CISC's CPU功能与结构

李曦@中科大11系

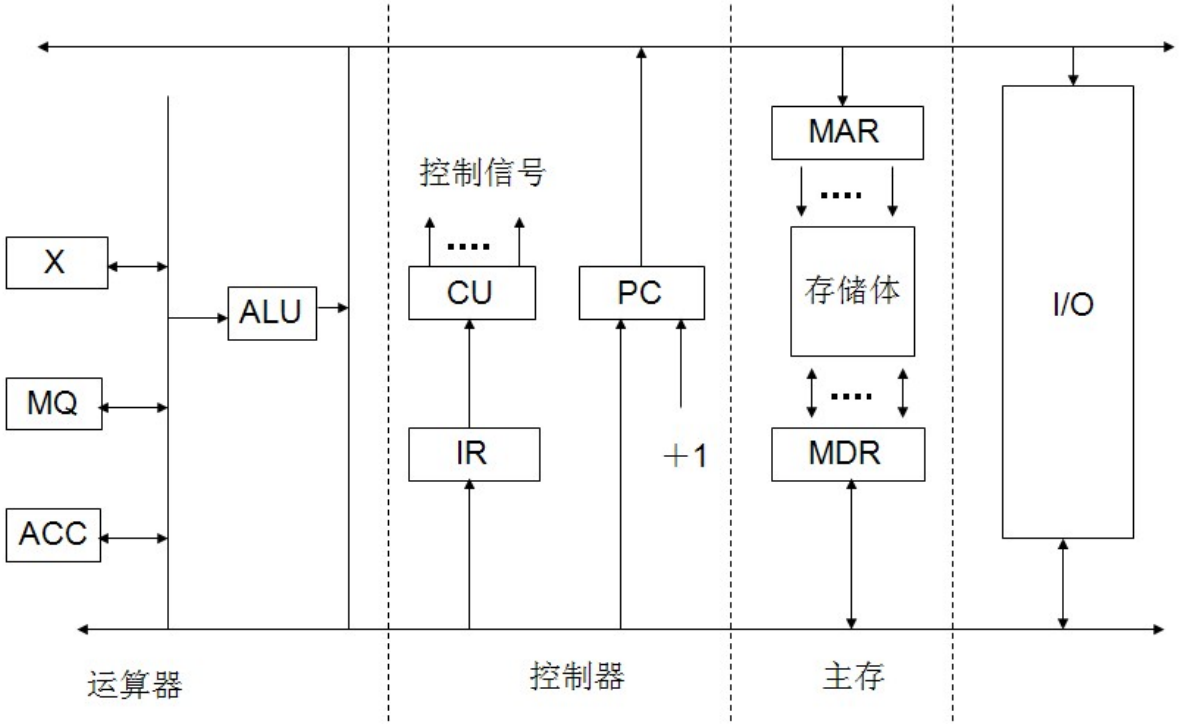
内容

- CPU功能
- CPU的结构
- 指令的执行过程分析
  - 时序

- William Stallings2010, “Computer Organization and Architecture” , 8th Edition
  - 第9版这部分内容online！

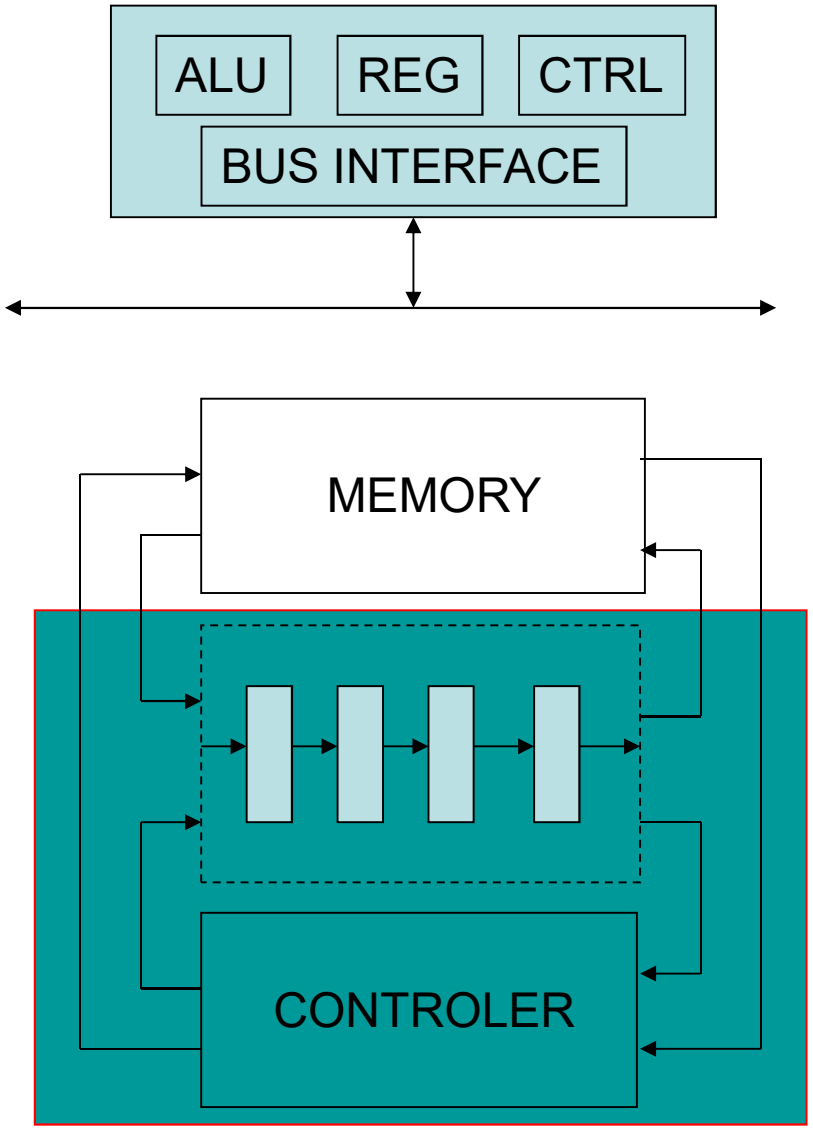
CPU功能

- 解释程序的指令，完成数据加工，产生系统控制信号
  - 取指：从内存的指令段中读取指令
  - 译码：对**op**和地址域进行分析
  - 执行：根据指令产生各种控制信号完成数据加工



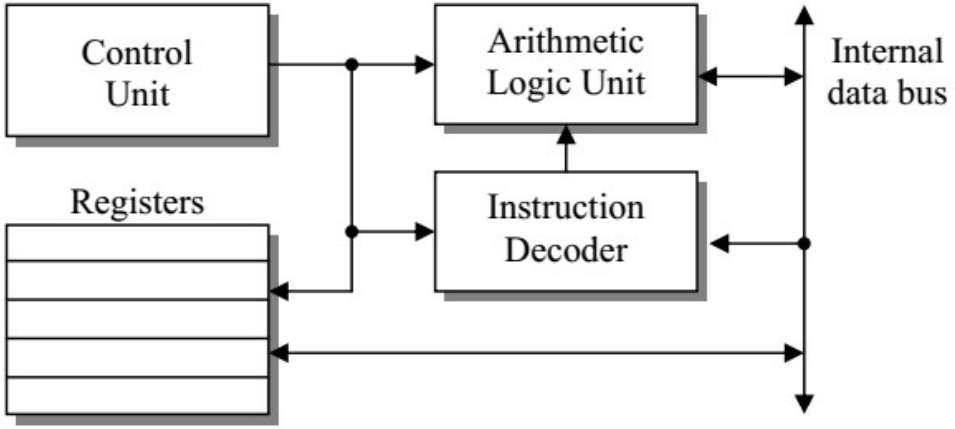
CPU的组成

- 按功能部件划分
  - 运算器（**ALU**）
  - 寄存器（寄存器堆）
  - 控制器（含中断控制）
  - 系统总线接口
  - ***MMU***
  - ***L1 CACHE***
  - 等
- 按数据流
  - 数据通路（**datapath**）：各种寄存器和运算器
  - 控制器：组合电路控制逻辑、微程序控制器

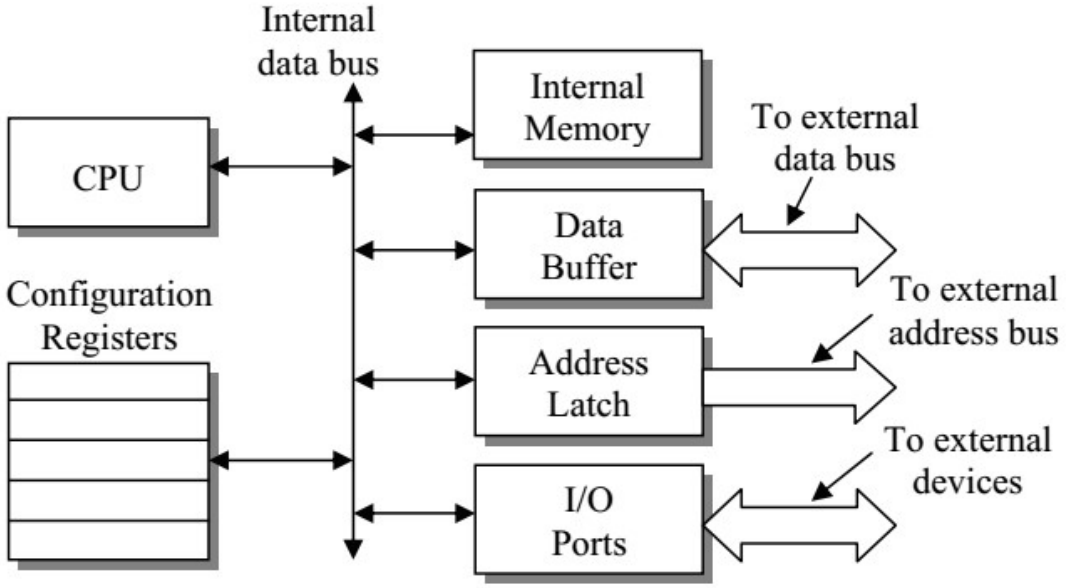


functional requirements for a processor

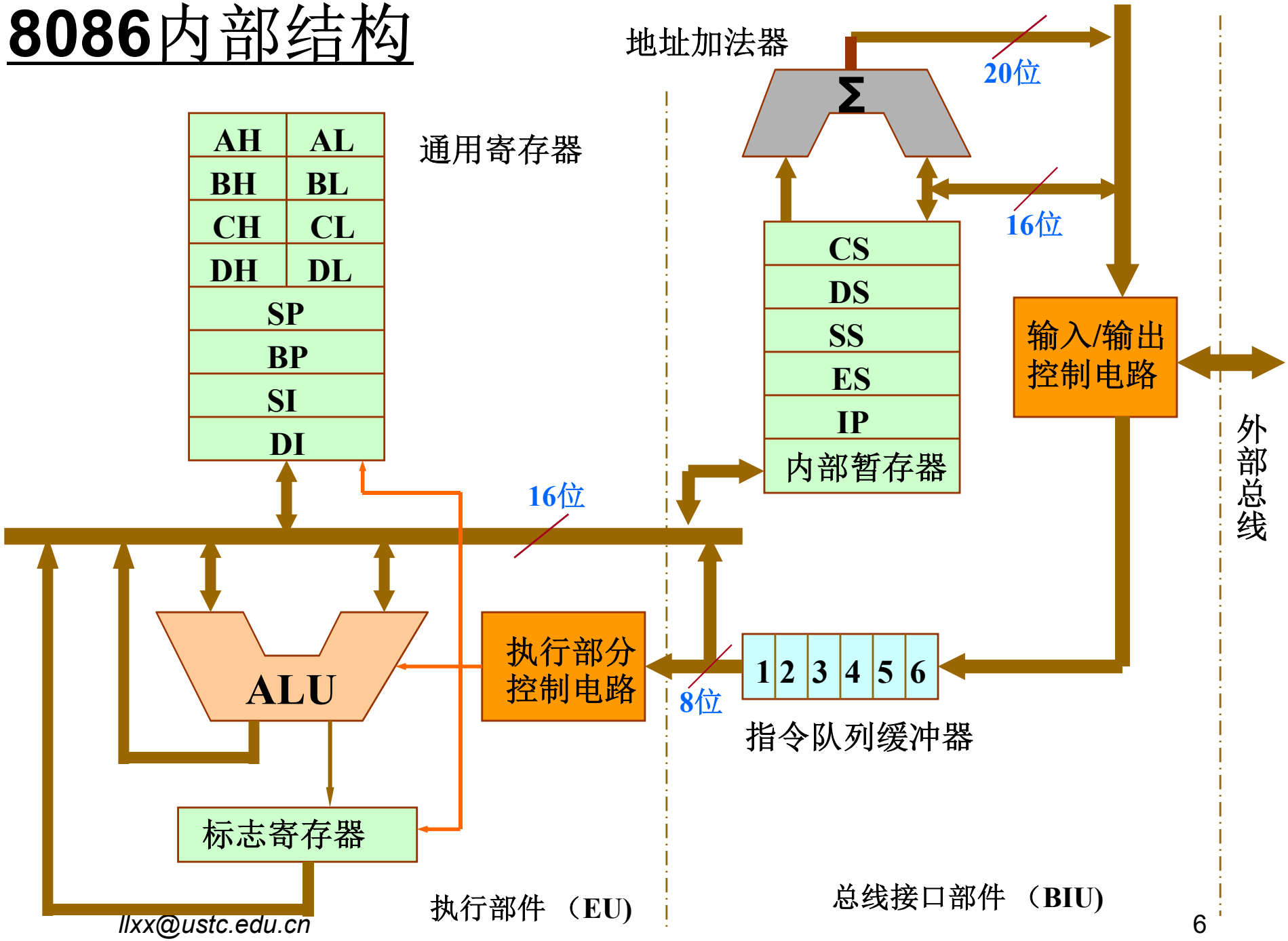
- Operations (opcodes)
- Addressing modes
- Registers



