清華大學
Tsinghua University

第二单元　第 二 讲

指令系统举例
控制器设计概述

刘卫东

计算机科学与技术系

本讲概要

清華大學
Tsinghua University

- Pentium指令系统简介
- MIPS指令系统简介
- MIPS指令系统典型指令
  - 数据通路设计
- 控制器设计
  - 时序设计

计算机的指令

- 指令
  - 指令是计算机运行的最小的功能单元，是指挥计算机硬件运行的命令，是由多个二进制位组成的位串，是计算机硬件可以直接识别和执行的一个信息体。
- 指令系统
  - 一台计算机能支持的全部指令
- 指令格式
  - 指令操作码
  - 操作数地址
- 寻址方式
  - 获得下一条指令的地址
  - 获得操作数地址

指令系统设计的两种思路

清華大學 Tsinghua University

依据硬件系统构成的复杂程度、指令执行的效率和并行性来划分，指令系统设计和实现有两种思路：

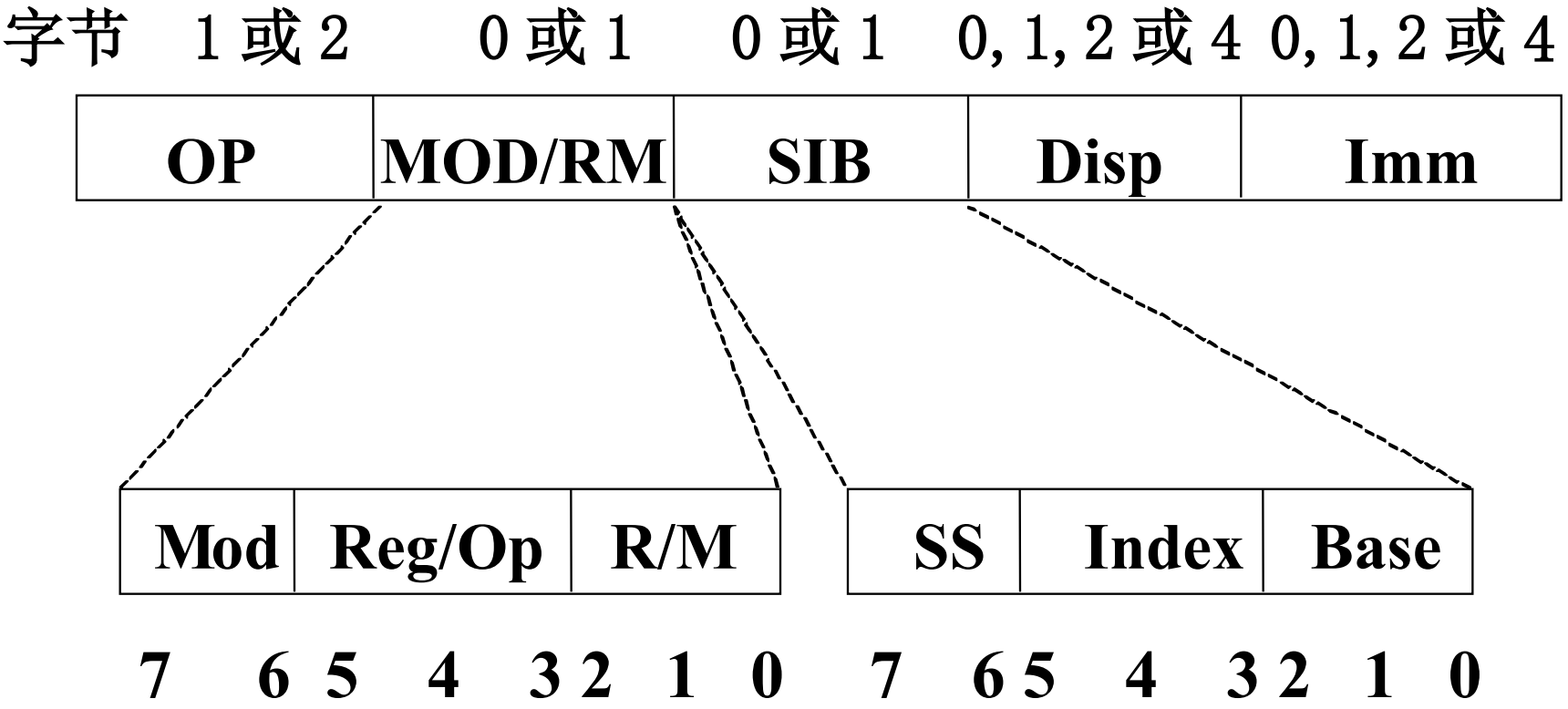
- **RISC（Reduced Instruction Set Computer）**

  通常称为精简指令系统的计算机。提供数目较少、格式与功能简单、运行高效的指令，追求的是计算机控制器实现简单，运行高速，更容易在单块超大规模集成电路的芯片内制做出来。指令并行性好。

- **CISC（Complex Instruction Set Computer）**

  通常称为复杂指令系统的计算机，是相对于RISC一词提出来的。其特点是：指令条数多，格式多样，寻址方式复杂，每条指令的功能强，优点是汇编程序设计容易些，但计算机控制器的实现困难多，很多指令被使用的机会比较少。指令并行度差。

Pentium指令格式



Pentium指令格式

- OP：指令操作码，某些指令在操作码中还包含有操作数长度或 立即数是否需扩充符号位 (S) 等信息；
- MOD／RM：与下一字节提供寻址信息。MOD／RM 指出操作数在寄存器中还是在存储器中。该字节分成3个字段：Mod字段与R／M字段可以产生32个编码，分别表示 8个寄存器和24种变址方法；Reg／op字段可以是寄存器号，或者作为 3位附加的操作码；R／M 字段是 1个操作数所在寄存器或者与Mod字段一起指出寻址方式。
- SIB：当MOD／RM为某些值时，需 SIB 参与决定寻址方式。SIB分成 3个字段， SS 指出变址寄存器的放大因子；Index 指出变址寄存器； Base 指出基址寄存器。
- Disp：当寻址方式指示用到 disp(位移量)时，存在 8位，16位或 32位的位移量 3种情况。
- Imm：对寻址方式是立即数，分为 8位, 16位或 32位 3 种。
- 前缀：指令前缀、段前缀、操作数前缀、地址长度前缀

Pentium寻址方式

1. 立即数
2. 寄存器
3. 直接　　E = Disp
4. 基址　　E = (B)　　B基址寄存器
5. 基址+偏移量　　E = (B)+Disp
6. 比例变址+偏移量　　E = (I)*S+Disp

   I变址寄存器，S为比例因子（1、2、4、8）
7. 基址+变址+偏移　　E = (B)+(I)+Disp
8. 基址+比例变址+偏移　　E = (B)+(I)*S+Disp
9. 相对　　指令地址 = (PC)+Disp

数据移动指令

MOV DST, SRC	数据从 SRC复制到 DST
PUSH SRC	把 SRC压入堆栈
POP DST	从堆栈弹出一字存入 DST
XCHG DS1, DS2	交换 DS1和 DS2
LEA DST, SRC	把SRC的有效地址存入DST
CMOV DST, SRC	条件复制

算术指令

ADD DST, SRC	把SRC加到DST中
SUB DST, SRC	从SRC中减去DST
MUL SRC	EAX乘以SRC（无符号）
IMUL SRC	EAX乘以SRC（带符号）
DIV SRC	EDX：EAX除以SRC（无符号）
IDIV SRC	EDX：EAX除以SRC（带符号）
ADC DST, SRC	把SRC加到DST中并加进位位
SBB DST,SRC	从SRC中减去DST和进位位
INC DST	DST加1
DES DST	DST减1
NEG DST	DST取反（也就是0－DST）

逻辑指令

AND DST, SRC	SRC 和 DST 进行逻辑与，结果存入 DST
OR DST, SRC	SRC 和 DST进行逻辑或，结果存入 DST
XOR DST, SRC	SRC 和 DST 进行逻辑异或，结果存入 DST
NOT DST	把 DST替换成二进制反码

二—十进制数指令

DAA	十进制调整
DAS	为减法进行十进制调整
AAA	为加法进行ASCII调整
AAS	为减法进行ASCII调整
AAM	为乘法进行ASCII调整
AAD	为除法进行ASCII调整

移位/循环移位指令

SAL/SAR DST, #	DST左移或者右移#位
SHL/SHR DST, #	DST逻辑左移或者右移#位
ROL/ROR DST, #	DST循环左移或者右移#位
RCL/RCR DST, #	通过进位位对DST移位#位

测试/比较和转移指令

TST SRC1, SRC2	逻辑与，根据结果设置标志位
CMP SRC1, SRC2	依SRC1-SRC2的结果设置标志位
JMP ADDR	跳转到ADDR
Jxx ADDR	基于标志执行条件转移
CALL ADDR	调用ADDR处的过程
RET	从过程返回
IRET	从中断返回
LOOPxx	循环直到条件满足
INT ADDR	初始化一个软件中断
INTO	如果溢出位被设置则发生中断

