中国科学技术大学
University of Science and Technology of China

计算机组成原理复习

王 超

中国科学技术大学计算机学院
嵌入式系统实验室

2019年春

内容提要

- Ch1-计算机系统概论
- Ch2.1-指令系统
- Ch2.2-程序的编译过程-参考,不要求
- Ch3-MIPS处理器设计-单周期-多周期
- Ch4.1-MIPS处理器设计-流水线
- Ch4.2-常见CPU流水线设计实例-参考，不要求
- Ch5-中断与异常
- Ch6-存储系统
- Ch7-总线系统
- Ch8-IO系统
- Ch9-指令调度与并行化-参考，不要求

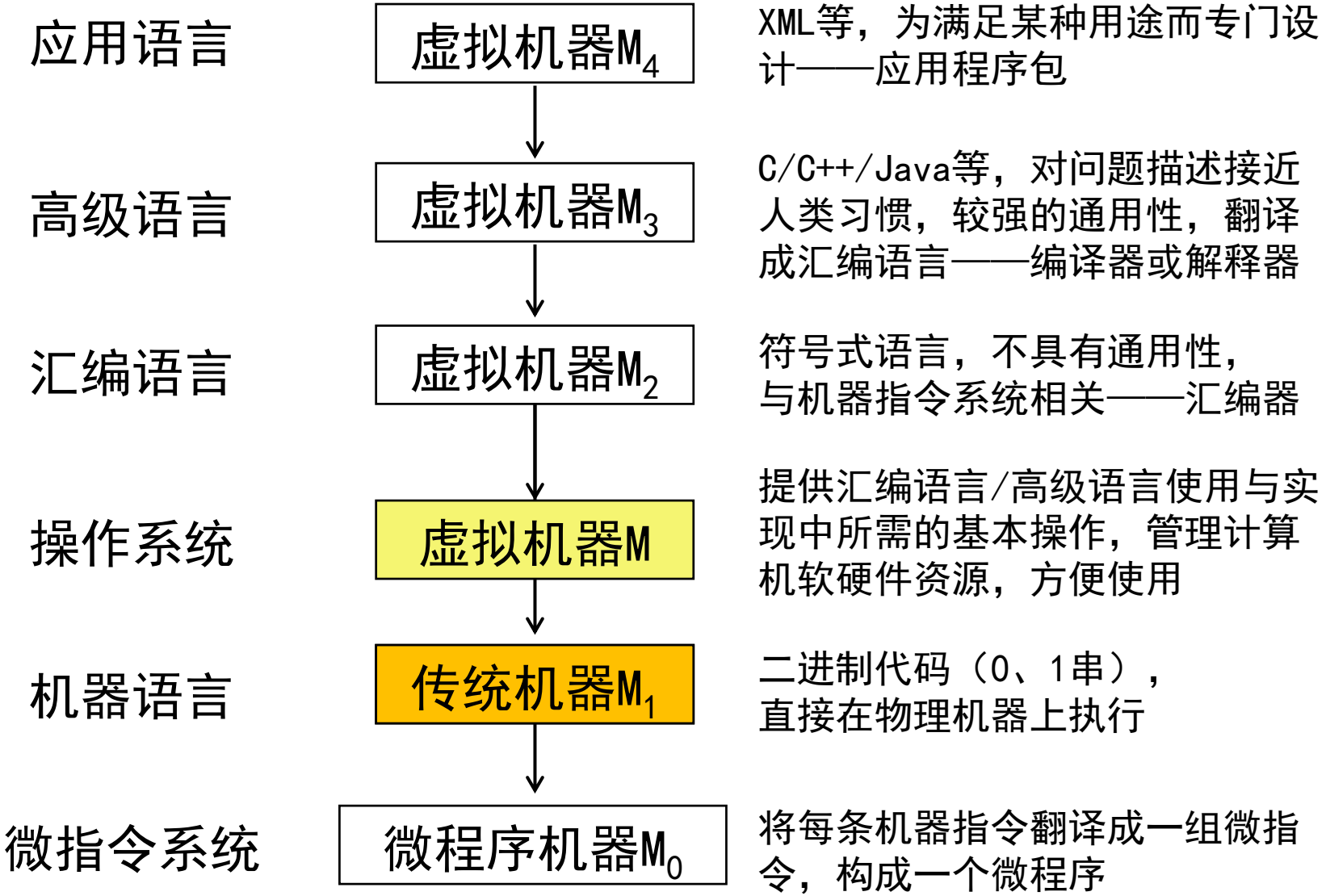
Ch1-计算机系统概论

中国科学技术大学
University of Science and Technology of China

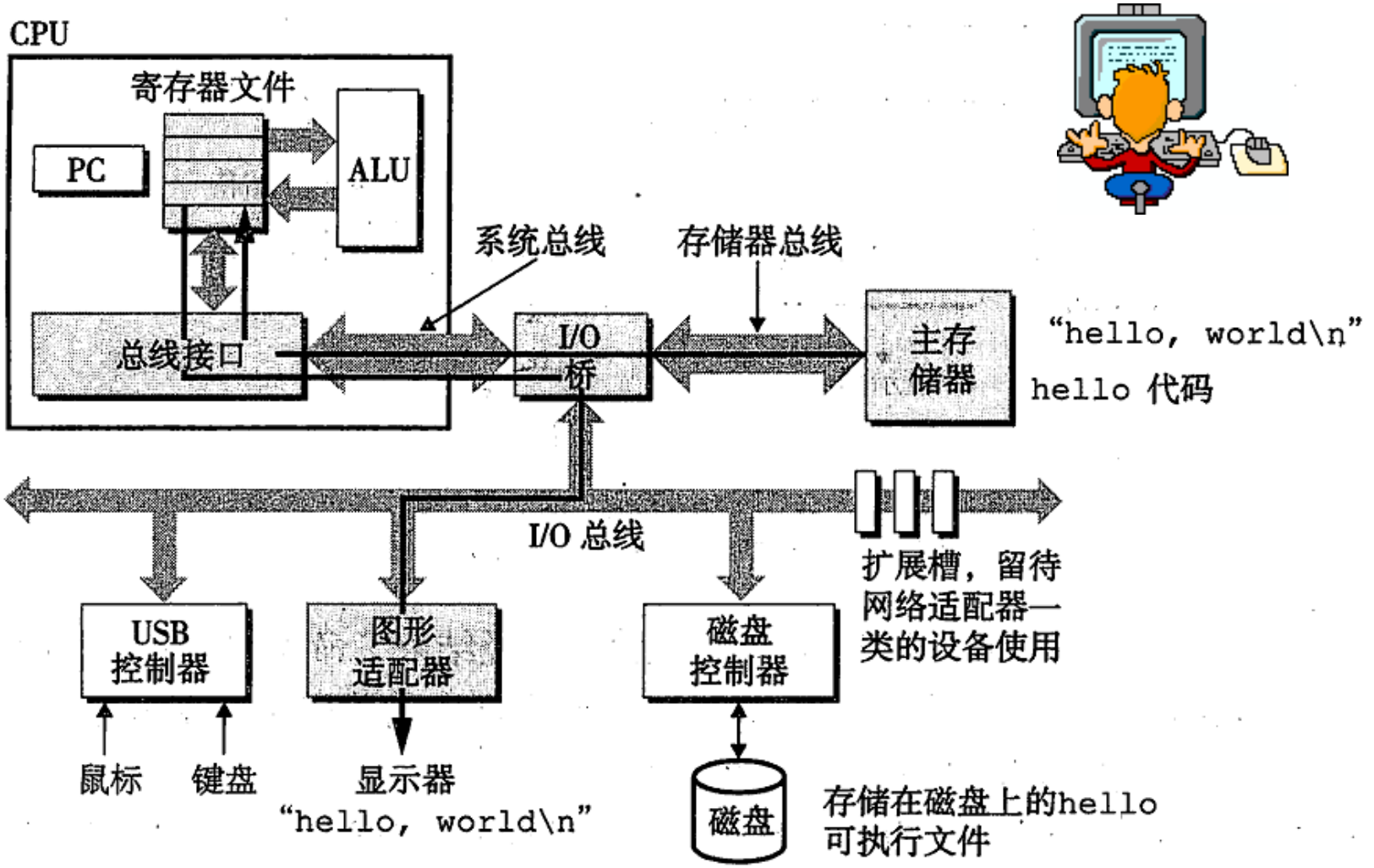
- 计算机系统概述
  - ✓ 计算机系统的层次结构
- 计算机的基本组成
  - ✓ 计算机硬件组成
  - ✓ 计算机软件组成
- 计算机硬件性能指标
  - ✓ 机器字长、存储容量、运行速度
- 计算机/CPU发展历史
  - ✓ 计算机的分代
  - ✓ 计算机设计面临的挑战
  - ✓ 芯片的制作过程
- 性能评估方法
  - ✓ 芯片成本计算
  - ✓ CPU时间
  - ✓ Amdahl定律

要求：
了解计算机系统的构成
熟悉计算机系统性能评估方法

计算机系统的层次结构


应用语言
虚拟机器M_4
XML等，为满足某种用途而专门设计——应用程序包
高级语言
虚拟机器M_3
C/C++/Java等，对问题描述接近人类习惯，较强的通用性，翻译成汇编语言——编译器或解释器
汇编语言
虚拟机器M_2
符号式语言，不具有通用性，与机器指令系统相关——汇编器
操作系统
虚拟机器M
提供汇编语言/高级语言使用与实现中所需的基本操作，管理计算机软硬件资源，方便使用
机器语言
传统机器M_1
二进制代码（0、1串），直接在物理机器上执行
微指令系统
微程序机器M_0
将每条机器指令翻译成一组微指令，构成一个微程序

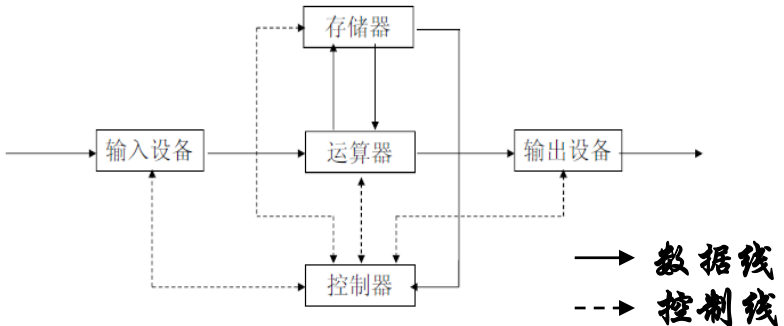
计算机处理“hello, world“的过程



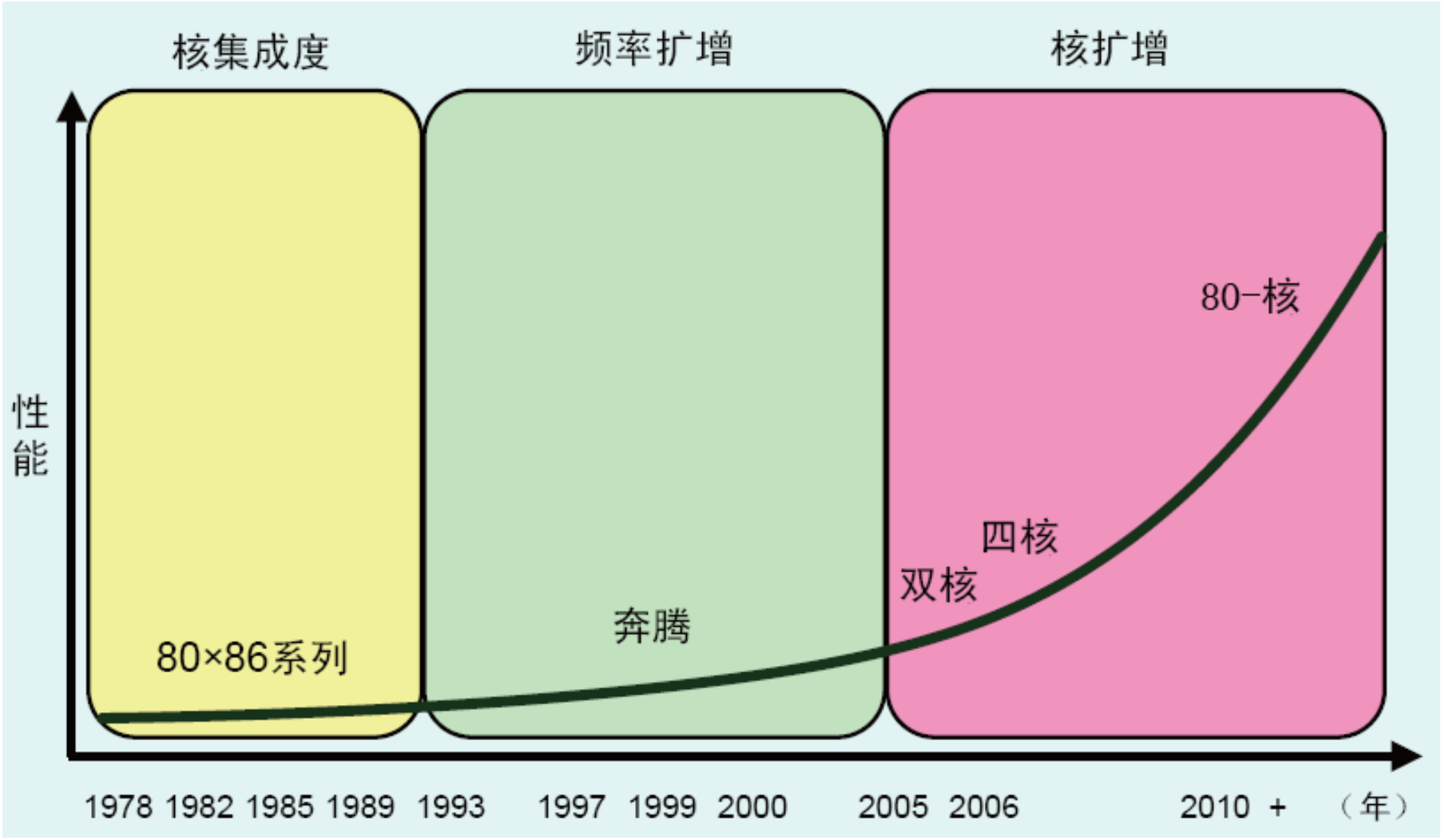
冯·诺依曼计算机组成结构

冯·诺依曼机

- ✓1945年，冯·诺依曼在研究EDVAC时提出“存储程序”概念，以此概念为基础研制的计算机统称为冯·诺依曼机
- ✓特点：
  - 五大组成部件，以运算器为中心
  - 数据和指令用二进制数表示，以同等地位存放于存储器中，按地址访问
  - 指令由操作码和地址码组成，在存储器中按顺序存放



CPU发展趋势图



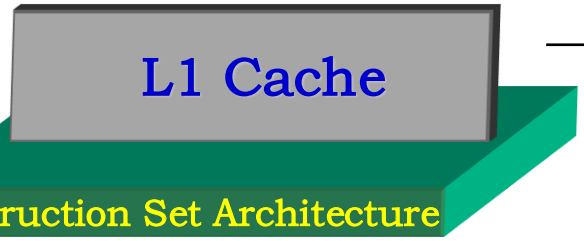
Higher Performance: Supercomputing in the Connected Era. Michael Dell2008

计算机发展过程中的问题

RAID

Burst技术

带宽、延迟



寻址

流水线和指令级并行

存储器
层次结构

存储墙/IO墙

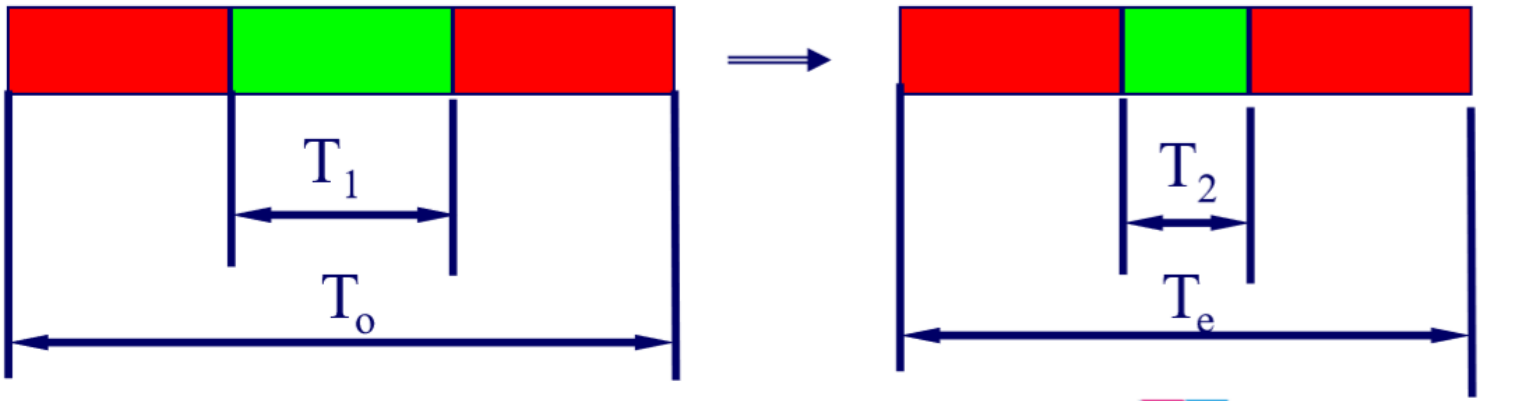
流水线、相关消除、超标量、
重排序、分支预测、向量、DSP

线延迟墙和频率墙

提升性能：Amdahl定律（计算）

$$f_e = T_1 / T_o \qquad S_e = T_1 / T_2$$

$$T_e = T_o\left[(1-f_e)+\frac{f_e}{S_e}\right] \qquad S = \frac{1}{(1-f_e)+\frac{f_e}{S_e}}$$



CPU性能公式

$$CPI = CLK / IC$$

$$IC \times CPI = CLK$$

$$T_{CPU} = CLK / f$$

时钟周期时间

$$T_{CPU} = CPI \times IC / f$$

$$T_{CLK} = 1 / f$$

$$T_{CPU} = CPI \times IC \times T_{CLK}$$

CPU性能计算例题

- 某台计算机只有Load/Store 指令能对存储器进行读/写操作，其它指令只对寄存器进行操作。根据对某程序跟踪实验结果，已知每种指令所占的比例及CPI数如下：

指令类型	指令所占比例	CPI
算逻指令	43%	1
Load指令	21%	2
Store指令	12%	2
转移指令	24%	2

CPU性能计算例题

- 求上述情况下的平均CPI。
- 假设该程序由M条指令组成。算逻运算中25%的指令两个操作数中的一个已在寄存器中，另一个必须在算逻指令执行前用Load指令从存储器取到寄存器。因此有人建议增加另一种算逻指令，其特点是一个操作数取自寄存器，另一个操作数取自存储器，即寄存器—存储器类型，假设这种指令的CPI等于2。同时，转移指令的CPI变为3。求新指令系统的平均CPI。

CPU性能计算例题

- CPI= (43×1+21×2+12×2+24×2)%=1.57
- 43%×25%=11%

指令类型	原比例	原CPI	新比例	CPI
新算逻指令	0%	0	11%	2
原算逻指令	43%	1	32%	1
Load指令	21%	2	10%	2
Store指令	12%	2	12%	2
转移指令	24%	2	24%	3

- CPI_{new} =(11×2+32×1+10×2+12×2+24×3)%
  =1.70

CPInew =(11×2+32×1+10×2+12×2+24×3) /89%
=1.91

Ch2-指令系统主要内容

1. 指令系统概述
   - 1.1 指令系统的发展
   - 1.2 指令系统的性能要求
   - 1.3 编程语言与硬件的关系

2. 指令格式
   - 2.1 指令的一般格式
   - 2.2 指令字长
   - 2.3 指令助记符

3. 操作数与操作类型
   - 3.1 操作数类型
   - 3.2 数据在存储器中的存储方式
   - 3.3 操作类型

4. 寻址方式
   - 4.1 指令寻址
   - 4.2 数据寻址

5. CISC与RISC
   - 6.1 CISC技术
   - 6.2 RISC技术
   - 6.3 CISC与RISC的比较

6. 指令系统设计与举例
   - 5.1 典型指令
   - 5.2 典型指令系统

要求：了解指令系统的原理、能够设计功能完备、性能优异的指令集

数据在存储器中的存放方式

◘字节顺序的问题

✓多字节数据的存储，涉及至

大尾端与小尾端 0X0123

字地址	31 24	23 16	15 8	7 0
0	0X0	0X1	0X2	0X3
4				

字地址

0

4

小尾端——低位在低地址

如下数据00000000 00000001 00000010 00000011

小尾端： 00000000 00000001 00000010 000000

addr+3 addr+2 addr+1 addr+

//低位在低地址

大尾端： 00000011 00000010 00000001 0000000

addr+3 addr+2 addr+1 addr+0

//高位在低地址

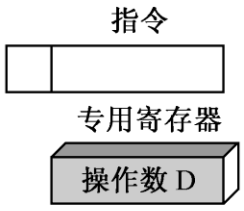
```
#include <stdio.h>
int main()
{
    int tt = 1;
    char *c = (char*)(&tt);
    if(*c == 1)
    {
        printf(      endian\n");
    }
    else
    {
        printf(      endian\n");
    }
    return 0;
}
```

