MUX	ALU-B端MUX	ALU
操作	无	选择PC	选择移位器	加
控制信号状态	无	ALUSrcA = 0	ALUSrcB = 11	ALUOp = 00

状态ST1：
- ALUSrcA = 0
- ALUSrcB = 11
- ALUOp = 00

4.2 多周期控制器设计

3. 数据通路控制信号状态（P2，Lw/Sw指令TC3）

- ALUOut ← A + *Signext(IR[15:0])*



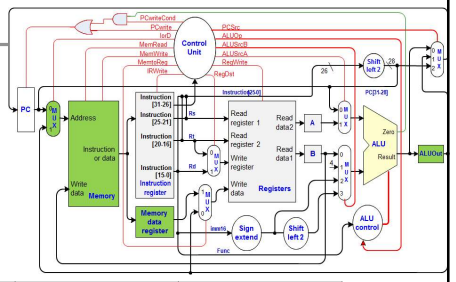
通路部件	ALU-A端MUX	ALU-B端MUX	ALU
操作	选择寄存器A	选择符号扩展	加
控制信号状态	ALUSrcA = 1	ALUSrcB = 10	ALUOp = 00

状态ST2：
- ALUScrA = 1
- ALUSrcB = 10
- ALUOp = 00

4.2 多周期控制器设计

4. 数据通路控制信号状态（P3，Lw指令TC4）

- DR ← M[ALUOut]



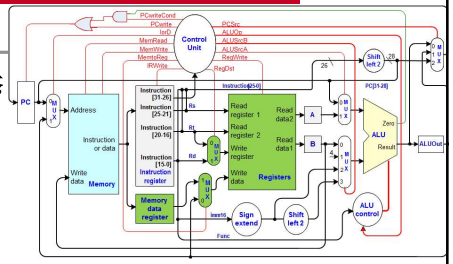
通路部件	存储器地址端MUX	存储器	数据寄存器DR
操作	选择ALUOut	读出	写入
控制信号状态	IorD = 1	MemRead	无

状态ST3：
- IorD = 1
- MemRead有效

4.2 多周期控制器设计

5. 数据通路控制信号状态（P4，Lw指令TC5）

- R[IR[20:16]] ← DR



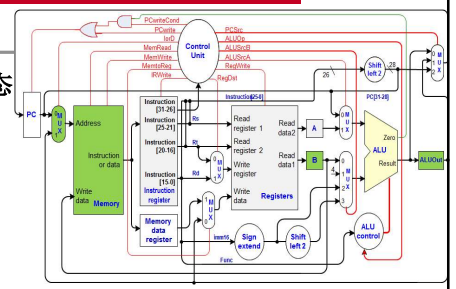
通路部件	寄存器堆写地址MUX	寄存器堆写数据MUX	寄存器堆
操作	选择Rt	选择DR	写入
控制信号状态	RegDst = 0	MemtoReg = 1	RegWrite

状态ST4：
- RegDst = 0
- MemtoReg = 1
- RegWrite有效

4.2 多周期控制器设计

6. 数据通路控制信号状态（P5，Sw指令TC4）

- M[ALUOut] ← B



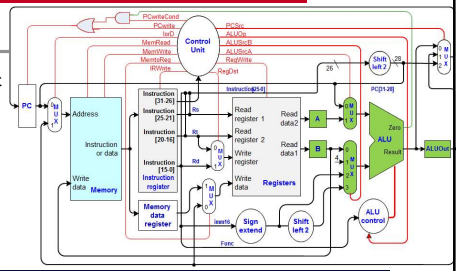
通路部件	存储器地址MUX	存储器	
操作	选择ALUOut	写入	
控制信号状态	IorD = 1	MemWrite	

