中国科学技术大学

University of Science and Technology of China

4.处理器设计-流水线

王 超

中国科学技术大学计算机学院

高能效智能计算实验室

2026年春

内容提要

- 流水线技术原理
- 流水线的分类
- CPU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

1.1产品生产流水线

(1) 一个问题

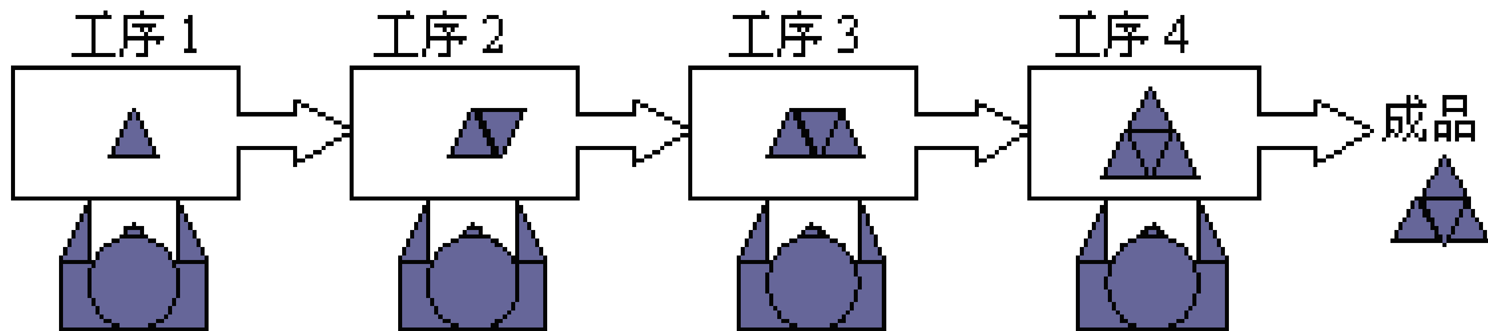
假设某产品的生产需要4道工序，该产品生产车间以前只有1个工人，1套生产该产品的机器。该工人工作8小时，可以生产120件（即每4分钟生产1件）。

要将该产品日产量提高到480件，如何能实现目标?

(2) 两种解决方案

方案一：增加3名工人、3套设备。

方案二：产品生产采用流水线方式，分为4道工序；增加3名工人，每人负责一道工序。



(3) 两种方案的工作过程对比

两种方案中，单件产品的生产时间均不变。

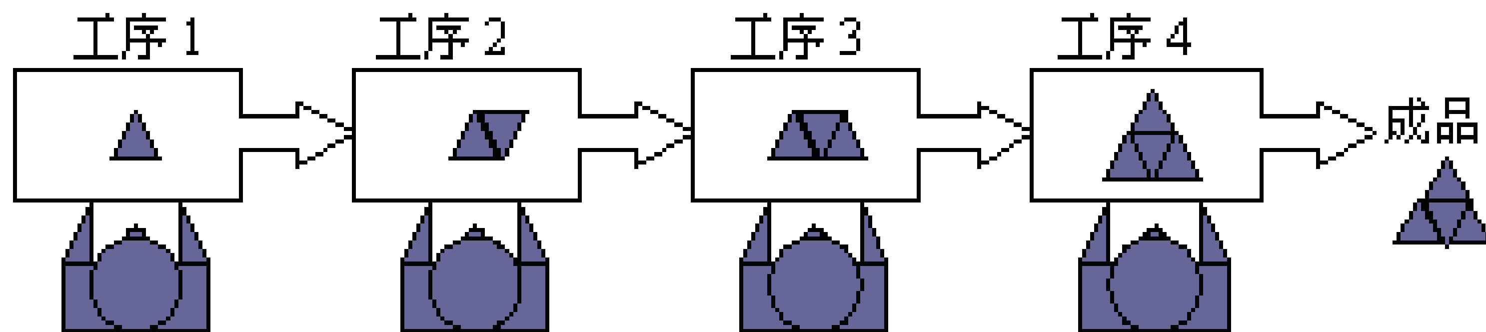
但在稳定情况下，

方案一：每4分钟，4件产品同时进入流水线，4件成品同时离开流水线，需要增加3套设备。

方案二：每分钟，1件产品进入流水线，1件成品离开流水线，不需要增加任何设备。

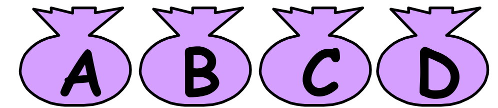
(4) 方案二的主要特点

每件产品还是要经过4道工序处理，单件产品的加工时间并没有改变，但它将不同产品的不同步骤重叠在一起，使得每件产品的产出时间从表面上看是从原来的4分钟缩减到1分钟，提高了产品的产出率。

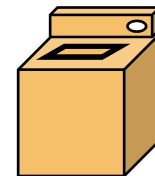


1.2 洗衣店流水线例子

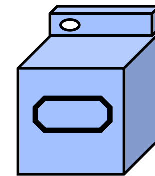
- A, B, C, D
  to wash, dry, and fold;



- Washer takes 30 minutes



- Dryer takes 30 minutes



- Folder takes 30 minutes

(1) 串行工作的洗衣店

6 PM 7 8 9 10 11 Midnight

30 30 30 30 30 30 30 30 30 30 30 30

任务顺序

A

B

C

D

- 洗衣店用 6小时完成了4个任务（洗衣店吞吐率0.67t/h）；
- 4个同学的等待时间均为1.5小时；
- Washer使用2小时（效率为0.33)；Dryer使用2小时（0.33)； Folder使用2小时（0.33)

(2) 流水工作的洗衣店X

6 PM 7 8 9 10 Midnight

30 30 30 30 30 30

A B C D

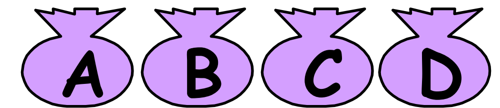
任务顺序

- 洗衣店用3小时完成了4个任务（1.33t/h）；
- 4个同学各等待了1.5小时;
- Washer使用2小时（0.66); Dryer使用2小时（0.66)； Folder使用2小时（0.66)；

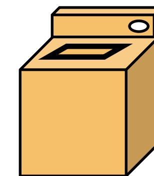
1.3 洗衣店流水线——洗衣店Y

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

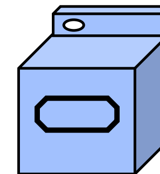
- A, B, C, D
  to wash, dry, and fold;



- Washer takes 30 minutes



- Dryer takes 40 minutes



- Folder takes 20 minutes

(1) 串行工作的洗衣店Y

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

6 PM 7 8 9 10 11 Midnight

30 40 20 30 40 20 30 40 20 30 40 20

任务顺序

A

B

C

D

- 洗衣店用 6小时完成了4个任务（0.67t/h）；
- 4个同学各等待了1.5小时；
- Washer使用2小时（0.33)；Dryer使用2小时40分（0.44)； Folder使用1小时20分（0.22)；

(2) 流水工作的洗衣店Y

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

6 PM 7 8 9 10 11 Midnight

30 40 40 40 40 20

任务顺序

A B C D

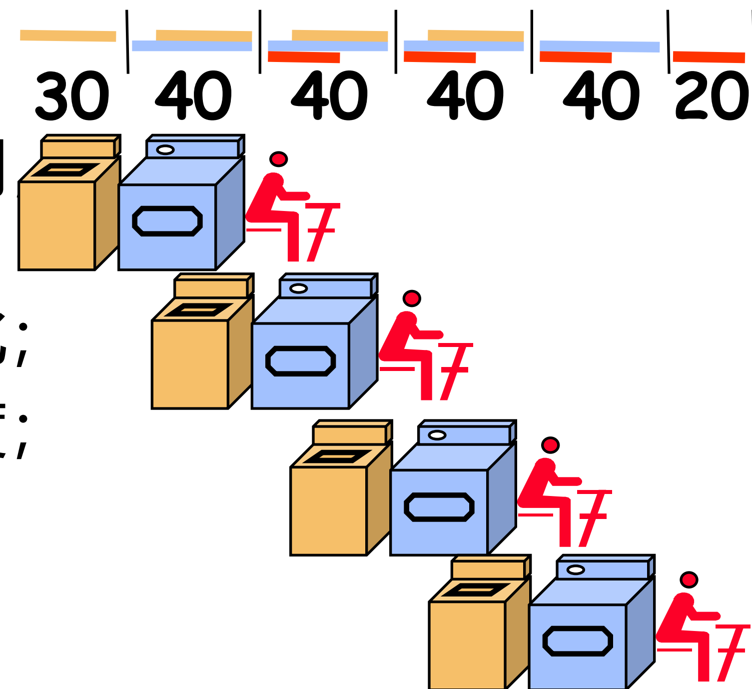
- 洗衣店用3.5小时完成了4个任务（1.14t/h）；
- 4个同学各等待了1.5小时;
- Washer使用2小时（0.57)；Dryer使用2小时40分（0.76)； Folder使用1小时20分（0.38)；

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

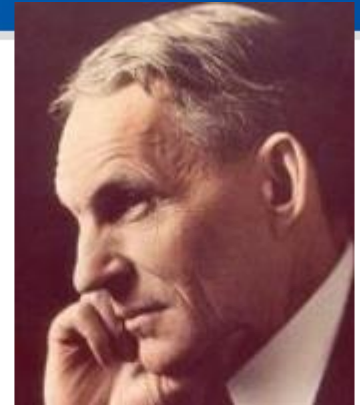
(3) 洗衣店的结论

	吞吐率	等待时间	效率（设备利用率）
串行均匀	0.67 t/h	1.5 h	0.33 0.33 0.33
流水均匀	1.33 t/h	1.5 h	0.66 0.66 0.66
串行非均匀	0.67 t/h	1.5 h	0.33 0.44 0.22
流水非均匀	1.14 t/h	1.5 h	0.57 0.76 0.38

- 流水线中多个任务是并行处理的；
- 流水线不能缩短单个任务的响应时间，但可以提高吞吐率；
- 吞吐率和效率（设备利用率）成正比；
- 流水线速度限制于最慢流水站的速度；
- 最大**加速比** = 流水站数
  - ✓ 流水站速度不匹配
  - ✓ 流水线“填充”和“排空”时间



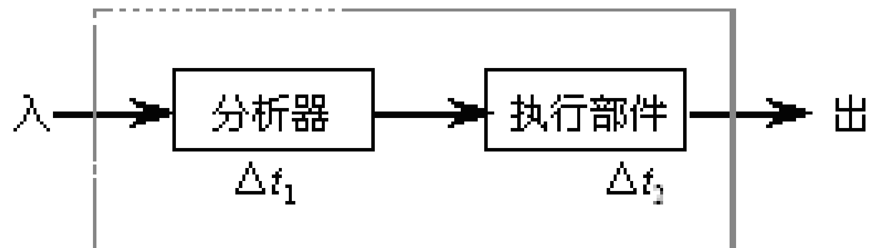
20世纪最大的发明之一



- 流水线
  - ✓ 流水线之前,汽车工业完全是手工作坊型的．每装配一辆汽车要728个人工小时,当时汽车的年产量大约12辆.这一速度远不能满足巨大的消费市场的需求.所以使得汽车成为富人的象征。
  - ✓ 1913年,福特提出在汽车组装中的流水线，汽车底盘在传送带上前行．前行中，逐步装上发动机，操控系统，车厢，方向盘，仪表，车灯，车窗玻璃、车轮，一辆完整的车组装成了。第一条流水线使每辆T型汽车的组装时间由原来的12小时28分钟缩短至10秒钟，生产效率提高了4488倍！

- 流水线是科学管理的一种体现
  - ✓ 科学管理之父 弗雷德里克·温斯洛·泰勒
  - ✓ 搬运生铁块试验、铁锹试验、金属切削实验

1. 3.计算机中的流水线



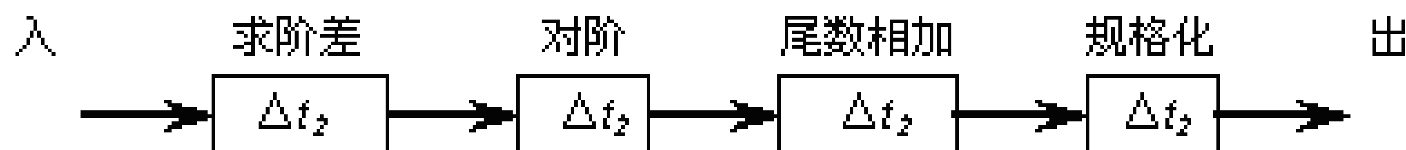
- 指令流水线
- 功能部件流水线 (浮点)
- Godson3~9　　Pentium4 ~31　　　ARM7~3
  MIPS/ARM9/PowerPC 405 ~ 5　ARM 10 6级　　ARM 11 8级
  ARM Coretex A8 13级　A9 8级

1.4.流水技术

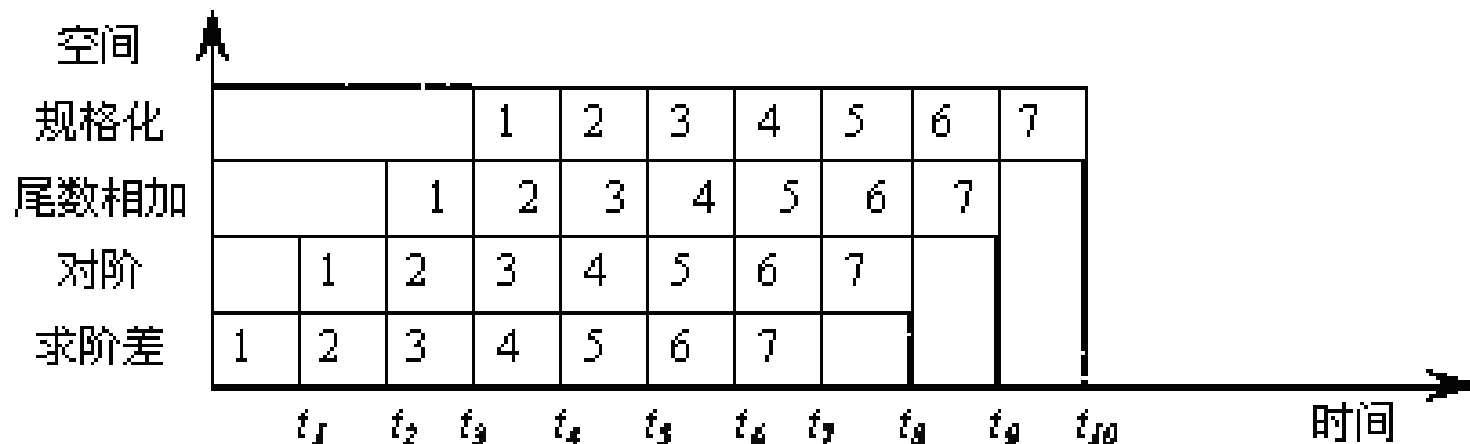
将重复的时序过程分解为若干子过程，每个子过程都可有效地在其专用功能段上与其它子过程同时执行，这种技术称为流水技术。

1.5.时-空图

从时间和空间两个方面描述流水线的工作过程，横坐标表示时间，纵坐标表示各流水段。



(a) 浮点加法流水线



(b) 描述流水线工作的时空图

1.6.流水线的特点

- 流水过程由多个相关的子过程组成，这些子过程称为流水线的“级”或“段”。段的数目称为流水线的“深度”。
- 每个子过程由专用的功能段实现，各功能段的时间应基本相等，通常为1个时钟周期（1拍）。否则，消耗时间长的功能段将成为流水线的瓶颈，会造成流水线的阻塞或断流。
- 流水线需要经过一定的填充时间才能稳定。
- 流水技术适合于大量重复的时序过程。
- 流水也是一种并行的手段。

2.流水线的分类

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

TI ASC的多功能流水线

- 1.单功能流水线和多功能流水线
  - ✓ 按流水线所完成的功能分类
  - ✓ **单功能流水线**，是指只能完成一种固定功能的流水线。
  - ✓ **多功能流水线**，是指各段可以进行不同的连接，从而完成不同的功能。

段	(a) 分段	(b) 浮点连接	(c) 定乘连接
1	输 入	输 入	输 入
2	减 阶	减 阶	
3	对阶移位	对阶移位	
4	相 加	相 加	
5	规 格 化	规 格 化	
6	相 乘		相 乘
7	累 加		累 加
8	输 出	输 出	输 出

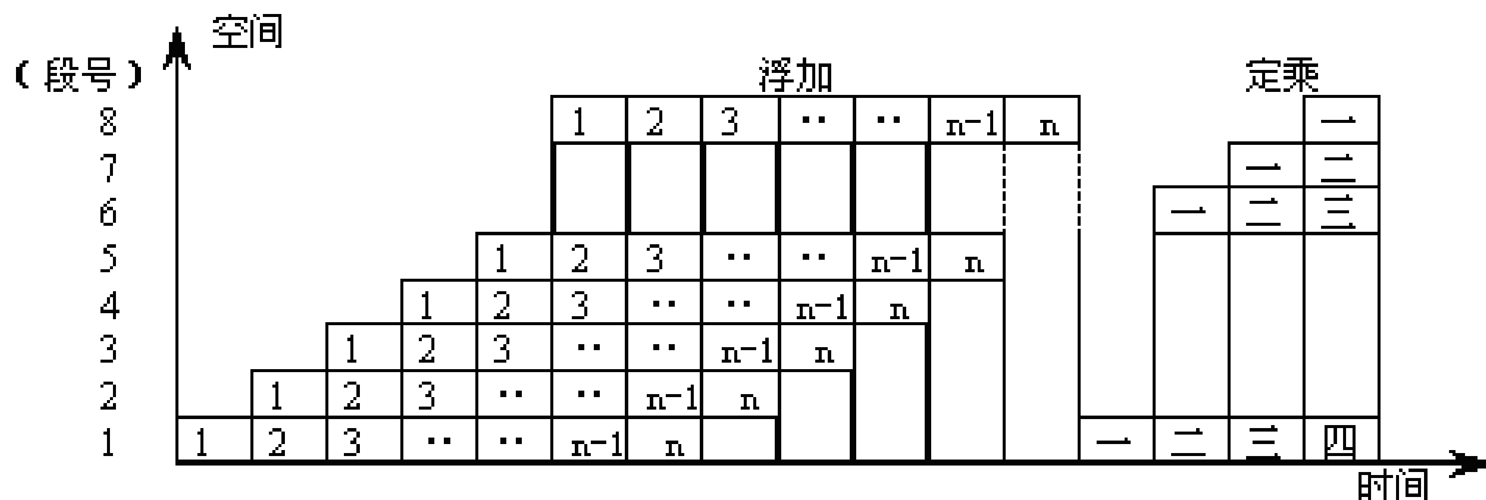
（a） 分段 （b） 浮点连接 （c） 定乘连接

2.流水线的分类

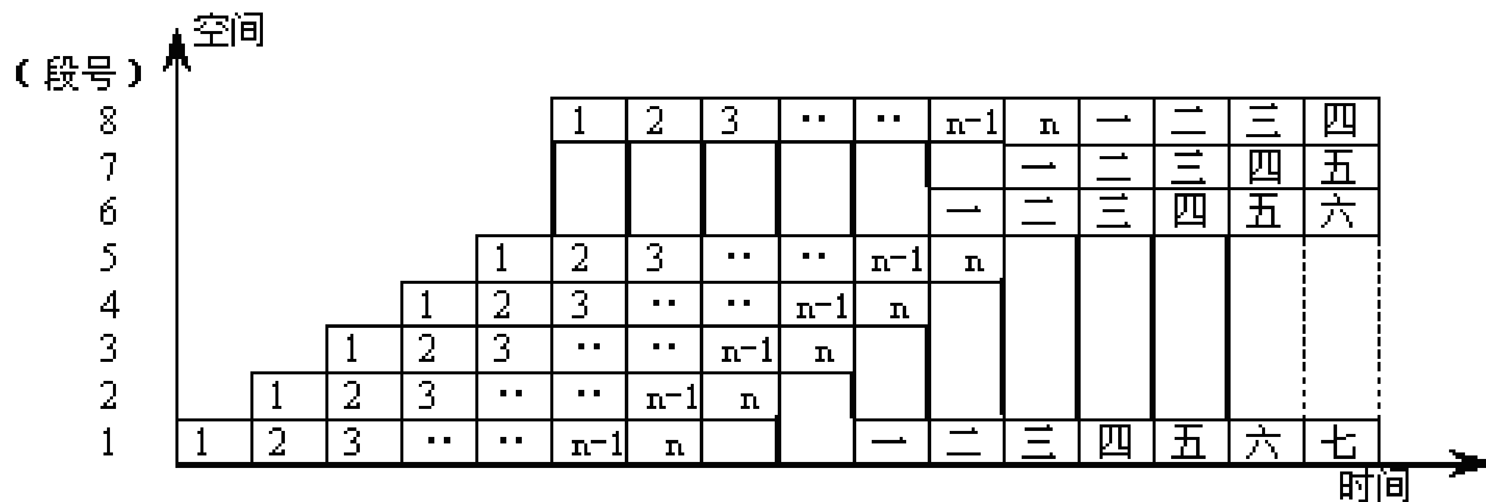
2.静态流水线和动态流水线

- ✓ 按同一时间内流水段的连接方式划分
- ✓ 静态流水线，是指在同一时间内，流水线的各段只能按同一种功能的连接方式工作。
- ✓ 动态流水线，是指在同一时间内，当某些段正在实现某种运算时，另一些段却在实现另一种运算。

动、静态流水线时空图



（a） 静态流水线

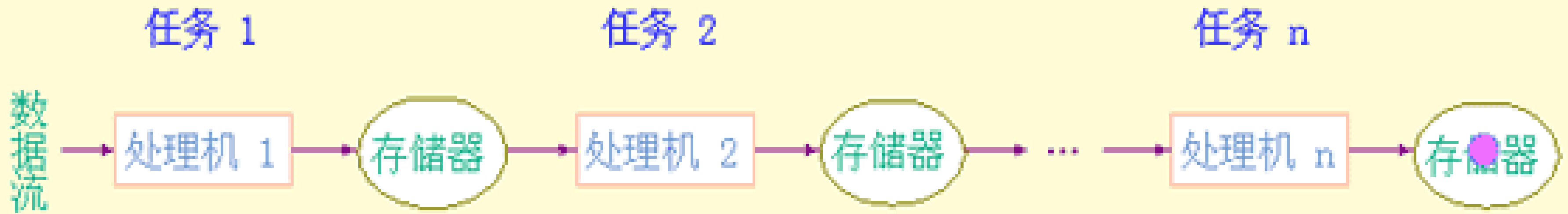


（b） 动态流水线

3.部件级、处理机级及处理机间流水线

- 按流水的级别划分
- 部件级流水线，又叫运算操作流水线，是把处理机的算术逻辑部件分段，使得各种数据类型的操作能够进行流水。
- 处理机级流水线，又叫指令流水线，是把解释指令的过程按照流水方式处理。
- 处理机间流水线，又叫宏流水线，是由两个以上的处理机串行地对同一数据流进行处理，每个处理机完成一项任务。

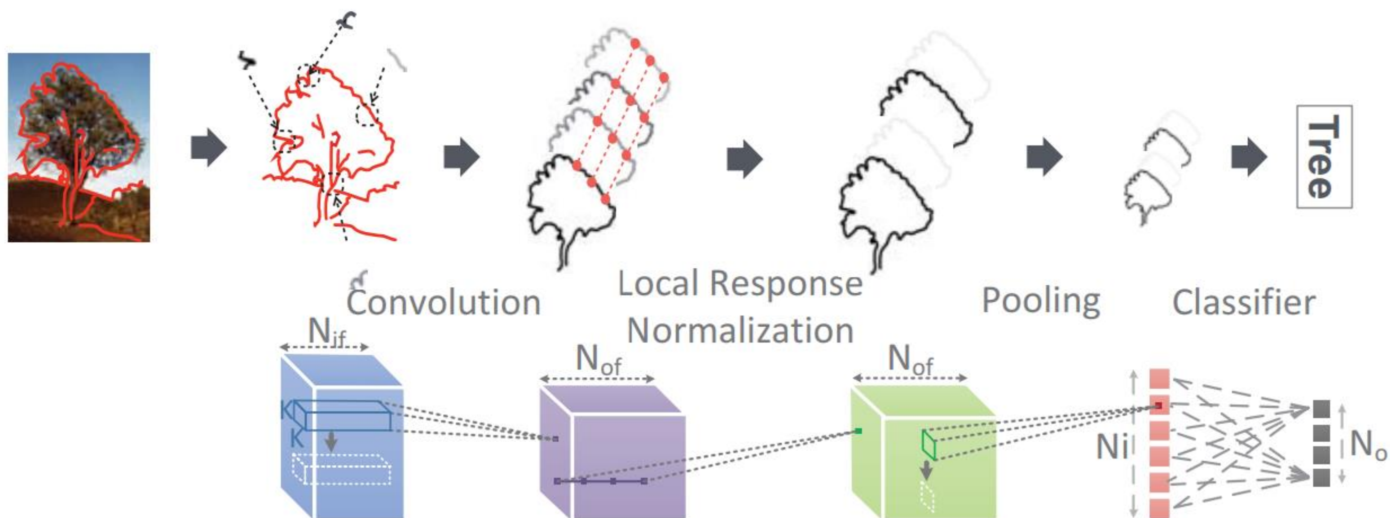
宏流水线-JPEG

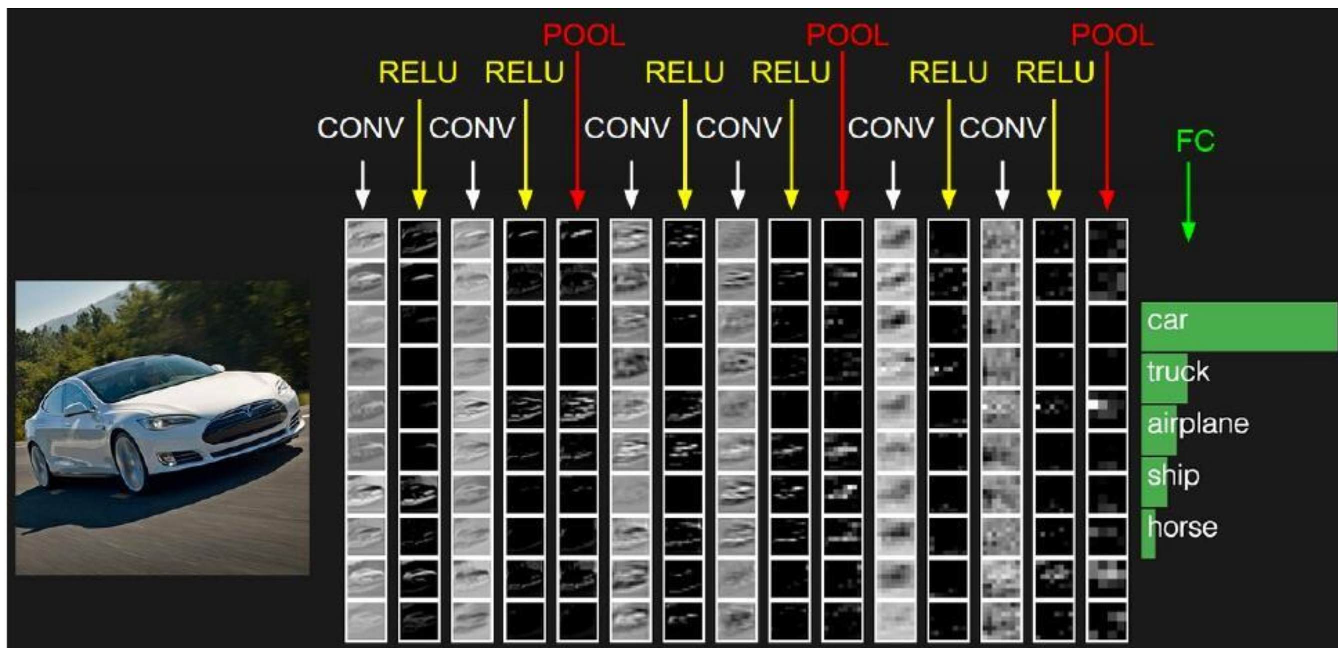


JPEG 编码的例子

f(i,j)
8x8
DCT
F(u,v)
Quantization
Fq(u,v)
Quantization Table
Coding Tables
Header
Tables
Data
01001011101…
Entropy Coding
DPCM
RLC
DC
AC
zig zag scan

