清華大學
Tsinghua University

第二单元　第四讲

多周期CPU设计

刘卫东

计算机科学与技术系

本讲提要

- 单周期CPU特点
- 单周期CPU主要不足
- 多周期CPU设计
  - 多周期CPU控制器基本组成
  - 多周期CPU设计

控制器的功能

冯.诺依曼结构的计算机

“存储程序”计算机，设置内存，存放程序和数据

在程序运行之前将程序调入内存，然后执行程序

计算机的功能是执行程序

程序是依次排列起来的指令序列

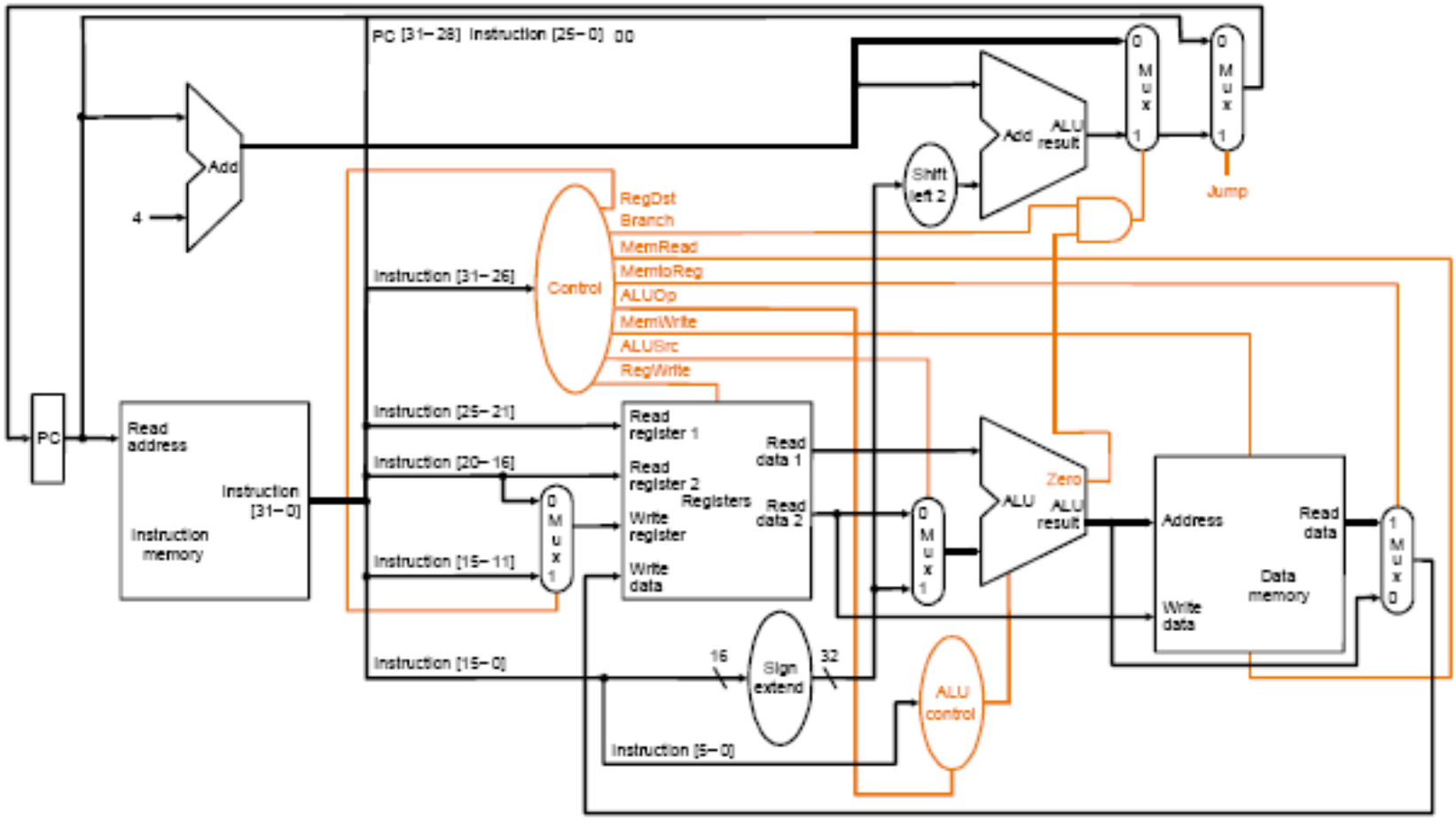
计算机执行程序的基本过程

从程序首地址开始执行第一条指令

(分步)执行每一条指令，并形成下一条待执行指令地址

自动地连续执行指令，直到程序的最后一条指令

完整的单周期CPU


PC [31–28] Instruction [25–0] 00
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]
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
Jump
ALU
Zero
ALU result
ALU control
Address
Read data
Data memory
Write data
M u x

单周期CPU特点

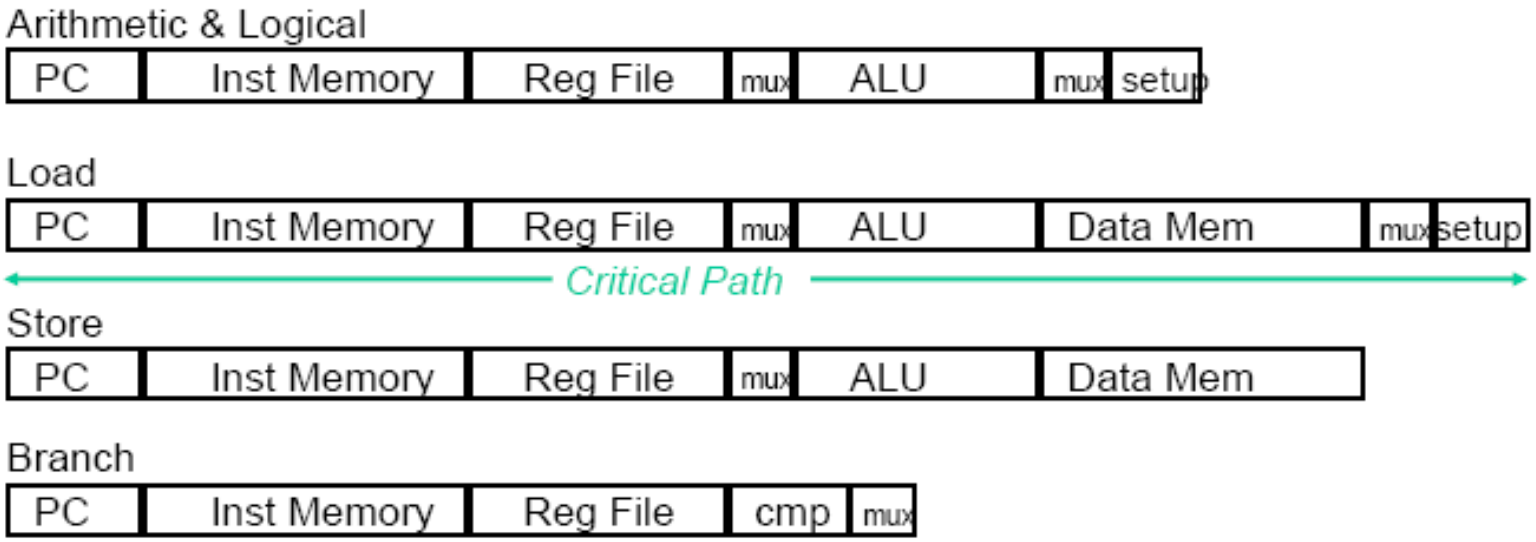
- 优点
  - 每条指令占用一个CPU周期
  - 逻辑设计简单，时钟设计也简单
- 缺点
  - 各组成部件的利用率不高
    - 各部件大部分时间在保持信号
  - 时钟周期将满足执行时间最长指令的要求
    - Load指令
  - CPI =1

单周期CPU性能

- 假定某单周期CPU各主要部件的延迟为：
  - 存储器（Memory）：2ns
  - 运算器（ALU/Adder）：2ns
  - 寄存器组（Register File）：1ns

Inst.	Inst. Mem	Reg. Read	ALU	Data Mem	Reg. Write	Total
ALU	2	1	2		1	6
lw	2	1	2	2	1	8
sw	2	1	2	2		7
br	2	1	2			5

单周期CPU性能



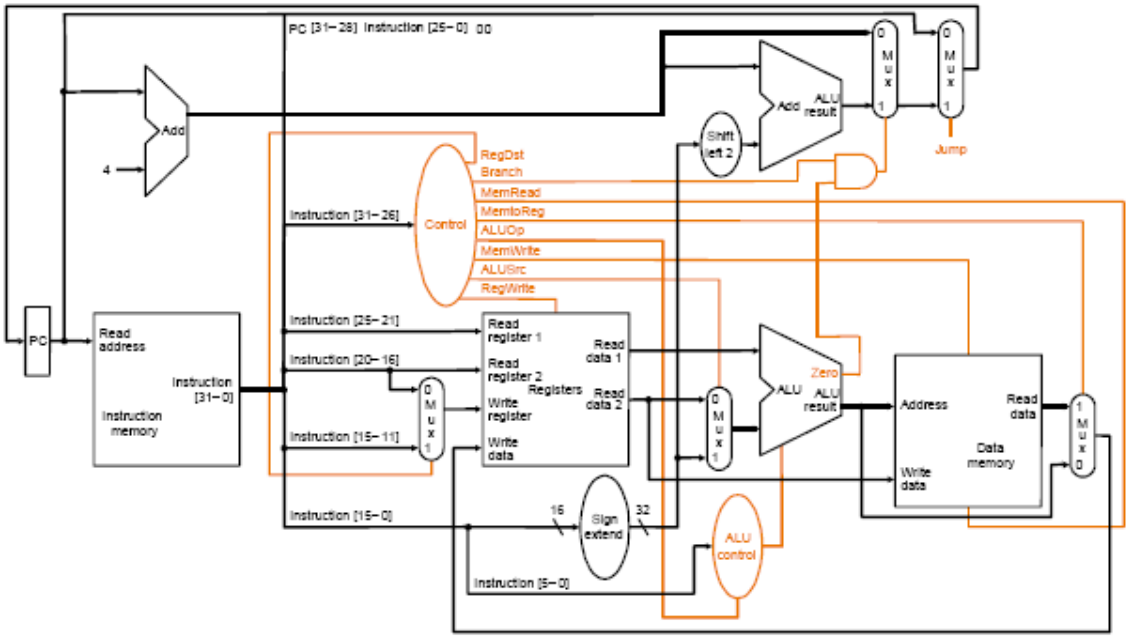
- 指令周期比较长
- 所有指令都必须使用最长的周期

单周期CPU性能

- 假设某单周期CPU，执行100条指令:
  - 25%的Load指令
  - 10%的Store指令
  - 45%的算逻指令
  - 20%的跳转指令
- 单周期的执行时间
  - 100*8 = 800ns
- 可能的优化
  - 25*8+10*7+45*6+20*5=640ns
  - Speedup=800/640=1.25

单周期CPU其他问题

- 事实上，指令和数据都保存在同一个存储器中
- 许多部件保持数据的时间过长，无法复用
  - 例如，Adder 是否可以利用ALU?