- Memory module interface
- I/O module interface
- Interrupts

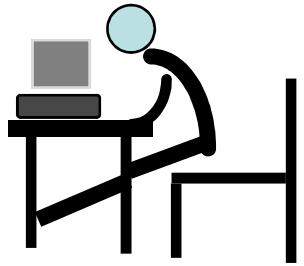


8086内部结构


地址加法器
Σ
20位
16位
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
SI
DI
通用寄存器
CS
DS
SS
ES
IP
内部暂存器
输入/输出
控制电路
外部总线
16位
ALU
执行部分
控制电路
1 2 3 4 5 6
8位
指令队列缓冲器
标志寄存器
执行部件 （EU)
总线接口部件 （BIU)

CPU中的寄存器

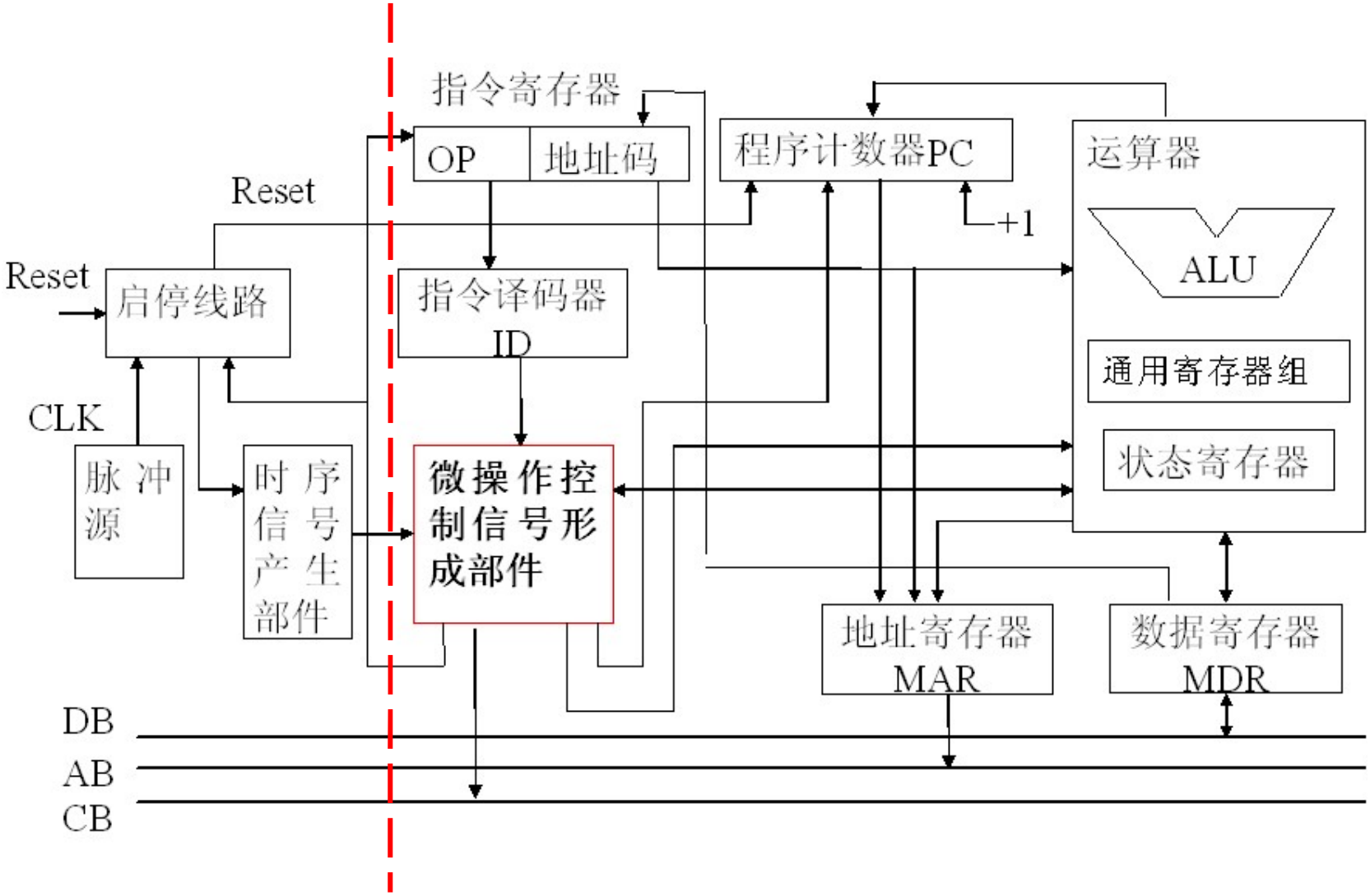
- 通用寄存器：运算
  - **X86：AX/BX/CX/DX 、CS/DS/SS/ES、SP/BP/SI/DI、**
- 状态寄存器：
  - **flag （F）、PSW**（工作方式、中断允许）
- 控制寄存器：
  - **PC（IP）、IR、MAR、MDR**
- 用户可见和不可见
  - **IR、MAR、MDR**等不可见
- 不同系统配置各不相同



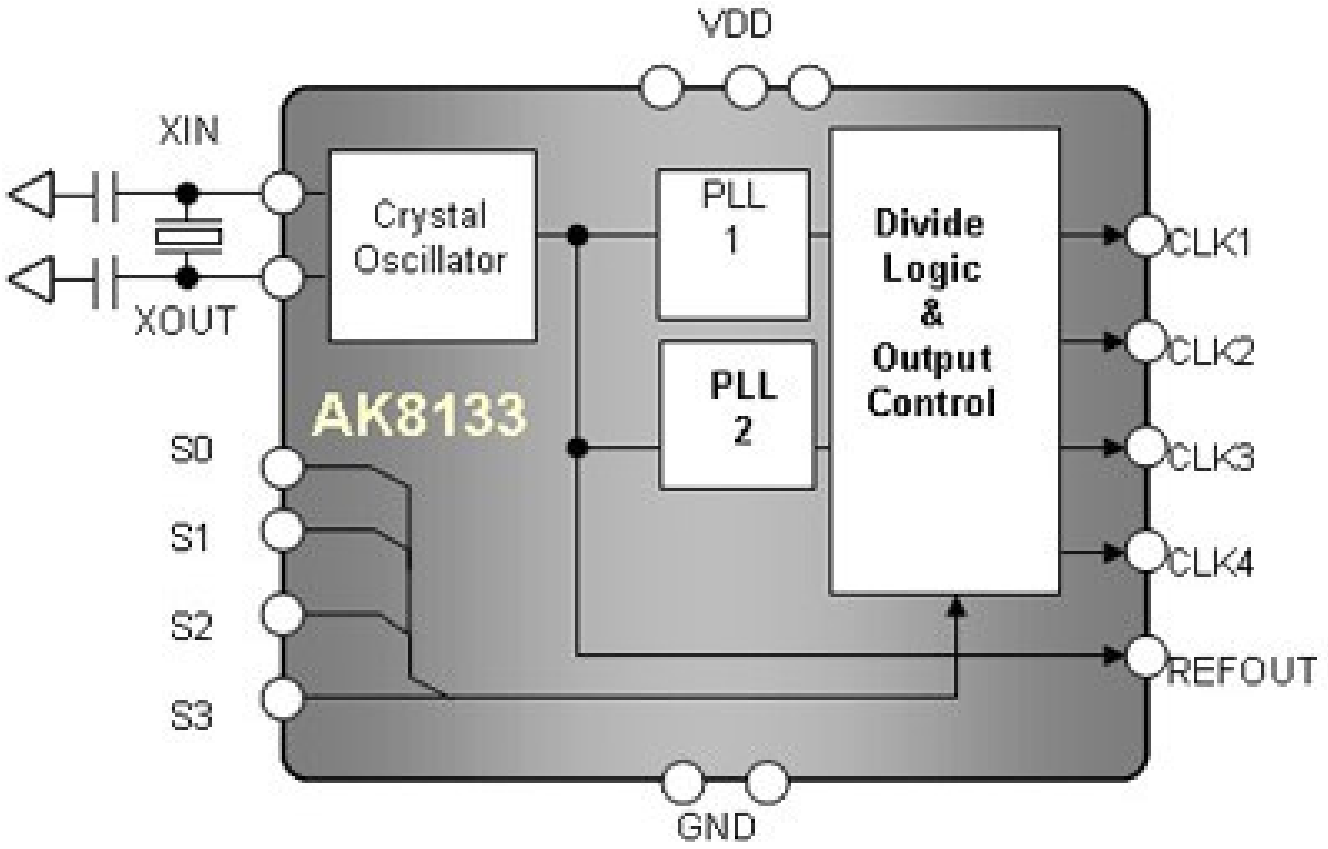
寄存器组织举例

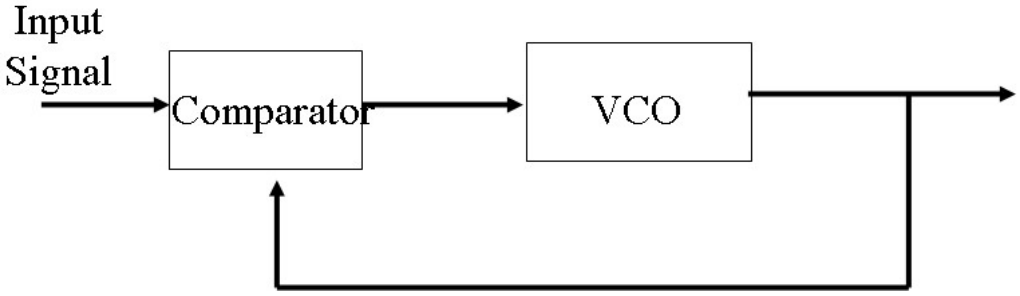

通用寄存器
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14 SP
15 SP
程序状态
标记控制字
PC 段
PC 偏移
P.S.A 段
P.S.A 偏移
(a) Z8000
通用寄存器
AX
BX
CX
DX
指针和变址
SP
BP
SI
DI
段
CS 代码
DS 数据
SS 堆栈
ES 附加
程序状态
IP 指令指针
F 标志
(b) 8086
数据寄存器
D0
D1
D2
D3
D4
D5
D6
D7
地址寄存器
A0
A1
A2
A3
A4
A5
A6
A7 用户栈指针
A7' 管理栈指针
程序状态
PC
状态
(c) MC 68000

定时


指令寄存器
OP
地址码
程序计数器PC
运算器
Reset
+1
ALU
Reset
启停线路
指令译码器
ID
通用寄存器组
CLK
脉冲源
时序信号产生部件
微操作控制信号形成部件
状态寄存器
地址寄存器
MAR
数据寄存器
MDR
DB
AB
CB

时钟脉冲发生器

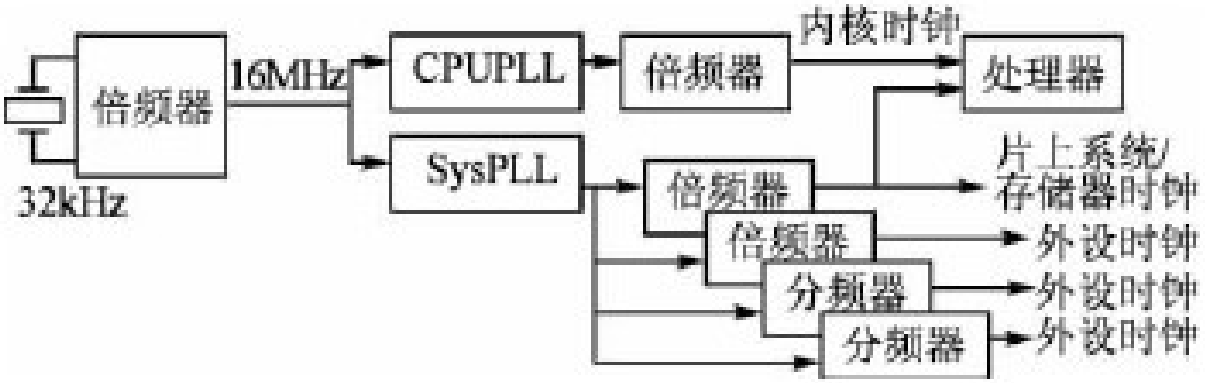




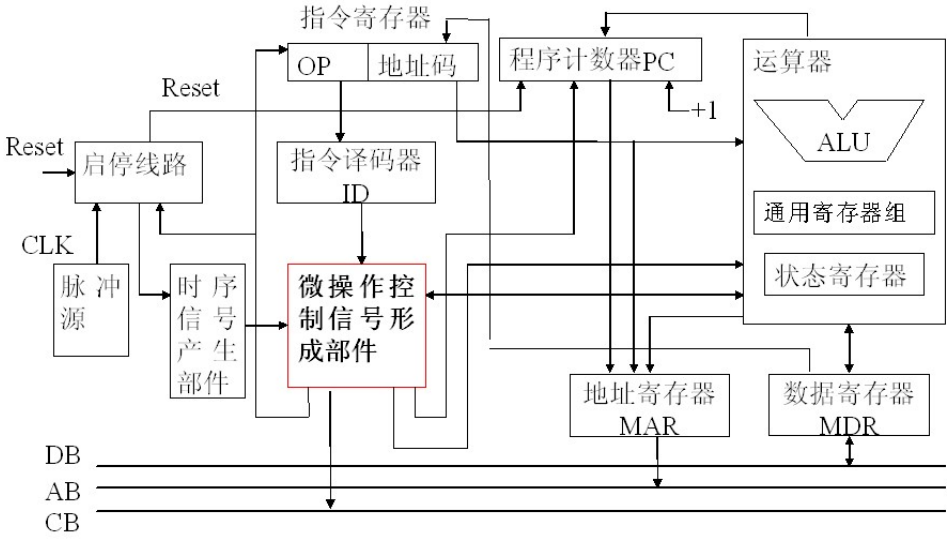
PLL：Freq controlled by applied input voltage