状态ST5：
- IorD = 1
- MemWrite有效

4.2 多周期控制器设计

7. 数据通路控制信号状态（P6，R型指令TC3）

- ALUOut ← A *op* B



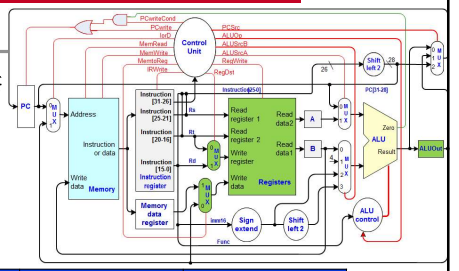
通路部件	ALU-A端MUX	ALU-B端MUX	ALU
操作	选择寄存器A	选择寄存器B	根据Func字段运算
控制信号状态	ALUSrcA = 1	ALUSrcB = 00	ALUOp = 10

状态ST6：
- ALUScrA = 1
- ALUSrcB = 00
- ALUOp = 10

4.2 多周期控制器设计

8. 数据通路控制信号状态（P7，R型指令TC4）

- R[IR[15:11]] ← ALUOut



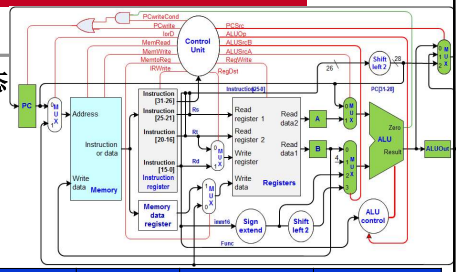
通路部件	寄存器堆写地址MUX	寄存器堆写数据MUX	寄存器堆
操作	选择Rd	选择ALUOut	写入
控制信号状态	RegDst = 1	MemtoReg = 0	RegWrite

状态ST7：
- RegDst = 1
- MemtoReg = 0
- RegWrite有效

4.2 多周期控制器设计

9. 数据通路控制信号状态（P8，Beq指令TC3）

- If (*A- B==0*) then
  PC ← ALUout



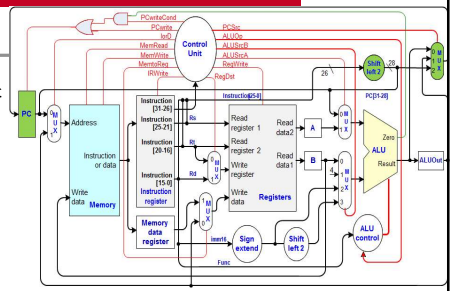
通路部件	ALU-A端MUX	ALU-B端MUX	ALU	PC输入端MUX	PC
操作	选择寄存器A	选择寄存器B	减	选择ALUOut	有条件写入
控制信号状态	ALUSrCA = 1	ALUSrcB = 00	ALUOp = 01	PCSrc = 01	PCWriteCond