多周期CPU

- 将指令执行过程分解成多个步骤
  - 和单周期CPU基本相同
  - 每条指令占用它需要的步骤数
- 每个步骤占用一个时钟周期
  - 尽量平衡各步骤间的延迟
  - 尽量限制每个步骤使用单一的主要部件
  - 控制器仅需提供当前步骤所需要的控制信号
- 前提
  - 保存好下一步骤要用到的值
    - 引入“新”的内部寄存器
  - 转到下一步骤执行
    - 引入状态标记当前步骤
    - 有限自动机

多周期CPU的控制器

清華大學 Tsinghua University

控制器的功能就是控制指令的执行过程

能够正确并且自动地连续执行指令

按程序中设定的指令次序执行

正确地分步完成每一条指令规定的功能

读取指令 → 分析指令 → 执行指令

进一步讲，就是向计算机各功能部件提供协调运行每一个步骤所需要的控制信号

控制器的组成

清華大學 Tsinghua University

① 程序计数器 PC

存放指令地址，有 增量 或 接收新值功能

② 指令寄存器 IR

存放指令内容：操作码与操作数地址

③ 指令执行步骤标记线路

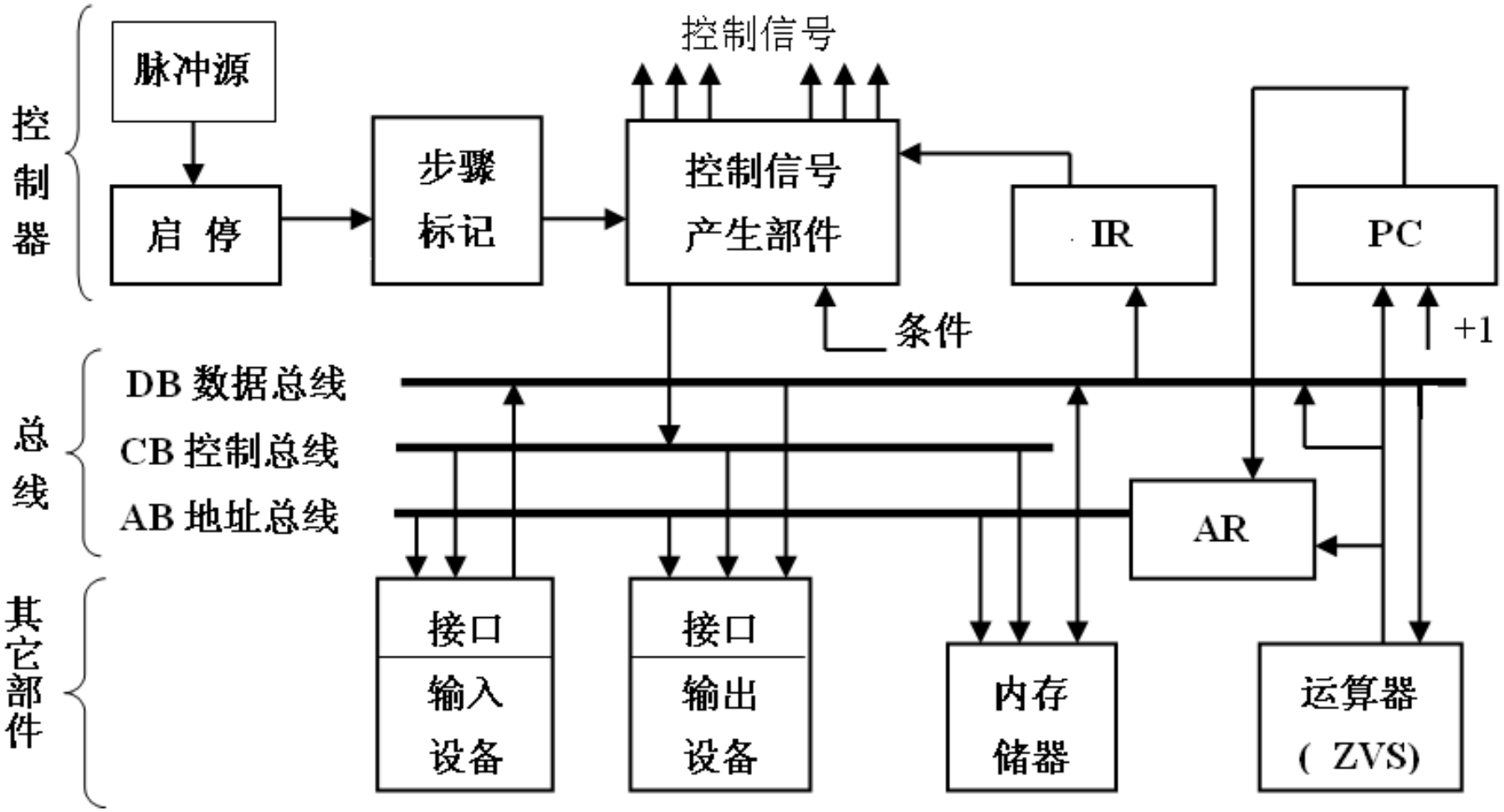
指明每条指令的执行步骤和相对次序关系

④ 控制信号产生线路

给出计算机各功能部件协同运行所需要的控制信号

主脉冲源与启停控制线路

多周期CPU控制器组成



两种不同类型的控制器

根据指令步骤标记线路和控制信号产生线路不同的组成和不同的运行原理，有两种不同类型的控制器：

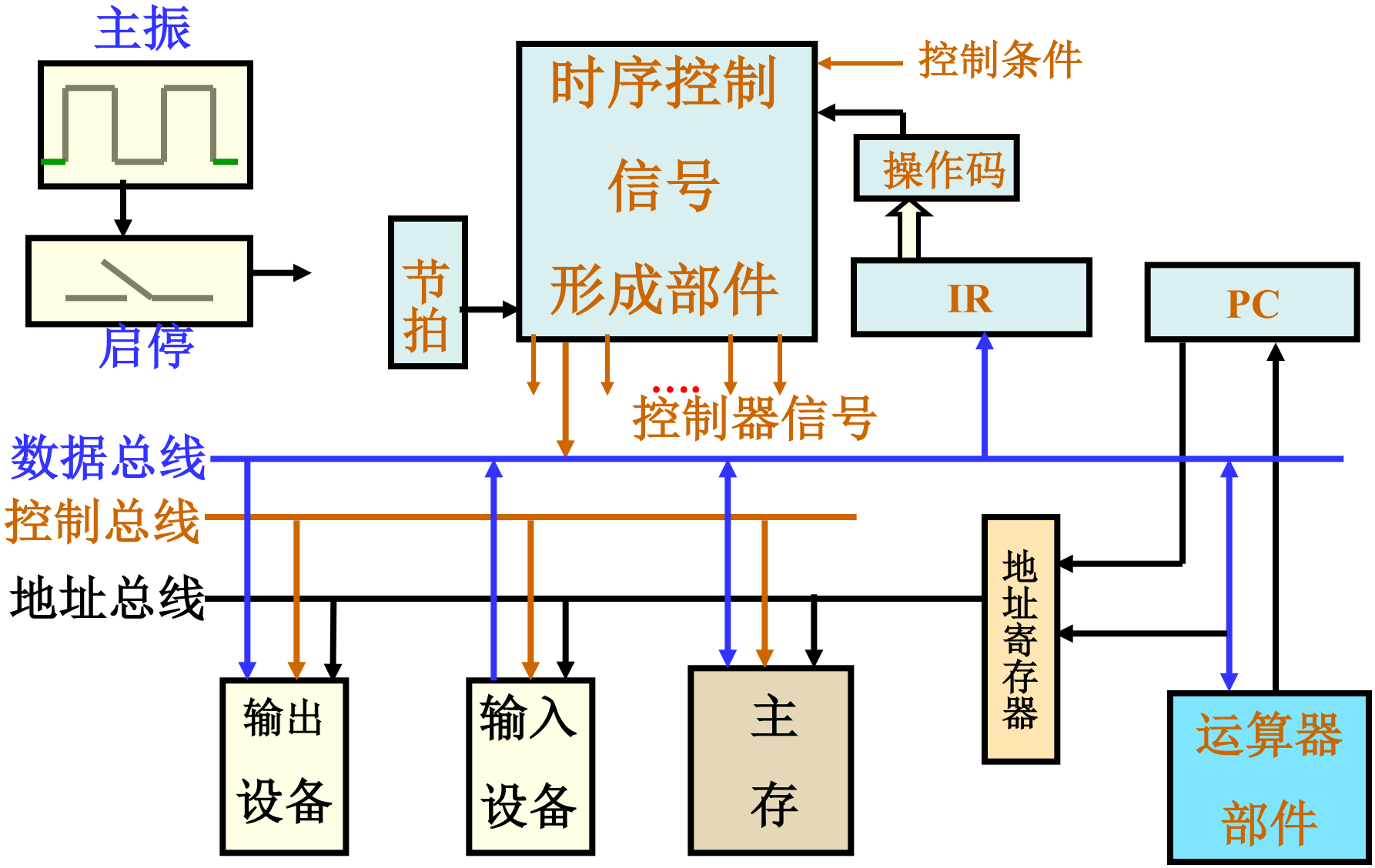
硬连线控制器(组合逻辑控制器)：

采用组合逻辑线路、依据指令及其执行步骤直接产生控制信号。

微程序控制器：

采用存储器电路把控制信号存储起来，依据指令执行的步骤读出要用到的信号组合。

硬连线控制器


主振
启停
节拍
时序控制
信号
形成部件
控制条件
操作码
IR
PC
控制器信号
数据总线
控制总线
地址总线
输出
设备
输入
设备
主
存
地址寄存器
运算器
部件