宏流水线-神经网络





4.标量流水处理机和向量流水处理机

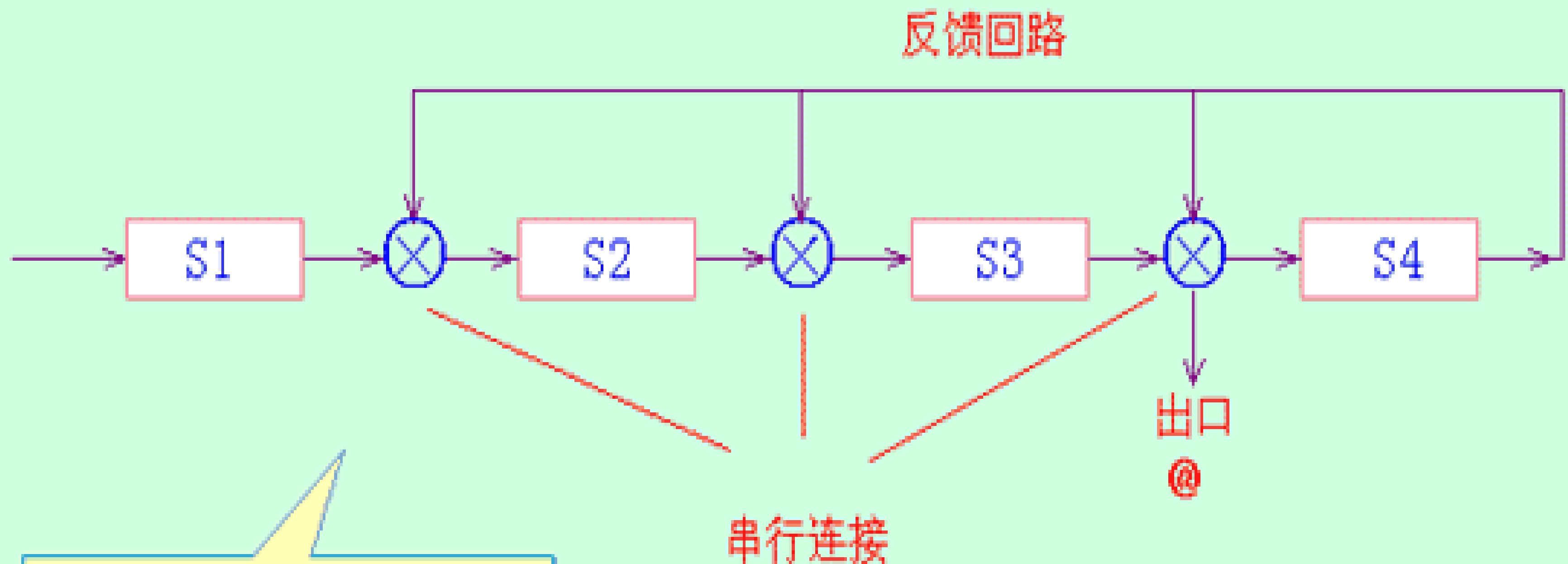
- 按照数据表示来进行分类
- 标量流水处理机，是指处理机不具有向量数据表示，仅对标量数据进行流水处理。
- 向量流水处理机，是指处理机具有向量数据表示

5.线性流水线和非线性流水线

- 按照是否有反馈回路来进行分类
- 线性流水线是指流水线的各段串行连接，没有反馈回路。
- 非线性流水线是指流水线中除有串行连接的通路外，还有反馈回路。
- 存在流水线调度问题。

非线性流水线

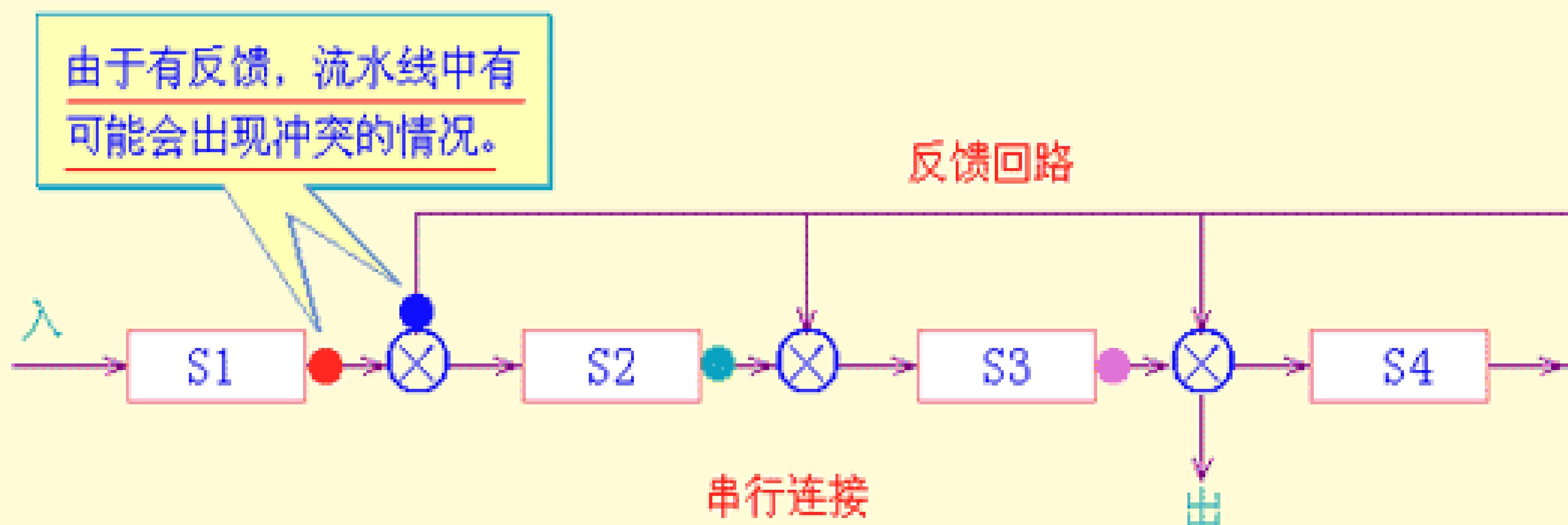
（举 例）



例如任务 @ ：

→ S1 → S2 → S3 → S4 → S2 → S3 → S4→ S3 →

流水线的调度问题



所以：

在非线性流水线中，一个重要的问题是确定什么时候向流水线引进新的输入，从而使新输入的数据和先前操作的反馈数据在流水线中不产生冲突。这就是所谓的流水线调度问题。

6.顺序流动流水线和乱序流动流水线

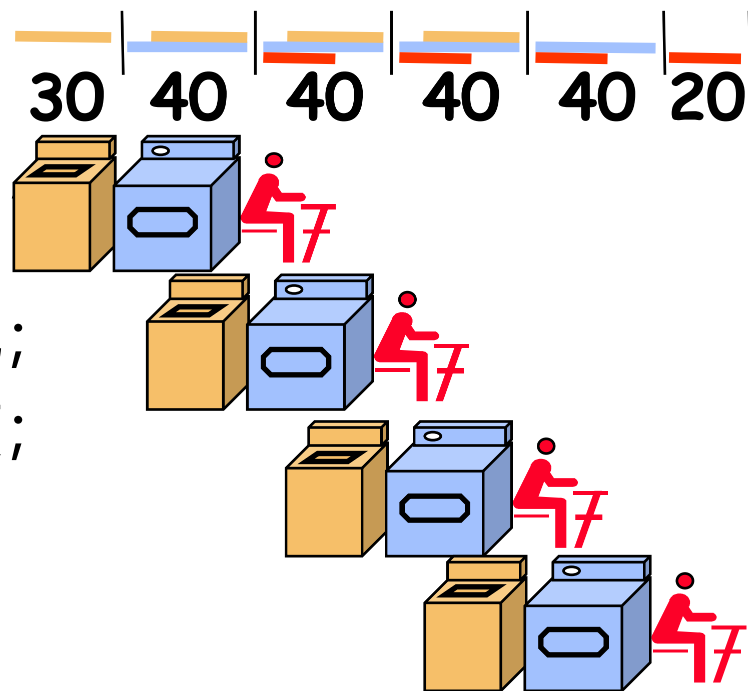
- 按照输出端任务流出顺序与输入端任务流入顺序是否相同划分

- 乱序流动流水线也可称为无序流水线、错序流水线

复习:流水线的若干结论

	吞吐率	等待时间	效率（设备利用率）
串行均匀	0.67 t/h	1.5 h	0.33 0.33 0.33
流水均匀	1.33 t/h	1.5 h	0.66 0.66 0.66
串行非均匀	0.67 t/h	1.5 h	0.33 0.44 0.22
流水非均匀	1.14 t/h	1.5 h	0.57 0.76 0.38

- 流水线中多个任务是并行处理的；
- 流水线不能缩短单个任务的响应时间但可以提高吞吐率；
- 吞吐率和效率（设备利用率）成正比；
- 流水线速度限制于最慢流水站的速度；
- 最大**加速比** = 流水站数
  - ✓ 流水站速度不匹配
  - ✓ 流水线“填充”和“排空”时间



复习：流水线的分类

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

- 单功能流水线：只能完成一种功能的流水线，如浮点加法流水线。
- 多功能流水线：流水线的各段可以进行不同的连接，从而使流水线在不同的时间完成不同的功能。
- 静态流水线：在某一时间段内，流水线的各段只能按同一种功能的连接方式工作，即只有当输入是一串相同性质的操作时其性能才能得到发挥。
- 动态流水线：在某一段时间内，某些段正在实现某类操作（定点乘），其他段却在实现另一类操作（浮点加）。
- 线性流水线：流水线的各段串行连接，没有反馈回路。
- 非线性流水线：流水线中除了串行的通路，还有反馈回路。
- 顺序流水线：流水线的流出顺序与其流入顺序相同。
- 乱序流水线：流水线的流出顺序与其流入顺序不同。

内容提要

- 流水线技术原理
- 流水线的分类
- 流水线的性能分析
- CPU的五级流水线实现
- 流水线的“依赖”及其处理

3.流水线性能分析

三项性能指标：吞吐率、加速比和效率

1.吞吐率

是衡量流水线速度的重要指标

- 吞吐率是指单位时间内流水线所完成的任务数或输出结果的数量。
- 最大吞吐率：TP_{max}是指流水线在达到稳定状态后所得到的吞吐率。
- 实际吞吐率：设流水线由m段组成，完成n个任务的吞吐率称为实际吞吐率，记作TP。

最大吞吐率

（1）最大吞吐率

- 假设流水线各段的时间相等，均为Δt_0，则：

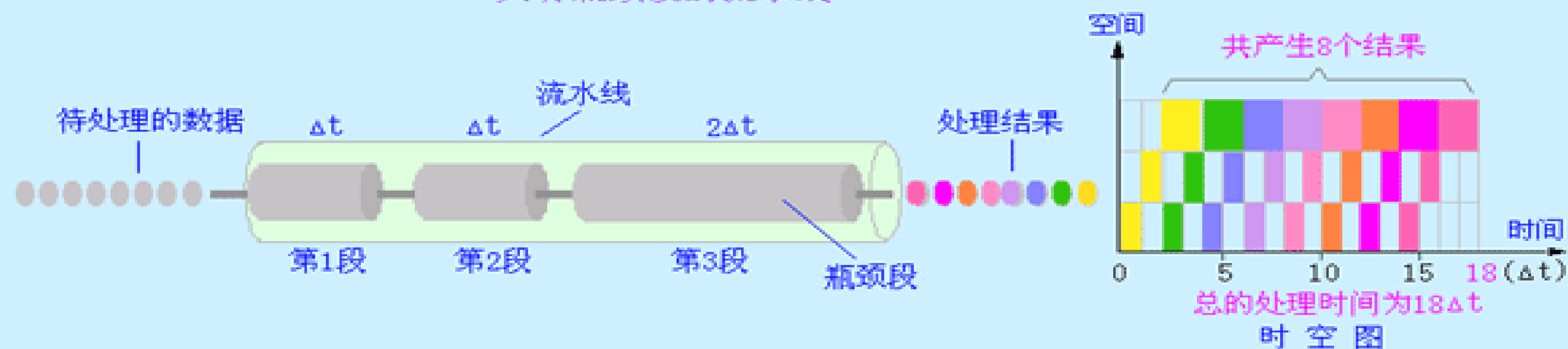
$$TP_{max} = 1/\Delta t_0$$

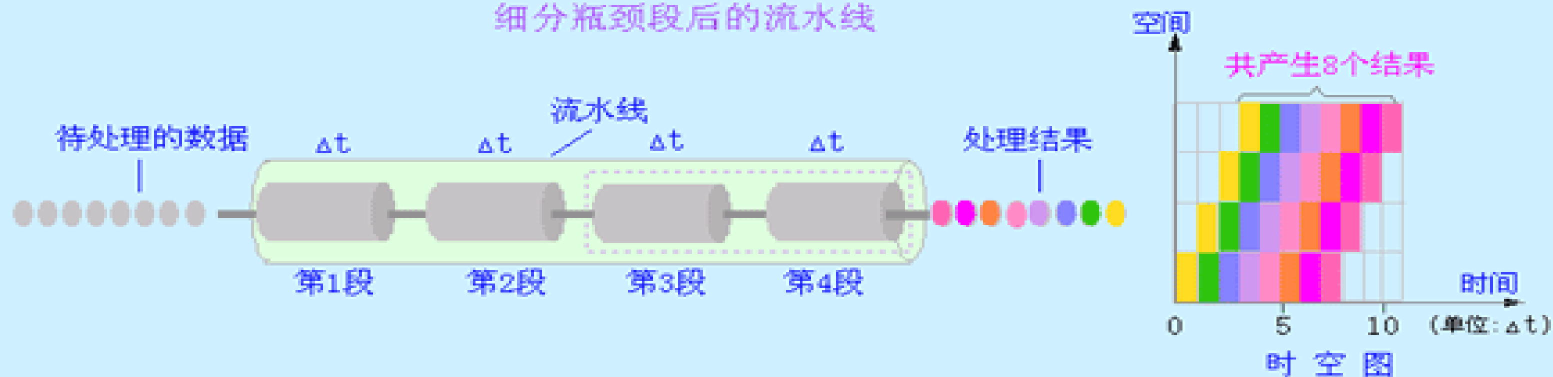
- 假设流水线各段时间不等，第i段时间为Δt_i，则：

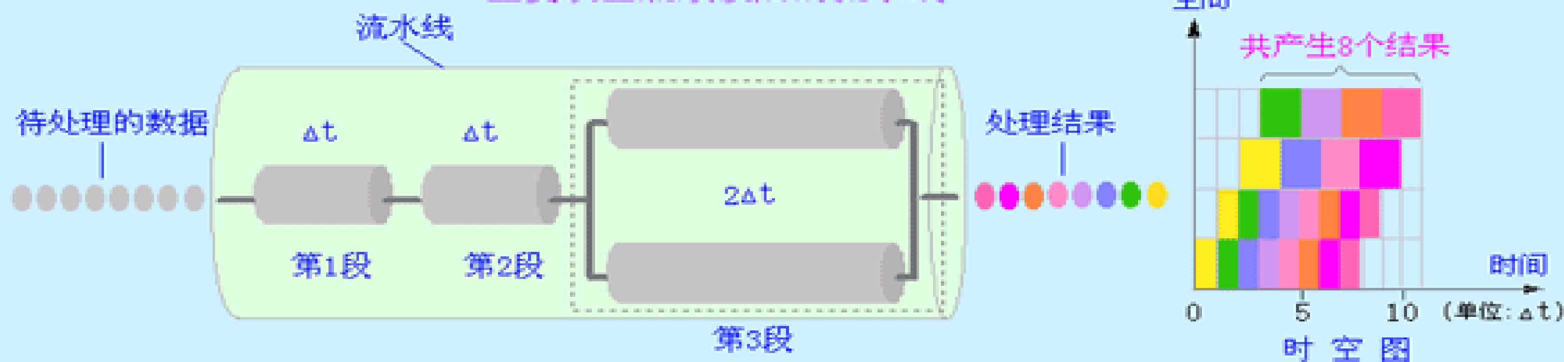
$$TP_{max} = 1/max\{\Delta t_i\}$$

  – 最大吞吐率取决于流水线中最慢一段所需的时间，该段成为流水线的瓶颈
  – 消除瓶颈的方法
    – 细分瓶颈段
    – 重复设置瓶颈段

流水线的改进







实际吞吐率-均匀流水线

（2）实际吞吐率

- 若各段时间相等（假设均为Δt_0），则完成时间

$T_{流水} = m\Delta t_0+(n-1)\Delta t_0$ (m=深度，n=任务数)

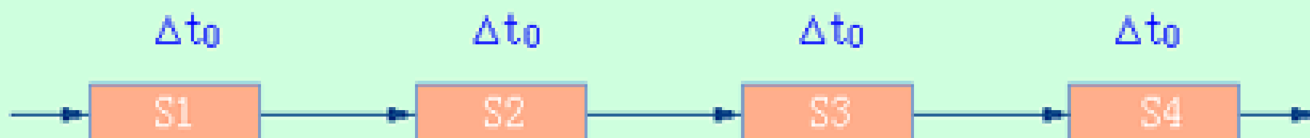
实际吞吐率

$$TP = \frac{n}{T_{流水}} = \frac{n}{m\Delta t_0 + (n-1)\Delta t_0}$$

$$= \frac{1}{\left(1+\frac{m-1}{n}\right)\Delta t_0} = \frac{TP_{max}}{1+\frac{m-1}{n}}$$

流水线的时-空图

（各段时间相等）



段

共完成 n 个任务

段													
S4				1	2	3	4	5	…	…	…	n-1	n
S3			1	2	3	4	5	…	…	…	n-1	n	
S2		1	2	3	4	5	…	…	…	n-1	n		
S1	1	2	3	4	5	…	…	…	n-1	n			

时间（单位：Δt）

$m\Delta t_0$ $\quad$ $(n-1)\Delta t_0$

$T_{流水} = m\Delta t_0 + (n-1)\Delta t_0$

时－空图

$$T_{流水} = m\Delta t_0 + (n-1)\Delta t_0$$

实际吞吐率-非均匀流水线

- 若各段时间不等（假设第i段为Δt_i），则完成时间

$$T=\sum_{i=1}^{m}\Delta t_i+(n-1)\Delta t_j$$

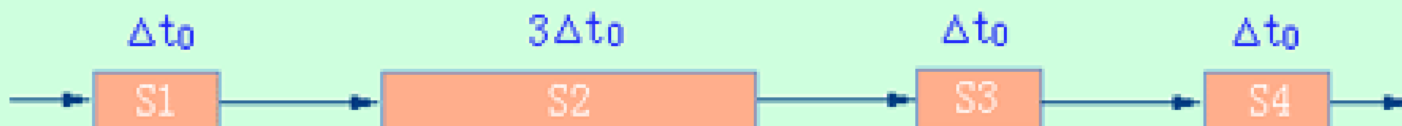
这里，$\Delta t_j=max\{\Delta t_i\}$

实际吞吐率 $$TP=\frac{n}{\sum_{i=1}^{m}\Delta t_i+(n-1)\Delta t_j}$$

非均匀流水线的时空图

流水线的时-空图

(各段时间不等)



段
S4
S3
S2
S1
时间
3Δt0

加速比

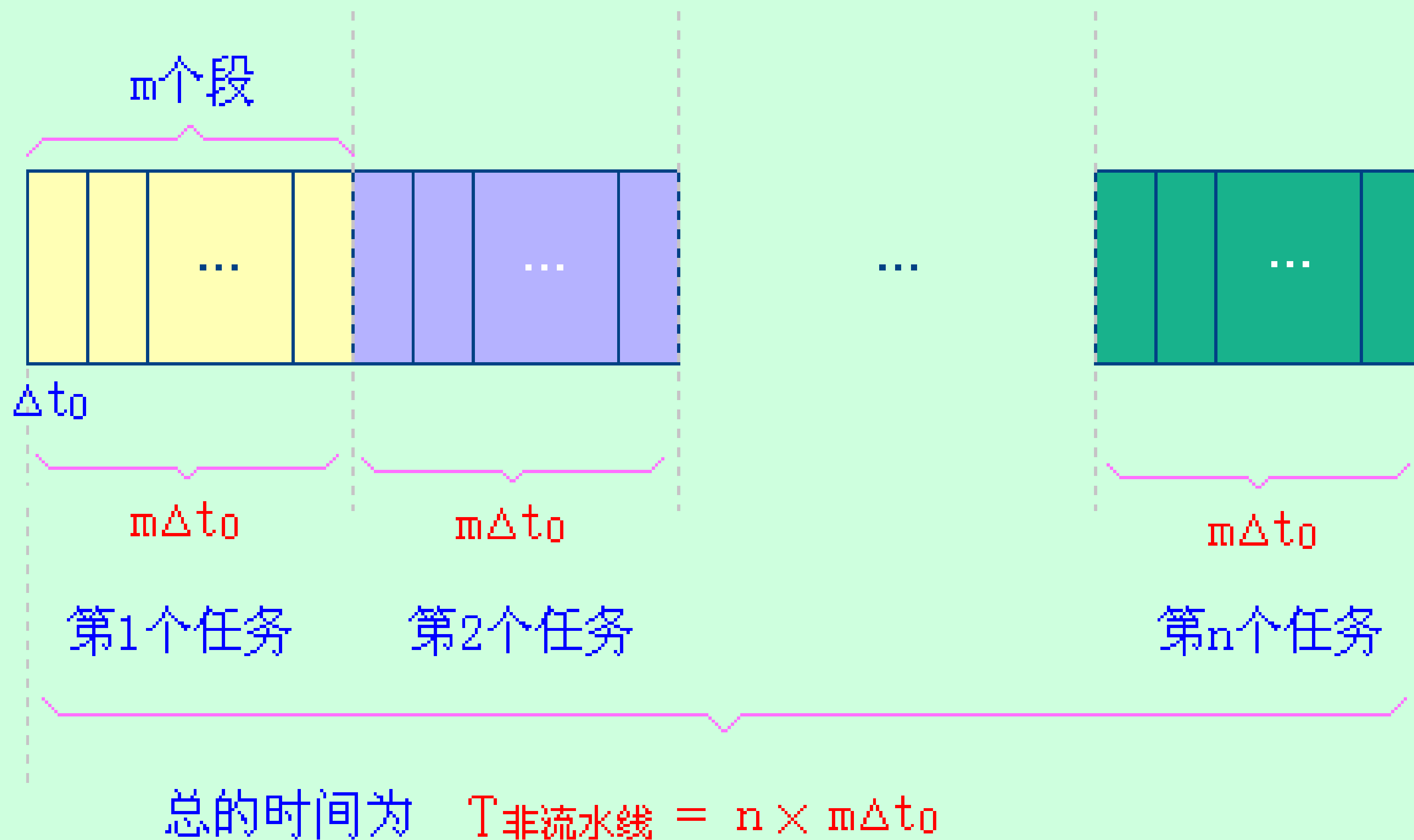
2. 加速比

- 加速比是指流水线速度与等功能的非流水线速度之比。
- 根据定义可知，加速比$S = T_{非流水} / T_{流水}$
- 若流水线为m段，每段时间均为$\triangle t_0$，则

$$T_{非流水} = nm\triangle t_0,\ T_{流水} = m\triangle t_0 + (n-1)\triangle t_0$$

$$S = \frac{mn}{m+n-1} = \frac{m}{1+\frac{m-1}{n}}$$

非流水方式所需的时间



3.效率

- 效率指流水线的设备利用率(部件运行时间与总时间的比率)。
- 由于流水线有通过时间（填充时间）和排空时间，所以流水线的各段并非一直满负荷工作，E<1
- 若各段时间相等，则各段效率也相等