状态ST8：
- ALUSrcA = 1
- ALUSrcB = 00
- ALUOp = 01
- PCSrc = 01
- PCWriteCond有效

4.2 多周期控制器设计

10. 数据通路控制信号状态（P9，J指令TC3）

- PC ← *PC[31:28]* || *IR[25:0]<<2*

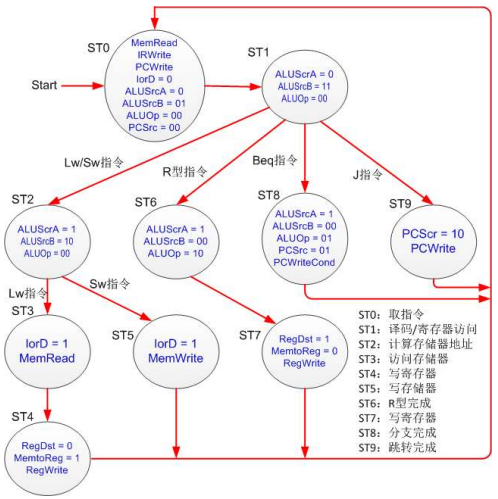


通路部件	PC输入端MUX	PC	
操作	选择转移地址	写入	
控制信号状态	PCSrc = 10	PCWrite	

状态ST9：
- PCScr = 10
- PCWrite有效

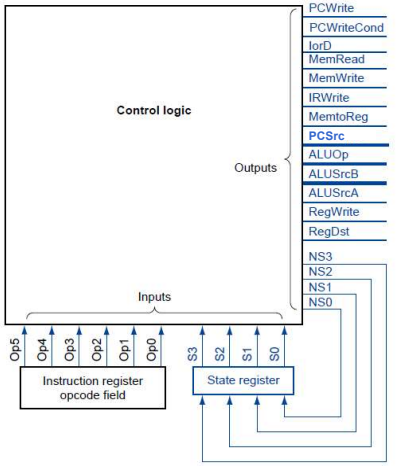
4.2 多周期控制器设计——主控单元

多周期
主控单元
有限状态图



4.2 多周期控制器设计——主控制单元

- 有限状态机FSM
  - ST0、ST1等10个状态
  - 状态编码用4位二进制表示，如$S_3S_2S_1S_0$=0110表示ST6
  - 次态用$NS_3NS_2NS_1NS_0$表示
- 摩尔型FSM
  - 控制信号输出仅取决于当前状态
  - 次态由现态和当前输入（IR寄存器中6位操作码）决定



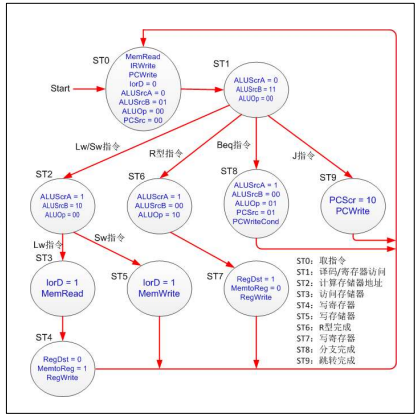
4.2 多周期控制器设计——主控制单元

- 控制信号逻辑示例
  控制信号仅和当前状态有关

$$PCWrite = ST0 + ST9$$
$$= \overline{S_3}\overline{S_2}\overline{S_1}\overline{S_0} + S_3\overline{S_2}\overline{S_1}S_0$$

$$ALUSrcA = ST2 + ST6 + ST8$$
$$= \overline{S_3}S_2\overline{S_1}\overline{S_0} + S_3\overline{S_2}S_1\overline{S_0} + S_3\overline{S_2}\overline{S_1}\overline{S_0}$$

$$RegWrite = ST4 + ST7$$
$$= \overline{S_3}S_2\overline{S_1}\overline{S_0} + S_3S_2S_1S_0$$

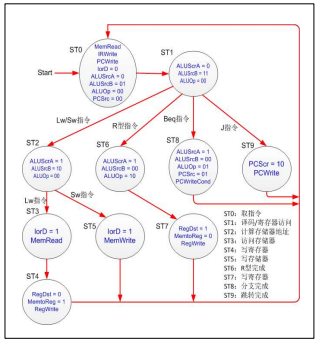


4.2 多周期控制器设计——主控制单元

- 次态信号逻辑，次态输出与当前状态和当前输入（Opcode）相关

次态信号真值表

当前状态输入 $S_3S_2S_1S_0$	操作码输入 ($Op_5Op_4Op_3Op_2Op_1Op_0$)	次态输出 $NS_3NS_2NS_1NS_0$
0000 (ST0)		0001 (ST1)
0001 (ST1)	100011 (lw指令)	0010 (ST2)
0001 (ST1)	101011 (sw指令)	0010 (ST2)
0001 (ST1)	000000 (R型指令)	0110 (ST6)
0001 (ST1)	000100 (Beq指令)	1000 (ST8)
0001 (ST1)	000010 (J指令)	1001 (ST9)
0010 (ST2)	100011 (lw指令)	0011 (ST3)
0010 (ST2)	101011 (sw指令)	0101 (ST5)
0011 (ST3)		0100 (ST4)
0110 (ST6)		0111 (ST7)
ST4, ST5, ST7, ST8, ST9		0000 (ST0)



$$NS_3 = ST1(Beq + J)$$
$$= \overline{S_3}S_2\overline{S_1}S_0(\overline{Op_5}Op_4\overline{Op_3}Op_2\overline{Op_1}\overline{Op_0} + \overline{Op_5}Op_4\overline{Op_3}Op_2\overline{Op_1}\overline{Op_0})$$

