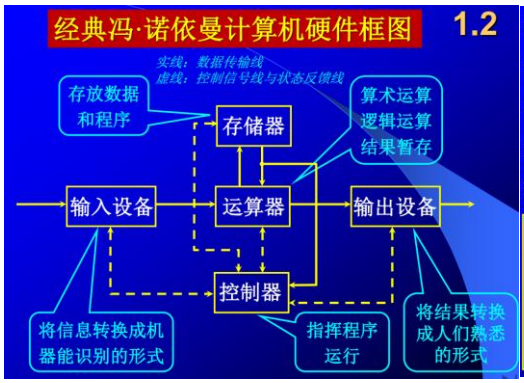
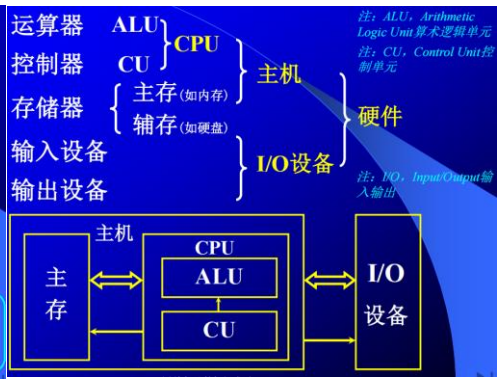
第一章 计算机系统概论

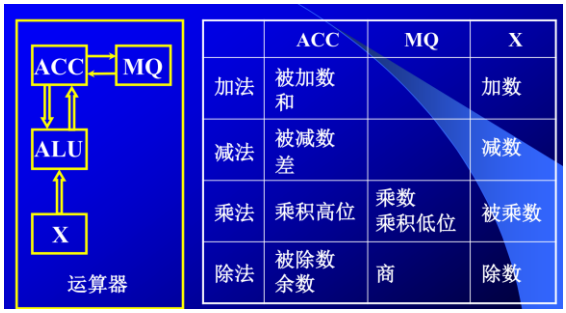
冯诺依曼体系结构



现代计算机硬件框图

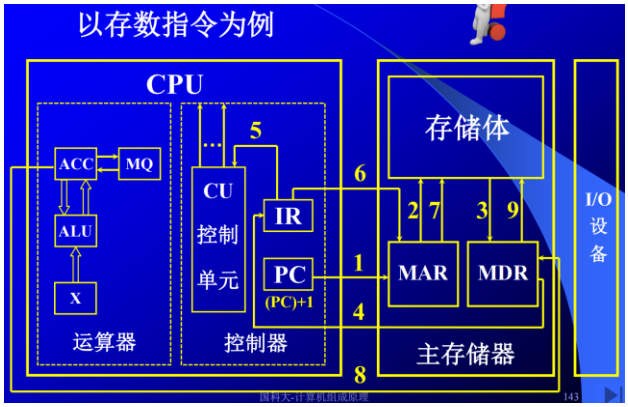


运算器的基本组成



	ACC	MQ	X
加法	被加数和		加数
减法	被减数差		减数
乘法	乘积高位	乘数 乘积低位	被乘数
除法	被除数余数	商	除数

*主机完成一条指令的过程



PC 此刻存储的地址,不一定是 0

存数

① PC → MAR
② 读
③ [0] → MDR
④ MDR → IR
⑤ IR → CU
⑥ Addr(IR) → MAR
⑦ 读mem (找到 Addr(mem))
⑧ ACC → MDR
⑨ MDR → Addr(mem)
⑩ PC+1

eg 取数指令 Addr

取址
① PC → MAR 当前要执行的指令地址
② 读存储体 (根据MAR)
③ [0] → MDR
④ MDR → IR

分析 ⑤ IR → CU

执行
⑥ Addr(IR) → MAR
⑦ 读 mem
⑧ [Addr] → MDR
⑨ MDR → ACC
⑩ PC+1 按字节编址下一条指令的长度
如: 32bit PC <= PC + 4

*性能评估指标

缩写/全称	解释	计算方式	影响因素
IC **Instruction Count**	一个程序的指令条数		程序、ISA 和编译器
CPI **Cycles Per Instruction**	**一个程序中：每个周期执行多少条指令** **其倒数为：每条指令要执行多少个周期**	CPI= $\frac{\text{Clock Cycles}}{\text{Instruction Count}}$	与 cpu 的硬件设计有关 与 ISA 无关
假定 CPI_i、F_i是各指令CPI和在程序中的出现频率，则程序综合/平均CPI为： $\text{CPI}_{avg} = \sum_{i=1}^{n} CPI_i \times F_i$ where $F_i = \frac{C_i}{Instruction\_Count}$			
CPU Time	**一个程序的总运行时长**	CPU Time=IC×CPI×Clock Cycle Time(即1/F)	
MIPS **Million Instructions Per Second**	**每秒钟运行几百万条指令**	MIPS=$\frac{IC}{CPU\ Time} \div 10^6$ 指令条数/总时长 =$\frac{F}{CPI} \div 10^6$ 每条指令所需周期×每秒有多少周期	

单靠 CPI 不能反映 CPU 性能！

假设有一种 CPU，执行每种指令都需要 1 个时钟周期，即单周期 CPU

那么单周期 CPU 的 CPI = 1，但单周期 CPU 的性能不够好！

第三章 系统总线

总线分类：片内总线、系统总线(各部件之间)、通信总线(系统之间)

总线性能指标：总线宽度、总线带宽、时钟同步/异步、总线复用、信号线数、总线控制方式、负载能力等。

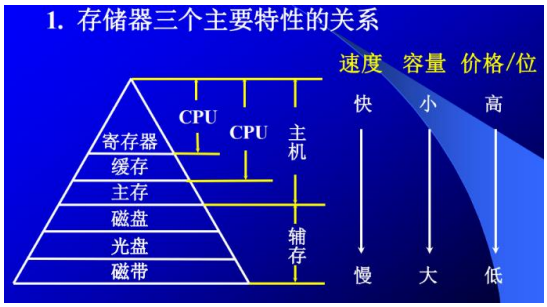
总线通信的四种方式：同步通信、异步通信(不/半/全互锁)、半同步通信、分离式通信

其中前三种通信方式只有主模块有权占用总线

第四章 存储器

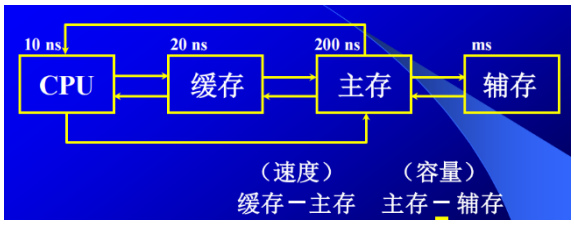
存取周期：存储器进行连续两次独立的存储器操作所需的最小间隔时间

存储器的层次结构图+三个主要特性的关系



三个特性：速度、容量、价格

缓存-主存层次 和 主存-辅存层次



层次结构的体现：存储系统层次结构主要体现在缓存 - 主存和主存 - 辅存这两个存储层次上。CPU 和缓存、主存都能直接交换信息；缓存能直接和 CPU 、主存交换信息；**主存**可以和 CPU 、缓存、**辅存**交换信息。

