

## 冯诺依曼型计算机特点

计算机由**运算器**，**控制器**，**存储器**，**输入和输出设备** 5 部分组成

采用存储程序的方式，程序和数据放在同一个存储器中，并以二进制表示。

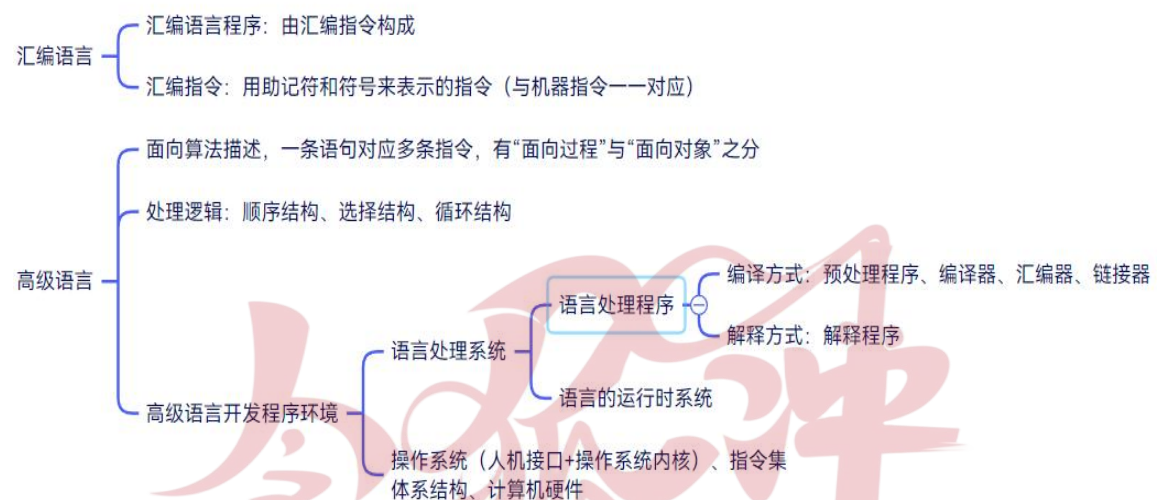
指令由**操作码和地址码**组成

指令在存储器中按执行顺序存放，由指令计数器(即**程序计数器 PC**)指明要执行的指令所在的  
储存单元地址，一般按顺序递增，但可按运算结果或外界条件而改变

机器以**运算器为中心**，输入输出设备与存储器间的数据传送都通过运算器

## 计算机语言

机器语言 —— 计算机唯一可以直接识别和执行的语言，机器指令由二进制表示



[2015 年] 计算机硬件能够直接执行的是 ( )。

- I. 机器语言程序
  - II. 汇编语言程序
  - III. 硬件描述语言程序
- A. 仅 I  
B. 仅 I、II  
C. 仅 I、III  
D. I、II、III

[2016 年] 将高级语言源程序转换为机器级目标代码文件的程序是 ( )。

- A. 汇编程序
- B. 链接程序
- C. 编译程序
- D. 解释程序

## 翻译程序：I

**汇编程序：**汇编语言源程序 → 机器语言目标程序

**编译程序：**高级语言源程序 → 汇编/机器语言目标程序，执行时只要启动目标程序即可

**解释程序：**将高级语言语句逐条翻译成机器指令并立刻执行，不生成目标文件

## 计算机性能指标

**机器字长：**指 CPU 内部用于整数运算的数据通路的宽度（等于 CPU 内部用于整数运算的运算器位数和通用寄存器宽度）

**指令字长：**一个指令字中包含的二进制代码位数

**存储字长：**一个存储单元存储的二进制代码长度

**数据通路带宽：**数据总线一次性所能传送信息的位数

**总线带宽：**总线宽度×总线工作频率

**CPU 时钟周期：**CPU 最小时间单位，每个动作至少需要一个时钟周期，是主频的倒数

**CPI：**执行一条指令所需的时钟周期数

**IPS：**每秒执行指令条数，KIPS、MIPS

**FLOPS：**每秒执行浮点运算次数

[2014 年] 程序 P 在机器 M 上的执行时间是 20s，编译优化后，P 执行的指令数减少到原来的 70%，而 CPI 增加到原来的 1.2 倍，则 P 在 M 上的执行时间是( )。