第六讲 MIPS处理器设计

4.3 多周期性能分析

- 假设主要功能单元的操作时间
  - 存储器 ：200ps
  - ALU ：100ps
  - 寄存器堆：50ps
  - 多路复用器、控制单元、PC、符号扩展单元、线路没有延迟

各类指令执行时间

步骤	R型指令	Lw指令	Sw指令	Beq指令	J指令	执行时间
取指令	IR ← M[PC]，PC ← PC + 4					200ps
读寄存器/译码	A ← R[IR[25:21]]， B ← R[IR[20:16]] ALUOut ← PC + Signext[IR[15:0]]<<2					100ps
计算	ALUOut ← A op B	ALUOut ← A + Signext(IR[15:0])		If (A-B==0) then PC ← ALUout	PC ← PC[31:28] || IR[25:0]<<2	100ps
R型完成/访问内存	R[IR[15:11]] ← ALUOut	DR ← M[ALUOut]	M[ALUOut] ← B			200ps
写寄存器		R[IR[20:16]] ← DR				50ps

4.2 多周期控制器分析

- 时钟周期
  - 时钟周期取各步骤中最长的时间，200ps

各类指令执行时间

时钟周期	R型指令	Lw指令	Sw指令	Beq指令	J指令	周期时间
TC1	IR ← M[PC]，PC ← PC + 4					200ps
TC2	A ← R[IR[25:21]]， B ← R[IR[20:16]] ALUOut ← PC + Signext[IR[15:0]]<<2					200ps
TC3	ALUOut ← A op B	ALUOut ← A + Signext(IR[15:0])		If (A-B==0) then PC ← ALUout	PC ← PC[31:28] || IR[25:0]<<2	200ps
TC4	R[IR[15:11]] ← ALUOut	DR ← M[ALUOut]	M[ALUOut] ← B			200ps
TC5		R[IR[20:16]] ← DR				200ps

4.3 多周期性能分析

- 各型指令所需的时钟周期数和时间
  - R型指令：800ps
  - lw指令 ：1000ps
  - sw指令 ：800ps
  - beq指令 ：600ps
  - j指令 ：600ps
- 假设指令在程序中出现的频率
  - lw指令 ：25%
  - sw指令 ：10%
  - R型指令：45%
  - beq指令：15%
  - j指令 ：5%
- 则一条指令的平均CPI
  - 5*25%+4*10%+4*45%+3*15%+3*5% = 4.05
- 一条指令的平均执行时间：
  - 1000*25%+800*10%+800*45%+600*15%+600*5% = 810ps

第六讲 MIPS处理器设计

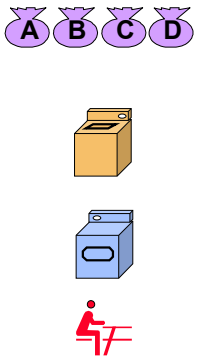
一．处理器设计概述
二．MIPS模型机
三．MIPS单周期处理器设计
四．MIPS多周期处理器设计
五．MIPS流水线处理器设计
1. 流水线及其冒险
2. 流水线设计的工程化方法

计算机性能评价

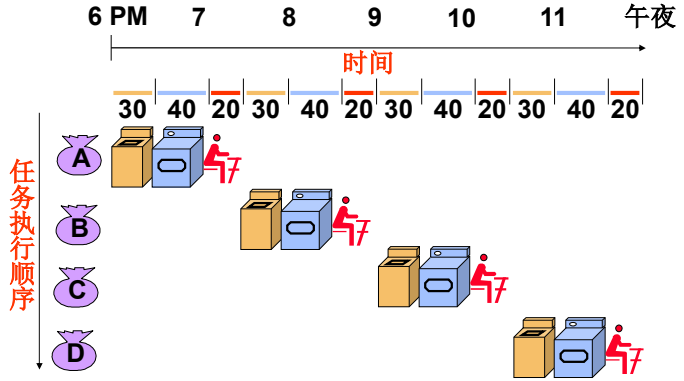
- 响应时间与吞吐量
  - 响应时间：从提交作业到完成作业所花费的时间
    - 响应时间是完成一个任务所花的时间总和，包括内存访问时间、执行IO操作的时间、以及运行必要的操作系统代码所需的时间。
  - 吞吐量：一定时间间隔内完成的作业数
    - 多任务操作系统更侧重于优化系统的整体吞吐量，而不会特别最小化某个特定程序的响应时间
  - 个人用户更关心响应时间，企业级计算机的管理人员更关心吞吐量
  - 对于企业级计算机以外的应用，响应时间是评价计算机性能的主要依据