数据寻址方式

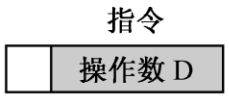
□基本的数据寻址方式分析

方式	算法	主要优点	主要缺点
隐含寻址	操作数在专用寄存器	无存储器访问	数据范围有限
立即寻址	操作数=A	无存储器访问	操作数幅值有限
直接寻址	EA=A	简单	地址范围有限
间接寻址	EA=(A)	大的地址范围	多重存储器访问
寄存器寻址	EA=R	无存储器访问	地址范围有限
寄存器间接寻址	EA=(R)	大的地址范围	额外存储器访问
偏移寻址	EA=A+(R)	灵活	复杂
段寻址（偏移）	EA=A+(R)	灵活	复杂
堆栈寻址	EA=栈顶	无存储器访问	应用有限

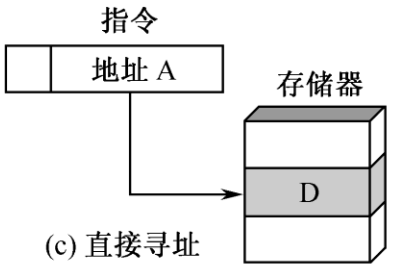
数据寻址方式示意图



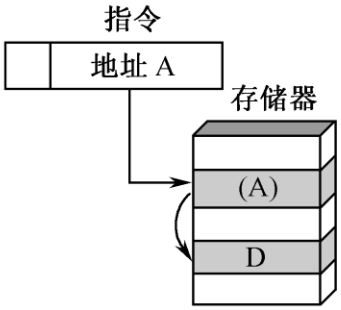
(a) 隐含寻址



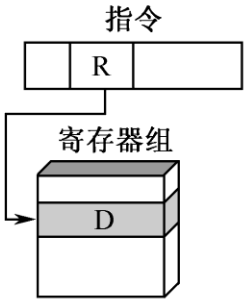
(b) 立即寻址



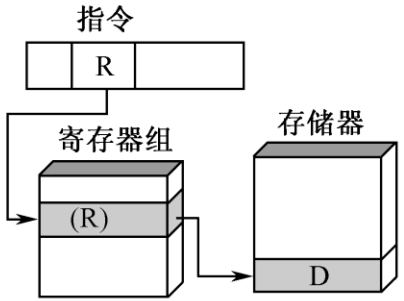
(c) 直接寻址



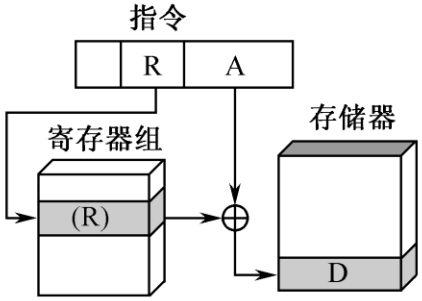
(d) 间接寻址



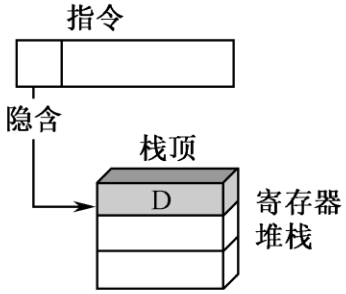
(e) 寄存器寻址



(f) 寄存器间接寻址



(g) 偏移寻址



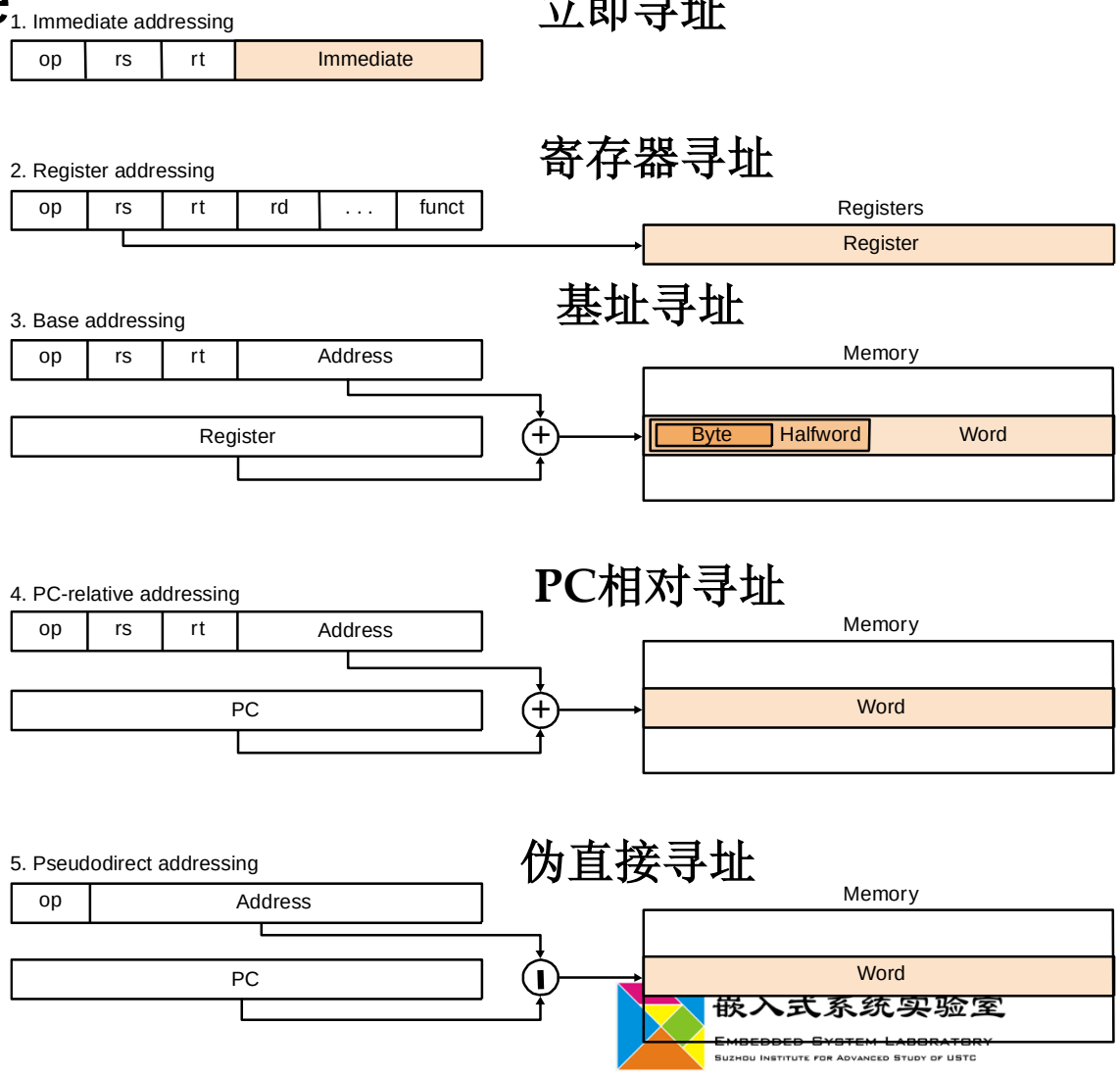
(h) 堆栈寻址

复杂指令集->简单指令集

	CISC	RISC
指令数目	一般多于200	一般小于100
指令格式	较多，一般多于4种	较少，一般小于4种
寻址方式	一般多于4种	一般小于4种
指令字长	不固定	等长
可访存指令	不限制	只有load/store
各种指令使用频率	相差很大	基本相当
各种指令执行时间	相差很大	多数为一个周期
程序代码长度	较短	较长
优化编译实现	很难	较容易
程序控制	微程序	组合逻辑

实例：MIPS寻址模式

- 寄存器寻址：R-type
- 基址寻址：I-type
- 立即寻址： I-type
- 相对寻址：J-type
- 伪直接寻址：J-type
  - ✓26位形式地址左移2位，与PC的高4位拼接



Ch3-MIPS处理器设计

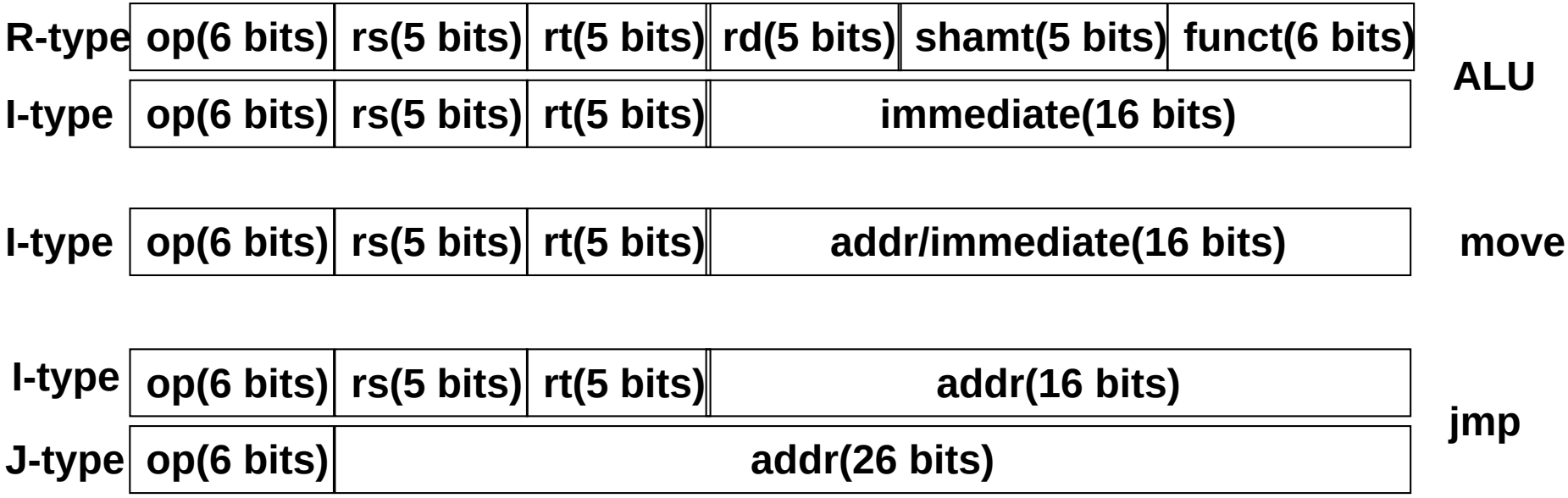
- MIPS指令集结构
- MIPS处理器结构
  - ✓单周期CPU设计
    - 功能部件
    - 数据通路
  - ✓多周期MIPS处理器设计
    - 数据通路
    - 控制信号和状态机

要求：

能够设计并实现单周期、多周期的处理器（数据通路->控制信号->状态机），并结合前边知识进行性能评估。

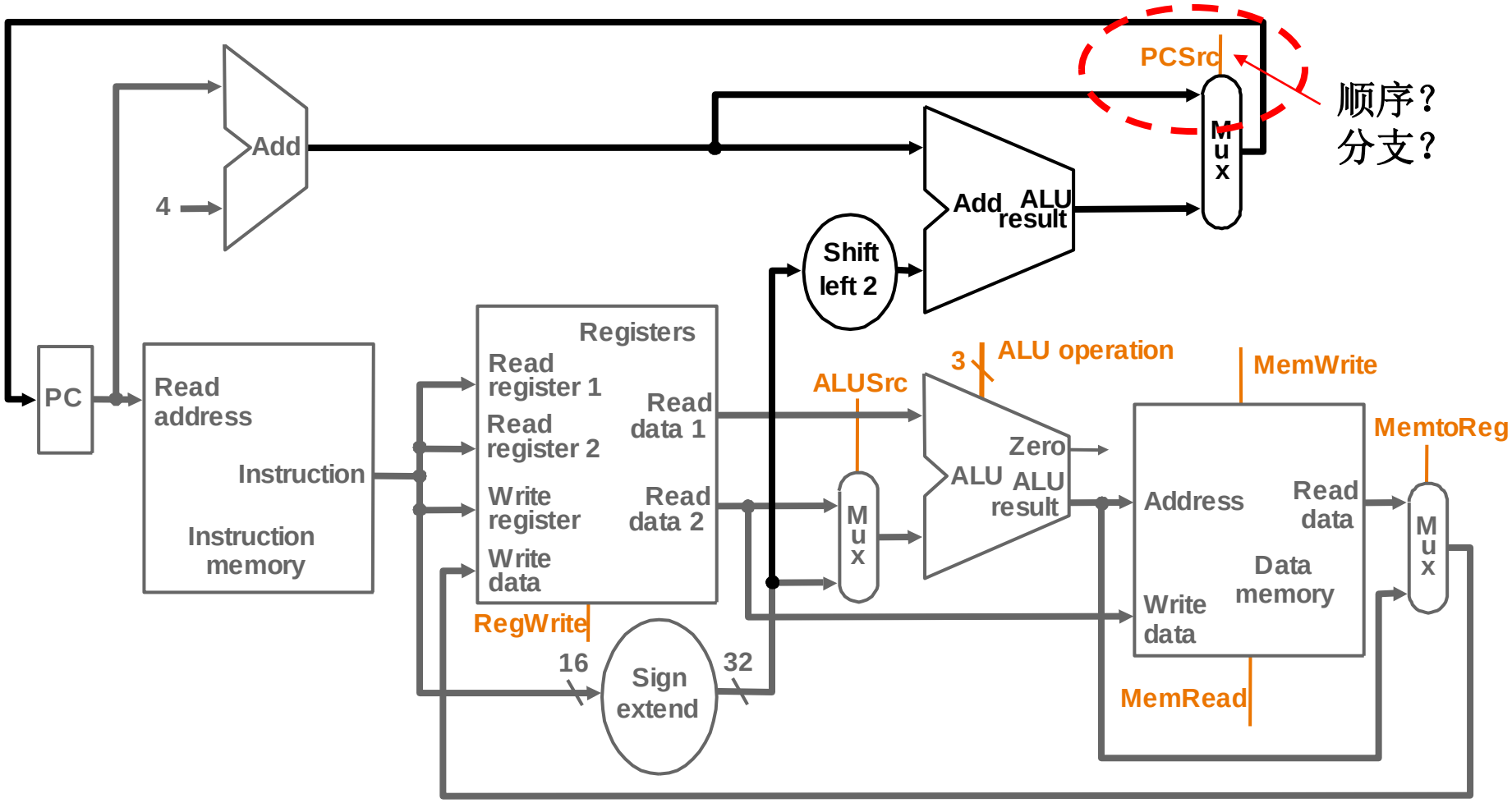
MIPS指令字格式

- 100余条指令（P&H 33条），32个通用寄存器
- 指令格式：定长32位
  - ✓R-type：arithmetic instruction
  - ✓I-type：data transfer, arithmetic instruction（如addi）
  - ✓J-type：branch instruction(conditional & unconditional)



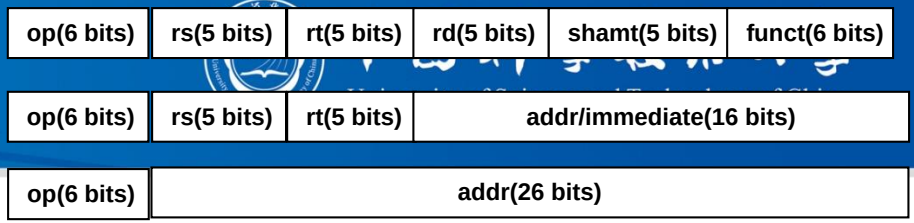
新版BEQ指令 J类->I类

单周期：R-/I-/J-type操作数据通路总图



控制信号如何处理？

单周期：ALU控制信号

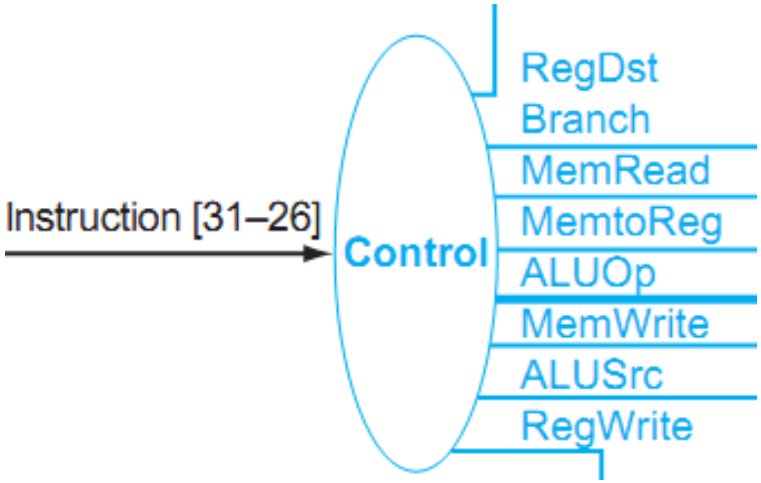


Instruction opcode	ALUop	Instruction operation	Funct field	desired ALU action	ALU ctrl input
LW	00	Load word	xxxxxx	add	0010
SW	00	store word	xxxxxx	add	0010
beq	01	Branch eq	xxxxxx	subtract	0110
R-type	10	Add	100000	Add	0010
R-type	10	Substract	100010	Substract	0110
R-type	10	And	100100	And	0000
R-type	10	Or	100101	Or	0001
R-type	10	**Set on less than**	101010	**Set on less than**	0111

2位ALUop和6位func组合，产生ALU_ctrl_input（即4位ALU operation）

单周期控制器：控制信号生成

- op域（6位）译码产生的控制信号：8个
  - ✓ RegDst：选择rt或rd作为写操作的目的寄存器
    - R-type指令（1）与 load指令二选一（0）
  - ✓ RegWrite：寄存器写操作控制
  - ✓ ALUSrc：ALU的第二个操作数来源
    - R-type指令（0）与 I-type指令（1）（含branch指令）二选一
  - ✓ ALUOp：R-type指令（2位）
  - ✓ MemRead：存储器读控制，load指令
  - ✓ MemWrite：存储器写控制，store指令
  - ✓ MemtoReg：目的寄存器数据来源
    - R-type指令与load指令二选一
  - ✓ Branch：是否分支指令（产生PCSrc）



I-type	op(6 bits)	rs(5 bits)	rt(5 bits)	addr (16 bits)		
R-type	op(6 bits)	rs(5 bits)	rt(5 bits)	rd(5 bits)	shamt(5 bits)	funct(6 bits)

- PCSrc：nPC来源控制，顺序与分支二选一
  - ✓ “是否beq指令（Branch）”&“ALU的Zero状态有效”