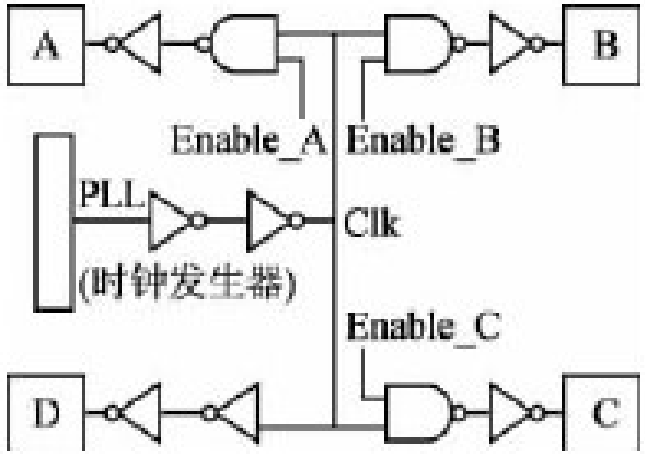
VCO = Voltage Controlled Oscillator

系统时钟



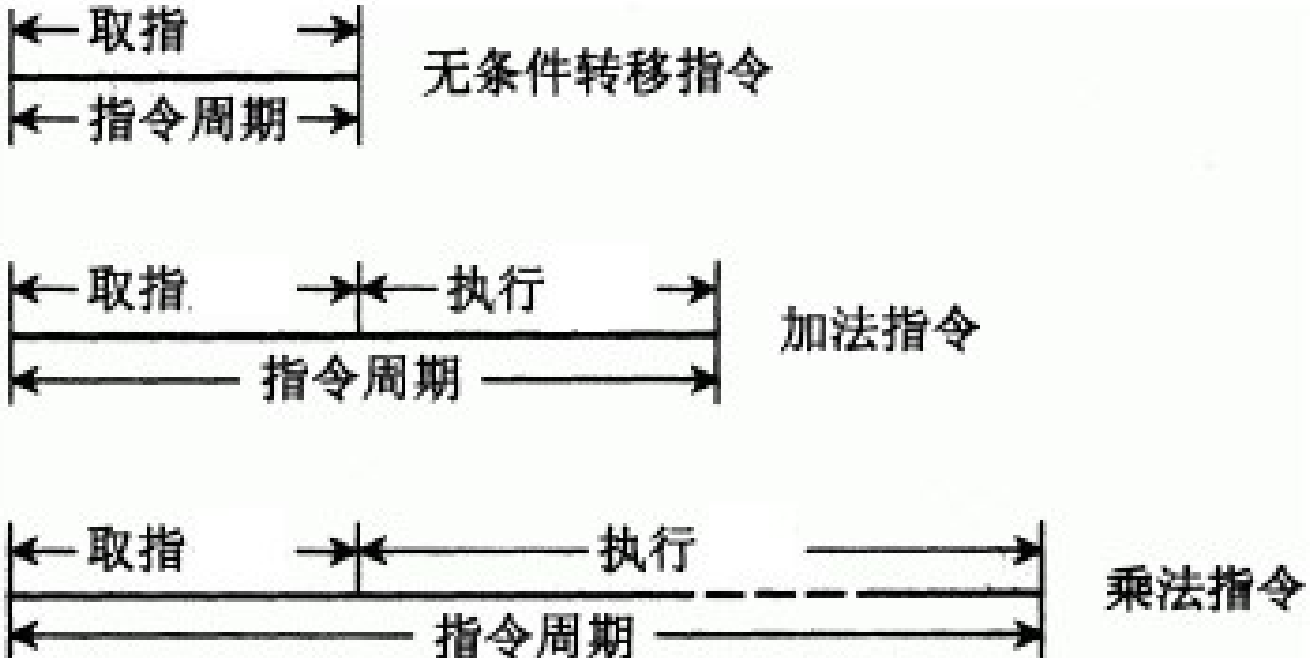
- 与门门控时钟（启停控制）：**HLT**指令？





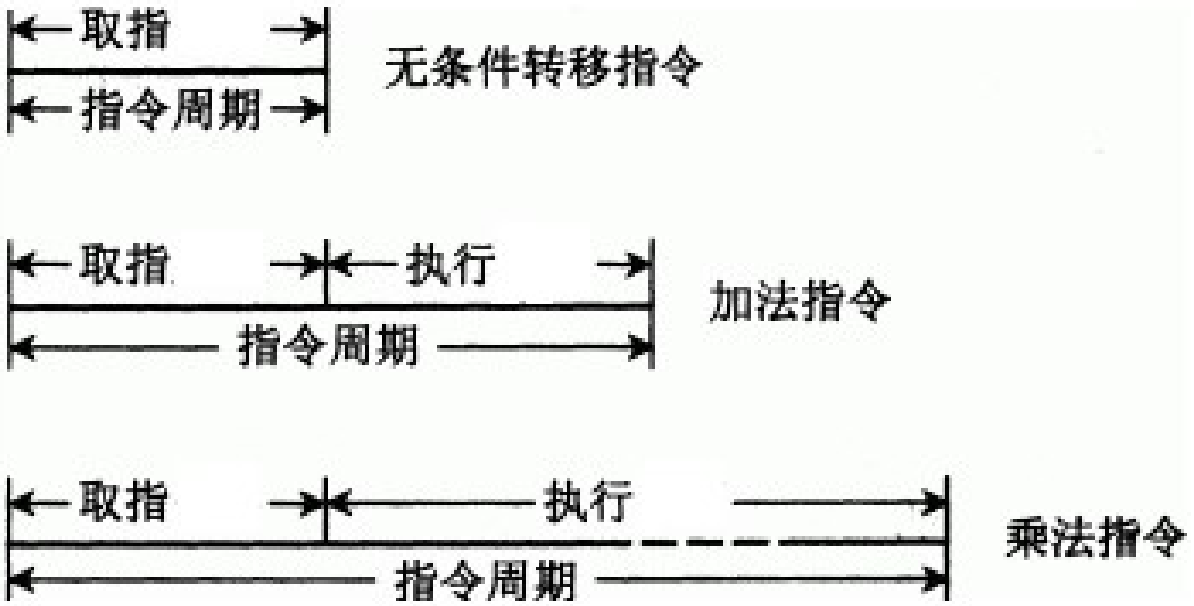
指令周期

- 执行一条指令所需的时间
  - 指令执行的过程是顺序完成各个机器周期中的微操作的过程
  - 通常，不同指令所需时间不同（如乘法操作的执行周期较长）

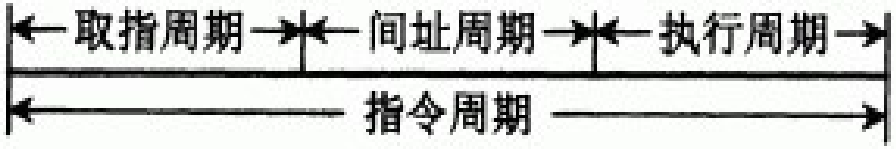


指令周期实现方式

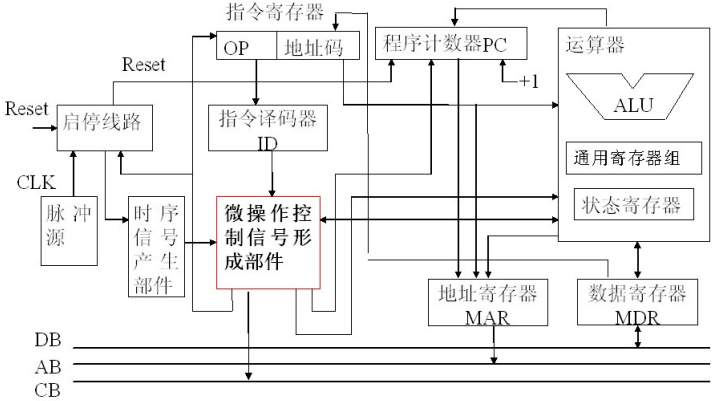
- 单周期实现
  - 定长单周期：一个周期一条指令
    - 周期宽度以数据通路最长的指令为准
  - 不定长单周期：一个周期一条指令，周期宽度各个指令不同

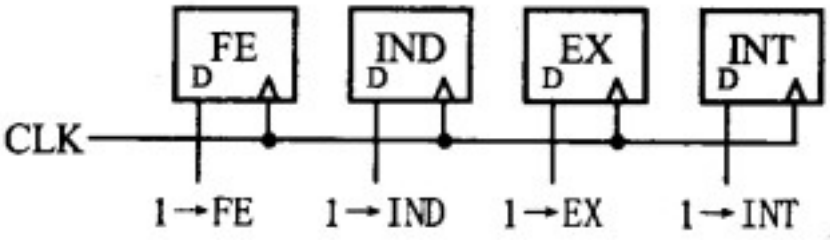


指令周期实现方式（cont）



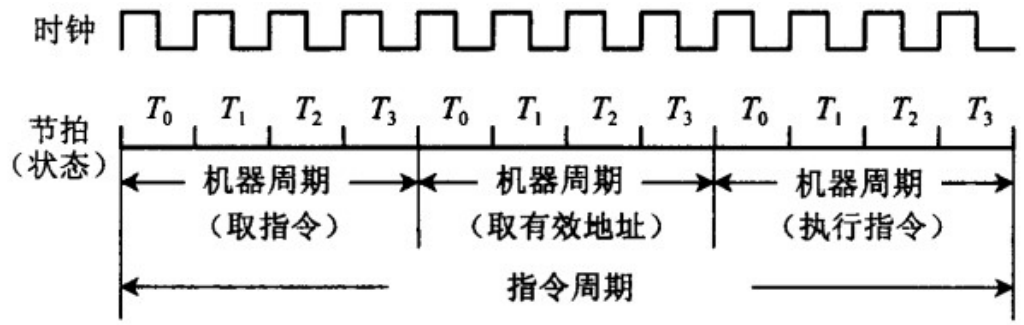
- 多周期实现
  - 多个CPU工作周期（机器周期）构成
    - 取指周期FE
    - 间址周期IND：间接寻址指令访存取出有效地址
    - 执行周期EX
    - 中断周期INT
- 时序信号产生部件
  - CPU中设置标志触发器指示当前的工作周期