缓存 - 主存层次主要解决 **CPU 和主存速度不匹配**的问题。主存和缓存之间的数据调动是由硬件自动完成的，对程序员是**透明的。**（不可见的）

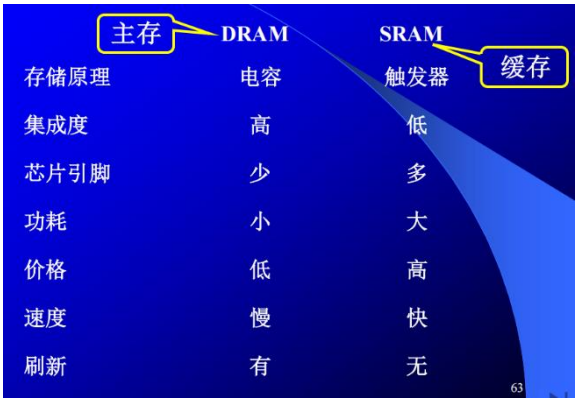
主存 - 辅存层次主要解决存储系统的**容量问题。**主存和辅存之间的数据调动是由**硬件和操作系统共同完成**的。

注意按字节寻址和按字寻址的区别！！

半导体存储芯片的**译码驱动方式**：线选法(每次选中一条字线)、重合法(选中 bit)

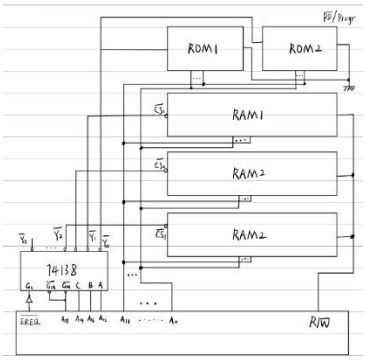
随机存取存储器 RAM，都是**易失性存储器**：静态 SRAM(SR 锁存器)、动态 DRAM(电容)

动态 RAM 刷新：集中刷新、分散刷新、分散刷新与集中刷新相结合(每隔 15.6μs 刷新一行，每次刷新一行占用一个存取周期，对每行而言都是间隔 2ms 内刷新一次。)



DRAM 与 SRAM 对比

存储器容量的扩展



4.15 设 CPU 共有 16 根地址线，8 根数据线，并用 $\overline{MREQ}$（低电平有效）作访存控制信号，R/$\overline{W}$ 作读/写命令信号（高电平为读，低电平为写）。现有这些存储芯片：ROM（2 K×8 位，4 K×4 位，8 K×8 位），RAM（1 K×4 位，2 K×8 位，4 K×8 位）及 74138 译码器和其他门电路（门电路自定）。

试从上述规格中选用合适的芯片，画出 CPU 和存储芯片的连接图。要求如下：

（1）最小 4 K 地址为系统程序区，4096～16383 地址范围为用户程序区。

（2）指出选用的存储芯片类型及数量。

（3）详细画出片选逻辑。

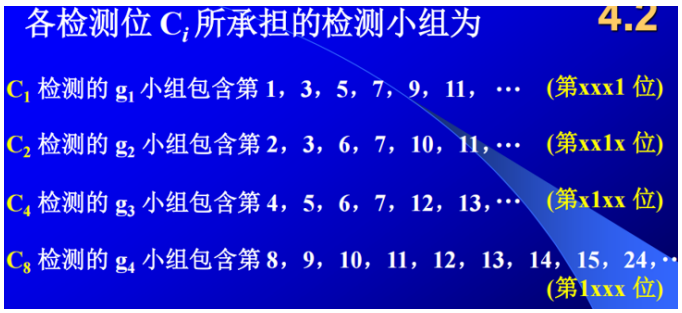
16,384 = 2^{14}，故需要的总存储容量为 16K×8 位。

系统程序区使用 2 片 4K×4 位的 ROM，位扩展；用户程序区使用 3 片 4K×8 位的 RAM。CPU 的 16 根地址线其中 12 根用于确定片内访存地址，3 根用于生成片选信号（实际有用的只有 2 根），剩余 1 根冗余地址线连接译码器使能信号。

存储器的校验

设欲检测的二进制代码为 n 位，为使其具有纠错能力，需增添 k 位检测位：**$2^k \geq n+k+1$**

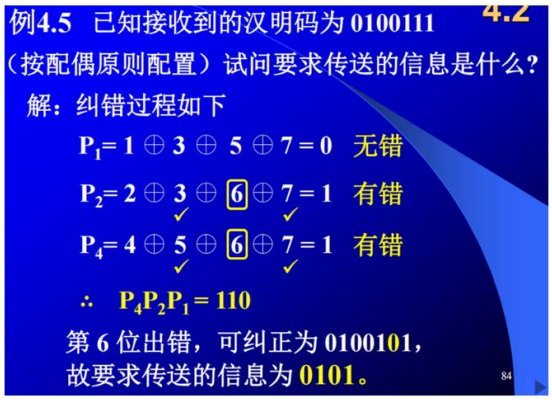
检测位的位置为：**2^i (i=0, 1, 2, 3, …)，即 1，2，4，8……**



偶校验配置： gi 小组(含 Ci)共有偶数个 1，即 Ci=gi 所有元素的异或

奇校验配置： gi 小组(含 Ci)共有奇数个 1，即 Ci=gi 所有元素的异或⊕1

纠错过程：



P2、P4 同时有 6、7，但 P1 也有

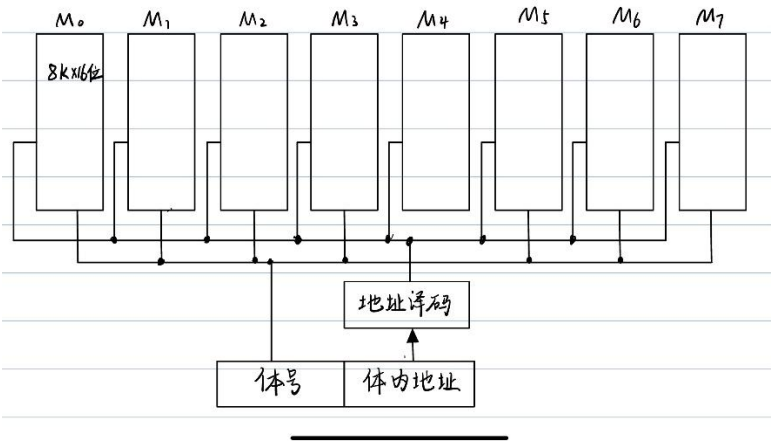
7，故只能是 6 出错

提高访存速度的措施：高性能存储芯片、采用层次结构、调整主存结构

单体多字系统、多体并行系统（低位交叉编址才能提高带宽）、

某机字长为 16 位，常规的存储空间为 64 K 字，若想不改用其他高速的存储芯片，而使访存速度提高到 8 倍，可采取什么措施？画图说明。

可采取多体并行系统，采用低位交叉编址，每个存储器容量为 8K×16 位，地址的最低 3 位为存储器体号：



Cache

现代计算机的 Cache 也制作在 CPU 中，Cache 对用户是透明的

命中率 **h= Nc / (Nc+Nm)**

平均访问时间 **$t_a = h \times t_c + (1 - h) \times t_m$ （t_c 为 Cache 存取周期，t_m 为主存存取周期）**