硬连线控制器组成与实现

清華大學 Tsinghua University

硬连线控制器由程序计数器PC、指令寄存器IR、节拍发生器Timer和控制信号产生部件4部分组成。

PC用于提供待读出指令在主存储器中的地址，

IR用于保存从主存储器中读出的指令内容，

Timer用于给出并维护指令执行步骤的编码，

控制信号产生部件用于依据指令内容（在IR中）和指令执行所处的操作步骤（Timer提供），用组合逻辑线路产生计算机本操作步骤中各个部件所需要的控制信号。

划分指令执行步骤，确定各步骤应执行的功能和步骤之间的衔接关系，以及确定各部件完成这些功能所需要的控制信号，是控制器设计的几个关键环节。

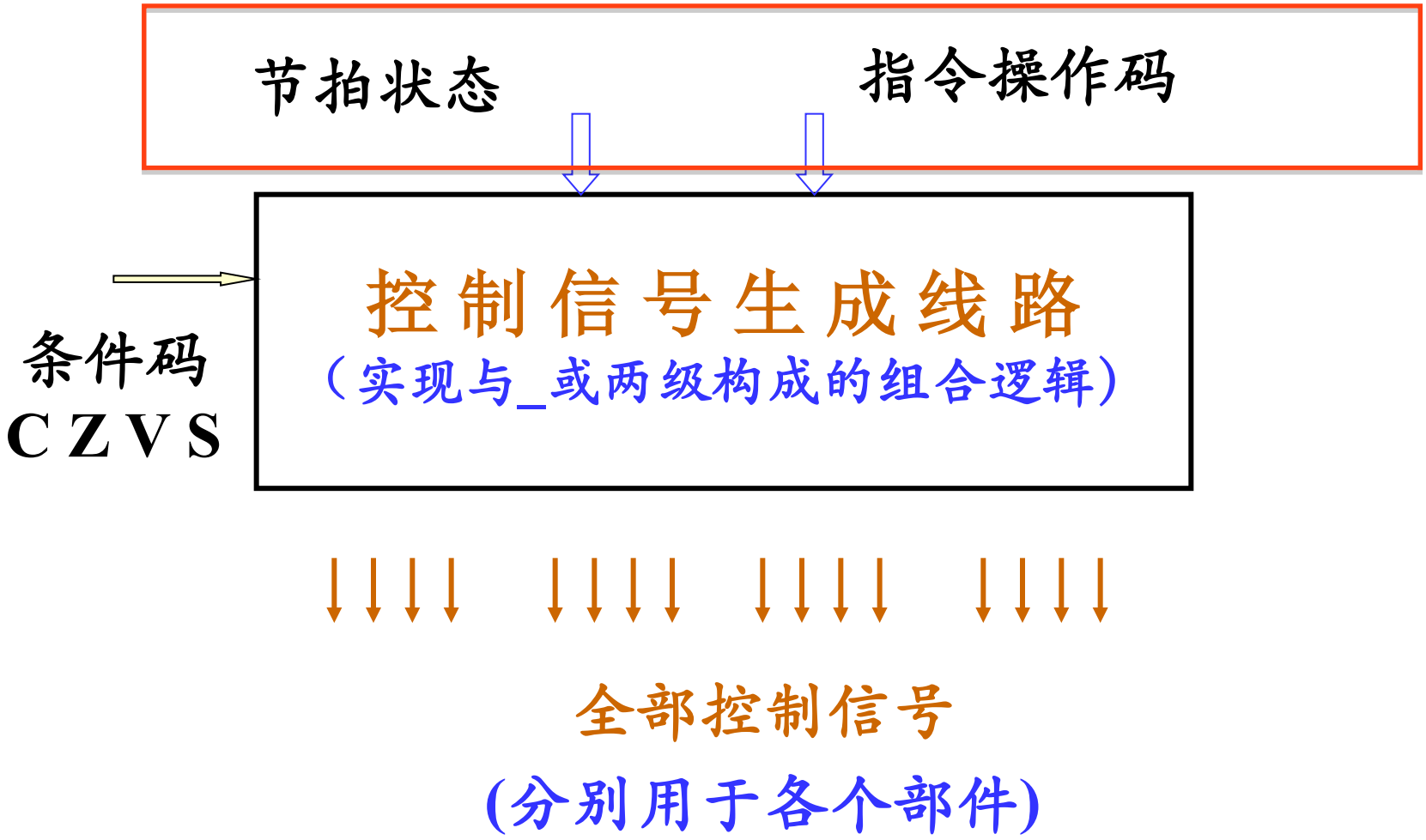
硬连线控制器组成与实现

清華大學 Tsinghua University

在多周期 CPU 系统中，要按照指令总的功能要求，把不同的功能序列划分到相应的步骤，再落实到不同的部件，控制器需要按照指令及其执行步骤，为计算机各个部件提供它们协同运行所需要的控制信号。

向各部件提供哪些控制信号，决定于各部件的运行要求。为此必须规划汇总各部件在各个执行步骤中要求使用的控制信号。这些信号是用组合逻辑电路产生的，可以表示为：信号i＝f(指令内容, 执行步骤 等)，通常表现为多个由与、或两级逻辑构成的表达式。

控制信号生成线路

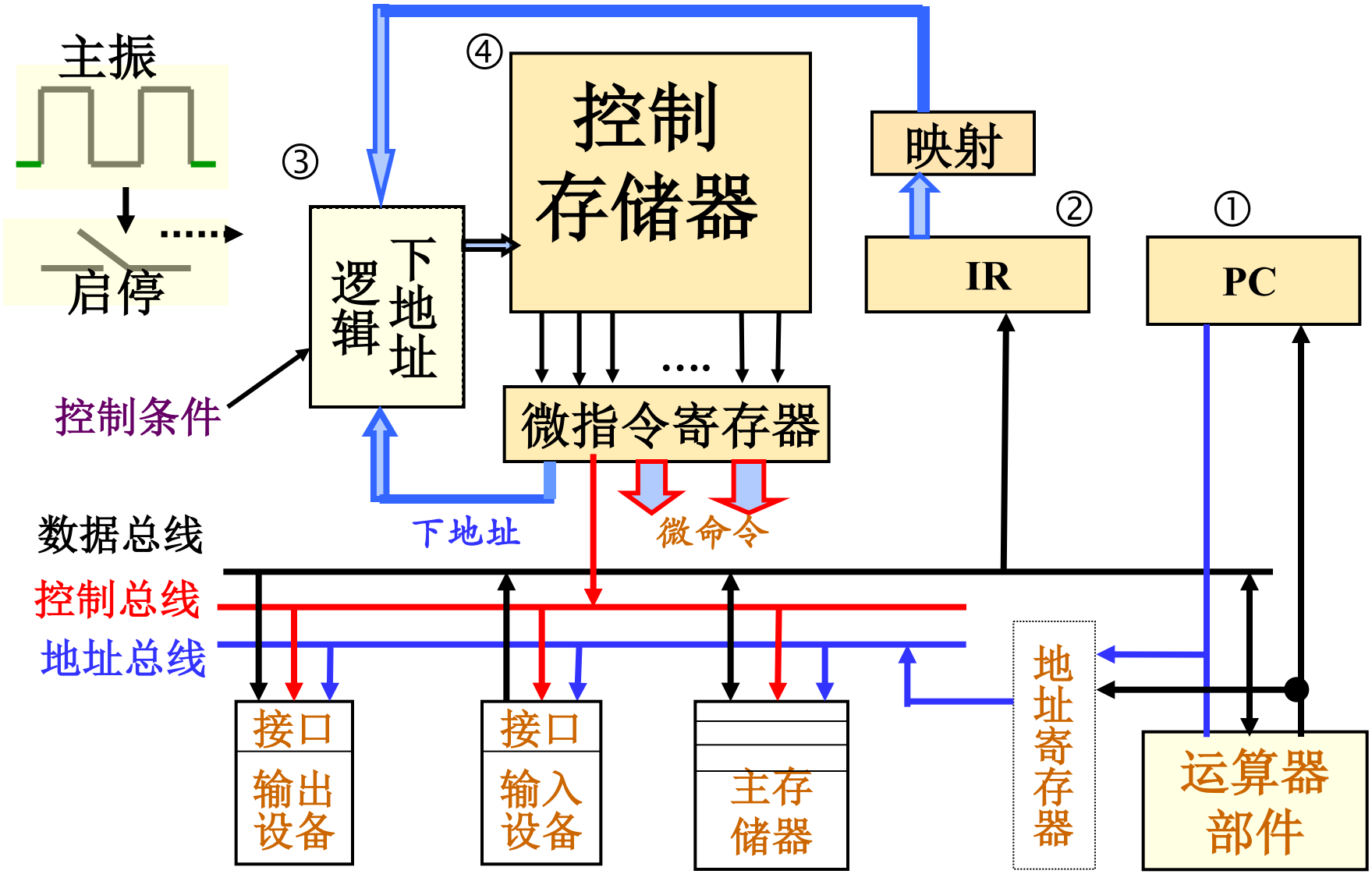

节拍状态
指令操作码
控制信号生成线路
(实现与_或两级构成的组合逻辑)
条件码
C Z V S
全部控制信号
(分别用于各个部件)

微程序控制器

- 采用控制存储器存储每条指令的每个执行步骤所需要的全部控制信号
  - 用微地址进行访问，读出控制信号并输出

- 采用下地址逻辑实现执行步骤之间的衔接
  - 根据指令操作码映射出该指令的首条微指令的地址
  - 每条微指令给出其下一步骤的微地址

微程序控制器的组成

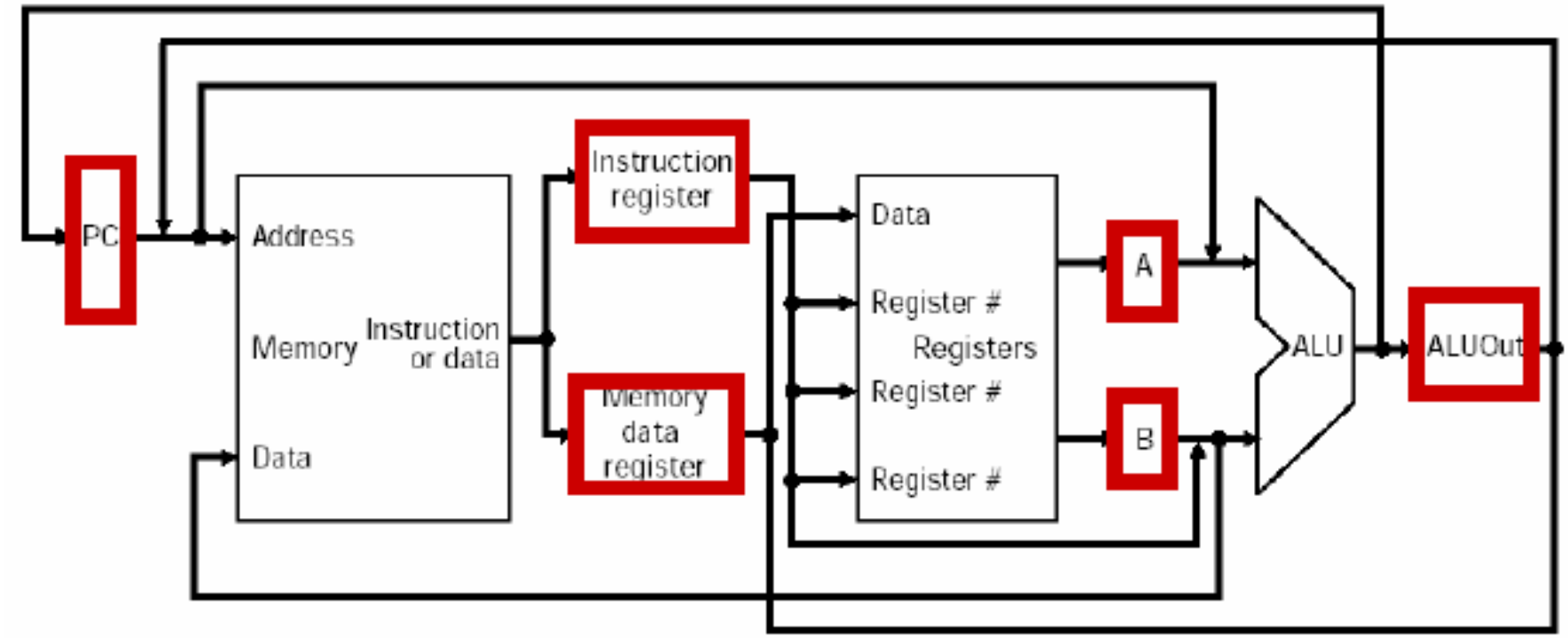

主振
启停
③
④
控制
存储器
映射
②
①
IR
PC
逻辑
下地址
控制条件
微指令寄存器
下地址
微命令
数据总线
控制总线
地址总线
接口
输出
设备
接口
输入
设备
主存
储器
地址
寄存
器
运算器
部件