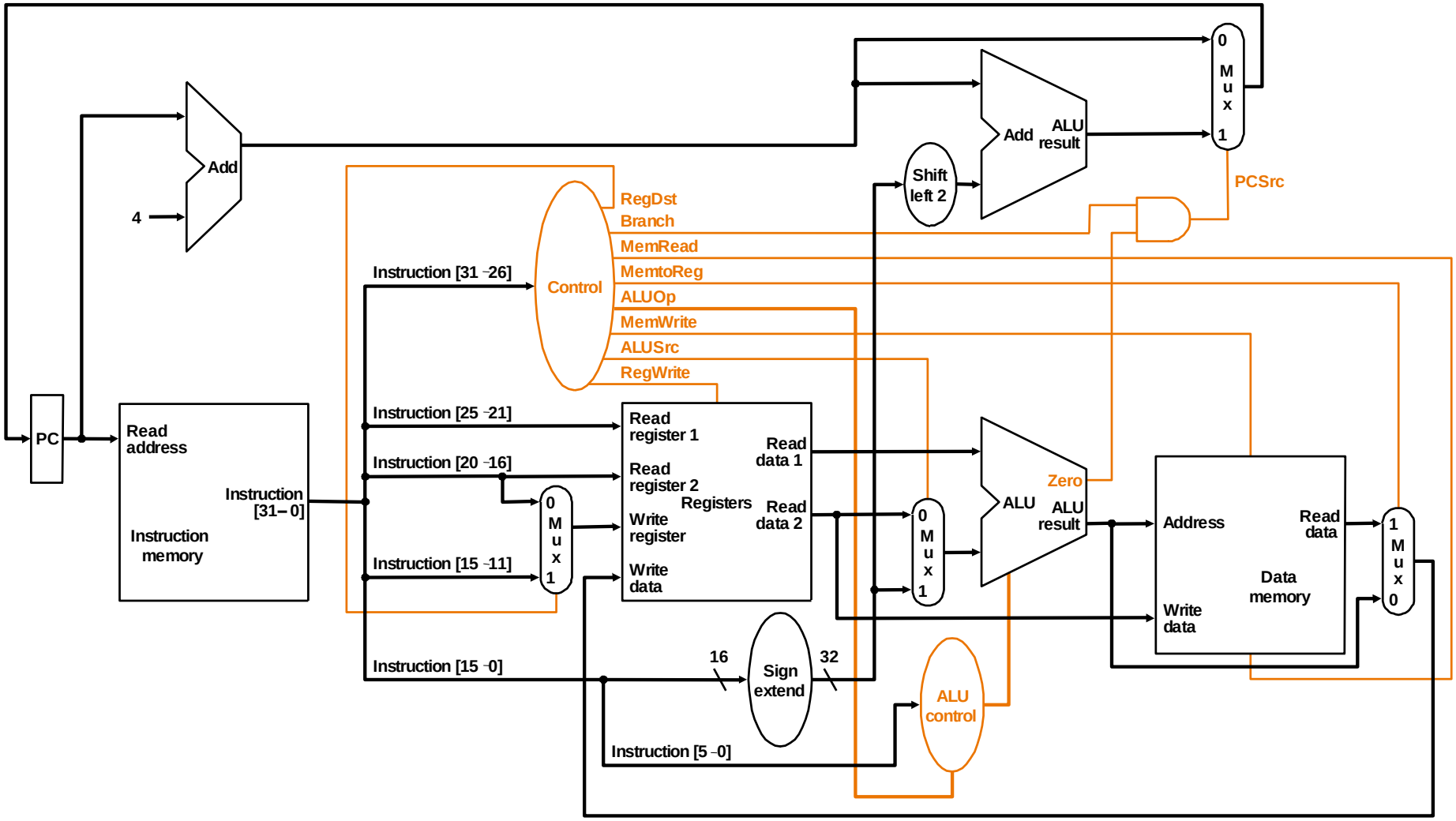
控制信号例子

6位OP译码产生8位控制信号

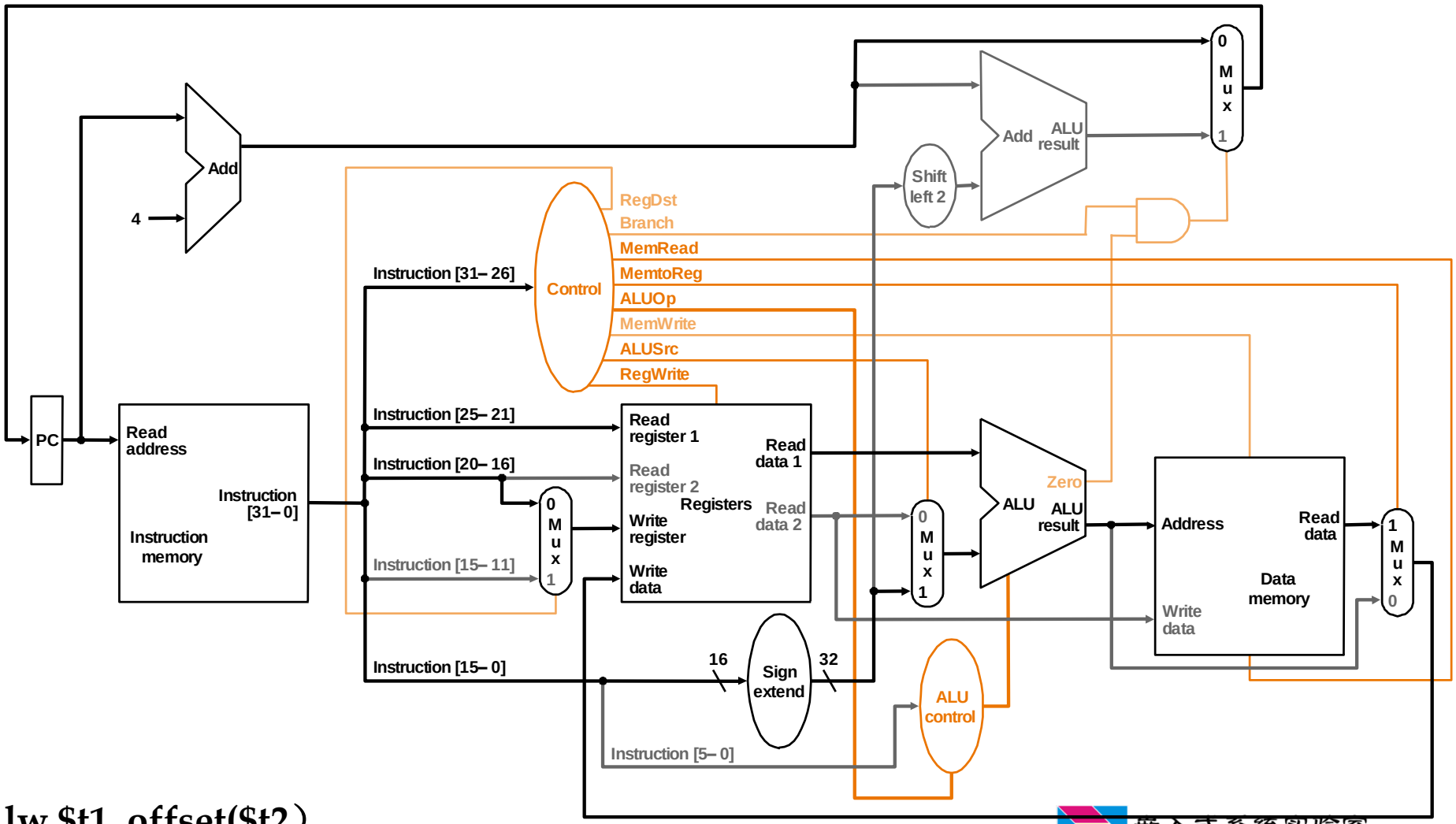
1. ALUOP 2位 + 6位Func
   ➔4位ALU 控制信号
2. 目的寄存器编号RS/RT
3. 目的寄存器来源
4. 目的寄存器写
5. ALU二操作数来源
6. 存储器读
7. 存储器写
8. 是否为Branch

	Signal name	R-type	lw （31）	sw （35）	beq
inputs	op5（26）	0	1	1	0
	op4	0	0	0	0
	op3	0	0	1	0
	op2	0	0	0	1
	op1	0	1	1	0
	op0（31）	0	1	1	0
outputs	RegDst				
	ALUSrc				
	MemtoReg				
	RegWrite				
	MemRead				
	MemWrite				
	Branch				
	ALUop1				
	ALUop0				

单周期：主控制部件


Add
4
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [31–26]
Control
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
0 M u x 1
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
Shift left 2
Add
ALU result
0 M u x 1
PCSrc
0 M u x 1
ALU
Zero
ALU result
ALU control
Address
Read data
Data memory
Write data
1 M u x 0

单周期例子：lw指令的执行路径

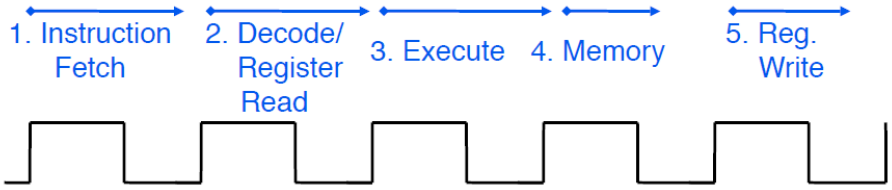


lw $t1, offset($t2）

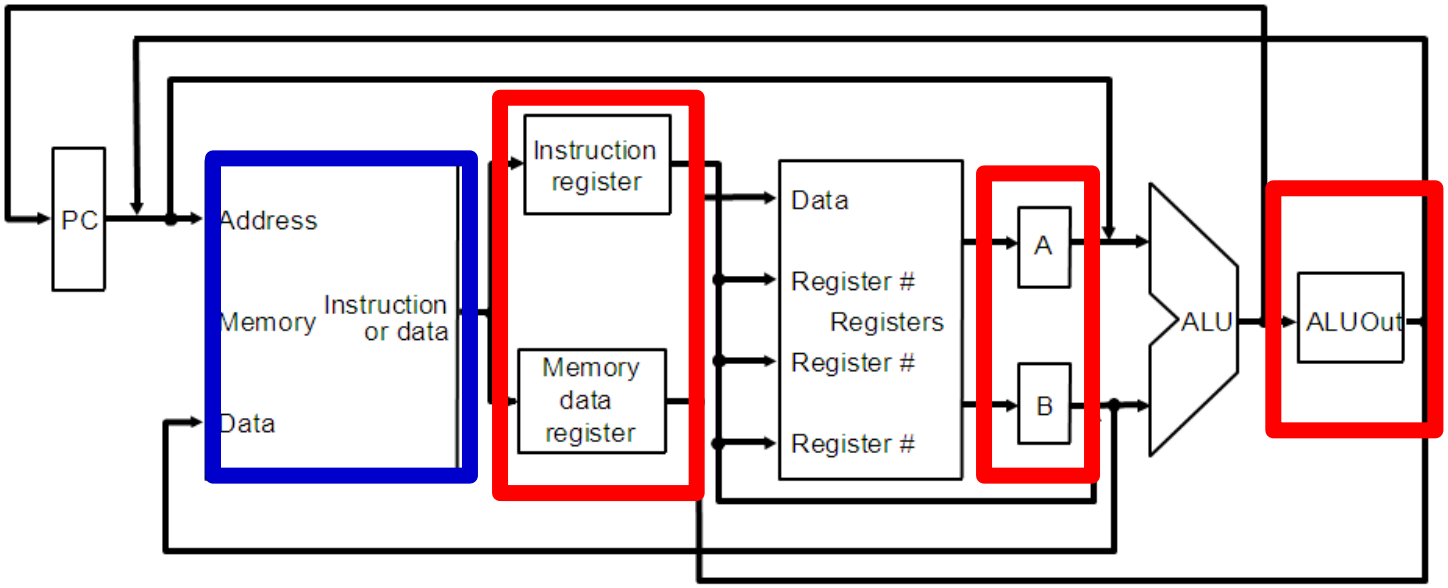
注意图中数据通路和控制信号（深色）

多周期-指令执行的阶段划分

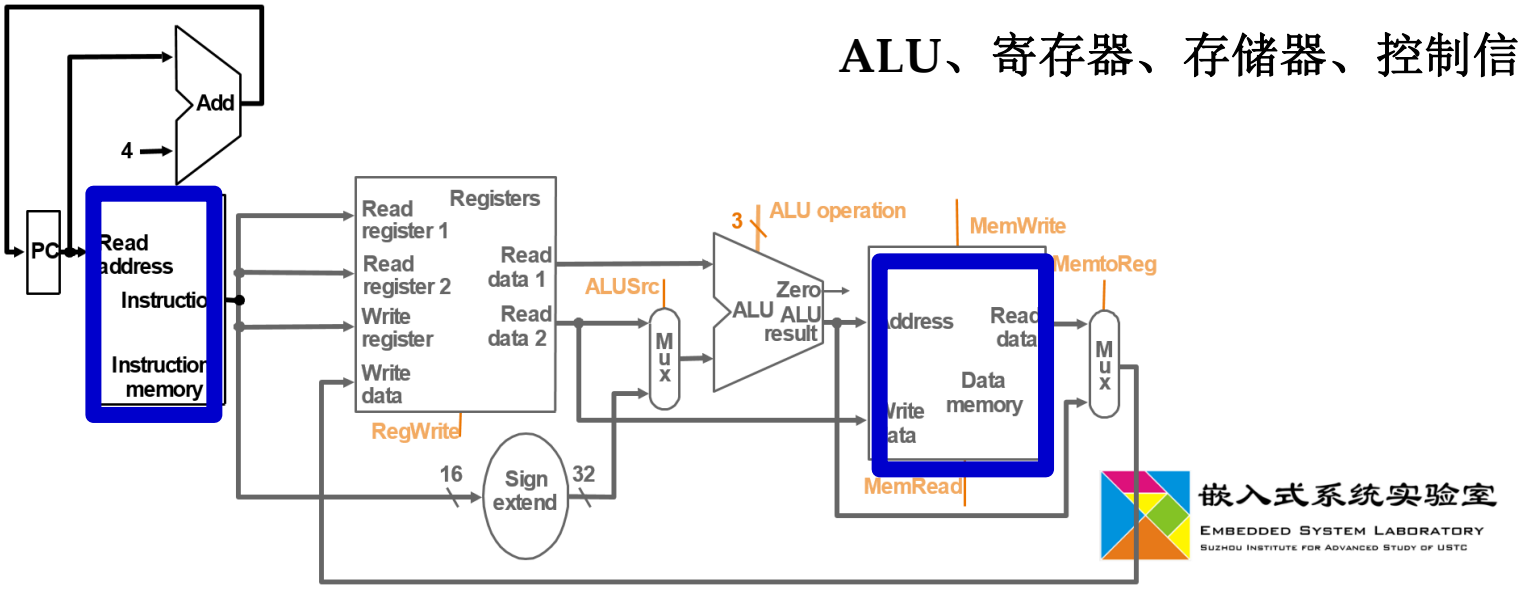
- 单周期问题->多周期？
- 共5个阶段
  - ✓取指
  - ✓译码阶段、计算beq目标地址
  - ✓执行：R-type指令执行、访存地址计算，分支完成阶段
  - ✓访存：lw读，store和R-type指令完成阶段
  - ✓写回：lw完成阶段
- 注意
  - ✓定长机器周期：机器周期=时钟周期
  - ✓不定长指令周期：分别为3、4、5个机器周期
- 控制器根据机器周期标识发出控制信号



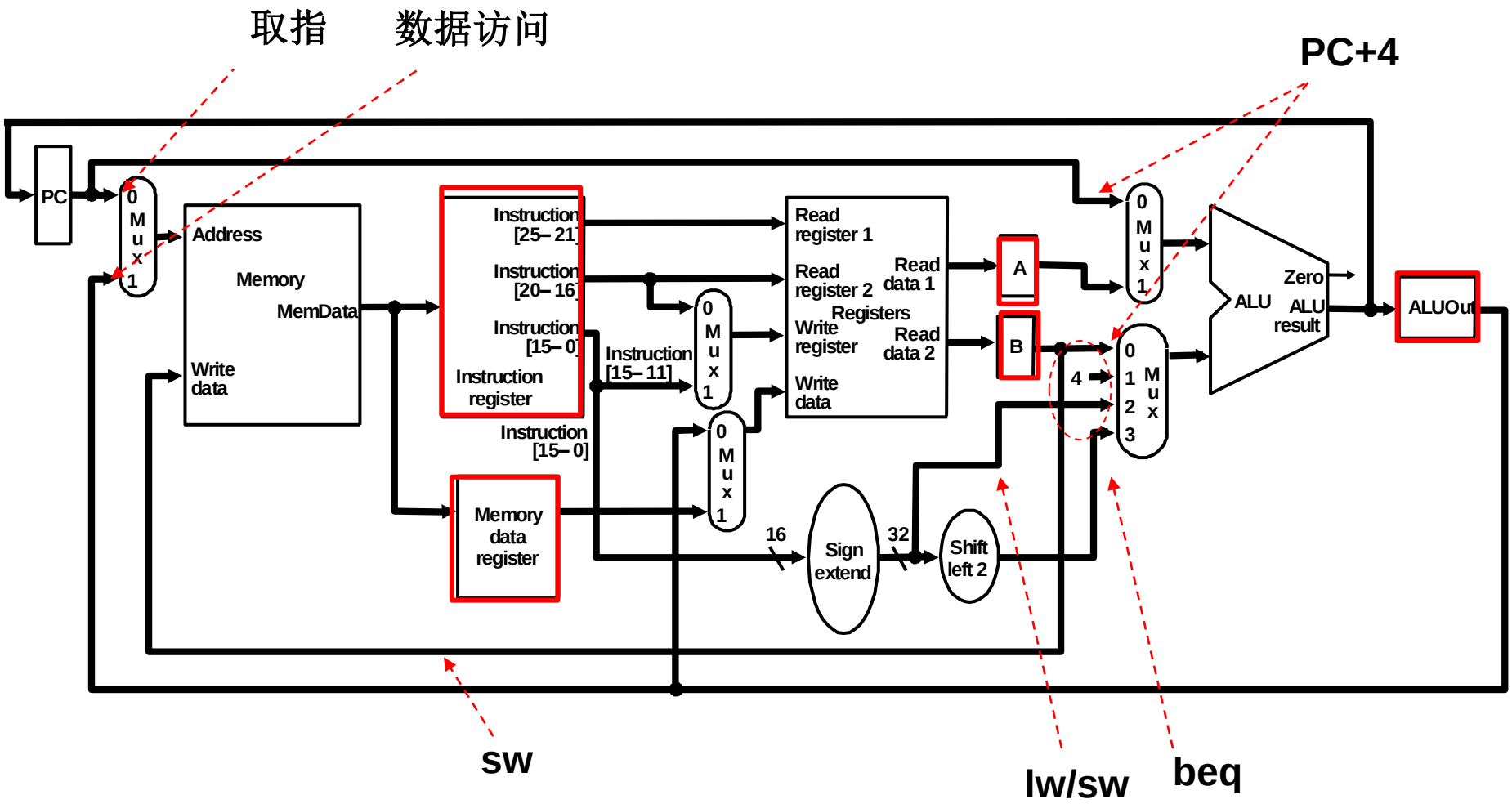
单周期和多周期数据通路区别



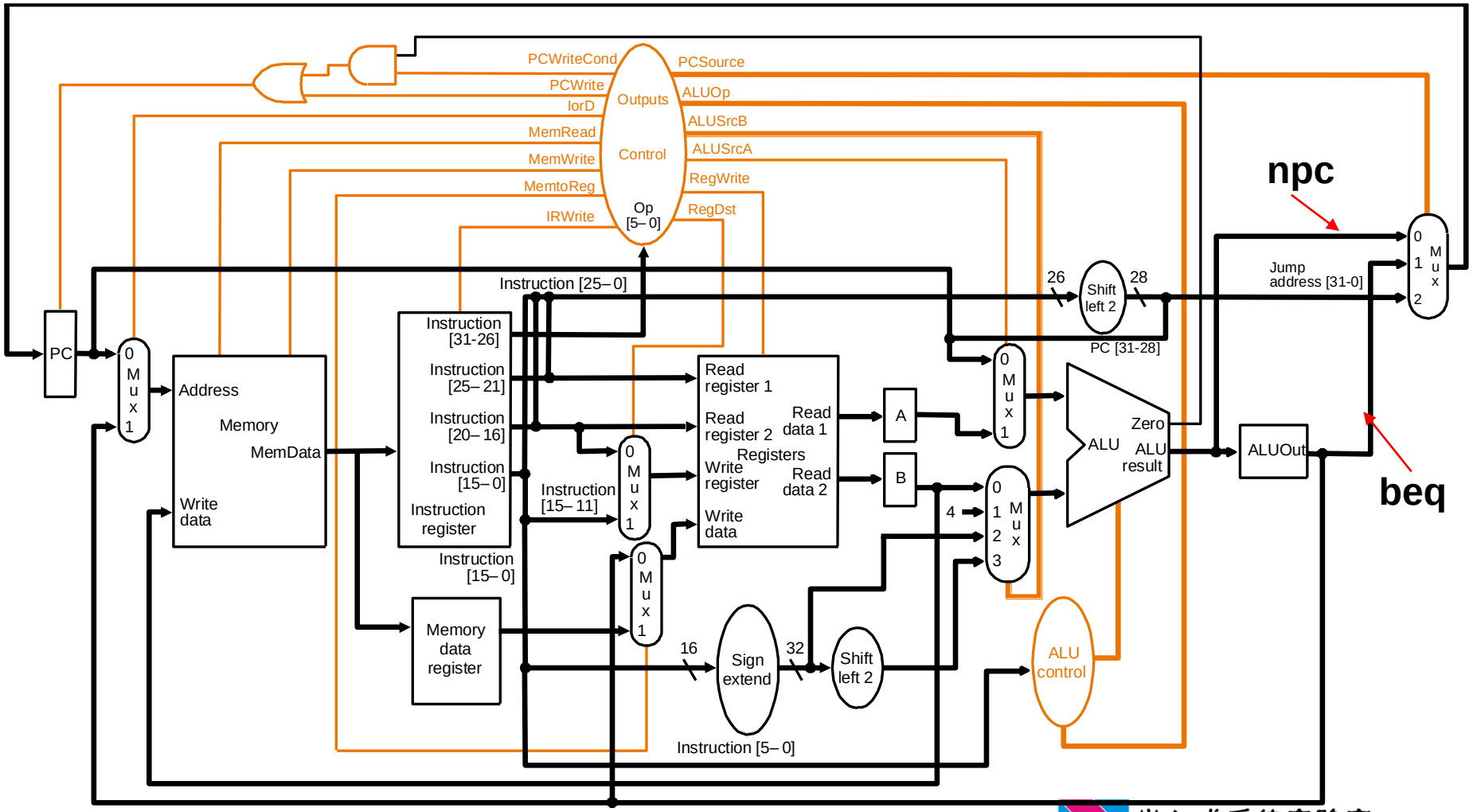
ALU、寄存器、存储器、控制信号



多周期数据通路


取指
数据访问
PC+4
PC
0
M
u
x
1
Address
Memory
MemData
Write
data
Instruction
[25– 21]
Instruction
[20– 16]
Instruction
[15– 0]
Instruction
register
Instruction
[15– 0]
Instruction
[15– 11]
Memory
data
register
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
A
B
4
16
Sign
extend
32
Shift
left 2
ALU
Zero
ALU
result
ALUOut
sw
lw/sw
beq