- A. 8.4 s
- B. 11.7 s
- C. 14 s
- D. 16.8 s

[2022 年] 某计算机主频为 1GHz，程序 P 运行过程中，共执行了 10000 条指令，其中，80% 的指令执行平均需 1 个时钟周期，20% 的指令执行平均需 10 个时钟周期。程序 P 的平均 CPI 和 CPU 执行时间分别是 ( )。

- A. 2.8, 28us
- B. 28, 28us
- C. 2.8, 28 ms
- D. 28, 28 ms

## 数据的表示

**真值：**机器数代表的实际值，现实中带正负号的数

**机器数：**用 0 和 1 编码的计算机内部的 0/1 序列

**原码：**最高位为符号位，数值部分不变

**反码：**原码除符号位外，按位取反

**补码：**反码末位加一，是原码与补码相互转化的过渡

**移码：**数值加一个偏置常数，通常在真值 X 上加  $2^n$ （便于浮点数加减运算时对阶，阶数有正有负，统一加一个常数让阶数都为正方便比较）

**有符号数：**最高位的 0/1 表示正/负

**无符号数：**整个机器字长全部二进制位均为数值位，无符号位

同时出现无符号数和有符号数，在 C 编译器中将隐式转换为无符号数

## n 位二进制数表示范围

类型	编码	表示范围	最大值	最小值	真值零表示
定点整数	原码	$-(2^{n-1}-1) \leq x \leq 2^{n-1}-1$	0, 111...111	1, 111...111	0, 000...0001, 000...000
	反码	$-(2^{n-1}-1) \leq x \leq 2^{n-1}-1$	0, 111...111	1, 000...000	0, 000...0001, 111...111
	补码	$-2^n \leq x \leq 2^{n-1}-1$	0, 111...111	1, 000...000	0, 000...000
定点小数	原码	$-(1-2^{-n}) \leq x \leq 1-2^{-n}$	0, 111...111	1, 111...111	0, 000...0001, 000...000
	反码	$-(1-2^{-n}) \leq x \leq 1-2^{-n}$	0, 111...111	1, 000...000	0, 000...0001, 111...111
	补码	$-1 \leq x \leq 1-2^{-n}$	0, 111...111	1, 000...000	0, 000...000

[2022年] 32 位补码所能表示的整数范围是 ( )。

- A.  $-2^{32} \sim 2^{31}-1$
- B.  $-2^{31} \sim 2^{31}-1$
- C.  $-2^{32} \sim 2^{32}-1$
- D.  $-2^{31} \sim 2^{32}-1$

[2021 年] 已知有符号整数用补码表示，变量  $x, y, z$  的机器数分别为 FFFDH, FFDH, 7FFCH，下列结论中，正确的是 ( )。

- A. 若  $x, y$  和  $z$  为无符号整数，则  $z < x < y$
- B. 若  $x, y$  和  $z$  为无符号整数，则  $x < y < z$
- C. 若  $x, y$  和  $z$  为有符号整数，则  $x < y < z$
- D. 若  $x, y$  和  $z$  为有符号整数，则  $y < x < z$

## 数据的运算——标志位

溢出标志OF (有符号数)  $\left\{ \begin{array}{l} \text{含义: 有符号数的加减法是否发生了溢出} \\ \text{计算: } OF = \text{最高位产生的进位} \oplus \text{次高位产生的进位} \end{array} \right.$

符号标志SF (有符号数)  $\left\{ \begin{array}{l} \text{含义: 有符号数加减运算结果的正负性} \\ SF = \text{最高位的本位和} \end{array} \right.$

零标志ZF  $\left\{ \begin{array}{l} \text{含义: 运算结果是否为0} \\ \text{计算: } n \text{ 位全0, } ZF = 1 \end{array} \right.$

进/借位标志CF (无符号)  $\left\{ \begin{array}{l} \text{含义: 加法时为进位标志, 减法时为借位标志} \\ \text{计算: } CF = \text{最高位进位} \oplus \text{sub (减法sub=1, 加法sub=0)} \end{array} \right.$