  即e1 = e2 = e3 =… = nΔt_0/$T_{流水}$
- 整个流水线效率

$$E = \frac{n\Delta t_0}{T_{流水}} = \frac{n}{m+n-1} = \frac{1}{1+\frac{m-1}{n}}$$

当n>>m时，E≈1

通过时间和排空时间

通过时间和排空时间

段

段									
S4				1	…	…	…	…	n
S3			1	2	…	…	…	n	
S2		1	2	3	…	…	n		
S1	1	2	3	4	…	n			

时间

通过时间

排空时间

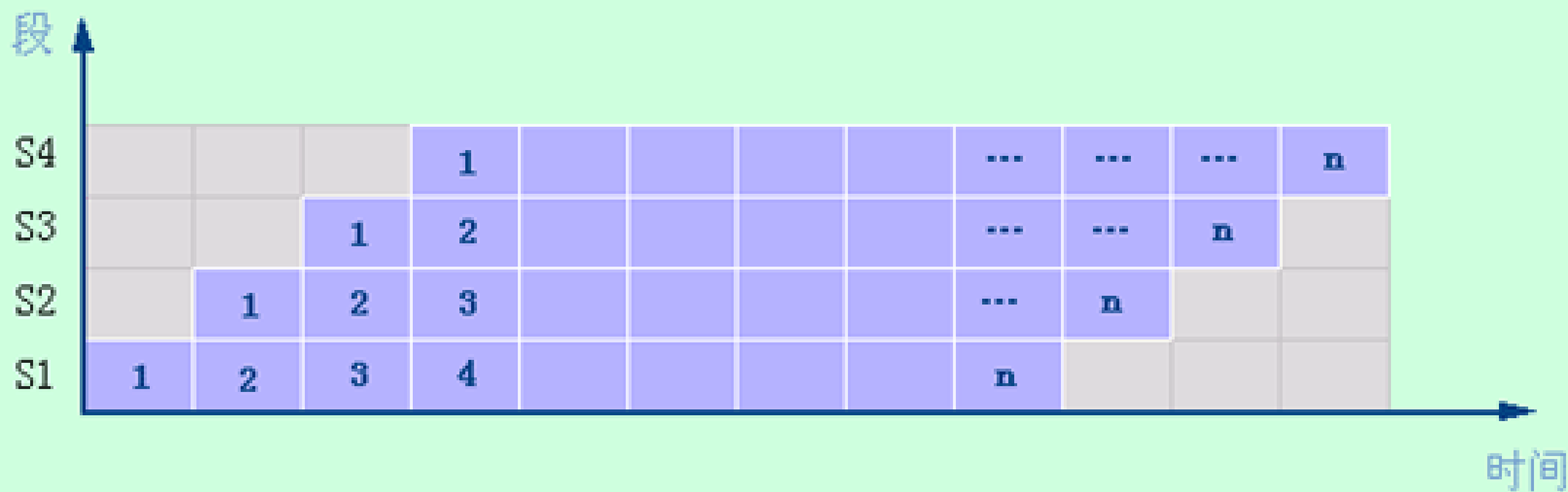
第1个任务进入

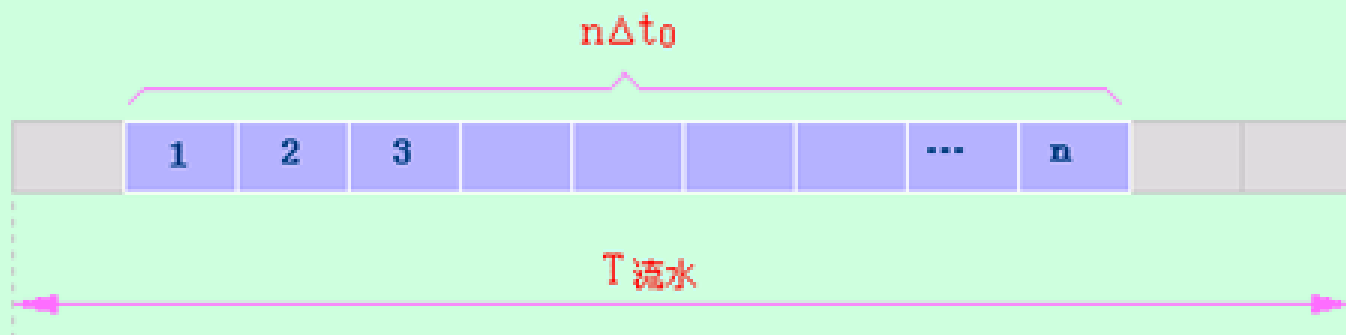
第1个任务完成

最后1个任务进入

最后1个任务完成

流水段的效率





所以 $e_2 = \dfrac{n\Delta t_0}{T_{流水}}$

效率在时空图上的表示

3.效率

- 从时-空图上看，效率就是n个任务所占的时空区与m个段总的时空区之比

  E=(Σei)/m=Σ（Tei/T流水）/m= ΣTei/（T流水×m）

- 根据这个定义，可以计算流水线各段时间不等时的流水线效率

$$E = \frac{n\text{ 个任务占用的时空区}}{m\text{ 个段总的时空区}}$$

性能参数之间的关系

4. 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。

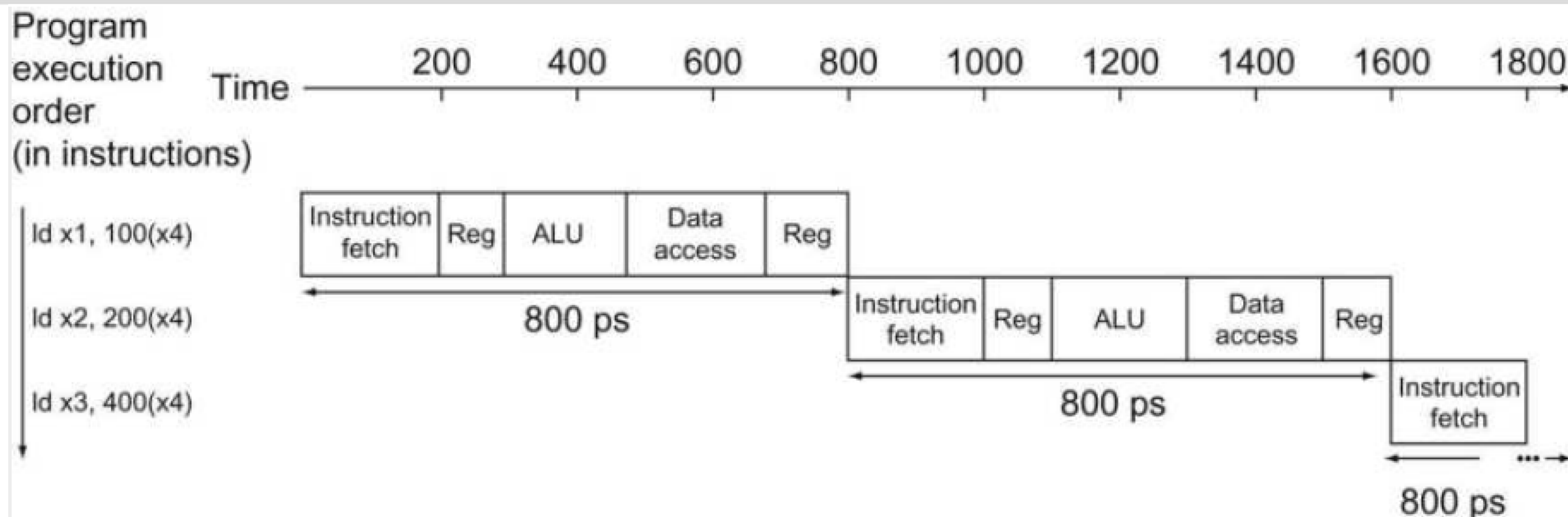
RISC-V的五级流水线实现

面向流水线实现的RISC-V指令集

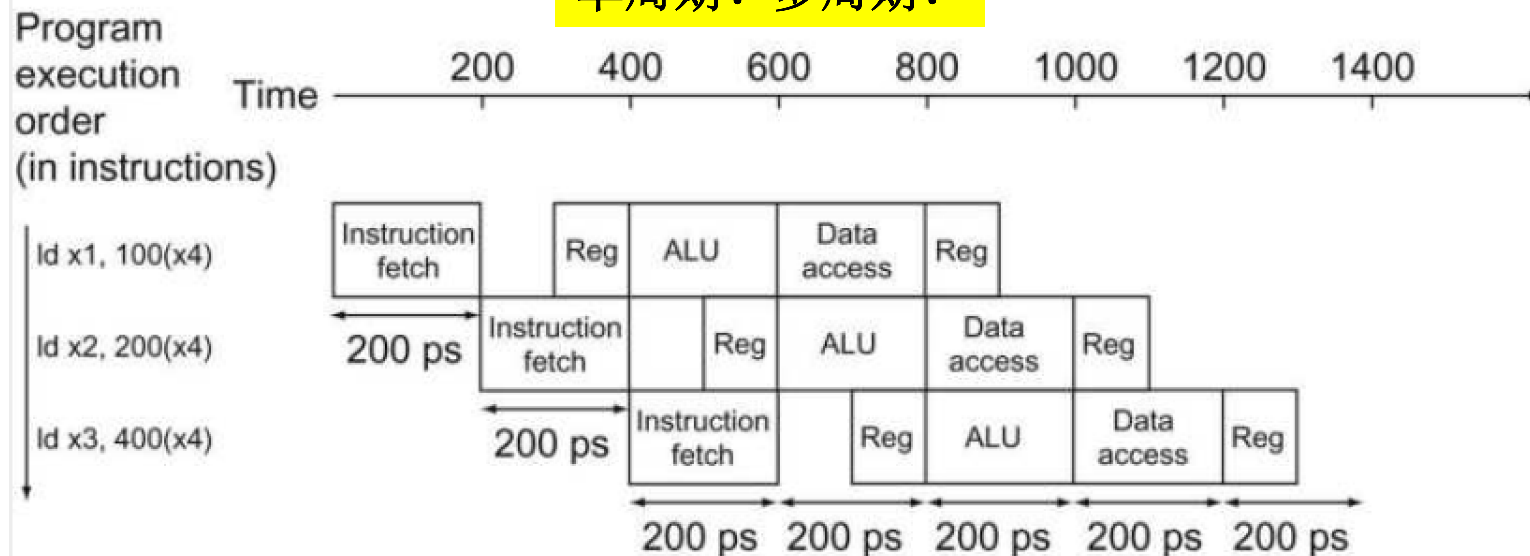
- 指令字等长
  - ✓ 适于取指和译码
- 指令格式简单
  - ✓ 源操作数位置固定，因此可以在译码的同时取操作数（读寄存器），否则要增加一级流水线段。
- 只有Load/Store访存
  - ✓ 意味着可以在执行阶段计算访存地址，然后在下一阶段访存。
  - ✓ 假设：如果R-type指令也可访存，则变为取指、译码、地址计算、访存、执行、写回等几个阶段
- 操作数在内存中要“字对齐”
  - ✓ 因此在取操作数时不需要访问存储器多次

RISC-V的5级流水线划分

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



单周期？多周期？



流水线的时空图

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

Time (in clock cycles)

CC 1 CC 2 CC 3 CC 4 CC 5 CC 6 CC 1 CC 2 CC 3

Program
execution
order
(in instructions)

lw $10, 20($1) — IM, Reg, ALU, DM, Reg

sub $11, $2, $3 — IM, Reg, ALU, DM, Reg

add $12, $3, $4 — IM, Reg, ALU, DM, Reg

lw $13, 24($1) — IM, Reg, ALU, DM, Reg

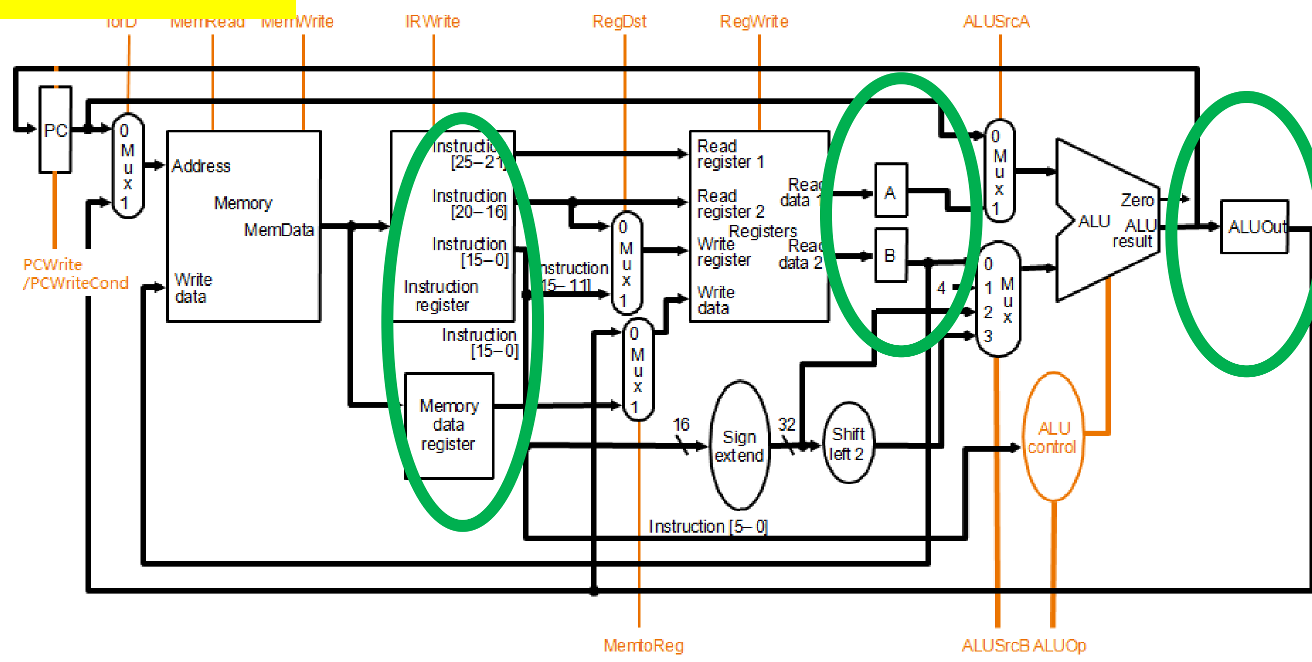
add $14, $5, $6 — IM, Reg, ALU, DM, Reg

设计流水线CPU的数据通路基于单周期 or 多周期?

寄存器?

ALU?

存储器?



指令数据通路划分

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

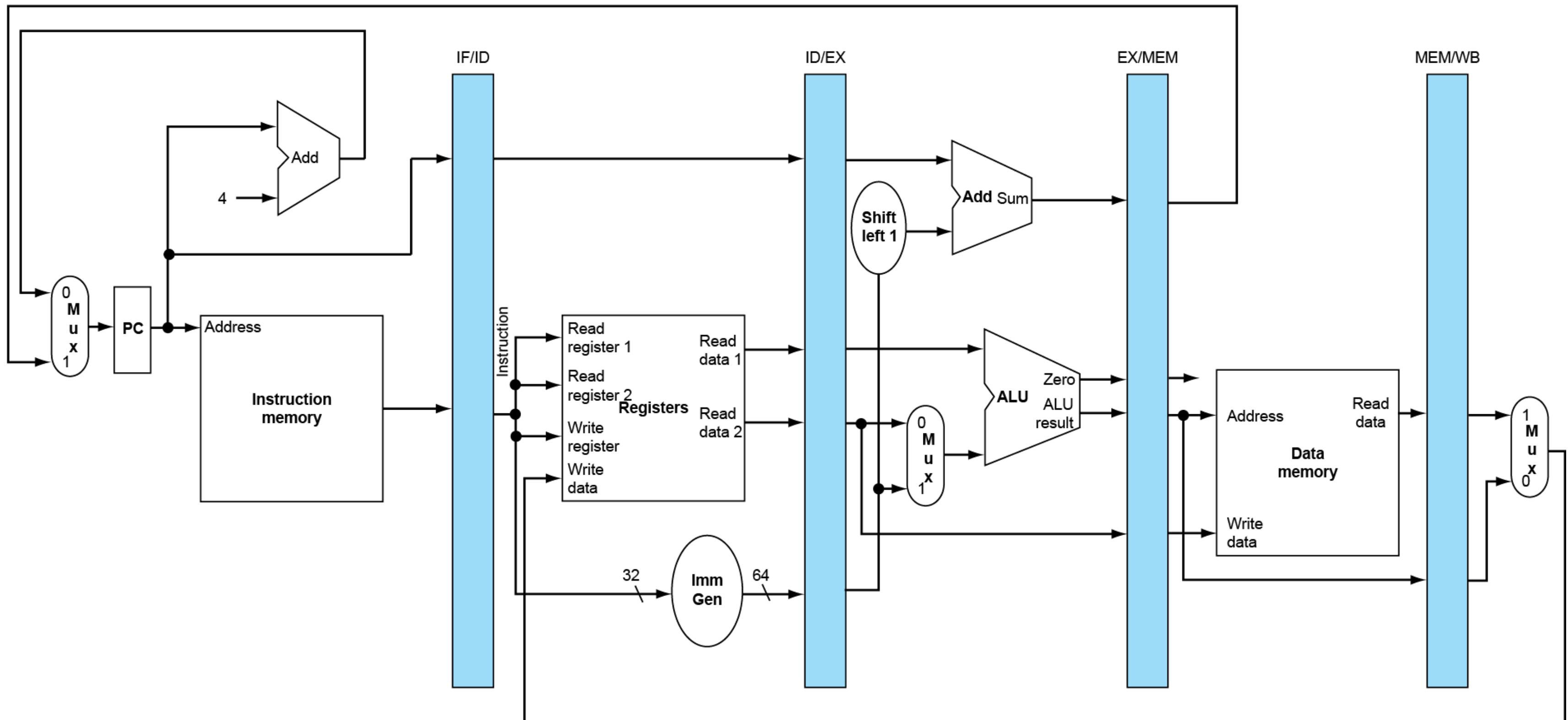
IF: Instruction fetch
ID: Instruction decode/ register file read
EX: Execute/ address calculation
MEM: Memory access
WB: Write back

4
Add
ADD
Sum
Shift left 1
0 Mux 1
PC
Address
Instruction
Instruction memory
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
0 Mux 1
Zero
ALU
ALU result
Address
Read data
Data memory
Write data
1 Mux 0
32
Imm Gen
64

流水线的段寄存器

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

有哪些寄存器？

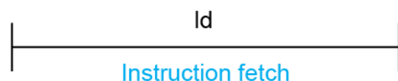


流水线的段寄存器

寄存器命名方式：IF/ID.IR；ID/EX.A; EX/MEM.Aluout；MEM/WB.MDR

load指令的取指阶段

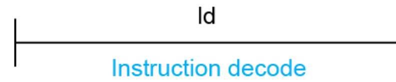
lw x1,100(x2)



load指令的译码阶段

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

lw x1,100(x2)



IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0
M
u
x
1
PC
Address
Instruction
memory
NPC
IR
Instruction
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
32
Imm
Gen
64
NPC
A
B
Imm
Shift
left 1
Add
Sum
ALU
Zero
ALU
result
Address
Read
data
Data
memory
Write
data

load指令的执行阶段

lw x1,100(x2)

ld

Execution

IF/ID
ID/EX
EX/MEM
MEM/WB

load指令的访存阶段

lw x1,100(x2)

ld

Memory

IF/ID

ID/EX

EX/MEM

MEM/WB

Add

4

0 M u x 1

PC

Address

Instruction memory

Instruction

Read register 1

Read register 2

Registers

Write register

Write data

Read data 1

Read data 2

32

Imm Gen

64

Shift left 1

Add Sum

0 M u x 1

ALU

Zero

ALU result

NPC'

Cond

ALUOut

B

Address

Data memory

Write data

Read data

MDR

ALUOut

0 M u x 1

load指令的写回阶段

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

lw x1,100(x2)

ld
Write-back

IF/ID
ID/EX
EX/MEM
MEM/WB

Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
32
Imm Gen
64
Shift left 1
Add Sum
M u x
0
1
ALU
Zero
ALU result
Address
Data memory
Read data
Write data
MDR
ALUOut
M u x
1
0

Lw指令在流水线中的执行

IF/ID ID/EX EX/MEM MEM/WB

Mux 0 1, PC, Add, 4, Address, Instruction memory, NPC, IR, Instruction

Read register 1, Read register 2, Registers, Write register, Write data, Read data 1, Read data 2

32, Imm Gen, 64

NPC, A, B, Imm, Shift left 1, Add Sum, Mux 0 1, ALU, Zero, ALU result

NPC', Cond, ALUOut, B, Address, Read data, Data memory, Write data

MDR, ALUOut, Mux 1 0

流水线的改正版(lw指令)

数据通路由谁控制?

IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
Mux 0 1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
32
Imm Gen
64
Shift left 1
Add Sum
Mux 0 1
ALU
Zero
ALU result
Address
Data memory
Write data
Read data
Mux 1 0

段间寄存器

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

IF/ID ID/EX EX/MEM MEM/WB

Add 4 Mux 0 1 PC Address Instruction memory NPC IR Instruction

Read register 1 Read register 2 Write register Write data Registers Read data 1 Read data 2

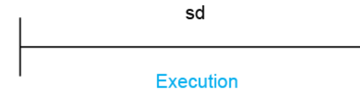
32 Imm Gen 64

NPC A B Imm IR Shift left 1 Add Sum Mux 0 1 ALU Zero ALU result

NPC' Cond ALUOut B IR Address Data memory Write data Read data

MDR ALUOut IR Mux 1 0

EX阶段 for Store指令



IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
PC
M u x
0
1
Address
Instruction memory
Instruction
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
32
Imm Gen
64
Shift left 1
Add Sum
ALU
Zero
ALU result
Address
Data memory
Write data
Read data

MEM阶段 for Store指令

ld

Memory

IF/ID
ID/EX
EX/MEM
MEM/WB

Add
4
0
M
u
x
1
PC
Address
Instruction
memory
Instruction
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
32
Imm
Gen
64
Shift
left 1
Add Sum
0
M
u
x
1
ALU
Zero
ALU
result
Address
Read
data
Data
memory
Write
data
0
M
u
x
1

WB阶段 for Store指令

sd

Write-back

IF/ID
ID/EX
EX/MEM
MEM/WB

Add
4
0
M
u
x
1
PC
Address
Instruction
memory
Instruction
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
32
Imm
Gen
64
Shift
left 1
Add Sum
0
M
u
x
1
ALU
Zero
ALU
result
Address
Data
memory
Write
data
Read
data
1
M
u
x
0

流水线的时空图

Time (in clock cycles)

CC 1 CC 2 CC 3 CC 4 CC 5 CC 6 CC 7 CC 8 CC 9

Program execution order (in instructions)

ld x10, 40(x1) — IM, Reg, ALU, DM, Reg

sub x11, x2, x3 — IM, Reg, ALU, DM, Reg

add x12, x3, x4 — IM, Reg, ALU, DM, Reg

ld x13, 48(x1) — IM, Reg, ALU, DM, Reg

add x14, x5, x6 — IM, Reg, ALU, DM, Reg

Time (in clock cycles)

Program execution order (in instructions)

	CC 1	CC 2	CC 3	CC 4	CC 5	CC 6	CC 7	CC 8	CC 9
ld x10, 40(x1)	Instruction fetch	Instruction decode	Execution	Data access	Write-back				
sub x11, x2, x3		Instruction fetch	Instruction decode	Execution	Data access	Write-back			
add x12, x3, x4			Instruction fetch	Instruction decode	Execution	Data access	Write-back		
ld x13, 48(x1)				Instruction fetch	Instruction decode	Execution	Data access	Write-back	
add x14, x5, x6					Instruction fetch	Instruction decode	Execution	Data access	Write-back

CC5周期的流水线状态

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

add x14, x5, x6	ld x13, 48(x1)	add x12, x3, x4	sub x11, x2, x3	ld x10, 40(x1)
Instruction fetch	Instruction decode	Execution	Memory	Write-back

IF/ID
ID/EX
EX/MEM
MEM/WB

Add
4
0 Mux 1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
32
Imm Gen
64
Shift left 1
Add Sum
0 Mux 1
ALU
Zero
ALU result
Address
Data memory
Write data
Read data
1 Mux 0

各类指令的完成时间?

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

多周期V.S.流水线

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

IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M u x
0
1
PC
Address
Instruction memory
NPC
IR
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
32
Imm Gen
64
NPC
A
B
Imm
IR
Shift left 1
Add Sum
ALU
Zero
ALU result
M u x
0
1
NPC'
Cond
ALUOut
B
IR
Address
Data memory
Write data
Read data
MDR
ALUOut
IR
M u x
1
0

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; //根据具体情况调整
	PC ← (if PCSrc {EX/MEM.NPC’} else {PC+4});		

ID	ID/EX.A ← Regs[IF/ID.IR_{rs1}]; ID/EX.B ← Regs[IF/ID.IR_{rs2}]; ID/EX.NPC ← IF/ID.NPC; ID/EX.IR ← IF/ID.IR; ID/EX.Imm ← $(IF/ID.IR_{15})^{16}\#\#IR_{15...0}$; sign extend		
	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<<1; EX/MEM.cond ← (ID/EX.A == ID/EX.B);

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_{rd}] ← MEM/WB.ALUOut; (R)	Regs[MEM/WB.IR_{rd}] ← MEM/WB.MDR;	

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

流水线中的控制信号

流水线控制信号

PCSrc
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
Instruction [31–0]
32
Imm Gen
64
Instruction [30, 14-12]
Instruction [11-7]
Shift left 1
Add Sum
Branch
ALUSrc
Zero
Add ALU result
ALU control
ALUOp
MemWrite
Address
Read data
Data memory
Write data
MemRead
MemtoReg
1
M
u
x
0

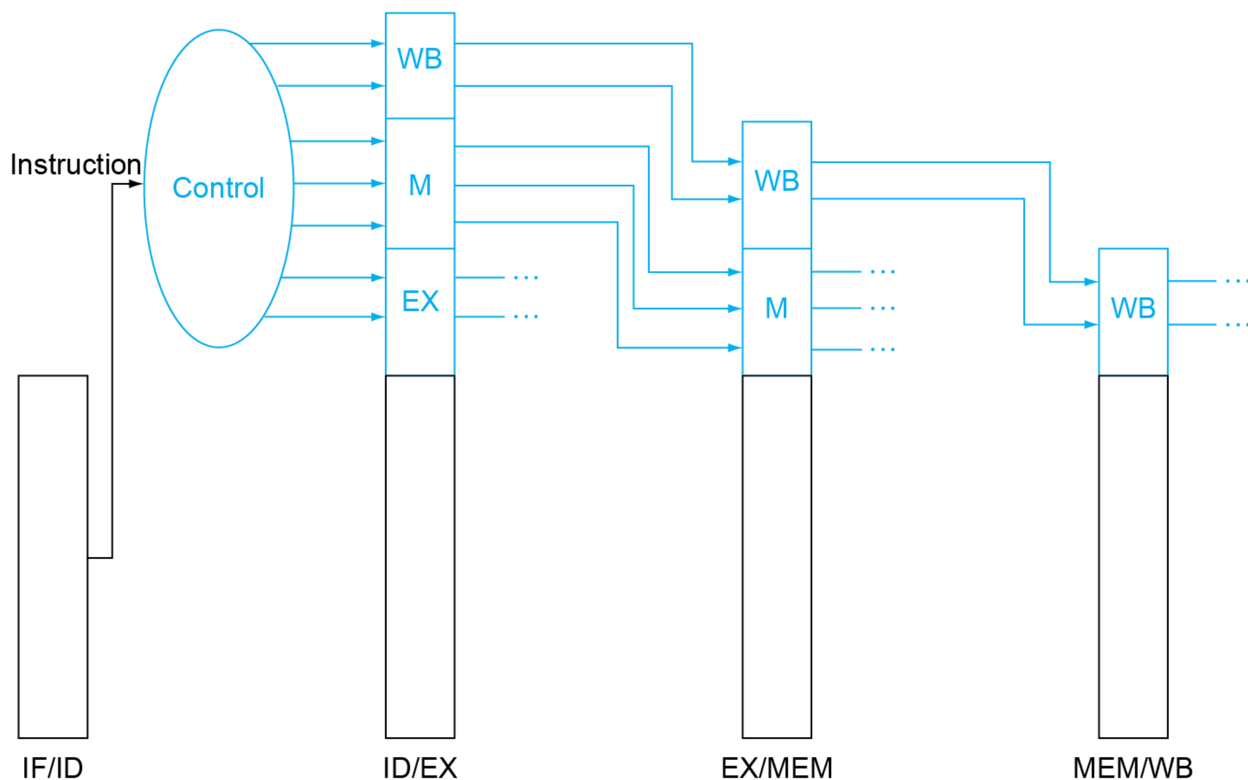
为何没有PC写入信号?