字符串指令

清華大學
Tsinghua University

LODS	读取一个串
STOS	保存串
MOVS	复制串
CMPS	比较两个串
SCAS	扫描一个串

条件码指令

STC	设置EFLAGS寄存器中的进位位
CLC	清除EFLAGS寄存器中的进位位
CMC	EFLAGS中的补码进位位
STD	设置EFLAGS寄存器中的方向位
CLD	清除EFLAGS寄存器中的方向位
STI	设置EFLAGS寄存器中的中断位
CLI	清除EFLAGS寄存器中的中断位
PUSHFD	EFLAGS寄存器入栈
POPFD	EFLAGS寄存器出栈
LAHF	从EFLAGS寄存器中读取AH
SAHF	把AH保存到EFLAGS寄存器中

其他指令

清華大學
Tsinghua University

SWAP DST	改变DST的字节顺序
CWQ	为了进行除法，把EAX扩展成EDX：EAX
CWDE	把AX中的16位数扩展成EAX
ENTER SIZE, LV	创建SIZE个字节的堆栈段
LEAVE	撤销ENTER创建的堆栈段
NOP	空操作
HLT	停机指令
IN AL, PORT	从PORT端口向AL输入一个字节
OUT PORT, AL	从AL向PORT端口输出一个字节
WAIT	等待中断

Pentium指令系统特点

- 指令格式复杂
- 寻址方式多样
- 通用寄存器较少
- 兼容直到 8086
- 编译系统复杂
- 指令流水实现复杂
- 典型 CISC 指令集

MIPS指令格式

	6 位	5 位	5 位	5 位	5 位	6 位
寄存器型	op	rs	rt	rd	shamt	funct
立即数型	op	rs	rt	address/immediate		
转移型	op	target				

所有的指令都是32位宽度。指令格式共 3种，即立即数型、转移型和寄存器型。

操作数寻址方式有基址加16位位移量的访存寻址、立即数寻址及寄存器寻址 3种。

MIPS算术指令

指令举例	操作	说明
add $1,$2,$3	$1=$2+$3	寄存器加法
sub $1,$2,$3	$1=$2-$3	寄存器减法
addi $1,$2,100	$1=$2+100	立即数加法
addu $1,$2,$3	$1=$2+$3	无符号数加法
subu $1,$2,$3	$1=$2-$3	无符号数减法
addiu $1,$2,100	$1=$2+100	无符号立即数加法
mfco $1,$epc	$1=$epc	读取异常 PC
mult $2,$3	Hi,Lo=$2×$3	乘法
multu $2,$3	Hi,Lo=$2×$3	无符号数乘法
div $2,$3	Lo=$2/$3, Hi=$2 mod $3	除法
divu $2,$3	Lo=$2/$3, Hi=$2 mod $3	无符号数除法
mfhi $1	$1=Hi	从 Hi 中取数据
mflo $1	$1=lo	从 lo 中取数据

MIPS逻辑指令

清華大學 Tsinghua University

指令举例	操作	说明
and $1,$2,$3	$1=$2&$3	与操作
or $1,$2,$3	$1=$2 | $3	或操作
andi $1,$2,100	$1=$2&100	立即数与操作
ori $1,$2,100	$1=$2 | 100	立即数或操作
sll $1,$2,10	$1=$2<<10	逻辑左移
srl $1,$2,10	$1=$2>>10	逻辑右移

MIPS数据传递指令

指令举例	操作	说明
lw $1,100($2)	$1=M[$2+100]	装入字
sw $1,100($2)	M[$2+100]=$1	存储字
lui $1,100	$1=100×$2^{16}$	装入立即数到高位

MIPS条件转移指令

清華大學 Tsinghua University

指令举例	操作	说明
beq $1,$2,100	if($1==$2) go to PC+4+100	相等时转移
bne $1,$2,100	if($1!=$2) go to PC+4+100	不相等时转移
slt $1,$2,$3	if($2<$3) $1=1;else $1=0	小于时置位
slti $1,$2,100	if($2<100) $1=1;else $1=0	小于立即数时置位
sltu $1,$2,$3	if($2<$3) $1=1;else $1=0	小于无符号数时置位
sltiu $1,$2,100	if($2<100) $1=1;else $1=0	无符号数小于立即数时置位

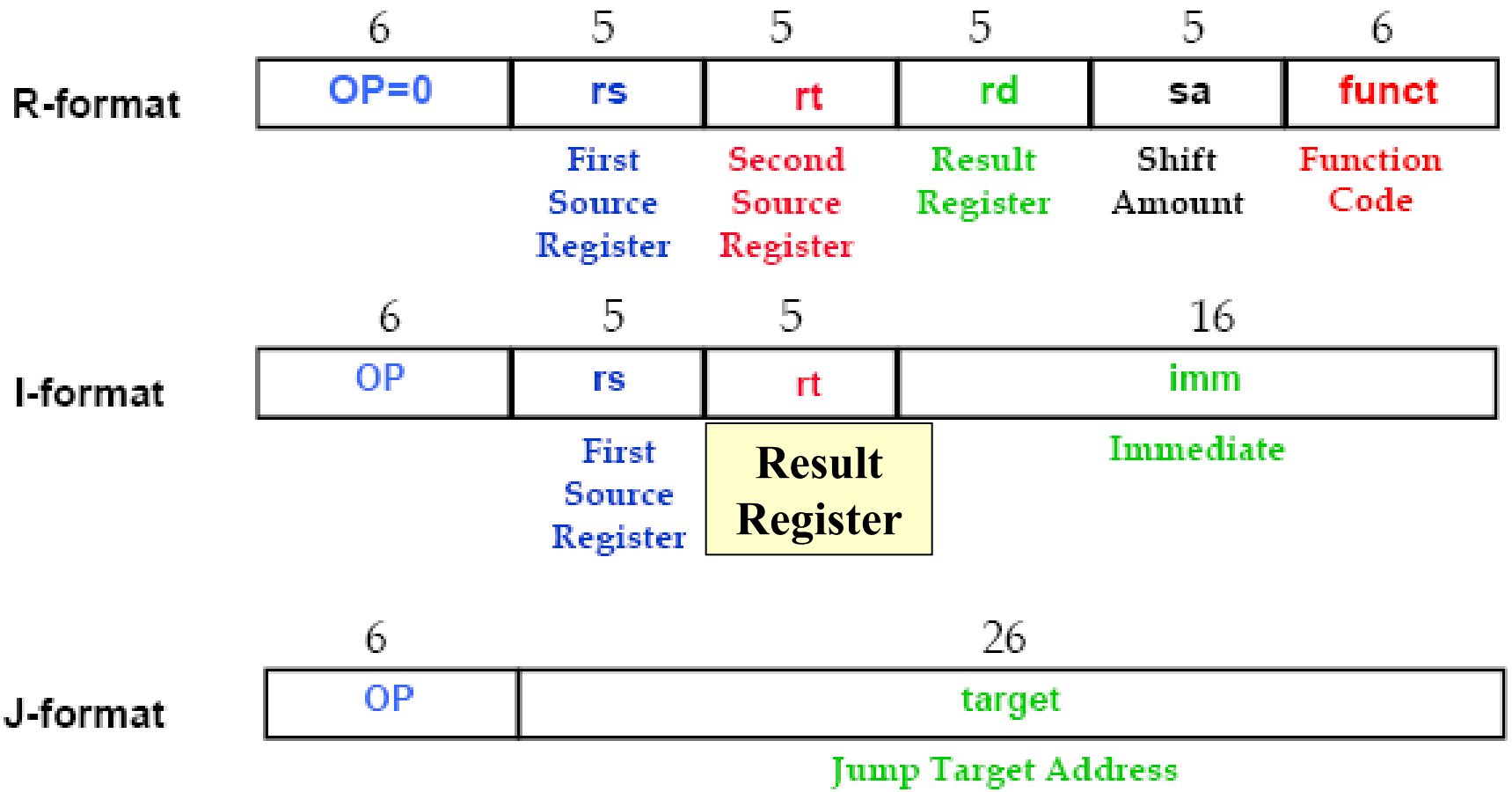
MIPS无条件转移指令

指令举例	操作	说明
j 10000	go to 10000	转移到 10000
jr $31	go to $31	转移到$31
jal 10000	$31=PC+4; go to 10000	转移并链接

