《计算机组成原理》

（第四讲习题答案）

厦门大学信息学院软件工程系 曾文华

2023年4月20日

第4章　存储系统

- 4.1　存储器概述
- 4.2　半导体存储器
- 4.3　主存的组织及与CPU 的连接
- 4.4　并行主存系统
- 4.5　高速缓冲存储器
- 4.6　虚拟存储器

- 例4.1：设计算机字长为64位，存储器容量为128MW（W表示字=64位，因此存储器的容量=128x2^{20}x64位），采用模数为8的存储器交叉方式进行组织（即m=8），存储周期T=200ns，数据总线宽度为64位，总线传输周期τ=25ns（T=mτ，200ns=8x25ns），若连续读8个字，分别计算顺序访问方式和交叉访问方式下存储器的带宽。

- 解：
  - 顺序方式下，连续读8个字所需的时间=8T=8x200ns=1600ns
  - 顺序方式下的存储器带宽=(8个字x64位)/1600ns=3.2x10^8bps=320Mbps

  - 交叉方式下，连续读8个字所需的时间 = T + (8-1)τ = 200ns + 7x25ns = 375ns
  - 交叉方式下的存储器带宽=(8个字x64位)/375ns≈1.37x10^9bps=1370Mbps

  - 交叉方式比顺序方式快：1370/320 = 4.27倍

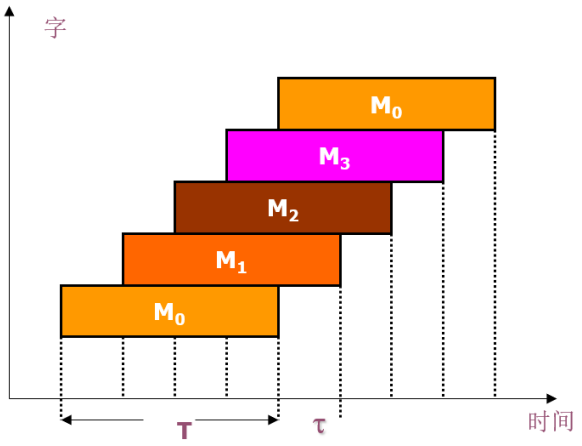


图4.30　流水存取示意图

- 例4.2：分析下面C语言代码的程序局部性：

```
int sumvec()
{
     int i,sum=0;
     for (i=0; i<1000; i++)
          sum += V[i];
     return sum;
}
```

- 解：
  - for循环体中的指令在主存中顺序存放，具有空间局部性；for循环体将被执行1000次，因此还具有时间局部性
  - 变量i、sum是单个变量，不存在空间局部性；但是这两个变量在每次循环执行时都会用到，因此具有时间局部性
  - 数组V[i]中包含1000个数据元素，这些数据在内存中是顺序存放，因此具有空间局部性

- 例4.3　某计算机字长32位，采用直接相联映射方式；主存容量4MB，按字节编址；cache的数据块副本缓冲区容量为4KB，cache块长度为8个字。
  ① 画出直接映射方式下主存地址划分情况，并说明每个字段位数。
  ② 设cache初始状态为空，若CPU顺序访问0-99号单元（字），从中读出100个字（假设主存一次读一个字），并重复此顺序10次，请计算cache的命中率。
  ③ 如果cache的存取时间是20ns，主存的访问时间是200ns，请问平均访问时间是多少？
  ④ cache-主存系统访问效率是多少？

- 解：
  – ①
    - 直接相联映射方式：主存地址 = 区地址（tag） + 行索引（index） + 块内偏移（offset）；cache地址 = cache块地址 + 块内偏移（offset）
    - 主存容量=4MB，因此主存地址=22位（2^{22}=4M）；cache的数据块副本缓冲区容量=4KB，因此cache地址=12位（2^{12}=4K）
    - cache块长度为8个字=8x32位=32B（字长=32位），因此offset=5位（2^5=32），cache块地址=12-5=7位
    - 因为cache的块地址与主存的行索引相等，因此：index=7位
    - 主存的区地址=22-7-5=10位
    - 因此，主存地址（22位） = 区地址（10位） + 行索引（7位） + 块内偏移（5位）

- 例4.3　某计算机字长32位，采用直接相联映射方式；主存容量4MB，按字节编址；cache的数据块副本缓冲区容量为4KB，cache块长度为8个字。
  ① 画出直接映射方式下主存地址划分情况，并说明每个字段位数。
  ② 设cache初始状态为空，若CPU顺序访问0-99号单元（字），从中读出100个字（假设主存一次读一个字），并重复此顺序10次，请计算cache的命中率。
  ③ 如果cache的存取时间是20ns，主存的访问时间是200ns，请问平均访问时间是多少？
  ④ cache-主存系统访问效率是多少？

- 解（续）：
  – ②
    - cache的每块有8个字（单元）；一开始cache是空的，CPU访问0号单元将不命中，并主存的将0～7号单元装入cache中，因此CPU访问1～7号单元将命中
    - 之后， CPU访问8号单元不命中，访问9～15号单元命中； CPU访问16号单元不命中，访问17～23号单元命中； CPU访问24号单元不命中，访问25～31号单元命中； CPU访问32号单元不命中，访问33～39号单元命中； CPU访问40号单元不命中，访问41～47号单元命中； CPU访问48号单元不命中，访问49～55号单元命中； CPU访问56号单元不命中，访问57～63号单元命中； CPU访问64号单元不命中，访问65～71号单元命中； CPU访问72号单元不命中，访问73～79号单元命中； CPU访问80号单元不命中，访问81～87号单元命中； CPU访问88号单元不命中，访问89～95号单元命中； CPU访问96号单元不命中，访问97～99号单元命中
    - 接下去，CPU重复访问9次时，全部命中
    - 因此，1000次访问，有13次不命中，其余987次都命中，命中率=987/1000=98.7%

cache数据块副本缓冲区

	第0字	第1字	第2字	第3字	第4字	第5字	第6字	第7字
第0行	0	1	2	3	4	5	6	7
第1行	8	9	10	11	12	13	14	15
第2行	16	17	18	19	20	21	22	23
第3行	24	25	26	27	28	29	30	31
第4行	32	33	34	35	36	37	38	39
第5行	40	41	42	43	44	45	46	47
第6行	48	49	50	51	52	53	54	55
第7行	56	57	58	59	60	61	62	63
第8行	64	65	66	67	68	69	70	71
第9行	72	73	74	75	76	77	78	79
第10行	80	81	82	83	84	85	86	87
第11行	88	89	90	91	92	93	94	95
第12行	96	97	98	99				
…								
…								
…								
第127行								