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：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	
	ALUOp	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	10	0	0	0	0	1	0
ld	00	1	0	1	0	1	1
sd	00	1	0	0	1	0	X
beq	01	0	1	0	0	0	X

控制的传递



- 需要控制缓存：类似load指令的目的寄存器号传递

流水线寄存器的控制

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

PCSrc
ID/EX
EX/MEM
MEM/WB
IF/ID
Control
WB
M
EX
Add
4
PC
Mux
0
1
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [31–0]
32
Imm Gen
64
Instruction [30, 14-12]
Instruction [11-7]
Shift left 1
Add Sum
ALUSrc
ALU
Zero
ALU result
ALU control
ALUOp
Branch
MemWrite
Data memory
Read data
MemRead
MemtoReg

MIPS 与RISC-V数据通路的区别

单周期数据通路-MIPS

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

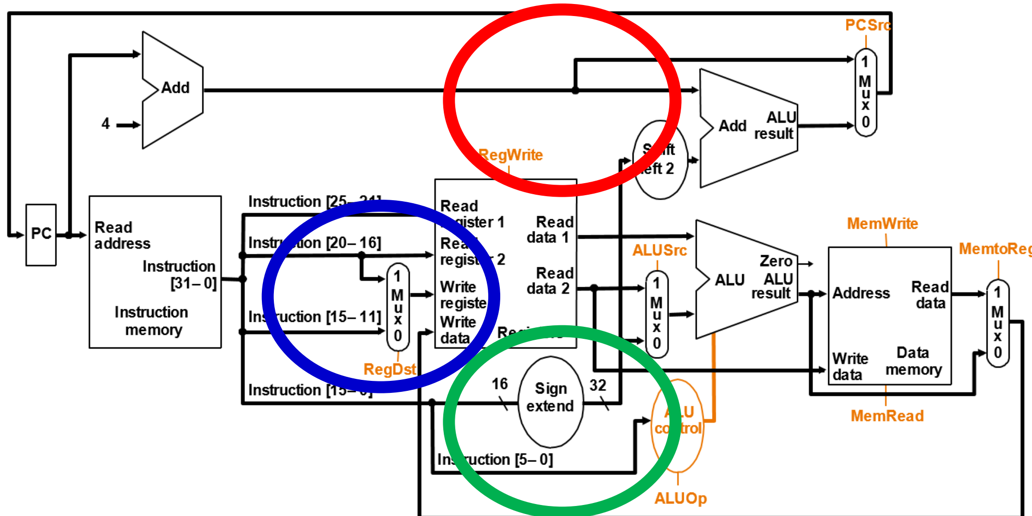
PCSrc
Add
4
Add
ALU result
Shift left 2
RegWrite
PC
Read address
Instruction [31– 0]
Instruction memory
Instruction [25– 21]
Instruction [20– 16]
Instruction [15– 11]
Instruction [15– 0]
Instruction [5– 0]
RegDst
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers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

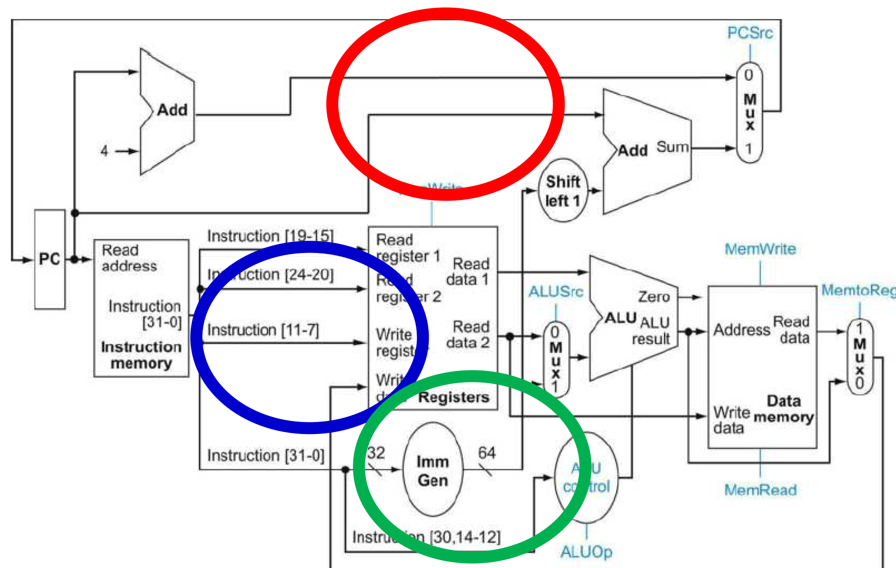
单周期数据通路-RISC-V

PCSrc
Add
4
0
M
u
x
1
Add Sum
Shift left 1
RegWrite
PC
address
Instruction [31-0]
Instruction memory
Instruction [19-15]
Instruction [24-20]
Instruction [11-7]
Read
register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
ALUSrc
0
M
u
x
1
Zero
ALU
ALU result
MemtoReg
Address
Read data
Data memory
Write data
1
M
u
x
0
Instruction [31-0]
32
Imm Gen
64
ALU control
Instruction [30,14-12]
ALUOp
MemRead

MIPS V.S. RISC-V 单周期

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





流水线数据通路-MIPS

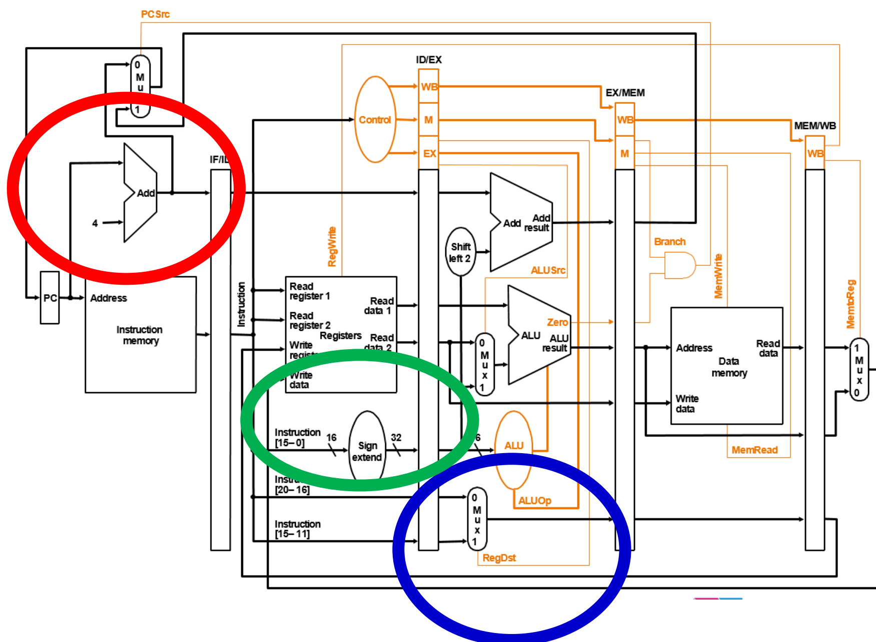
中国科学技术大学
University of Science and Technology of China

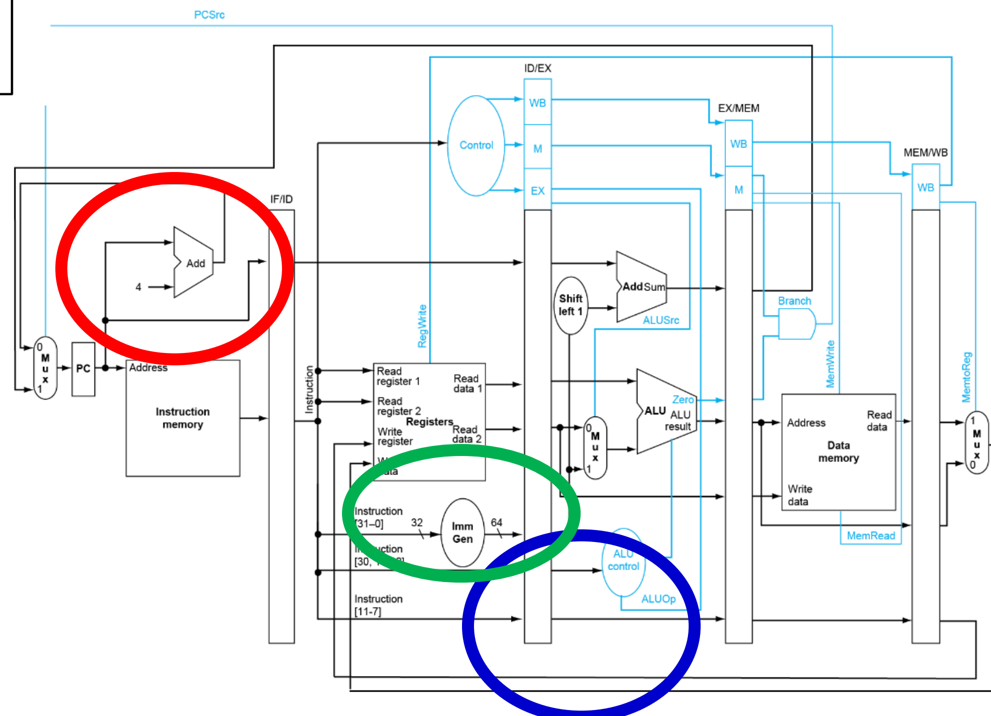
PCSrc
IF/ID
ID/EX
EX/MEM
MEM/WB
0 M u x 1
Add
4
PC
Address
Instruction memory
Instruction
RegWrite
Control
WB
M
EX
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [15–0]
16
Sign extend
32
Instruction [20–16]
Instruction [15–11]
Shift left 2
Add
Add result
ALUSrc
0 M u x 1
ALU
Zero
ALU result
6
ALU control
ALUOp
RegDst
Branch
MemWrite
Address
Data memory
Read data
Write data
MemRead
MemtoReg
1 M u x 0

流水线数据通路-RISC-V

PCSrc
ID/EX
EX/MEM
MEM/WB
IF/ID
Control
WB
M
EX
Add
4
Mux
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
Shift left 1
Add Sum
ALUSrc
Branch
MemWrite
MemtoReg
ALU
Zero
ALU result
Address
Read data
Data memory
Write data
MemRead
Instruction [31–0]
32
Imm Gen
64
Instruction [30, 14-12]
ALU control
ALUOp
Instruction [11-7]

MIPS V.S. RISC-V 流水线





MIPS 与RISC-V数据通路的区别（单周期、流水线）

1、分支目标地址PC计算方法

2、目标寄存器编号、多路选择器

3、立即数扩展、移位

4、…☺

考虑为何造成上述区别？（指令集编码、设计偏好等）

相 关

流水线的相关与冒险

- 依赖（Dependencies ），冲突（Hazards，冒险）
- 结构相关：多条指令需要使用同一资源。当指令在重叠执行的过程中，硬件资源满足不了指令重叠执行的要求，发生资源冲突时将产生**结构冒险**。
- 数据相关：当一条指令需要用到前面指令的执行结果。而这些指令均在流水线中重叠执行时，就可能引起**数据冒险**。
- 控制相关：由分支指令（如跳转、条件判断）引起的依赖，后续指令的执行依赖于分支指令结果。当流水线遇到分支指令和其他会改变PC值的指令时，会发生**控制冒险**。

流水线的相关与冒险

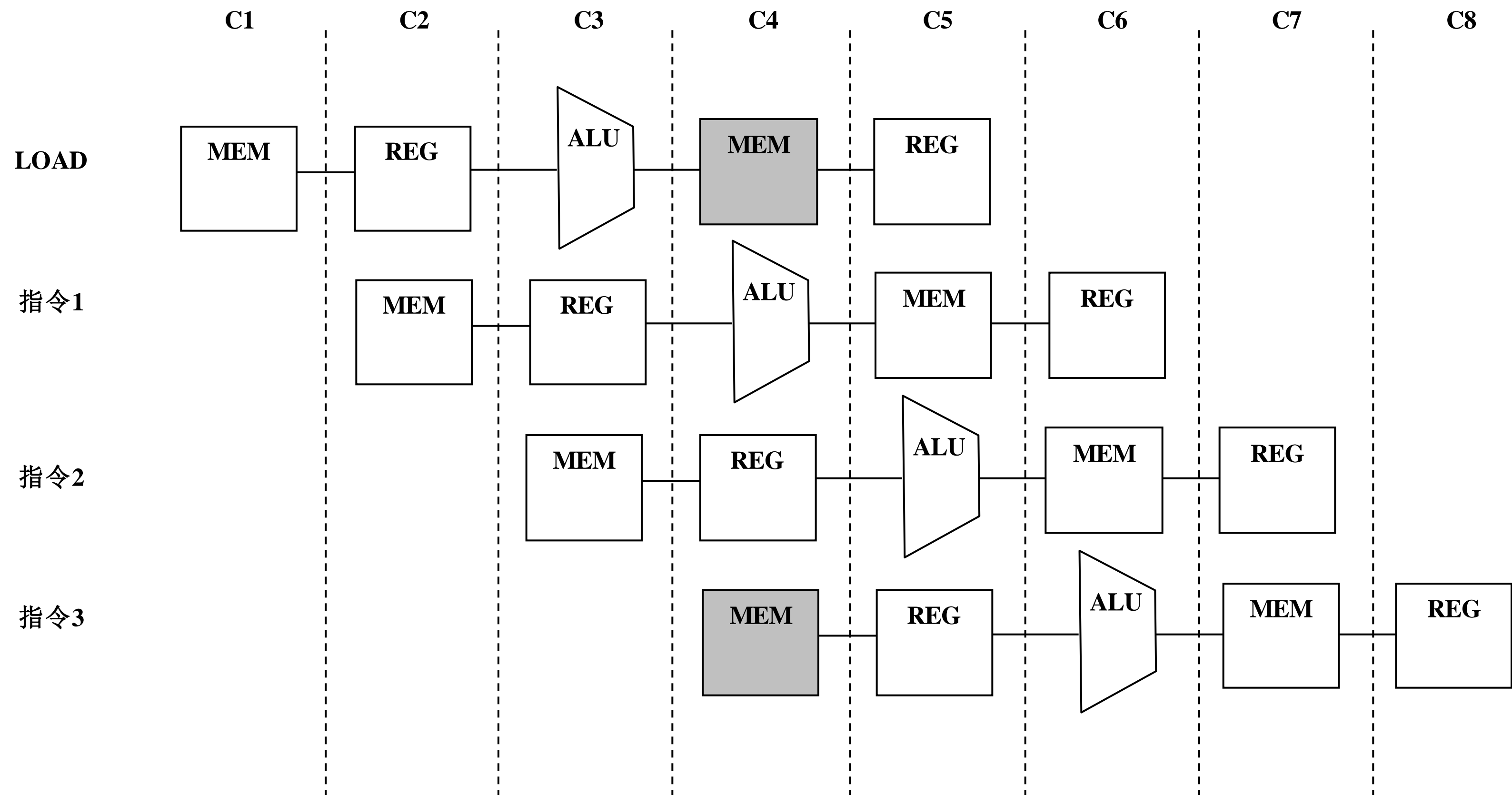
中国科学技术大学 University of Science and Technology of China

特征	相关（Dependencies）	冒险（Hazards）
本质	程序固有的依赖关系	流水线硬件实现中遇到的问题
关注点	指令间的逻辑关系	流水线执行的实际冲突
是否可避免	不可避免（由程序逻辑决定）	可通过硬件技术缓解（如旁路、预测等）
解决方式	无法消除，需硬件或编译器处理	通过流水线优化（如停顿、前递、预测）

联系

- **相关是冒险的根源**：数据、控制、结构相关分别导致数据、控制、结构冒险。
- **冒险是相关的后果**：相关在流水线中被实际执行时，才会表现为冒险。

结构相关



- 典型的结构相关由访存和寄存器造成
- 问：访存相关和寄存器相关的例子？

解决方案1：流水线停顿（stall）

C1 C2 C3 C4 C5 C6 C7 C8

LOAD MEM REG ALU MEM REG

指令1 MEM REG ALU MEM REG

指令2 MEM REG ALU MEM REG

指令3 MEM REG ALU MEM REG

- 气泡（bubble）：流水线停顿一个周期
  - ✓实现：不改变PC，重新取指

解决方案2：访存结构冲突消除

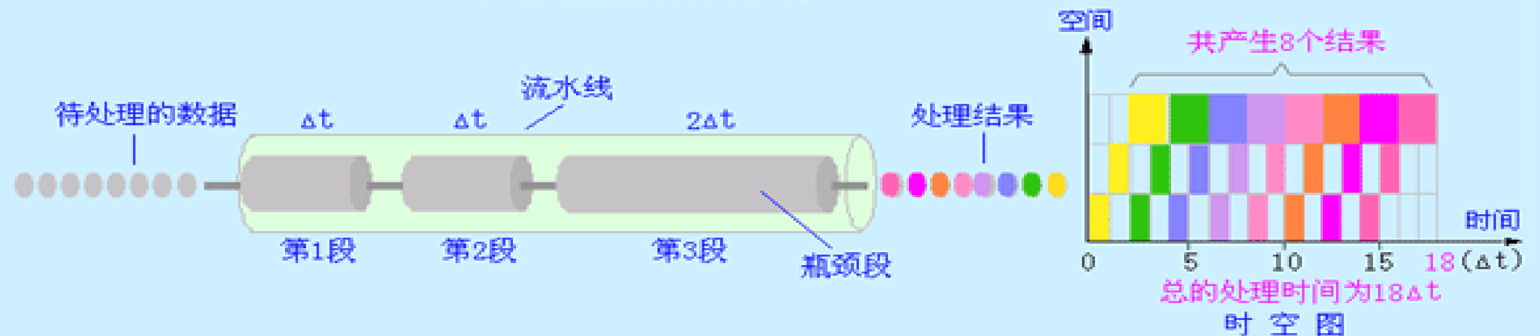
中国科学技术大学
University of Science and Technology of China

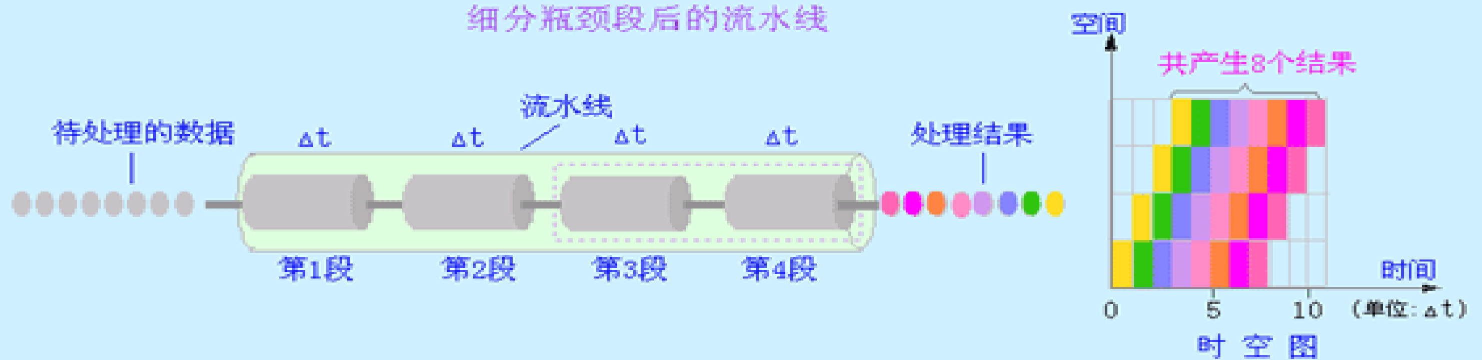
	C1	C2	C3	C4	C5	C6	C7	C8
LOAD	MEM	REG	ALU	MEM D	REG			
指令1		MEM	REG	ALU	MEM	REG		
指令2			MEM	REG	ALU	MEM	REG	
指令3				MEM I	REG	ALU	MEM	REG

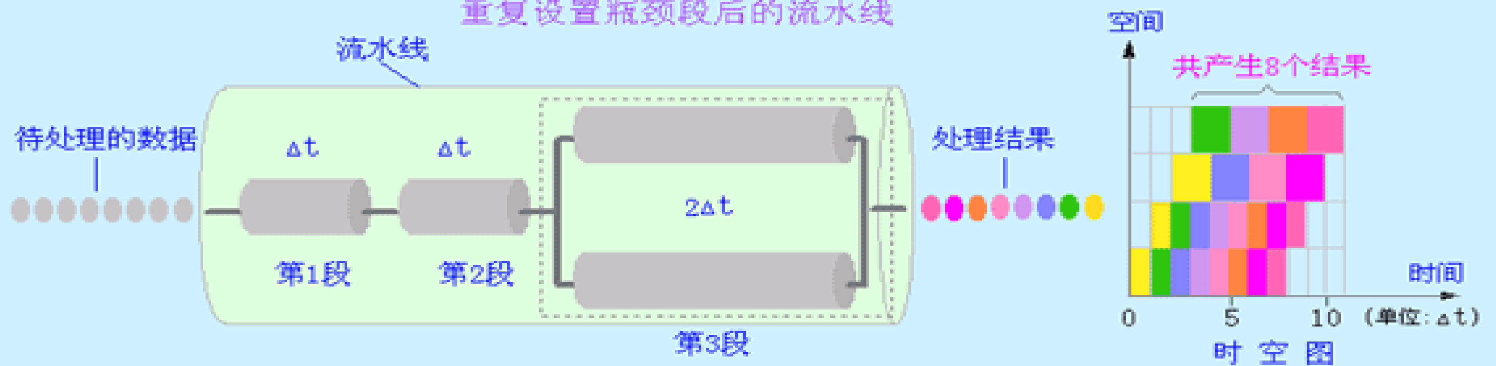
- 访存结构冒险化解：哈佛结构

结构相关

流水线的改进







数据相关

1. 数据相关简介

实例：

```
SUB  x2,  x1 , x3
AND  x12, x2 , x5
OR   x13, x6 , x2
ADD  x14, x2 , x2
SW   x15, 100(x2)
```

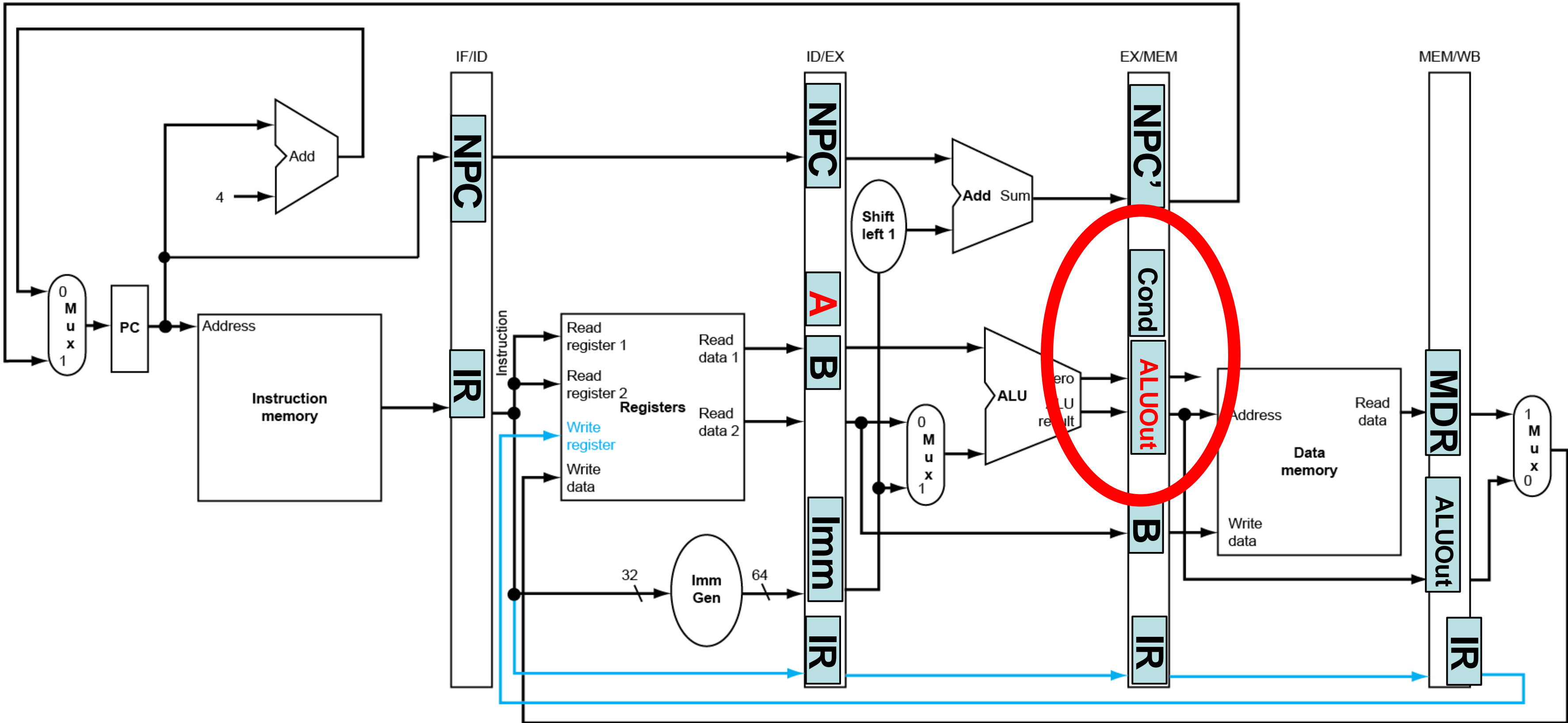
产生原因：当指令在流水线中重叠执行时，流水线有可能改变指令读/写操作数的顺序，使之不同于它们在非流水实现时的顺序，这将导致数据相关。

OR x13， x6 , x2

SUB x2， x1 , R3

AND x12， x2 , x5

问： x2的正确值在哪？



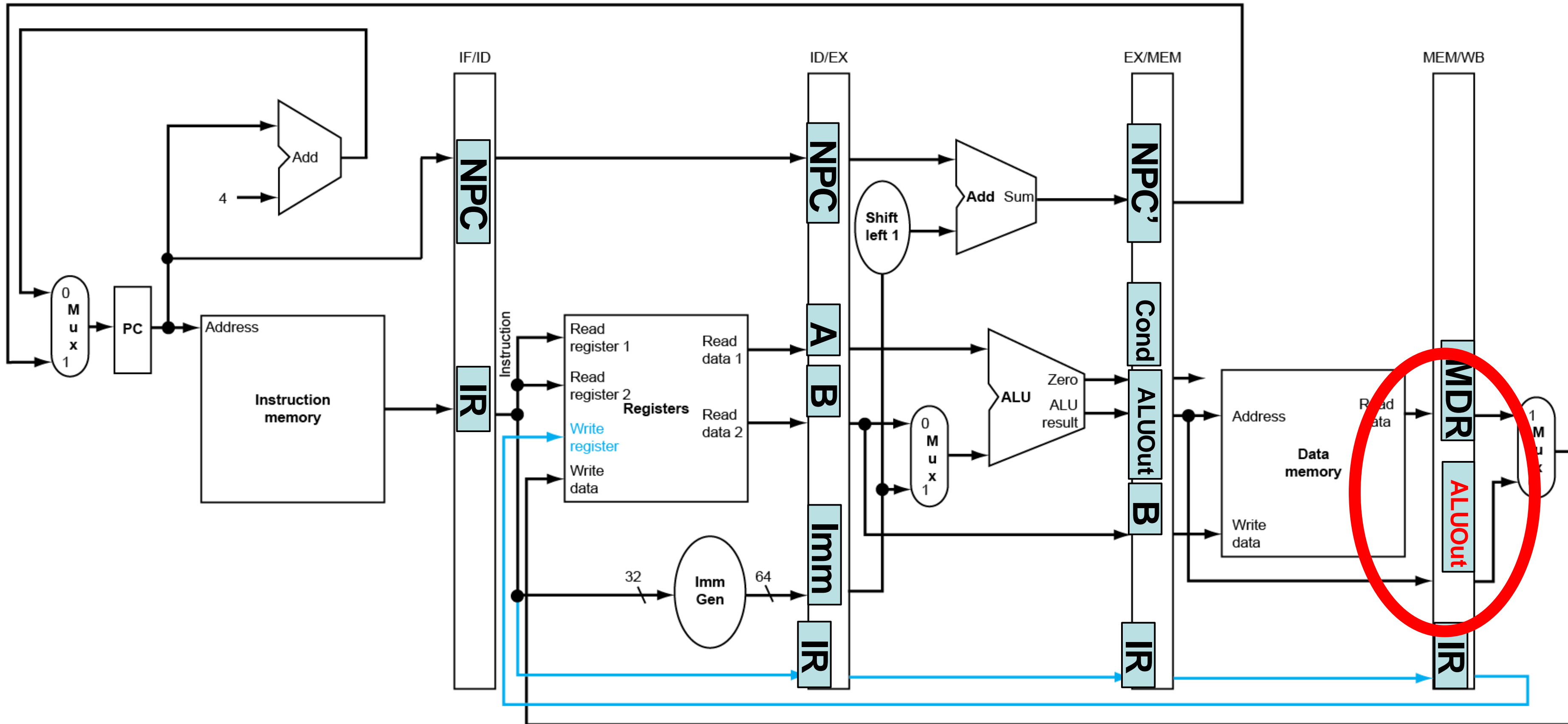
ADD x14, x2 , x2

OR x13, x6 , x2

SUB x2, x1 , R3

AND x12, x2 , x5

问：x2的正确值在哪？



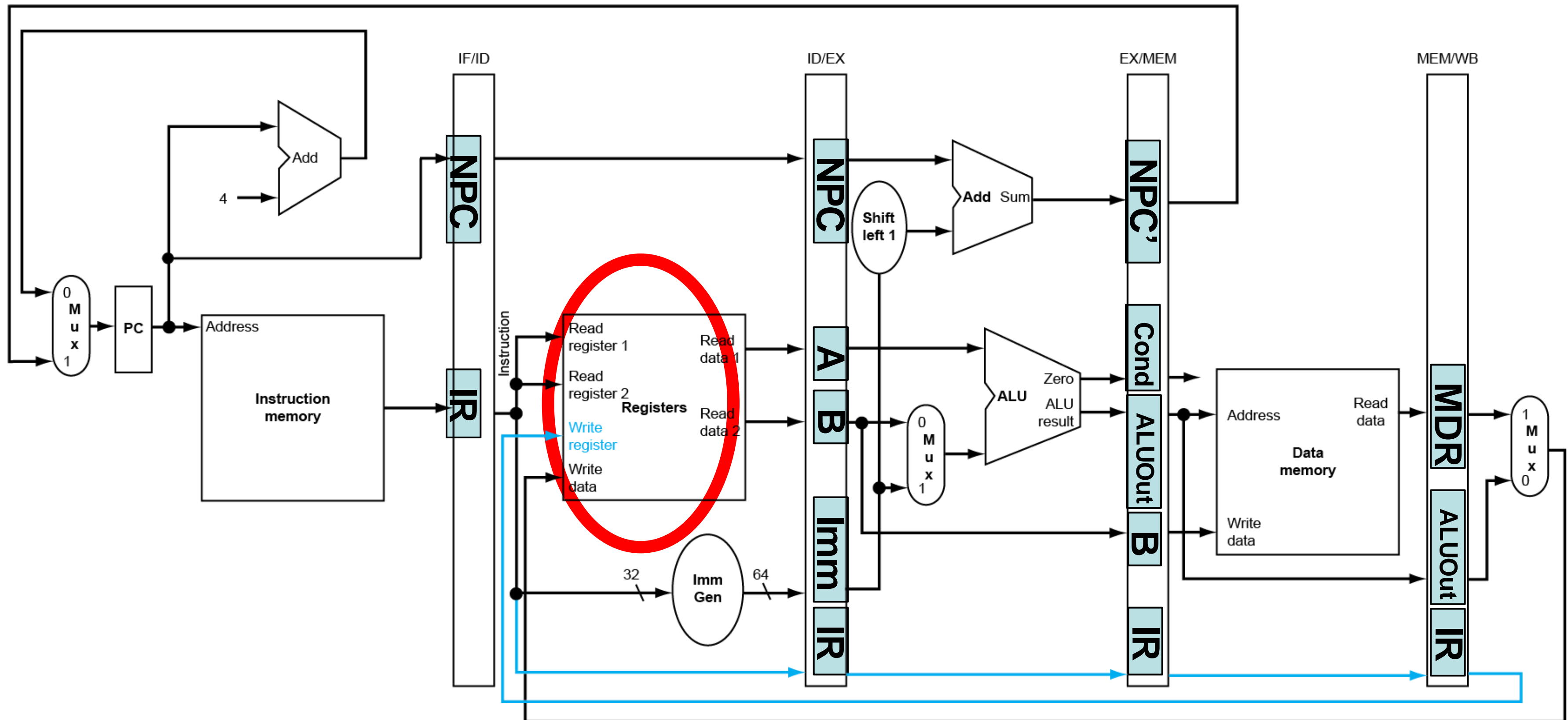
ADD x14, x2 , x2

SUB x2, x1 , R3

SW x15, 100(x2)

OR x13, x6 , x2

AND x12, x2 , x5



数据相关（哪个周期数据可用？）

Time (in clock cycles)

Value of register x2:	CC 1	CC 2	CC 3	CC 4	CC 5	CC 6	CC 7	CC 8	CC 9
	10	10	10	10	10/–20	–20	–20	–20	–20

Program execution order (in instructions)

sub x2, x1, x3

and x12, x2, x5

or x13, x6, x2

add x14, x2, x2

sd x15, 100(X2)

IM Reg DM Reg

数据相关分类

- 写后读相关（RAW）： j的执行要用到i的计算结果，j可能在i写入其计算结果之前就先行对保存该结果的寄存器进行读操作。
  - ✓ “读”快，“写”慢
- 写后写相关（WAW）： j和i的目的操作数一样，写入顺序错误，在目标寄存器中留下的是i的值而不是j的值。
  - ✓ 后面的“写”快，前面的“写”慢
- 读后写相关（WAR）： j可能在i读取某个源寄存器的内容之前就先对该寄存器进行写操作，导致i读出的是错误的数据。
  - ✓ 后面的“写”快，前面的“读”慢
- 流水线何时会产生WAW和WAR相关?