MIPS指令系统特点

- 指令格式简单
- 寻址方式较少
- 通用寄存器较多
- 指令数量较少
- 编译系统简单高效
- 容易实现流水操作
- **典型的 RISC 指令集**

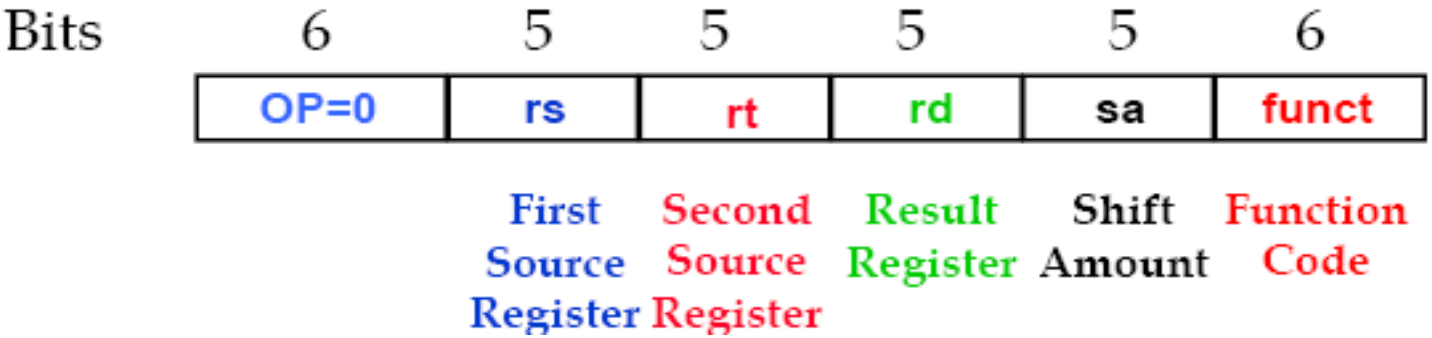
MIPS指令格式



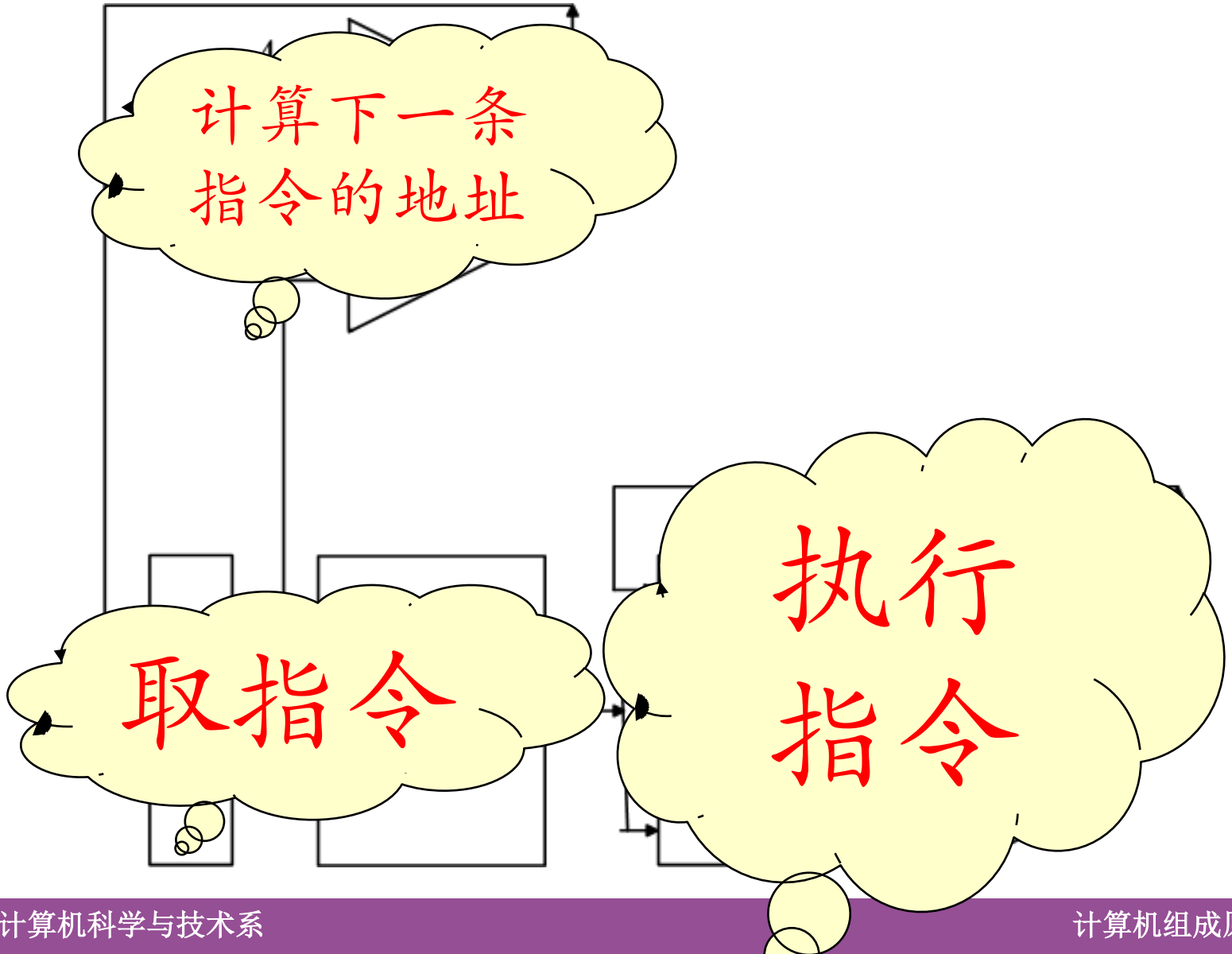
选择典型指令来说明设计MIPS CPU datapath的过程

MIPS指令——addu\subu

- addu\subu
  - addu rd rs rt
  - subu rd rs rt
- 指令功能
  - R[rd] = R[rs] + R[rt]
  - R[rd] = R[rs] − R[rt]



