

# 第7章 布置作业

- 课后4、6、7、9、10、17、24

4. 用  $64\text{K} \times 1$  位的 DRAM 芯片构成  $256\text{K} \times 8$  位的存储器，要求：

1) 计算所需芯片数，并画出该存储器的逻辑框图。

2) 若采用分散刷新方式，每单元刷新间隔不超过  $2\text{ms}$ ，则刷新信号周期是多少？若采用集中刷新方式，则存储器刷新一遍最少用多少读/写周期？

参考答案：

1)  $256\text{KB} / 64\text{K} \times 1 \text{ 位} = 32 \text{ 片}$ 。

2) 采用分散刷新时，刷新周期是存储周期的 2 倍；

因为 DRAM 芯片存储阵列为  $64\text{K}=256 \times 256$ ，所以集中刷新时，整个存储器刷新一遍需要 256 个存储（读写）周期。

（异步刷新的刷新周期为  $2\text{ms}/256=7.8\mu\text{s}$ ）

## P289 第6题 参考答案

CPU 地址线共 16 位，故存储器地址空间为：0000H – FFFFH，其中 8000H – FFFFH 为 RAM 区，共  $2^{15}=32K$  个单元，容量为 32KB，故需 8Kx4 位的芯片数为  $32KB/8Kx4 \text{ 位}=4x2=8$  片。

若 CPU 地址线为 24 位，ROM 区为 000000H – 007FFFH，则如何？

ROM 区大小为 32KB，总大小为  $16MB=2^{14}KB=512x32KB$ ，所以 RAM 区大小为  $511x32KB$  共需使用 RAM 芯片数为  $511x32KB/8Kx4 \text{ 位}=511x4x2$  个芯片。

## P289 第7题 参考答案

对于4体交叉访问的存储系统，理想情况下，每隔 $1/4$ 周期可读写一个数据，假定这个时间为 $\Delta t$ 。每个存储模块内的地址分布如下：

模块0：0、4、8、12、16 ...

模块1：1、5、9、13、17、... 37、...、41、...

模块2：2、6、10、14、18、...

模块3：3、7、11、15、19、... 51、...、67、...

很显然，如果相邻4次访问中给定的访存地址出现在同一模块内，就会发生访存冲突，所以17和9、37和17、13和37、8和4会发生冲突。41和13也在同一模块内且访问间隔小于4个 $\Delta t$ ，但是由于访问第8单元发生冲突而使其访问延迟3个 $\Delta t$ 进行，从而使得41号单元也延迟3个 $\Delta t$ 访问，因而，其访问不会和13号单元的访问发生冲突。

## P289 第9题 参考答案

对于按行优先存放在内存的多维数组，如果按列优先访问数组元素，则空间局部性就差；如果在一个循环体中某个数组元素只被访问一次，则时间局部性就差。

假定二维数组a[1000][1000]按行优先存放在内存，以下给出的4个程序片段用于对数组a进行相应的处理，它们具有相同的功能，但数组访问的时间局部性和空间局部性截然不同(不考虑编译器的优化)

(1) 时间局部性和空间局部性都好的程序片段：

```
for (i=0;i<1000; i++)
    for (j=0;j<1000;j++){
        sum=sum+a[i][j];
        product=product*a[i][j];
        square=square+a[i][j]*a[i][j];}
```

(2) 时间局部性好，空间局部性差的程序片段：

```
for (j=0;j<1000; j++)
    for (i=0;i<1000;i++){
        sum=sum+a[i][j];
        product=product*a[i][j];
        square=square+a[i][j]*a[i][j];}
```

(3) 空间局部性好，时间局部性差的程序片段：

(4) 时间局部性和空间局部性都差的程序片段：

## P289 第9题 参考答案

对于按行优先存放在内存的多维数组，如果按列优先访问数组元素，则空间局部性就差；如果在一个循环体中某个数组元素只被访问一次，则时间局部性就差。

假定二维数组a[1000][1000]按行优先存放在内存，以下给出的4个程序片段用于对数组a进行相应的处理，它们具有相同的功能，但数组访问的时间局部性和空间局部性截然不同(不考虑编译器的优化)