[2010 年] 假定有四个整数用 8 位补码分别表示:  $r_1 = \text{FEH}$ ,  $r_2 = \text{F2H}$ ,  $r_3 = \text{90H}$ ,  $r_4 = \text{F8H}$ , 若将运算结果存放在一个 8 位寄存器中, 则下列运算会发生溢出的是 ( )

- A.  $r_1 \times r_2$   $r_1 = -2$
- B.  $r_2 \times r_3$   $r_2 = -4$
- C.  $r_1 \times r_4$   $r_3 = -112$
- D.  $r_2 \times r_4$   $r_4 = -8$

[2018 年] 减法指令 “sub R1, R2, R3” 的功能为 “ $(R1) - (R2) \rightarrow R3$ ”, 该指令执行后将生成进位/借位标志 CF 和溢出标志 OF。若  $(R1) = \text{FFFF FFFFH}$ ,  $(R2) = \text{FFFF FFF0H}$ , 则该减法指令执行后, CF 与 OF 分别为 ( )。

- A. CF = 0, OF = 0
- B. CF = 1, OF = 0
- C. CF = 0, OF = 1
- D. CF = 1, OF = 1

$$[-R2]_{\text{补}} = 00000010\text{H}$$

[2023 年] 已知  $x, y$  为 int 类型, 当  $x = 100$ ,  $y = 200$  时, 执行 “ $x$  减  $y$ ” 指令得到的溢出标志 OF 和借位标志 CF 分别为 0, 1, 那么当  $x = 10$ ,  $y = -20$  时, 执行该指令得到的 OF 和 CF 分别为 ( )。

- A. OF = 0, CF = 0
- B. OF = 0, CF = 1
- C. OF = 1, CF = 0
- D. OF = 1, CF = 1

## 浮点数的表示

浮点数真值格式:  $N = r^E \times M$  ( $r$  阶码的底,  $E$  阶码用移码表示,  $M$  尾数,  $S$  数符)

规格化数形式:  $\pm 1.M \times r^E$  为了表示更多有效数字, 规定规格化数的小数点前为 1

IEEE754 单精度:  $\pm 1.M \times 2^E$  ( $S$  数符,  $E$  阶码用移码表示, 尾数  $M$  用原码表示, 隐含尾数最高数位的 1)

阶码全 0, 尾数  $M = 0$ , 真值  $X = \pm 0$

阶码全 0, 尾数  $M \neq 0$ , 非规格化小数

阶码全 1, 尾数  $M = 0$ , 真值  $X = \pm \infty$

阶码全 1, 尾数  $M \neq 0$ , 非数值 NaN

[2011 年] float 型数据通常用 IEEE754 单精度格式表示。若编译器将 float 型变量  $x$  分配在一个 32 位浮点寄存器 FR1 中, 且  $x = -8.25$ , 则 FR1 的内容是 ( )。

- A. C1040000H
- B. C2420000H
- C. C1840000H
- D. C1C20000H

[2012年] float型（即IEEE754单精度浮点数格式）能表示的最大正整数是（ ）

- A.  $2^{126} - 2^{103}$
- B.  $2^{127} - 2^{104}$
- C.  $2^{127} - 2^{103}$
- D.  $2^{128} - 2^{104}$

[2018年] IEEE 754 单精度浮点格式表示的数中，最小的规格化正数是（ ）。

- A.  $1.0 \times 2^{-126}$
- B.  $1.0 \times 2^{-127}$
- C.  $1.0 \times 2^{-128}$
- D.  $1.0 \times 2^{-149}$