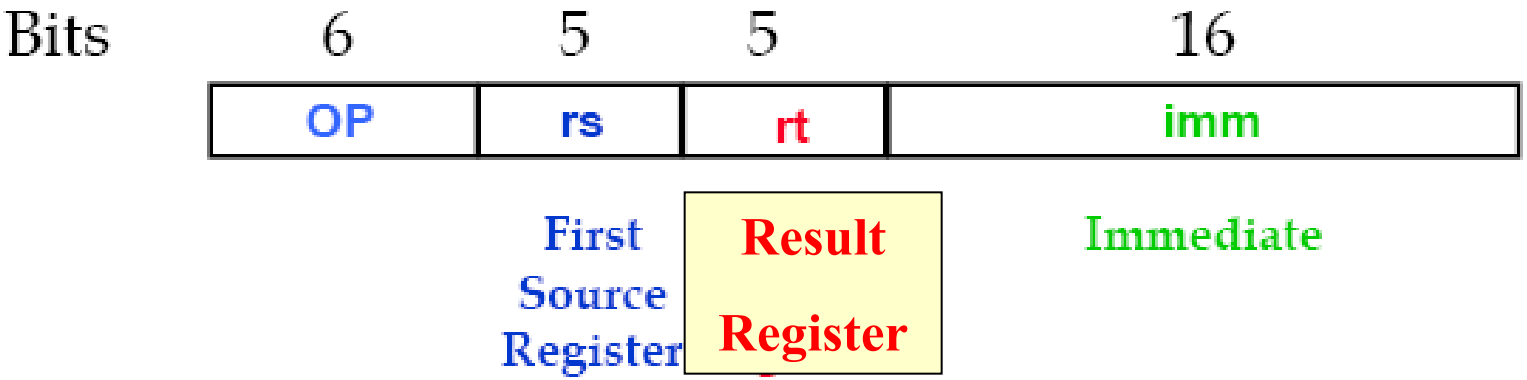
数据通路设计1



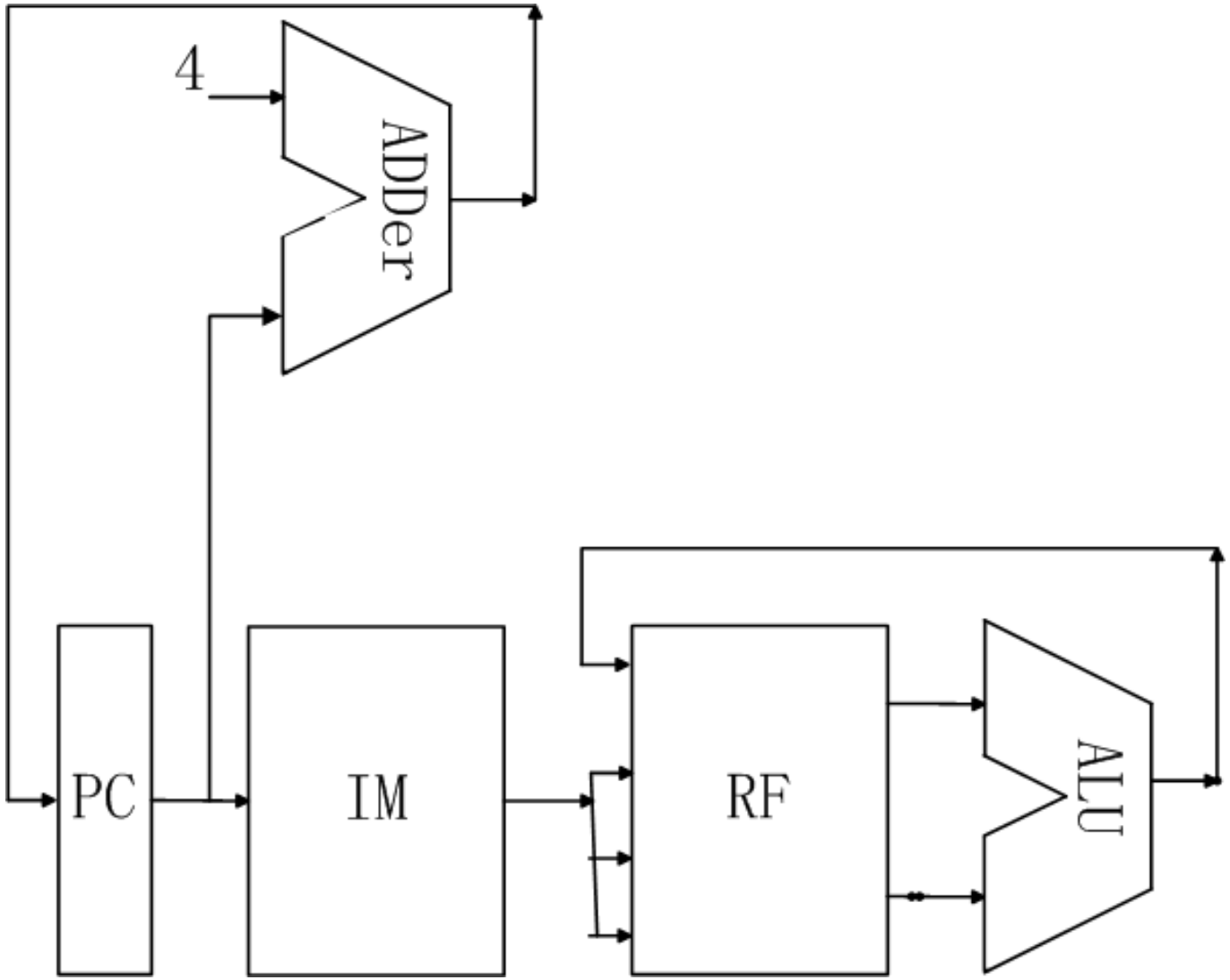
MIPS指令——ori

- ori
  - ori rt rs imm

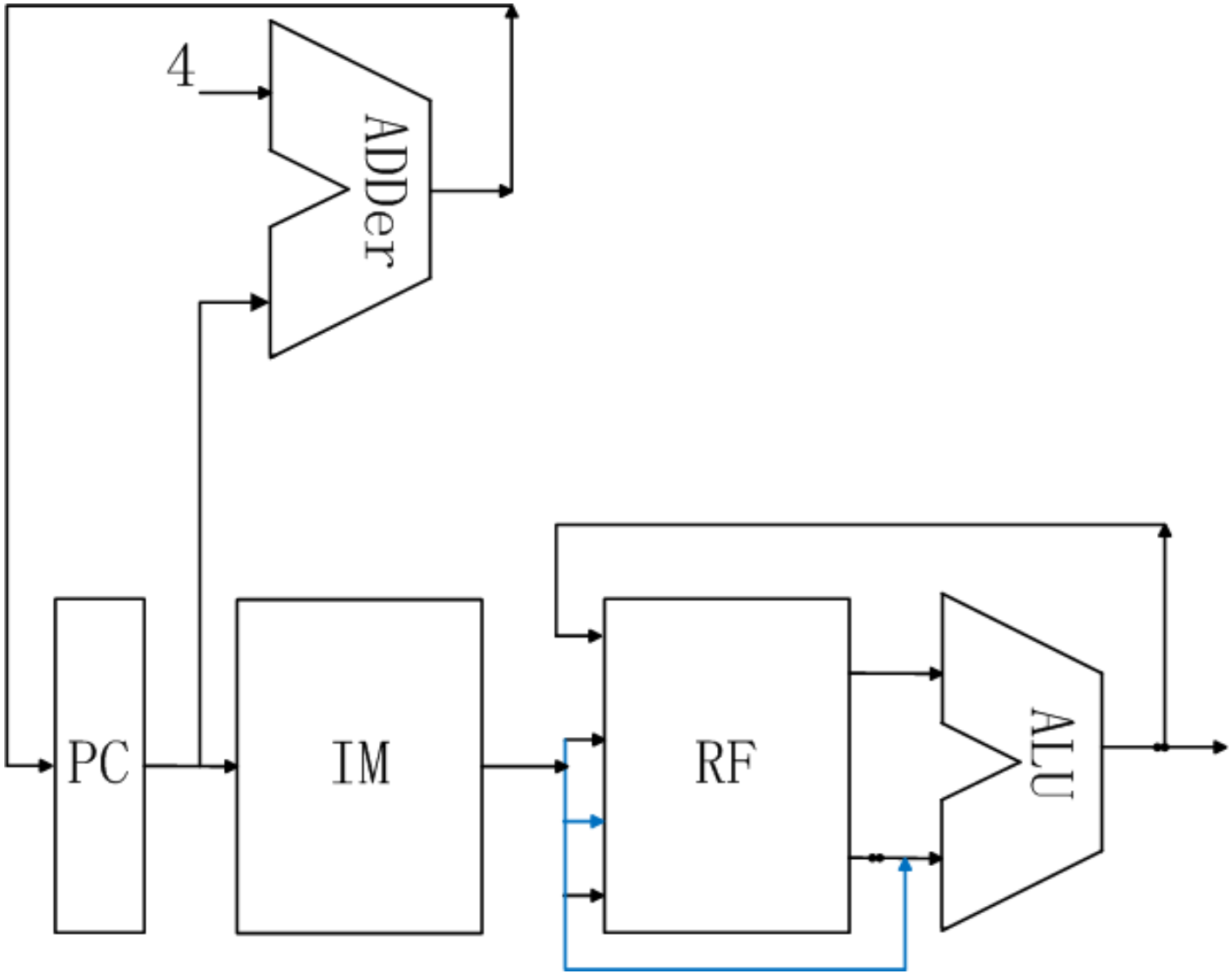
- 指令功能
  - R[rt] = R[rs] OR ZeroExt(imm)