(1) 时间局部性和空间局部性都好的程序片段：

(2) 时间局部性好，空间局部性差的程序片段：

(3) 空间局部性好，时间局部性差的程序片段：

```
for (i=0;i<1000; i++)
    for (j=0;j<1000;j++)
        sum=sum+a[i][j];
for (i=0;i<1000; i++)
    for (j=0;j<1000;j++)
        product=product*a[i][j];
for (i=0;i<1000; i++)
    for (j=0;j<1000;j++)
        square=square+a[i][j]*a[i][j];
```

(4) 时间局部性和空间局部性都差的程序片段：

## P289 第9题 参考答案

对于按行优先存放在内存的多维数组，如果按列优先访问数组元素，则空间局部性就差；如果在一个循环体中某个数组元素只被访问一次，则时间局部性就差。

假定二维数组a[1000][1000]按行优先存放在内存，以下给出的4个程序片段用于对数组a进行相应的处理，它们具有相同的功能，但数组访问的时间局部性和空间局部性截然不同(不考虑编译器的优化)

(1) 时间局部性和空间局部性都好的程序片段：

(2) 时间局部性好，空间局部性差的程序片段：

(3) 空间局部性好，时间局部性差的程序片段：

(4) 时间局部性和空间局部性都差的程序片段：

```
for (j=0;j<1000; j++)
    for (i=0;i<1000;i++)
        sum=sum+a[i][j];
for (j=0;j<1000; j++)
    for (i=0;i<1000;i++)
        product=product*a[i][j];
for (j=0;j<1000; j++)
    for (i=0;i<1000;i++)
        square=square+a[i][j]*a[i][j];
```

## P289 第10题 参考答案

(1) cache共有 $64\text{KB}/128\text{B}=512$ 行，直接映射方式下，cache行号占9位；由于每个主存块大小为128B，按字节编址，故块内地址为7位；主存地址空间大小为1GB，所以主存地址位数为30位。主存地址中标记有 $30-9-7=14$ 位。

所以，主存地址共有以下三个字段：高14位为标记，中间9位为行索引，低7位为块内地址。

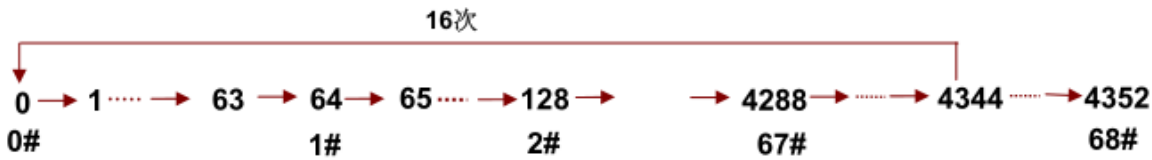
(2) 因为直接映射不考虑替换算法，所以cache行中没有用于替换的控制位；因为采用直写方式，所以cache行中也没有修改位。每个cache行中包含1位有效位、14位标记位和128B的数据，因此cache总容量为 $512 * (1+14+128*8) = 519.5\text{KB}$ 。



# P291 第17题 参考答案

- (1) cache的行数为 $4\text{KB}/64\text{B}=64$ ;因为采用4路组相联, 所以每组有4行, 共16组。主存地址空间大小为64KB, 按字节编址, 所以主存地址有16位, 其中低6位为块内地址, 中间4位为组号(组索引), 高6位为标记。
- (2)因为采用写回策略, 所以cache每行中要有一个修改位(dirty bit): 因为每组有4行, 所以每行有2位LRU位, 此外, 每行还有6位标记、1位有效位和64KB数据, 共有64行, 故总容量为 $64*(6+1+1+2+64*8)=33408$ 位。
- (3)因为块大小为64B, CPU总共访问了4345个单元, 因为 $4345/64\approx 67.89$ , 所以第0~4344单元应该对应前68块(第0~67块), 也即CPU访问过程是对主存的前68块连续访问16次。下图给出了访问过程中主存块和cache行之间的映射关系。 图中列方向是cache的16个组, 行方向是每组的4行。

所以 0~4344 单元应该对应前 68 块, 也即处理器的访问过程是对前 68 块连续访问 16 次。



总访存次数:  $16\times 4345=69520$   
Cache 共有 16 组, 每组 4 行, 替换情况如下图所示:

	第0行	第1行	第2行	第3行
0组	0/64/48	16/0/64	32/16	48/32
1组	1/65/49	17/1/65	33/17	49/33
2组	2/66/50	18/2/66	34/18	50/34
3组	3/67/51	19/3/67	35/19	51/35
4组	4	20	36	52
...	...	...	...	...
...	...	...	...	...
15组	15	31	47	63