### c 语言中位扩展和位截断：

扩展 { 无符号数：0扩展（前面补0）  
带符号数：符号扩展（前面补符）

截断 —— 强行将高位丢弃，故可能发生溢出

不同类型转换：

char → int —— 前面补0

int ↔ unsigned —— 都可能溢出丢失数据

int → float —— 不会发生溢出，但可能有数据舍入

double → float/int —— 可能发生溢出，可能舍入

float/double → int —— 数据向0方向截断

char → int → long → double  
float → double —— 范围精度由小变大，无损失

## 数据的存储

**大端方式**：从最高有效字节到最低有效字节的顺序存储数据（**从左到右**，正常思维）

**小端方式**：从最低有效字节到最高有效字节的顺序存储数据（**从右到左**，便于机器）

**边界对齐问题**：存储数据不满字长（首地址不能整除自身长度），填充空白，虽然浪费空间，但提高效率（若数据不对齐可能额外存取一次）

[2010 年] 假定变量  $i$ 、 $f$  和  $d$  的数据类型分别为 `int`、`float` 和 `double` (`int` 型用补码表示，`float` 型和 `double` 型分别用 IEEE754 单精度和双精度浮点数格式表示)，已知  $i = 785$ 、 $f = 1.5678E3$ ， $d = 1.5E100$ ，若在 32 位机器中执行下列关系表达式，则结果为“真”的是 ( )。

I.  $i == (\text{int})(\text{float})i$

II.  $f == (\text{float})(\text{int})f$

III.  $f == (\text{float})(\text{double})f$

IV.  $(d + f) - d == f$

A. 仅 I 和 II     B. 仅 I 和 III     C. 仅 II 和 III     D. 仅 III 和 IV

[2016 年] 有如下 C 语言程序段：

```
short si = -32767;
```

```
unsigned short usi = si;
```

执行上述两条语句后，`usi` 的值为 ( )。

A. -32767

B. 32767

C. 32768

D. 32769

[2012 年] 某计算机存储器按字节编址，采用小端方式存放数据。假定编译器规定 `int` 型和 `short` 型长度分别为 32 位和 16 位，并且数据按边界对齐存储。某 C 语言程序段如下：

```
struct {
```

```
int a; char b; short c;
```

```
} record;
```

```
record.a = 273;
```

若 `record` 变量的首地址为 `0xC008`，地址 `0xC008` 中的内容及 `record.c` 的地址分别为 ( )

A. `0x00`，`0xC00D`

B. `0x00`，`0xC00E`

C. `0x11`，`0xC00D`

D. `0x11`，`0xC00E`

[2016 年] 某计算机字长为 32 位，按字节编址，采用小端方式存放数据。假定有一个 `double` 型变量，其机器数表示为 `1122334455667788H`，存放在以 `00008040H` 开始的连续存储单元中，则存储单元 `00008046H` 中存放的是 ( )。

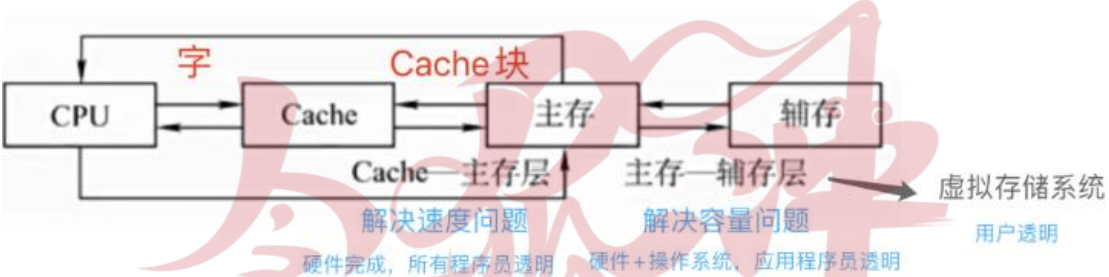
A. `22H`

B. `33H`

C. `77H`

存储系统

对比项	SRAM（静态随机存储器）	DRAM（动态随机存储器）
存储原理	触发器（双稳态电路存储0/1）	电容（充放电存储电荷）
读出特性	非破坏性读出	破坏性读出（需重写）
刷新需求	不需要刷新	需刷新（电荷仅维持2ms）
地址线设计	同时送行列地址	地址线复用（分两次送行列地址，减少引脚数）
运行速度	快（无需刷新/重写操作）	慢（受刷新周期影响）
集成度	低（每个存储单元需6个逻辑元件）	高（仅需1-3个逻辑元件）
芯片体积	大	小
存储成本	高（结构复杂）	低
主要用途	高速缓存（Cache）	主存储器