- 例4.3　某计算机字长32位，采用直接相联映射方式；主存容量4MB，按字节编址；cache的数据块副本缓冲区容量为4KB，cache块长度为8个字。
  ① 画出直接映射方式下主存地址划分情况，并说明每个字段位数。
  ② 设cache初始状态为空，若CPU顺序访问0-99号单元（字），从中读出100个字（假设主存一次读一个字），并重复此顺序10次，请计算cache的命中率。
  ③ 如果cache的存取时间是20ns，主存的访问时间是200ns，请问平均访问时间是多少？
  ④ cache-主存系统访问效率是多少？

- 解（续）：
  - ③
    - 平均访问时间：$t_a=ht_c+(1-h)t_m$
    - 其中，t_c=20ns，t_m=200ns，h=0.987
    - 因此，t_a=0.987x20ns + (1-0.987)x200ns = 22.34ns（接近cache的存取时间）
  - ④
    - 访问效率：$e = t_c/t_a$= 20ns/22.34ns = 89.5%

- **例4.4** 主存地址空间大小为256MB，按字节编址。指令cache和数据cache分离，均有8行，每个cache行的数据块大小均为64B。数据cache采用直接相联映射方式。现有两个功能相同的程序A和B，其伪代码如下所示。假定int型数据为32位补码，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，首地址为320（十进制）。请回答如下问题，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的总容量是多少？
  ② 数组元素a[0][31]、a[1][1]所在主存块对应的cache行分别是多少？假设cache行号从零开始。
  ③ 程序A和B的数据访问命中率各是多少？那个程序的执行时间更短?

程序A:

```
int a[256][256];
int sum_array1()
{
  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_array2()
{
  int i,j,sum=0;
    for (j = 0; j < 256; j++)
        for (i = 0; i < 256; i++)
          sum += a[i][j];
    return sum;
}
```

- 例4.4　主存地址空间大小为256MB，按字节编址。指令cache和数据cache分离，均有8行，每个cache行的数据块大小均为64B。数据cache采用直接相联映射方式。现有两个功能相同的程序A和B，其伪代码如下所示。假定int型数据为32位补码，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，首地址为320（十进制）。请回答如下问题，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的总容量是多少？
  ② 数组元素a[0][31]、a[1][1]所在主存块对应的cache行分别是多少？假设cache行号从零开始。
  ③ 程序A和B的数据访问命中率各是多少？那个程序的执行时间更短?
- 解：
  - ①
    - 直接相联映射：主存地址 = 区地址tag + 行索引index + 块内偏移offset
    - 主存容量=256MB，因此主存地址=28位（2^{28}=256M）；cache有8行，因此index=3位（2^3=8）
    - 每个cache行的数据块大小=64B，故offset=6位（2^6=64）；因此，tag=28-3-6=19位
    - 每个cache行包括：有效位（1位） + tag（19位） + 数据块大小（64B）=20+64x8=532位 =(532/8)字节 =66.5B；因此，数据cache总容量=cache行数 x 行大小=8x66.5B=532B

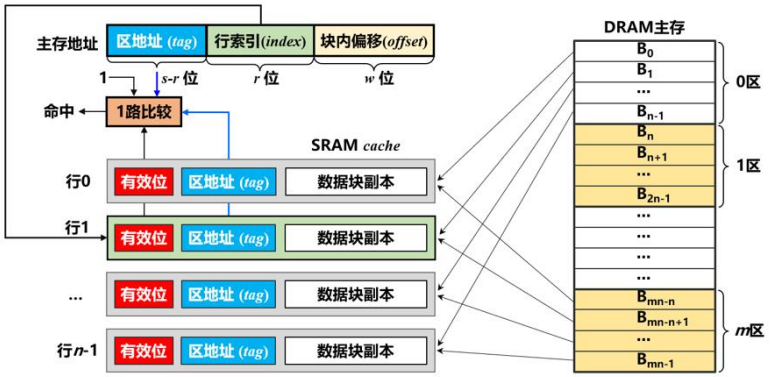


图4.38　直接相联映射的逻辑示意图

- 例4.4　主存地址空间大小为256MB，按字节编址。指令cache和数据cache分离，均有8行，每个cache行的数据块大小均为64B。数据cache采用直接相联映射方式。现有两个功能相同的程序A和B，其伪代码如下所示。假定int型数据为32位补码，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，首地址为320（十进制）。请回答如下问题，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的总容量是多少？
  ② 数组元素a[0][31]、a[1][1]所在主存块对应的cache行分别是多少？假设cache行号从零开始。
  ③ 程序A和B的数据访问命中率各是多少？那个程序的执行时间更短?
- 解（续）：
  - ②
    - 数组按行优先方式存放，首地址为320（十进制），每个数组元素占32位（int型数据，32位=4个字节），则二维数组a在内存中的分布如下：

内存地址	320	324	328	332	336	……	444	……	1340	1344	1348
行优先	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	……	a[0][31]	……	a[0][255]	a[1][0]	a[1][1]

- a[0][31]所在的内存地址=320 + 31x4 = 444；对应的主存块号=[444/64]=6
- a[1][1]所在的内存地址=320 + 256x4 + 4 = 1348；对应的主存块号=[1348/64]=21
  - [ ]表示取整