## P291 第17题 参考答案

(3)因为块大小为64B，CPU总共访问了4345个单元，因为 $4345/64 \approx 67.89$ ，所以第0~4344单元应该对应前68块(第0~67块)，也即CPU访问过程是对主存的前68块连续访问16次。下图给出了访问过程中主存块和cache行之间的映射关系。图中列方向是cache的16个组，行方向是每组的4行。

主存的第0~15块分别对应cache的第0~15组，可以放在对应组的任意一行中，在此假定按顺序存放在第0行；主存的第16~31块也分别对应cache的第0~15组，放到第1行中；同理，主存的第32~47块分别放到cache的第0~15组的第3行中。这样访问主存的第0~63块都是没有冲突的，每块都是第一次cache中没有找到，然后把这一块调到cache对应组的某一行中，这样该块后面的每次访问都能在cache中找到。因此，每一块只有第一单元未命中，其余63个单元都命中。

主存的第64~67块分别对应cache的第0~3组，此时，这4组的4个行都已被占满，选择一个主存块淘汰出来，采用LRU算法，第0~3块替换出来，

**LRU 算法：**第一次循环，每一块只有第一次未命中，其余都命中；

以后 15 次循环，有 20 块的第一字未命中，其余都命中。

所以,命中率  $p$  为  $(69520-68-15 \times 20)/69520=99.47\%$

平均访存时间为：  $p \times \text{Hit Time} + (1-p) \times \text{Miss Penalty}$

$=1 \times p + 10 \times (1-p) = 1 \times 0.9947 + 10 \times 0.0053 = 1.0477$  个时钟周期

## □P292第24题 参考答案

参考答案：

- 1) 16 位虚拟地址中低 7 位为页内偏移量，高 9 位为虚页号；虚页号中高 7 位为 TLB 标记，低 2 位为 TLB 组索引。
- 2) 12 位物理地址中低 7 位为页内偏移量，高 5 位为物理页号。
- 3) 12 位物理（主存）地址中，低 2 位为块内地址，中间 4 位为 Cache 行索引，高 6 位为标志。
- 4) 地址 067AH=0000 0110 0111 1010B，所以，虚页号为 0000011 00B，映射到 TLB 的第 00 组，将 0000011B=03H 与 TLB 第 0 组的四个标志比较，虽然和其中一个相等，但对应的有效位为 0，其余都不等，所以 TLB 缺失，访问主存中的慢表。

直接查看 0000011 00B =00CH 处的页表项，有效位为 1，取出物理页号 19H=11001B，和页内偏移 111 1010B 拼接成物理地址：11001 111 1010B。根据中间 4 位 1110 直接找到 Cache 第 14 行(即：第 E 行)，有效位为 1，且标记为 33H=110011B，正好等于物理地址高 6 位，故命中。根据物理地址最低两位 10，取出 4AH=01001010B。

## 补充题. 2010年考研题

(12 分)某计算机的主存地址空间大小为 256 MB,按字节编址。指令 Cache 和数据 Cache 分离,均有 8 个 Cache 行,每个 Cache 行大小为 64 B,数据 Cache 采用直接映射方式。现有两个功能相同的程序 A 和 B,其伪代码如下所示: **注意: 行也就是块, 即块大小为64字节**

程序 A:

```
int a[256][256];
...
int sum_array 1( )
{
    int i,j,sum=0;
    for(i=0;i<256;i++)
        for(j=0;j<256;j++)
            sum+=a[i][j];
    return sum;
}
```

程序 B:

```
int a[256][256];
...
int sum_array 2( )
{
    int i,j,sum=0;
    for(j=0;j<256;j++)
        for(i=0;i<256;i++)
            sum+=a[i][j];
    return sum;
}
```

**注意: 程序A-行优先访问, 程序B-列优先访问。**

## 2010年考研题 (续)

假定 `int` 类型数据用 32 位补码表示, 程序编译时 `i`, `j`, `sum` 均分配在寄存器中, 数组 `a` 按行优先方式存放, 其首地址为 320 (十进制数)。请回答下列问题, 要求说明理由或给出计算过程。