芯片引脚：

**SRAM：** 芯片引脚 = 地址线 + 数据线 + 片选线(1) + 读写控制线(2)

**DRAM：** 芯片引脚 = 地址线/2（地址复用） + 数据线 + 行列通选线(2) + 读写控制线(2)（片选线由行列通选线代替）

[2010 年] 下列有关 RAM 和 ROM 的叙述中，正确的是 ( )。

- I. RAM 是易失性存储器，ROM 是非易失性存储器
  - II. RAM 和 ROM 都采用随机存取方式进行信息访问
  - III. RAM 和 ROM 都可用作 Cache
  - IV. RAM 和 ROM 都需要进行刷新
- A. 仅 I 和 II  
B. 仅 II 和 III  
C. 仅 I、II 和 III  
D. 仅 II、III 和 IV

[2014 年] 某容量为 256MB 的存储器由若干 4M×8 位的 DRAM 芯片片构成，该 DRAM 芯片的地址引脚和数据引脚总数是 ( )。

- A. 19  
B. 22

- C. 30  
D. 36

[2018 年] 假定 DRAM 芯片中存储阵列的行数为  $r$ 、列数为  $c$ ，对于一个  $2K \times 1$  位的 DRAM 芯片，为保证其地址引脚数最少，并尽量减少刷新开销，则  $r$ 、 $c$  的取值分别是 ( )。

- A. 2048, 1  
B. 64, 32  
C. 32, 64  
D. 1, 2048

## 主存储器——性能指标

64×8等等	
主存容量	计算机可寻址的最小信息单元是一个存储字
	主存储器存储单元的总数
存取速度	由存储器存取时间和存储周期表示
存储器存取时间	启动一次存储器操作(读/写)到完成该操作所经历的时间
存储周期	连续启动两次独立的存储器操作所间隔的最小时间

## 主存储器——容量扩展

**位扩展：**用多个存储器芯片对字长进行扩充，地址线连接方式不变

**字扩展：**增加存储器中字的数量，提高存储器的寻址范围，由片选信号区分芯片的地址范围

**线选法：**用除片内寻址外的高位地址线直接连接至各个存储芯片的片选端， $n$  条线对应  $n$  个选片信号

**译码片选法：**用除片内寻址外的高位地址通过译码器产生片选信号， $n$  条线对应  $2n$  个选片信号

**字位扩展：**假设一个存储器的容量为  $M \times N$  位，若使用  $L \times K$  位存储器芯片，那么，这个存储器共需要  $(M/L) \times (N/K)$  个存储器芯片

[2009 年] 某计算机主存容量为 64 KB，其中 ROM 区为 4 KB，其余为 RAM 区，按字节编址。现要用  $2K \times 8$  位的 ROM 芯片和  $4K \times 4$  位的 RAM 芯片来设计该存储器，需要上述规格的 ROM 芯片数和 RAM 芯片数分别是 ( )。

- A. 1, 15  
B. 2, 15  
C. 1, 30

## 主存储器——多体交叉存储器

**单体多字存储器：**存储器只有一个存储体（每个存储单元存储  $m$  个字，总线宽度  $m$  个字，每次性并行取出  $m$  个字）

**多体并行存储器：**

**高位交叉编址：**通常会连续访问，因此实际效果相当于单纯的扩容

**低位交叉编址：**

程序存放在相邻模块中，采用流水线存取

**存取周期  $T =$  存取时间  $r$ （总线传输周期） + 恢复时间**

当存储模块数  $\geq T/r$  时，可使流水线不间断

存储器宽度：存取速率 = 字数 · 字长 / 时间

**每个存储周期内可读写地址连续的  $m$  个字，取  $m$  个字耗时： $T + (m - 1) \cdot r$**

微观上， $m$  个模块被串行访问；宏观上，每个存储周期内所有模块被并行访问

[2015 年] 某计算机使用四体交叉编址存储器，假定在存储器总线上出现的主存地址（十进制）序列为 8005, 8006, 8007, 8008, 8001, 8002, 8003, 8004, 8000，则可能发生访存冲突的地址对是（ ）。