- 因为采用直接相联映射方式，主存块号与cache行号的对应关系为：
  - cache行号 = 主存块号 mod n
  - n=8（cache有8行）

- a[0][31]所在主存块对应的cache行是：6 mod 8 = 第6行
- a[1][1]所在主存块对应的cache行是：21 mod 8 = 第5行

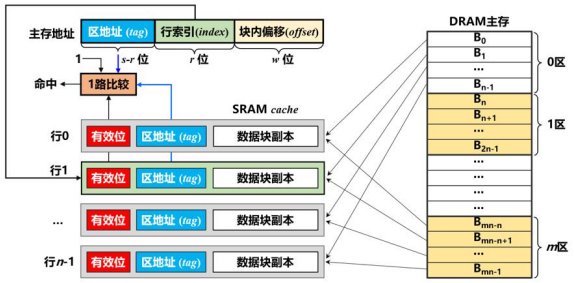


图4.38　直接相联映射的逻辑示意图

- 例4.4　主存地址空间大小为256MB，按字节编址。指令cache和数据cache分离，均有8行，每个cache行的数据块大小均为64B。数据cache采用直接相联映射方式。现有两个功能相同的程序A和B，其伪代码如下所示。假定int型数据为32位补码，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，首地址为320（十进制）。请回答如下问题，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的总容量是多少？
  ② 数组元素a[0][31]、a[1][1]所在主存块对应的cache行分别是多少？假设cache行号从零开始。
  ③ 程序A和B的数据访问命中率各是多少？那个程序的执行时间更短?
- 解（续）：
  - ③
    - 每个cache行的数据块大小为64B，其可以存放64B/32位=64x8/32 = 16个int型数据