WAW和WAR相关

ADD指令提前写回

LW R1, 0, (R2)	IF	ID	EX	MEM1	MEM2	WB
ADD R1, R2, R3		IF	ID	EX	WB	

ADDD F4,F0,Name（ALU指令访存)
LD F0,0(R1)

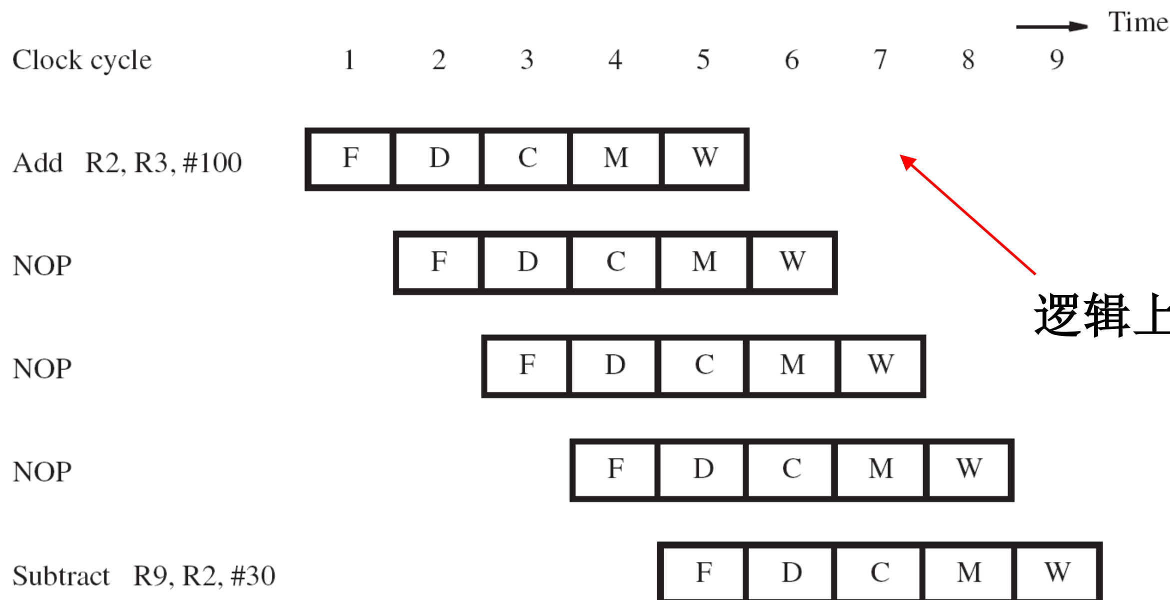
IF	ID	EX	MEM1	MEM1	MEM1	RD
	IF	ID	EX	WB		

RAW的判断规则

- 沿流水线传递寄存器号
  - ✓e.g., ID/EX.RegisterRs1表示流水线寄存器ID/EX 中Rs1的寄存器号
- EX级中ALU操作数寄存器号为ID/EX.RegisterRs1, ID/EX.RegisterRs2
- EX段
  - 1a. **EX/MEM.RegisterRd = ID/EX.RegisterRs1**
  - 1b. **EX/MEM.RegisterRd = ID/EX.RegisterRs2**
- MEM段
  - 2a. **MEM/WB.RegisterRd = ID/EX.RegisterRs1**
  - 2b. **MEM/WB.RegisterRd = ID/EX.RegisterRs2**

消除RAW

- Stall
  - ✓ 硬件控制，动态技术（流水线互锁 Interlock）
  - ✓ 编译技术，静态技术
    - 插入NOP
    - 调度无关指令：兼容性问题（新旧版本“延迟槽”数可能不同）



逻辑上直到C6，数据才可用

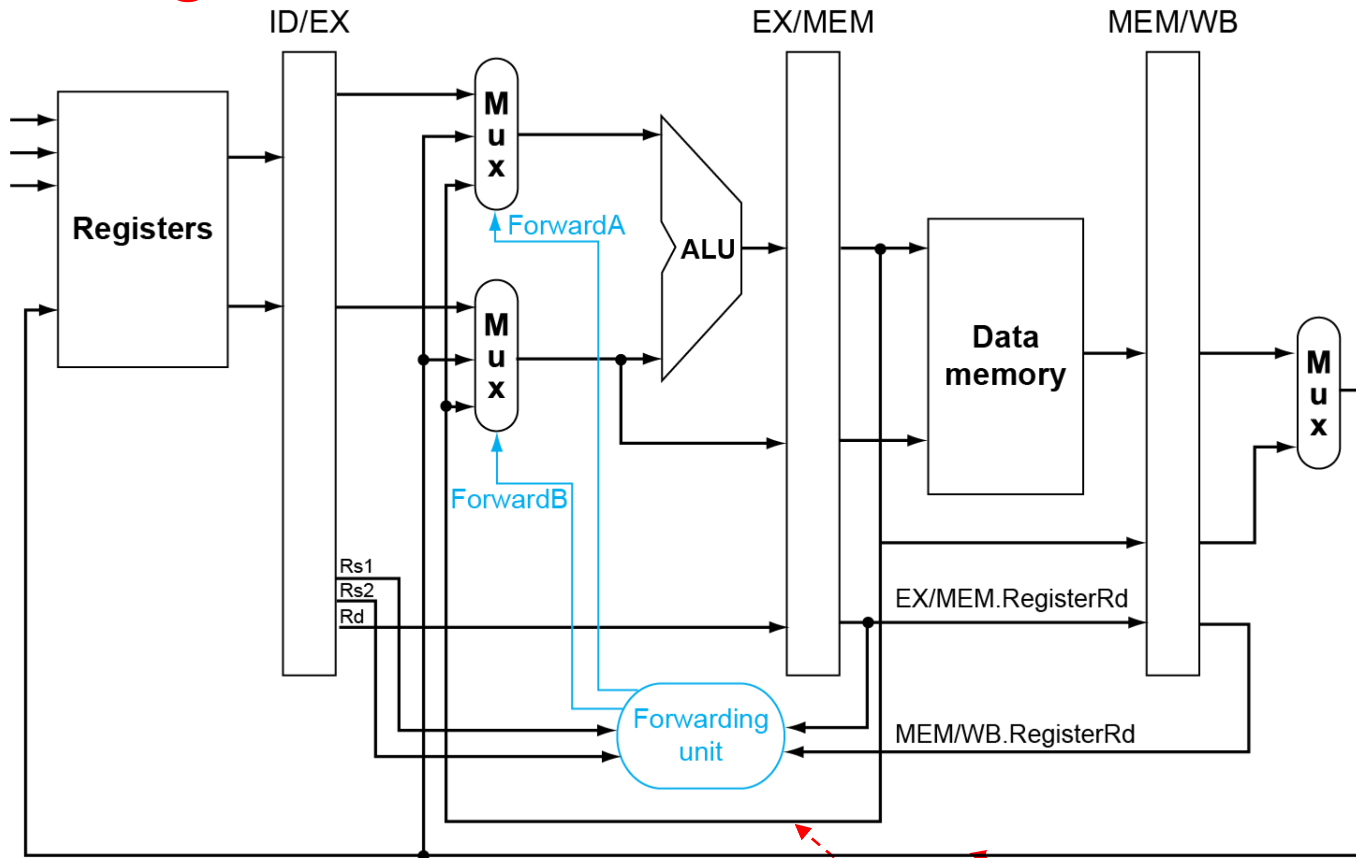
- 硬件和软件实现的区别

消除RAW：数据定向技术

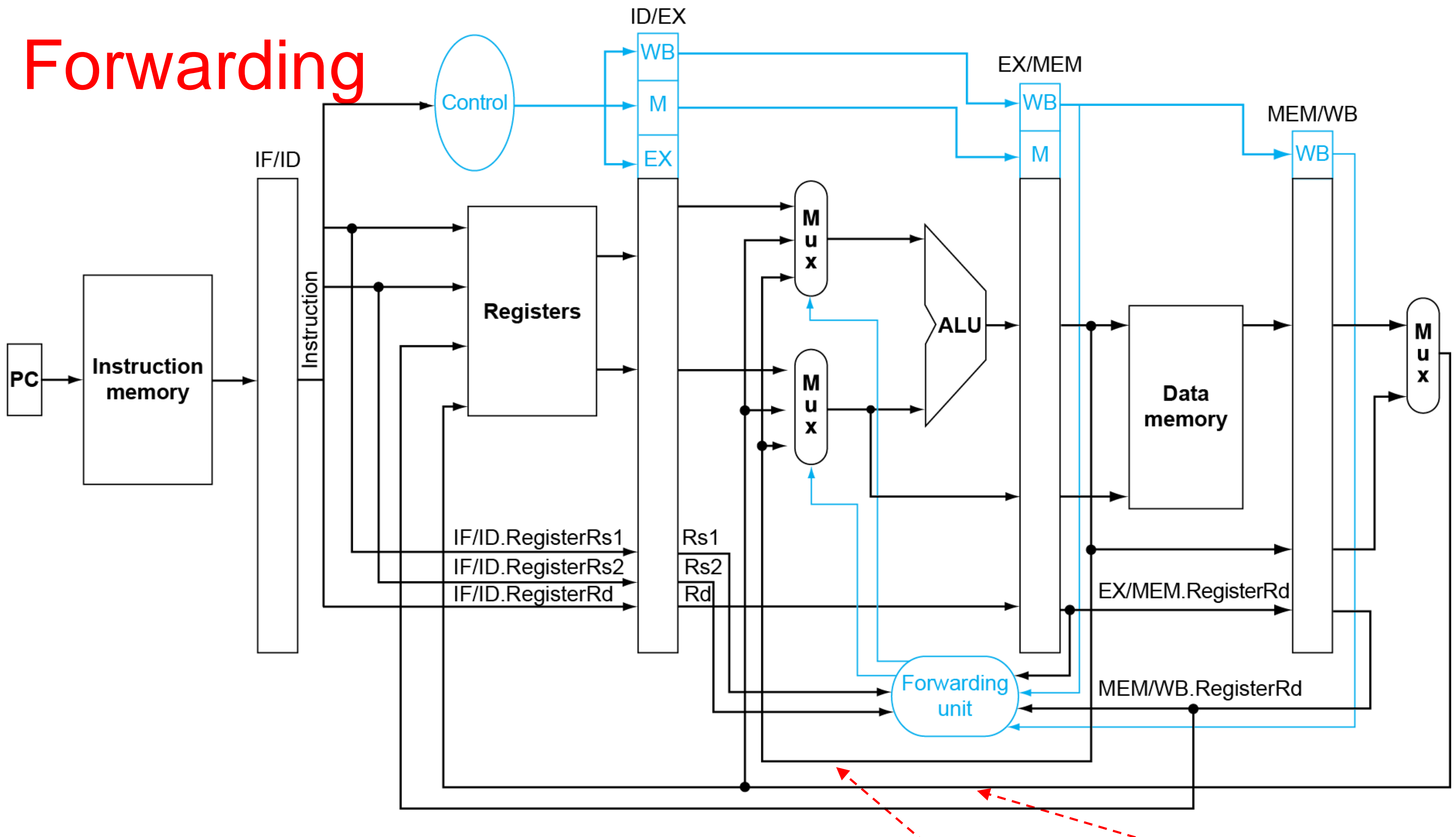
- forwarding，bypassing（定向或旁路，硬件）
  - ✓数据从实际存放的位置直接传到需要的位置

	C1	C2	C3	C4	C5	C6	C7	C8
ADD	MEM	REG	ALU	MEM	REG			
SUB		MEM	REG	ALU	MEM	REG		
AND			MEM	REG	ALU	MEM	REG	
指令				MEM	REG	ALU	MEM	REG

Forwarding

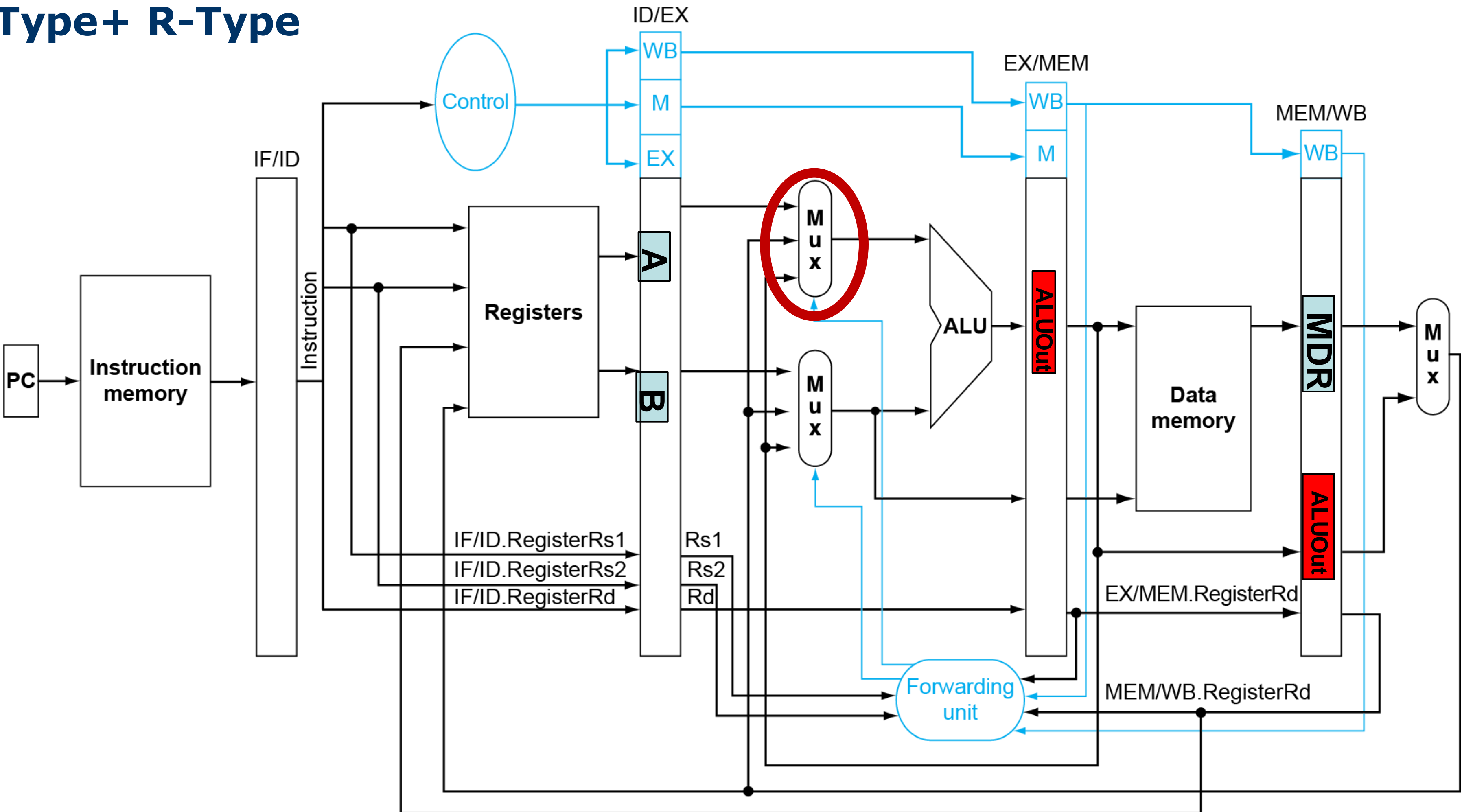


Instruction	Execution/address calculation stage control lines		Memory access stage control lines			Write-back stage control lines	
	ALUOp	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	10	0	0	0	0	1	0
ld	00	1	0	1	0	1	1
sd	00	1	0	0	1	0	X
beq	01	0	1	0	0	0	X



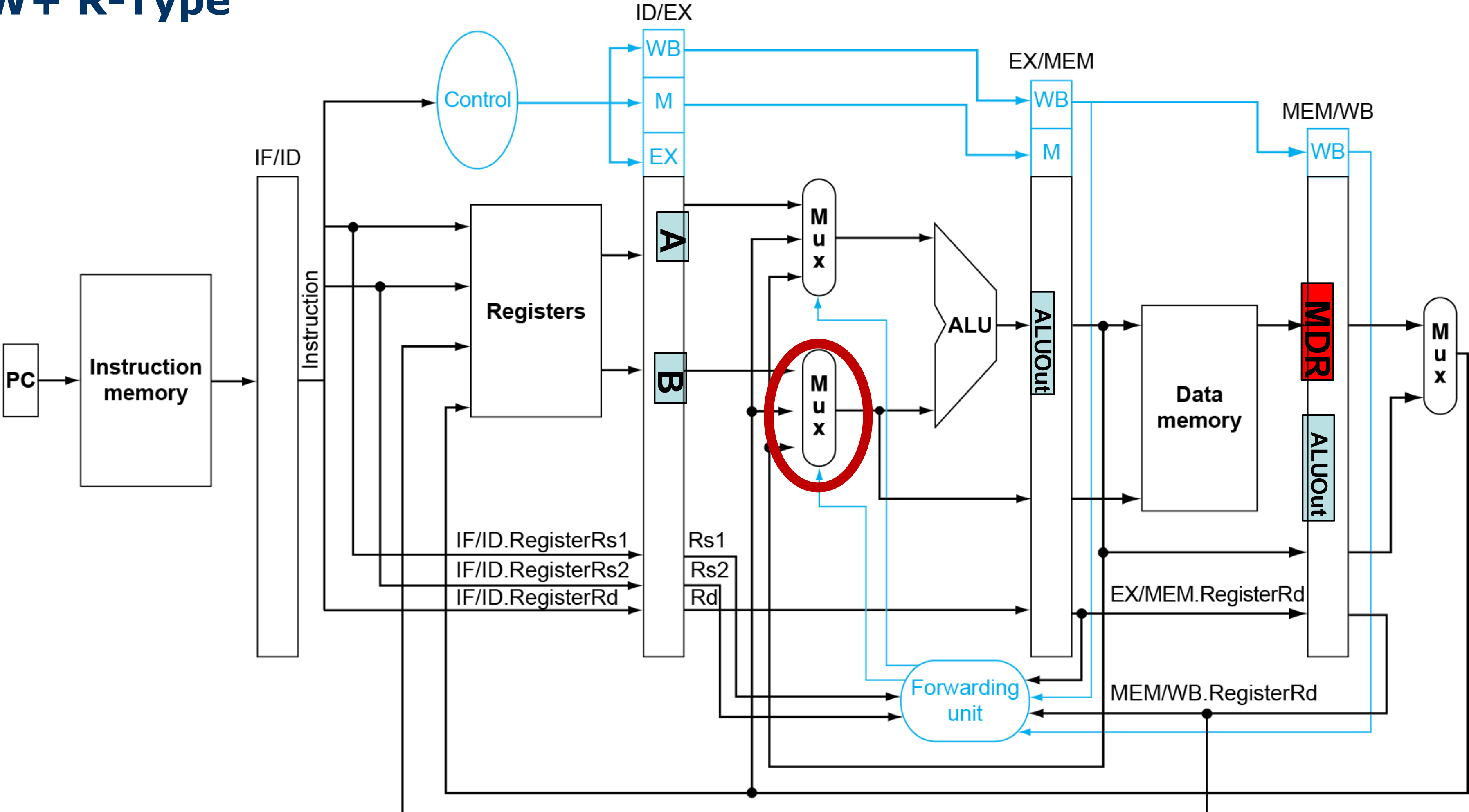
Instruction	Execution/address calculation stage control lines		Memory access stage control lines			Write-back stage control lines	
	ALUOp	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	10	0	0	0	0	1	0
ld	00	1	0	1	0	1	1
sd	00	1	0	0	1	0	X
beq	01	0	1	0	0	0	X

R-Type+ R-Type



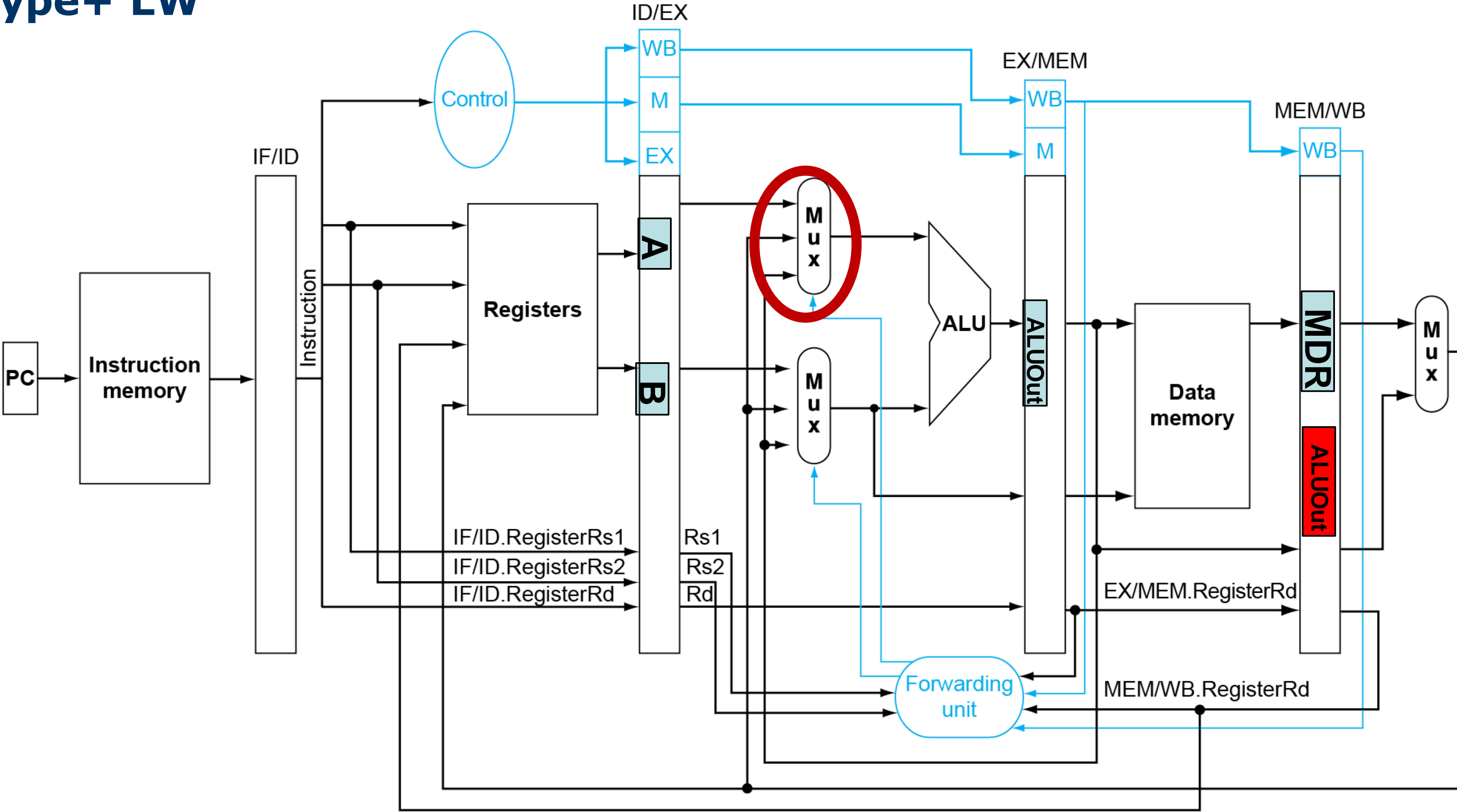
ADD x1, x2, x3	IF	ID	EX	ME	WB				
SUB x5, x1, x7		IF	ID	EX	ME	WB			
XOR x6, x1, x7			IF	ID	EX	ME	WB		
OR x7, x1, x7				IF	ID	EX	ME	WB	

LW+ R-Type



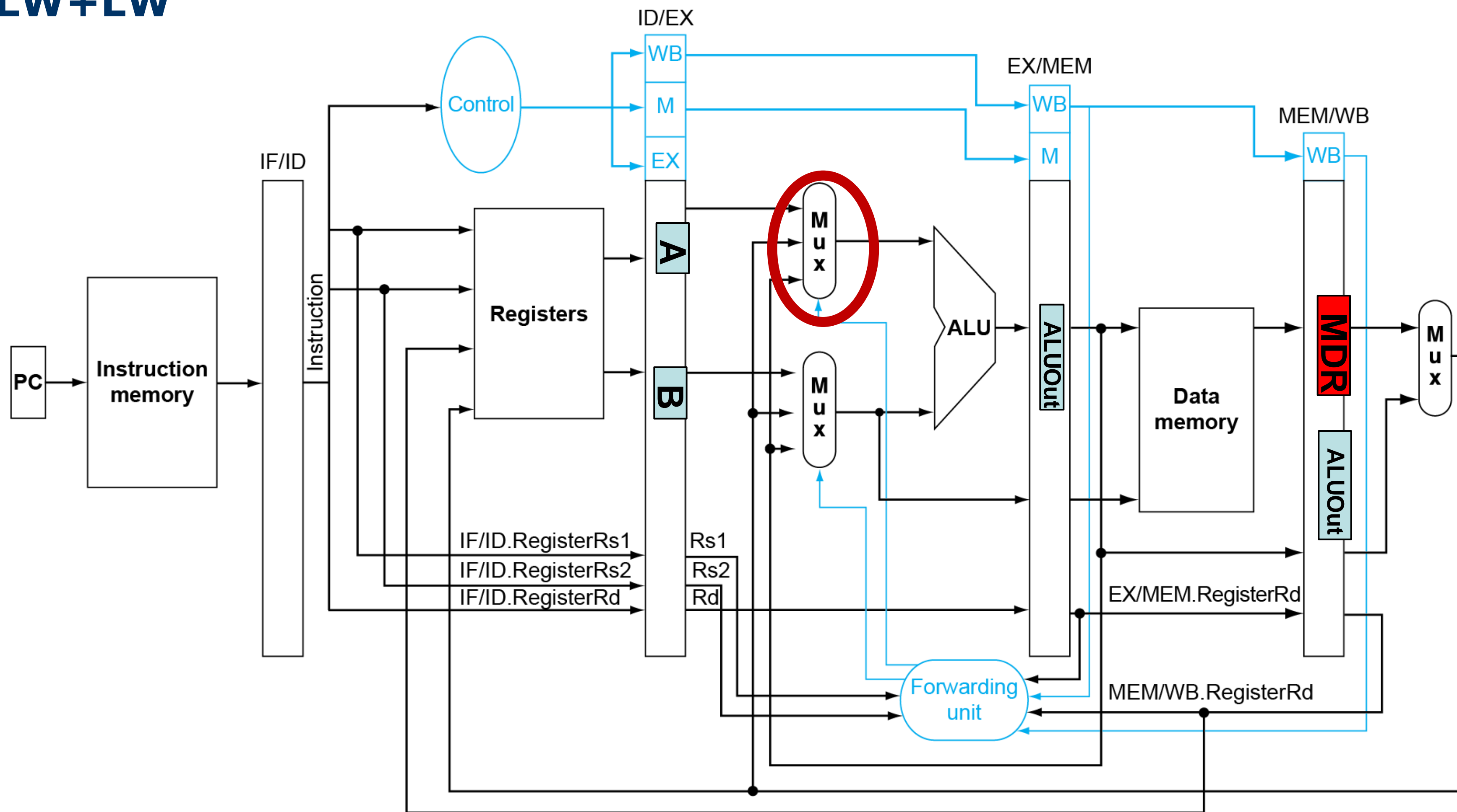
LW x1, 45(x2)	**IF**	**ID**	**EX**	**ME**	**WB**			
SUB x8, x6, x7		**IF**	**ID**	**EX**	**ME**	**WB**		
ADD x5, x7, x1			**IF**	**ID**	**EX**	**ME**	**WB**	