主控制部件

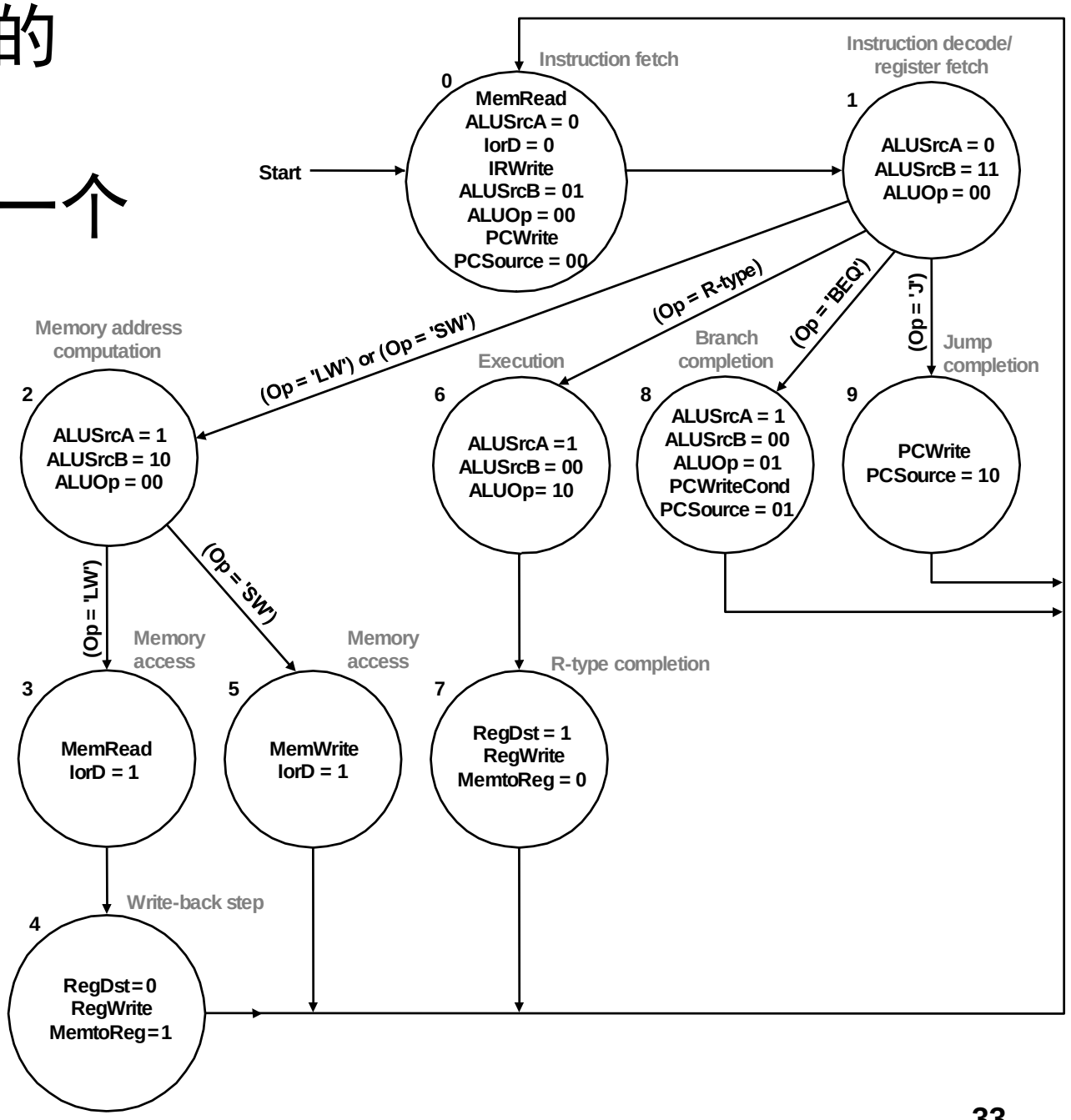

PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite
Outputs
Control
Op
[5– 0]
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
npc
beq
Jump
address [31-0]
Instruction [25– 0]
26
Shift
left 2
28
PC [31-28]
PC
0
M
u
x
1
Address
Memory
MemData
Write
data
Instruction
[31-26]
Instruction
[25– 21]
Instruction
[20– 16]
Instruction
[15– 0]
Instruction
register
Instruction
[15– 0]
Instruction
[15– 11]
Memory
data
register
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
A
B
4
ALU
Zero
ALU
result
ALUOut
ALU
control
16
Sign
extend
32
Shift
left 2
Instruction [5– 0]

Multicycle RTL

Step	R-Type	lw/sw	beq/bne	j
IF	IR = Mem[PC] PC = PC + 4			
ID	A = Reg[IR[25-21]] B = Reg[IR[20-16]] ALUOut = PC + (SE(IR[15-0]) << 2)			
EX	ALUOut = A op B	ALUOut = A +SE(IR[15-0])	If (A==B) then PC = ALUOut	PC = PC[31-28] || (IR[25-0]<<2
MEM	Reg[IR[15-11]] = ALUOut	MDR=Mem[ALUOut] Mem[ALUOut] = B		
WB		Reg[IR[20-16]] = MDR		

多周期控制器的MooreFSM，每个状态需要一个时钟周期。

- 取指：PC+4
- 译码: BEQ算地址
- 执行
  - ✓ R执行
  - ✓ LW/SW算地址
  - ✓ 分支完成
- 访存
  - ✓ R/SW完成
  - ✓ Load访存
- 写回
  - ✓ Load完成



Ch4-MIPS流水线

- 流水线技术原理
- MIPS的五级流水线实现
- 流水线的性能分析
- 流水线的“依赖”及其处理
  - ✓结构相关
  - ✓数据相关
  - ✓控制相关

要求：

1、能够设计并实现流水化的处理器（数据通路->控制信号->状态机），进行流水线的性能评估。
2、掌握各种相关产生的原因、解决方法

流水线的性能评价

吞吐率、加速比和效率的关系

TP = $n/T_{流水}$　　S=$T_{非流水}/T_{流水}$　　*最大加速比m*

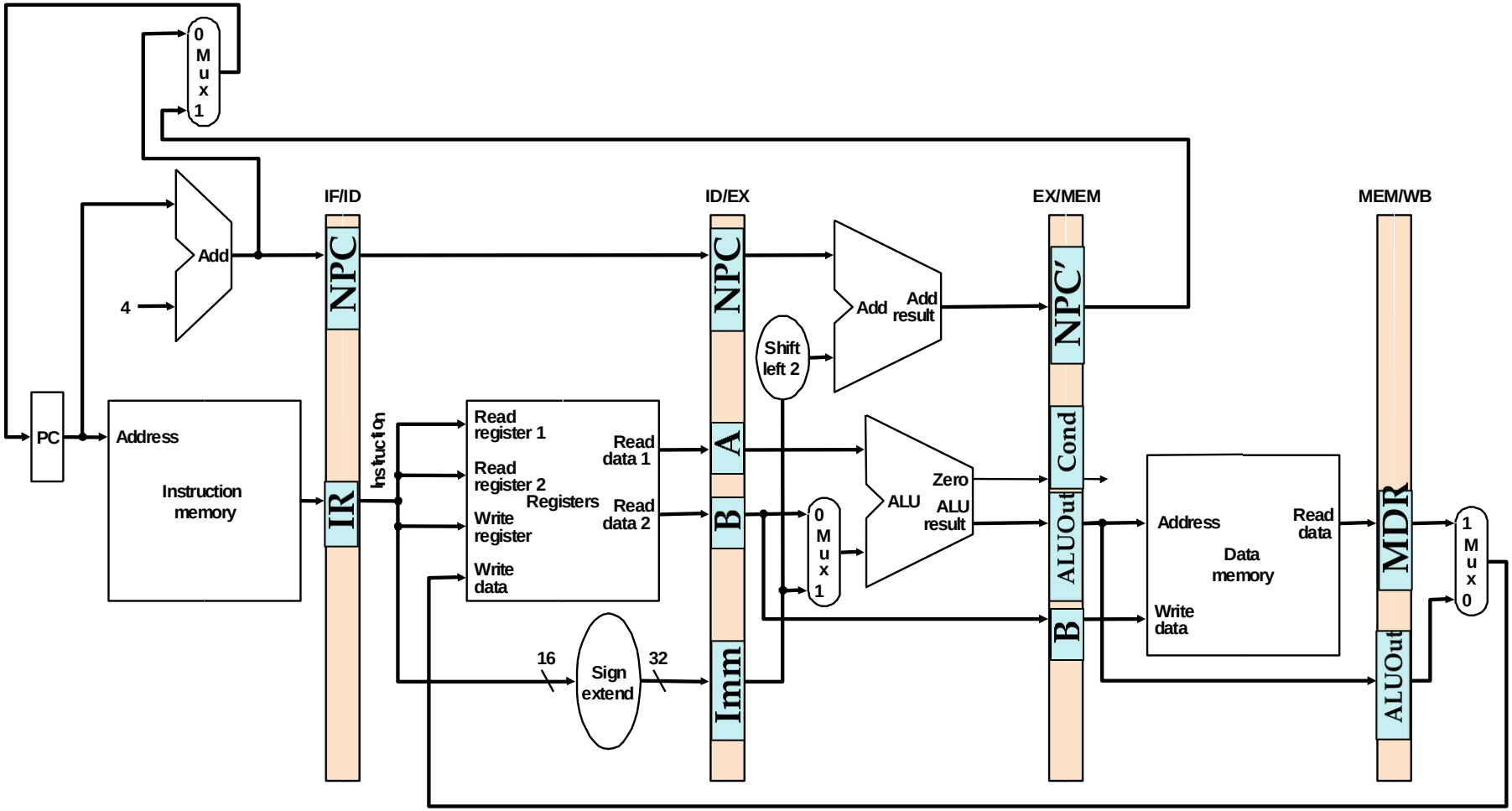
- $E = n\Delta t_0/T_{流水}=mn\Delta t_0/(T_{流水}m)= S/m$

  效率是实际加速比*S*与最大加速比*m*之比。

- $E = n\Delta t_0/T_{流水}= (n/T_{流水}) \cdot \Delta t_0=TP\Delta t_0$

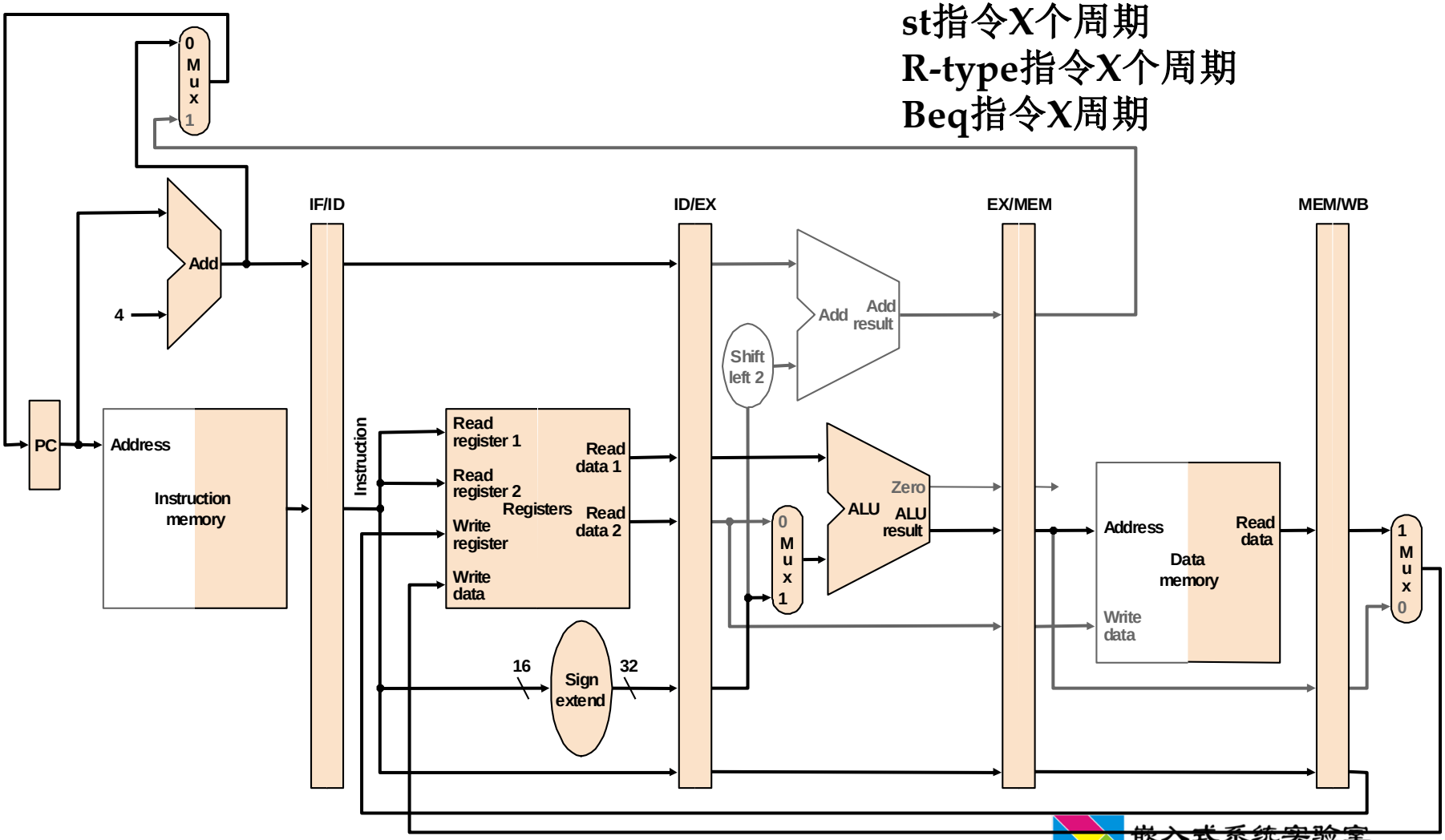
  当Δt_0不变时，流水线的效率与吞吐率呈正比。为提高效率而采取的措施，也有助于提高吞吐率。

流水线的段寄存器


0
M
u
x
1
Add
4
PC
Address
Instruction
memory
IF/ID
NPC
IR
Instruction
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
16
Sign
extend
32
ID/EX
NPC
A
B
Imm
Shift
left 2
Add
Add
result
0
M
u
x
1
ALU
Zero
ALU
result
EX/MEM
NPC′
Cond
ALUOut
B
Address
Data
memory
Write
data
Read
data
MEM/WB
MDR
ALUOut
1
M
u
x
0

各类指令在哪个周期结束

ld指令X周期
st指令X个周期
R-type指令X个周期
Beq指令X周期



MIPS 流水线的每个流水段的操作（示意，不要求）

流水段	任何指令类型		
IF	IF/ID.IR ← Mem[PC]; IF/ID.NPC ← PC+4; PC ← (if PCSrc {EX/MEM.NPC’} else {PC+4});		
ID	ID/EX.A ← Regs[IF/ID.$IR_{25...21}$]; ID/EX.B ← Regs[IF/ID.$IR_{20...16}$]; ID/EX.NPC ← IF/ID.NPC; ID/EX.IR ← IF/ID.IR; ID/EX.Imm ← (IF/ID.IR_{15})^{16}##$IR_{15...0}$;		
	ALU 指令(R类/I类)	Load/Store 指令	分支指令
EX	EX/MEM.IR ← ID/EX.IR; EX/MEM.ALUOut ← ID/EX.A op ID/EX.B; 或 EX/MEM.ALUOut ← ID/EX.A op ID/EX.Imm; EX/MEM.cond ← 0;	EX/MEM.IR ← ID/EX.IR; EX/MEM.B ←ID/EX.B; EX/MEM.ALUOutput ← ID/EX.A + ID/EX.Imm; EX/MEM.cond ← 0;	EX/MEM.NPC’ ← ID/EX.NPC+ID/EX.Imm<<2; EX/MEM.cond ← (ID/EX.A == ID/EX.B);

MIPS 流水线的每个流水段的操作（续）

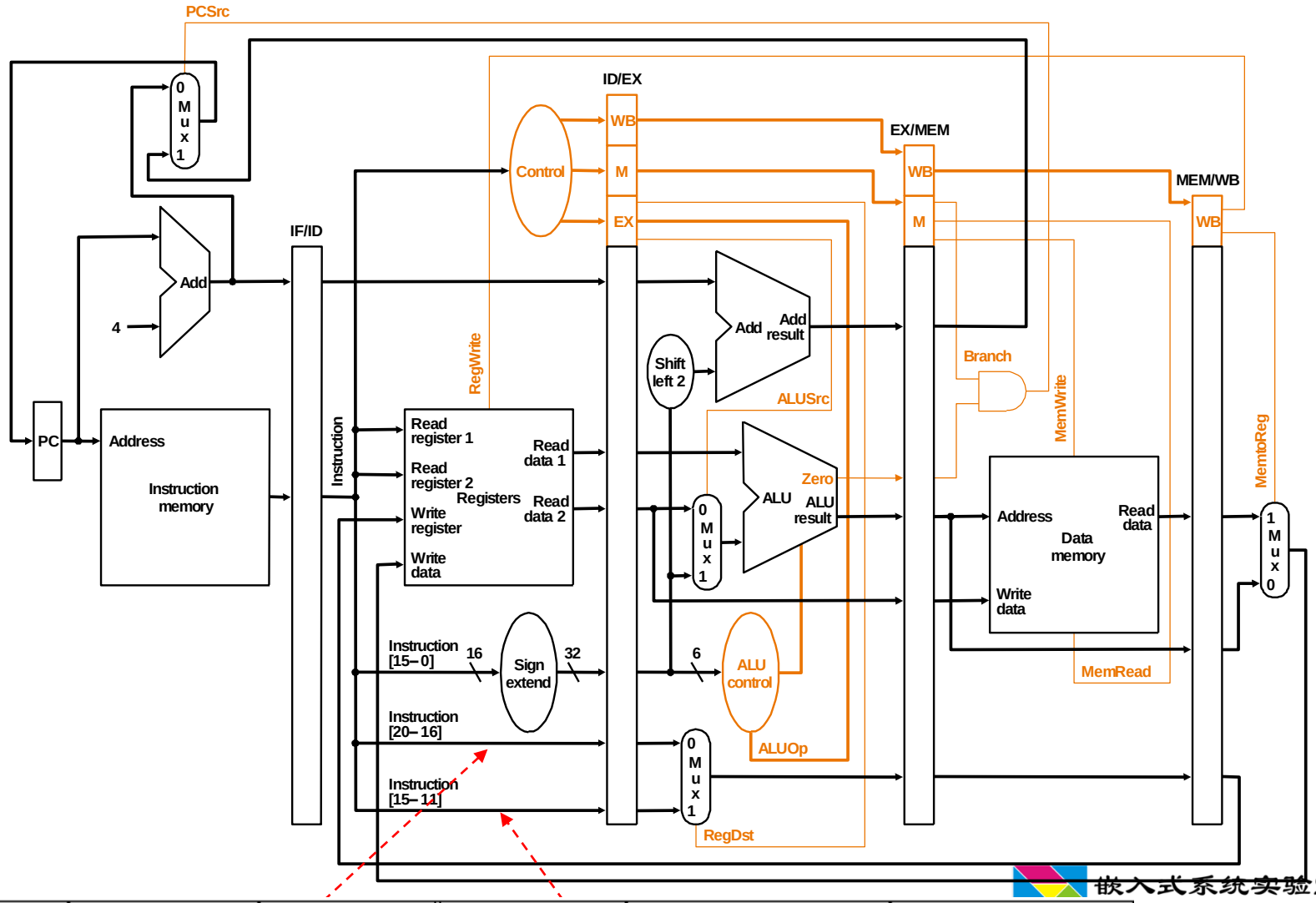
流水段	任何指令类型		
	ALU 指令	Load/Store 指令	分支指令
MEM	MEM/WB.IR ←EX/MEM.IR; MEM/WB.ALUOut ← EX/MEM.ALUOut;	MEM/WB.IR ← EX/MEM.IR; MEM/WB.MDR ← Mem[EX/MEM.ALUOut]; 或 Mem[EX/MEM.ALUOut] ← EX/MEM.B;	PCSrc ← EX/MEM.cond & EX/MEM.Branch;
WB	Regs[MEM/WB.$IR_{15...11}$] ← MEM/WB.ALUOut; (R) 或 Regs[MEM/WB.$IR_{20...16}$] ← MEM/WB.ALUOut; (I)	Regs[MEM/WB.$IR_{20...16}$] ← MEM/WB.MDR;	

流水线控制信号

- 所有控制信号名及其功能与非流水线版相同
  - ✓ 取指：读IM，写PC（每个周期写入一次， 不需控制信号）
  - ✓ 译码/寄存器读：没有控制信号
  - ✓ 执行/地址计算：RegDst，ALUOp，ALUSrc
  - ✓ 访存：Branch(=>PCSrc)，MemRead，MemWrite，
  - ✓ 写回：MemtoReg，RegWrite
  - ✓ 流水线段寄存器：每个周期写入一次，不需要单独的写控制
- 需要将控制分配给不同的流水线段

Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	RegDst	ALUOp1	ALUOp0	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

流水线寄存器的控制



op(6 bits)	rs(5 bits)	rt(5 bits)	rd(5 bits)	shamt(5 bits)	funct(6 bits)