- A. 8004 和 8008
- B. 8002 和 8007
- C. 8001 和 8008
- D. 8000 和 8004

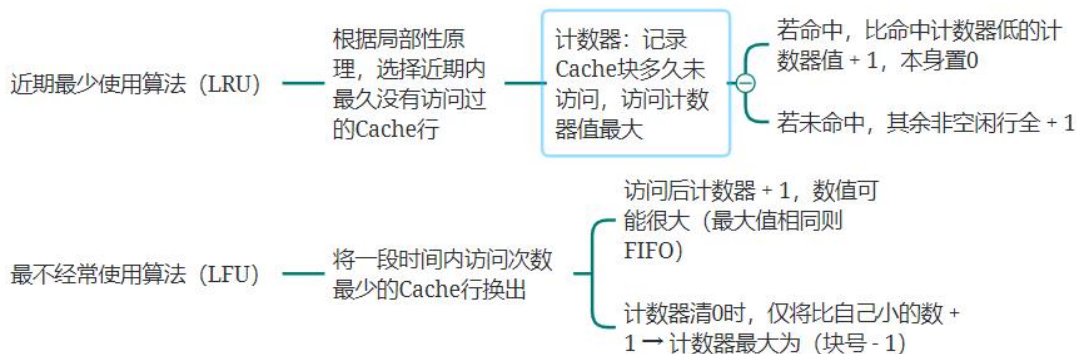
[2017 年] 某计算机主存按字节编址，由 4 个 64M×8 位的 DRAM 芯片采用交叉编址方式构成，并与宽度为 32 位的存储器总线相连，主存每次最多读/写 32 位数据。若 double 型变量  $x$  的主存地址为 804001AH，则读取  $x$  需要的存储周期数是（ ）

- A. 1
- B. 2
- C. 3
- D. 4

## 高速缓冲存储器——Cache 替换算法

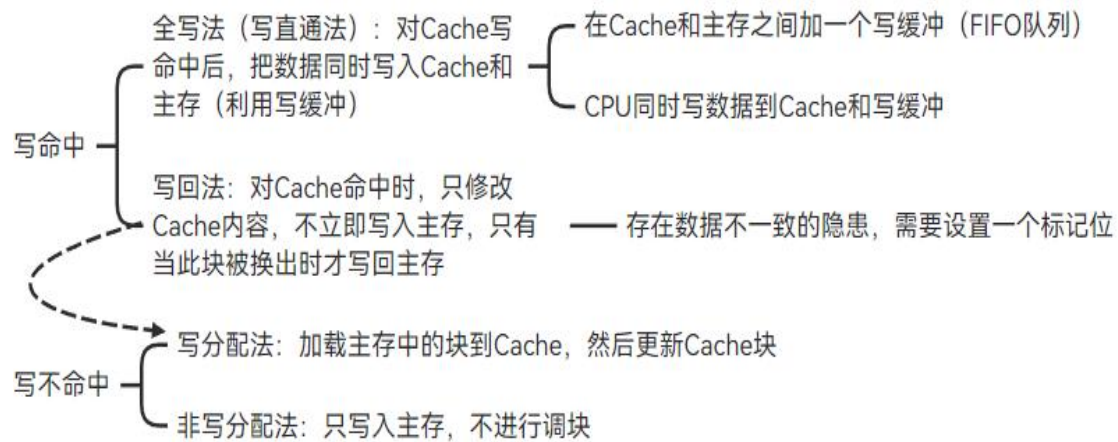
随机算法 —— 随机确定替换的Cache块

先进先出算法（FIFO） —— 选择最早调入的进行替换



抖动：频繁换入换出（频繁访问块数 > Cache 行数量）

## 高速缓冲存储器——Cache 一致性问题



[2009 年] 某计算机的 Cache 共有 16 块，采用二路组相联映射方式 (即每组 2 块)。每个主存块大小为 32 B，按字节编址，主存 129 号单元所在主存块应装入的 Cache 组号是 ( )。

- A. 0
- B. 2
- C. 4
- D. 6

[2021 年] 若计算机主存地址为 32 位，按字节编址，Cache 数据区大小为 32 KB，主存块大小为 32 B，采用直接映射方式和回写 (Write Back) 策略，则 Cache 行的位数至少是 ( )。