多周期CPU控制器设计

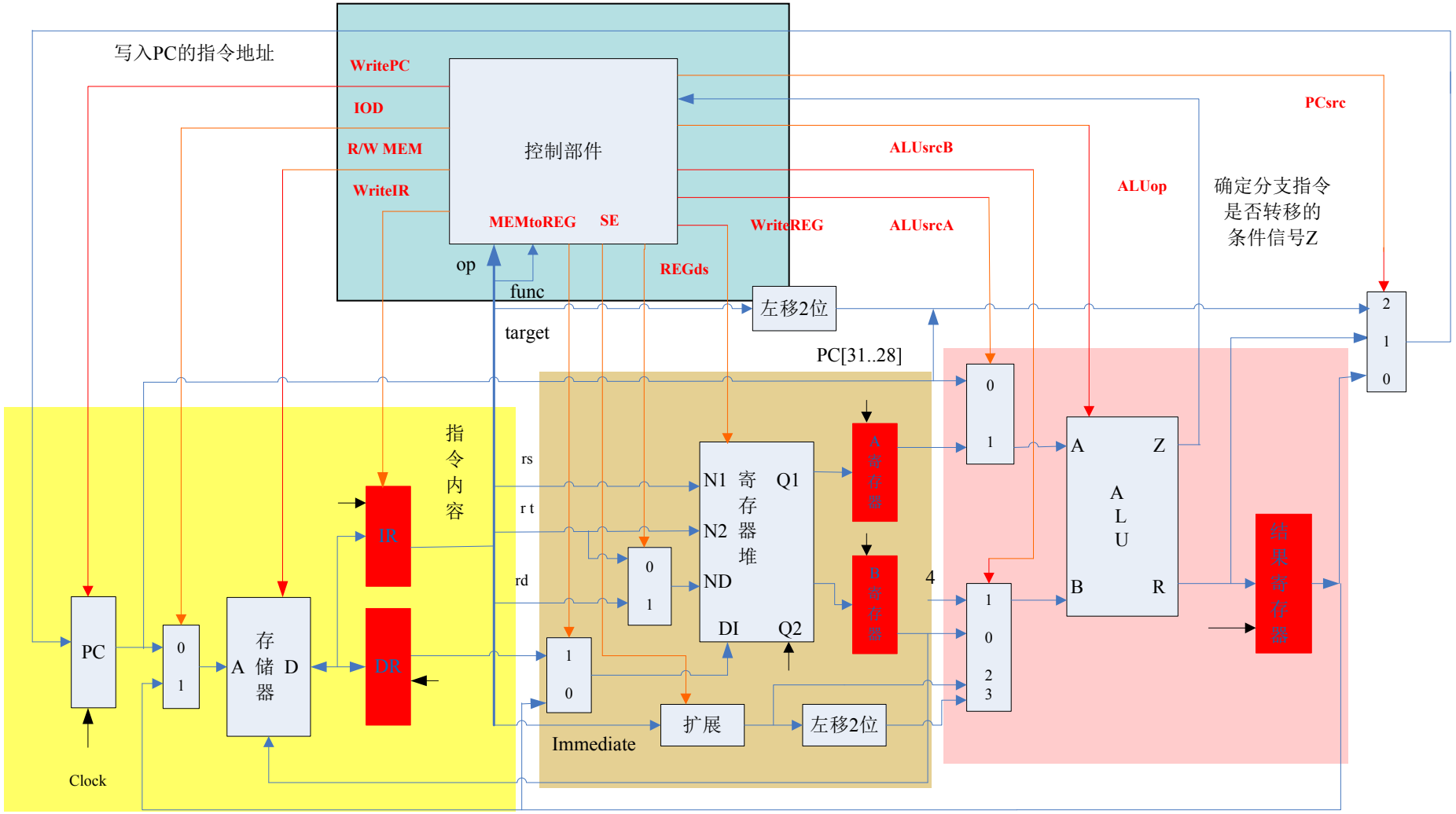
清華大學 Tsinghua University

- 确定数据通路
- 划分指令执行步骤
  - 指令流程图
- 安排每条指令每个步骤的功能,并给出相应的控制信号
  - 指令流程表
- 为指令执行步骤设计状态机
- 为每个步骤的控制信号设计控制信号生成逻辑