数据通路设计1

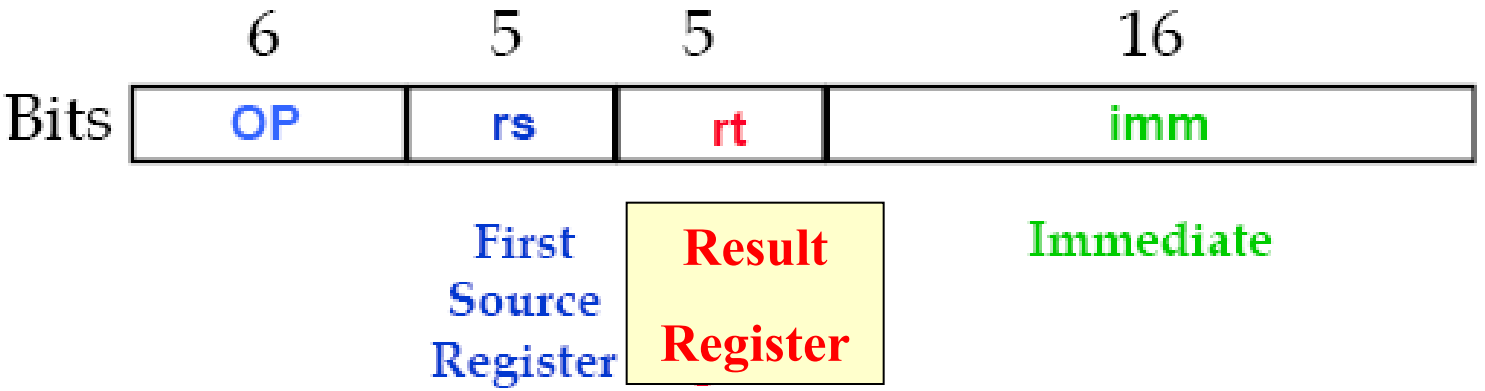


数据通路设计2

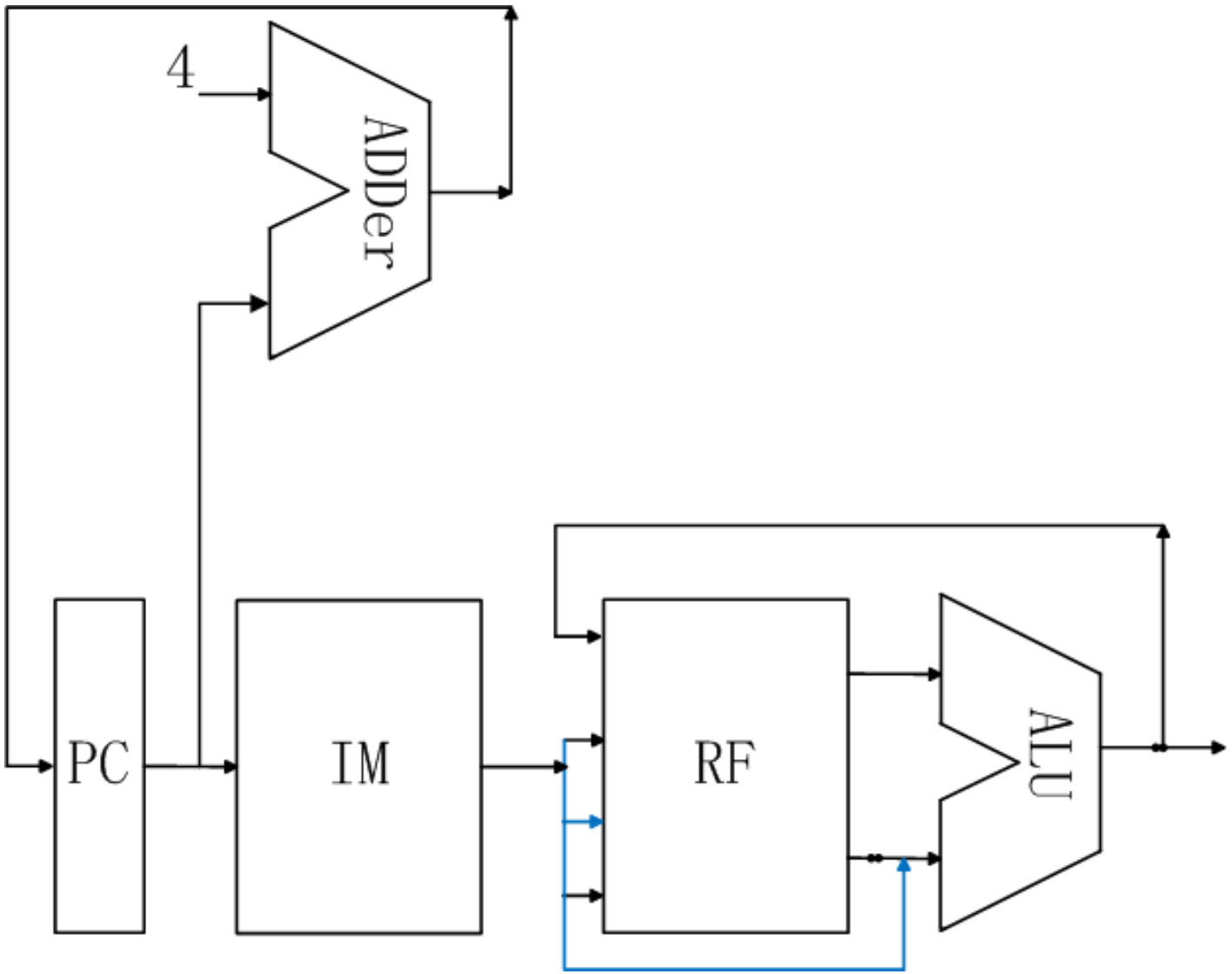


MIPS指令—Load/Store

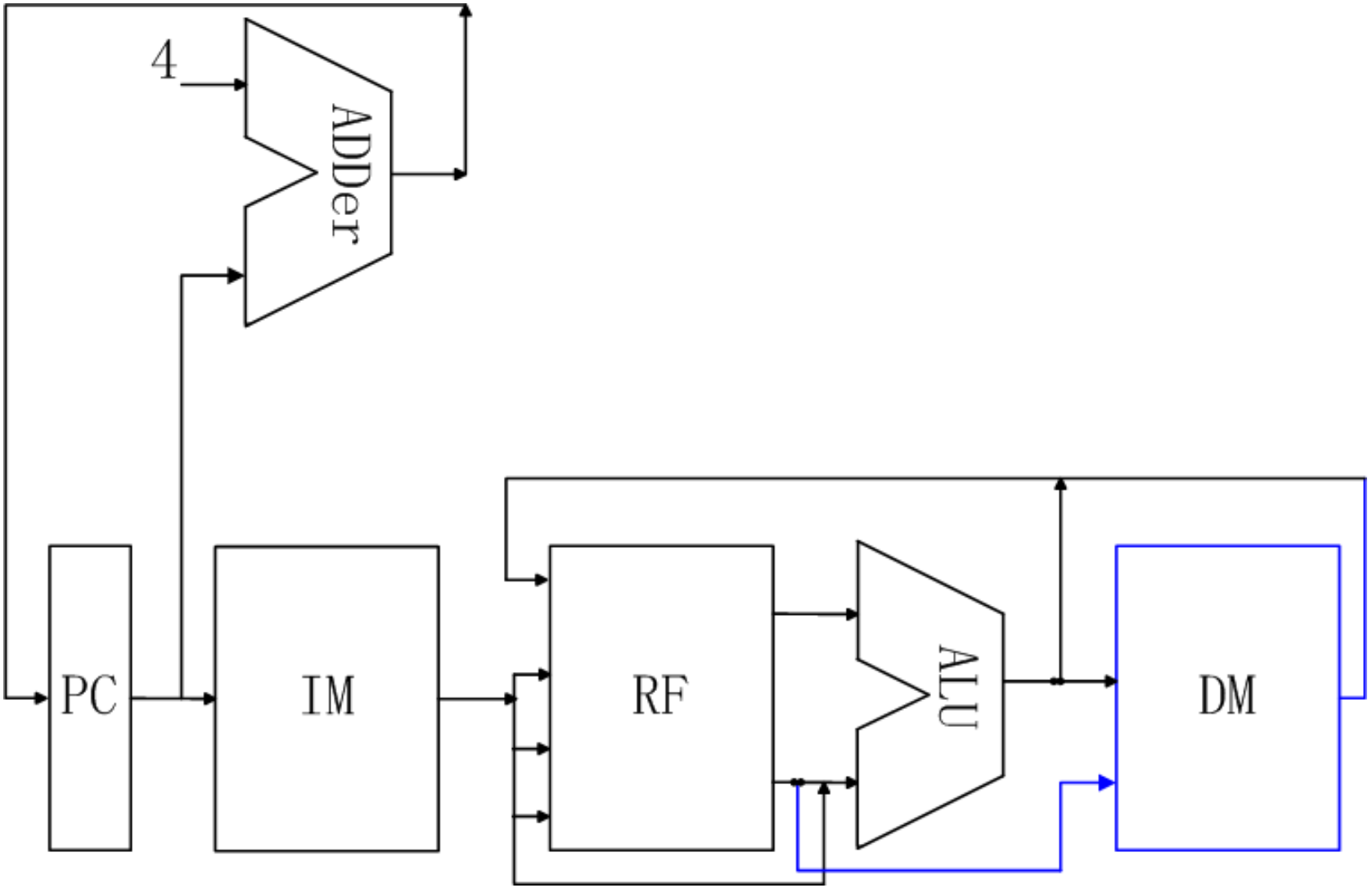
- load
  - lw rt rs imm
  - Addr = R[rs] + SignExt(imm)
  - R[rt] = MEM[Addr]
- store
  - sw rt rs imm
  - Addr = R[rs] + SignExt(imm)
  - MEM[Addr] = R[rt]