程序A:
```
int a[256][256];
int sum_array1()
{
  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A中数据的访问顺序与存储顺序相同（均为行优先方式），第1次访问的数组元素是a[0][0]，缺失，需要载入数据块（载入16个int型数据：a[0][0]～a[0][15]），后续15次访问均命中；程序A的所有数据访问都符合这一规律；因此命中率=15/16=93.75%

程序B:
```
int a[256][256];
int sum_array2()
{
  int i,j,sum=0;
    for (j = 0; j < 256; j++)
        for (i = 0; i < 256; i++)
          sum += a[i][j];
    return sum;
}
```

- 程序B按照数组的列执行外层循环，其数据访问顺序与存储顺序不同（访问顺序为列优先方式，存储顺序为行优先方式）；第1次访问的数组元素是a[0][0]，缺失，需要载入数据块（载入16个int型数据：a[0][0]～a[0][15]）；第2次访问的数组元素是a[1][0]，仍然是缺失；程序B的所有数据访问都符合这一规律，均为缺失；因此命中率=0

- 因此，程序A比程序B执行时间更短（命中率高，访存速度快，执行时间短）

程序A

存储顺序（行优先）	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	……	a[0][31]	……	a[0][255]	a[1][0]	a[1][1]
访问顺序（行优先）	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	……	a[0][31]	……	a[0][256]	a[1][0]	a[1][1]

程序B

存储顺序（行优先）	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	……	a[0][31]	……	a[0][255]	a[1][0]	a[1][1]
访问顺序（列优先）	a[0][0]	a[1][0]	a[2][0]	a[3][0]	a[4][0]	……	a[31][0]	……	a[256][0]	a[0][1]	a[1][1]

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访问该行一次，计数器加1；当需要替换时，对所有可替换行的计数值进行比较，将计数值最小的行替换掉
  - LFU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访问该行一次，计数器加1；当需要替换时，对所有可替换行的计数值进行比较，将计数值最小的行替换掉
  - LFU算法的访问过程：

前3次分别访问数据块1、2、3，因为cache是空的，因此需要载入

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访问该行一次，计数器加1；当需要替换时，对所有可替换行的计数值进行比较，将计数值最小的行替换掉
  - LFU算法的访问过程：

第4次访问数据块2，cache中有数据块2，为命中，同时将数据块2的访问次数加1（2^1）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访问该行一次，计数器加1；当需要替换时，对所有可替换行的计数值进行比较，将计数值最小的行替换掉
  - LFU算法的访问过程：

第5次访问数据块1，cache中有数据块1，为命中，同时将数据块1的访问次数加1（1^1）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访问该行一次，计数器加1；当需要替换时，对所有可替换行的计数值进行比较，将计数值最小的行替换掉
  - LFU算法的访问过程：

第6次访问数据块3，cache中有数据块3，为命中，同时将数据块3的访问次数加1（3^1）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- **例4.5**：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访
    换时，对所有可替换行的计数值进行比较，将
  - LFU算法的访问过程：

第7次访问数据块1，cache中有数据块1，为命中，同时将数据块1的访问次数再加1（1^2）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访问该行一次，计数器加1；当需要替换时，对所有可替换行的计数值进行比较，将计数值最小的行替换掉
  - LFU算法的访问过程：

第8次访问数据块4，cache中没有，不命中，需要载入

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访问该行一次，计数器加1；当需要替换时，对所有可替换行的计数值进行比较，将[illegible]
  - LFU算法的访问过程：

第9次访问数据块4，cache中有数据块4，为命中，同时将数据块4的访问次数加1（4^1）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存
～7），共访问16次，访问次序为：1、2、3
、5、6、7、5。请分析LFU算法和LRU算法的

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访
换时，对所有可替换行的计数值进行比较，将
  - LFU算法的访问过程：

第10次要访问数据块5，cache中没有，需要载入，但是cache中已经满了，按照LFU算法替换计数值最小的那一行，第1行、第2行、第3行的计数值相同（2^1、3^1、4^1），都是最小，此时可采用随机替换算法或FIFO算法确定从3行中替换哪一行，这里采用FIFO算法，因此替换第1行（数据块2最先载入）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访
    换时，对所有可替换行的计数值进行比较，将
  - LFU算法的访问过程：

第11次要访问数据块6，cache中没有，需要载入，但是cache中已经满了，按照LFU算法替换计数值最小的那一行，第1行（5^0）的计数值最小，因此替换第1行

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访
    换时，对所有可替换行的计数值进行比较，将
  - LFU算法的访问过程：

第12次要访问数据块7，cache中没有，需要载入，但是cache中已经满了，按照LFU算法替换计数值最小的那一行，第1行（6^0）的计数值最小，因此替换第1行

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的

第13次要访问数据块5，cache中没有，需要载入，但是cache中已经满了，按照LFU算法替换计数值最小的那一行，第1行（7^0）的计数值最小，因此替换第1行

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访
    换时，对所有可替换行的计数值进行比较，将
  - LFU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的

第14次要访问数据块6，cache中没有，需要载入，但是cache中已经满了，按照LFU算法替换计数值最小的那一行，第1行（5^0）的计数值最小，因此替换第1行

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访
    换时，对所有可替换行的计数值进行比较，将
  - LFU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访
    换时，对所有可替换行的计数值进行比较，将
  - LFU算法的访问过程：

第15次要访问数据块7，cache中没有，需要载入，但是cache中已经满了，按照LFU算法替换计数值最小的那一行，第1行（6^0）的计数值最小，因此替换第1行

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的

第16次要访问数据块5，cache中没有，需要载入，但是cache中已经满了，按照LFU算法替换计数值最小的那一行，第1行（7^0）的计数值最小，因此替换第1行