R-Type+ LW



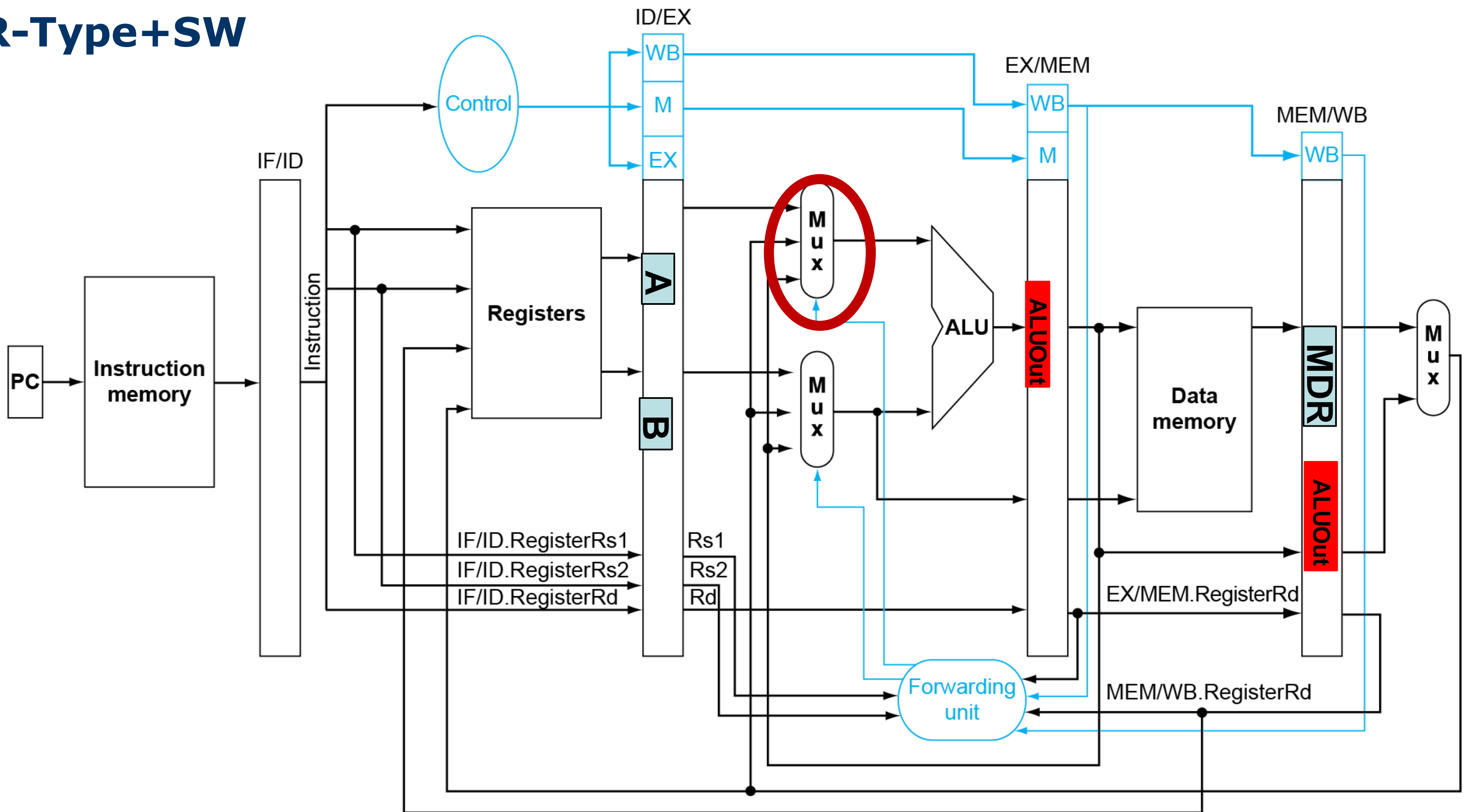
ADD x1, x2, x3	**IF**	**ID**	**EX**	**ME**	**WB**			
SUB x8, x6, x7		**IF**	**ID**	**EX**	**ME**	**WB**		
LW x5, 45(x1)			**IF**	**ID**	**EX**	**ME**	**WB**	

LW+LW



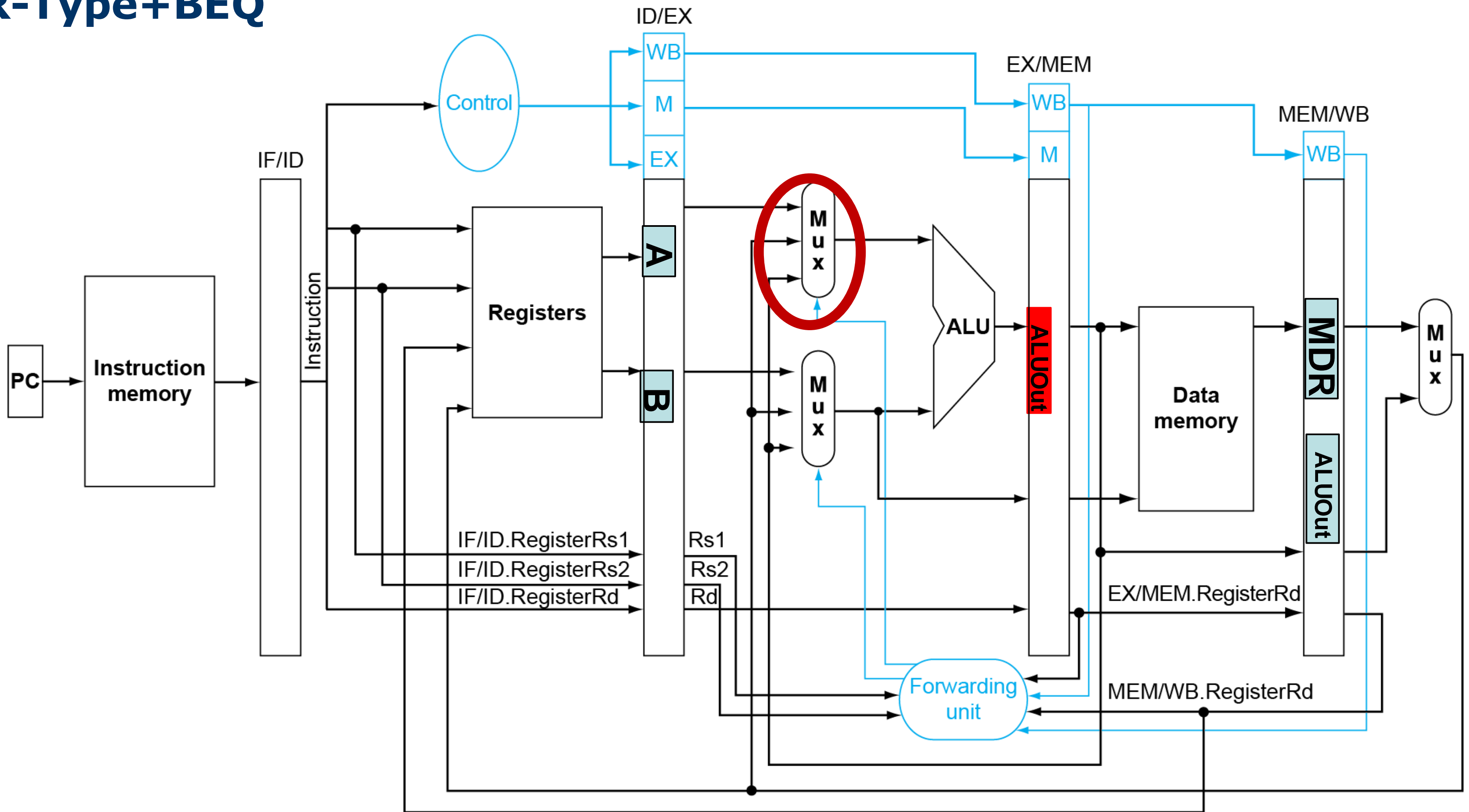
LW	**x1, 30(x2)**	**IF**	**ID**	**EX**	**ME**	**WB**		
SUB	**x8, x6, x7**		**IF**	**ID**	**EX**	**ME**	**WB**	
LW	**x5, 45(x1)**			**IF**	**ID**	**EX**	**ME**	**WB**

R-Type+SW



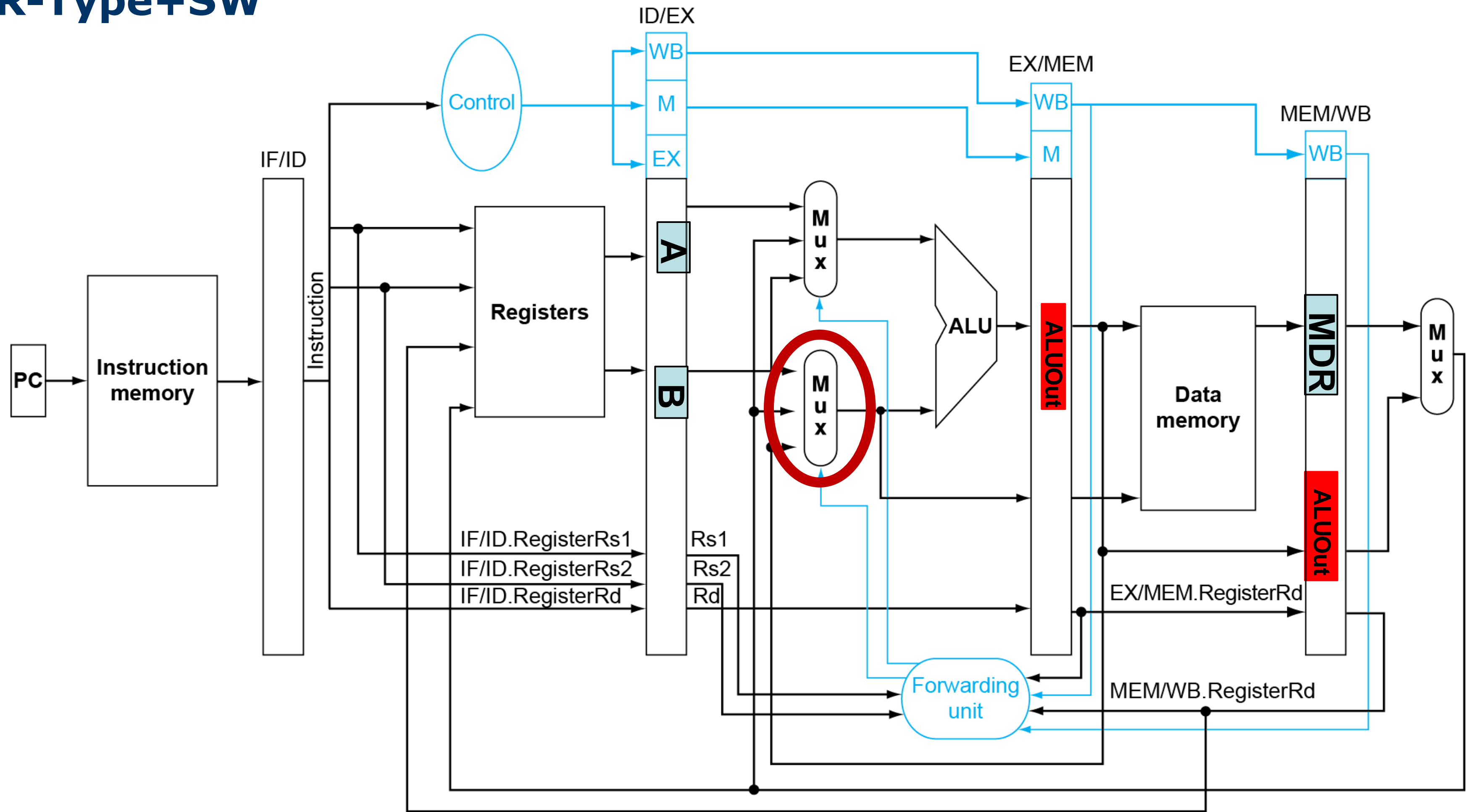
ADD	x1, x2, x3	IF	ID	EX	ME	WB		
SW	x5, 30(x1)		IF	ID	EX	ME	WB	
SW	x6, 45(x1)			IF	ID	EX	ME	WB

R-Type+BEQ



ADD	x1, x2, x3	IF	ID	EX	ME	WB	
BEQ	x1, x4, Offset		IF	ID	EX	ME	WB

R-Type+SW



ADD x1, x2, x3	IF	ID	EX	ME	WB			
SW x1, 30(x3)		IF	ID	EX	ME	WB		
SW x1, 45(x4)			IF	ID	EX	ME	WB	

LW+SW

ID/EX
WB
M
EX
Control
EX/MEM
WB
M
MEM/WB
WB
IF/ID
Instruction
PC
Instruction memory
Registers
A
B
Mux
Mux
ALU
ALUOut
Data memory
MDR
ALUOut
Mux
IF/ID.RegisterRs1
IF/ID.RegisterRs2
IF/ID.RegisterRd
Rs1
Rs2
Rd
EX/MEM.RegisterRd
Forwarding unit
MEM/WB.RegisterRd

LW	x1, 56(x2)	IF	ID	EX	ME	WB			
SW	x1, 45(x3)		IF	ID	EX	ME	WB		
SW	x1, 45(x4)			IF	ID	EX	ME	WB	

Forwarding之后的时空图

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

Time (in clock cycles) →

	CC 1	CC 2	CC 3	CC 4	CC 5	CC 6	CC 7	CC 8	CC 9
Value of register x2 :	10	10	10	10	10/–20	–20	–20	–20	–20
Value of EX/MEM :	X	X	X	–20	X	X	X	X	X
Value of MEM/WB :	X	X	X	X	–20	X	X	X	X

Program execution order (in instructions)

sub x2, x1, x3

and x12, x2, x5

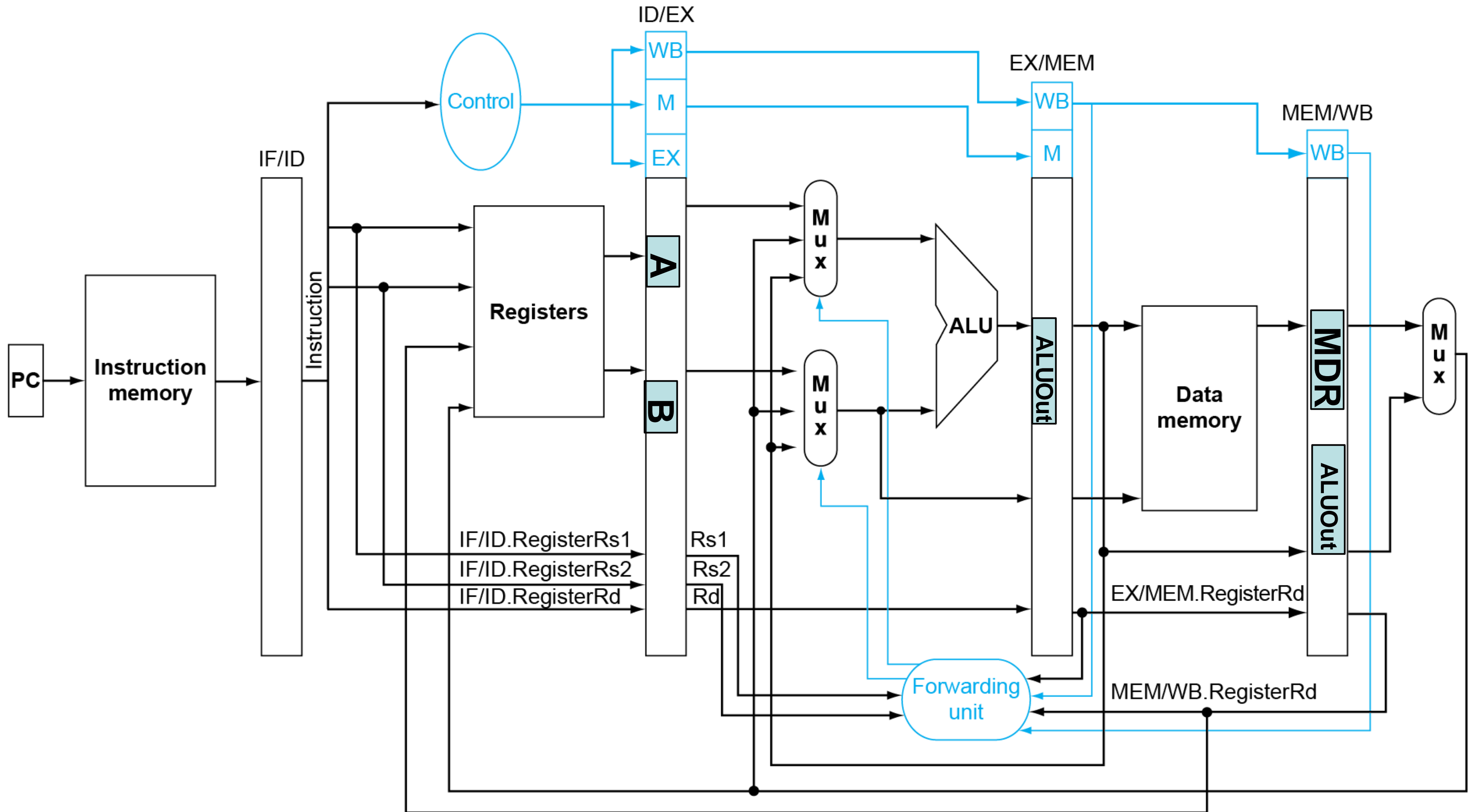
or x13, x6, x2

add x14, x2, x2

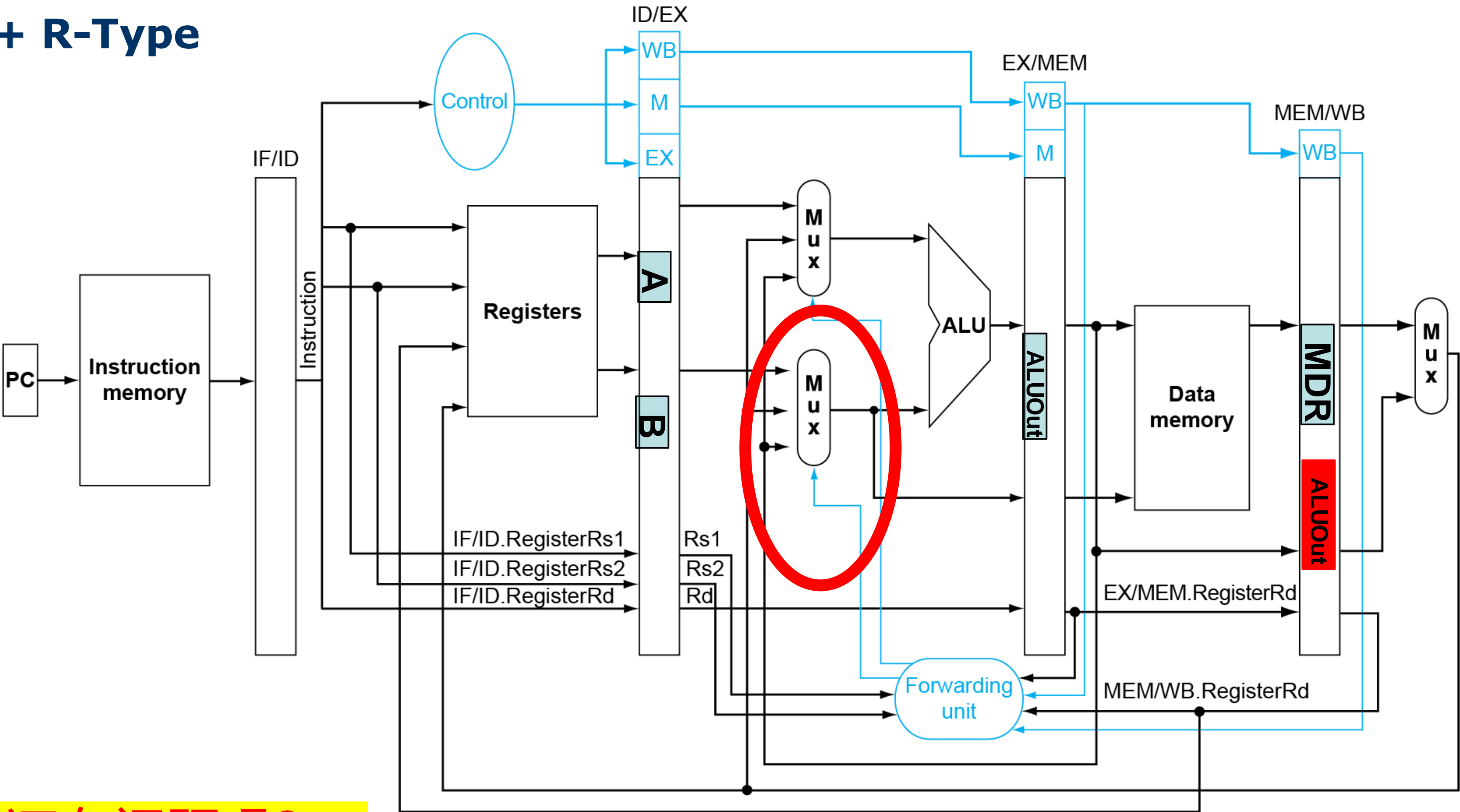
sw x15, 100(x2)

Forwarding之后每个时钟周期都能执行一条指令！

等等，还有问题吗？



LW+ R-Type



LW	x1, 45(x2)	IF	ID	EX	ME	WB			
ADD	x5, x7, x1		IF	ID	EX	ME	WB		
ADD	x5, x7, x1			IF	ID	EX	ME	WB	

Forwarding无法解决的RAW

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

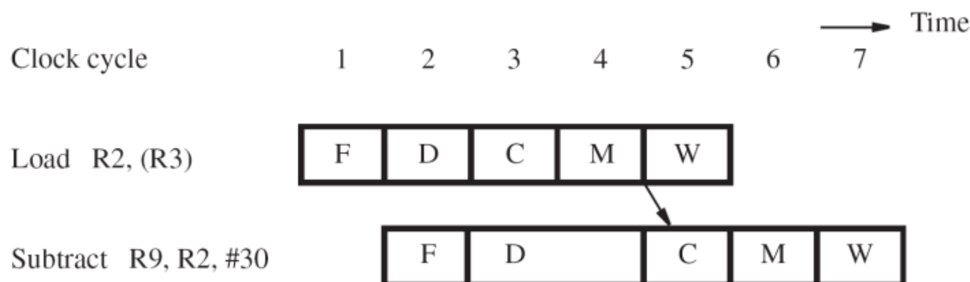
Time (in clock cycles)

Program execution order (in instructions)

	CC 1	CC 2	CC 3	CC 4	CC 5	CC 6	CC 7	CC 8	CC 9
lw x2, 20(x1)	IM	Reg		DM	Reg				
and x4, x2, x5		IM	Reg		DM	Reg			
or x8, x2, x6			IM	Reg		DM	Reg		
add x9, x4, x2				IM	Reg		DM	Reg	
slt x1, x6, x7					IM	Reg		DM	Reg

Interlock（互锁）：解决LW相关

- 检测数据依赖：在ID段！
- stall：“freeze up and bubble down”
  - ✓ 冻结其前面的流水段：IF，ID
    - 增加两个控制信号：PCWrite和IF/IDWrite
      - 阻止更新PC和IF/ID
      - 使之持续重复相同的操作
  - ✓ 向EX流水段插入气泡
    - 将ID/EX的控制清零
      - 使功能部件暂停一个周期，再重新发出控制



Interlock

ID/EX.MemRead and
((ID/EX.RegisterRd = IF/ID.RegisterRs1) or
(ID/EX.RegisterRd = IF/ID.RegisterRs2))

freeze up

interlock

bubble down

Id?