多周期CPU的Datapath

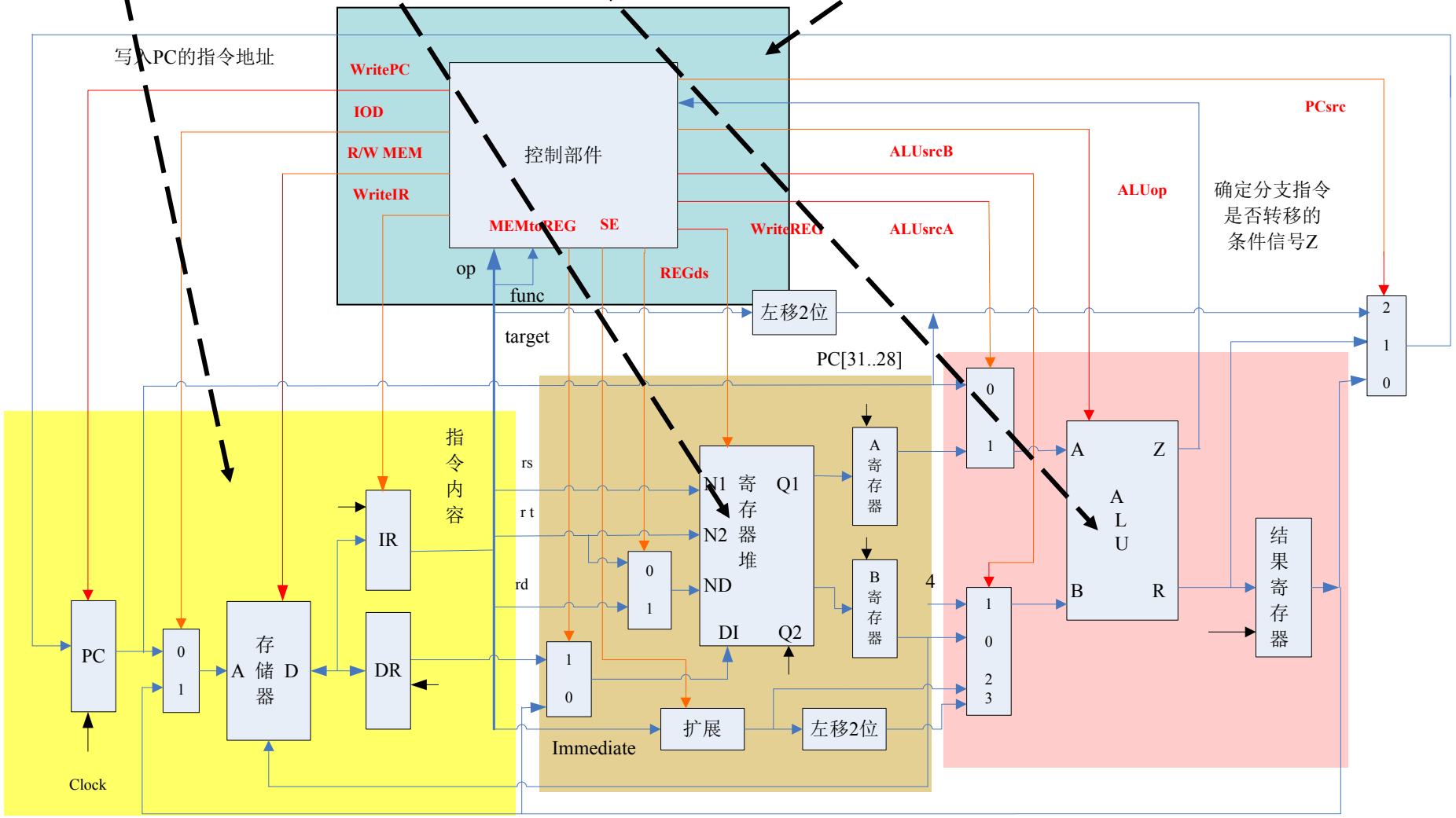


MIPS计算机硬件组成

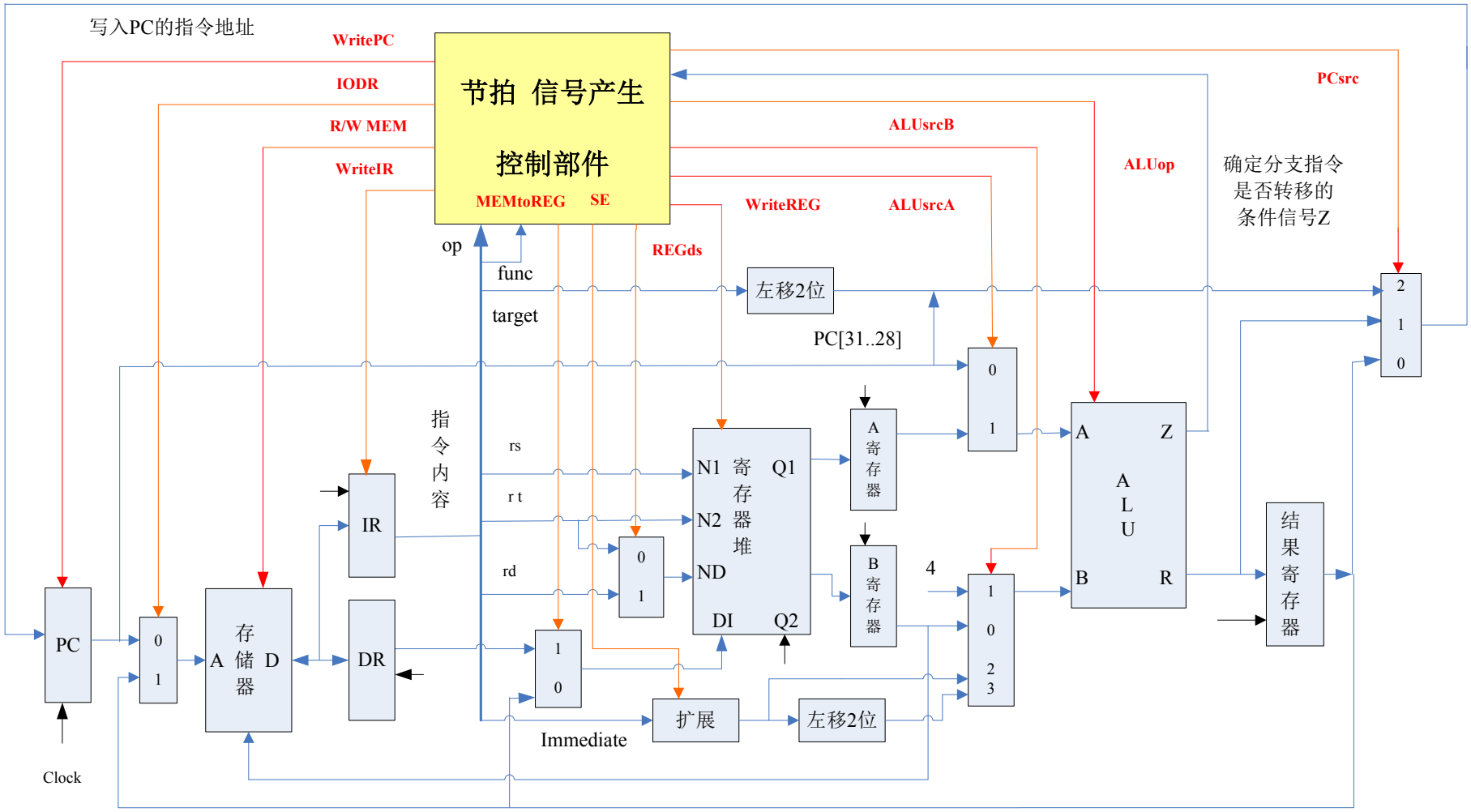


MIPS计算机硬件组成

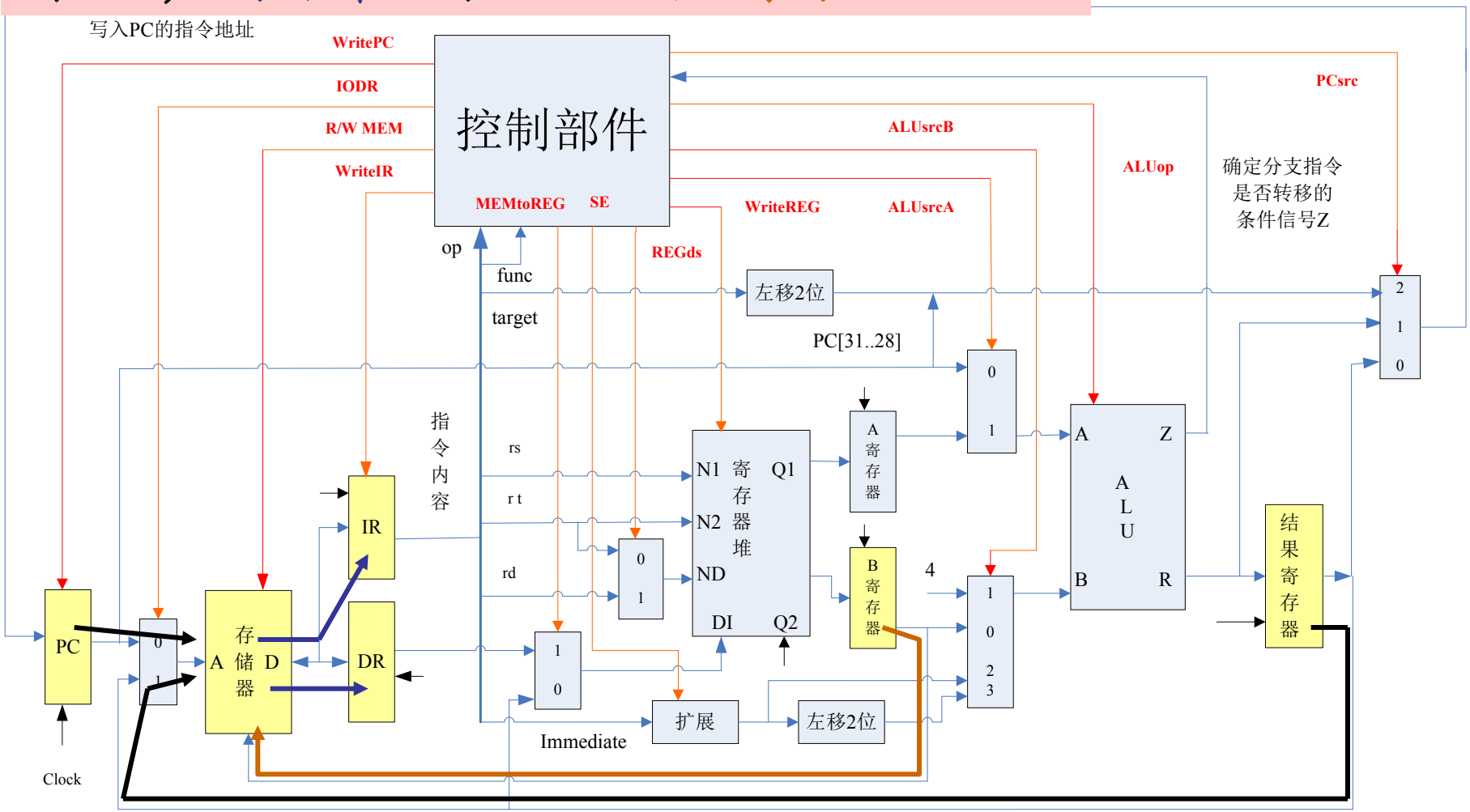
由 存储器、寄存器堆、ALU部件、控制部件 4大部分组成



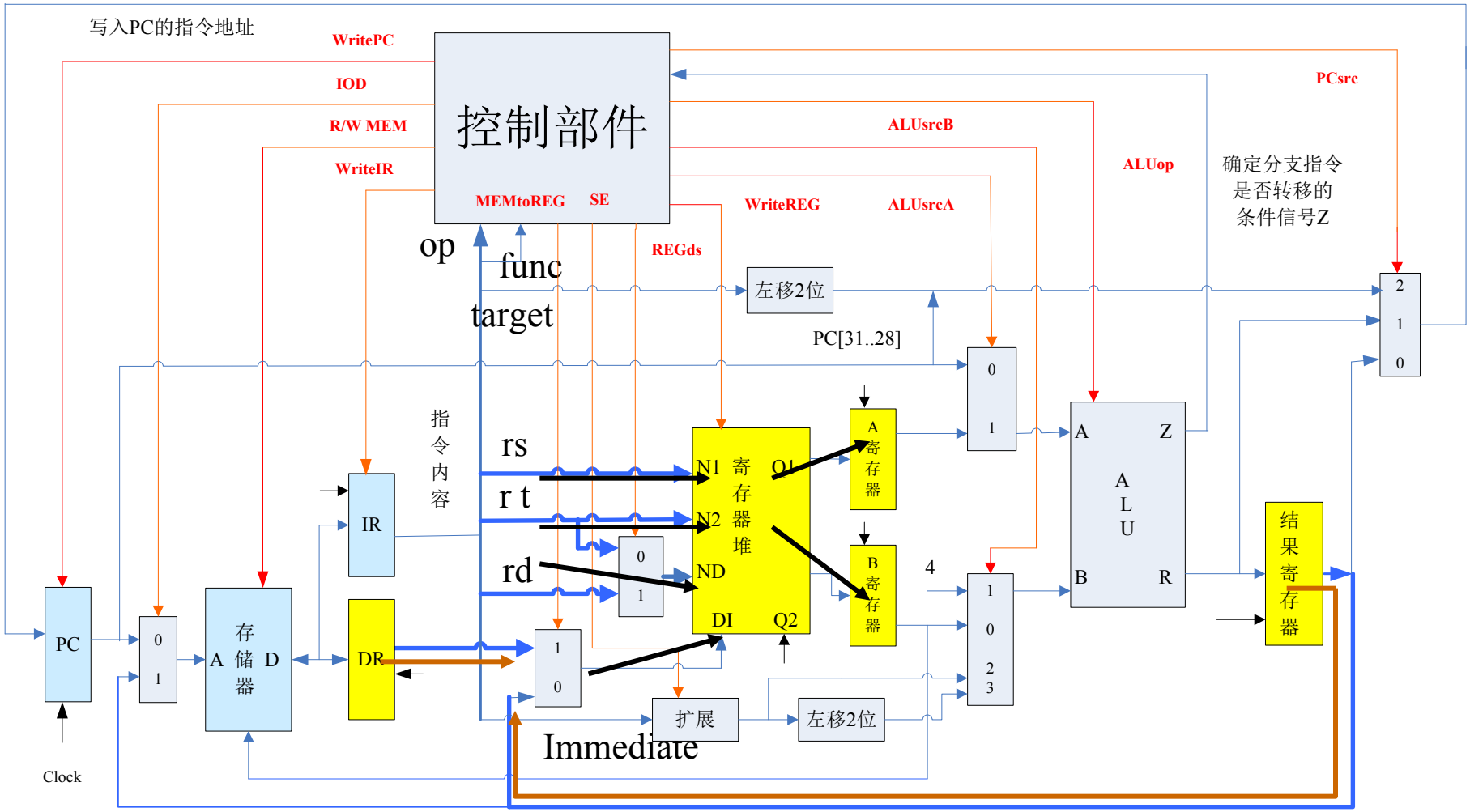
控制部件由节拍发生器和控制信号产生线路组成，分别完成标明指令执行步骤和向各个部件提供控制信号的功能。