数据通路设计2

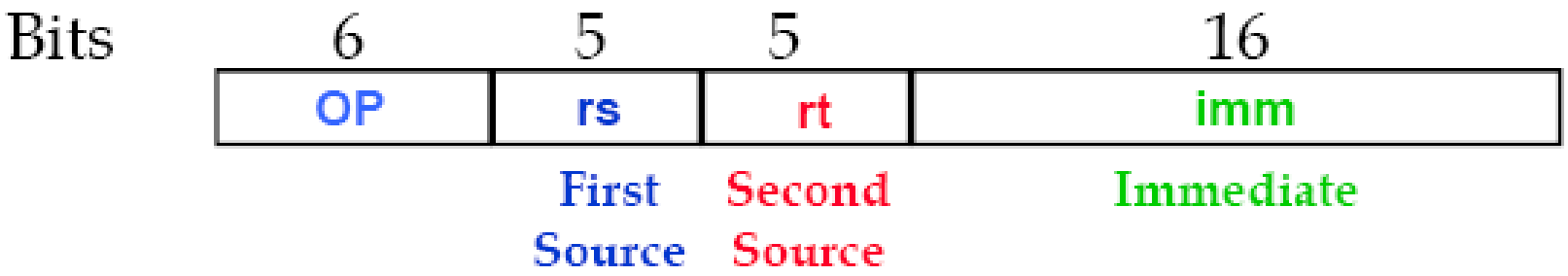


数据通路设计3

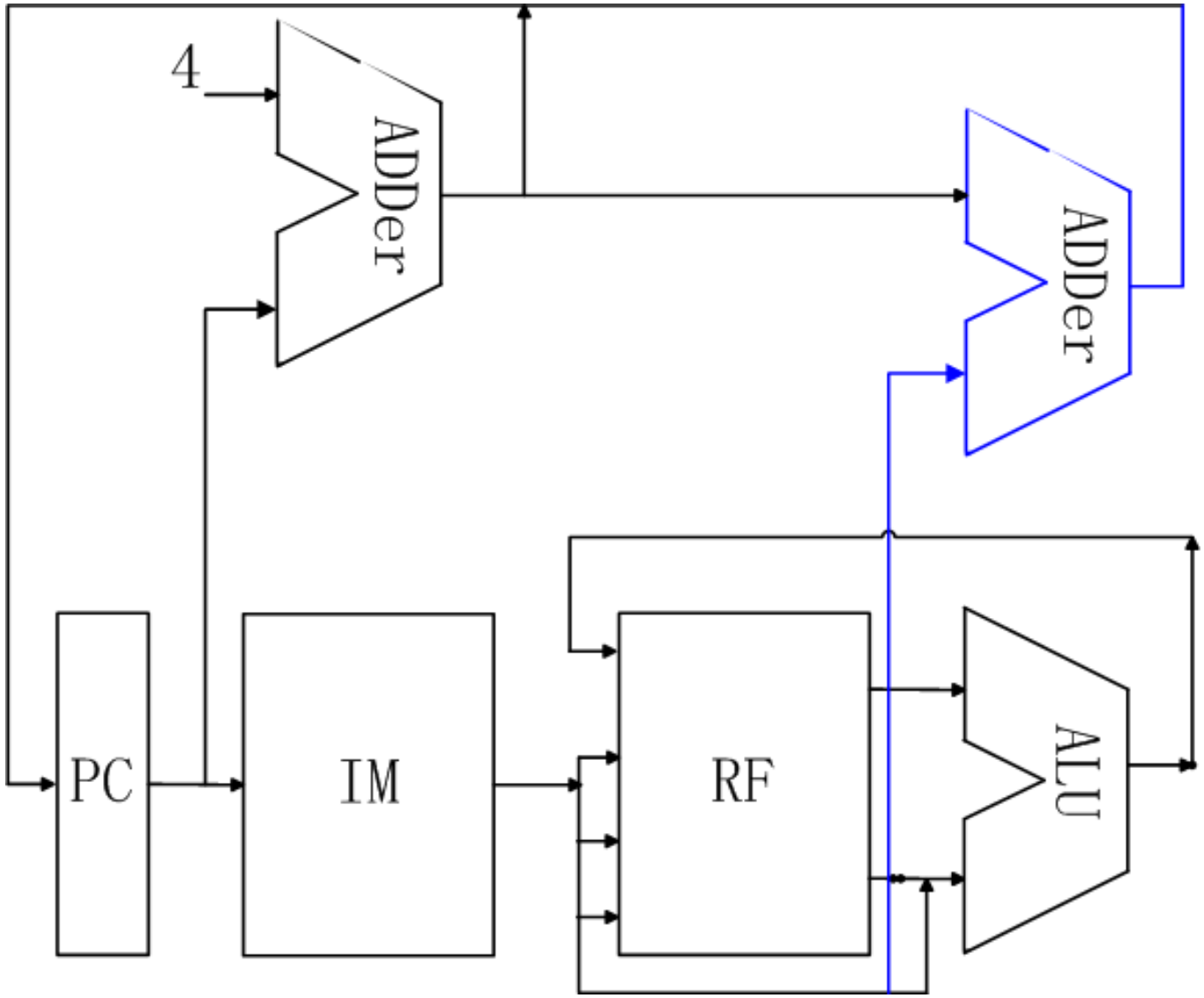


MIPS指令——BEQ

- BEQ
  - beq rs rt imm
  - if R[rs] = R[rt]
  - then PC ← (PC +4)+ SignExt(imm)
  - Else PC← PC+4

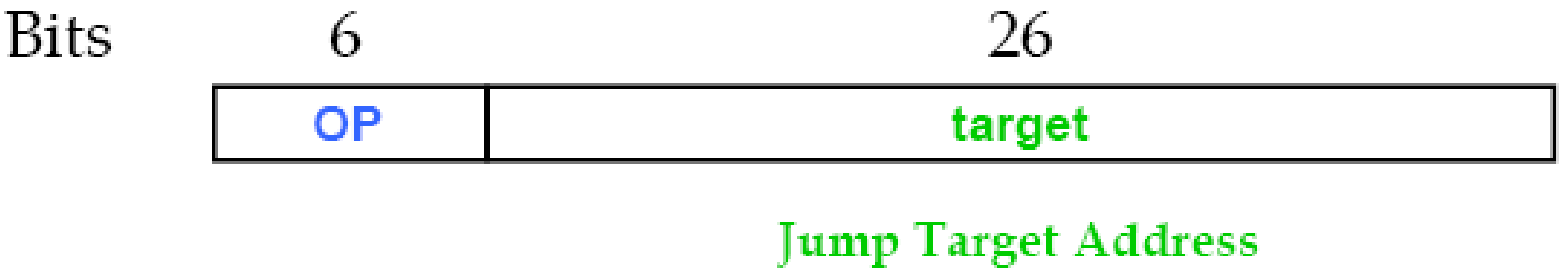


数据通路设计4

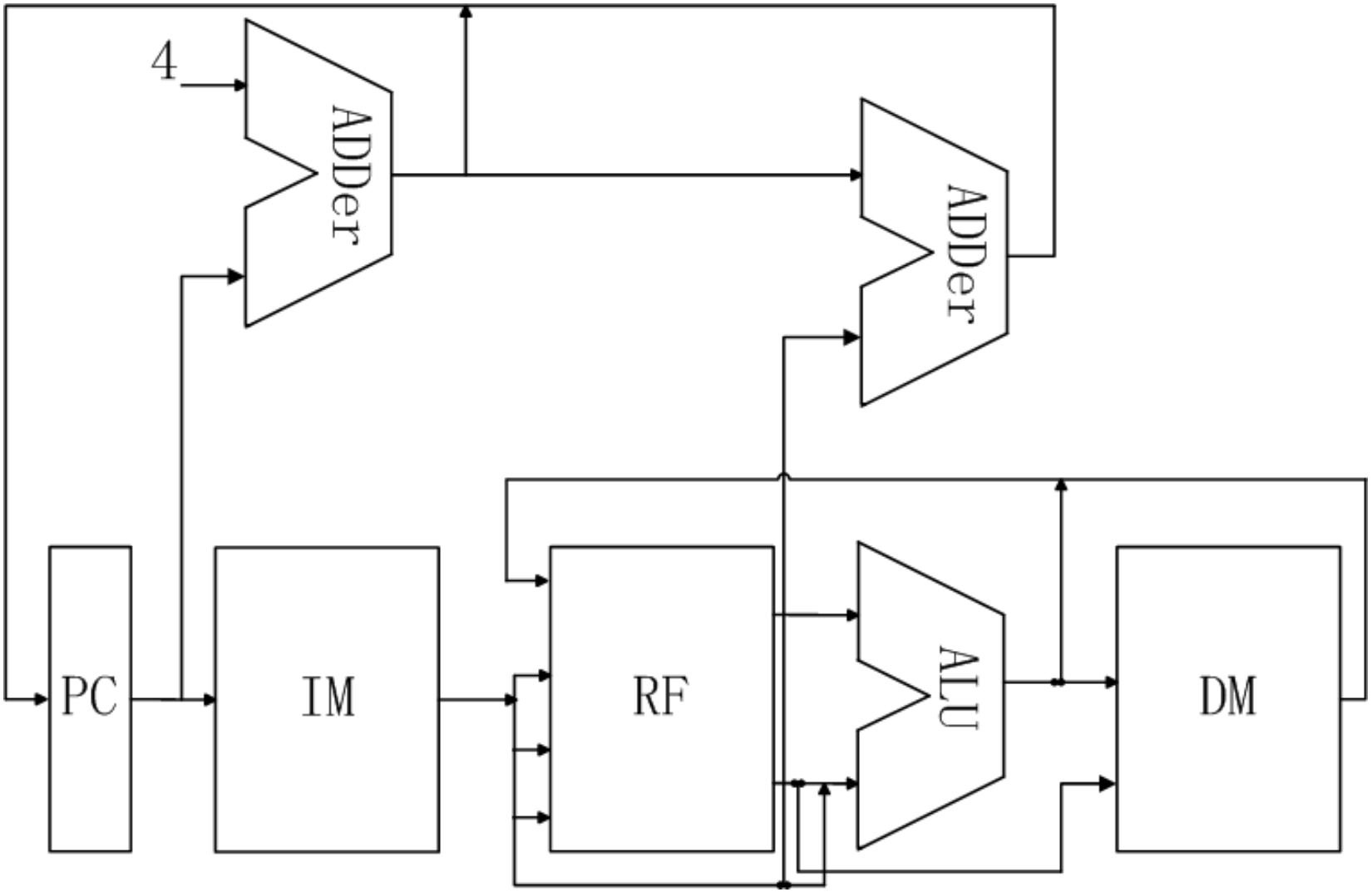


MIPS指令——Jump

- J target
  - PC[31:0]←PC[31:28]||target[25:0]||[00]