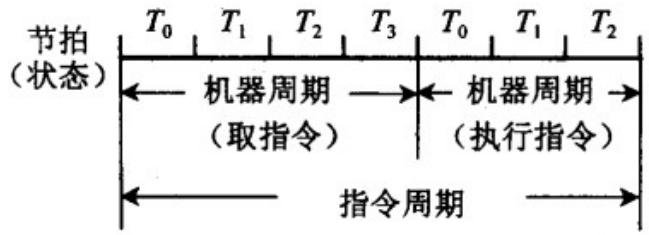


机器周期实现方式

- 定长机器周期：所有机器周期的节拍数相等
- 不定长机器周期：不同机器周期的节拍数不等
  - 如：不同指令的执行周期的节拍数可以不等



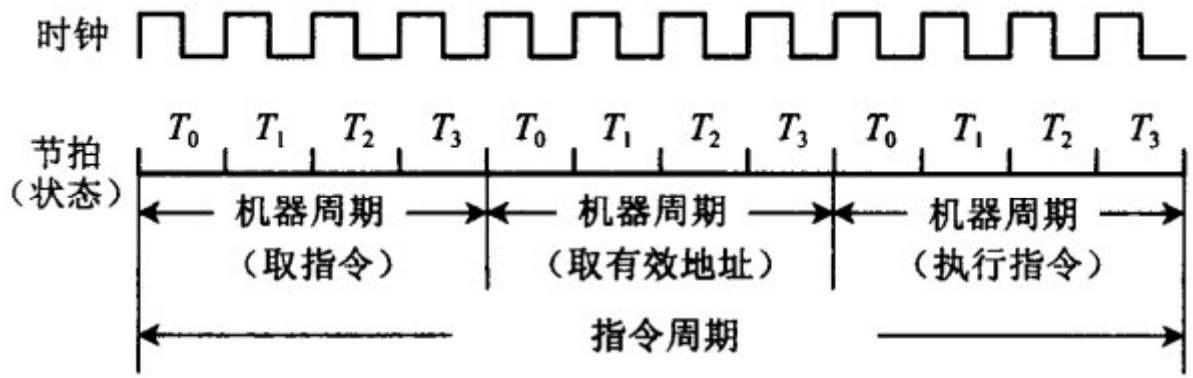
（a） 定长的机器周期



（b） 不定长的机器周期

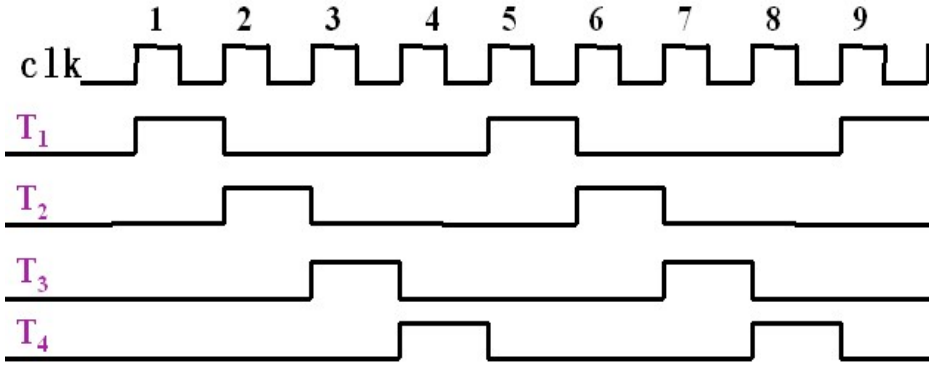
多级时序系统

- 指令周期＝n个机器周期
- 机器周期＝n个节拍（状态）
- 节拍＝n个时钟周期（clk）= 1 个clk？



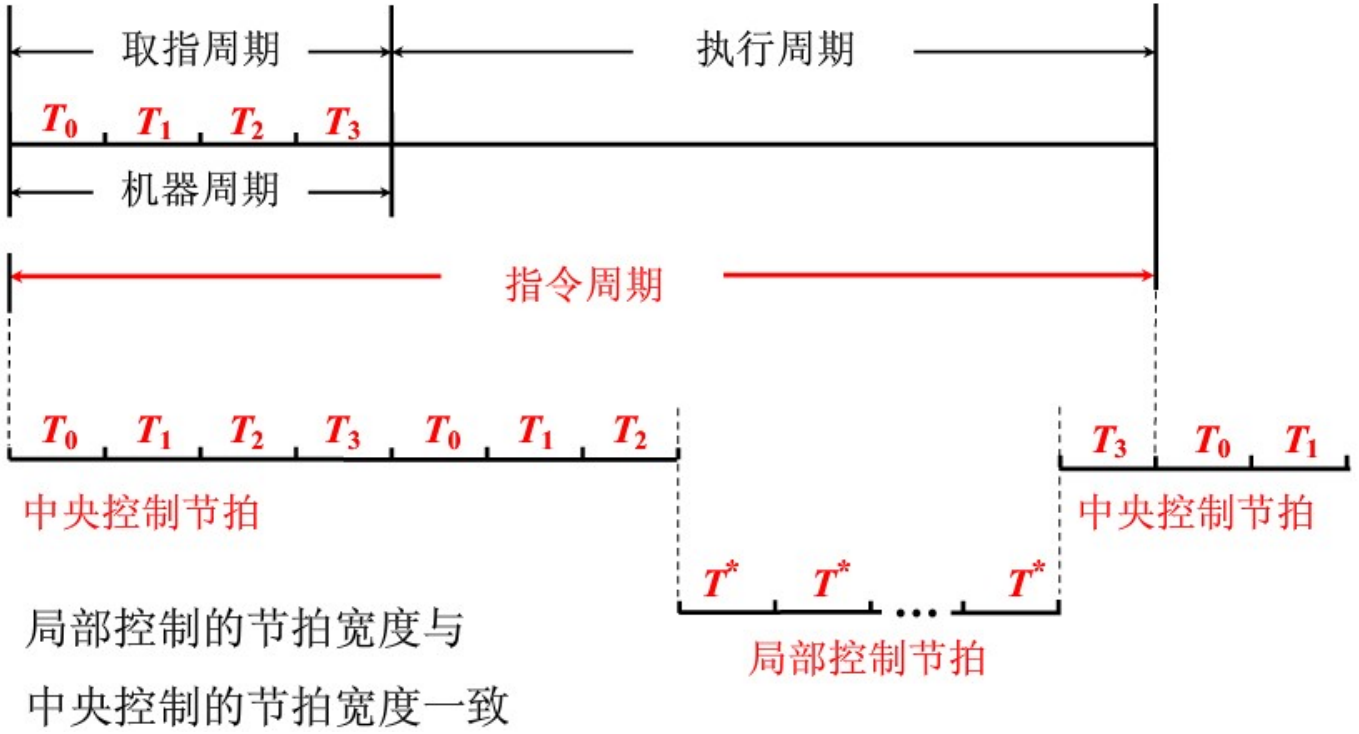
（a） 定长的机器周期

（b） 不定长的机器周期



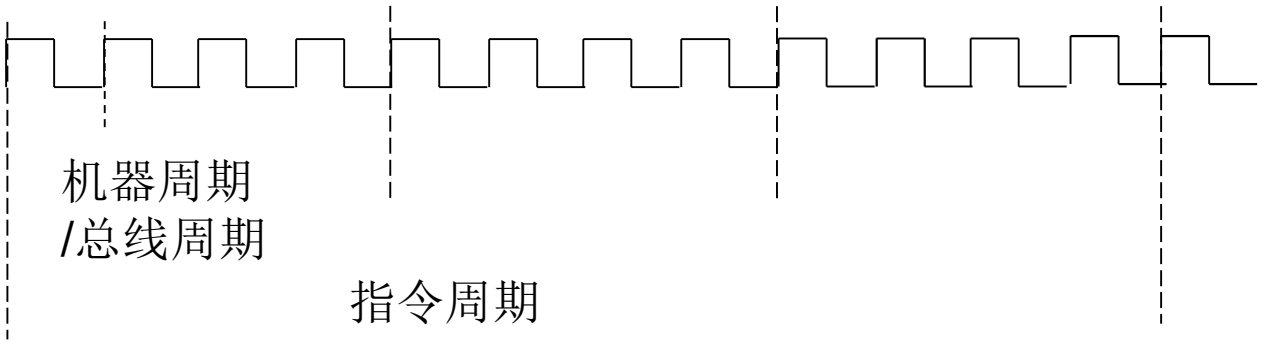
时钟同步控制方式

- 中央控制：多数指令的机器周期等长
- 局部控制：少数复杂指令的某些操作进行局部控制
  - 局部时钟与全局时钟要同步（保持一定的比例关系）

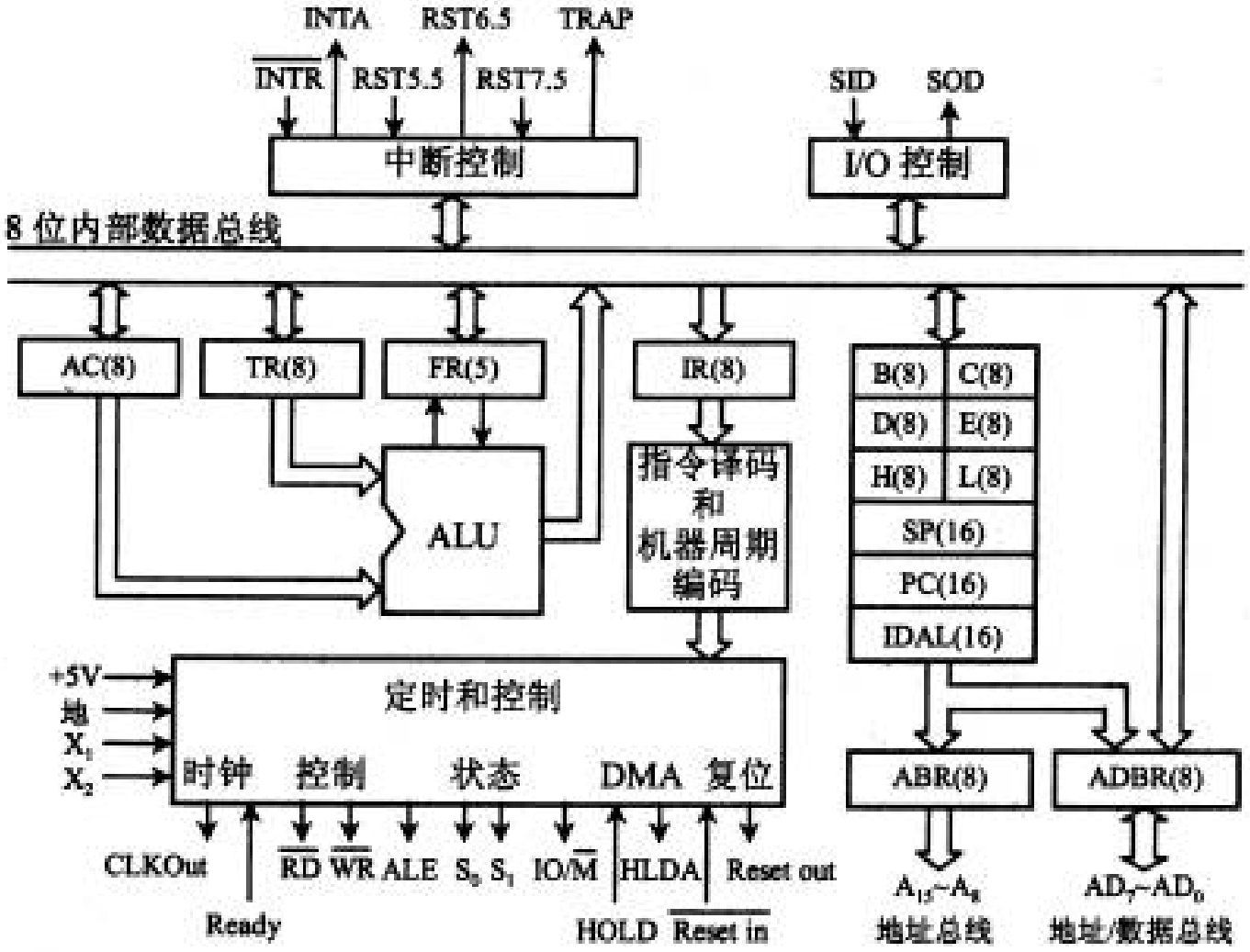


注意一些概念

- 指令周期 **vs.** 机器周期 **vs. CPU**工作周期 **vs.** 总线周期
  - 机器周期（**CPU**工作周期）：取指、间址、执行
  - 时钟周期（**T1/T2/T3/T4/Tw**。。）、节拍、状态
- 机器性能：程序执行时间**P=**指令数×**CPI**×**T**
  - 程序中的指令数由**ISA**和编译器确定
  - **CPI**为平均值：由处理器的实现技术和任务特征确定