效率 **$e = t_c / t_a$ （Cache 存取周期/平均访问时间）**

Cache –主存的地址映射：直接映射、全相联映射、组相联映射

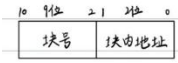
程序访问的局部性原理

时间局部性：刚被访问过的单元很可能不久又被访问

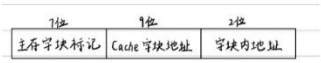
空间局部性：刚被访问过的单元的邻近单元很可能被访问

主存容量为 256K = 2^{18} 字，按字寻址需要 18 位地址，按字节寻址需要 20 位地址

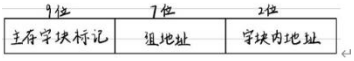
(1) 块长为 4 个字，容量为 2K 字，则共有 2^9=512 块。Cache 地址格式为：



(2) 直接映射方式下，主存地址中直接包含 Cache 地址



(3) 四路组相联，4 个地址视为一组，Cache 字块地址/组地址少 2 位



(4) 全映射方式，没有 Cache 字块区分，无字块地址

(5) 字长为 32bit，则一个 Cache 块内有 4×4=16B，块内按字节寻址需要 4 位地址

辅助存储器

4.38 磁盘组有 6 片磁盘，最外两侧盘面可以记录，存储区域内径 22 cm，外径 33 cm，道密度为 40 道/cm，内层密度为 400 位/cm，转速 3 600 r/min。

(1) 共有多少存储面可用？

(2) 共有多少柱面？

(3) 盘组总存储容量是多少？

(4) 数据传输率是多少？

(1) 6 片磁盘，每片最外两侧盘面可以记录，共有 6×2=12 存储面可用

(2) 有效存储半径=(33-22)÷2=5.5cm，柱面数=40 道/cm×5.5cm = 220 道

(3) 内层道周长=22cm×π=69.08cm，道容量=400 位/cm×69.08cm=27632b=3454 B

面容量=3454B×220=759880 B，盘组总容量=759880 B×12=9118560 B

(4) 数据传输率 = 3454 B×60 转/s=207240 B/s

4.39 某磁盘存储器转速为 3 000 r/min，共有 4 个记录盘面，每毫米 5 道，每道记录信息 12 288 字节，最小磁道直径为 230 mm，共有 275 道，求：

(1) 磁盘存储器的存储容量。

(2) 最高位密度(最小磁道的位密度)和最低位密度。

(3) 磁盘数据传输率。

(4) 平均等待时间。

(1) 存储容量=275 道×12288 B/道×4 面 = 13516800B

(2) 最高位密度=12288 B÷(230mm×π) ≈ 17B/mm

最大磁道直径=230mm+275/5 mm = 340mm

最低位密度=12288 B÷(340mm×π) = 11B/mm

(3) 数据传输率=12288 B×50 转/s = 614400B/s

(4) 平均等待时间=(60s / 3000)÷2 = 0.01s

第五章 输入输出系统

I/O 设备与主机信息传送的控制方式：程序查询方式(串行)、程序中断方式(并行)、DMA 方式

DMA 与主存交换数据三种方式：停止 CPU 访问主存、周期挪用、DMA 与 CPU 交替访问

DMA 周期挪用的三种可能：

- CPU 此时不访存
- CPU 正在访存：结束后将总线占有权让出

- CPU 与 DMA 同时请求访存：I/O 优先访存

例 5.3 一个 DMA 接口可采用周期窃取方式把字符传送到存储器，它支持的最大批量为 400 个字节。若存取周期为 100 ns，每处理一次中断需 5 μs，现有的字符设备的传输率为 9 600 bps。假设字符之间的传输是无间隙的，若忽略预处理所需的时间，试问采用 DMA 方式每秒因数据传输需占用处理器多少时间？如果完全采用中断方式，又需占用处理器多少时间？

解：根据字符设备的传输率为 9 600 bps，则每秒能传输

$$9\ 600/8 = 1\ 200\ \text{B}(1\ 200\ \text{个字符})$$

若采用 DMA 方式，传送 1 200 个字符共需 1 200 个存取周期，考虑到每传 400 个字符需中断处理一次，因此 DMA 方式每秒因数据传输占用处理器的时间是

$$0.1\ \mu\text{s} \times 1\ 200 + 5\ \mu\text{s} \times (1\ 200 / 400) = 135\ \mu\text{s}$$

若采用中断方式，每传送一个字符要申请一次中断请求，每秒因数据传输占用处理器的时间是

$$5\ \mu\text{s} \times 1\ 200 = 6\ 000\ \mu\text{s}$$

例 5.4 假设磁盘采用 DMA 方式与主机交换信息，其传输速率为 2 MBps，而且 DMA 的预处理需 1 000 个时钟周期，DMA 完成传送后处理中断需 500 个时钟周期。如果平均传输的数据长度为 4 KB，试问在硬盘工作时，50 MHz 的处理器需用多少时间比率进行 DMA 辅助操作（预处理和后处理）？