数据相关

流水线带来的问题：结构、数据、控制相关

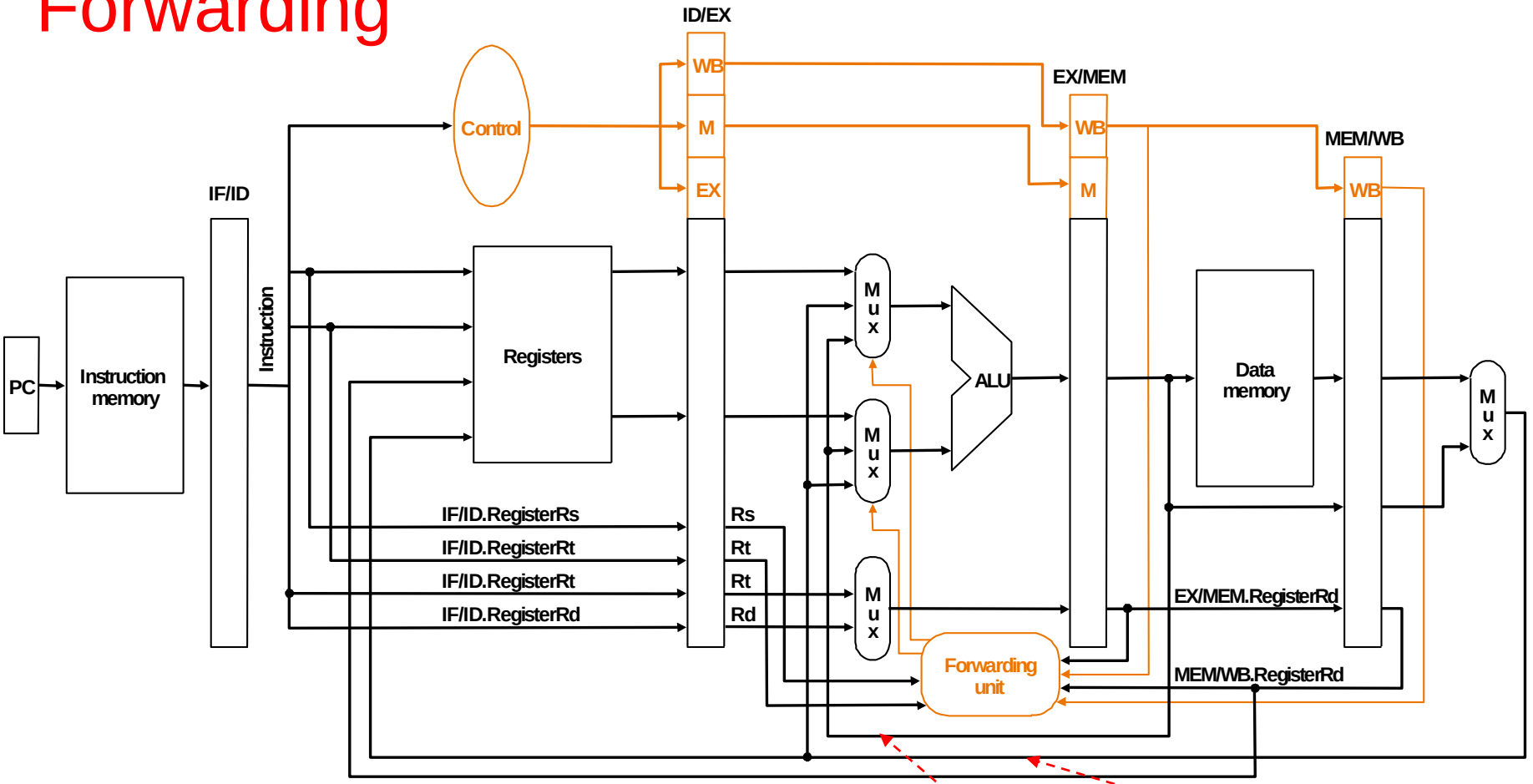
1. 数据相关简介

实例：

```
SUB  $2,  $1 ,  R3
AND  $12,  $2 ,  $5
OR   $13,  $6 ,  $2
ADD  $14,  $2 ,  $2
SW   $15,  100($2)
```

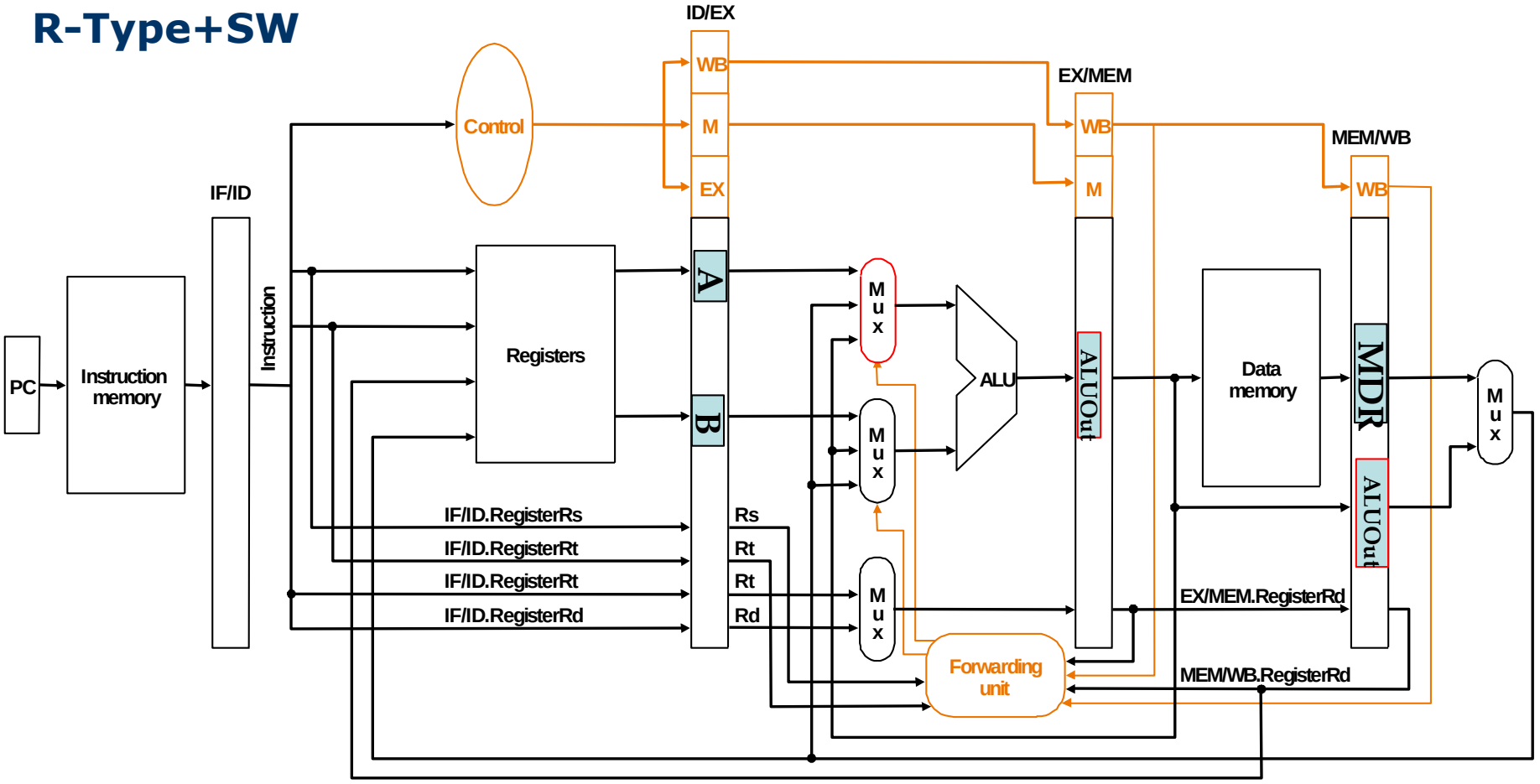
产生原因：当指令在流水线中重叠执行时，流水线有可能改变指令读/写操作数的顺序，使之不同于它们在非流水实现时的顺序，这将导致数据相关。

Forwarding



Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	RegDst	ALUOp1	ALUOp0	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

R-Type+SW

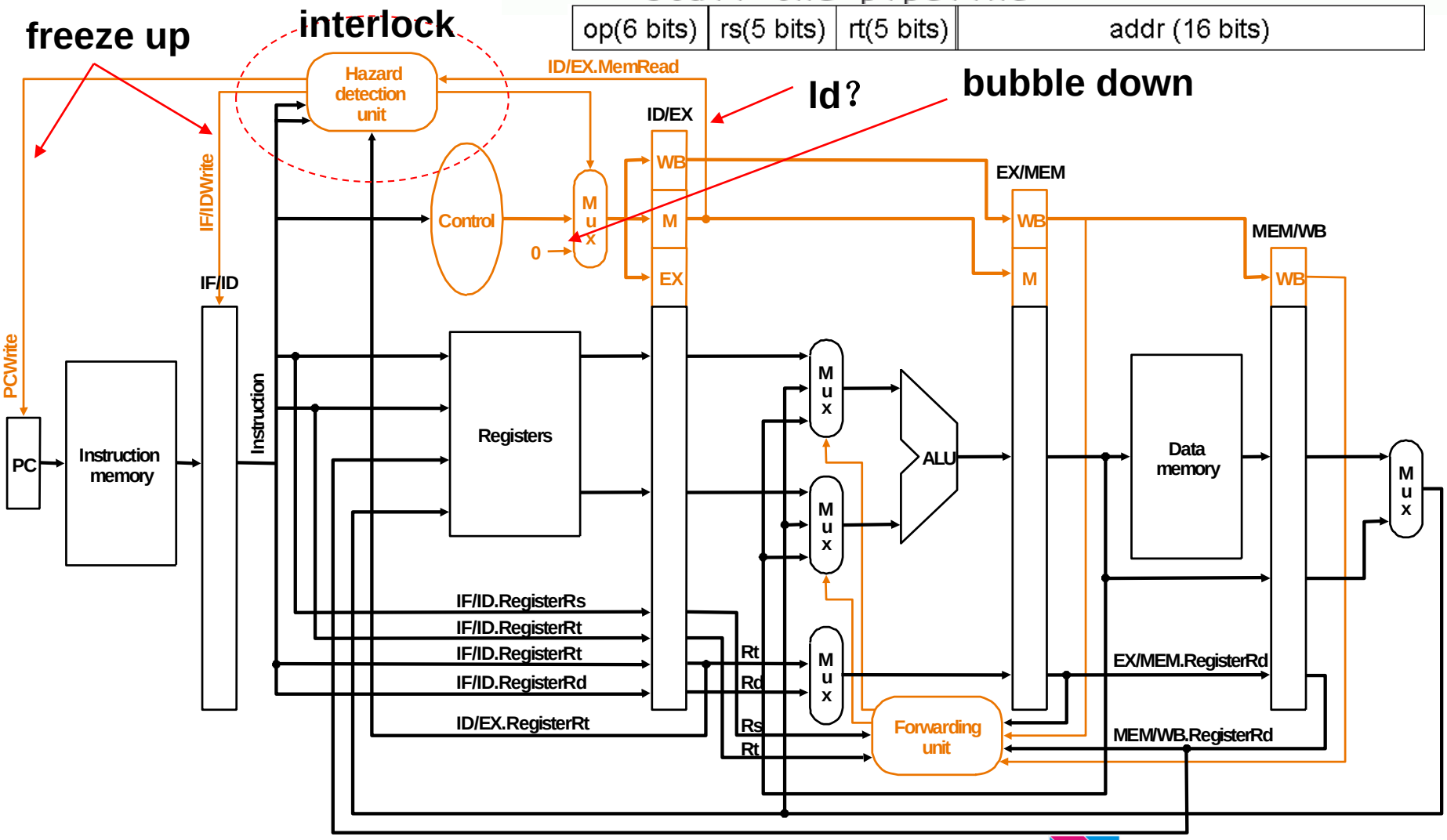



```
ADD  $1, $2, $3    IF  ID   EX   ME  WB
SW   $5, 30($1)         IF   ID   EX  ME  WB
SW   $6, 45($1)              IF   ID  EX  ME  WB
```

无法解决的相关：Interlock互锁

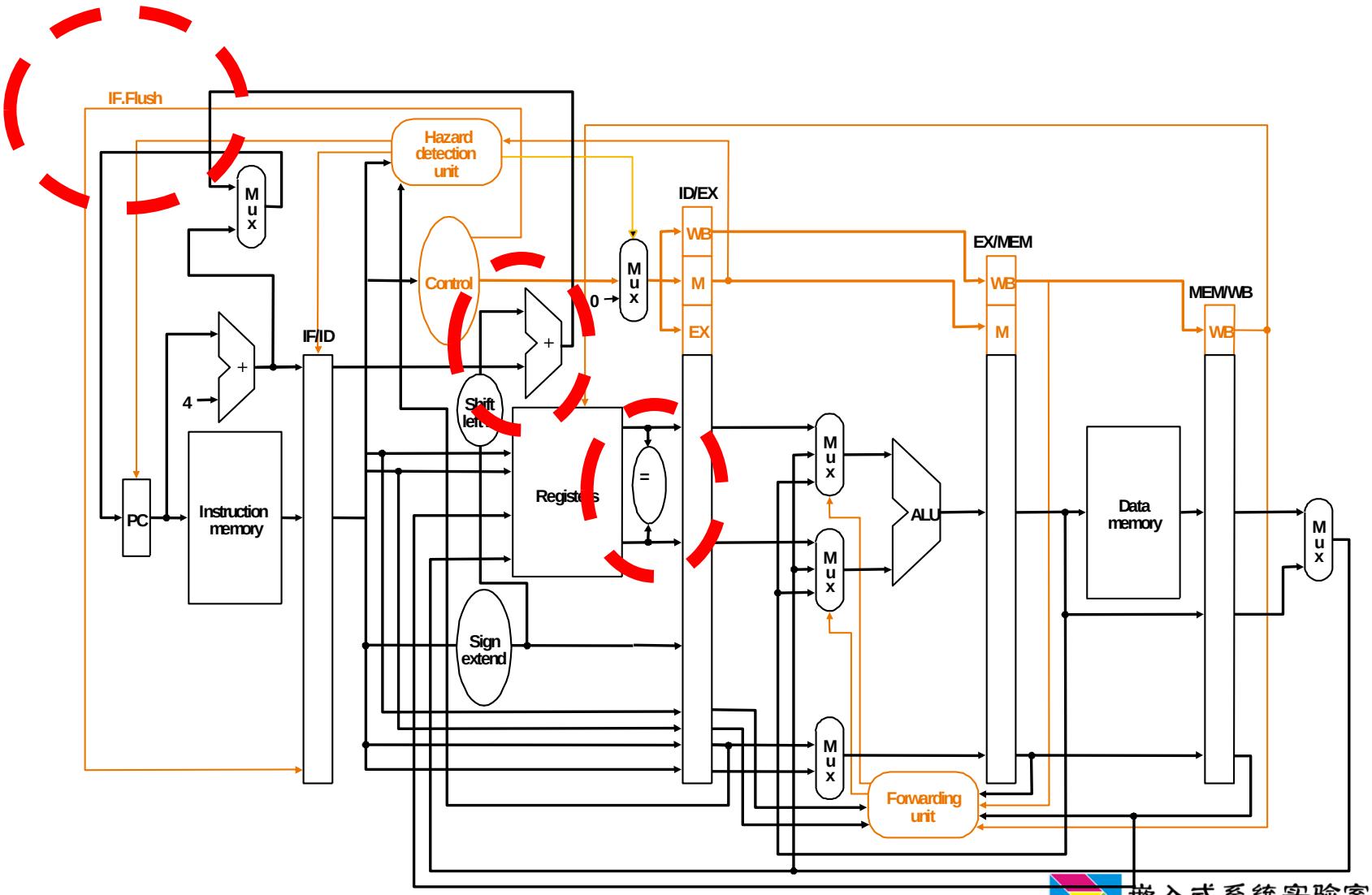
```
if (ID/EX.MemRead and
    ((ID/EX.RegisterRt = IF/ID.RegisterRs) or
     (ID/EX.RegisterRt = IF/ID.RegisterRt)))
      stall the pipeline
```

op(6 bits)	rs(5 bits)	rt(5 bits)	addr (16 bits)

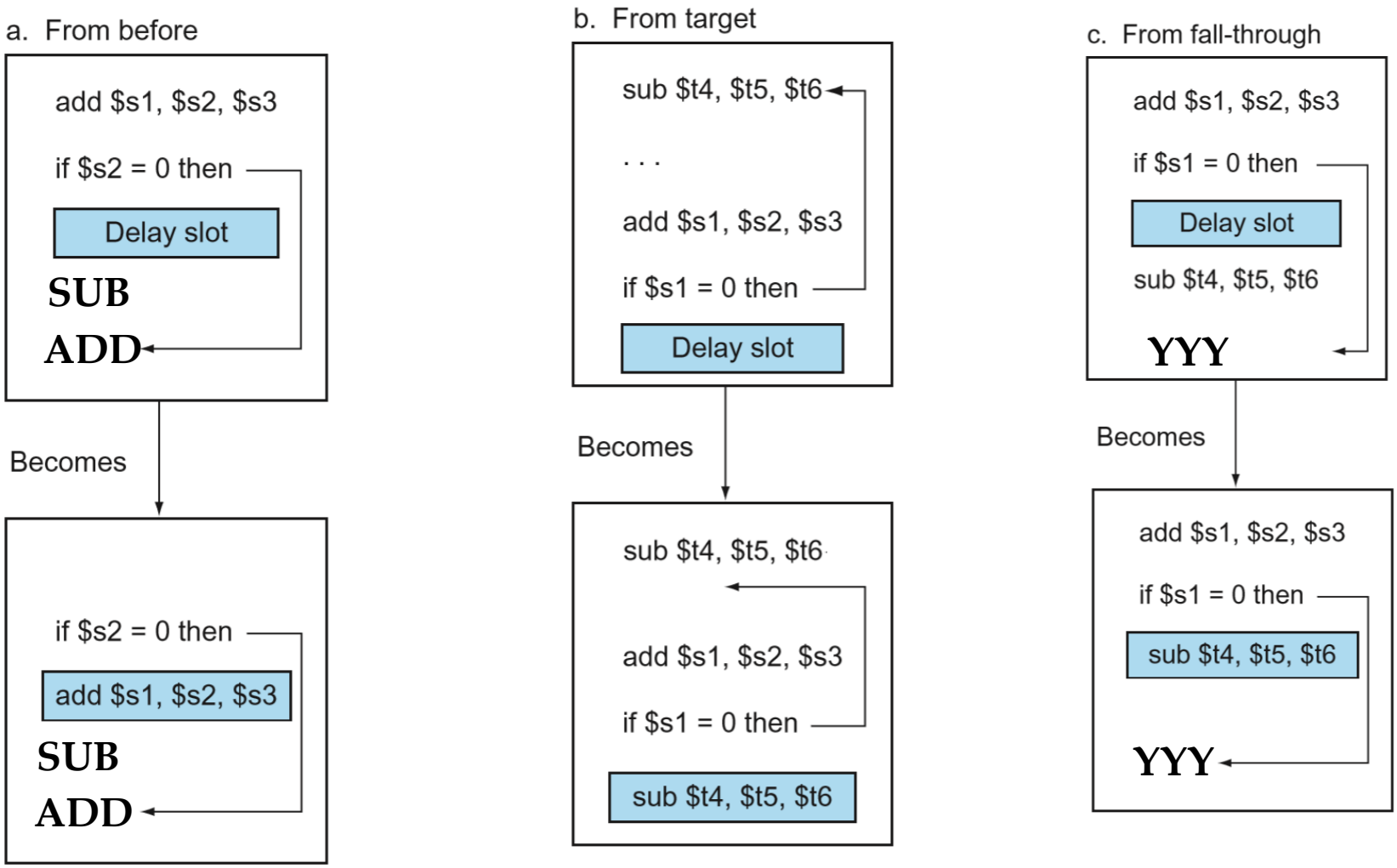


op(6 bits)	rs(5 bits)	rt(5 bits)	rd(5 bits)	shamt(5 bits)	funct(6 bits)