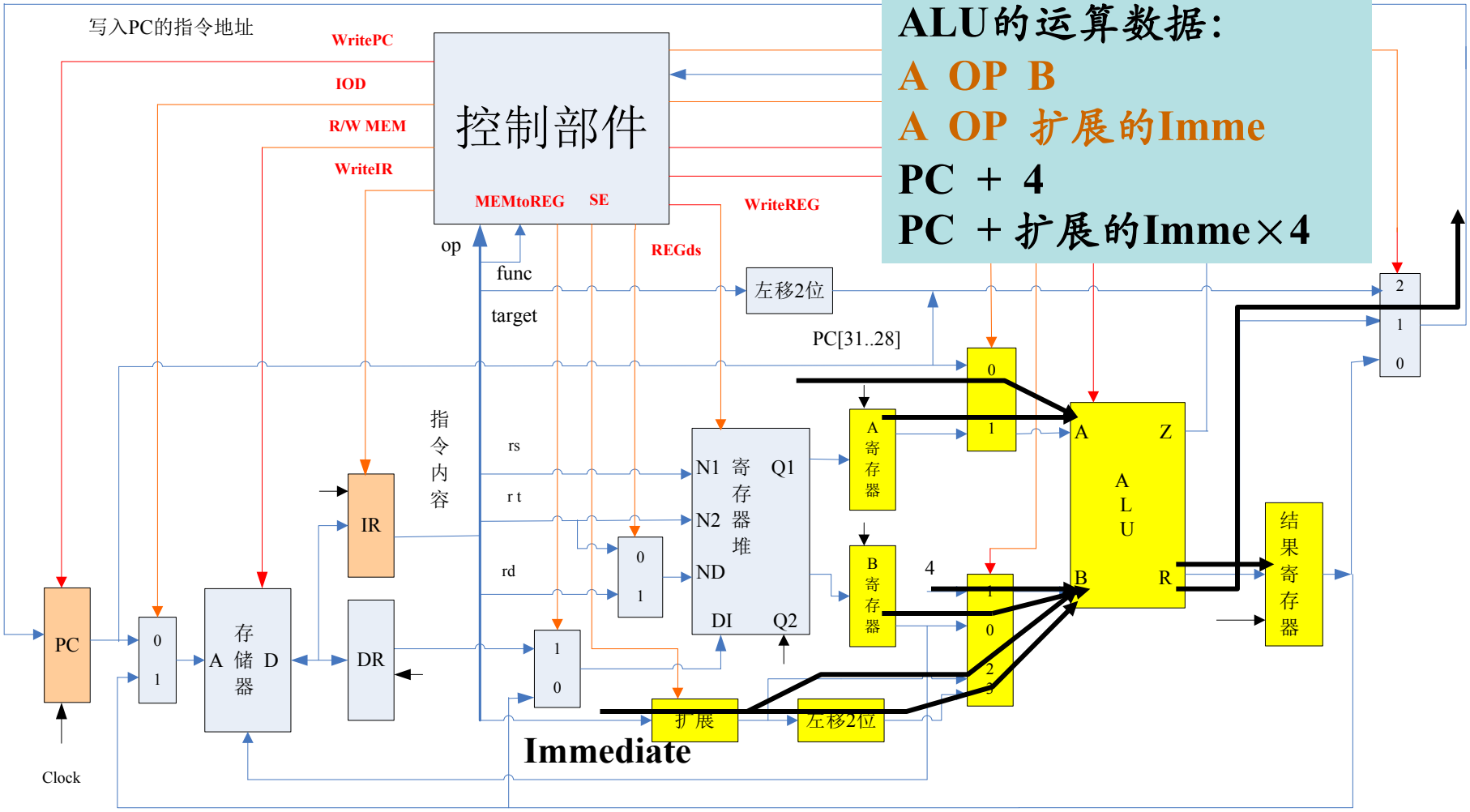
存储器既存指令又存数据。读指令时由 PC 提供地址，读出的指令保存到 IR；读写数据时由结果寄存器提供地址，读操作的读出数据保存到 DR， 写操作的写入数据由 B寄存器给出。



寄存器堆由32个寄存器组成，可以用N1(rs)、N2(rt)同时读出两个寄存器的内容，分别存于A、B寄存器，可以用 ND (rd 或 rt) 把 DI 端的数据写入，被写入数据来自结果寄存器 或 DR。



ALU完成算术和逻辑运算，两路输入分别为A和B，其中A路输入可选择A寄存器或PC，B路输入可选择B寄存器或常数4、IR.immediate经符号扩展或扩展后又左移两位的值。



MIPS机指令执行步骤

清華大學 Tsinghua University

R:寄存器型

op	rs	rt	rd	shamt	funct

I:立即数型

op	rs	rt	address / immediate

J: 跳转型

op	target

MIPS机的取指操作可1步完成，在取指之后，

J 型指令用 PC 高4位 拼接 Target 可以只经过1步完成，

相对转移 (I型) 指令经读寄存器堆、ALU运算可2步完成，

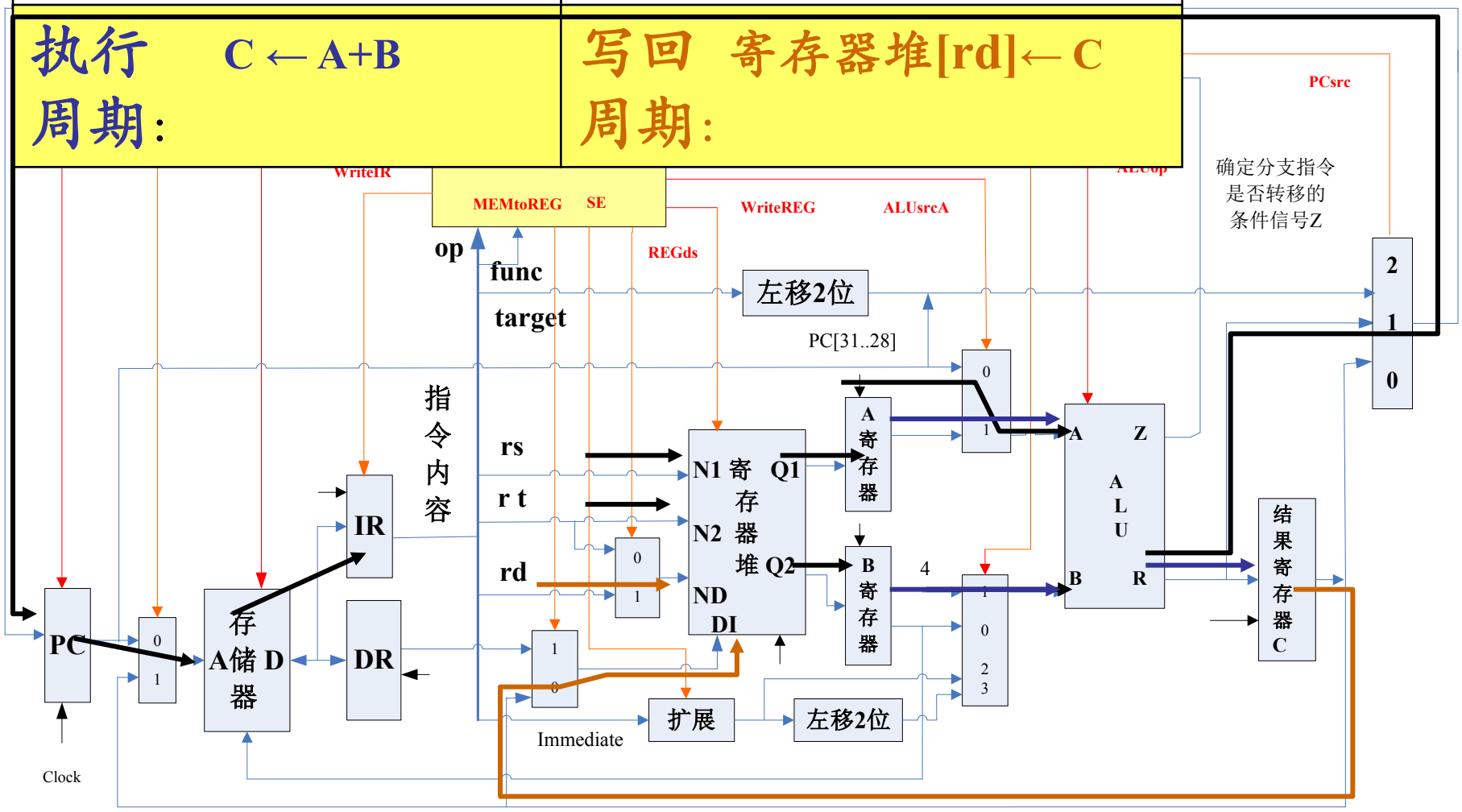
R 型指令经读寄存器堆、ALU运算和结果写回可3步完成，

读内存指令经读寄存器堆、ALU算地址、读内存数据到 DR、把 DR内容写入寄存器堆可4步完成。

MIPS机的 ADDU 指令 的执行过程

清華大學 Tsinghua University

取指 IR←MEM[PC] 周期： PC ←PC+4	译码 周期： A ← [rs] B ← [rt]
执行 C ← A+B 周期：	写回 寄存器堆[rd]← C 周期：



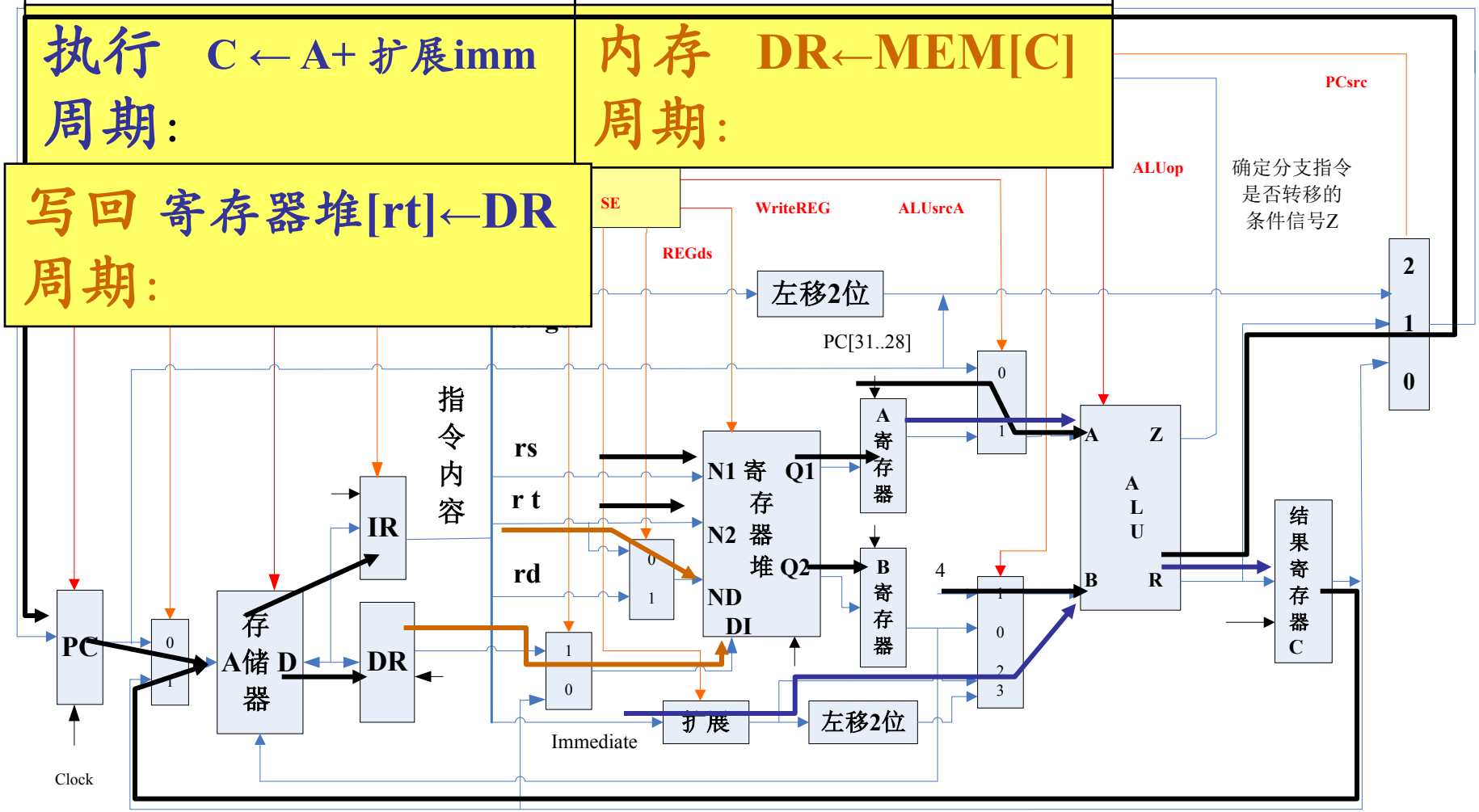
R型指令的实现(ADDU)