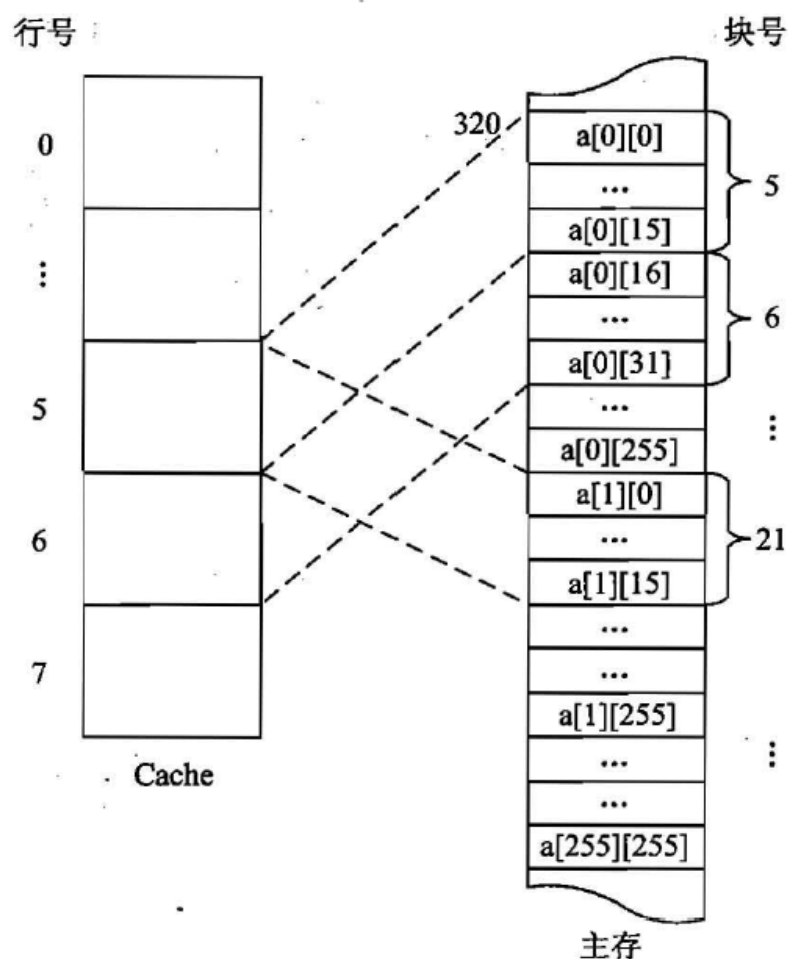
- (1) 若不考虑用于 Cache 一致性维护和替换算法的控制位, 则数据 Cache 的总容量为多少?
- (2) 数组元素 `a[0][31]` 和 `a[1][1]` 各自所在的主存块对应的 Cache 行号分别是多少 (Cache 行号从 0 开始)?
- (3) 程序 A 和 B 的数据访问命中率各是多少? 哪个程序的执行时间更短?

# 2010年考研题

## 【答案要点】

(1) 数据 Cache 的总容量为:4 256 位(532 字节)。

(2) 数组 a 在主存的存放位置及其与 Cache 之间的映射关系如下图所示。a[0][31]所在主存块映射到 Cache 第 6 行, a[1][1]所在主存块映射到 Cache 第 5 行。



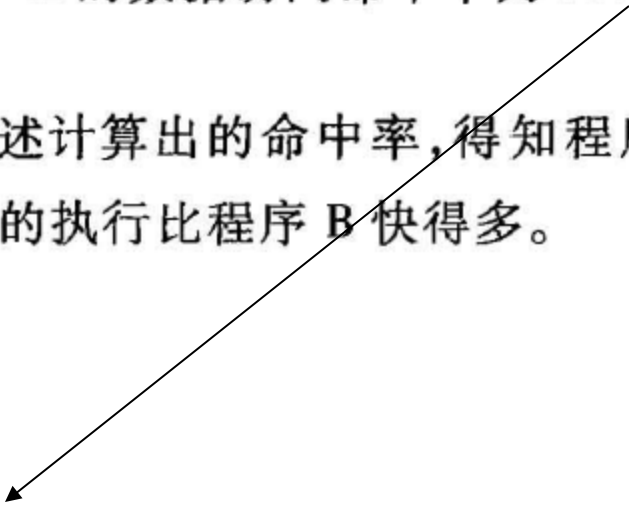


## 2010年考研题

(3) 编译时  $i$ ,  $j$ ,  $sum$  均分配在寄存器中, 故数据访问命中率仅考虑数组  $a$  的情况。

① 程序 A 的数据访问命中率为 93.75%; ② 程序 B 的数据访问命中率为 0。

根据上述计算出的命中率, 得知程序 B 每次取数都要访问主存, 所以程序 A 的执行比程序 B 快得多。


$$15/16=0.9375$$

## 2010年考研题 (具体解释)

1) 主存容量256MB，按字节寻址的地址位数应为28位，数据Cache分为8行（用3位地址），每行64B（用6位地址），因此Cache中每个字块的Tag字段的位数应是 $28-9=19$ 位，还要使用一个有效位，二者合计为20位；因此数据Cache的总容量应为： $64B \times 8 + (20/8 \times 8)B = 532B$