控制相关：1、降低延迟开销



2、分支优化技术-分支延迟



3、动态分支预测（2位）

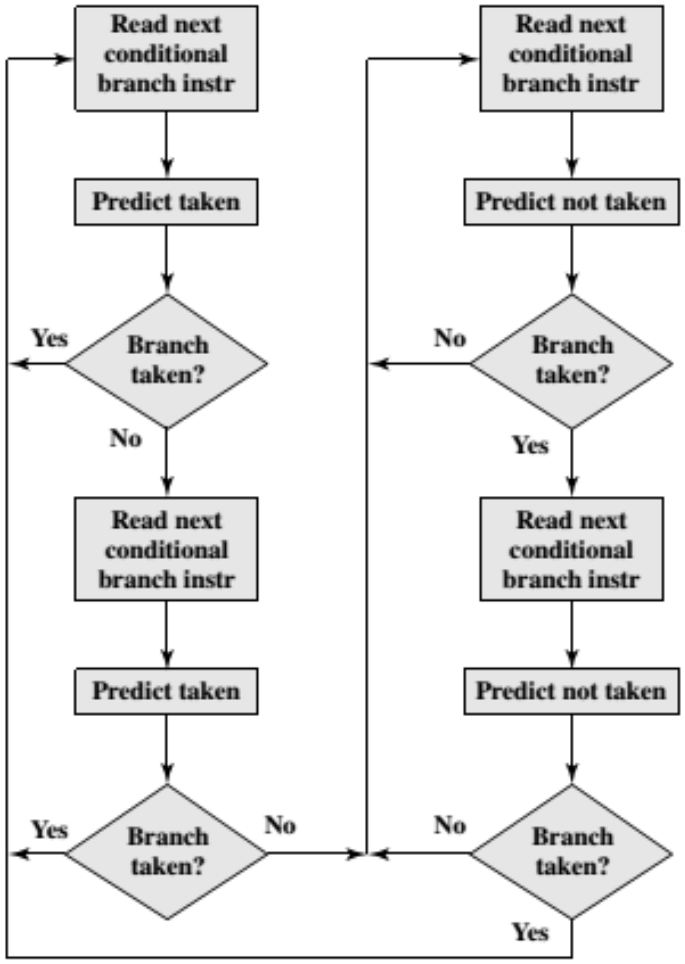
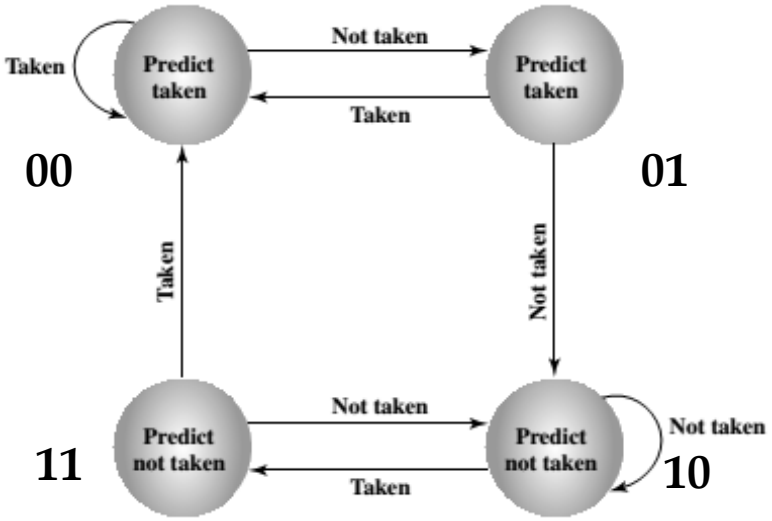
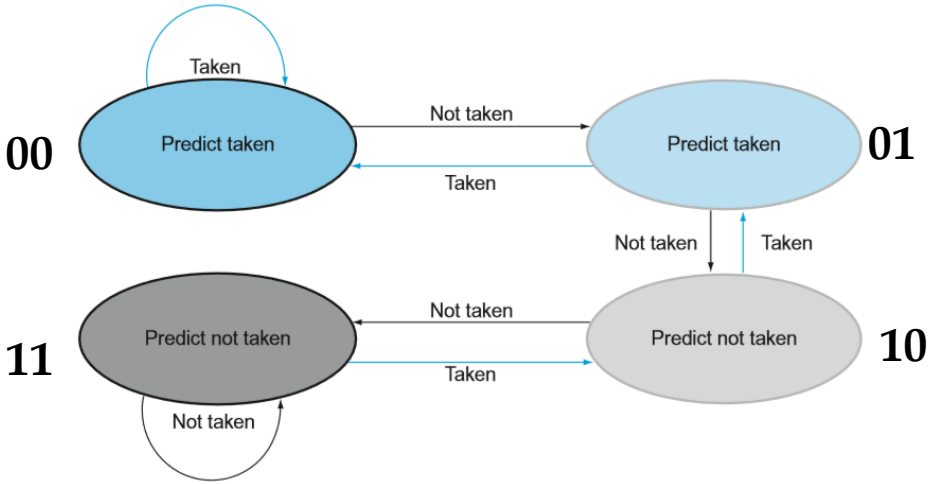


Figure 12.18 Branch Prediction Flowchart



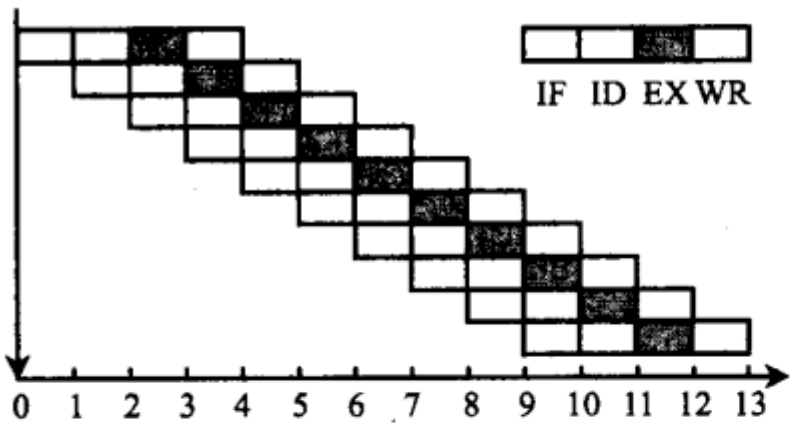


流程图：无时序　　2位预测器　　状态图：有时序

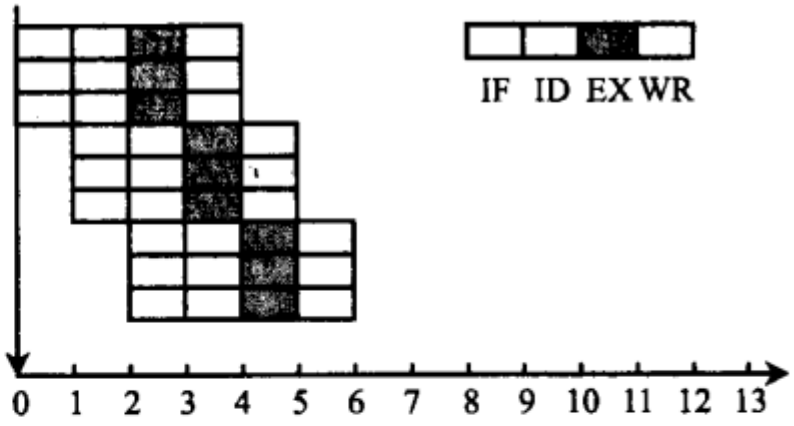
相关总结

- 结构相关
  - ✓ 原因：硬件资源不足产生的冲突
  - ✓ 解决方案：等待（软件编译器，硬件Stall）
  - ✓ 解决方案：哈佛结构（存储器相关）
- 数据相关
  - ✓ 原因：数据依赖
  - ✓ 解决方案：等待（软件编译器，硬件Stall）
  - ✓ 解决方案：定向路径（RAW相关）
  - ✓ 解决方案：流水线互锁（LW+指令）
- 控制相关
  - ✓ 原因：分支指令引起的PC冲突
  - ✓ 解决方案：等待（软件编译器，硬件Stall）、硬件ID段检测
  - ✓ 解决方案：延迟分支（三种调度方法）
  - ✓ 解决方案：分支预测（一位、两位）

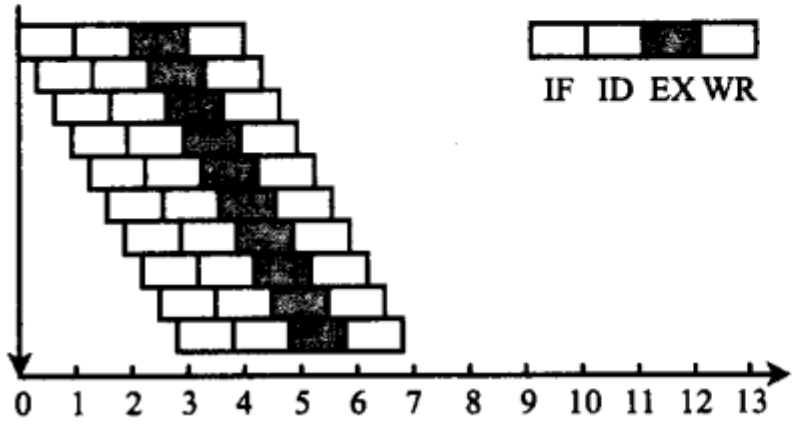
四种流水技术比较



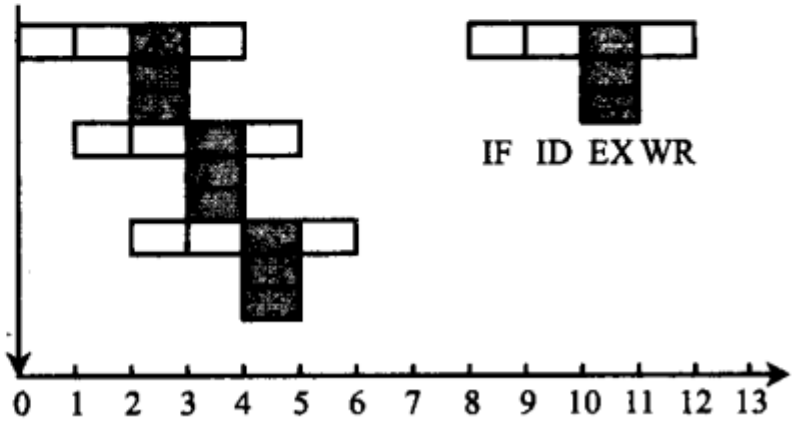
(a) 普通流水



(b) 超标量流水



(c) 超流水线



(d) 超长指令字

Ch5 中断与异常

- 中断的基本概念
- 中断要解决的若干问题
- MIPS流水线异常实现

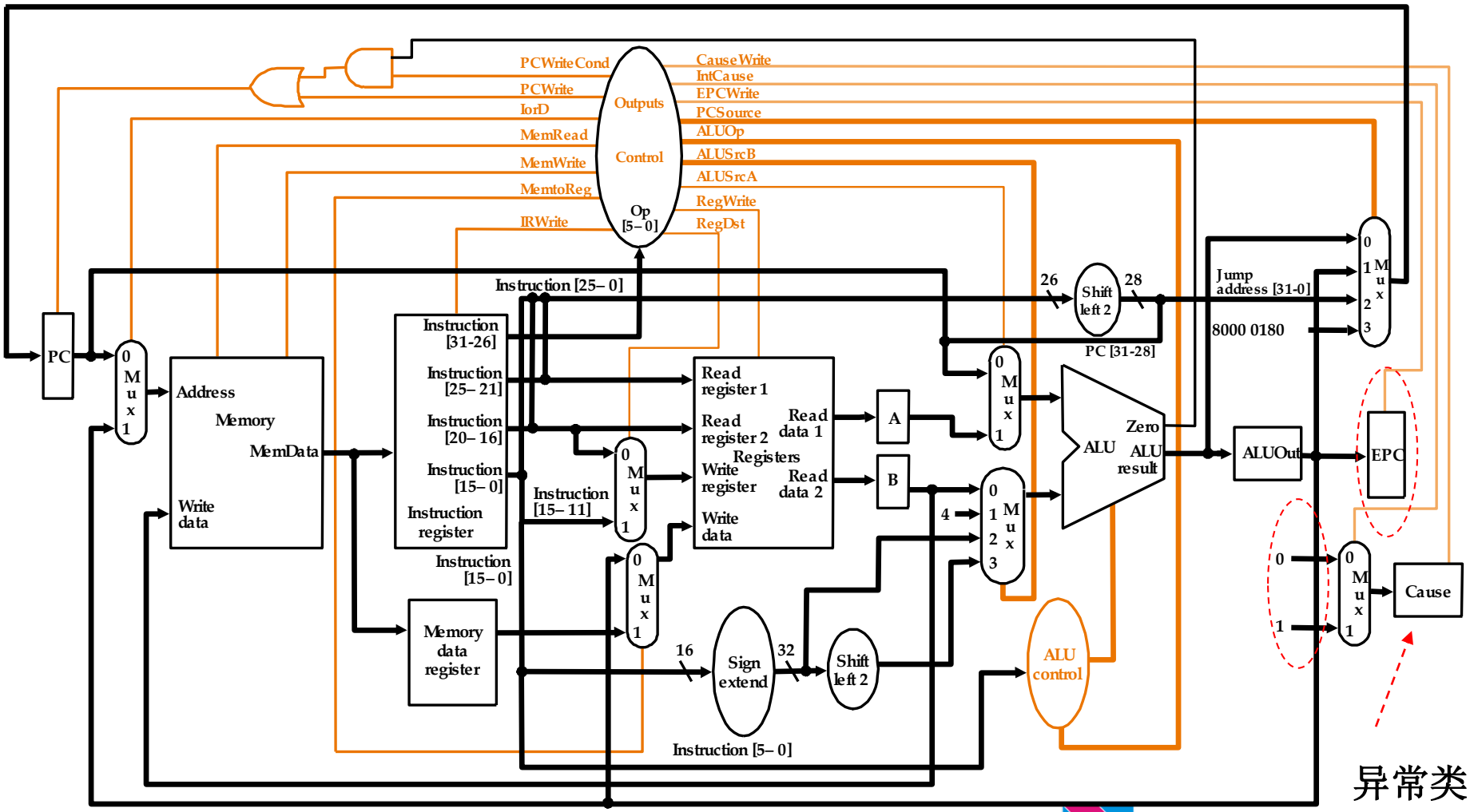
要求：
1、掌握中断的基本概念（与IO相结合）
2、实现支持异常处理的MIPS流水线（数据通路->控制信号->状态机）

中断管理需要解决的问题

1. 中断源如何向CPU提出中断请求（软/硬中断）
2. 多个中断请求时如何确定优先响应顺序
3. CPU响应中断的条件、时间点、方式
4. 响应中断后如何保护现场
5. 如何转入中断服务程序执行
6. 如何恢复现场，如何返回断点处执行
7. 中断处理过程中出现新的中断请求如何处理

重点：中断机构

MIPS多周期模式的异常处理



异常类型

两种异常的处理

Step	R-Type	lw/sw	beq/bne	j	Exception
IF	IR = Mem[PC] PC = PC + 4				
ID	A = Reg[IR[25-21]] B = Reg[IR[20-16]] ALUOut = PC + (SE(IR[15-0]) << 2)				
EX	ALUOut = A op B	ALUOut = A +SE(IR[15-0])	If (A==B) then PC = ALUOut	PC = PC[31-28] || (IR[25-0]<<2	
MEM	Reg[IR[15-11]] = ALUOut	MDR=Mem[ALUOut] Mem[ALUOut] = B			
WB		Reg[IR[20-16]] = MDR			

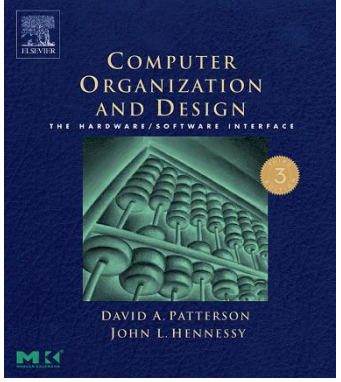
异常指令

PC=0X80000000
EPC=PC-4
CAUSE=0

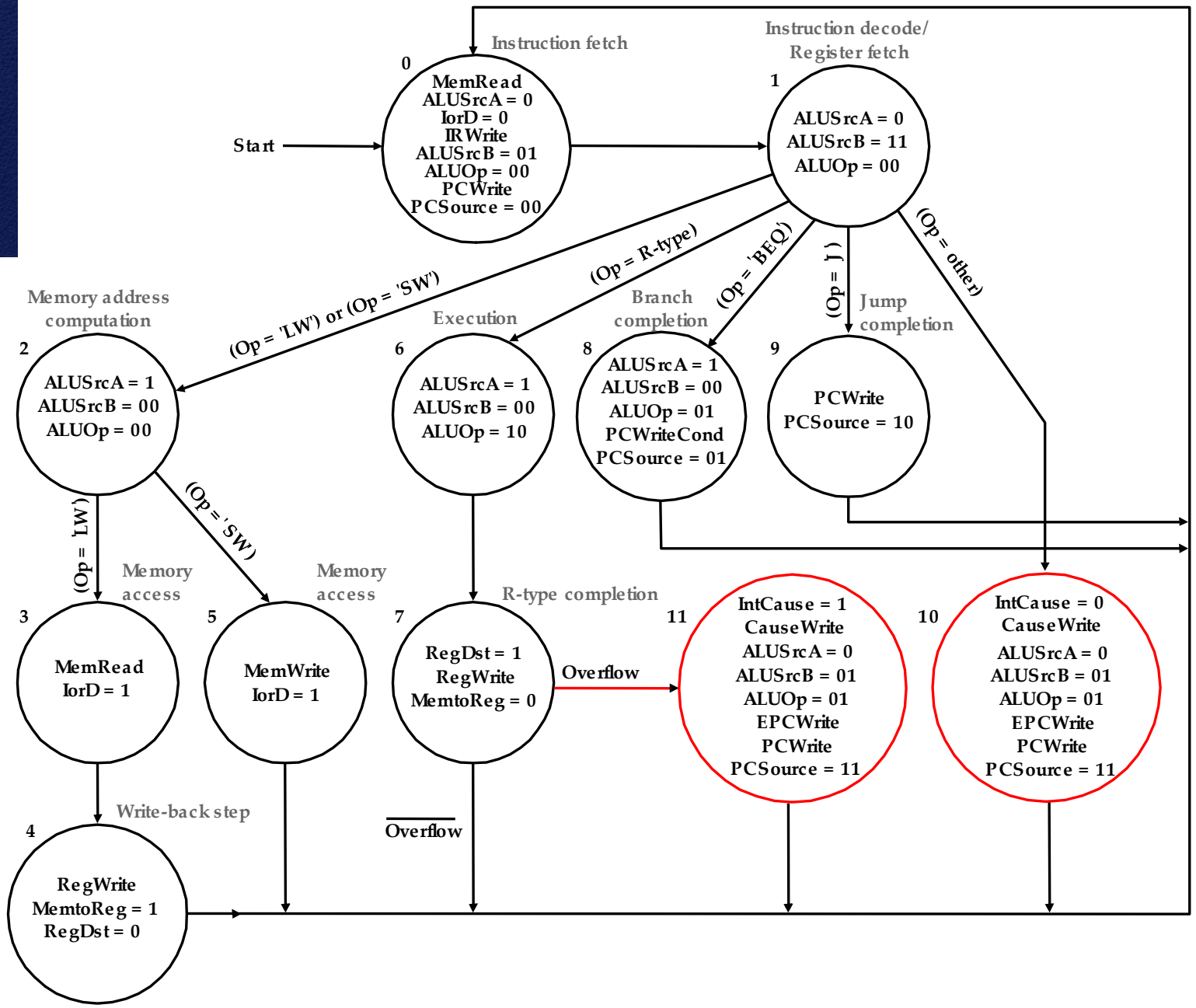
算术溢出
PC=0X80000180
EPC=PC-4
CAUSE=1

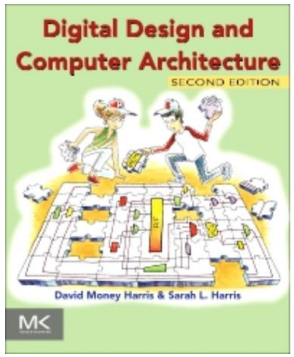
两种异常的处理

问：算术溢出应该在哪个周期？

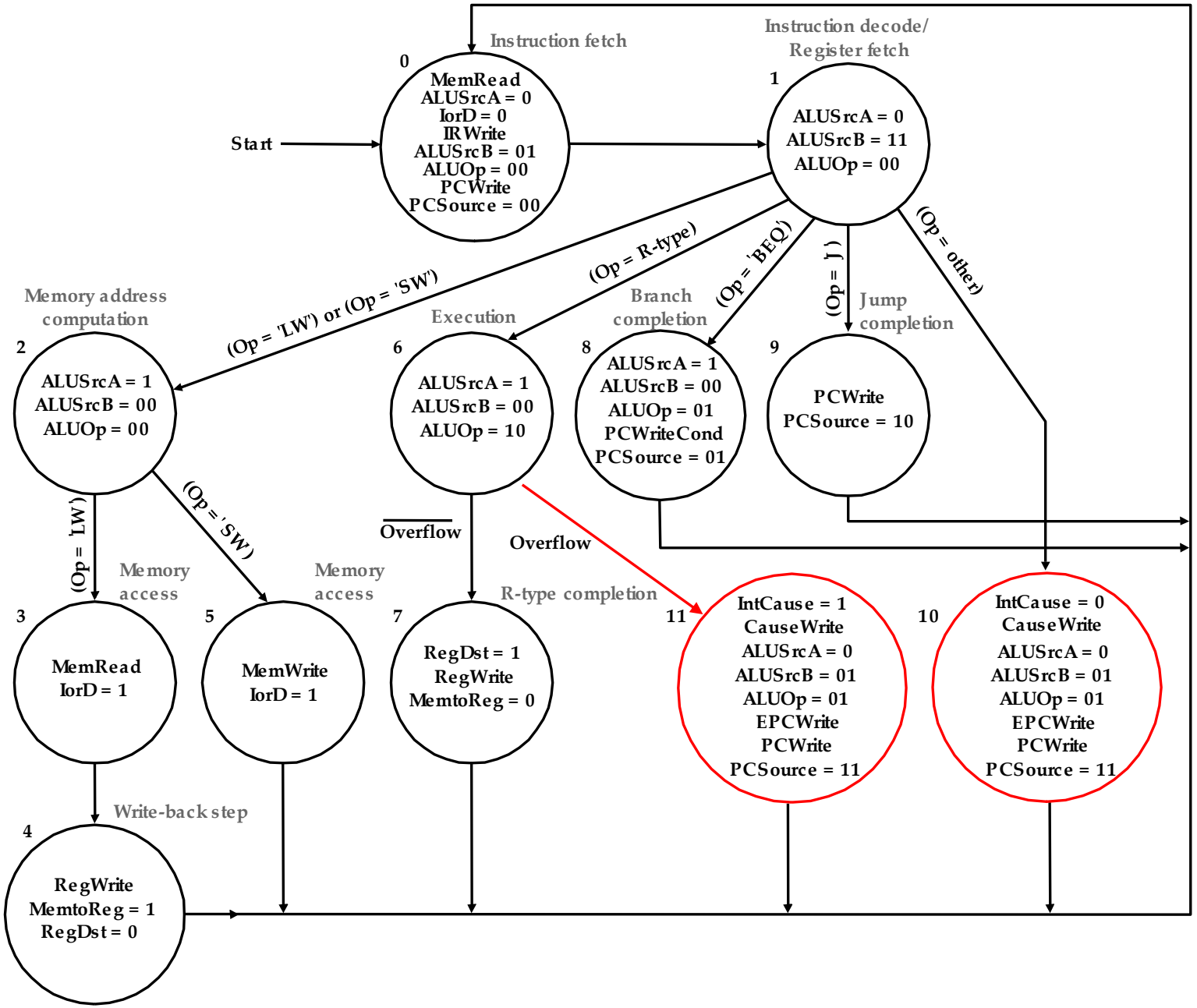

COMPUTER
ORGANIZATION
AND DESIGN
THE HARDWARE/SOFTWARE INTERFACE
DAVID A. PATTERSON
JOHN L. HENNESSY

异常处理控制


Start
Instruction fetch
0
MemRead
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = 00
PCWrite
PCSource = 00
Instruction decode/
Register fetch
1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00
(Op = 'LW') or (Op = 'SW')
(Op = R-type)
(Op = 'BEQ')
(Op = 'J')
(Op = other)
Memory address
computation
2
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 00
Execution
6
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 10
Branch
completion
8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond
PCSource = 01
Jump
completion
9
PCWrite
PCSource = 10
(Op = 'LW')
(Op = 'SW')
Memory
access
3
MemRead
IorD = 1
Memory
access
5
MemWrite
IorD = 1
R-type completion
7
RegDst = 1
RegWrite
MemtoReg = 0
Overflow
11
IntCause = 1
CauseWrite
ALUSrcA = 0
ALUSrcB = 01
ALUOp = 01
EPCWrite
PCWrite
PCSource = 11
10
IntCause = 0
CauseWrite
ALUSrcA = 0
ALUSrcB = 01
ALUOp = 01
EPCWrite
PCWrite
PCSource = 11
Write-back step
4
RegWrite
MemtoReg = 1
RegDst = 0
Overflow