清華大學 Tsinghua University

- 取指令
  - IODR=0, ALUsrcA=0, ALUsrcB=01,ALUop=00, PCsrc=00
  - MEMread, IRwrite, PCwrite
- 译码/取操作数
  - ALUsrcA=0, ALUsrcB=11,ALUop=00
- 执行运算
  - ALUsrcA=0, ALUsrcB=00,ALUop=00
- 写回寄存器
  - RegDST=1,RegWrite, MemtoReg = 0

MIPS机的 LW 指令 的执行过程

取指 IR←MEM[PC] 周期： PC ←PC+4	译码 周期： A← [rs]
执行 C ← A+ 扩展imm 周期：	内存 DR←MEM[C] 周期：
写回 寄存器堆[rt]←DR 周期：	



MIPS机的 BEQ 指令 的执行过程

取指 IR←MEM[PC]

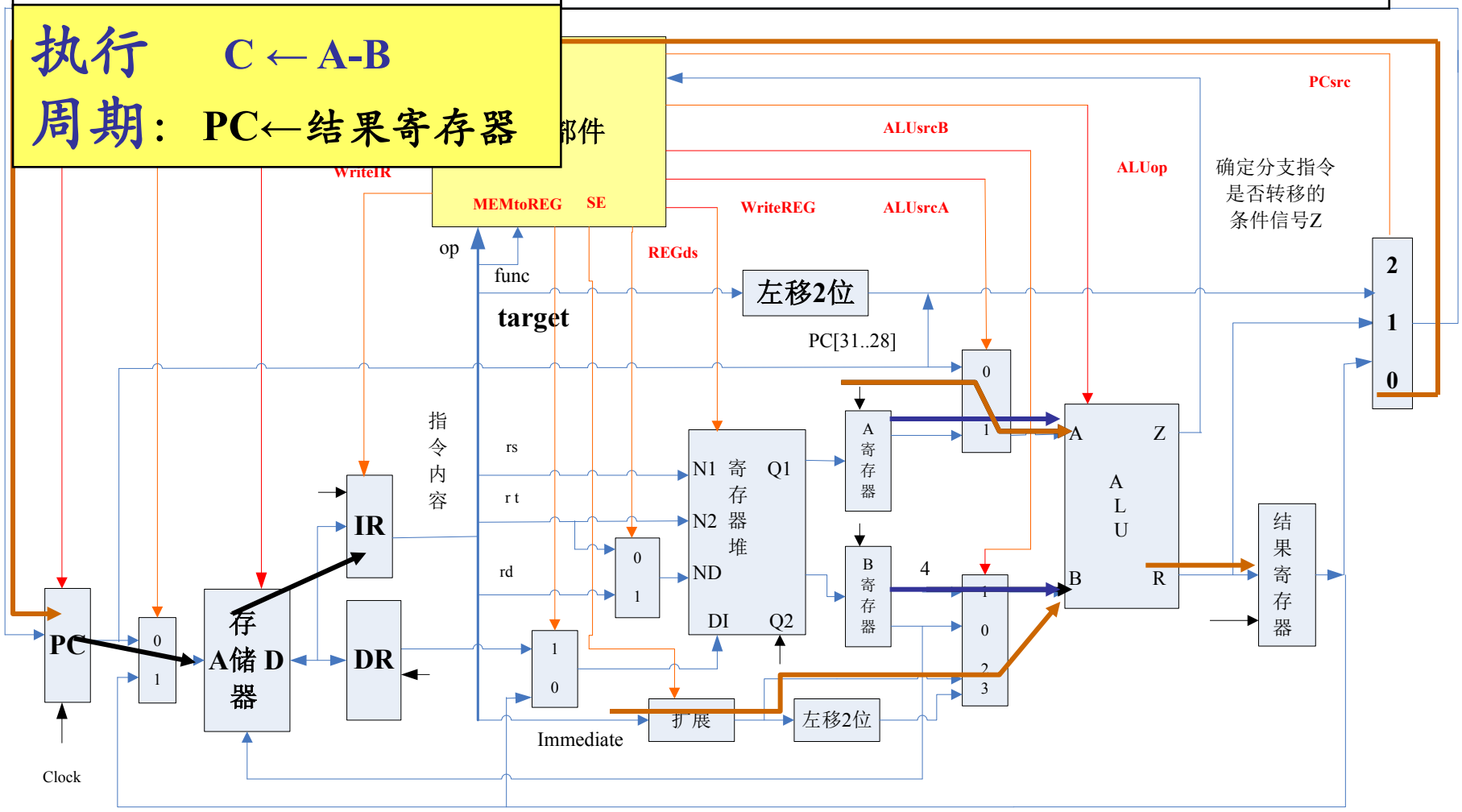
周期： PC ←PC+4

译码

周期： 结果寄存器←PC +SignExt(imm)