完整数据通路设计



指令功能如何实现？

- 硬件组成
  - ALU
  - MEM
  - Register File
  - ADDer
  - PC
  - Mux
- 数据通路
  - 如何实现指令的功能？

指令的执行过程与控制

冯.诺依曼结构的计算机，即存储程序计算机，设置内存，存放程序和数据，在程序运行之前存入。

执行程序：

正确从程序首地址开始；

正确分步地执行每一条指令，

还要形成下条待执行指令的地址；

并保证自动地连续执行指令，

直到结束程序的最后一条指令。

从主存储器读来一条指令，分析指令，按指令的功能要求完成执行过程，本条指令完成后自动开始下一条指令的执行过程，由硬件本身完成。

指令的执行步骤

清華大學 Tsinghua University

当前的计算机系统中，流行的做法是把一条指令的执行过程划分为如下的 5 个阶段，是由指令的功能和计算机硬件结构共同决定的，有内在的逻辑关系。

读取指令 (IF), 从存储器读来指令并形成下条指令地址

指令译码 (ID), 指令译码，读寄存器堆为 ALU准备数据

执行运算 (EXE), ALU 执行数据运算或计算存储器地址

存储器读写 (MEM), 完成存储器的读操作或者写操作

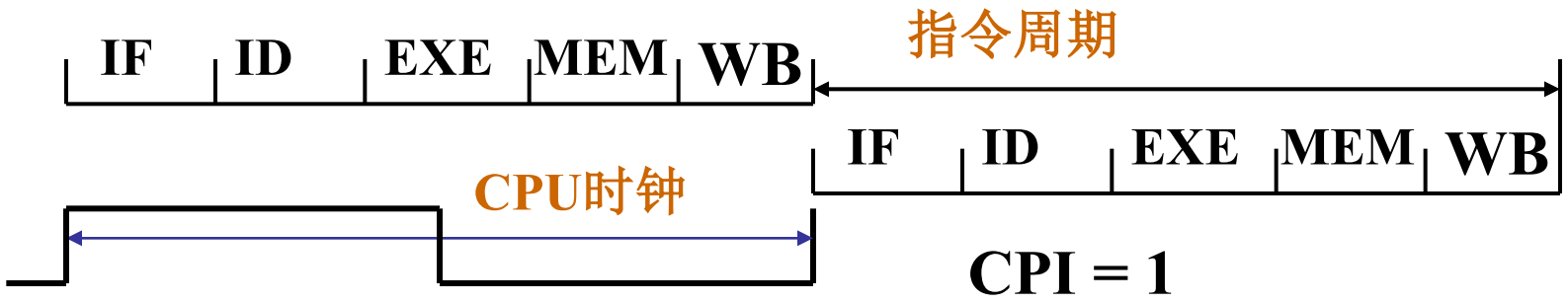
写回 (WB), 写ALU的结果或存储器读出数据到寄存器堆

从如何为不同指令安排这几个阶段，几个阶段如何衔接考虑，大体有 3 种可行方案，各自都对计算机的内部结构、部件设置及其控制方式有着不同要求。

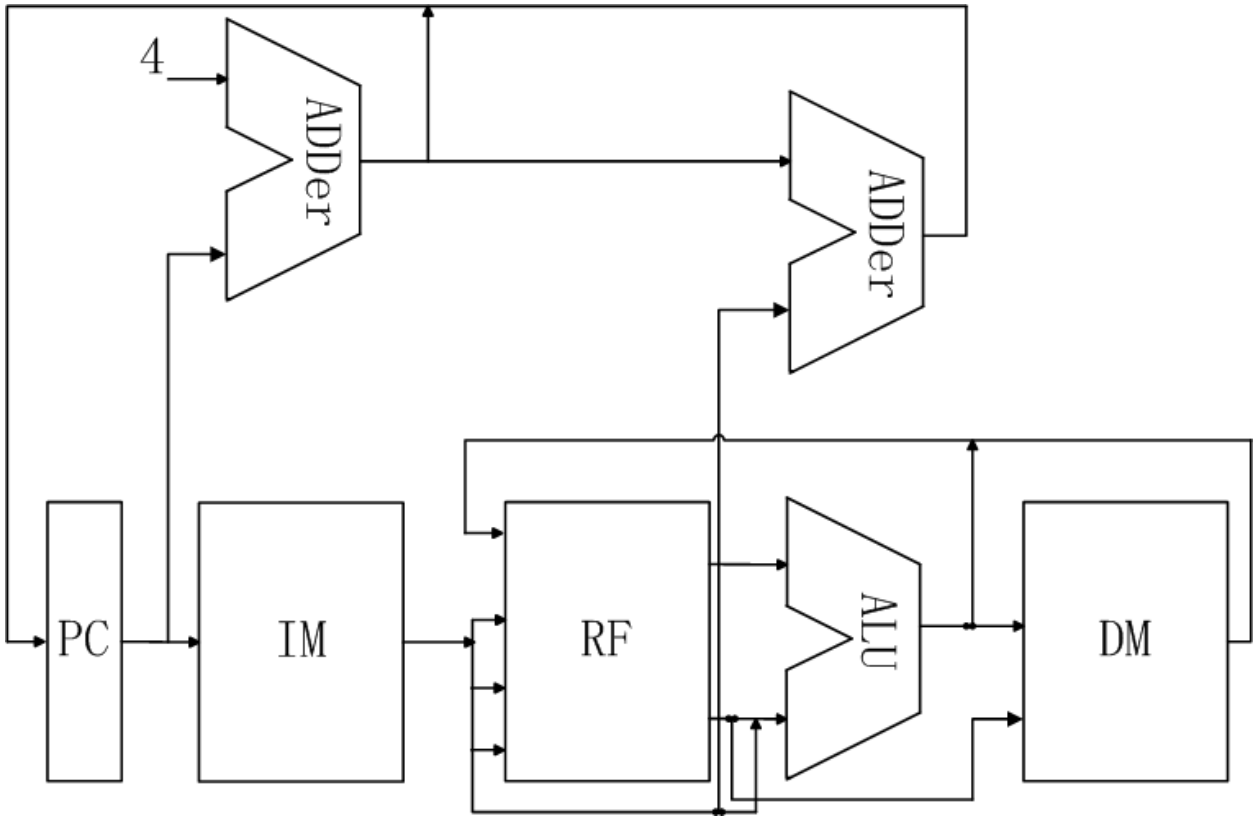
单周期CPU

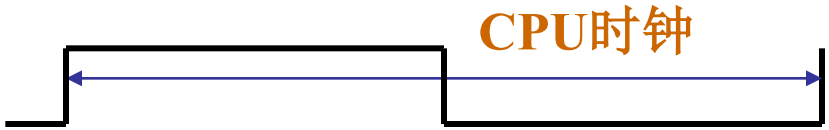
计算机一条指令的执行时间被称为指令周期，一个CPU时钟时间被称为 CPU周期 (在某些计算机中，还可再把一个CPU周期区分为几个更小的步骤，称其为节拍)。执行每条指令平均使用的 CPU周期个数 被称为 CPI

全部指令都选用 一个 CPU周期 完成的系统被称为单周期CPU，指令串行执行，前一条指令结束后才启动下条指令。每条指令都用 5个步骤的时间完成，控制各部件运行的信号在整个指令周期不变化。单周期CPU 用于早期计算机，系统性能和资源利用率很低，相对当前技术变得不再实用。