解：DMA 传送过程包括预处理、数据传送和后处理 3 个阶段。传送 4 KB 的数据长度需

$$(4\ \text{KB})/(2\ \text{MBps}) = 0.002\ \text{s}$$

如果磁盘不断进行传输，每秒所需 DMA 辅助操作的时钟周期数为

$$(1\ 000 + 500)/0.002 = 750\ 000$$

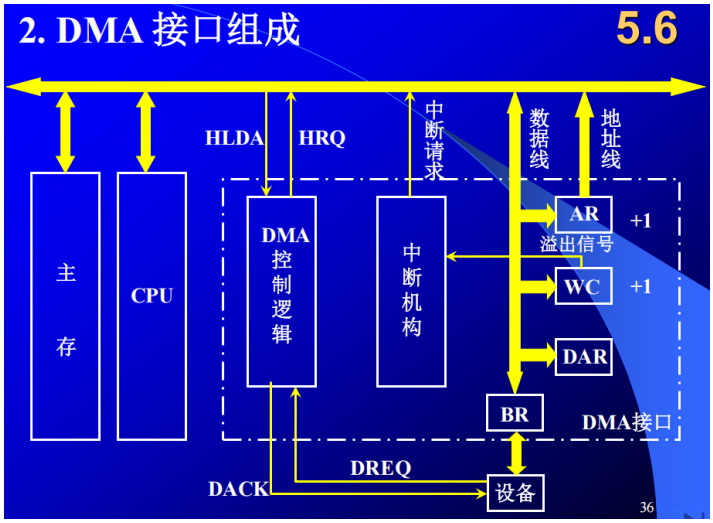
故 DMA 辅助操作占用 CPU 的时间比率为

$$[750\ 000 / (50 \times 10^6)] \times 100\% = 1.5\%$$

DMA 传送过程：

预处理、数据传送、

后处理



主存地址寄存器 AR：存放主存中需要交换数据的地址，每次传送数据前必须设置好。传送过程中，每交换一次数据，将 AR 内容 + 1，直到一批数据传送完成为止。↵

字计数器 WC：预设为待交换字数的补码值，每交换一次自加 1，直到为 0 即产生最高位进位时，表示该批数据传送完毕。↵

数据缓冲寄存器 BR；设备地址寄存器 DAR↵

DMA 控制逻辑：DREQ（设备每准备好一个数据字/一个字传送完毕时，向 DMA 接口提出申请）、HRQ（收到申请的 DMA 接口向 CPU 发出总线使用权的请求信号）、HLDA（CPU 响应信号）↵

中断机构：收到 WC 的溢出信号后，向 CPU 发出中断请求，让 CPU 做 DMA 操作的后处理，目的是报告一批数据传送结束。↵

设备地址寄存器 DAR：存放 I/O 设备的设备码↵

DARK：通知设备已被授予一个 DMA 周期↵

DMA 方式与程序中断方式的比较

	中断方式	DMA 方式
(1) 数据传送	程序	硬件
(2) 响应时间	指令执行结束	存取周期结束
(3) 处理异常情况	能	不能
(4) 中断请求	传送数据	后处理
(5) 优先级	低	高

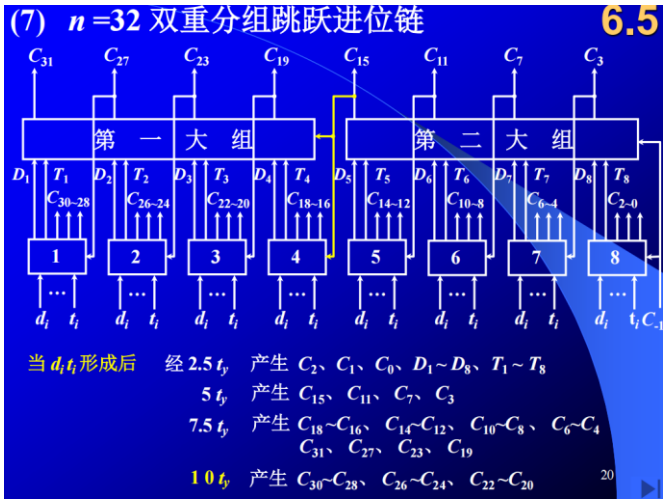
第六章 数的编码与运算

IEEE 754

	数符 S	阶码	尾数	总位数
单精度	1	8	23	32
双精度	1	11	52	64
临时实数	1	15	64	80

阶码：移码表示，偏移值为 $2^7-1=127$ / $2^{k-1}-1$ ，k 为阶码位数

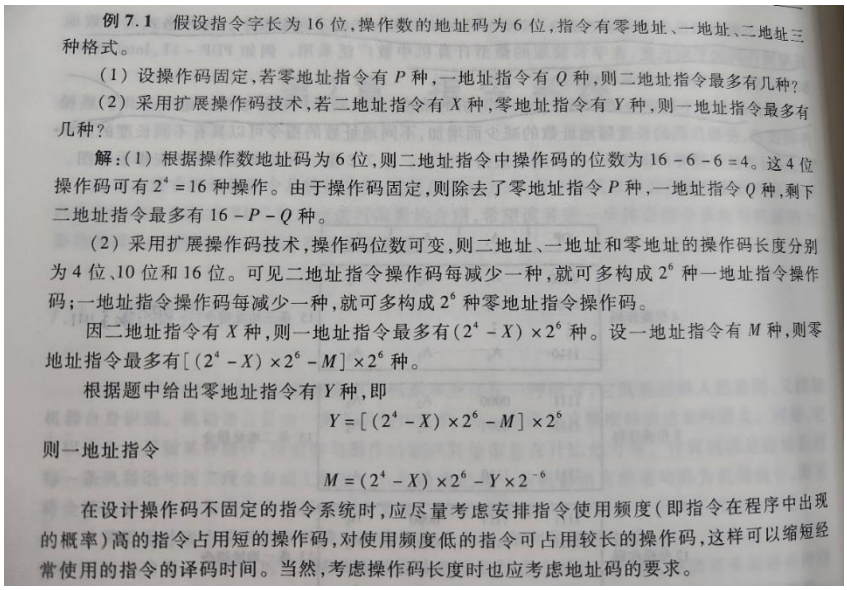
尾数（绝对值）：用原码规格化形式表示，省略整数位的 1



浮点加减：①对阶(小阶向大阶对齐)②求和③规格化④舍入⑤溢出判断

浮点乘除：①阶码加减②尾数相乘/除③规格化④舍入⑤溢出判断

第七章 指令系统



例 7.1 假设指令字长为 16 位，操作数的地址码为 6 位，指令有零地址、一地址、二地址三种格式。

（1）设操作码固定，若零地址指令有 P 种，一地址指令有 Q 种，则二地址指令最多有几种？

（2）采用扩展操作码技术，若二地址指令有 X 种，零地址指令有 Y 种，则一地址指令最多有几种？

解：（1）根据操作数地址码为 6 位，则二地址指令中操作码的位数为 16 - 6 - 6 = 4。这 4 位操作码可有 $2^4 = 16$ 种操作。由于操作码固定，则除去了零地址指令 P 种，一地址指令 Q 种，剩下二地址指令最多有 $16 - P - Q$ 种。

（2）采用扩展操作码技术，操作码位数可变，则二地址、一地址和零地址的操作码长度分别为 4 位、10 位和 16 位。可见二地址指令操作码每减少一种，就可多构成 2^6 种一地址指令操作码；一地址指令操作码每减少一种，就可多构成 2^6 种零地址指令操作码。