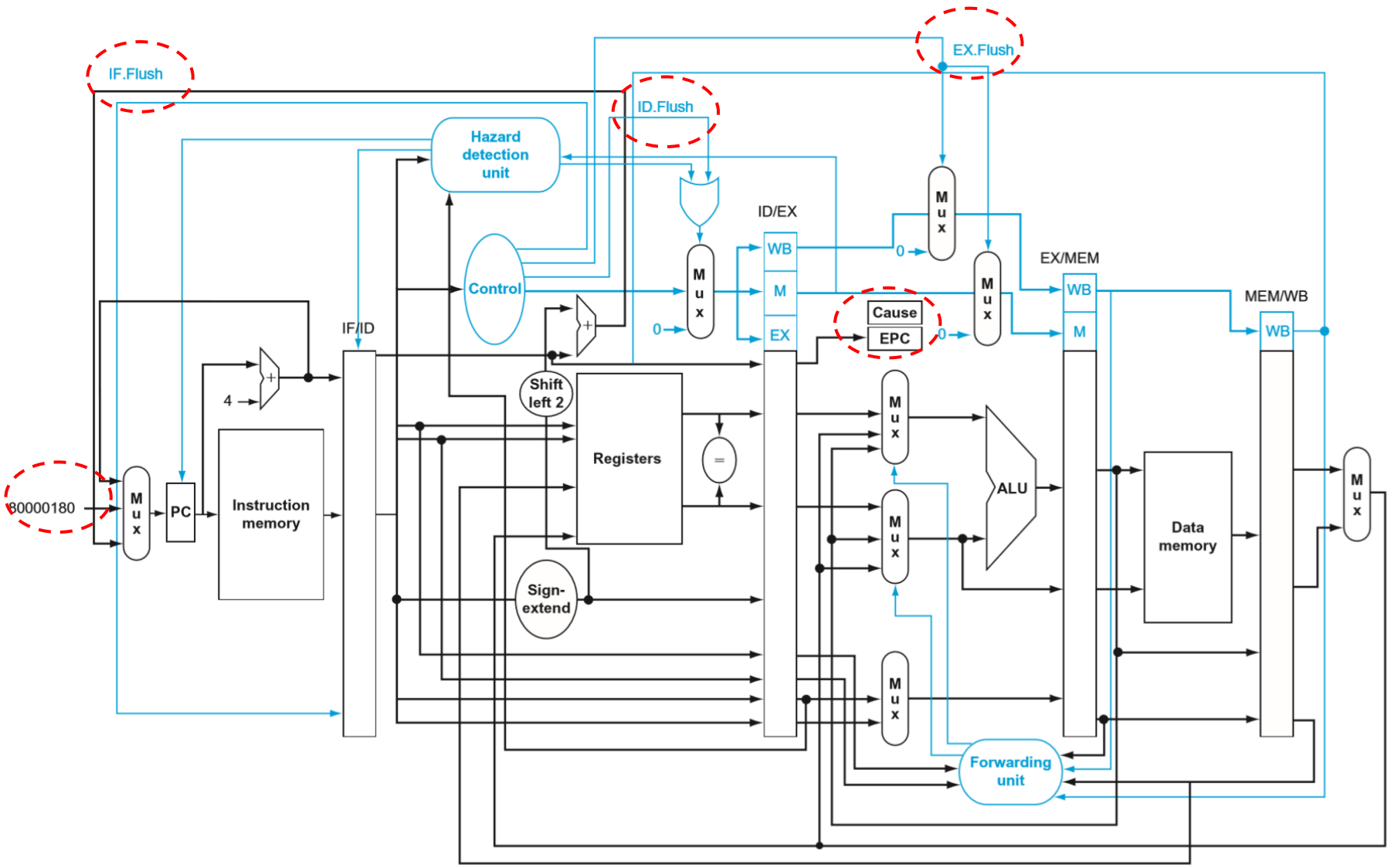


异常处理控制



Digital Design and Computer Architecture, Second Edition，David Harris， Sarah Harris.

在MIPS流水线中非精确-例子



Ch6-存储系统主要内容

1. 存储器概述
   - 1.1 存储器的分类
   - 1.2 存储系统的层次结构

2. 主存储器
   - 2.1 主存概述
   - 2.2 半导体存储芯片简介
   - 2.3 SRAM存储器
   - 2.4 DRAM存储器
   - 2.5 ROM只读存储器
   - 2.6 存储器与CPU的连接
   - 2.7 并行存储（1）— 双端口存储器
   - 2.8 并行存储（2）— 多模块交叉

3. 高速缓冲存储器Cache
   - 3.1 Cache的基本原理
   - 3.2 Cache的基本结构
   - 3.3 Cache与主存的地址映射
   - 3.4 Cache存储块的替换策略
   - 3.5 Cache写策略
   - 3.6 Cache组织举例

4. 虚拟存储器
   - 4.1 虚拟存储器的基本概念
   - 4.2 页式虚拟存储器
   - 4.3 段式虚拟存储器
   - 4.4 段页式虚拟存储器
   - 4.5 虚存的替换算法

要求：

1、掌握缓存、虚存的原理和概念。

2、实现硬件的Cache控制器。

存储器与CPU的连接示例一（1）

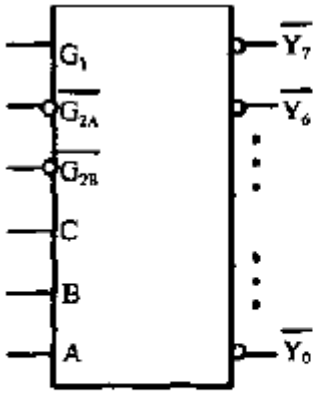
- 例 设CPU有16根地址线、8根数据线，并用$\overline{MREQ}$作为访存控制信号（低电平有效），用WR作为读/写控制信号（高电平为读，低电平为写）。现有下列存储芯片：1K×4位RAM、4K×8位RAM、8K×8位RAM、2K×8位ROM、4K×8位ROM、8K×8位ROM及74138译码器和各种门电路。画出CPU与存储器的连接图，要求如下：

  1）主存地址空间分配为：

  6000H ~ 67FFH为系统程序区；6800H ~ 6BFFH为用户程序区

  2）合理选用上述存储芯片，说明各选几片

  3）详细画出存储芯片的片选逻辑图

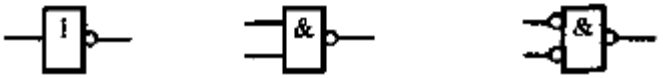


74 138 译码器

G_1, $\overline{G_{2A}}$, $\overline{G_{2B}}$为控制端

C, B, A 为变量输入端

$\overline{Y_0}$…$\overline{Y_7}$为变量输出端



校验

- 奇偶校验
- CRC校验
- 海明校验

P3= D4⊕D3 ⊕D2
0 = 1 ⊕ 0 ⊕ 1
P2= D4 ⊕D3 ⊕D1
0 = 1 ⊕ 0 ⊕ 1
P1 = D4 ⊕D2 ⊕D1
1 = 1 ⊕ 1 ⊕ 1
最后，海明码为**1010101**

- 海明码的接收端的公式:
  - ✓ $S_3= P_3⊕ D_4⊕D_3 ⊕D_2$
    $S_2= P_2⊕D_4 ⊕D_3 ⊕D_1$
    $S_1= P_1⊕D_4 ⊕D_2 ⊕D_1$
  - ✓ 假定 海明码1010101在传送中变成了1000101
    $S_3= P_3⊕ D_4⊕D_3 ⊕D_2=0⊕1⊕0 ⊕0 =1$
    $S_2= P_2⊕D_4 ⊕D_3 ⊕D_1=0⊕1⊕ 0 ⊕1=0$
    $S_1= P_1⊕D_4 ⊕D_2 ⊕D_1=1⊕1⊕ 0 ⊕1=1$
    因此,由$S_3S_2S_1$= 101,指出第5位错,应由0变1

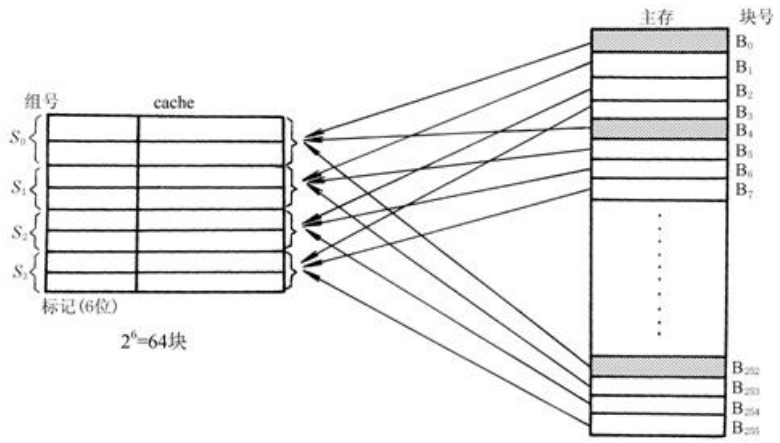
有效信息为1010,生成多项式G(x)=1011,将其编成CRC码。

Cache的映射策略

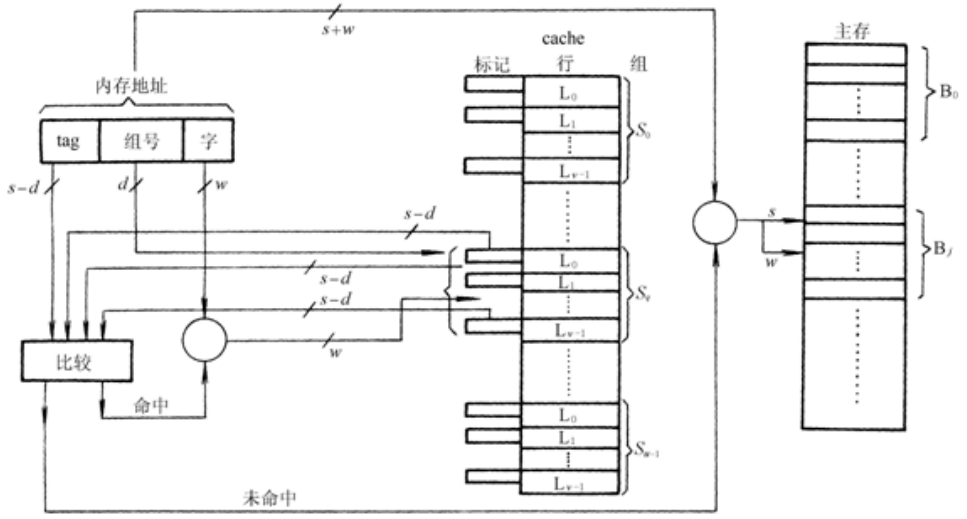
□地址映射（例子）

✓地址映射的方式（Cache组织方式）

- 全相联映射——灵活性大的映射关系
- 直接映射——固定的映射关系
- 组相联映射——前两种的折中



(a) 组相联映射示意图(4组)



(b) 组相联cache的检索过程

Cache替换策略

替换策略

✓替换产生的原因

- Cache容量比主存小得多，某个新的主存块要调入Cache时，其对应可存放的Cache块可能都已经被使用

✓替换策略与Cache组织方式密切相关

- 直接映射，直接替换对应Cache块即可
- 全相联和组相联映射，选择某一块——替换算法

✓常用的替换算法

- **先进先出FIFO算法**——选择最早调入Cache的块进行替换
- **随机法**——利用随机数产生器，随机选择被替换的块
- **最少使用LFU算法**——被访问的块计数器**增加1**，替换时选择计数值**最小**的块（同时清零其计数器），不能反映cache近期访问情况
- **最近最少使用LRU算法**——被访问的块计数器**置0**，其他块的计数器**增加1**，替换时选择计数值**最大**的块，符合cache的工作原理

Cache替换策略

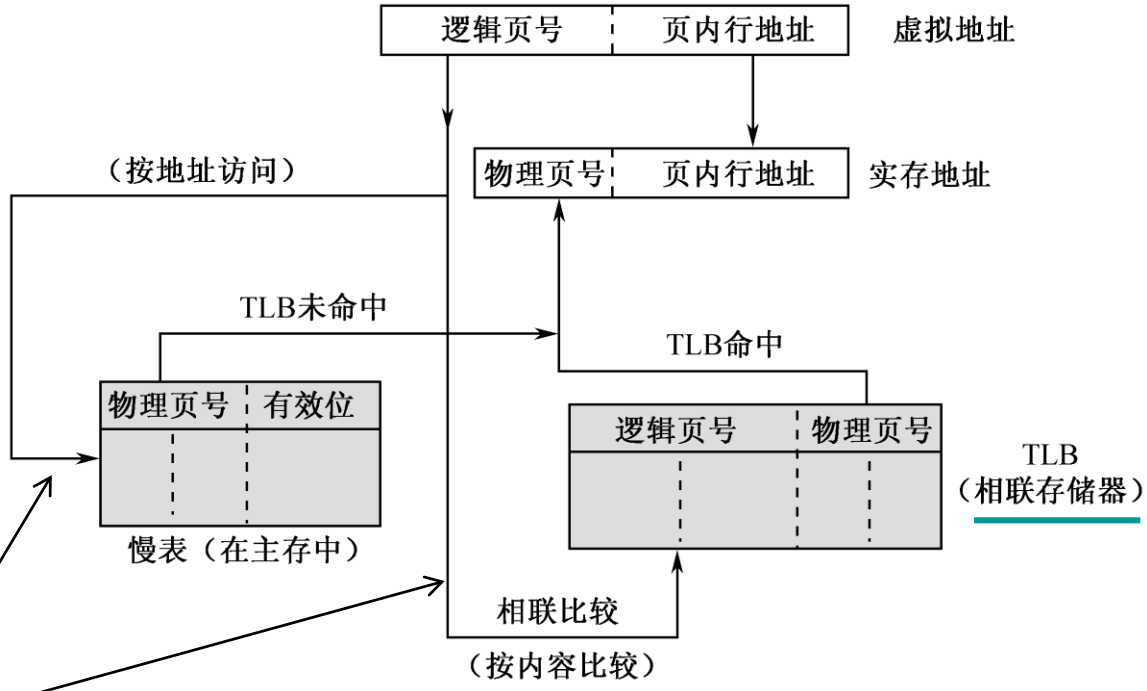
- 例 设cache有1、2、3、4共4个块，a、b、c、d等为主存中的块,访问顺序一次如下：a、b、c、d、b、b、c、c、d、d、a ,下次若要再访问e块。
  问，采用LFU和LRU算法替换结果是不是相同？

	LFU（最不经常使用）					LRU（近期最少使用）				
	说明	1块	2块	3块	4块	说明	1块	2块	3块	4块
a	a进入	1	0	0	0	a进入	0	1	1	1
b	b进入	1	1	0	0	b进入	1	0	2	2
c	c进入	1	1	1	0	c进入	2	1	0	3
d	d进入	1	1	1	1	d进入	3	2	1	0
b	命中	1	2	1	1	命中	4	0	2	1
b	命中	1	3	1	1	命中	5	0	3	2
c	命中	1	3	2	1	命中	6	1	0	3
c	命中	1	3	3	1	命中	7	2	0	4
d	命中	1	3	3	2	命中	8	3	1	0
d	命中	1	3	3	3	命中	9	4	2	0
a	命中	2	3	3	3	命中	0	5	3	1
e	替换a	1	0	0	0	替换b	1	0	4	2

Cache写策略

- 写回—按写分配：
  - ✓命中，只写cache；
  - ✓失效，调块，只写cache；
- 写回—不按写分配
  - ✓命中，只写cache；
  - ✓失效，只写主存；
- 写直达—按写分配：
  - ✓命中，写cache写主存；
  - ✓失效，调块，写cache写主存；
- 写直达—不按写分配：
  - ✓命中，写cache写主存；
  - ✓失效，只写主存；

虚存TLB的地址映射



根据逻辑页号同时查找快表和慢表

- 按地址访问：将逻辑页号作为慢表表项的索引
- 按内容比较：将逻辑页号作为匹配关键字在快表表项内容中进行查找

根据程序局部性原理，多数虚存访问都将通过TLB完成，从而能够有效降低访存的时间延迟

Ch7-总线主要内容

1. 总线概述
   - 1.1 总线的基本概念
   - 1.2 总线的分类
   - 1.3 总线的特性和性能指标
   - 1.4 总线标准

2. 总线结构
   - 2.1 单总线结构
   - 2.2 多总线结构
   - 2.3 总线内部结构
   - 2.4 总线结构实例

3. 总线仲裁
   - 3.1 集中式仲裁
   - 3.2 分布式仲裁

4. 总线通信
   - 4.1 总线接口
   - 4.2 总线操作与总线周期
   - 4.3 串行/并行传送
   - 4.4 总线通信方式

     同步、异步、半同步、分离式

要求：

1、掌握总线仲裁和总线通信的基本方式。

2、了解总线控制器的硬件实现。

Ch8-IO系统主要内容

1. I/O系统概述
   - 1.1 I/O系统的发展概况
   - 1.2 I/O系统的组成
   - 1.3 I/O系统与主机的联系

2. I/O接口
   - 2.1 I/O接口概述
   - 2.2 I/O接口的功能与组成
   - 2.3 I/O接口类型

3. I/O信息交换方式
   - 3.1 程序查询方式
   - 3.2 程序中断方式
   - 3.3 DMA方式
   - 3.4 通道方式

4. I/O设备
   - 4.1 IO设备概述
   - 4.2 常见的输入和输出设备

5. 辅助存储器
   - 5.1 辅存概述
   - 5.2 磁记录原理与记录方式
   - 5.3 磁盘存储器
   - 光盘存储器
   - FLASH存储器