单周期CPU

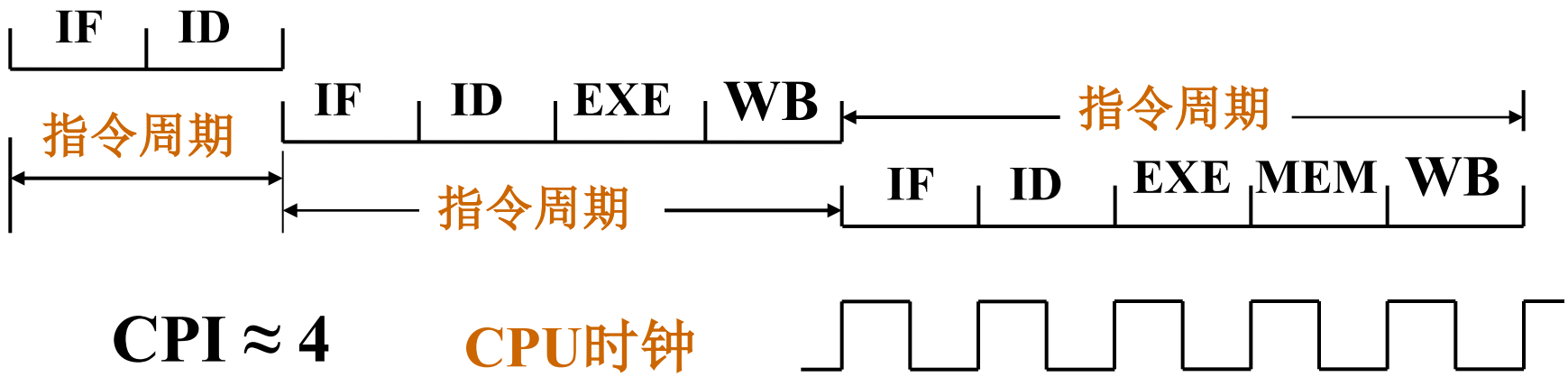




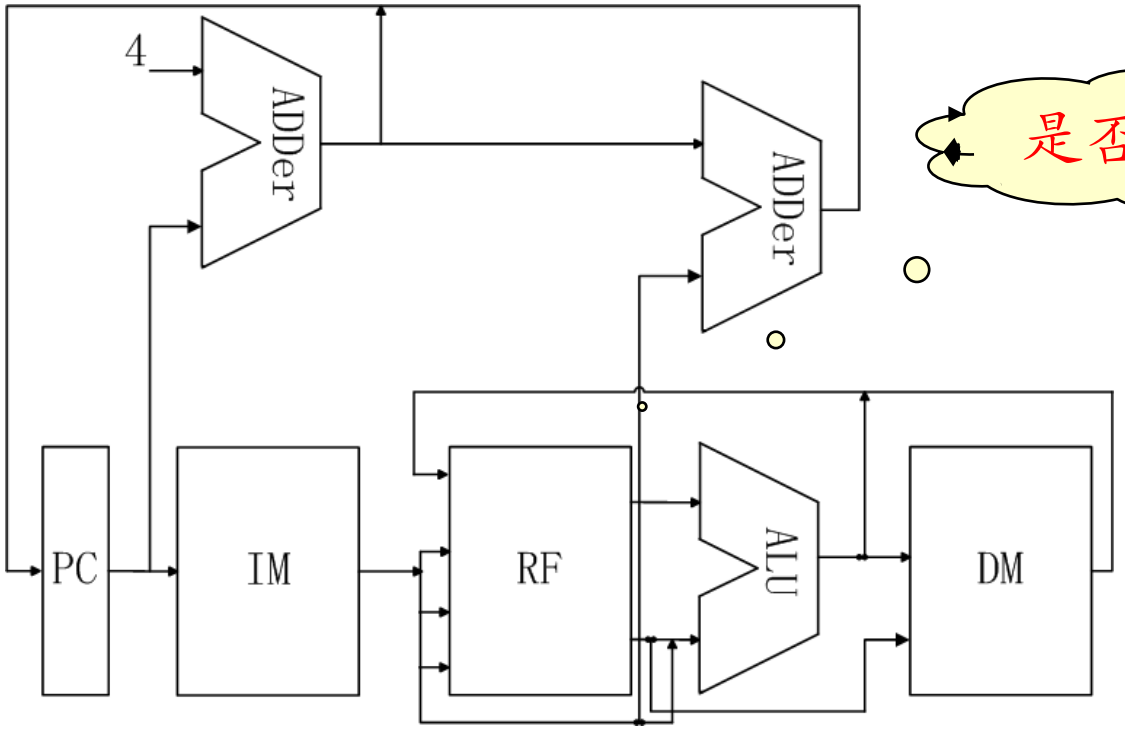
多周期CPU

计算机一条指令的执行时间被称为指令周期，一个CPU时钟时间被称为CPU周期。

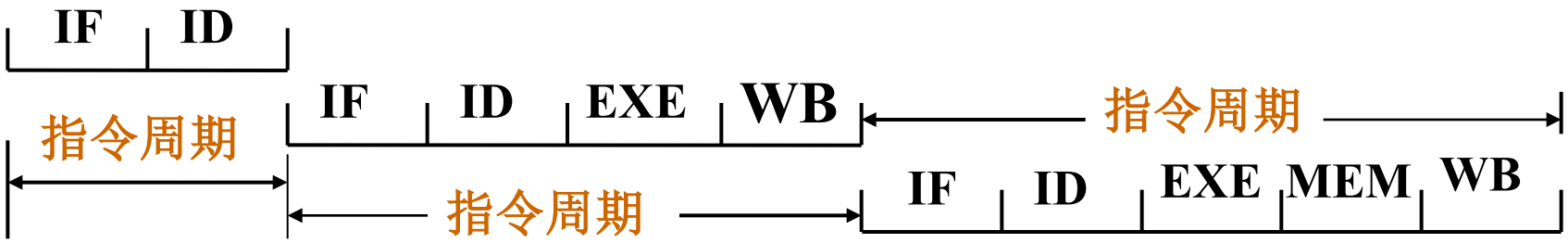
依据不同指令各自的功能需求为其选择不等的执行步骤的系统被称为多周期CPU，控制各部件运行的控制信号随着指令执行步骤改变，系统性能和资源利用率更高。相邻指令可以完全串行执行，也可能部分时间重叠，多周期CPU（相比单周期CPU）更实用。



多周期CPU



是否适用?

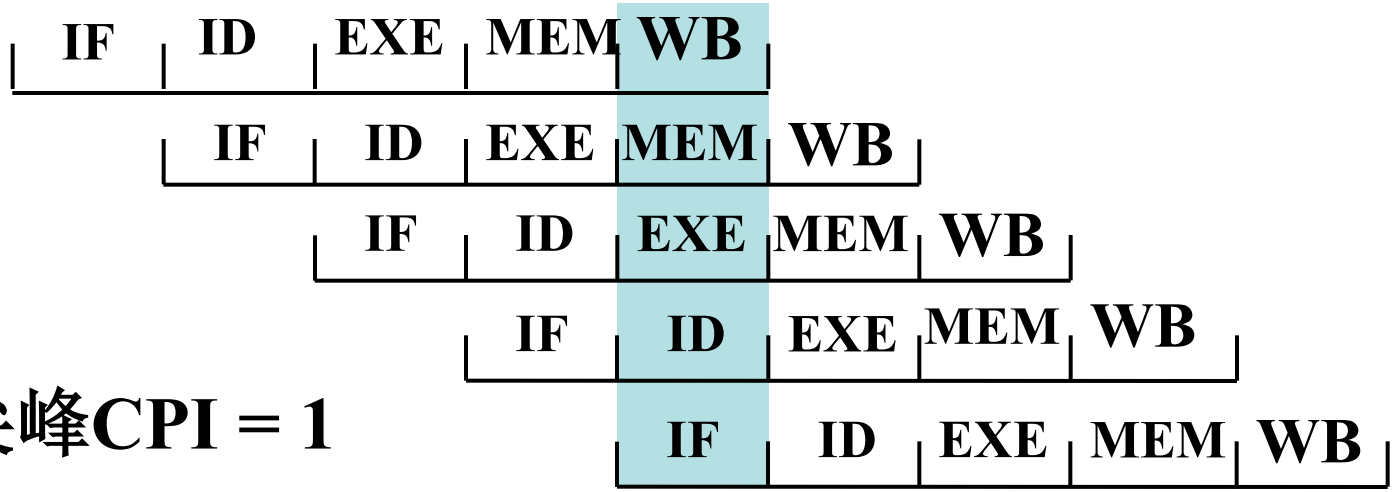


CPI ≈ 4

CPU时钟

指令流水线CPU

全部指令都是选用5个步骤完成，执行时间相同，但相邻指令的执行并不是完全串行的，执行时间有所重叠，例如每结束指令的一个执行步骤就启动下条指令，这被称为指令流水线技术，所有部件都高速运行，尖峰速度每个CPU时钟执行一条指令，系统性能和资源利用率更高，显著地提高系统的性能价格比，但计算机结构和控制器的设计、实现略显复杂。当前计算机中普遍使用这种方案。



尖峰CPI = 1

小结

- 设计目标直接决定了指令的取舍
- 指令系统受到技术条件的制约
- 兼容性是指令系统的重要要求
- RISC，以简洁换取性能提高
- CISC，以丰富换取编程方便
- 指令执行
  - 单步骤串行
  - 多步骤串行
  - 流水

阅读和思考

- 阅读
  - 教材第4章
- 思考
  - 比较Pentium和MIPS指令系统
- 实践
  - 分析THCO MIPS指令系统的特点，并对其指令格式进行分类
  - 设计能实现THCO MIPS指令系统的数据通路，尤其要注意专用寄存器的实现方法