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访
    换时，对所有可替换行的计数值进行比较，将
  - LFU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（1）
  - LFU算法：cache的每行设置一个计数器，每访问该行一次，计数器加1；当需要替换时，对所有可替换行的计数值进行比较，将计数值最小的行替换掉
  - LFU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^0	1^0	1^0	1^1	1^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2
cache 第1行		2^0	2^0	2^1	2^1	2^1	2^1	2^1	2^1	5^0	6^0	7^0	5^0	6^0	7^0	5^0
cache 第2行			3^0	3^0	3^0	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1	3^1
cache 第3行								4^0	4^1	4^1	4^1	4^1	4^1	4^1	4^1	4^1
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	替换	替换	替换	替换

  - LFU算法的命中率=5/16=31.25%

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设置一个计数器，当访问某cache行时，该cache行的计数器清0，其它cache行的计数器加1；因此计数器是用于统计未被访问的次数；当需要替换时，比较所有可替换行的计数值，将计数值最大的行替换掉

  - LRU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设置一个计数器，当访问某cache行时，该cache行的计数器清0，其它cache行的计数器加1；因此计数器是用于统计未被访问的次数；当需要替换时，比较所有可替换行的计数值，将计数值最大的行替换掉
  - LRU算法的访问过程：

第1次访问数据块1，因为cache中是空的，需要载入，并将数据块1的计数值清0（1^0）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设置一个计数器，当访问某cache行时，该cache行的计数器清0，其它cache行的计数器加1；因此计数器是用于统计未被访问的次数；当需要替换时，比较所有可
  - LRU算法的访问过程：

第2次访问数据块2，因为cache中是空的，需要载入，将数据块2的计数值清0（2^0），并将数据块1的计数值加1（1^1）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设置一个计数器，当访问某cache行时，该cache行的计数器清0，其它cache行的计数器加1；因此计数器是用于统计未被访问的次数；当需要替换时，比较所有可
  - LRU算法的访问过程：

第3次访问数据块3，因为cache中是空的，需要载入，将数据块3的计数值清0（3^0），并将数据块1、数据块2的计数值加1（1^2、2^1）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设置一个计数器，当访问某cache行时，该cache行的计数器清0，其它cache行的计数器加1；因此计数器是用于统计未被访问的次数；当需要替换时，比较所有可
  - LRU算法的访问过程：

第4次访问数据块2，cache中有，为命中，将数据块2的计数值清0（2^0），并将数据块1、数据块3的计数值加1（1^3、3^1）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设置一个计数器，当访问某cache行时，该cache行的计数器清0，其它cache行的计数器加1；因此计数器是用于统计未被访问的次数；当需要替换时，比较所有可
  - LRU算法的访问过程：

第5次访问数据块1，cache中有，为命中，将数据块1的计数值清0（1^0），并将数据块2、数据块3的计数值加1（2^1、3^2）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设置一个计数器，当访问某一行时，该行的计数器清0，其它cache行的计数器加1。需要替换时，比较所有可

第6次访问数据块3，cache中有，为命中，将数据块3的计数值清0（3^0），并将数据块1、数据块2的计数值加1（1^1、2^2）

  - LRU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设置一个计数器，当访问某行时，该行的计数器清0，其它cache行的计数器加1。需要替换时，比较所有可

第7次访问数据块1，cache中有，为命中，将数据块1的计数值清0（1^0），并将数据块2、数据块3的计数值加1（2^3、3^1）

  - LRU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设置一个计数器，当访问某一行时，该行的计数器清0，其它cache行的计数器加1。需要替换时，比较所有可

第8次访问数据块4，cache中没有，需要载入，将数据块4的计数值清0（4^0），并将数据块1、数据块2、数据块3的计数值加1（1^1、2^4、3^2）

  - LRU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设置一个计数器，当访问某行时，该行的计数器清0，其它cache行的计数器加1。需要替换时，比较所有可

第9次访问数据块4，cache中有，为命中，将数据块4的计数值清0（4^0），并将数据块1、数据块2、数据块3的计数值加1（1^2、2^5、3^3）

  - LRU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设
    器清0，其它cache行的计
    需要替换时，比较所有可

第10次访问数据块5，cache中没有，需要替换，根据LRU算法，替换4个数据块中近期访问最少的那个数据块（1^2、2^5、3^3、4^0），即数据块2（2^5），同时将数据块5的计数值清0（5^0），将数据块1、3、4的计数值加1（1^3、3^4、4^1）

  - LRU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- **例4.5**：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设
    器清0，其它cache行的计
    需要替换时，比较所有可

  - LRU算法的访问过程：

第11次访问数据块6，cache中没有，需要替换，根据LRU算法，替换4个数据块中近期访问最少的那个数据块（1^3、5^0、3^4、4^1），即数据块3（3^4），同时将数据块6的计数值清0（6^0），将数据块1、5、4的计数值加1（1^4、5^1、4^2）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- **例4.5**：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设
    器清0，其它cache行的计
    需要替换时，比较所有可

  - LRU算法的访问过程：

第12次访问数据块7，cache中没有，需要替换，根据LRU算法，替换4个数据块中近期访问最少的那个数据块（1^4、5^1、6^0、4^2），即数据块1（1^4），同时将数据块7的计数值清0（7^0），将数据块5、6、4的计数值加1（5^2、6^1、4^3）

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设
    器清0，其它cache行的计
    需要替换时，比较所有可

第13次访问数据块5，cache中有，为命中，同时将数据块5的计数值清0（5^0），将数据块7、6、4的计数值加1（7^1、6^2、4^4）

  - LRU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设
    器清0，其它cache行的计
    需要替换时，比较所有可