洗衣房：生活中的流水线

- 处理器：洗衣房
- 4条指令：张三、李四、王五、周大麻子
  - 指令：洗衣服
  - 指令过程：洗涤→烘干→熨整
- 指令各阶段延迟
  - 洗衣：30分钟
  - 烘干：40分钟
  - 熨整：20分钟

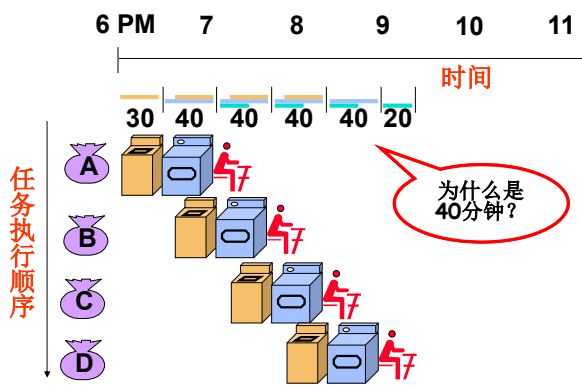


串行洗衣房



- 串行洗衣房：6个小时完成4个任务
- 如果采用流水线技术，那么。。。

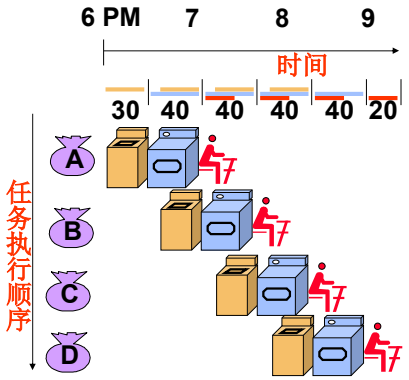
流水化洗衣房：尽可能早的开始工作



- 流水化洗衣房：3.5小时完成4个任务

61

流水线性质



- 流水线不改善单个任务处理延迟(latency)，但改善了整体工作负载的吞吐率
- 流水线速率受限于最慢的流水段
- 多个任务同时工作，但占用不同的资源
- 潜在加速比 = 流水线级数
- 流水段执行时间不平衡，则加速比下降
- 填充流水线和排放流水线，加速比下降

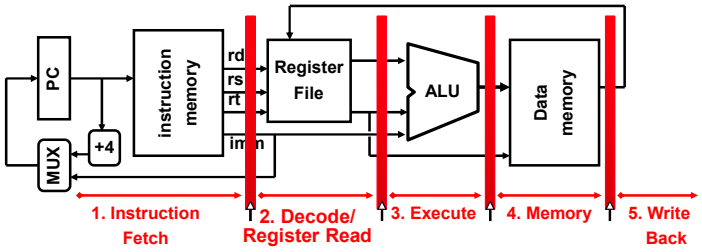
62

MIPS数据通路的5个阶段

1) IF: 取指令（Instruction Fetch）, PC值变化
2) ID: 指令译码（Instruction Decode）, 读寄存器
3) EX: 执行运算（Execution）， ALU操作
   Load/Store: 计算地址
   其他指令: 执行运算操作
4) MEM: 访存
   Load: 从Memory中读取数字
   Store: 将数据写入Memory
5) WB: 数据写回寄存器（Write Data Back to Register）

63

流水线数据通路



- 在不同阶段之间增加寄存器
  - 保存前一个周期产生的信息
- 5 阶段流水线
  - 意味着时钟频率实际上可以看作变为原来的5倍

64