因二地址指令有 X 种，则一地址指令最多有 $(2^4 - X) \times 2^6$ 种。设一地址指令有 M 种，则零地址指令最多有 $[(2^4 - X) \times 2^6 - M] \times 2^6$ 种。

根据题中给出零地址指令有 Y 种，即

$$Y = [(2^4 - X) \times 2^6 - M] \times 2^6$$

则一地址指令

$$M = (2^4 - X) \times 2^6 - Y \times 2^{-6}$$

在设计操作码不固定的指令系统时，应尽量考虑安排指令使用频度（即指令在程序中出现的概率）高的指令占用短的操作码，对使用频度低的指令可占用较长的操作码，这样可以缩短经常使用的指令的译码时间。当然，考虑操作码长度时也应考虑地址码的要求。

变址寻址中，变址寄存器一般在寻址后会自增，以便找到下一个数据。

数据寻址

形式地址：一般地址码字段都是形式地址，记作 A

真实地址：有寻址方式和形式地址共同确定，记作 EA

1. 立即寻址（立即数寻址）：不用寻址，操作数本身就在指令中。寻址特征：#

2. 直接寻址：EA=A，指令中的形式地址即为操作数的真实地址

   缺点：限制范围；必须修改 A 的值才能修改操作数的地址

3. 隐含寻址：（全部或部分）操作数地址隐含在操作码或某个寄存器中

   减少了一个地址字段，有利于缩短指令字长

4. 间接寻址：有效地址由形式地址间接提供 EA=(A)。

有一次间接寻址和多次间接寻址。**多次间接寻址时用存储字的首位来标识间接寻址是否结束。所以间接寻址的寻址范围为 $2^{存储字长-1}$**

执行阶段 2(或 n+1 次)访存。

5. 寄存器寻址：（操作数存在寄存器中）有效地址即为寄存器编号，EA=Ri，**不访存**

6. 寄存器间接寻址：有效地址在寄存器中 EA =（Ri），取得 EA 后仍需要访存

便于编制循环程序（方便回到跳转点）

7. 基址寻址：在程序的执行过程中 基址寄存器内容不变，形式地址 A 可变

（1）采用专用寄存器作基址寄存器：EA =（BR）+ A　BR 为基址寄存器

在程序的执行过程中 BR 内容不变，形式地址 A 可变

（2）采用通用寄存器作基址寄存器：由用户指定哪个通用寄存器作为基址寄存器，但**基址寄存器的内容由操作系统确定**

基址寻址有利于多道程序

8. 变址寻址：EA =（IX）+A，**IX 的内容由用户给定**

在程序的执行过程中 IX 内容可变，形式地址 A 不变

便于处理数组问题：A 设定为数组首地址，通过改变 IX 来访问任一元素

适合编制循环程序

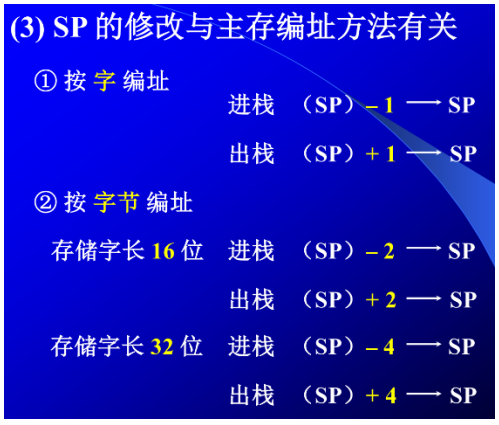
9. 相对寻址：EA =（PC）+ A ，A 是相对于当前指令的位移量（可正可负，补码）

注意（PC）是该寻址指令的下一条指令所在地址，即先更新 PC，再进行寻址

有利于编写浮动程序（数据与指令的相对位置不变）

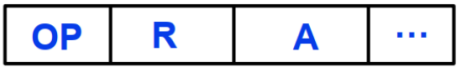
10. 堆栈寻址：操作数存放再堆栈中，由 SP 指向栈顶（存/取地址）

也可以视为一种隐含寻址。写进出栈 SP 的更新时，注意栈底地址是大于还是小于栈顶地址。



注意按字编址和按字节编址的区别！！！

假设：A=地址字段值，R=寄存器编号，
EA=有效地址， (X)=X中的内容



方式	算法	主要优点	主要缺点
立即 #	操作数=A	指令执行速度快	操作数幅值有限
直接	EA=A	有效地址计算简单	地址范围有限
间接 @	EA=(A)	有效地址范围大	多次存储器访问
寄存器	操作数=(R)	指令执行快，指令短	地址范围有限
寄间接	EA=(R)	地址范围大	额外存储器访问
偏移	EA=(R)+A	灵活	复杂
堆栈	EA=栈顶	指令短	应用有限

偏移方式：将直接方式和寄存器间接方式结合起来
有：基址BR / 变址IX / 相对(*)PC 三种
MIPS不区分基址还是变址，统一为偏移寻址方式

汇编语言程序员可见：可以通过指令来**直接或间接**读取、修改或依赖其内容来控制程序流程或处理数据。PC、ACC、MQ、X、通用寄存器

程序员不可见：CPU 内部组织和数据通路中为了实现指令功能而设置的，它们对程序员是透明的。MAR、MDR、IR

一相对寻址的转移指令占 3 个字节，第一字节是操作码，第二、三字节为相对位移量，而且数据在存储器中采用以高字节地址为字地址的存放方式。假设 PC 当前值是 4000H。试问当结果为 0，执行“JZ*+35”和“JZ*-17”指令时，该指令的第二、第三字节的机器代码各为多少?