第14次访问数据块6，cache中有，为命中，同时将数据块6的计数值清0（6^0），将数据块7、5、4的计数值加1（7^2、5^1、4^5）

  - LRU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设
    器清0，其它cache行的计
    需要替换时，比较所有可

第15次访问数据块7，cache中有，为命中，同时将数据块7的计数值清0（7^0），将数据块5、6、4的计数值加1（5^2、6^1、4^6）

  - LRU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设
    器清0，其它cache行的计
    需要替换时，比较所有可

第16次访问数据块5，cache中有，为命中，同时将数据块5的计数值清0（5^0），将数据块7、6、4的计数值加1（7^1、6^2、4^7）

  - LRU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache 第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache 第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache 第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache 第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

- 例4.5：假设cache有4行，某程序要访问主存的7个数据块（数据块编号为：1～7），共访问16次，访问次序为：1、2、3、2、1、3、1、4、4、5、6、7、5、6、7、5。请分析LFU算法和LRU算法的访问过程，并计算命中率。

- 解：（2）
  - LRU算法： cache的每行设置一个计数器，当访问某cache行时，该cache行的计数器清0，其它cache行的计数器加1；因此计数器是用于统计未被访问的次数；当需要替换时，比较所有可替换行的计数值，将计数值最大的行替换掉

  - LRU算法的访问过程：

地址流（访问次序）	1	2	3	2	1	3	1	4	4	5	6	7	5	6	7	5
cache第0行	1^0	1^1	1^2	1^3	1^0	1^1	1^0	1^1	1^2	1^3	1^4	7^0	7^1	7^2	7^0	7^1
cache第1行		2^0	2^1	2^0	2^1	2^2	2^3	2^4	2^5	5^0	5^1	5^2	5^0	5^1	5^2	5^0
cache第2行			3^0	3^1	3^2	3^0	3^1	3^2	3^3	3^4	6^0	6^1	6^2	6^0	6^1	6^2
cache第3行								4^0	4^0	4^1	4^2	4^3	4^4	4^5	4^6	4^7
命中情况	载入	载入	载入	命中	命中	命中	命中	载入	命中	替换	替换	替换	命中	命中	命中	命中

  - LRU算法的命中率=9/16=56.25%；显然比LFU算法的命中率高（5/16=31.25%）

- 例4.6：假设主存容量=16MB，按字节寻址；虚拟存储器容量=4GB，采用页式虚拟存储器，页面大小为4KB。请回答：
- （1）计算物理页号、页内偏移、虚拟页号字段各为多少位？
- （2）计算页表中页表项的数量。
- （3）若部分页表内容如表4.10所示，求对应于虚拟地址00015240H和03FFF180H的物理地址。

表4.10 部分页表内容

虚页号	有效位	实页号	虚页号	有效位	实页号
00010H	0	002H	03FFFH	0	1
00015H	1	035H	……	……	……
……	……	……			

- 解：
- （1）
  - 主存容量=16MB，因此物理地址=24位（2^{24}=16M）
  - 虚拟存储器容量=4GB，因此虚拟地址=32位（2^{32}=4G）
  - 页面大小=4KB，因此页内偏移=12位（2^{12}=4K）
  - 物理页号=物理地址-页内偏移=24-12=12位
  - 虚拟页号=虚拟地址-页内偏移=32-12=20位

虚拟地址（32位）= 虚拟页号（20位）+ 页内偏移（12位）

物理地址（24位）= 物理页号（12位）+ 页内偏移（12位）

- 例4.6：假设主存容量=16MB，按字节寻址；虚拟存储器容量=4GB，采用页式虚拟存储器，页面大小为4KB。请回答：
- （1）计算物理页号、页内偏移、虚拟页号字段各为多少位？
- （2）计算页表中页表项的数量。
- （3）若部分页表内容如表4.10所示，求对应于虚拟地址00015240H和03FFF180H的物理地址。

表4.10 部分页表内容

虚页号	有效位	实页号	虚页号	有效位	实页号
00010H	0	002H	03FFFH	0	1
00015H	1	035H	……	……	……
……	……	……			

虚拟地址（32位）= 虚拟页号（20位）+ 页内偏移（12位）
物理地址（24位）= 物理页号（12位）+ 页内偏移（12位）

- 解（续）：
- （2）
  - 页表项的数量（即页表中有多少行）与虚拟页号的位数有关；虚拟页号=20位，则页表项的数量=2^{20}=1,048,576项，即页表有1,048,576行
- （3）
  - 虚拟地址=00015240H=00015H（虚拟页号）+ 240H（页内偏移）
  - 虚拟地址=03FFF180H=03FFFH（虚拟页号）+ 180H（页内偏移）
  - 查表4.10，虚页号=00015H，对应的实页号=035H，且有效位=1，因此虚拟地址00015240H对应的物理地址=035H（物理页号）+240H（页内偏移）=035240H
  - 查表4.10，虚页号=03FFFH，对应的有效位=0，表示该虚拟页不在主存中

- 例4.7：某计算机存储器按字节编址，虚拟（逻辑）地址空间大小为16MB，主存（物理）地址空间大小为1MB，页面大小为4KB，cache采用直接相联映射方式，共8行；主存与cache之间交换块大小为32B。系统运行到某一时刻时，页表的部分内容和cache的部分内容分别如下图4.56（a）、图4.56（b），图中页框号及标记字段的内容均为十六进制形式。请回答以下问题：
- （1）虚拟地址共有几位？哪几位表示虚页号？物理地址共有几位？哪几位表示页框号（物理页号）？
- （2）使用物理地址访问cache时，物理地址应划分为哪几个字段？请说明每个字段的位数及在物理地址中的位置。
- （3）虚拟地址001C60H所在的页面是否在主存中？若在主存中，则该虚拟地址对应的物理地址是什么？访问该地址时cache是否命中？请说明理由。
- （4）假设为该计算机配置一个四路组相联的TLB（可存放8个页表项），若其当前内容如图4.56（c）所示（十六进制），则此时虚拟地址024BACH所在的页面是否存在于主存中？请说明理由。

虚页号	有效位	页框号	……
0	1	06	……
1	1	04	……
2	1	15	……
3	1	02	……
4	0	……	……
5	1	2B	……
6	0	……	……
7	1	32	……

图4.56（a） 页表部分内容

行号	有效位	标记	……
0	1	020	……
1	0	……	……
2	1	01D	……
3	1	105	……
4	1	064	……
5	1	14D	……
6	0	……	……
7	1	27A	……

图4.56（b） cache部分内容

组号	有效位	标记	页框号	有效位	标记	页框号	有效位	标记	页框号	有效位	标记	页框号
0	0	__	__	1	001	15	0	__	__	1	012	1F
1	1	013	2D	0	__	__	1	008	7E	0	__	__

图4.56（c） TLB的部分内容

- 例4.7：某计算机存储器按字节编址，虚拟（逻辑）地址空间大小为16MB，主存（物理）地址空间大小为1MB，页面大小为4KB，cache采用直接相联映射方式，共8行；主存与cache之间交换块大小为32B。系统运行到某一时刻时，页表的部分内容和cache的部分内容分别如下图4.56（a）、图4.56（b），图中页框号及标记字段的内容均为十六进制形式。请回答以下问题：
- （1）虚拟地址共有几位？哪几位表示虚页号？物理地址共有几位？哪几位表示页框号（物理页号）？
- （2）使用物理地址访问cache时，物理地址应划分为哪几个字段？请说明每个字段的位数及在物理地址中的位置。
- （3）虚拟地址001C60H所在的页面是否在主存中？若在主存中，则该虚拟地址对应的物理地址是什么？访问该地址时cache是否命中？请说明理由。
- （4）假设为该计算机配置一个四路组相联的TLB（可存放8个页表项），若其当前内容如图4.56（c）所示（十六进制），则此时虚拟地址024BACH所在的页面是否存在于主存中？请说明理由。