要求：

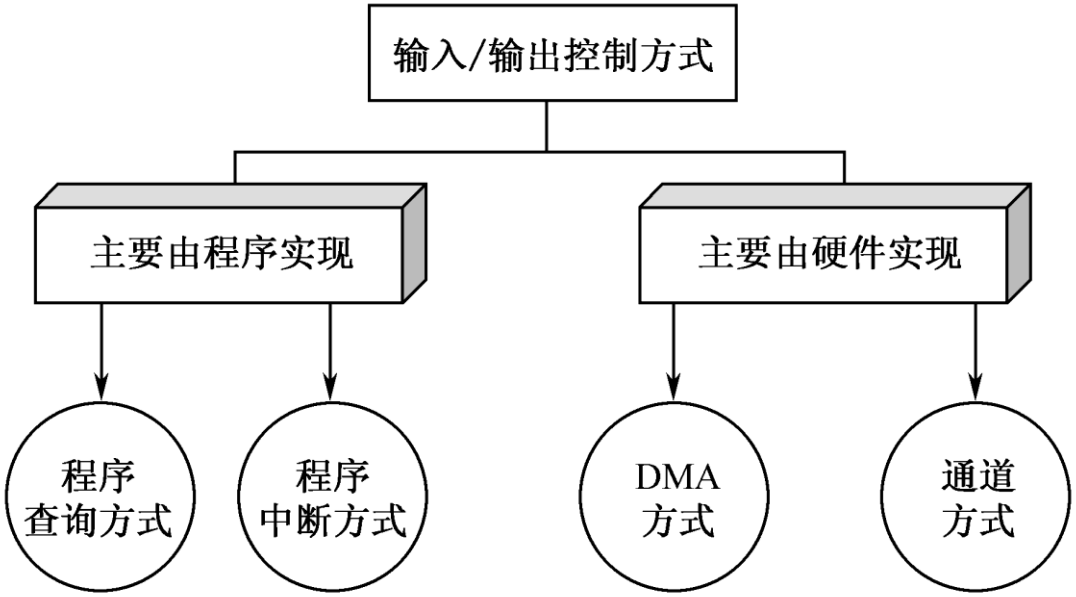
1、掌握IO信息交换方式。

2、了解IO通道与DMA控制器的硬件实现。

IO信息交换方式

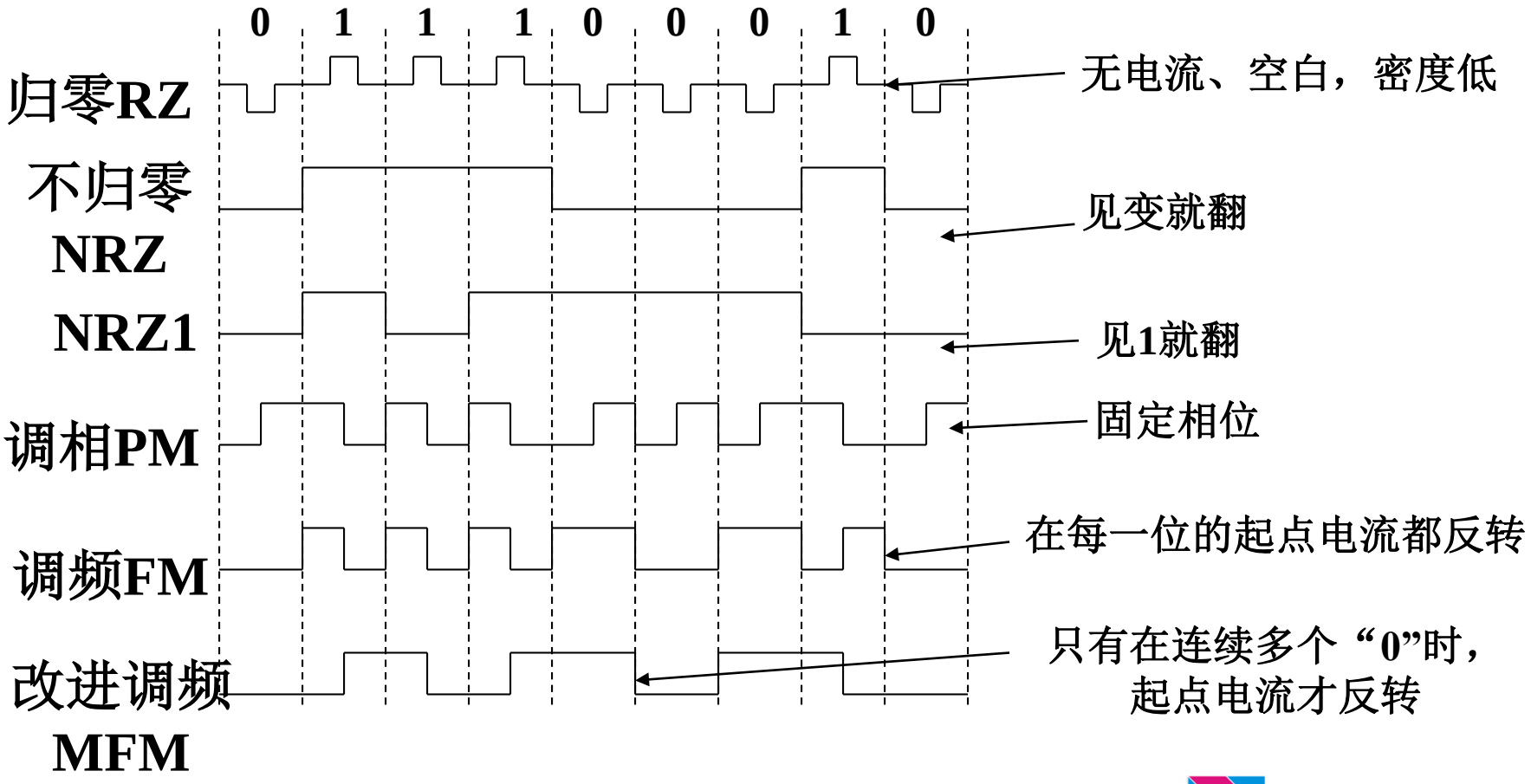
信息交换方式

- ✓程序查询方式、程序中断方式、DMA方式、通道方式



硬盘磁记录方式—编码方式

□写电流波形的形式



总结1：计算机系统设计方法

- 计算机软硬件系统的运行
  - ✓ 手写软件代码如何在计算机运行？
  - ✓ 开机如何运行（结合OS）？
  - ✓ 手机等嵌入式设备完成一个典型的人机交互操作？
- 从0开始设计一个计算机系统
  - ✓ 如何设计一个CPU（指令集、单周期、多周期、流水线、数据通路、控制信号、**状态机**）
  - ✓ 如何设计一个存储（Cache）控制器（存储层次化、Cache模块、映射、替换、状态机）
  - ✓ 如何设计一个总线控制器（总线协议、状态机）
  - ✓ 如何设计一个IO控制器（DMA控制器、通道处理器、状态机）

总结2：计算机系统的设计工具

- 软件
  - ✓代码编写（Java/C++/脚本语言）
  - ✓代码的编译工具链（预处理、编译、汇编、链接）
  - ✓代码调试（执行过程中寄存器、内存的状态Gdb）
- 硬件
  - ✓硬件描述语言工具　Verilog/ VHDL/ C->RTL
  - ✓硬件综合、仿真
    - Mentor->ModelSim //Synopsys //Cadence //Vivado ISE
  - ✓FPGA开发板调试　Xilinx/Intel(Altera)
  - ✓芯片流片　TSMC SMIC GF

总结3：计算机系统的评价指标

- **CPU时间**
  - ✓CPI
  - ✓IC
  - ✓主频
- 加速比（Amdahl定律）
  - ✓部件加速比
  - ✓可改进比例
- 流水线
  - ✓吞吐率
  - ✓加速比
  - ✓效率

指令字长
机器字长
存储字长

存储器带宽
总线带宽

机器周期
时钟周期
指令周期

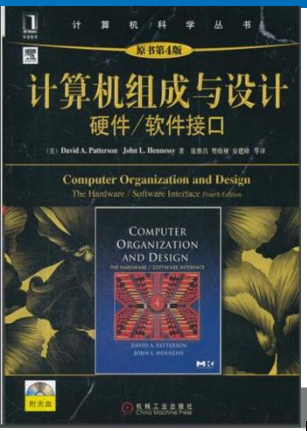
- Cache
  - ✓命中率
  - ✓失效开销
  - ✓访问时间
- 硬盘
  - ✓道密度
  - ✓位密度
  - ✓寻址时间
  - ✓传输率

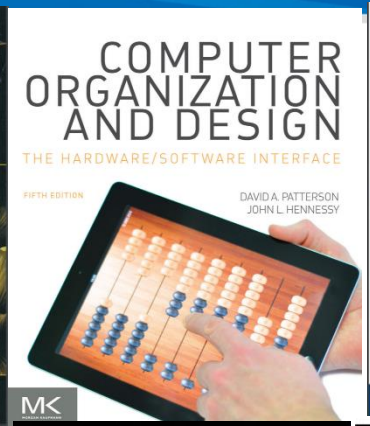
期末考试安排

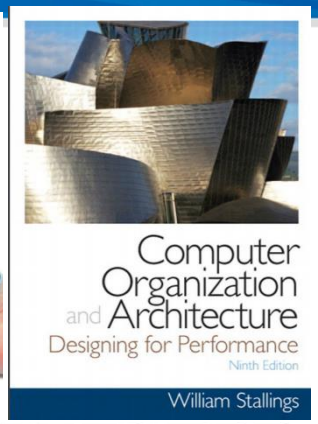
- 6月18日 8:30-10:30
- 闭卷考试，可以带计算器（不带存储功能）
- 不能用手机做计算器
- 简单题、计算题、综合题
- 最终成绩=考试+作业+实验

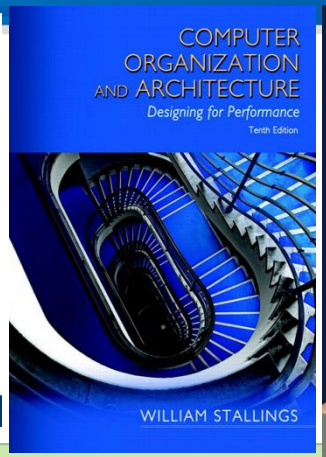
计算机组成原理-体系结构教材

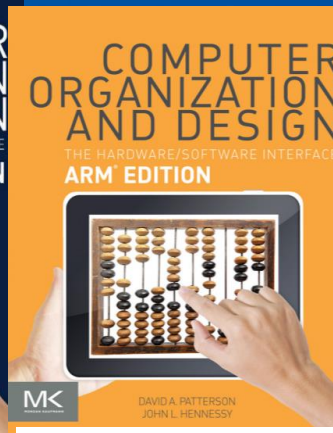
中国科学技术大学
University of Science and Technology of China

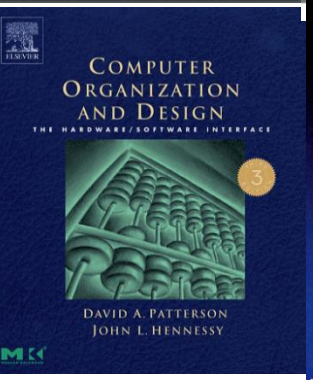


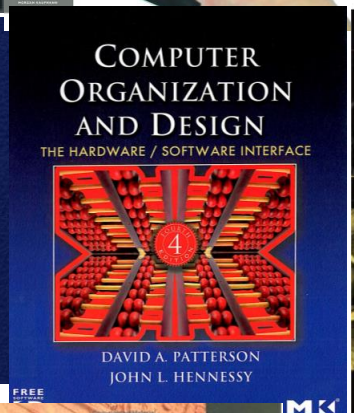


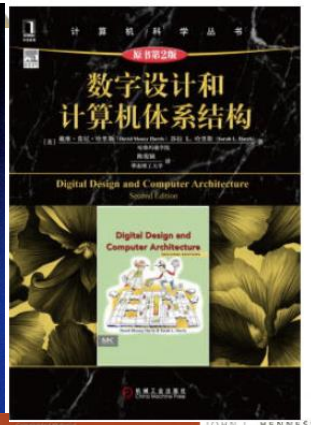


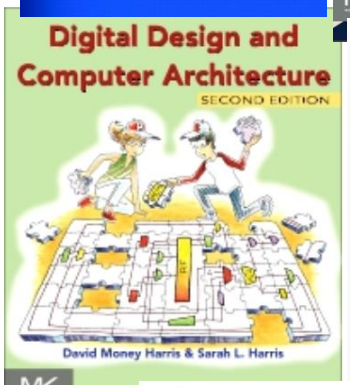


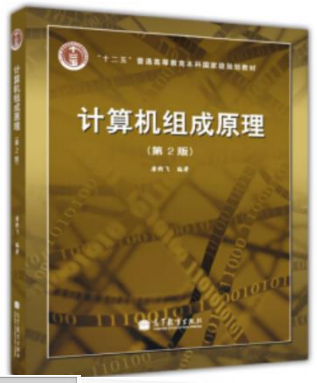


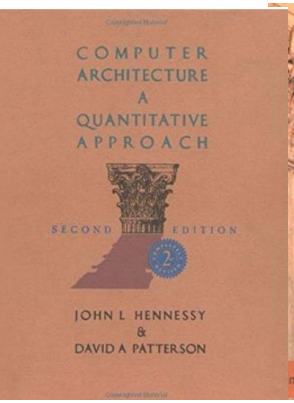


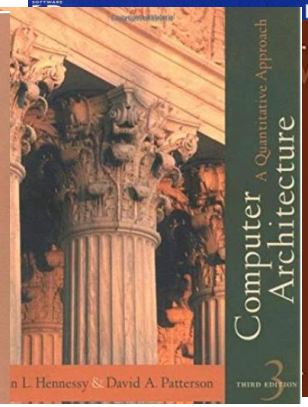


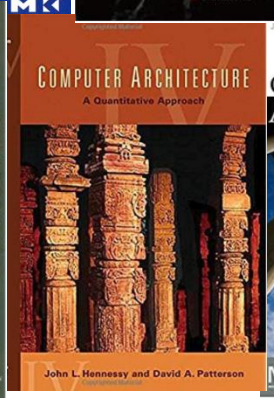


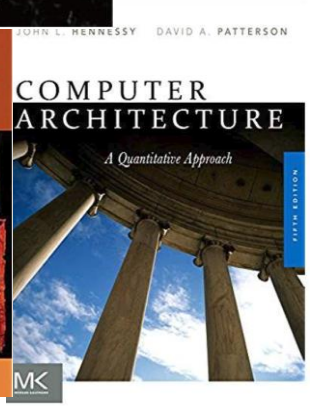


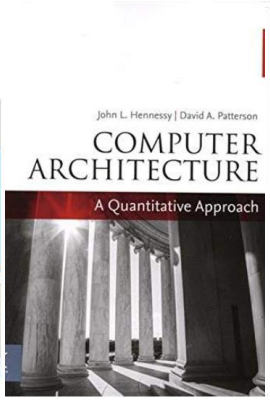












计算机系统基础参考书


深入理解计算机系统
Computer Systems
A Programmer's Perspective


计算机系统
核心概念及软硬件实现
Computer Systems


计算机系统
系统架构与操作系统的高度集成
Computer Systems

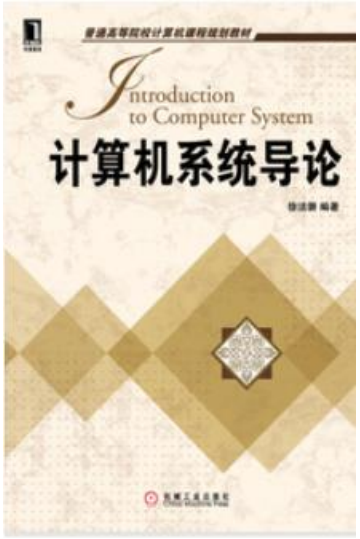

计算机系统概论

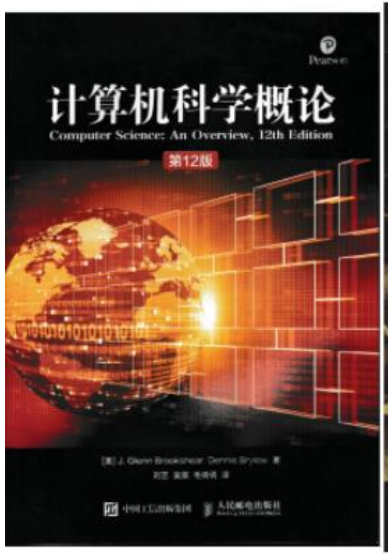

计算机系统结构
Computer Systems Architecture

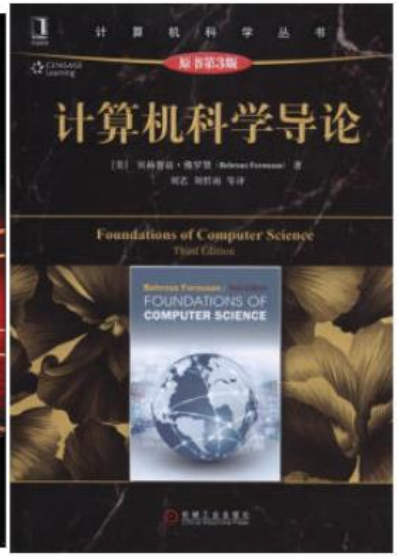

计算机系统原理


INTRODUCTION TO COMPUTER SYSTEMS
计算机系统基础 第2版


计算机系统
基础概念及编程实践

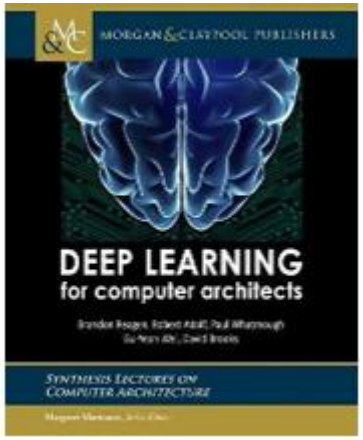

Introduction to Computer System
计算机系统导论

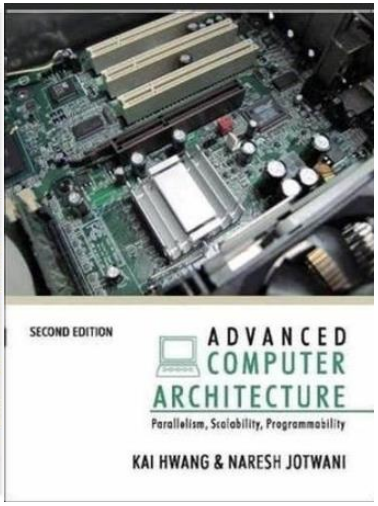

计算机科学概论
Computer Science: An Overview, 12th Edition
第12版

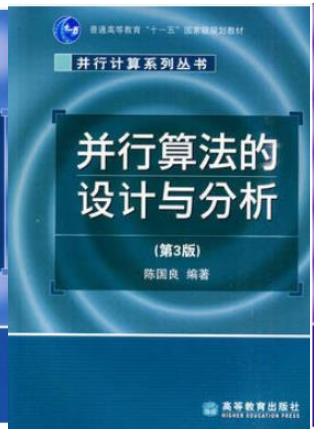

计算机科学导论
Foundations of Computer Science
FOUNDATIONS OF COMPUTER SCIENCE

前沿-面向研究生的参考书

中国科学技术大学
University of Science and Technology of China



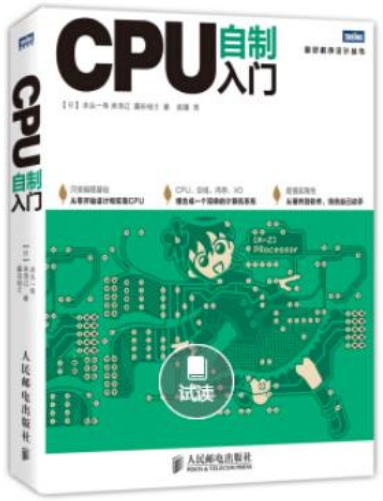




Synthesis Lectures on Computer Architecture，https://www.morganclaypool.com/toc/cac/1/1

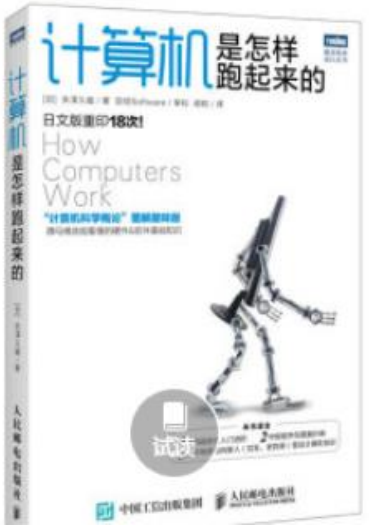
Computer Science 152/252: CS152 Computer Architecture and Engineering CS252 Graduate Computer Architecture
http://www-inst.eecs.berkeley.edu/~cs152/sp19/

课外书

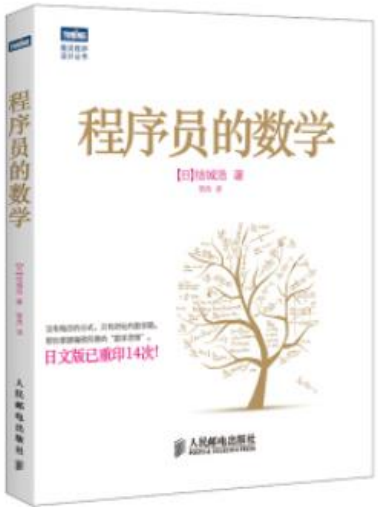

CPU自制入门
试读
人民邮电出版社


自制编译器
How to Develop a Compiler
从零开始制作真正的编译器
试读
中国工信出版集团
人民邮电出版社

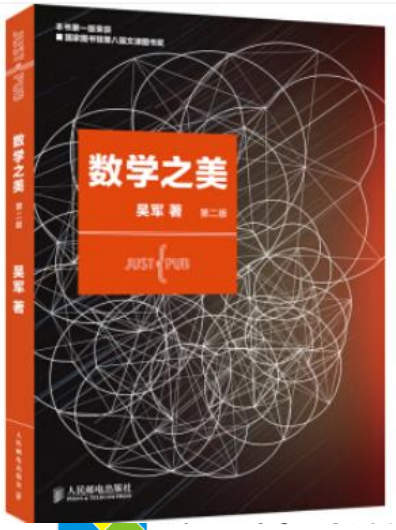

Broadview
Microsoft
永不褪色的计算机科学经典著作
C O D E
编码
隐匿在计算机软硬件背后的语言
Code:The Hidden Language of Computer Hardware and Software
中国工信出版集团


计算机是怎样跑起来的
日文版重印18次!
How Computers Work
试读
中国工信出版集团
人民邮电出版社


大话计算机
计算机系统


程序员的数学
日文版已重印14次!
人民邮电出版社


计算机科学丛书
原书第2版
计算机程序的构造和解释
Structure and Interpretation of Computer Programs
Second Edition


数学之美
吴军 著
人民邮电出版社

实验室招收本科生，从事面向人工智能/神经网络应用的加速器与智能计算机

联系方式：王超
中国科学技术大学计算机学院
cswang@ustc.edu.cn

Thank you