Hazard detection unit

ID/EX.MemRead

PCWrite

IF/DWrite

IF/ID

ID/EX

EX/MEM

MEM/WB

WB

M

EX

Control

Mux

0

PC

Instruction memory

Instruction

Registers

ALU

Data memory

ForwardA

ForwardB

Forwarding unit

IF/ID.RegisterRs1

IF/ID.RegisterRs2

IF/ID.RegisterRd

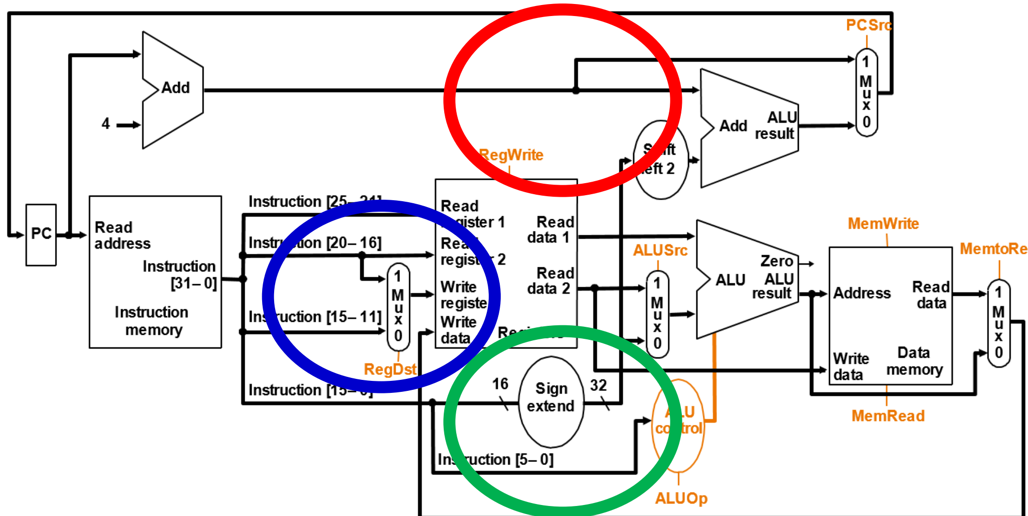
Rd

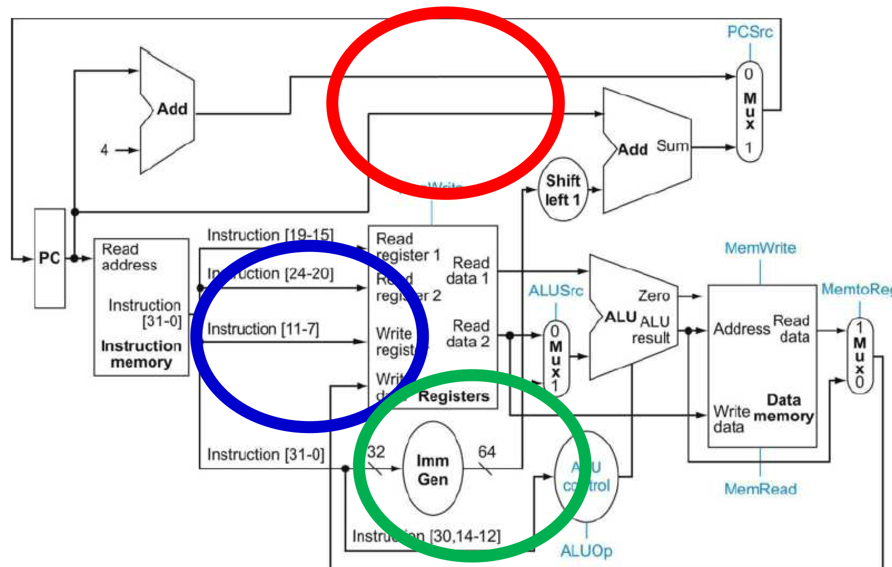
Rs1

Rs2

MIPS V.S. RISC-V 单周期

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





单周期数据通路-RISC-V

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

科学技术大学
y of Science and Technology of China

Design of the RISC-V Instruction Set Architecture

by

Andrew Shell Waterman

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor David Patterson, Chair
Professor Krste Asanović
Associate Professor Per-Olof Persson

Spring 2016

RISC-V 手册

一本开源指令集的指南

DAVID PATTERSON, ANDREW WATERMAN

翻译：勾凌睿、黄成、刘志刚
校阅：包云岗

第七章 压缩指令 **69**
7.1 导言 69
7.2 RV32GC，Thumb-2，microMIPS 和 x86-32 的比较 71
7.3 结束语 71
7.4 扩展阅读 71

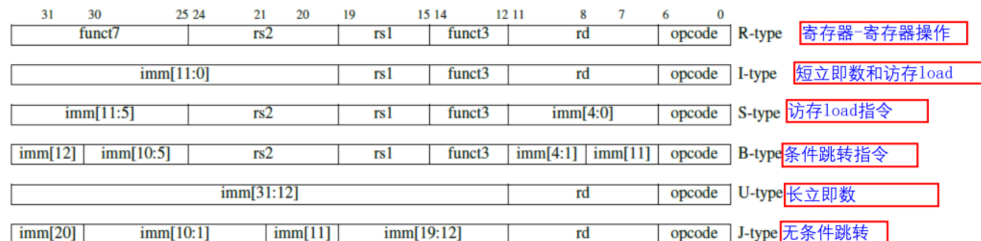
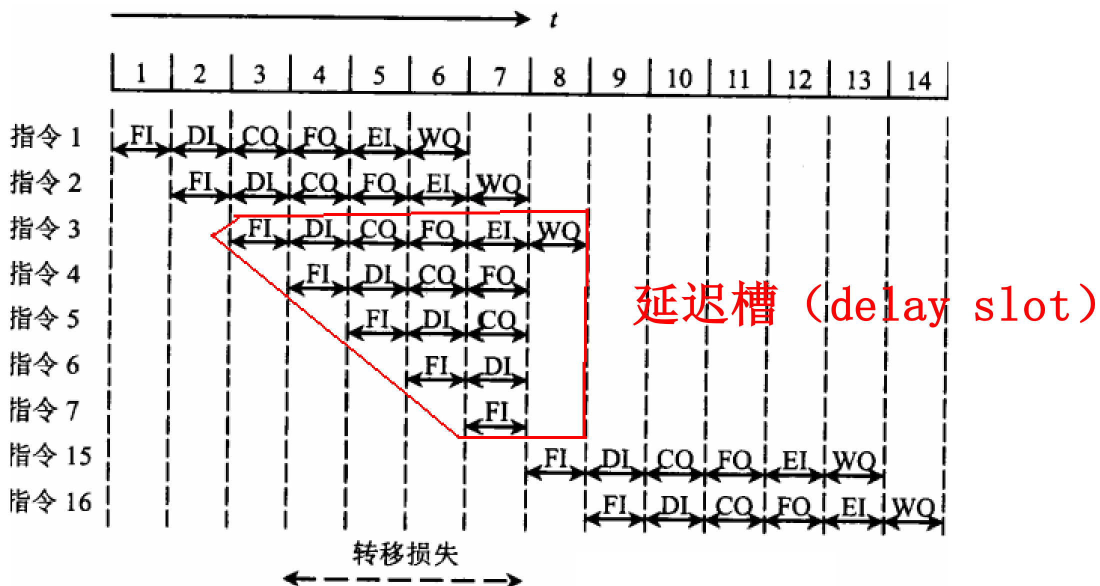


图 2.2：RISC-V 指令格式。我们用生成的立即数值中的位置（而不是通常的指令立即数域中的位置）(imm[*x*])标记每个立即数子域。第十章解释了控制状态寄存器指令使用 I 型格式的稍微不同的做法。（本图基于 Waterman 和 Asanovi'c 2017 的图 2.2）。

控制相关

- 假设指令3是一条条件转移指令
  - ✓ 必须待指令2的结果出现后(第7个时间单元)，才能决定下一条指令是4(条件不满足, no taken)还是15(条件满足, taken)。

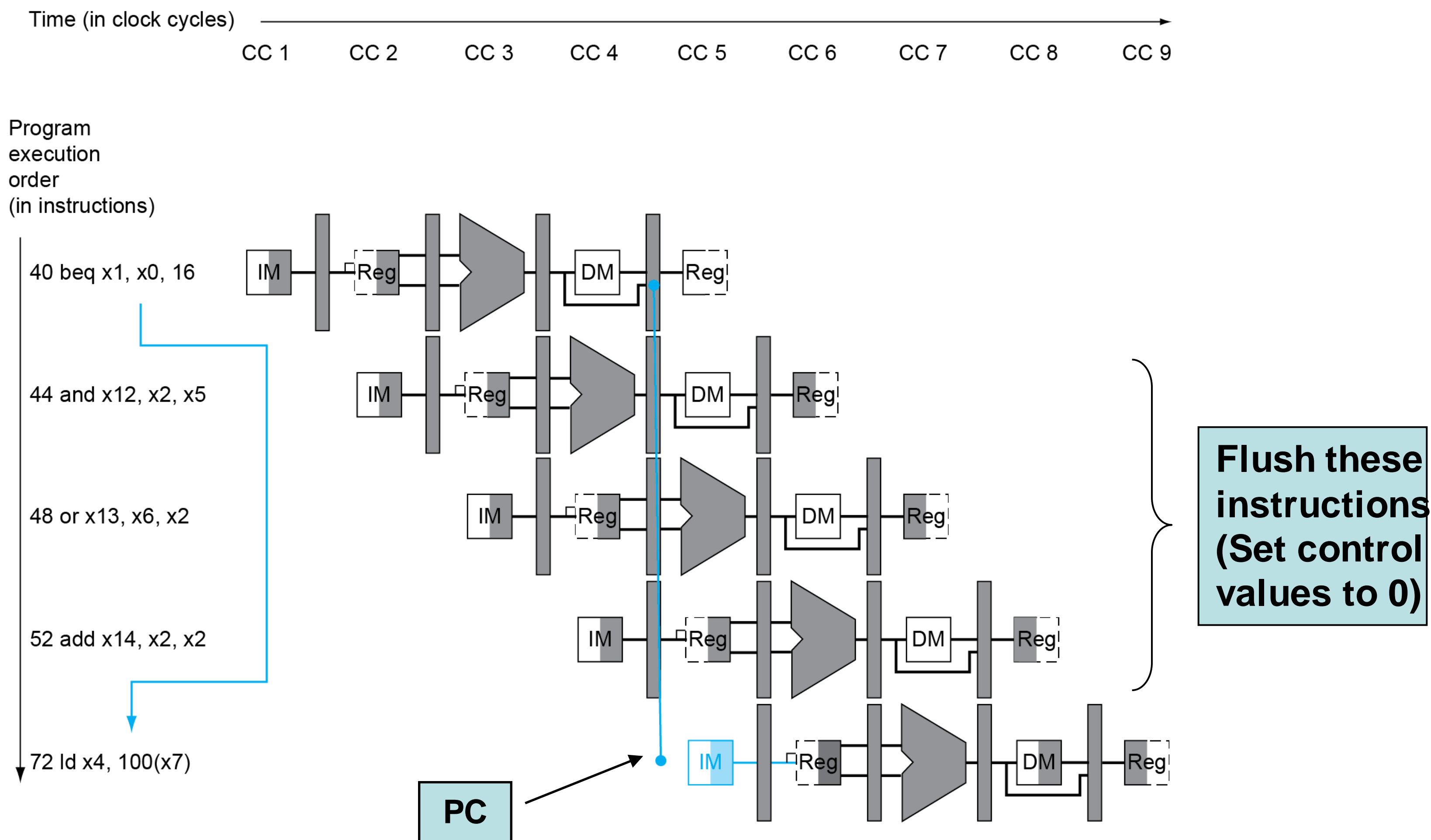


延迟槽（delay slot）

流水线冻结（freeze）与排空（flush，将IR置0）

- 取指(FI):从存储器取出一条指令并暂时存入指令部件的缓冲区。
- 指令译码(DI):确定操作性质和操作数地址的形成方式。
- 计算操作数地址(CO):计算操作数的有效地址,涉及寄存器间接寻址、间接寻址、变址、基址、相对寻址等各种地址计算方式。
- 取操作数(FO):从存储器中取操作数(若操作数在寄存器中,则无须此阶段)。
- 执行指令(EI):执行指令所需的操作,并将结果存于目的位置(寄存器中)。
- 写操作数(WO):将结果存入存储器。

如果分支结果在MEM级确定



优化Beq指令的完成时间

如何让**BEQ**指令在第三和第五个周期内完成？

缩短分支的延迟

- 将用于确定结果的硬件移到ID级
  - ✓ 目标地址加法器
  - ✓ 寄存器比较器
- Example: branch taken

```
36:   sub  x10, x4, x8
40:   beq  x1,  x3, 16  // PC-relative branch
                        // to 40+16*2=72
44:   and  x12, x2, x5
48:   orr  x13, x2, x6
52:   add  x14, x4, x2
56:   sub  x15, x6, x7
      ...
72:   ld   x4, 50(x7)
```

分支成功的情况

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

and x12, x2, x5
beq x1, x3, 16
sub x10, x4, x8
before<1>
before<2>

IF.Flush
Hazard detection unit
Control
ID/EX
EX/MEM
MEM/WB
IF/ID
WB
M
EX
Mux
PC
Instruction memory
Registers
Shift left 1
Imm Gen
ALU
Data memory
Forwarding unit
x1
x3
x4
x8
48
44
4
72
40
32
16
0
10

Clock 3

分支成功的情况

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

ld x4, 50(x7) | Bubble (nop) | beq x1, x3, 16 | sub x10, . . . | before<1>

IF.Flush
Hazard detection unit
Control
IF/ID
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Mux
PC
Instruction memory
Registers
=
Shift left 1
Imm Gen
ALU
Data memory
Forwarding unit
76
72
4
0
10

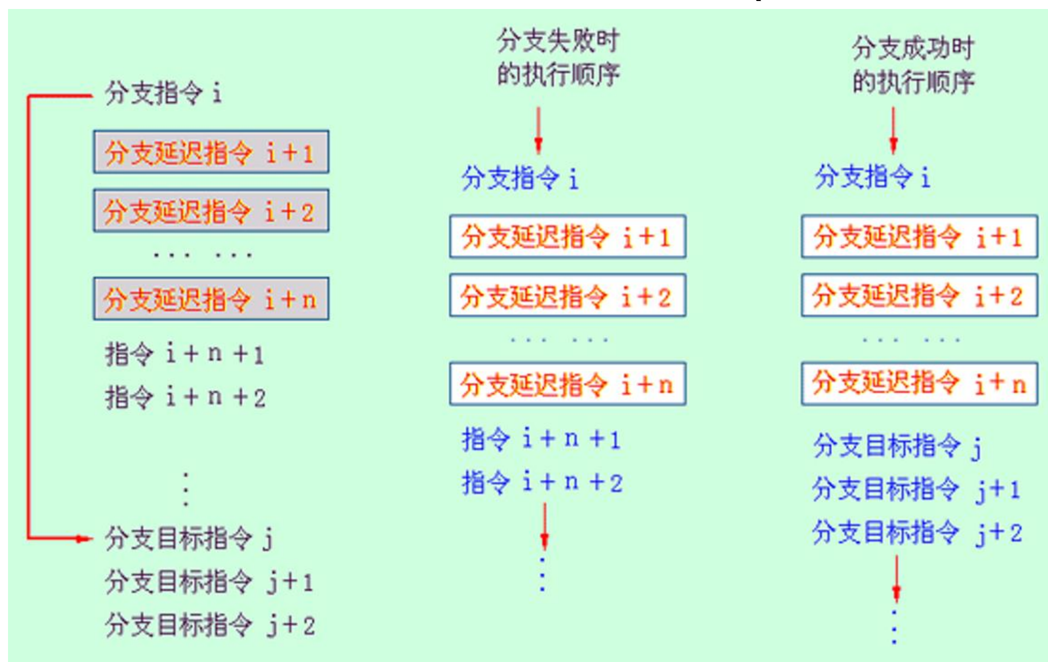
Clock 4

BEQ在ID段结束，分支延迟=1

分支优化技术-降低开销

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

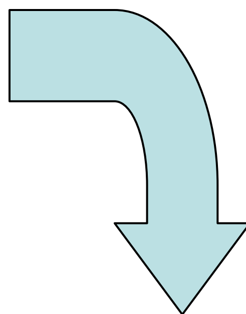
- 延迟分支（delayed branch）： 编译调度
  - ✓ 按一定的模式向延迟槽（delay slot）中填入某些指令
  - ✓ 无论分支是否成功，延迟槽中的指令都将被执行
- 延迟分支的来源
  - ✓ 使用分支前的指令填充（从前调度）
  - ✓ 以分支目标指令填充（从目标处调度）
  - ✓ 以分支不发生时的下一条指令填充（从失败处调度）



减少转移损失—延迟转移，编译器

	.....
I0:	R1+R2→R3
I1:	BRGT R5,R6, jump:
I2:	R4+R7→R8
	.....
	jump:

将代码重新安排如下：

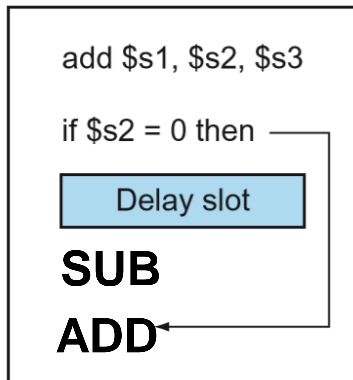


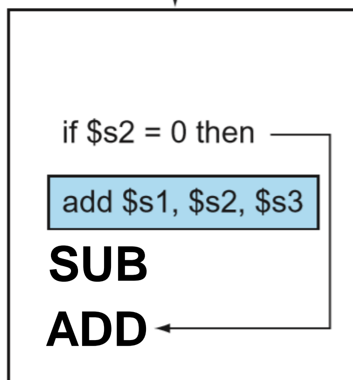
	.....
I1:	BRGT R5,R6, jump:
I0:	R1+R2→R3
I2:	R4+R7→R8
	.....
	jump:

延迟转移是一个体系结构特性。也就是说是软硬之间的一个协议。硬件会执行任何跟在转移指令后面的指令，不管转移是否发生。软件负责调度一条有用指令填入这个槽。若不能找到有用指令就在代码中填入NOP指令。

延迟分支的来源

- ✓ 使用分支前的指令填充（从前调度）





调度前

Taken(分支成功):
1 ADD
2 IF
3 STALL
4 ADD

Not Taken(分支不成功):
1 ADD
2 IF
3 STALL
4 SUB
5 ADD

调度后

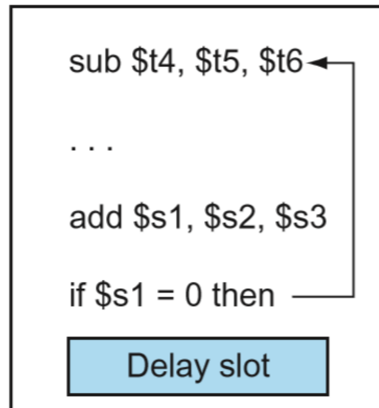
Taken(分支成功):
1 IF
2 ADD
3 ADD

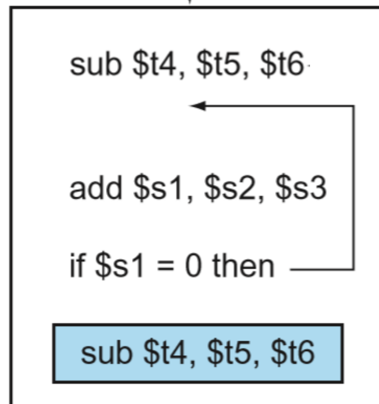
Not Taken(分支不成功):
1 IF
2 ADD
3 SUB
4 ADD

从目标处调度

◻ 延迟分支的来源

✓ 以分支目标指令填充（从目标处调度）





调度前

Taken(分支成功):
1 SUB
2 XXX
3 ADD
4 IF
5 STALL
6 SUB

Not Taken(分支不成功):
1 SUB
2 XXX
3 ADD
4 IF
5 STALL
6 YYY

调度后

Taken(分支成功):
1 SUB
2 XXX
3 ADD
4 IF
5 SUB

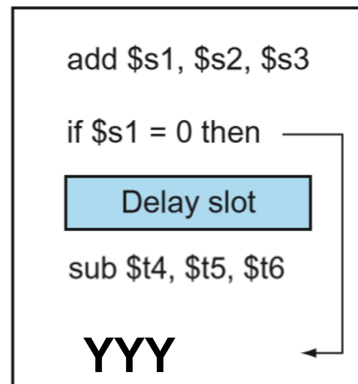
Not Taken(分支不成功):
1 SUB
2 XXX
3 ADD
4 IF
5 SUB
6 YYY

从失败处调度

延迟分支的来源

- ✓ 以分支不发生时的下一条指令填充（从失败处调度）

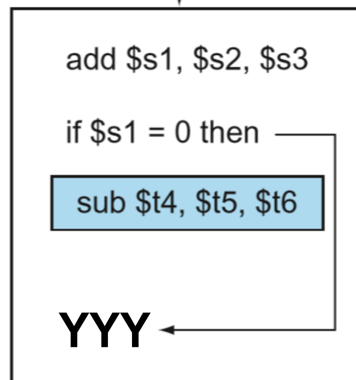
调度前



Taken(分支成功):
1 ADD
2 IF
3 STALL
4 YYY

Not Taken(分支不成功):
1 ADD
2 IF
3 STALL
4 SUB
5 YYY

调度后



Taken(分支成功):
1 ADD
2 IF
3 SUB
4 YYY

Not Taken(分支不成功):
1 ADD
2 IF
3 SUB
4 YYY

三种延迟分支方法的要求及效果

控制相关=>分支=>降低分支开销(硬)=>延迟分支（软)

调度策略	对调度的要求	其作用前提
从前调度	被调度的指令必须与分支结果无关	任何情况
从目标处调度	必须保证在分支失败时执行被调度的指令不会导致错误	分支成功时
从失败处调度	必须保证在分支成功时执行被调度的指令不会导致错误	分支失败时

通过分支成功与否选择合适的调度策略！
问题：如何提前知道分支是否成功？

分支优化技术-分支预测

- 问题：如果能够提前知道分支是否成功，采取相应的措施，可以提高性能。
- 分支预测技术（Prediction）
  - ✓静态预测：投机执行（Speculation）
    - Taken
    - Not Taken
    - 向后转移taken，向前转移not taken
  - ✓动态预测：分支预测器（Branch predictor）
    - 一位饱和计数器（saturating counter）
    - 两位饱和计数器

静态预测，编译器

- 大多数RISC微处理器一般都采用了静态转移预测。3种：
  - ✓ 预测总是不会发生转移（预测分支失败）。
    - 这是最简单最经济的转移预测策略。
    - 在转移实际发生之前，它预测指令执行是顺序的。
  - ✓ 预测转移总是发生（预测分支成功）。
    - 只要转移指令在实际中进行转移的百分比远高于没有发生转移的情况，这种转移预测策略可以很好的工作。
  - ✓ 对于向后的转移预测转移会发生，向前的转移预测转移不会发生。
    - 这种预测策略与循环的结构非常匹配。在循环中，如果向后的转移用于迭代，向前的转移用于退出循环。由于迭代次数很多，而退出只有一次，那么这种预测对循环是非常准确的。

预测成功和预测失败应该采取什么策略？

预测分支失败

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

Program execution order (in instructions)

Time (in clock cycles)

CC 1 CC 2 CC 3 CC 4 CC 5 CC 6 CC 7 CC 8 CC 9

40 beq x1, x3, 72

44 and x12, x2, x5

48 or x13, x6, x2

52 add x14, x2, x2

72 lw x4, 50(x7)

IM Reg DM Reg

动态预测，预测器

- 动态转移预测基于实际运行时的转移历史：
  - ✓ 1位转移预测缓冲器。将所有转移指令地址放入一个缓冲器并配置一个状态位用于记录这条指令是否发生了转移。
    - 当取到一条新的转移指令时，查看缓冲器是否有这条指令。如果这条指令不在缓冲器，静态的预测这条指令并在指令的状态输出已知时更新缓冲器。如果指令已在缓冲器中，根据以前状态预测这条指令。如果预测正确，那是最好的；否则，更新缓冲器中的状态位。
  - ✓ 2位转移预测缓冲器。
    - 转移预测过程与1位时相同，只是用两个状态位跟踪转移历史。
    - 只有当连续两次预测不成功时才修改转移预测的方向。
    - 用两位的目的是在特定条件下，如嵌套循环退出时，实现滞后作用。

以硬件复杂性和成本为代价，动态转移预测可以稍微提高转移预测的精度。

动态分支预测（2位）

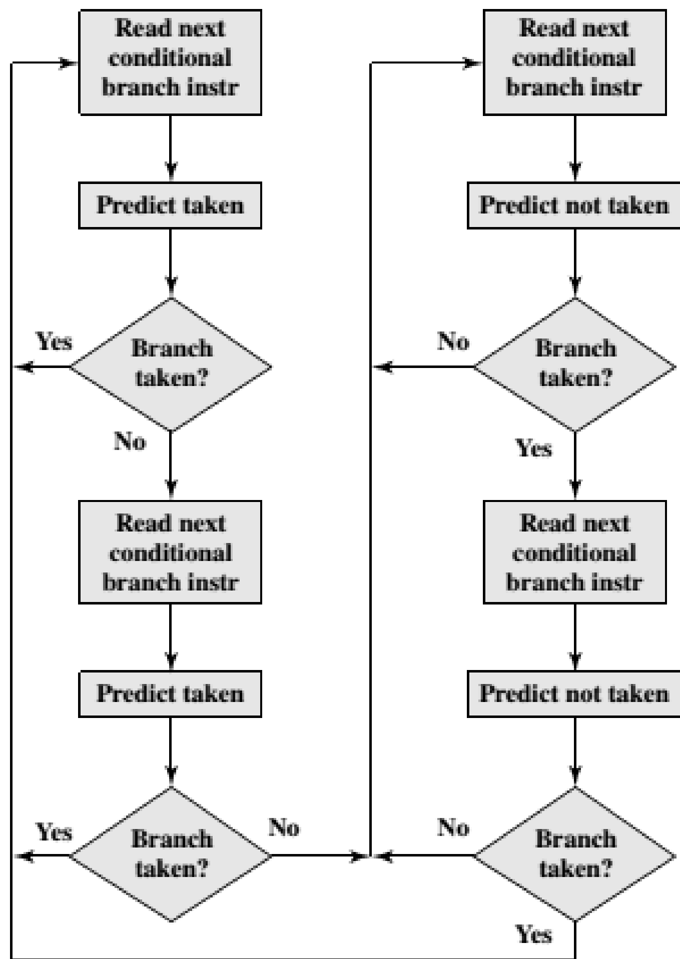
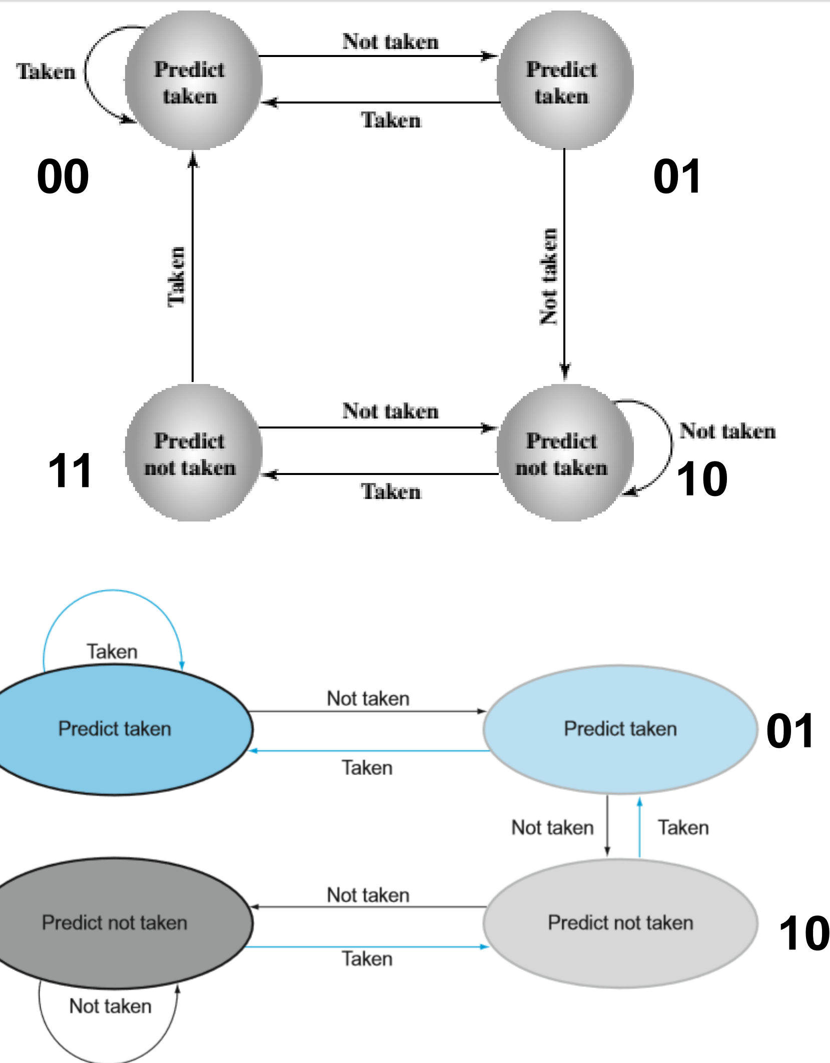


Figure 12.18 Branch Prediction Flowchart



流程图：无时序　　　　2位预测器　　　　状态图：有时序

实际的取指流水线

Next fetch address prediction
Fetch address calculation
Instruction Cache access
Instructions drive to decode stage

Branch Target Buffer (BTB)
Next sequential fetch addr
Branch Predictor
Return Address Stack (RAS)

offset
index
...