PC 当前值为 4000H，取出三个字节的转移指令后，PC 值修改为 4003H。

JZ*+35 的相对位移量为 35-3=32 (十进制),该指令的第二字节是 00H,第三字节是 20H。

JZ*-17 的相对位移量为 -17-3=-20 (十进制),该指令的第二字节是 FFH,第三字节是 ECH。

第八章 CPU 的结构与功能

异常（Exception）和中断（Interrupt）

内部“异常”：在（或导致）CPU 内部发生的意外事件或特殊事件

硬故障中断：电源掉电、硬件线路故障等

程序性中断：指向某条指令时发生的“例外”，如算术溢出、缺页、越界、越权、非法指令、除数为 0，堆栈溢出、访问超时、断点设置、单步、系统调用等

外部“中断”：在 CPU 外部发生的特殊事件，通过“中断请求”信号向 CPU 请求处理。

如实时钟、控制台、打印机缺纸、外设准备好、采样计时到、DMA 传输结束等

中断处理(Interrupt handling)：**用于处理异常和中断**

CPU 内部结构：寄存器、ALU、CU、中断系统

1. 用户可见寄存器（CPU 执行机器语言访问的寄存器）

(1) **通用寄存器**：存放操作数；可作某种寻址方式所需的专用寄存器

(2) **数据寄存器**：存放操作数（满足各种数据类型）；两个寄存器拼接存放双倍字长数据

(3) **地址寄存器**：存放地址，位数满足最大地址范围；用于特殊寻址方式、段基值、栈指针

(4) **条件码寄存器** 存放条件码，可作**程序分支的依据**；如正、负、零、溢出、进位等

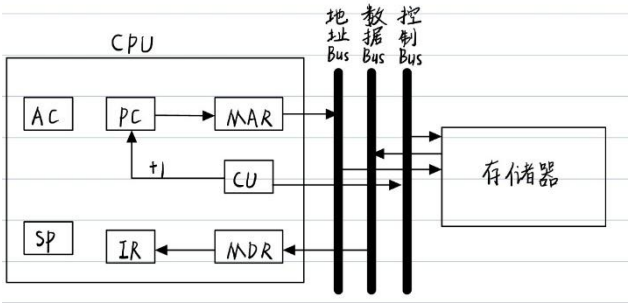
2. 控制和状态寄存器

(1) 控制寄存器： MAR、MDR、IR 都是用户不可见的，**PC 是用户可见的**

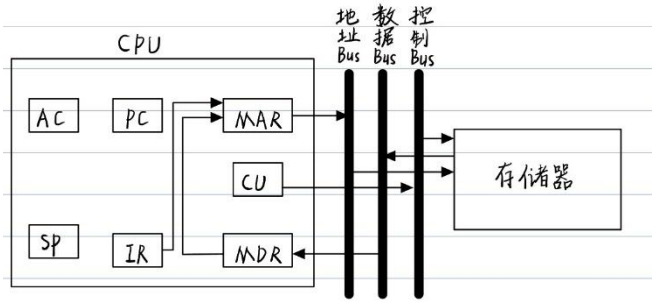
(2) 状态寄存器：状态寄存器存放条件码；PSW 寄存器存放程序状态字

指令周期的数据流（LDA @ X， 间接寻址)

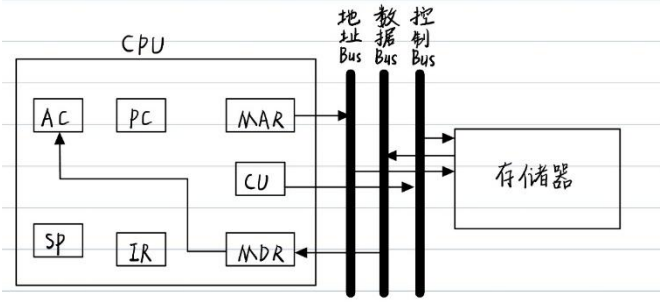
取指



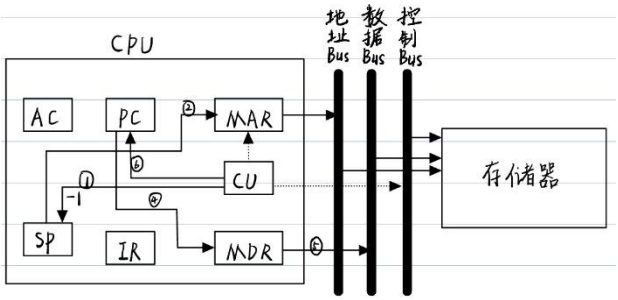
间址

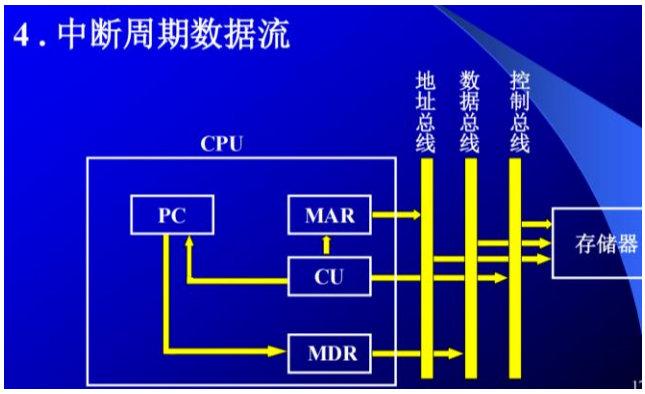


执行



中断





如何提高机器速度

1. 提高**访存速度**(chap4)：**Cache 等分级存储**、**多体并行**、高速芯片

2. 提高 I/O 和主机间的**传送速度**(chap5)：**DMA**、多总线结构、中断、I/O 处理机、通道

3. 提高**运算器速度**(chap6)：改进运算算法、**快速进位链**、高速芯片

4. 提高整机处理能力：提高器件性能、开发系统的并行性(粗粒度：软件实现；细粒度：硬

件，如**流水线**)、流水线多发技术(**超标量**：配置多个功能部件；**超流水线**：一个时钟周期再分段，一个部件在周期内使用多次；**超长指令字**：多条能并行操作的指令合并)

吞吐率：单位时间内 m 级流水线完成指令数量

（无限条指令）最大吞吐率 T_{pmax} = 1/Δt ，Δt 为一个阶段的用时

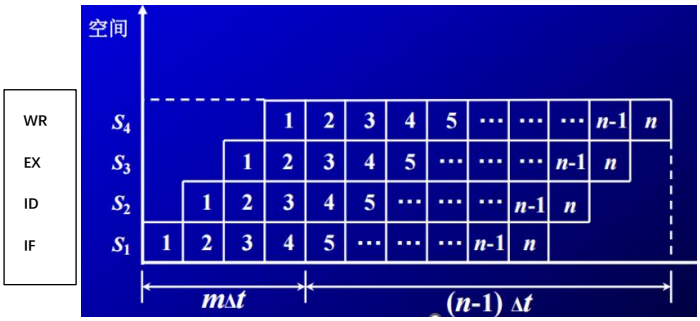
（n 条指令） 实际吞吐率 T_p = 指令条数/总用时

流水线建立时间 = (m-1)Δt，总用时 = (m-1)Δt +nΔt

加速比：Sp = mnΔt/总用时

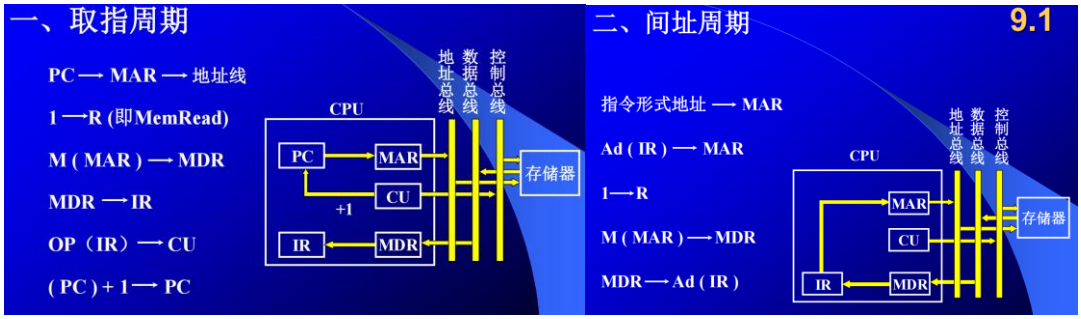
效率：e = 处于工作状态的时空区/总时空区 = mnΔt/(n*总用时)