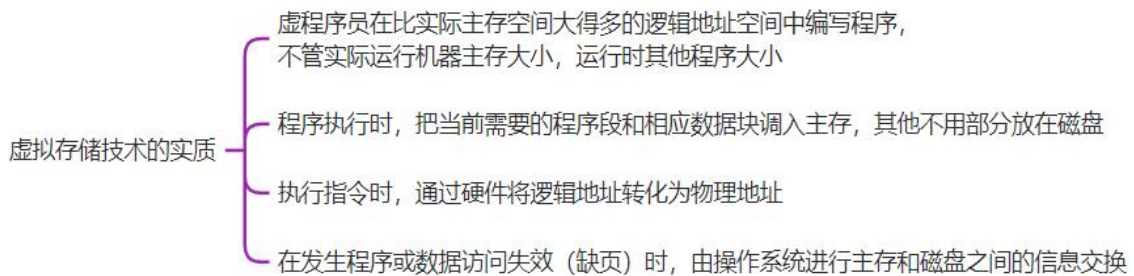
- A. 275
- B. 274
- C. 258
- D. 257

[2022 年] 若计算机主存地址为 32 位，按字节编址，某 Cache 的数据区容量为 32 KB，主存块大小为 64 B，采用 8 路组相联映射方式，该 Cache 中比较器的个数和位数分别为 ( )。

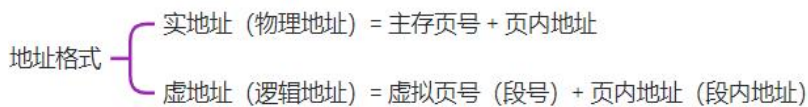
- A. 8, 20
- B. 8, 23
- C. 64, 20
- D. 64, 23

## 虚拟存储器

概念：采用“按需调页”方式分配主存



大页面、全相联、写回法 — 磁盘开销大，故页命中率比Cache命中率更重要



## 页式虚拟存储器

- CPU生成的虚拟地址被划分为两部分：虚拟页号(VPN) | 页内偏移量(Offset)
  - 若有快表：MMU通过VPN在TLB中查找对应的物理页框号PFN
  - 若TLB中存在有效映射（VPN→PFN），直接获取PFN，跳转到物理地址生成步骤。
  - TLB缺失：MMU需查询主存中的页表（多级页表可能需多次访问），从页表获取PFN后，更新TLB缓存该映射（按替换策略如LRU淘汰旧条目）
- 页表查询：通过页表基址寄存器(PTBR)定位当前进程的页表起始地址。使用虚拟页号作为索引查找页表项(PTE)，获取对应的物理页框号(PFN)和状态位
- 若页表项的有效位为0（页未调入内存），触发缺页中断：操作系统选择替换页，从磁盘加载目标页到空闲物理页框，更新页表项的有效位和物理页框号
- 物理地址生成：将页表项中的物理页框号与虚拟地址的页内偏移量拼接，得到物理地址。
- 内存访问：使用物理地址访问主存数据。若数据不在Cache中，触发Cache缺失并从主存加载

## 段式虚拟存储器

便于编写程序，编译器优化和操作系统调度管理（对程序员不透明）

段可作为独立逻辑单位被其他程序调用，形成段间连接，产生规模较大的程序

- CPU生成的虚拟地址被划分为两部分：段号 | 段内偏移量
- 段表查询：从段表基址寄存器中获取当前进程的段表起始地址。以段号为索引，在段表中查找对应段表项。段表项：段号 + 段首址 + 装入位 + 段长
- 越界检查：若偏移量  $\geq$  段长，触发越界中断。
- 若装入位为0（段未在内存），触发缺段中断：操作系统选择替换段，从外存加载目标段到空闲内存区域，更新段表项基址和装入位
- 将段基址与段内偏移量相加，得到物理地址(PA)， $PA = \text{段基址} + \text{Offset}$ 。
- 使用物理地址访问主存数据。若数据不在Cache中，触发Cache缺失并加载

## 段页式虚拟存储器

段、页式结合，程序按模块分段，段内分页，程序对主存的调入调出仍以页为基本传送单位

1. 获取段表基址：通过段表基址寄存器（STBR）找到当前进程的段表起始地址
2. 定位段表项（STE）：以段号S为索引，在段表中查找对应表项，获取该段的页表起始地址和页表长度
3. 根据段表项中的页表起始地址和段内页号P，找到页表项，若页未调入（有效位为0），触发缺页中断。
4. 缺页中断处理（若触发）
5. 将物理页框号（PFN）与页内偏移量D拼接，得到物理地址（PA）
6. 通过PA访问物理内存数据，若数据不在Cache中，触发Cache缺失并加载。