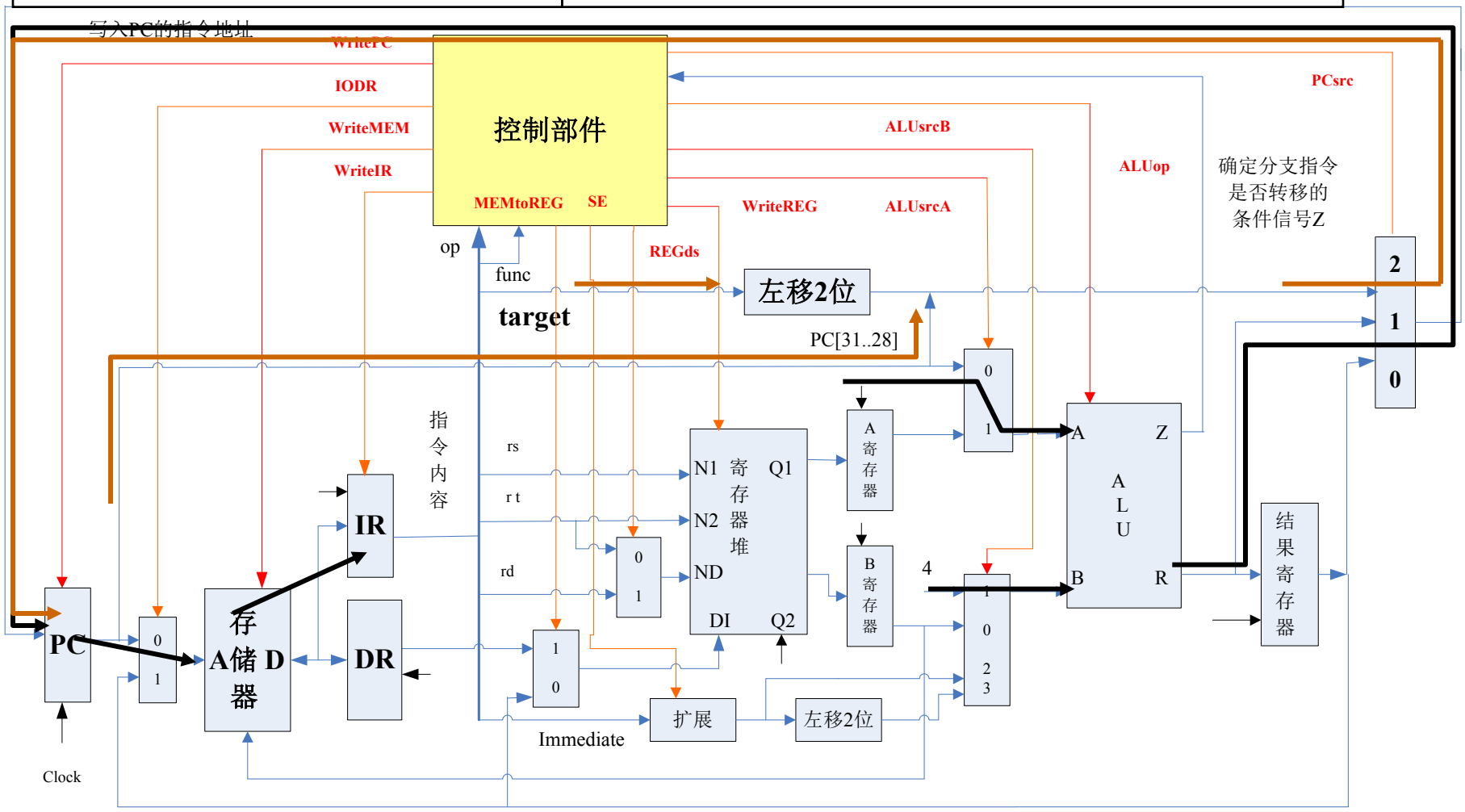
执行 C ← A-B

周期： PC←结果寄存器

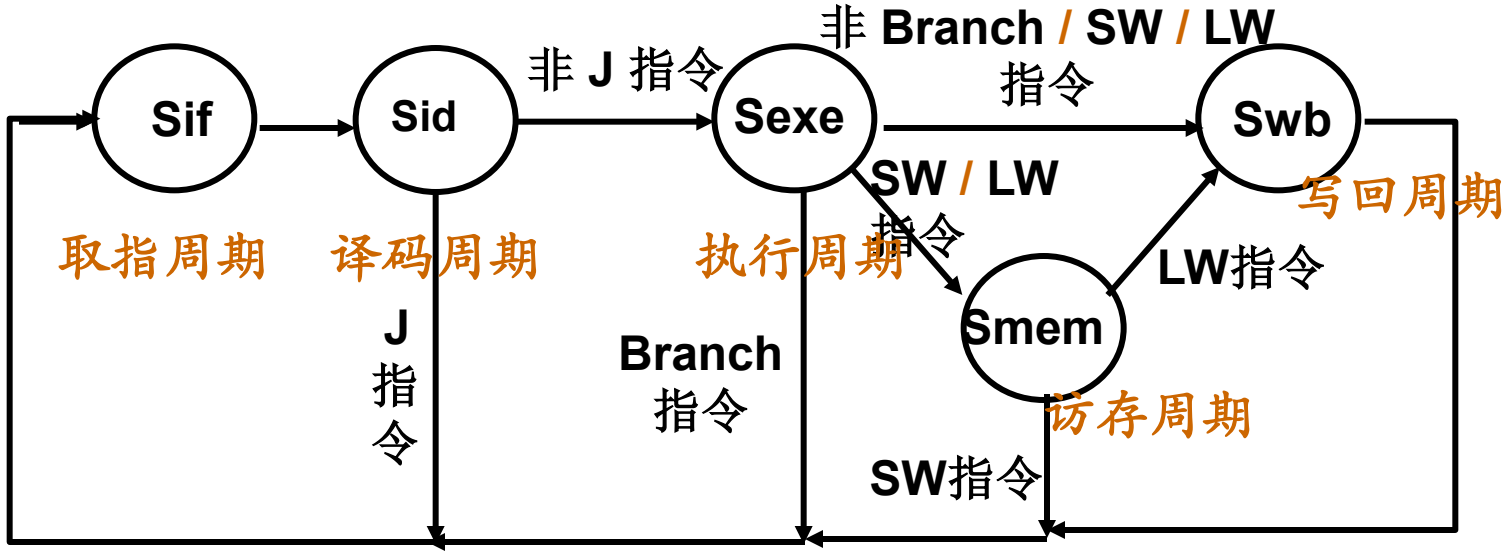


MIPS机的 J 指令 的执行过程

取指 IR←MEM[PC]	译码
周期： PC ←PC+4	周期： PC ←PC[31..28] || target 《 2

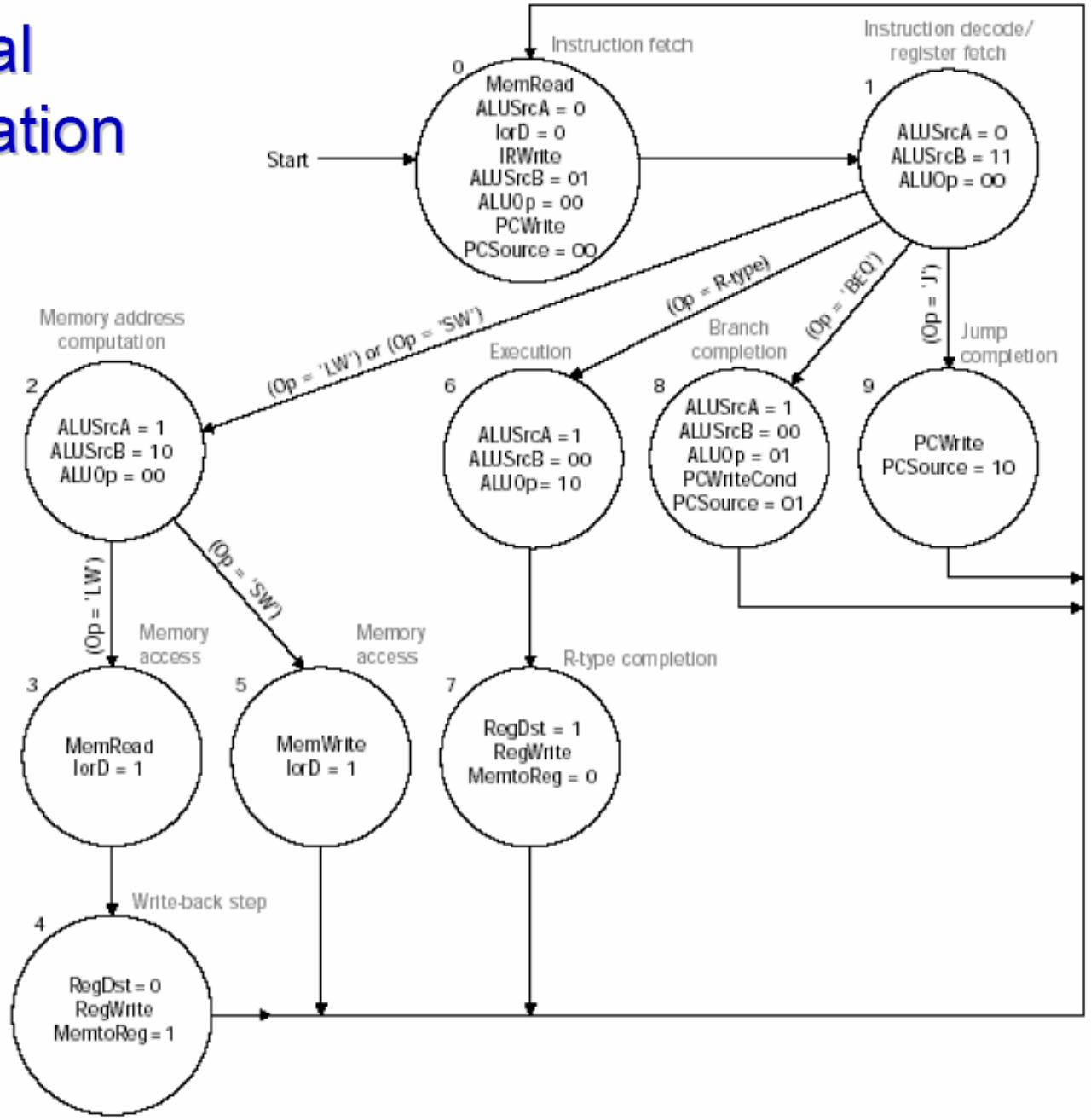


状态转移图和指令各执行步骤的操作功能

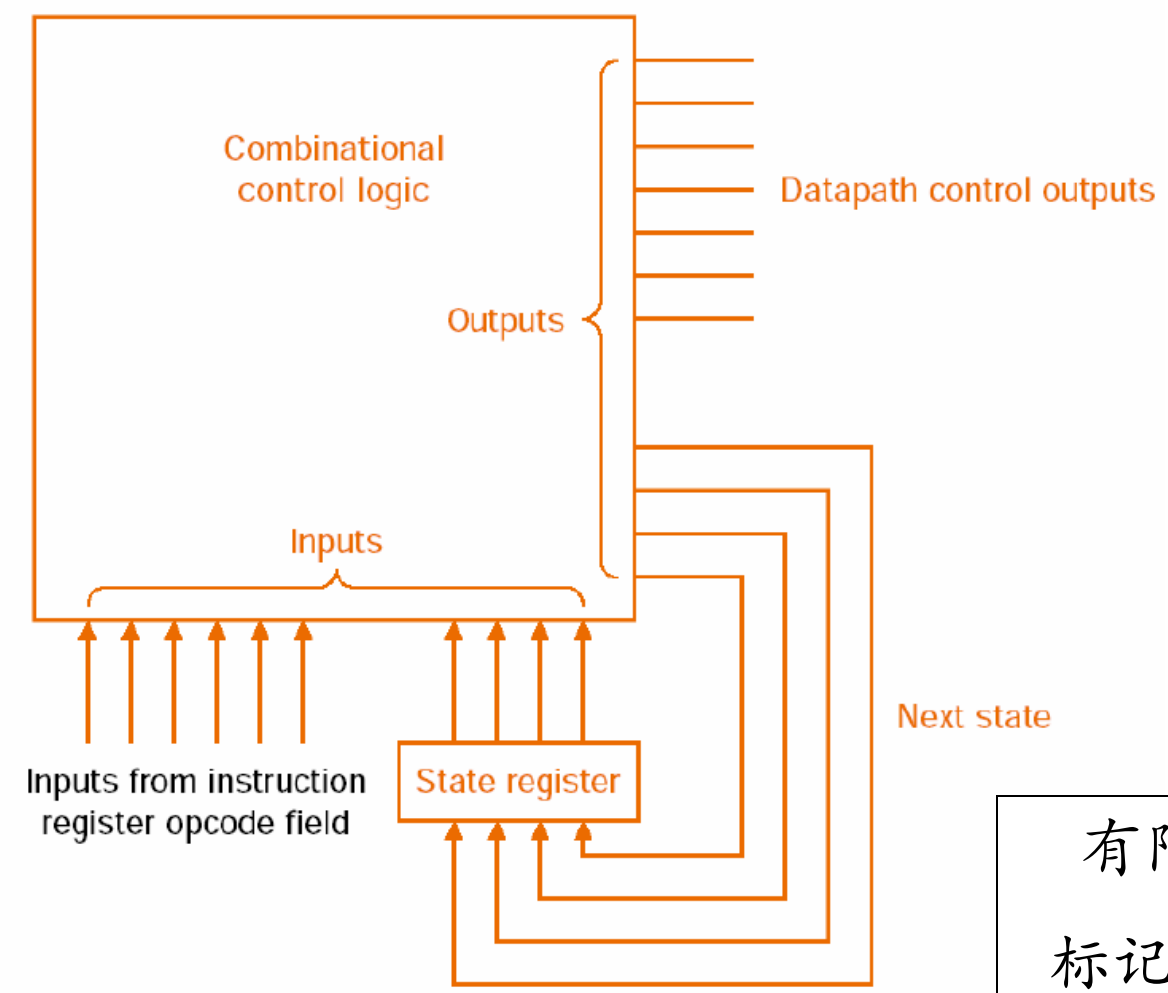


指令步骤	读取指令	指令译码	执行运算	内存读写	数据写回
J 指令	IR←MEM[PC] PC ← PC+4	PC←PC[31..28]‖(target<<2)			
Branch型		C←PC+(符号扩展(imm)<<2)	若条件成立 则 PC←C		
R 类型			C←A op B		Reg[rd]←C
Sw指令		A←Reg[rs] B←Reg[rt]	C←A + 符号扩展(Imm)	Mem[C]←B	
Lw指令				DR←Mem[C]	Reg[rt]←DR

Graphical Specification of FSM



FSM Implementation



有限状态机

标记执行步骤

7

实现示例

```
entity Controler_seven is
    Port ( rst : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           clk0: in STD_LOGIC;
           instructions : in  STD_LOGIC_VECTOR (15 downto 0);
           bZero_ctrl: in std_logic);
end Controler_seven;
```

实现示例

清華大學
Tsinghua University

```
architecture Behavioral of Controler_seven is
 signal bzero : std_logic ;
type controler_state is (instruction_fetch,decode,execute,mem_control,write_reg);
signal state : controler_state ;
signal PCWrite : std_logic ;
signal PCWriteCond : std_logic ;
signal PCSource : std_logic ;
signal ALUOp : std_logic_vector(1 downto 0) ;
signal ALUSrcA : std_logic ;
signal ALUSrcB : std_logic_vector(1 downto 0) ;
signal MemRead : std_logic ;
signal MemWrite : std_logic ;
signal IRWrite : std_logic ;
signal MemtoReg : std_logic_vector(1 downto 0) ;
signal RegWrite : std_logic_vector(2 downto 0) ;
signal RegDst : std_logic_vector(1 downto 0) ;
signal IorD : std_logic ;
```

实现示例

清華大學 Tsinghua University

```
architecture Behavioral of Controler_seven is
 signal bzero : std_logic ;
type controler_state is (instruction_fetch,decode,execute,mem_control,write_reg);
signal state : controler_state ;
signal PCWrite : std_logic ;
signal PCWriteCond : std_logic ;
signal PCSource : std_logic ;
signal ALUOp : std_logic_vector(1 downto 0) ;
signal ALUSrcA : std_logic ;
signal ALUSrcB : std_logic_vector(1 downto 0) ;
signal MemRead : std_logic ;
signal MemWrite : std_logic ;
signal IRWrite : std_logic ;
signal MemtoReg : std_logic_vector(1 downto 0) ;
signal RegWrite : std_logic_vector(2 downto 0) ;
signal RegDst : std_logic_vector(1 downto 0) ;
signal IorD : std_logic ;
; 后续代码
```

小结

- 多周期CPU
  - 按指令的执行步骤，每个步骤占用一个CPU周期
  - 不同指令的指令周期不同
  - 指令串行执行
  - 提高了整体性能
- 进一步的改进
  - 各部件的利用率依然偏低
    - 指令并行执行？是否可行？如何进行？

阅读和思考

清華大學
Tsinghua University

- 阅读
  - 教材第4章
  - 有关参考书
- 思考
  - 多周期CPU的特点，并与单周期CPU比较
  - 如何实现指令流水?