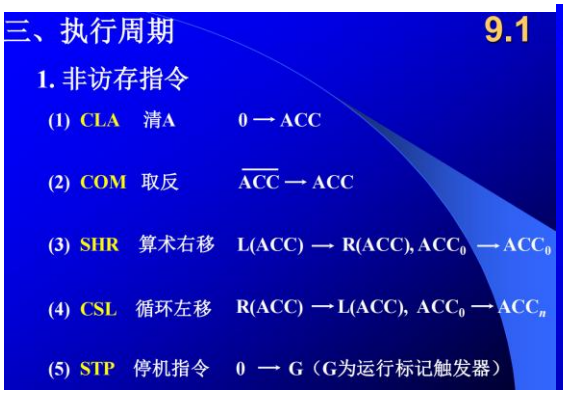
时空图：

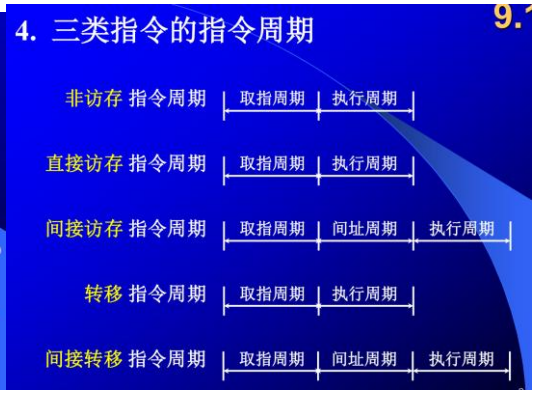


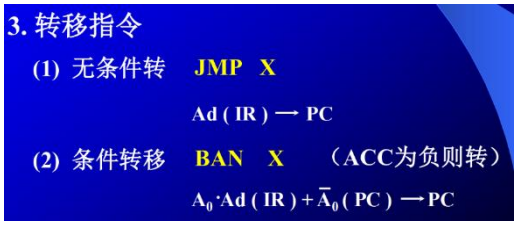
第九章 控制单元的功能

微操作命令









中断周期微操作命令多看几遍！

中断识别程序入口地址 M → PC

注意：栈底为高地址，栈顶为低地址。**最后一步关中断！！**

注：间址周期最后得到的有效地址存放在 MDR！执行周期时第一条命令为 MDR->MAR

指令	JMP @ B	控制信号
取指阶段	PC->MAR->地址线 1->R M(MAR)->数据线->MDR MDR->IR OP(IR)->CU	PCo, MARi MDRo, IRi

	(PC)+1->PC	
间址&执行阶段	(B)->MAR 1->R M(MAR)->数据线->MDR MDR->PC	Bo, MARi MDRo, PCi

寄存器间接寻址需要访存!!!!

指令周期：CPU 取出并执行一条指令所需的全部时间，即完成一条指令的时间。

机器周期：**所有指令执行过程中的一个基准时间，通常以存取周期作为机器周期。**

时钟周期：机器主频的倒数，也可称为节拍，是控制计算机操作的最小单位时间。

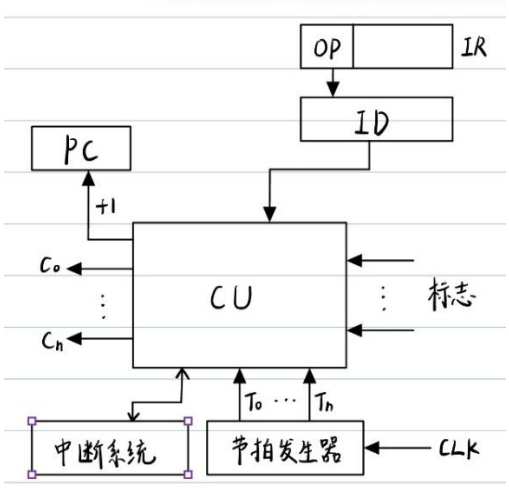
三者关系：一个指令周期包含若干个机器周期，一个机器周期又包括若干个时钟周期，每个指令周期内的机器周期数可以不等，每个机器周期内的时钟周期数也可以不等。

在机器周期所含时钟周期数相同的前提下，两机平均指令执行速度之比等于两机主频之比

机器速度不仅与主频有关 ，还与机器周期中所含时钟周期数 以及指令周期中所含的机器周期数有关

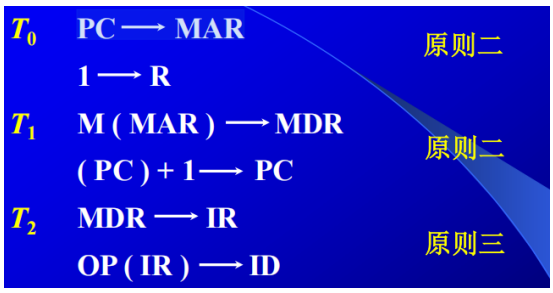
第十章 控制单元的设计

组合逻辑控制单元的组成框图



微操作的节拍安排

取指周期微操作的节拍安排



间址周期微操作的节拍安排

中断周期微操作的节拍安排

10.2 写出完成下列指令的微操作及节拍安排(包括取指操作)。

(1) 指令“ADD R_1, X”完成将 R_1 寄存器的内容和主存 X 单元的内容相加结果存于 R_1 的操作。

(2) 指令“ISZ X”完成将主存 X 单元的内容增 1,并根据其结果若为 0,则跳过下一条指令执行。

(1) 假设使用等长机器周期　　(2) 设 Z 为 ALU 计算结果为 0 标志

取指周期	
T0	PC->MAR, 1->R

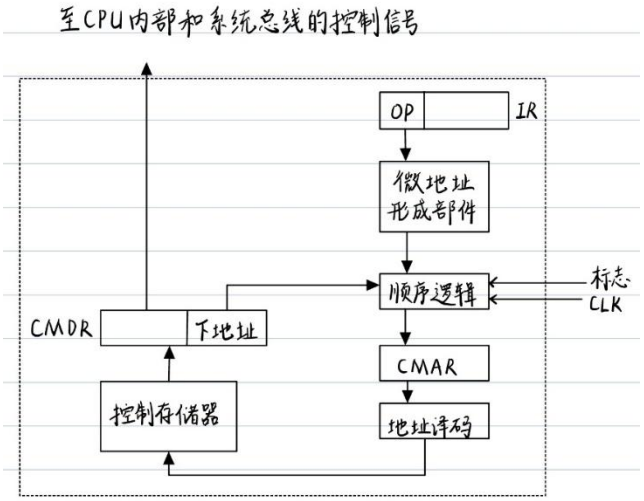
T1	M(MAR)->MDR, (PC)+1->PC
T2	MDR->IR, OP(IR)->ID
执行周期 1	
T0	Ad(IR)->MAR, 1->R
T1	M(MAR)->MDR
T2	(R1)+(MDR)->ACC
执行周期 2	
T0	
T1	
T2	ACC->R1