Addr. dec.
I-Cache Data Array
Way mux
Aligner
Instruction bytes

Addr. dec.
I-Cache Tag Array
=
Hit/Miss

Addr. dec.
I-TLB Array

Next fetch address from branch misprediction or exception

Cache miss也需要stall

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

除了结构、数据和控制相关需要暂停流水线之外，Cache miss也需要stall

I-Cache miss or D-Cache miss causes pipeline to stall

Interface to L2 Cache or Main Memory

Latency Numbers Every Programmer Should Know

1ns

L1 cache reference: 1ns

Branch mispredict: 3ns

L2 cache reference: 4ns

Mutex lock/unlock: 17ns

100ns =

Main memory reference: 100ns

1,000ns ≈ 1µs

Compress 1KB wth Zippy: 2,000ns ≈ 2µs

10,000ns ≈ 10µs =

Send 2,000 bytes over commodity network: 0.0ns ≈ 0.2µs

SSD random read: 16,000ns ≈ 16µs

Read 1,000,000 bytes sequentially from memory: 9,000ns ≈ 9µs

Round trip in same datacenter: 500,000ns ≈ 500µs

1,000,000ns = 1ms =

Read 1,000,000 bytes sequentially from SSD: 200,000ns

Disk seek: 4,000,000ns ≈ 4ms

Read 1,000,000 bytes sequentially from disk: 2,000,000ns ≈ 2ms

Packet roundtrip CA to Netherlands: 150,000,000ns ≈ 150ms

http://www.eecs.berkeley.edu/~rcs/research/interactive_latency.html

发挥分支预测器的效益

分支规律，预测准确率可达90%以上

```
if (data[c] >= 128)
    sum += data[c];
```

```
data[] = 226, 185, 125, 158, 198, 144, 217, 79, 202, 118,  14, 150, 177, 182, 133, ...
branch =   T,   T,   N,   T,   T,   T,   T,  N,   T,   N,   N,   T,   T,   T,   N  ...

       = TTNTTTTNTNNTTTN ...   (基本上随机出现 - 很难预测)
```

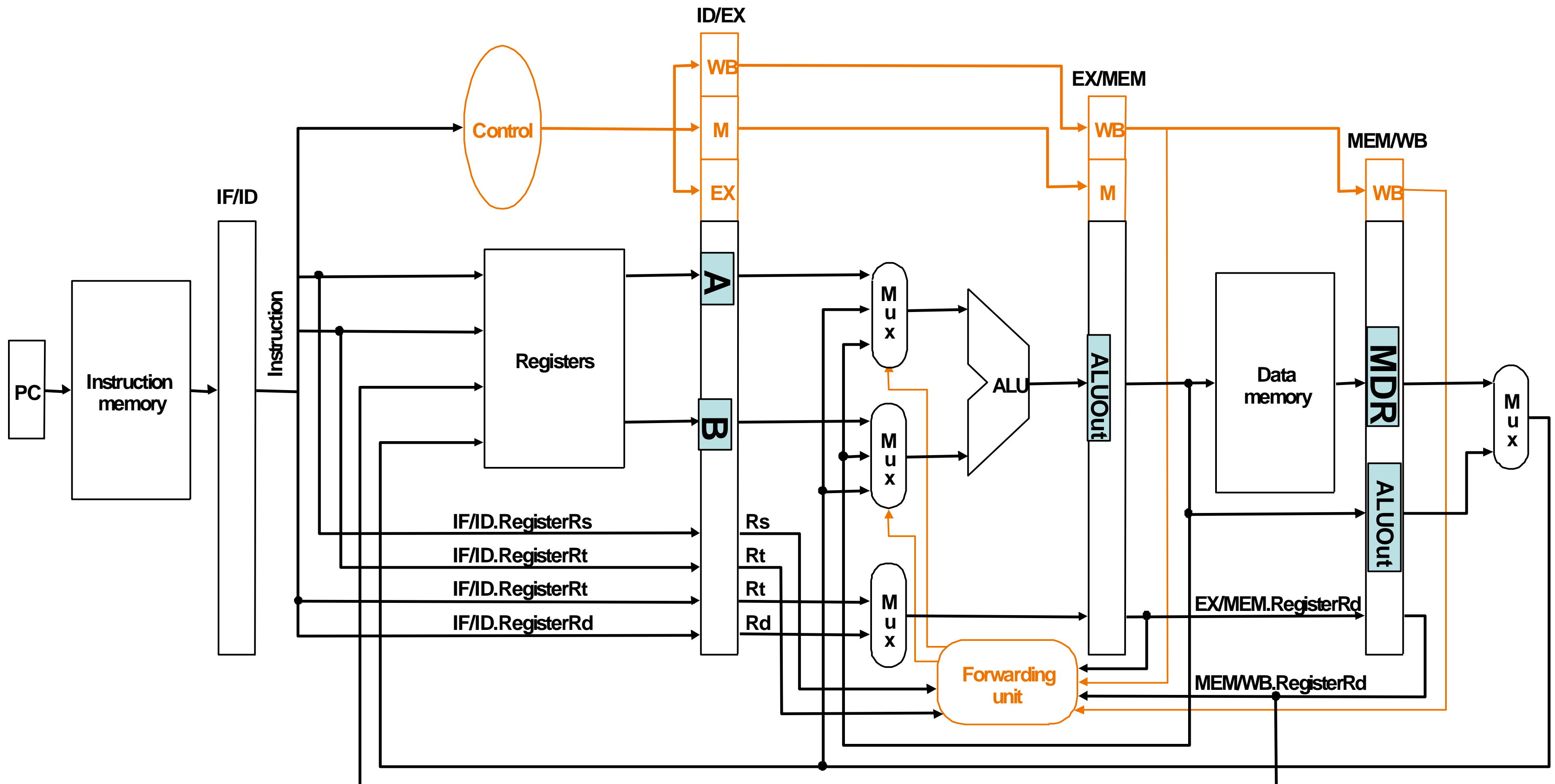
```
T = 该分支被选中
N = 该分支没有被选择

data[] = 0, 1, 2, 3, 4, ... 126, 127, 128, 129, 130, ... 250, 251, 252, ...
branch = N  N  N  N  N  ...   N    N    T    T    T  ...   T    T    T  ...

       = NNNNNNNNNNNN ... NNNNNNNTTTTTTTTT ... TTTTTTTTTT  (很容易进行预测)
```

数据预排序，提升分支预测的有效性

相关总结

- 结构相关
  - ✓ 原因：硬件资源不足产生的冲突（解决**谁来做**的问题）
  - ✓ 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**数在哪**的问题）
  - ✓ 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**做什么**的问题）
  - ✓ 解决方案：等待（软件编译器，硬件Stall）
  - ✓ 解决方案：尽早写PC（算地址、算条件）,降低延迟=1
  - ✓ 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）

练习：画出下边指令序列中的定向路径



SUB x1, x5, x4

ADD x1, x1, x4

BEQ x1, x4, Offset

LW x1, 45(x3)

ADD x1, x1, x4

SW x1, 5(x3)

格式：SUB->ADD: EX/MEM.ALUOUT->ID/EX.A

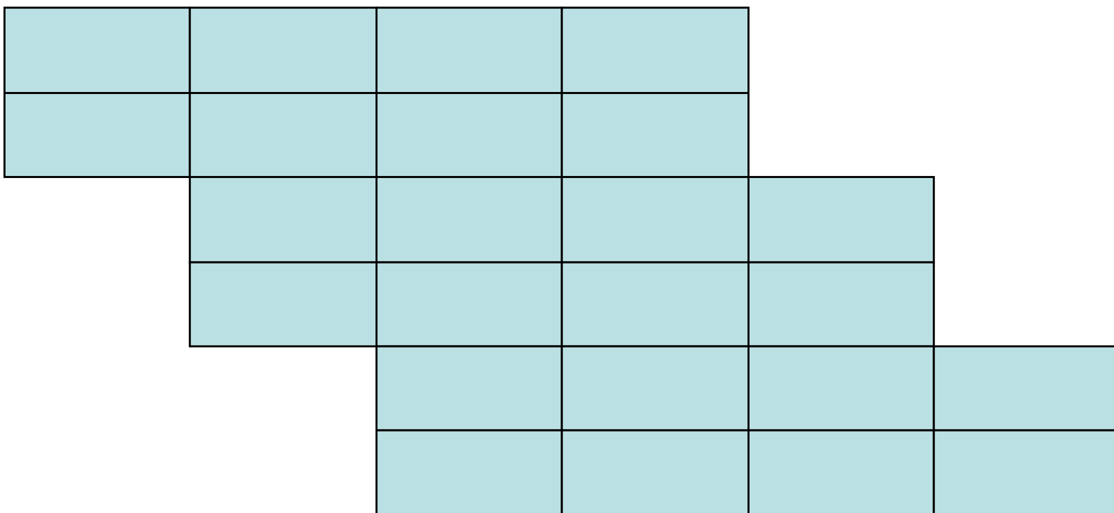
流水线中的多发射技术

流水线中的多发射技术

- 通过减少流水线的停顿提升性能
  - ✓CPI (Cycles per Instruction)=1
  - ✓IPC (Instructions Per Cycle)
- 如何使得IPC>1？
  - ✓在一个时钟周期内流出多条指令
- 常见的多发射技术(Multiple issue)
  - ✓超标量技术（SuperScalar）
  - ✓超长指令字技术（Very Long Instruction Word)
  - ✓超流水线技术（SuperPipeline）
  - ✓SIMD技术（Single Instruction Multi Data）

1、超标量技术（SuperScalar）

- 在任一时刻取指并执行一条指令的简单的处理器被称为“简单标量处理器” SimpleScalar
- 超标量：指在每个时钟周期内可同时发射并执行多条指令
  - ✓ 系统结构要求：处理机中配置多个功能部件和指令译码电路，以及多个寄存器端口和总线，以便能实现同时执行多个操作
- 编译器支持
- 动态指令集调度
  - ✓ 记分牌
  - ✓ Tomasulo



A Two-Way Superscalar Pipeline

Fetch unit

Instruction queue

寄存器端口和总线

Reg	Reg	Reg
Reg	Reg	Reg

译码电路

Dispatch unit

Arithmetic unit

功能部件

Load/Store unit

Write results

2、Superpipeline

□超流水线=深度流水线

	Cycle 1		Cycle 2		Cycle 3		Cycle 4		Cycle 5		Cycle 6		Cycle 7		Cycle 8	
Phase	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
	Fetch	ALU	Mem	Write												
		Fetch	ALU	Mem	Write											
			Fetch	ALU	Mem	Write										
				Fetch	ALU	Mem	Write									
					Fetch	ALU	Mem	Write								
						Fetch	ALU	Mem	Write							
							Fetch	ALU	Mem	Write						
								Fetch	ALU	Mem	Write					

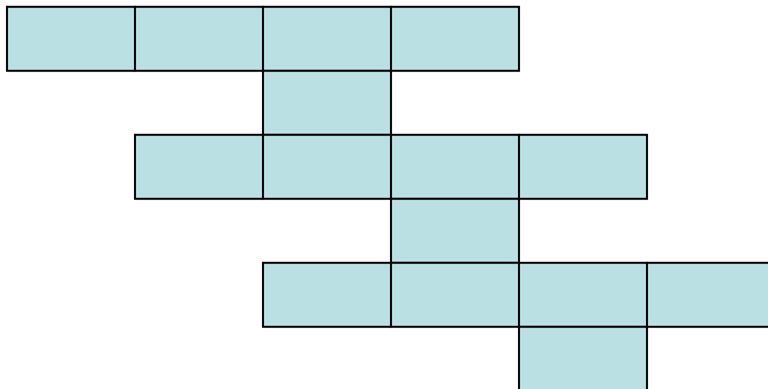
Clock

在一个时钟周期内实现更细的流水段

3、超长指令字技术

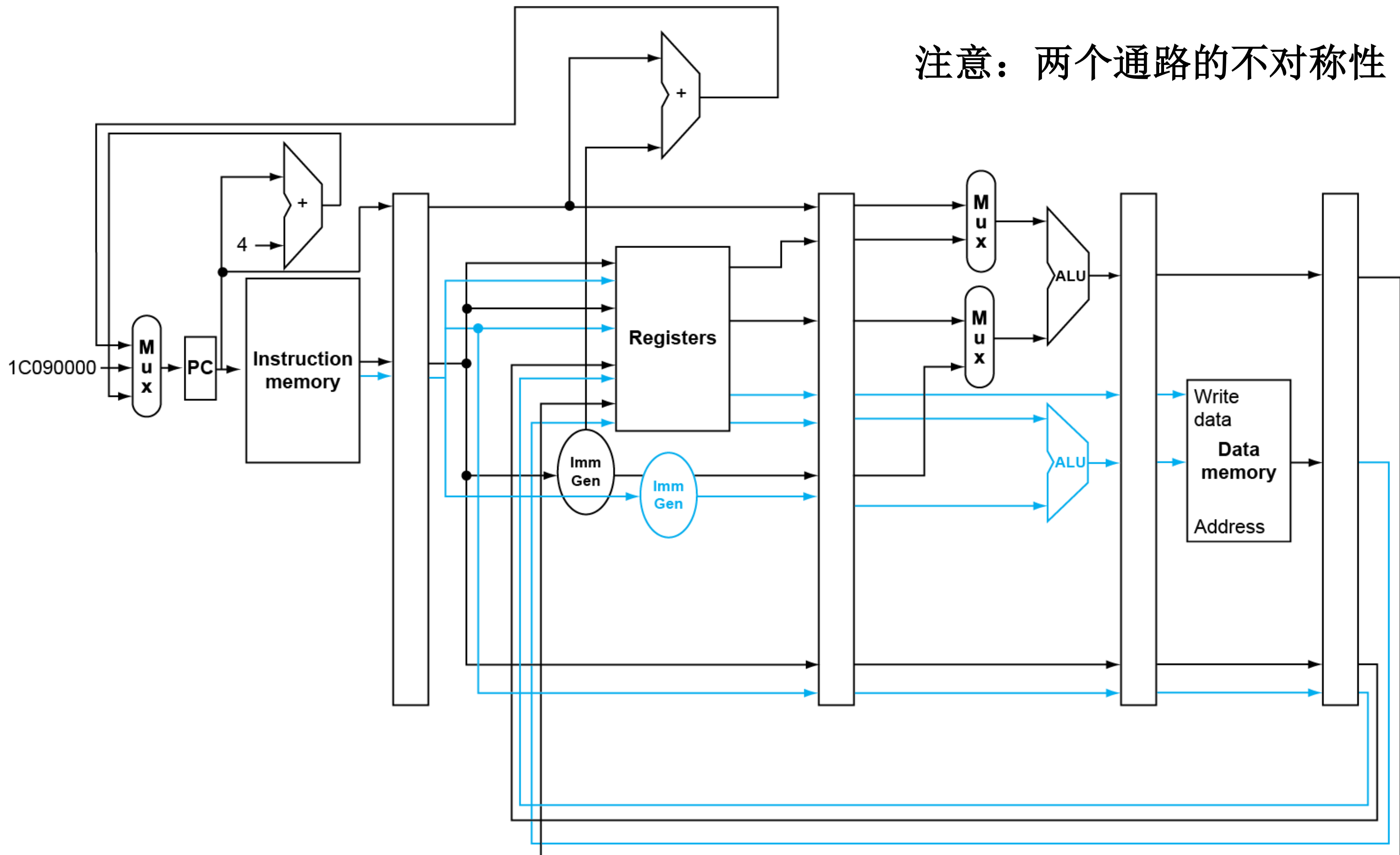
中国科学技术大学 University of Science and Technology of China

- 静态指令集调度
- VLIW把多条能并行操作的指令组合成一条具有多个操作码字段的超长指令(指令字长可达几百位)，由这条超长指令控制VLIW机中多个独立工作的功能部件，由每一个操作码字段控制一个功能部件，相当于同时执行多条指令。
- 超长指令字(VLIW)技术和超标量技术都是采用多条指令在多个处理部件中并行处理的体系结构，在一个时钟周期内能流出多条指令。



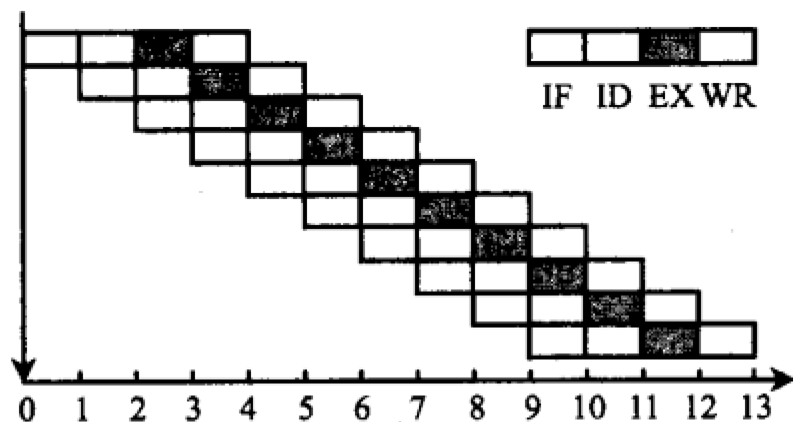
VLIW datapath

注意：两个通路的不对称性

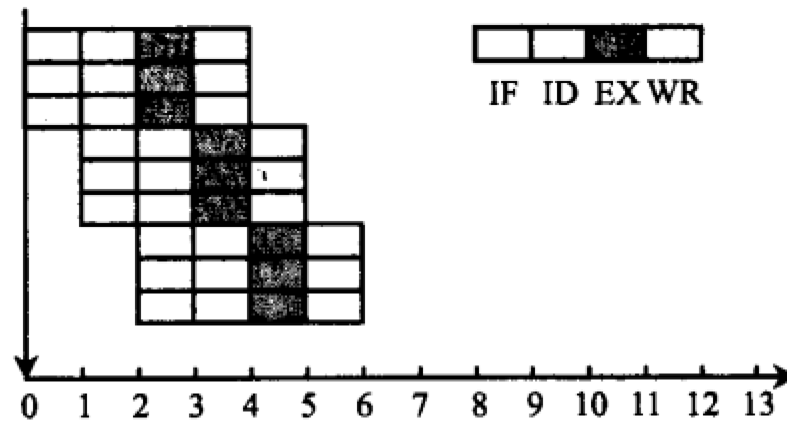


四种流水技术比较

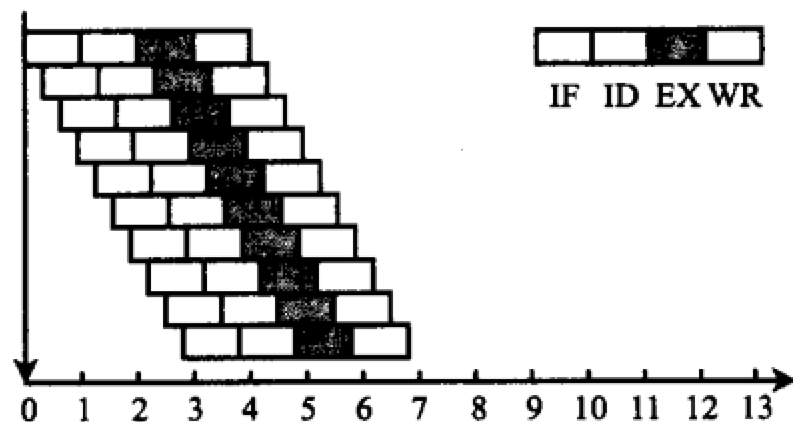
中国科学技术大学
University of Science and Technology of China



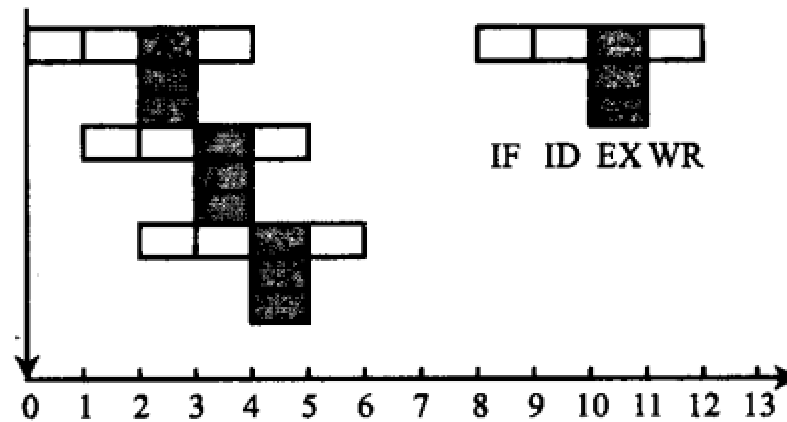
(a) 普通流水



(b) 超标量流水



(c) 超流水线



(d) 超长指令字

Superscalar vs. VLIW

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

Superscalar	VLIW
□指令长度短	□指令长度长
□动态控制	□静态控制
□硬件设计复杂	□译码、发射指令的硬件设计简单
□编译器简单	□编译器复杂
□二进制兼容性好	□二进制兼容性差
□芯片功耗高	□芯片功耗低
□芯片面积大	□芯片面积小
□芯片成本高	□芯片成本低
□需要多的寄存器	□需要更多的寄存器

动态指令集调度

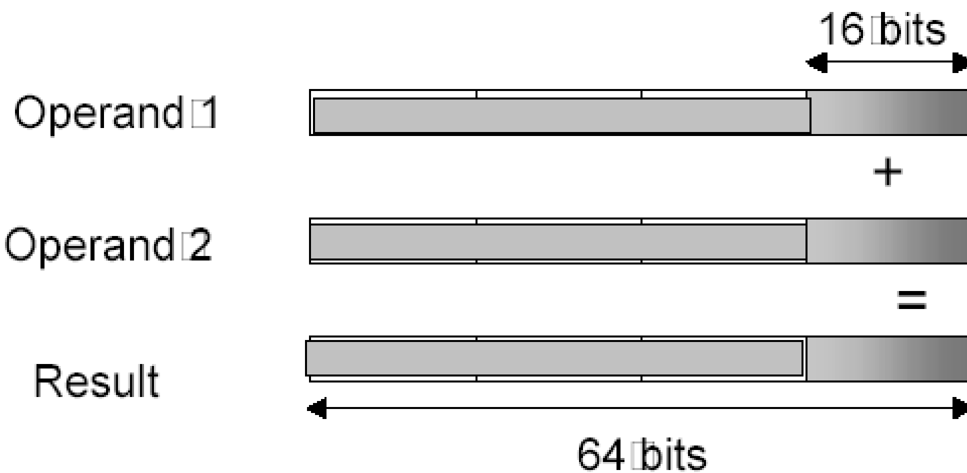
Pentium Pro Pentium II,III,IV, AMD Athlon, MIPS R10K R12K, Sun UltraSpac, PowerPC 603,G3,G4,G5,Alpha 21264

静态调度

Itanium

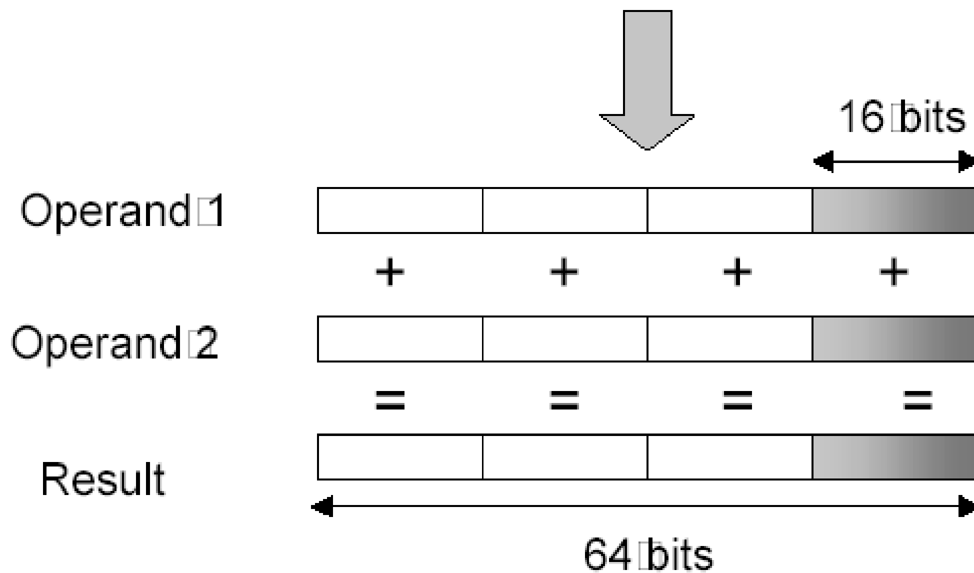
Transmeta: Crusoe

指令级并行V.S.数据级并行
单指令多数据SIMD



48 bits are wasted!

Can we use them in any way?



4 operations in 1 cycle
SPEEDUP: 4X??

Called SIMD:
single-instruction multiple-data parallelism

SIMD Instructions

MD-technique

- Multiple data operands per operation
- 奔腾MMX、 SSE、 GPU、DSP、银河、天河、神威

Vector instruction:

```
for (i=0, i++, i<64)
   c[i] = a[i] + 5*b[i];

c = a + 5*b
```

Assembly:

```
set      v1,64
ldv      v1,0(r2)
mulvi    v2,v1,5
ldv      v1,0(r1)
addv     v3,v1,v2
stv      v3,0(r3)
```

问题与总结

单周期、多周期、流水线

Cycle 1 | Cycle 2

Clk

Single Cycle Implementation:

Load	Store	Waste

Cycle 1 | Cycle 2 | Cycle 3 | Cycle 4 | Cycle 5 | Cycle 6 | Cycle 7 | Cycle 8 | Cycle 9 | Cycle 10

Clk

Multiple Cycle Implementation:

Load					Store				R-type
Ifetch	Reg	Exec	Mem	Wr	Ifetch	Reg	Exec	Mem	Ifetch

Pipeline Implementation:

Load	Ifetch	Reg	Exec	Mem	Wr	
Store		Ifetch	Reg	Exec	Mem	
R-type			Ifetch	Reg	Exec	Mem	Wr

注意：
多周期和流水线指令周期都不定长，但流水线每个周期“流出”一条指令。

练习：单周期 vs. 多周期 vs.流水线

- 例：假设部件延迟MEM=2ns， R-Type ALU=2ns，Reg=1ns，其他部件没有延时。某程序编译生成100条指令，其中LW 20%,SW 20%,R-Type ALU 40%,BEQ 20%。采用MIPS/RISC-V的5级流水线，并假设LW互锁和分支延迟如下，其他无开销。
  - ✓ 有一半的LW执行后为R-Type ALU，因此需要互锁1个周期；
  - ✓ 分支延迟为1个周期，采用等待策略；

  求这段代码在单周期、多周期、流水线三种方式的性能对比。
- 思路：
  - ✓ 单周期：每个时钟周期=?每条指令需要多少周期?总共需要多少周期?
  - ✓ 多周期：每个时钟周期=?每条指令需要多少周期?总共需要多少周期？ LW=？ SW=？ R-Type =？ BEQ=?
  - ✓ 流水线：每个时钟周期=? 无开销总共需要多少周期? LW互锁需要加多少？ 分支延迟需要加多少?

- 解答：设MEM=2ns，ALU和加法器=2ns，Reg=1ns，其他部件没有延时。
  - ✓ 单周期：周期为所有操作时间之和，8ns
    - 执行100条指令所需的时间为8ns×100=800ns
  - ✓ 多周期：周期按操作时间最长的阶段定义，2ns
    - LW-5个周期=10ns　SW-4个周期=8ns
    - ALU-4个周期=8ns　分支-3个周期=6ns
    - 执行100条=10*20+8*20+8*40+6*20=800ns
  - ✓ 流水线：周期按操作时间最长的阶段定义，2ns
    - 执行100条=10+99*2=208ns，加性能损失2ns*10+2ns*20=268ns
  - ✓ 性能对比
    - 多周期=单周期=800ns
    - 流水线性能=268ns，加速比=2.99

小结与思考（一）

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

- CPU设计方法：ISA，数据通路，控制信号，状态机，测试与评估（功能、性能）
- ISA的实现模式
  - ✓ 单周期：指令周期（定长）由一个机器周期组成。
  - ✓ 多周期：指令周期（变长）由多个机器周期组成。每个功能部件只在特定机器周期执行。
  - ✓ 流水线：指令周期（变长）由多个机器周期（流水级）组成，每个功能部件完成不同指令的相同阶段的执行。
- 思考：
  - ✓ 单周期和流水线的关系？
  - ✓ 流水线中实现一个周期访存（取指，取数）的前提条件是什么？
  - ✓ 流水线执行第一条指令时，IF在取指，其他段在做什么？
  - ✓ 各流水段的控制信号何时生成与释放？
  - ✓ 寻址方式如何实现的？哪些寻址方式适合流水线？
  - ✓ 如果R-type指令也可访存，则应如何设计流水线？
  - ✓ MIPS能否采用“取指、译码、执行”三段流水线？
  - ✓ Stall原因与判断规则？如何实现stall？

小结与思考（二）

思考

- ✓理想流水线加速比=？IPC=?
- ✓为何单周期、多周期的控制信号不需要buffer，而流水线的控制信号需要buffer?
- ✓哪些情形可能造成流水线stall，有哪些解决方案?
- ✓影响流水线性能发挥的因素有哪些，可以采用哪些手段减小这些因素的影响?
- ✓指令流水线中在哪个阶段会产生什么相关?
- ✓为何MIPS只有I-type指令访存/RISC-V的I类S类
- ✓典型的流水线的多发射技术有哪些?

小结与思考（二）

- 习题：
  - ✓ COD5: 4.16.1~3
  - ✓ COD5: 4.23
- 思考题： （选做，不交）
  - ✓ --影响流水线性能发挥的因素有哪些?
  - ✓ --为何RV只有Load/store指令访存
  - ✓ --单周期、多周期的控制信号无须buffer，而流水线的控制信号需要buffer的原因
  - ✓ --流水线控制器的实现方式

联系方式

Acknowledgements:
This slides contains materials from following lectures:
- **Computer Architecture (NUDT)**
- **CS 152 CS 252 (UC Berkeley)**

Research Area：
- 基于分布式系统, GPU，FPGA的神经网络、图计算加速
- 人工智能和深度学习（寒武纪）芯片及智能计算机

Contact：
- 中国科学技术大学计算机学院
- 嵌入式系统实验室（西活北一楼）
- 高效能智能计算实验室 （中科大苏州研究院）

cswang@ustc.edu.cn
http://staff.ustc.edu.cn/~cswang