2) 数组a在主存的存放位置及其与Cache之间的映射关系如下图所示。

数组A[0][31]所在的主存块对应的Cache行号是：

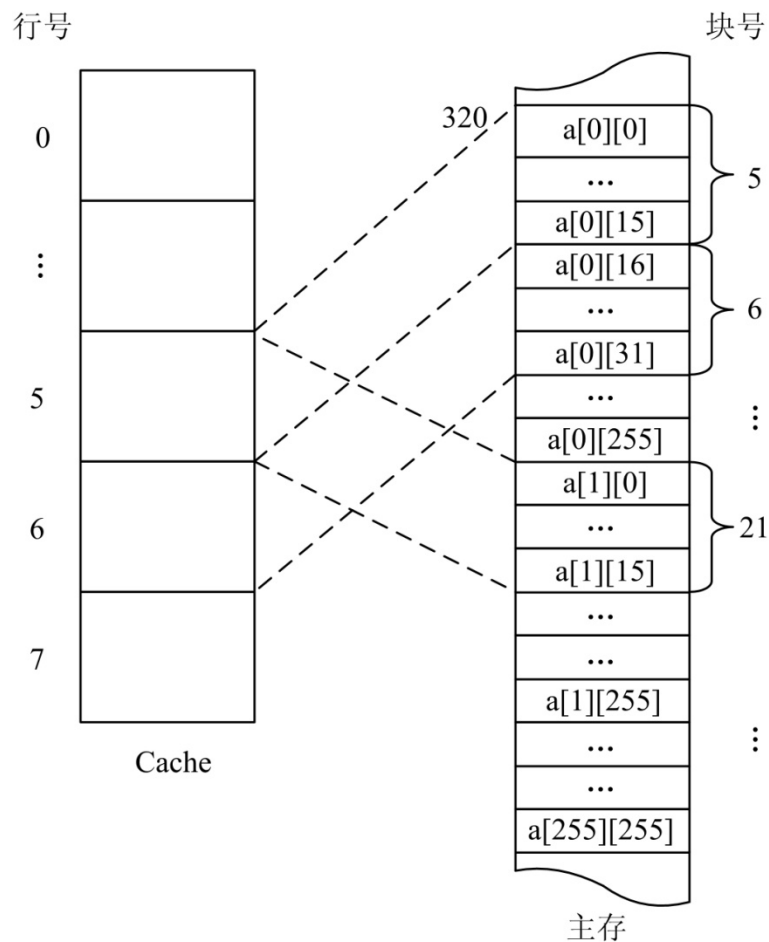
$$(320 + 31 \times 4) \div 64 = 6,$$

数组A[1][1]所在主存块对应的Cache行号：

$$((320 + 256 \times 4 + 1 \times 4) \div 64) \bmod 8 = 5.$$

所以 a[0][31]所在主存块映射到Cache第6行，

a[1][1]所在主存块映射到Cache第5行。





## 2010年考研题 (具体解释)

3)编译时i, j, sum均分配在寄存器中，故数据访问命中率仅考虑数组a的情况。

①这个程序的特点是数组中的每一个int 类型的数据只被使用一次。数组A按行优先存放，数据Cache正好放下数组半行中的全部数据，即数据的存储顺序与使用次序有更高的吻合度，每个字块存16个int类型的数据，访问每个字块中头一个字不会命中，但接下来的15个字都会命中，访问全部字块都符合这一规律，命中率是15/16，即程序A的数据访问命中率为93.75%；

②程而程序B是按照数组的列执行外层循环，在内层循环过程中，将连续访问不同行的同一列的数据，不同行的同一列数据使用的是同一个Cache单元，每次都不会命中，命中率是0，程序执行特别慢。

根据上述计算出的命中率，得出程序B每次取数都要访问主存，所以程序A的执行比程序B快得多。