- 解：（1）
  - 虚拟地址 = 虚页号 + 页偏移
  - 虚拟（逻辑）地址空间大小为16MB，因此，虚拟地址=24位（2^{24}=16M）
  - 页面大小为4KB，因此，页偏移=12位（2^{12}=4K）
  - 所以，虚页号=24-12=12位
  - 物理地址 = 实页号（页框号）+ 页偏移
  - 主存（物理）地址空间大小为1MB，因此，物理地址=20位（2^{20}=1M）
  - 所以，页框号（物理页号）=20-12=8位

- 例4.7：某计算机存储器按字节编址，虚拟（逻辑）地址空间大小为16MB，主存（物理）地址空间大小为1MB，页面大小为4KB，cache采用直接相联映射方式，共8行；主存与cache之间交换块大小为32B。系统运行到某一时刻时，页表的部分内容和cache的部分内容分别如下图4.56（a）、图4.56（b），图中页框号及标记字段的内容均为十六进制形式。请回答以下问题：
- （1）虚拟地址共有几位？哪几位表示虚页号？物理地址共有几位？哪几位表示页框号（物理页号）？
- （2）使用物理地址访问cache时，物理地址应划分为哪几个字段？请说明每个字段的位数及在物理地址中的位置。
- （3）虚拟地址001C60H所在的页面是否在主存中？若在主存中，则该虚拟地址对应的物理地址是什么？访问该地址时cache是否命中？请说明理由。
- （4）假设为该计算机配置一个四路组相联的TLB（可存放8个页表项），若其当前内容如图4.56（c）所示（十六进制），则此时虚拟地址024BACH所在的页面是否存在于主存中？请说明理由。

- 解：（2）
  - 因为cache采用直接相联映射方式，因此，物理地址（主存地址）=标记tag + 行索引字段index + 块内偏移offset
  - 主存与cache之间交换块大小为32B，因此，块内偏移=5位（2^5=32）
  - cache共有8行，因此，行索引字段=3位（2^3=8）
  - 主存（物理）地址空间大小为1MB，因此，物理地址=20位（2^{20}=1M）
  - 所以，标记tag=20-3-5=12位
  - 物理地址（主存地址）=标记tag + 行索引字段index + 块内偏移offset = 12位 + 3位 + 5位 =20位

- 例4.7：某计算机存储器按字节编址，虚拟（逻辑）地址空间大小为16MB，主存（物理）地址空间大小为1MB，页面大小为4KB，cache采用直接相联映射方式，共8行；主存与cache之间交换块大小为32B。系统运行到某一时刻时，页表的部分内容和cache的部分内容分别如下图4.56（a）、图4.56（b），图中页框号及标记字段的内容均为十六进制形式。请回答以下问题：
- （1）虚拟地址共有几位？哪几位表示虚页号？物理地址共有几位？哪几位表示页框号（物理页号）？
- （2）使用物理地址访问cache时，物理地址应划分为哪几个字段？请说明每个字段的位数及在物理地址中的位置。
- （3）虚拟地址001C60H所在的页面是否在主存中？若在主存中，则该虚拟地址对应的物理地址是什么？访问该地址时cache是否命中？请说明理由。
- （4）假设为该计算机配置一个四路组相联的TLB（可存放8个页表项），若其当前内容如图4.56（c）所示（十六进制），则此时虚拟地址024BACH所在的页面是否存在于主存中？请说明理由。

- 解：（3）
  - 虚拟地址=虚页号+页偏移 = 12位 + 12位；虚拟地址001C60H = 001H + C60H；虚页号=001=1，查页表（图4.56（a）），有效位=1，表示命中，即虚拟地址001C60H所在的页面在主存中
  - 由图4.56（a），该虚拟地址对应的页框号=04；因此，物理地址=实页号（页框号）+页偏移=04 + C60H = 04C60H
  - 又根据：物理地址（主存地址）=标记tag + 行索引字段index + 块内偏移offset=12位+3位+5位
  - 物理地址（主存地址）=04C60H = 0000 0100 1100 0110 0000 = 04C + 3 + 0；标记=04C，行索引字段=011=3
  - 根据行索引字段=3，查图4.56（b），有效位=1，但是标记=105，与主存地址的标记04C，不相等，表示cache不命中

虚页号	有效位	页框号	……
0	1	06	……
1	1	04	……
2	1	15	……
3	1	02	……
4	0	……	……
5	1	2B	……
6	0	……	……
7	1	32	……

图4.56（a） 页表部分内容

行号	有效位	标记	……
0	1	020	……
1	0	……	……
2	1	01D	……
3	1	105	……
4	1	064	……
5	1	14D	……
6	0	……	……
7	1	27A	……

图4.56（b） cache部分内容

- 例4.7：某计算机存储器按字节编址，虚拟（逻辑）地址空间大小为16MB，主存（物理）地址空间大小为1MB，页面大小为4KB，cache采用直接相联映射方式，共8行；主存与cache之间交换块大小为32B。系统运行到某一时刻时，页表的部分内容和cache的部分内容分别如下图4.56（a）、图4.56（b），图中页框号及标记字段的内容均为十六进制形式。请回答以下问题：
- （1）虚拟地址共有几位？哪几位表示虚页号？物理地址共有几位？哪几位表示页框号（物理页号）？
- （2）使用物理地址访问cache时，物理地址应划分为哪几个字段？请说明每个字段的位数及在物理地址中的位置。
- （3）虚拟地址001C60H所在的页面是否在主存中？若在主存中，则该虚拟地址对应的物理地址是什么？访问该地址时cache是否命中？请说明理由。
- （4）假设为该计算机配置一个四路组相联的TLB（可存放8个页表项），若其当前内容如图4.56（c）所示（十六进制），则此时虚拟地址024BACH所在的页面是否存在于主存中？请说明理由。

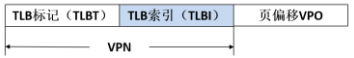


图4.51 组相联时虚拟地址划分

- 解：（4）
  - 组相联映射的TLB，虚拟地址（24位）=虚页号 + 页偏移 = TLB标记 + TLB索引 + 页偏移；见图4.51
  - 因为TLB只有8个页表项，因此，四路组相联映射的TLB有2组、每组4行，即TLB索引=1位（2^1=2）
  - 页偏移=12位，因此TLB标记=24-1-12=11位
  - 虚拟地址=024BACH= 0000 0010 0100 1011 1010 1100 = TLB标记 + TLB索引 + 页偏移 = 012H + 0 + BACH
  - 即TLB索引=0，表示是TLB的第0组；TLB标记=0000 0010 010=012H，查图4.56（c），有效位=1，表示页面在主存中；页框号=1FH，则主存地址（物理地址）= 实页号（页框号）+ 页偏移 = 1FH + BACH = 1FBACH

组号	有效位	标记	页框号	有效位	标记	页框号	有效位	标记	页框号	有效位	标记	页框号
0	0	__	__	1	001	15	0	__	__	1	012	1F
1	1	013	2D	0	__	__	1	008	7E	0	__	__

图4.56（c） TLB的部分内容

习题（P147–152）

- 4.1
- 4.2
- 4.3
- 4.6
- 4.8
- 4.11
- 4.13
- 4.14
- 4.17

习题答案（P147-152）

- **4.1** 解释下列名称
  - 存取时间：也称存储器的访问时间，指启动一次存储器操作（读或写）到该操作完成所经历的时间；读时间和写时间可能不同，例如DRAM读慢写快，闪存（U盘）读快写慢。
  - 存取周期：指连续启动两次访问操作之间的最短时间间隔；存取周期 = 存取时间 + 恢复时间（存储器状态的稳定恢复时间）；存取周期 > 存取时间。
  - 存储器带宽：指单位时间内存储器所能传输的信息量；存储器带宽的单位：位/秒，b/s；字节/秒，B/s；存储器带宽与存取时间的长短和一次传输的数据位的多少有关；存取时间越短、数据位越宽，存储器带宽越高。
  - 存储单元：也称为存储元，存放1位二进制数。
  - 边界对齐的数据存放：假设计算机的字长为32位（即1个字=32位=4字节），则边界对齐要求：双字数据（64位，8字节）起始字节地址最末3位为000（8的倍数）；单字数据（32位，4字节）起始字节地址最末2位为00（4的倍数）；半字数据（16位，2字节）起始字节地址最末1位为0（2的倍数）；单字节数据（8位，1字节）起始字节地址可以任意。
  - 大端存储：Big-Endian，存储器的低字节地址单元中存放的是数据的最高字节。采用大端方式更符合人的思维习惯。