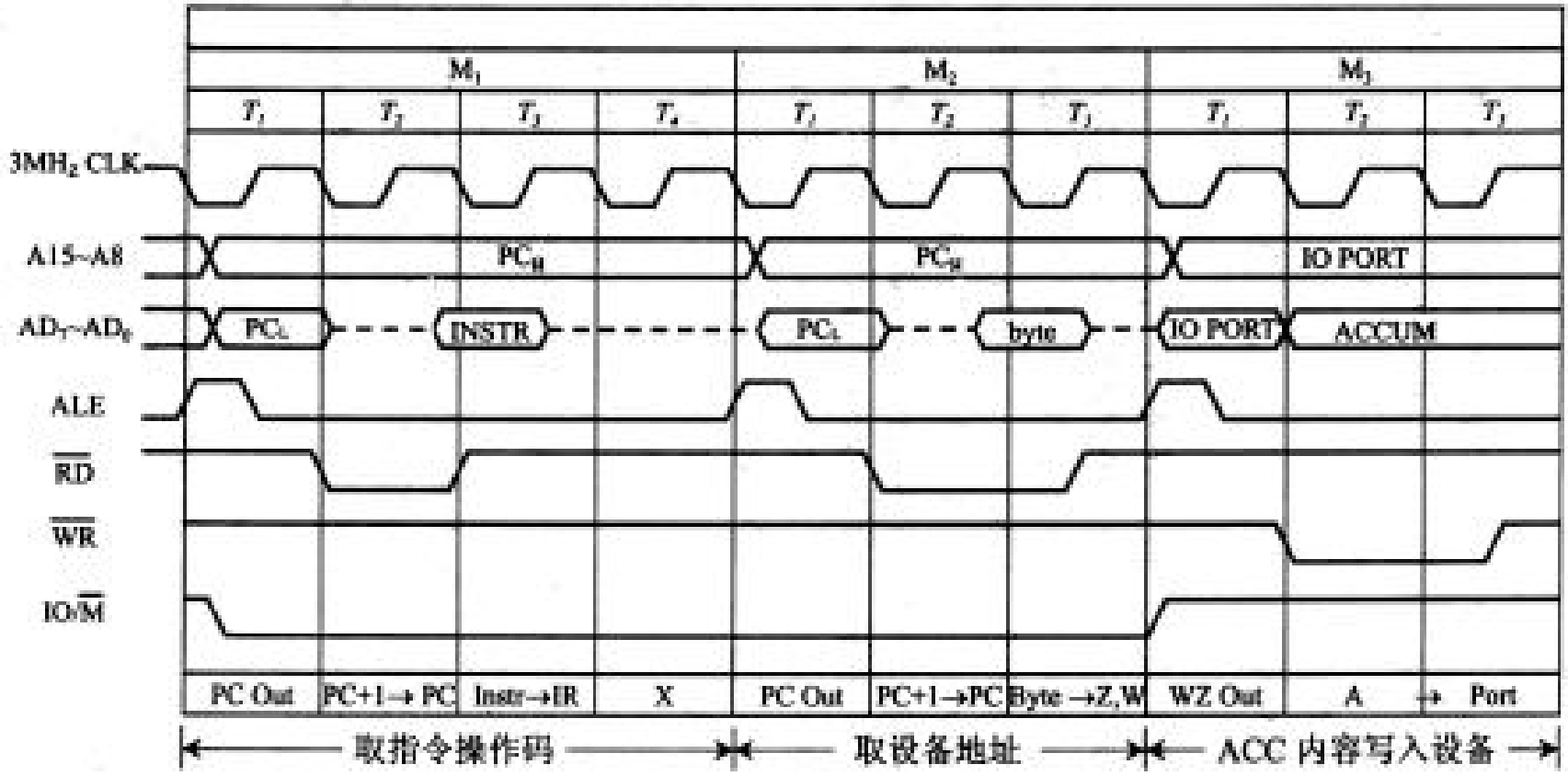


Intel 8085结构



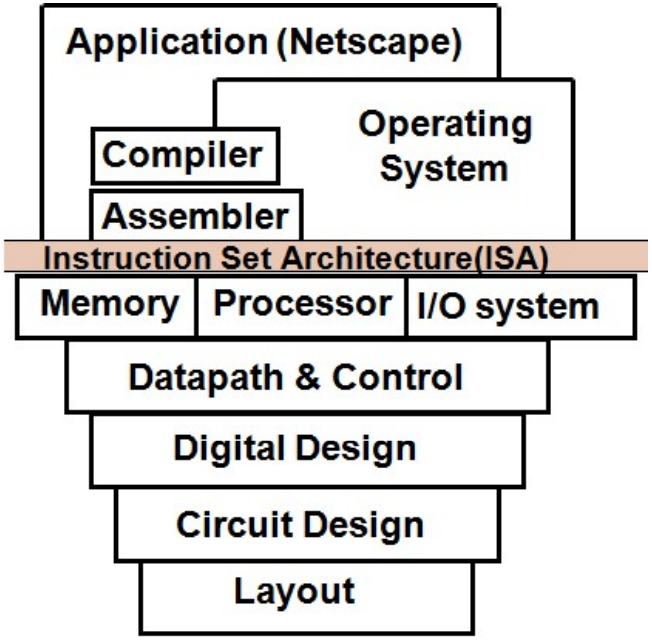
8085指令时序（P392）

- **out AC，xx**；输出指令，将**AC**写入设备**xx**
- 指令字长**16**位，总线**8**位
- 每条指令**3**～**5**个机器周期，每个机器周期**3**～**5**个节拍



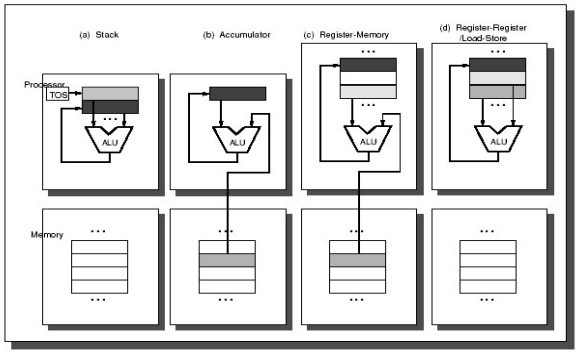
控制器设计思路

- 目标
  - 生成各个节拍所需的控制信号
- 步骤（6步）
  - 确定**ISA**
  - 确定处理器的微结构
    - 一个**ISA**可以有多种实现方案
  - 确定时序体系
    - 采用多级时序系统
  - 分析每条指令在各个机器周期的微操作
  - 将每条指令的各个微操作分配到各个节拍
  - 逻辑综合
    - 将所有指令在各个节拍的微操作进行综合，生成各个节拍所需的控制信号，据此产生控制逻辑



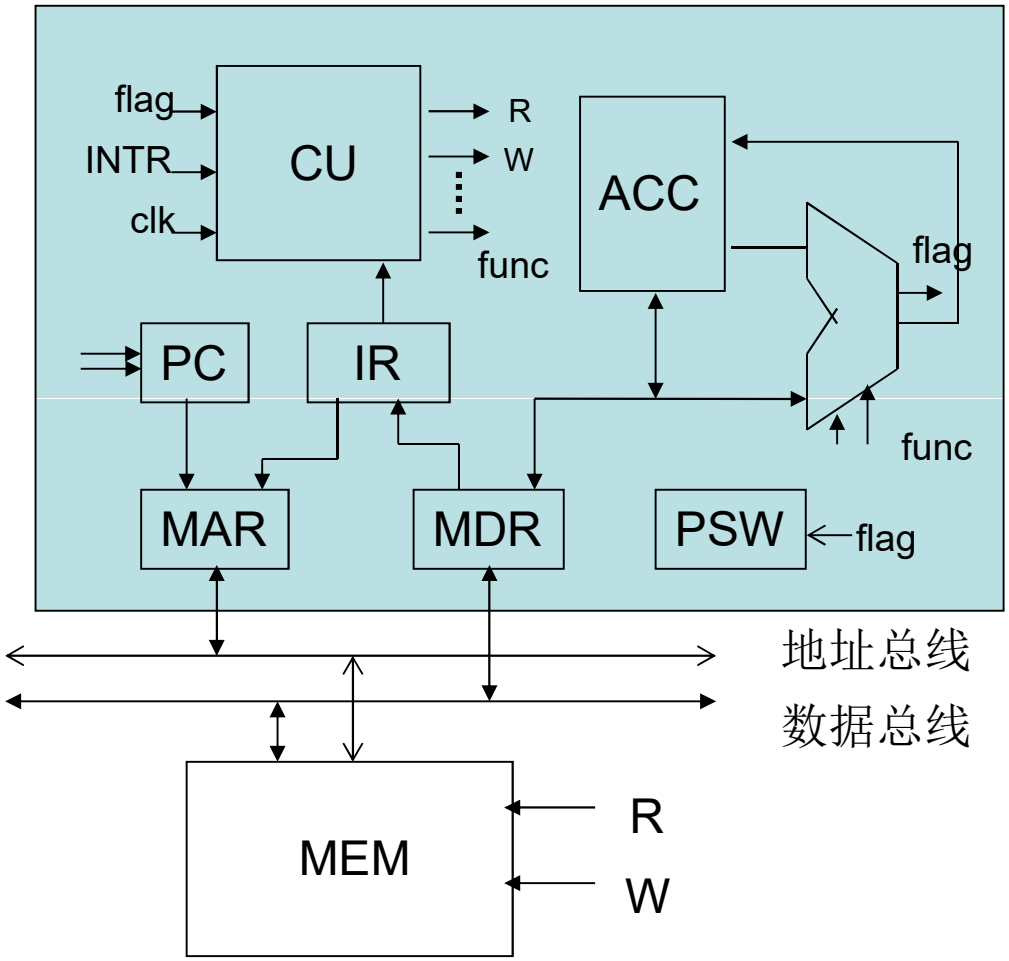
A模型计算机的ISA

- 10条指令
  - 非访存指令：CLA, COM, SHR, CSL, STP
    - CLA: 清ACC
    - COM: ACC取反
    - SHR: ACC算术右移一位, 符号位不变
    - CSL: ACC循环左移一位
    - STP：停机，标志触发器G置“0”
  - 访存指令：ADD X, STA X, LDA X
    - ADD X: ACC与地址X的内容相加，结果在ACC中
    - STA X: 将ACC存入X中
    - LDA X: 将X中的内容读入ACC
  - 转移指令：JMP X, BAN X
    - JMP X: 无条件转移至X
    - BAN X: 如果前一条指令的执行结果为负
- “memory-based operand-addressing mode”
  - opr在存储器和ACC中（也称“累加器型”）



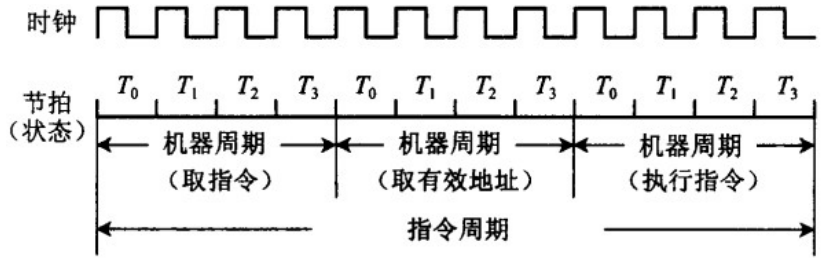
A模型计算机的ISA（con't）

- **Accumulator Arch**
  - Single Accumulator
  - 允许一个操作数在MEM中
  - 无堆栈？
- 指令字格式：一地址?
- 操作数寻址方式
  - 直接寻址
  - 间接寻址：@X
  - 寄存器寻址（Acc）
  - 隐含寻址
  - ？？？
- **PSW：NZ、EINT、STP**



指令周期的数据流与微操作

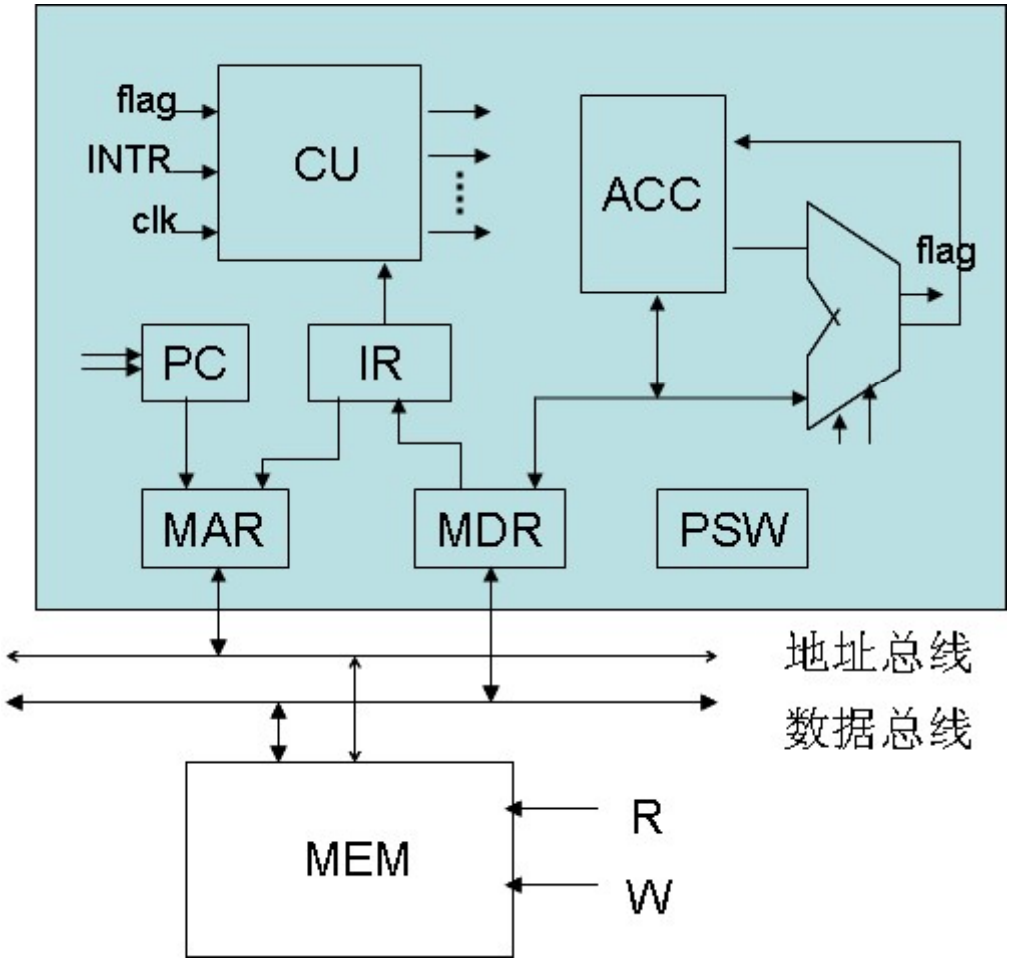
- **A模型CPU**的指令周期模式
  - 多（机器）周期实现方式
  - 定长机器周期
  - 不定长指令周期