取指周期	
T0	PC->MAR, 1->R
T1	M(MAR)->MDR, (PC)+1->PC
T2	MDR->IR, OP(IR)->ID
执行周期 1	
T0	Ad(IR)->MAR, 1->R
T1	M(MAR)->MDR
T2	1+(MDR)->ACC
执行周期 2	
T0	ACC->MDR
T1	1->W
T2	MDR->M(MAR), (PC)+1·Z ->PC

采用组合逻辑方法设计控制单元：思路清晰简单，但线路结构复杂且不规范。

因此采用微程序设计方案克服以上缺点：**一条机器指令对应一个微程序 存入 ROM**

微程序控制单元的基本框图



控制存储器 CM

微程序处理指令的流程（M 为取指阶段微程序入口地址）

取指阶段	M->CMAR
	CM(CMAR)->CMDR
	由 CMDR 发命令，并形成下条微指令地址
	Ad(CMDR)->CMAR
	CM(CMAR)->CMDR
	由 CMDR 发命令，并形成下条微指令地址
	Ad(CMDR)->CMAR
	CM(CMAR)->CMDR
	由 CMDR 发命令

执行阶段	OP(IR)->微地址形成部件->CMAR
	CM(CMAR)->CMDR
	由 CMDR 发命令，并形成下条微指令地址
	Ad(CMDR)->CMAR
	CM(CMAR)->CMDR
	由 CMDR 发命令，并形成下条微指令地址
	…
	Ad(CMDR)->CMAR
	CM(CMAR)->CMDR
	由 CMDR 发命令
	回到取指阶段

微指令的编码方式（控制方式）

（1）直接编码方式：**每一位代表一个微操作命令。　译码速度最快，但占用字长较长**

(2) 字段直接编码方式：将微指令的控制字段分成若干“段”，每段经译码后发出控制信号。

（显式编码：一个控制信号由一个字段译码决定）

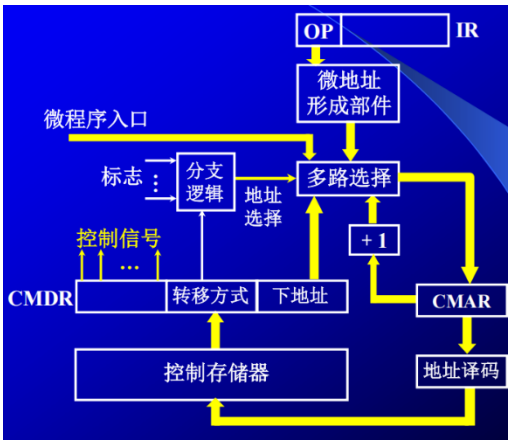
每个字段中的命令是互斥的，注意：状态种数=控制信号数+1(全不激活)

缩短了微指令字长，但增加了译码时间

（3）字段间接编码方式：隐式编码：一个控制信号需要经过多个字段译码决定

（4）混合编码：直接和字段（直接和间接）混合使用　（5）其他：微指令中设常数字段等

后续微指令地址形成方式原理图



五个来源：CMAR+1，下地址，OP 经过微地址形成部件、微程序入口硬件、分支逻辑

第十一章 中断响应

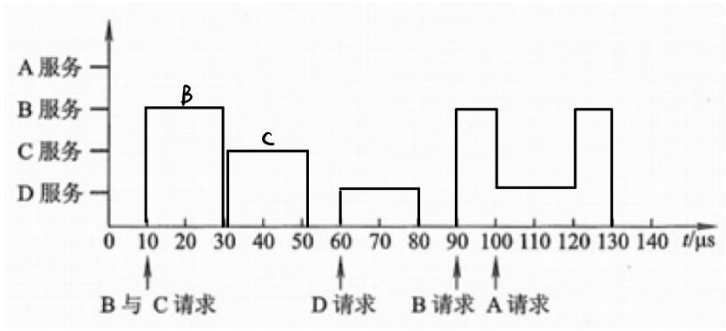
中断请求触发器越多，说明计算机处理中断的能力越强。可以集中在 CPU 的中断系统内，也可以分散在各个中断源的接口电路中

中断隐指令：（1）保护程序断点（2）寻找中断服务程序的入口地址（3）关中断

保护现场：保存断点(中断隐指令完成)、保护寄存器内容、恢复现场(中断服务程序完成)

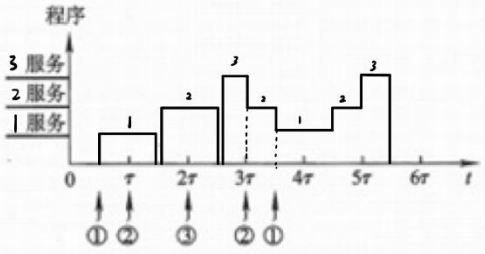
实现多重中断的条件：中断服务程序将 EINT 置 1，重新开中断。

响应优先级与处理优先级下的 CPU 运行轨迹



历史请求均不会被忽略：

8.27 设某机有 3 个中断源，其优先级按 1→2→3 降序排列。假设中断处理时间均为 τ，在下图所示的时间内共发生 5 次中断请求，图中①表示 1 级中断源发出中断请求信号，其余类推，画出 CPU 执行程序的轨迹。



屏蔽字：设 1 表示屏蔽指定请求，设 0 表示可被指定请求占优

多重中断的断点保护：

(1) 断点进栈：中断隐指令完成

(2) 断点存入主存“ 0 ”地址：中断隐指令完成，但对于多重中断，必须在每次中断服务程序的开中断指令前**将 0 地址单元的内容转存到其他地址单元中。**

I/O 中断：CPU 启动 I/O 设备后继续执行现行程序，只有当 I/O 设备准备就绪向 CPU 提出请求后，再暂时中断 CPU 现行程序转入 I/O 服务程序

宏观 上 CPU 和 I/O 并行 工作

微观 上 CPU 中断现行程序 为 I/O 服务

响应优先级：就 **I/O 中断**而言，**速度越高的 I/O 设备，优先级越高**，因为若 CPU 不及时响应高速 I/O 的请求，其信息可能会立即丢失

中断服务程序流程：保护现场、中断服务、恢复现场、中断返回