[2010 年] 下列命令组合的一次访存过程中，不可能发生的是（ ）。

- A. TLB 未命中，Cache 未命中，Page 未命中
- B. TLB 未命中，Cache 命中，Page 命中
- C. TLB 命中，Cache 未命中，Page 命中
- D. TLB 命中，Cache 命中，Page 未命中

[2020 年] 下列关于 TLB 和 Cache 的叙述中，错误的是（ ）。

- A. 命中率都与程序局部性有关
- B. 缺失后都需要去访问主存
- C. 缺失处理都可以由硬件实现
- D. 都由 DRAM 存储器组成

[2022 年] 某计算机主存地址为 24 位，采用分页虚拟存储管理方式，虚拟地址空间大小为 4 GB，页大小为 4 KB，按字节编址。某个进程的页表部分内容如下表所示。当 CPU 访问虚拟地址 00082840H 时，虚一实地址转换的结果是（ ）。

- A. 得到主存地址 024840H
- B. 得到主存地址 180840H
- C. 得到主存地址 018840H
- D. 检测到缺页异常

虚页号	页框号	存在位
82	024H	0
...	...	...
129	180H	1
130	018H	1

## 外部存储器——磁盘存储器

磁盘驱动器 + 磁盘控制器 + 盘片

磁盘地址 —— 驱动器号 + 柱面（磁道）号 + 盘面号 + 扇区号

磁盘容量：存储的字节总数

- 非格式化容量：磁化单元总数
- 格式化容量：按某种记录格式所能存储信息的总量

记录密度

- 道密度：沿磁盘半径方向单位长度的磁道数
- 位密度：磁道单位长度上能记录的二进制代码位数
- 面密度：位密度与道密度的乘积

平均存取时间 = 寻道时间（磁头移动到目的磁道） + 旋转延迟时间（磁头定位到所在扇区） + 传输时间（传输数据所花费的时间） —— 平均延迟时间 = 转半圈时间

数据传输率：单位时间向主机传送数据的字节数（磁盘转速 $r$ 转每秒，每条磁道容量 $N$ 个字节，数据传输率 $D_r = r \cdot N$ ）

磁盘阵列：提升读写速度，提升容错能力

RAID0 —— 无冗余和无校验的磁盘阵列

RAID1 —— 镜像的磁盘阵列

RAID2 —— 纠错的海明码的磁盘阵列

RAID3 —— 位交叉奇偶校验的磁盘阵列

RAID4 —— 块交叉奇偶校验的磁盘阵列

RAID5 —— 块级分布式奇偶校验的磁盘阵列

RAID6 —— 二维块交叉奇偶校验的磁盘阵列

## 外部存储器——固态硬盘 SSD

原理：Flash Memory，电可擦除ROM，即EEPROM

组成

- 闪存翻译层：负责翻译逻辑块号，找到对应页
- 存储介质：闪存芯片组

读写

- 页为单位，相当于磁盘扇区
- 以块为单位擦除，每页可以写一次，读无限次
- 随机访问。读快写慢，写的页如果有数据不能写入，需要将块内其他页复制到新的块中，写入新的页

同一块擦除过多可能坏掉 —— 磨损均衡技术

- 动态磨损均衡 —— 写入数据时，优先选择累计擦除次数少的新闪存块
- 静态磨损均衡 —— SSD监测并自动进行数据分配、迁移，让老旧的块承担更多读为主的任务

[2013 年] 某磁盘的转速为 10000 转/分，平均寻道时间是 6ms，磁盘传输速率是 20MB/s，磁盘控制器延迟为 0.2ms，读取一个 4KB 的扇区所需的平均时间约为 ( )。

- A. 9 ms
- B. 9.4 ms
- C. 12 ms
- D. 12.4 ms

[2019 年] 下列关于磁盘存储器的叙述中，错误的是 ( )。

- A. 磁盘的格式化容量比非格式化容量小
- B. 扇区中包含数据、地址和校验等信息
- C. 磁盘存储器的最小读/写单位为 1 字节
- D. 磁盘存储器由磁盘控制器、磁盘驱动器和盘片组成