- 数据流：指令执行时数据的流动
- 微操作：各个机器周期所包含的动作
  - 取指周期
  - 间址周期
  - 执行周期
    - 各个指令不同，包括取数、计算、写结果等

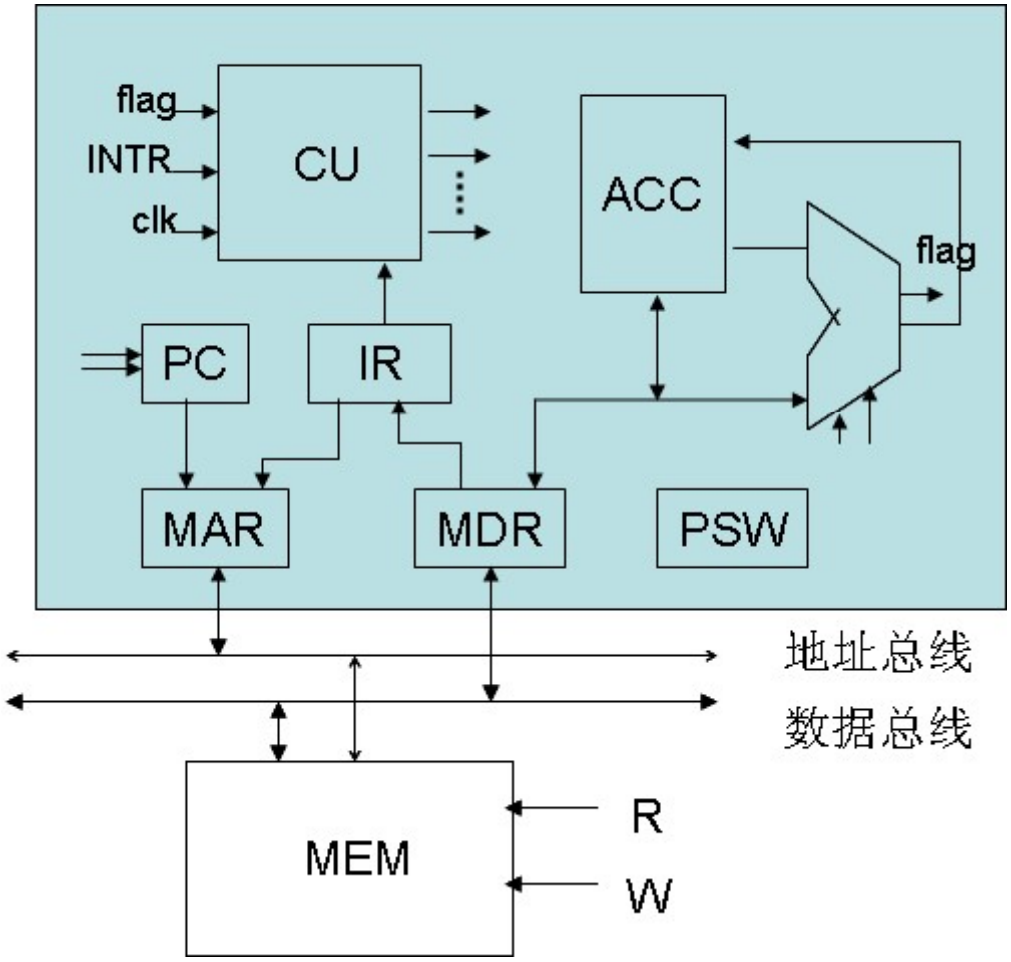
取指周期

- 根据**PC**从**MEM**中取指令
  - **PC->MAR**
  - **CU**向**MEM**发读令，**1->R**
  - 得到指令字，**M(MAR)-> MDR**
  - 指令字写入指令字寄存器，**MDR->IR**
  - **CU**控制形成下一条指令的地址，**PC+1->PC**



间址周期

- 取操作数的有效地址
  - 形式地址送往**MAR，Ad(IR)->MAR**
  - **CU**发读令，**1->R**
  - 得到有效地址，**M(MAR) ->MDR**
  - **MDR->Ad(IR)**

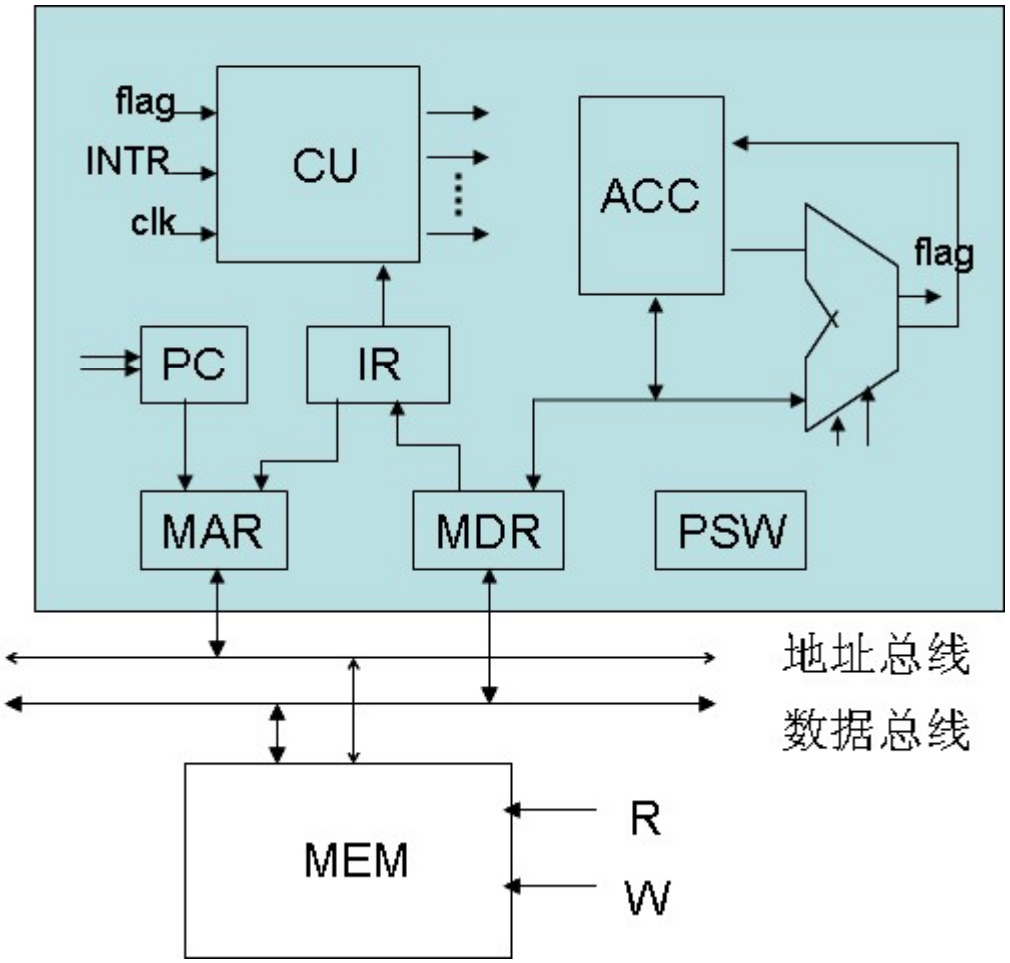


执行周期——非访存指令

- **CLA: 清ACC**
  - **0->ACC**
- **COM: ACC取反**
  - **/ACC->ACC**
- **SHR: ACC算术右移一位, 符号位不变**
  - **L(ACC)->R(ACC), ACC_0->ACC_0**
- **CSL: ACC循环左移一位**
  - **R(ACC)->L(ACC), ACC_0->ACC_n**
- **STP：停机，标志触发器G置“0”**
  - **0->G**

执行周期——访存指令

- **ADD X: ACC与地址X的内容相加，结果在ACC中**
  - **Ad(IR)->MAR**
  - **1->R**
  - **M(MAR)->MDR**
  - **ACC+MDR->ACC**
- **STA X: 将ACC存入X中**
  - **Ad(IR)->MAR**
  - **1->W**
  - **ACC->MDR**
  - **MDR->M(MAR)**
- **LDA X: 将X中的内容读入ACC**
  - **Ad(IR)->MAR**
  - **1->R**
  - **M(MAR) ->MDR**
  - **MDR ->ACC**

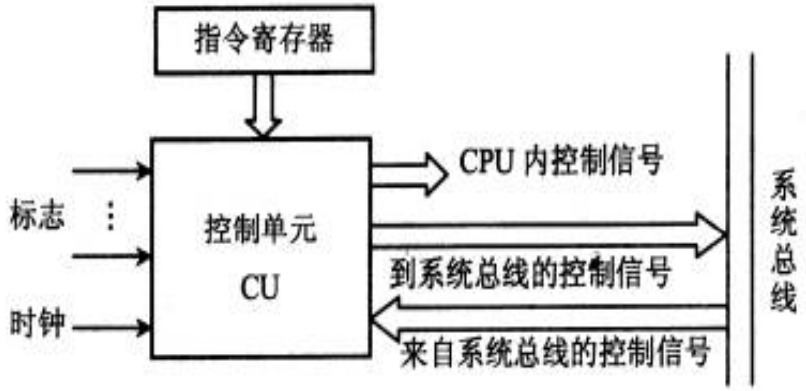


执行周期——转移指令

- **JMP X:** 无条件转移至**X**
  - **Ad(IR)->PC**
- **BAN X:** 如果前一条指令的执行结果为负（即**ACC**最高位为“**1**”）则转移至**X**
  - **ACC_0*Ad(IR)+/ACC_0* (PC)->PC**

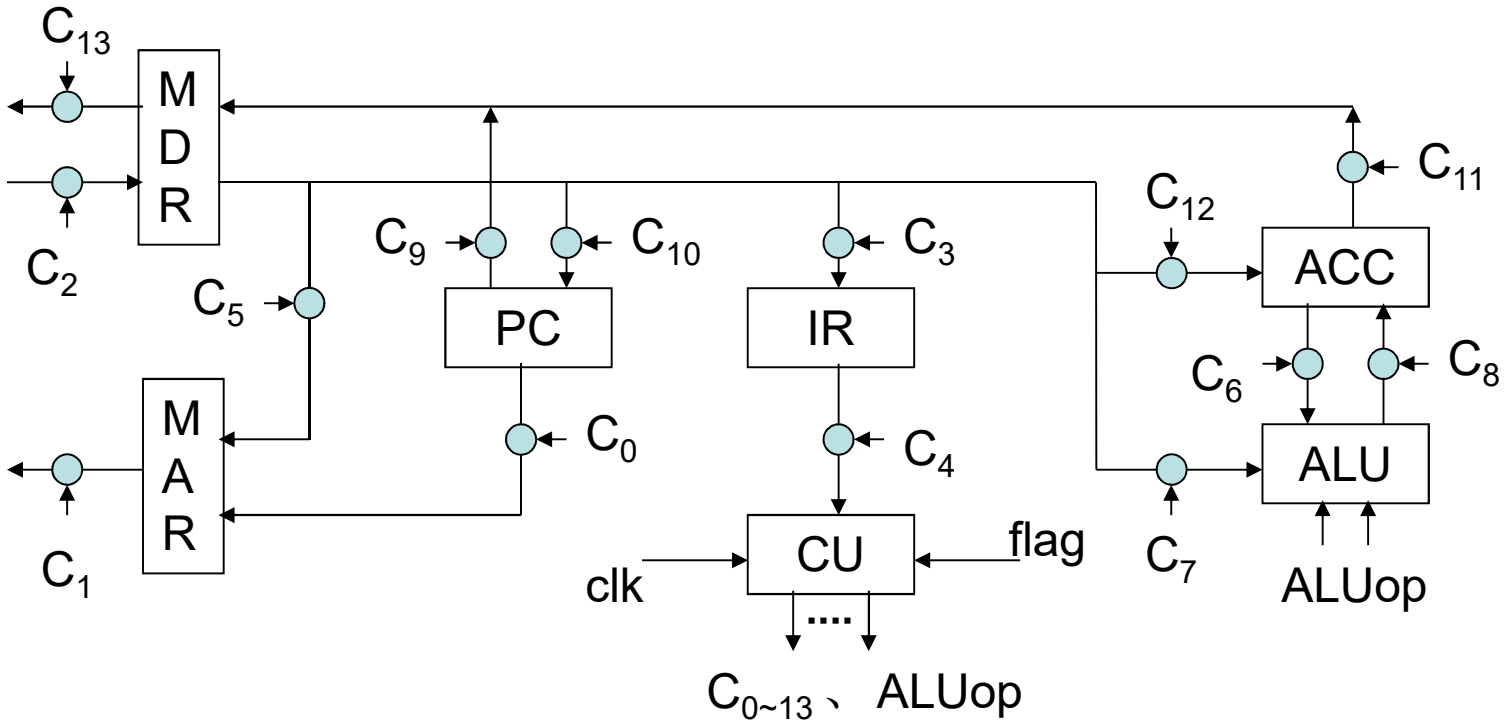
控制单元（CU）的功能（$9.2）

- 在时钟控制下，产生完成各个微操作所需的控制信号
- CU的外特性
  - 输入
    - 时钟节拍
    - 指令寄存器（IR）的op域
    - 标志：指示CPU的当前状态
    - 中断、DMA等外部控制信号
  - 输出
    - 微操作控制信号
    - 总线控制信号（访存、I/O控制等）



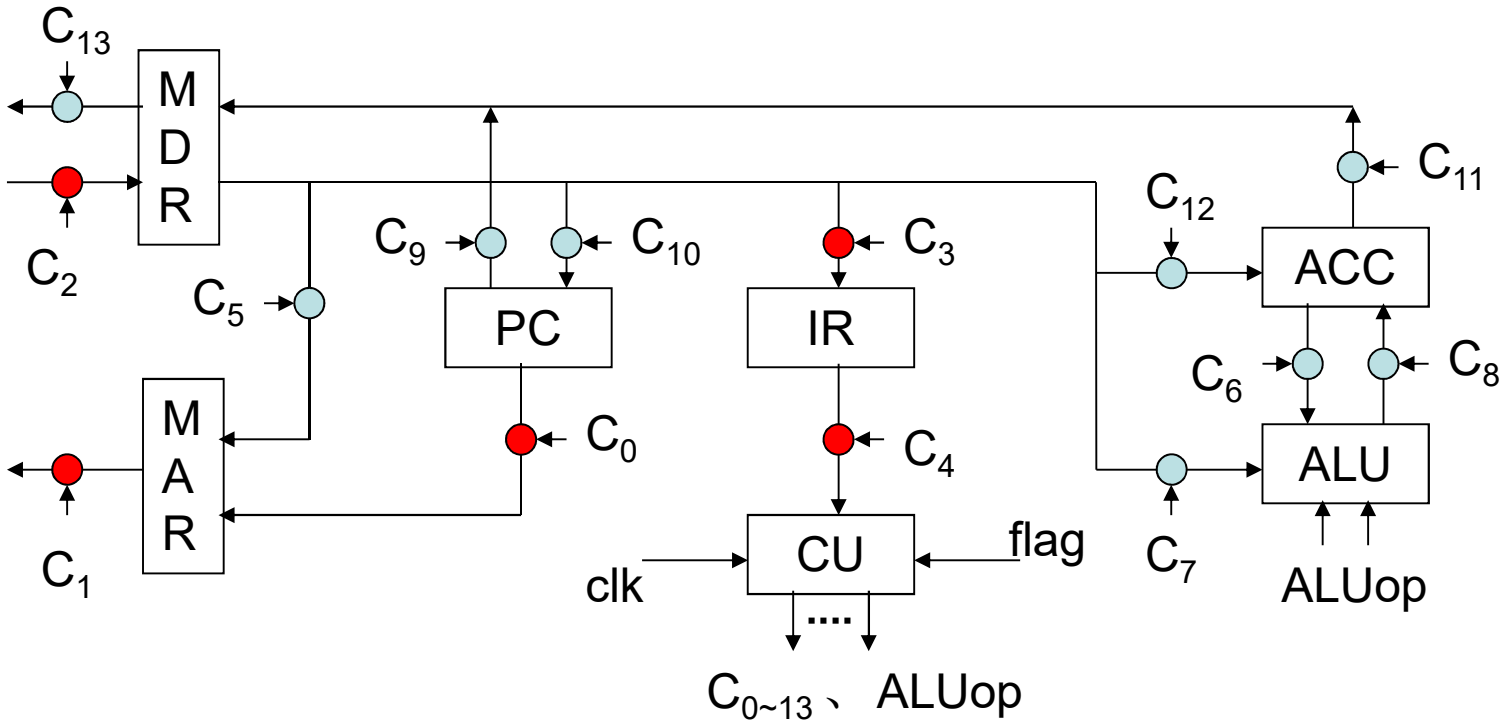
控制单元的外特性

ADD @X的控制—分散连接方式



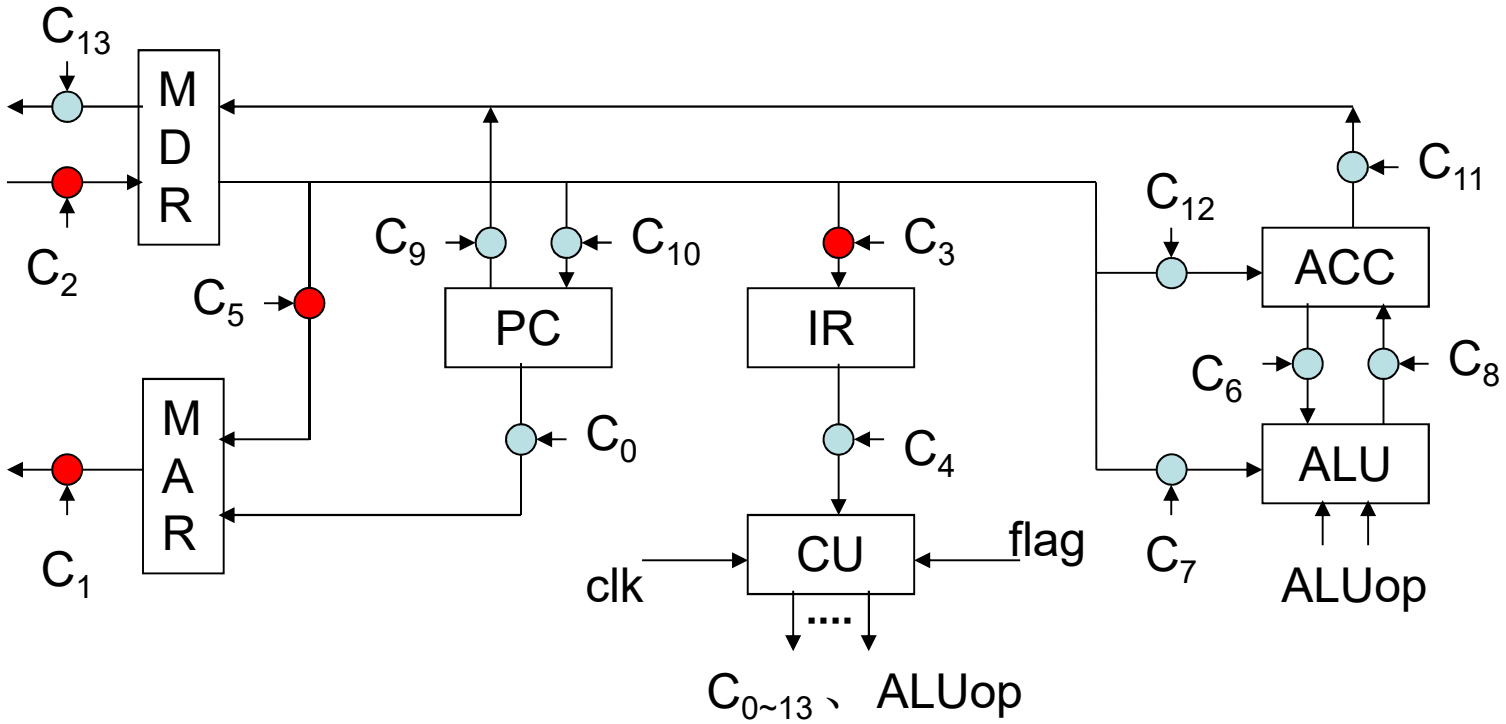
- 因为**IR=MDR,** 故省略了**IR->MAR**的路径

ADD @X的控制—取指



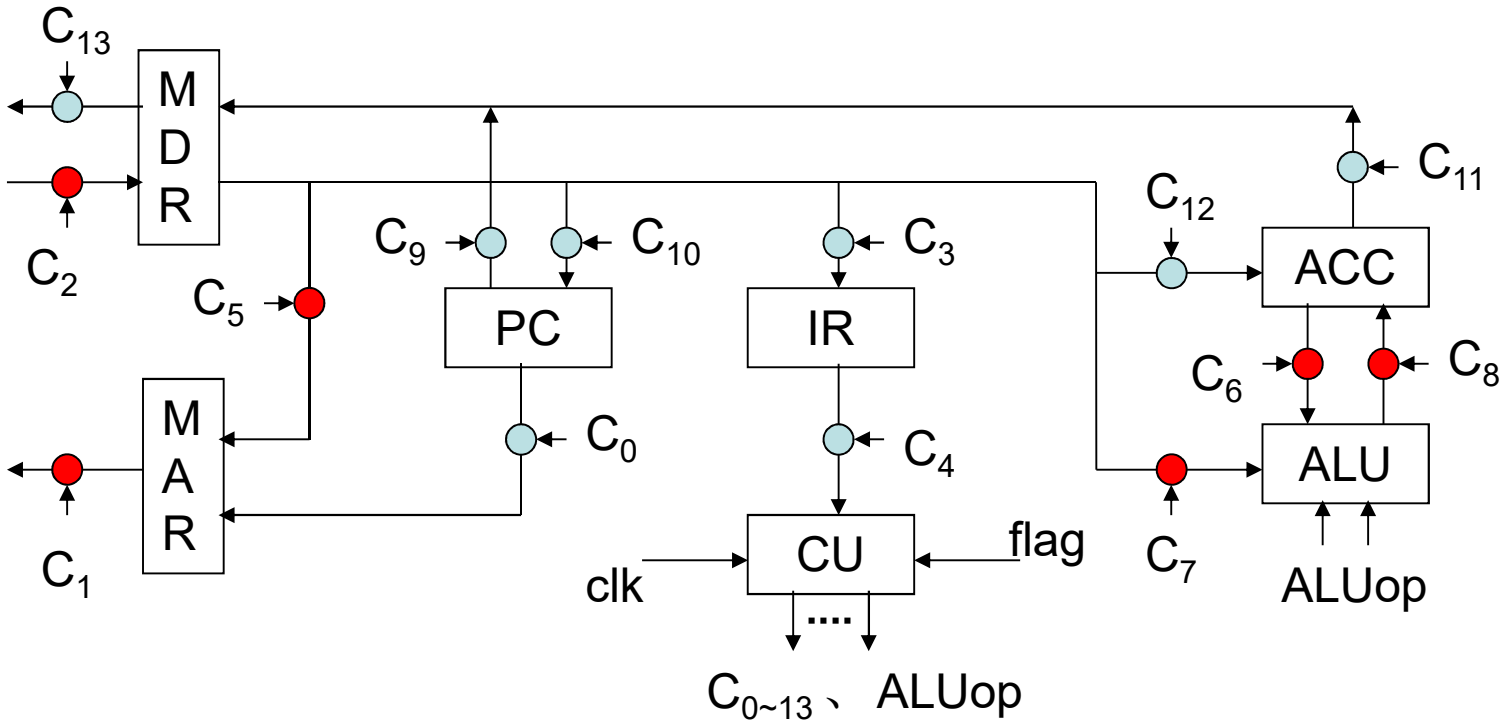
- C_0、C_1、C_2、C_3、C_4

ADD @X的控制—间址



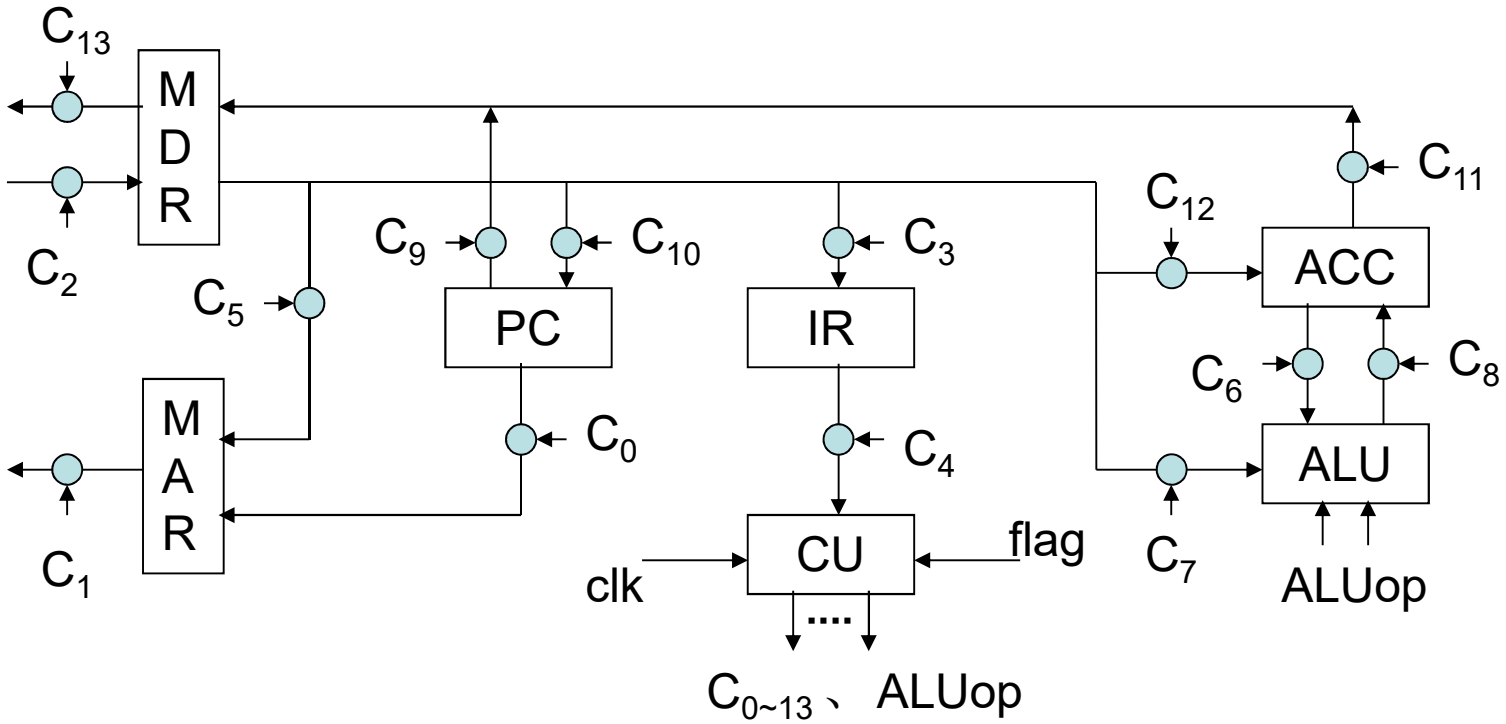
- **C_5、C_1、C_2、C_3，得到EA**

ADD @X的控制—执行



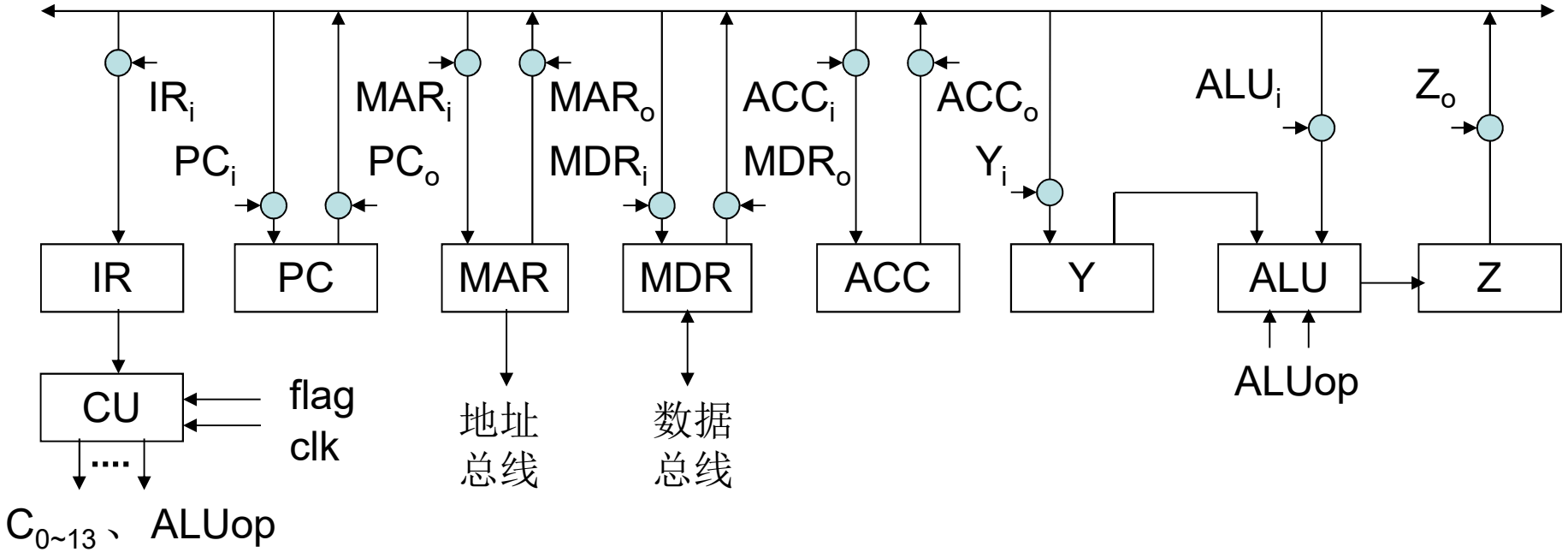
- 取数：C_5、C_1、C_2
- 计算：C_6、C_7
- 写回：C_8

A模型的系统时序？



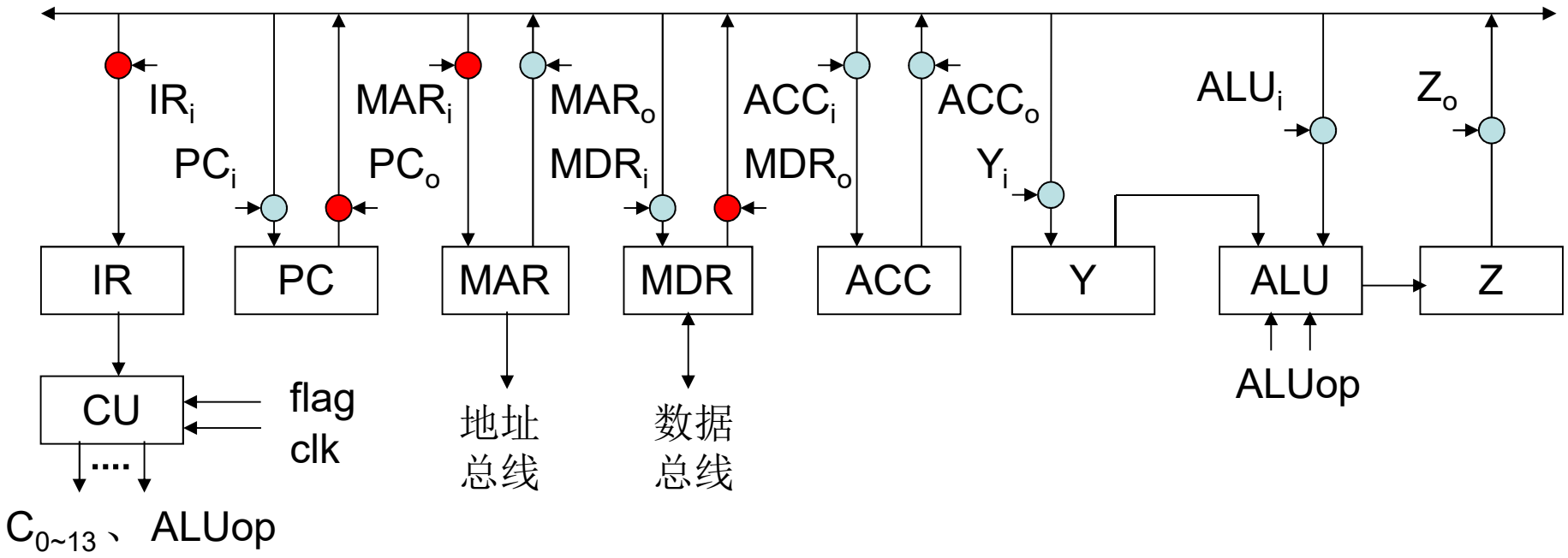
- 取指、间指、执行分别用到的功能部件？

ADD @X的控制—总线连接方式



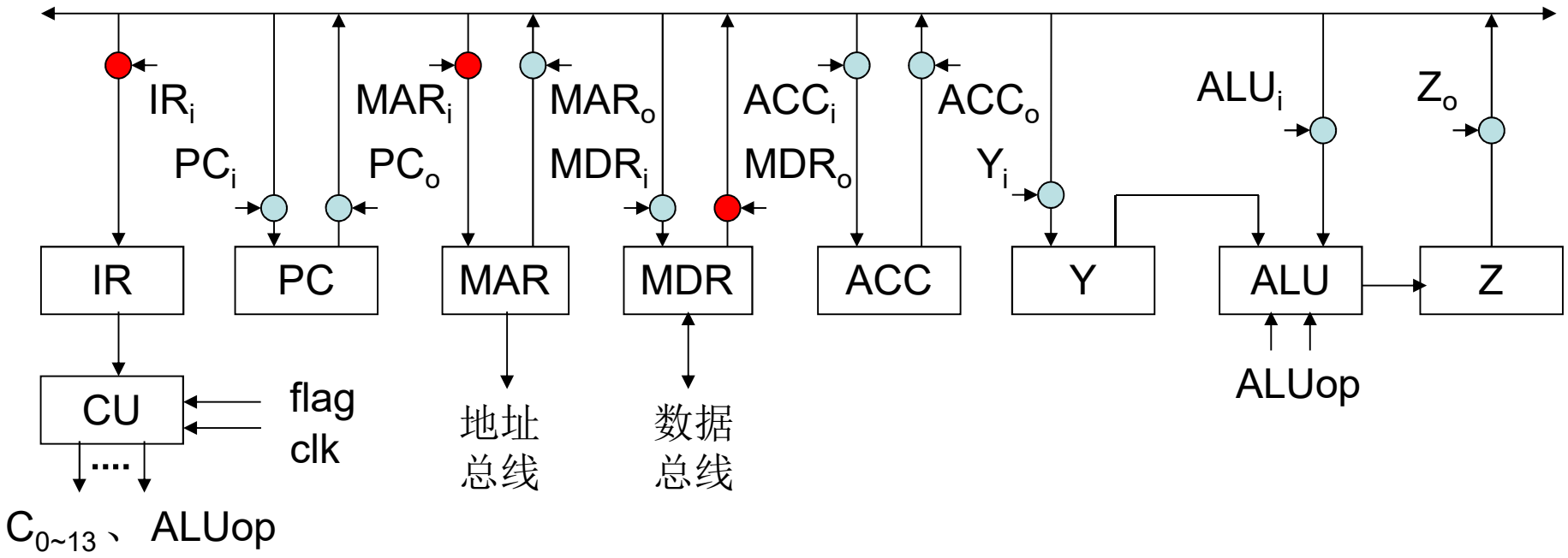
- 寄存器**Y**、**Z**的作用：暂存**ALU**的操作数

ADD @X的控制—取指



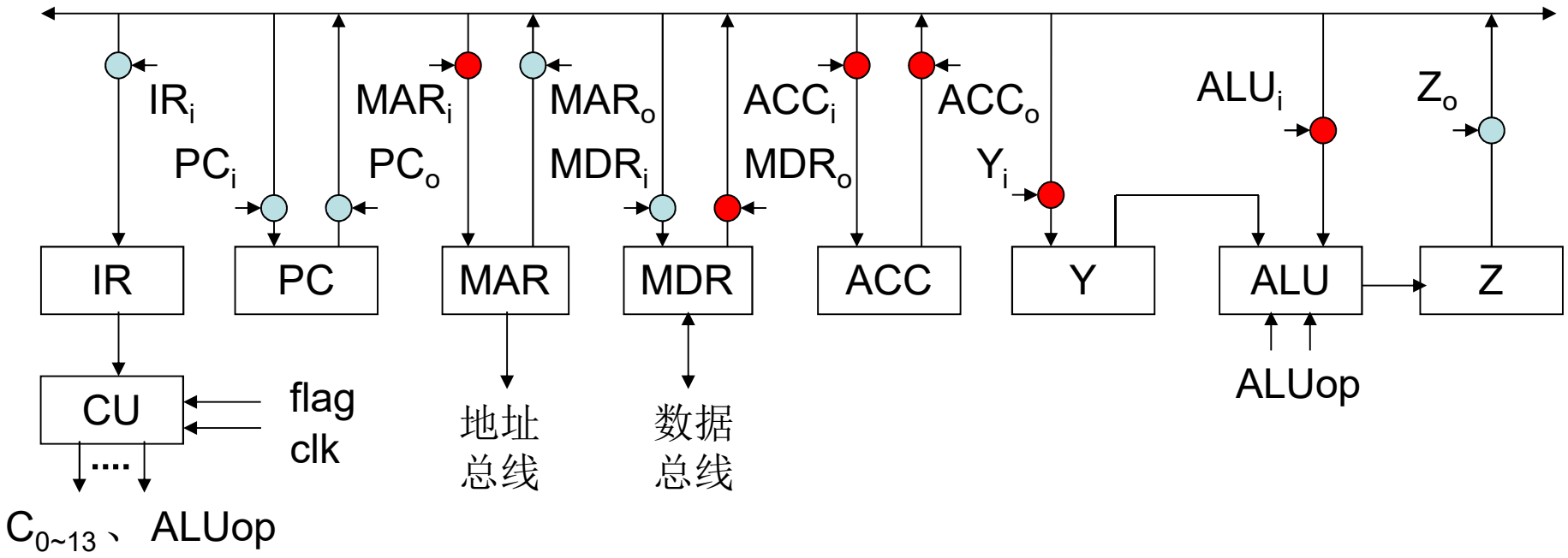
- **PC_o、MAR_i、MDR_o、IR_i**

ADD @X的控制—间址



- **MDR_o、 MAR_i 、 MDR_o、 IR_i**

ADD @X的控制—执行



- 取数：**MDR$_o$、MAR$_i$ 、MDR$_o$、Y$_i$**
- 计算：**ACC$_o$、ALU$_i$**
- 写回：**Z$_o$、ACC$_i$**

小结

- **CPU**的结构与功能
- 指令的执行过程
  - 多个机器周期
  - 多个微操作
- **CU**的功能：产生各种控制信号（微指令）
- **CU**的定时：按照微操作的顺序在不同的节拍发出不同的控制信号
- 要好好研究各个指令的数据通路！！！
  - 如果**A**模型增加“立即寻址”？
- 时钟周期宽度如何确定？
- 作业：**8.4(1)**、**9.1**、**9.3**、**9.6**