  - 小端存储：Little-Endian，存储器的低字节地址单元中存放的是数据的最低字节。采用小端方式则有利于计算机处理。

- **4.1** 解释下列名称（续）
  - 静态存储器：存储体以静态MOS存储元为基本单元组成的存储器称为静态MOS存储器，也称为静态随机存取存储器（SRAM：Static Random Access Memory）；静态MOS存储器（SRAM）存储1位二进制数需要6个MOS管，其存储密度较低。
  - 动态存储器：存储体以动态MOS存储元为基本单元组成的存储器称为动态MOS存储器，也称为动态随机存取存储器（DRAM：Dynamic Random Access Memory）；动态MOS存储器（DRAM）则采用数量较少的MOS管（如4个MOS管、3个MOS管、1个MOS管）和存储电容来存储1位二进制数，其存储密度较高。
  - 刷新：因为DRAM电容上的电荷会逐渐泄露，数据只能保存较短的时间，为避免数据丢失，必须定期采用类似读操作的方式对存储单元补充电荷，这个过程称为刷新。
  - 刷新周期：信息存储到数据丢失之前的时间称为最大刷新周期；存储器实际完成两次完整刷新之间的时间间隔称为刷新周期，常见的刷新周期有2ms、4ms、8ms等。
  - 字扩展：字扩展又称容量扩展或地址总线扩展；例如，可以利用8个256Kx8位的SRAM芯片，构成1个2Mx8位的存储器。
  - 位扩展：字扩展又称容量扩展或地址总线扩展；例如，可以利用8个256Kx8位的SRAM芯片，构成1个2Mx8位的存储器。
  - 多体交叉存储器：包括高位多体交叉存储器和低位多体交叉存储器。高位多体交叉存储器的结构与存储器字扩展完全相同，也称为顺序编址模式；高位地址经过译码器用于选择不同的存储体，低位地址同时送各个存储体。低位多体交叉存储器的结构与存储器字扩展也基本相同，只是低位地址经过译码器用于选择不同的存储体，而高位地址同时送各个存储体；低位多体交叉也称为交叉编址模式。

- **4.1 解释下列名称（续）**
  - 高速缓冲存储器：SRAM的特点是速度快、容量小、价格高，DRAM的特点是速度慢、容量大、价格低，因此计算机的主存（内存，内存条）通常采用DRAM，而不是采用SRAM。为了提高CPU访问主存的速度，通常在CPU与主存之间增加一个隐藏的小容量快速SRAM，称为cache（高速缓冲存储器）。因为CPU执行的程序具有较强的局部性，可以将主存中经常访问或即将访问的数据的副本调度到cache中，使得大部分数据访问可以在cache中进行，从而提升系统的性能。
  - 双端口存储器：有两个相互独立的端口（左口、右口）。当左右两个端口的地址不同时，两个端口使用各自的地址线、数据线和控制线，此时可以同时对存储体的两个不同的存储单元进行访问（读写）。当左右两个端口的地址相同时，将发生冲突（因为存储体不允许同时对同一个存储单元进行读写操作）；此时由判断逻辑决定是哪一个端口先访问存储体，如左边的端口先访问存储体，然后将右边端口的/$Busy_R$置为低电平（/$Busy_R$=0，表示右边的端口不能访问存储体）；等左边的端口访问结束后，再将右边端口的/$Busy_R$置为高电平（/$Busy_R$=1，表示右边的端口可以访问存储体）。
  - 相联存储器：CAM，Content Addressable Memory；普通的存储器是按地址进行访问，而相联存储器则是按内容进行访问；相联存储器用于存放cache的查找表，其内部存储的基本数据是键值对（key，value）；相联存储器的输入不是地址，而是检索关键字key，输出是key对应的value值。
  - 时间局部性：是指当程序访问一个存储位置时，该位置在未来可能会被多次访问；如程序的循环结构和调用过程就很好地体现了时间局部性。
  - 地址映射：如何将主存地址空间映射到cache的地址空间？即主存的某一块将载入到cache的哪一块中？地址映射有3种方法：全相联映射（Full Associative Mapping），直接相联映射（Direct Mapping），组相联映射（Set Associative Mapping）。
  - 直接相联映射：主存的某一块只能映射到cache的固定块中。
  - 全相联映射：主存的某一块可以映射到cache的任意块中。
  - 组相联映射：主存的某一块只能映射到cache的固定组中，在该组中可以映射到任意块中。

- **4.1 解释下列名称（续）**
  - 命中率：如果CPU要访问的数据在cache中，则称为命中（Hit）。如果CPU要访问的数据不在cache中，则称为数据缺失（Miss，也称为不命中），此时要将缺失数据从主存调入cache中才能访问数据。假设N_c为命中cache的次数，N_m为从主存中访问信息的次数（即不命中的次数），则命中率h（Hit Ratio）：$h = N_c/(N_c+N_m)$。
  - 虚拟存储器：虚拟存储器处于“主存-辅存”存储层次，是为了解决存储系统“存储容量大”的问题；cache处于“CPU-主存”存储层次，是为了解决存储系统“访问速度快”的问题。有了虚拟存储器，程序员可以使用虚拟地址（虚拟地址空间比主存的物理地址空间要大很多）进行编程，这样程序员的编程不再受实际主存空间大小的限制。虚拟存储器也充分利用了程序的局部性，采用按需加载的方式加载程序代码和数据；根据程序局部性原理，通常程序只需要加载很小一部分空间即可运行，这种方式避免了将程序全部载入内存，大大提高了主存的利用率。
  - 页框号：虚拟地址（VA）= 虚拟页号（VPN）+ 虚拟页偏移（VPO）；物理地址（PA）= 物理页号（PPN）+ 物理页偏移（PPO）；虚拟页号也称为虚页号；物理页号也称为页框号、实页号。
  - 页表（慢表）：虚拟地址到物理地址的映射，本质上就是将虚拟页号（虚页号）转换为物理页号（实页号）；页式虚拟存储器中虚拟地址到物理地址之间的转换是基于页表进行的。页表是一张保存虚拟页号和物理页号对应关系的查找表，是一个由若干个表项组成的数组。页表采用虚拟页号作为索引进行访问，每一个表项主要包括：有效位和物理页号（PPN），另外还包括：修改位、使用位、权限位等信息。
  - 页表项：页表中的每一行称为页表项（PTE：Page Table Entry）。
  - TLB（快表）：为了进一步提高虚拟存储器地址转换的速度，现代计算机中都设置一个转换旁路缓冲区（Translation Look-aside Buffer，TLB），用于缓冲经常访问的页表项（PTE，即页表中经常会被访问的行）。TLB本质上就是一个容量较小的cache，为提高查找速度，大多采用全相联或组相联方式，且采用随机替换算法。TLB也称为快表，存放在主存中的页表称为慢表；快表是按内容访问的，慢表是按地址访问的。在进行地址转换时，往往同时查快表和慢表，如果查快表命中，则从快表中得到物理页号，同时终止查慢表的过程；如果查快表不命中，则从慢表中得到物理页号。

- **4.1 解释下列名称（续）**
  - LRU算法：近期最少使用：Least Recently Used，LRU算法将近期内最久未被访问过的cache行替换掉。cache的每行设置一个计数器，当访问某cache行时，该cache行的计数器清0，其它cache行的计数器加1；因此计数器是用于统计未被访问的次数。当需要替换时，比较所有可替换行的计数值，将计数值最大的行替换掉（即近期最久未被访问的行）。
  - LFU算法：最不经常使用：Least Frequently Used，LFU。LFU算法将访问次数最少的cache行（数据块）替换掉。cache的每行设置一个计数器，每访问该行一次，计数器加1；当需要替换时，对所有可替换行的计数值进行比较，将计数值最小的行替换掉（即最不经常使用的行）。
  - cache一致性：因为cache是主存一部分内容的副本，在写入cache时，需要保证cache与主存数据的一致性；即cache的内容改变了，主存相应的内容也要改变。cache的写入策略主要有：写回法、写穿法。
  - 写回法：采用写回法，当写入cache时，只修改cache的内容，并不立即修改主存的内容；只有当该cache行被替换出去时，才将脏数据（cache被修改后的数据称为脏数据）写回主存。写回法可以使cache在大部分时间（没有发生替换的时间）的写入速度等于访问cache的速度；只有在发生替换时，cache的写入速度才等于访问主存的速度（此时要执行写入主存的操作）。采用写回法时，cache的每一行必须配置一个修改位，也称为脏位（Dirty Bit）；若该行被修改，则脏位=1，否则为0；当该行被替换出去时，如果脏位=1，则需要将该行的内容写入主存。
  - 写穿法：写穿法也称为直写法，当写入cache时，同时将该数据块也写回主存。写穿法的优点：cache的每行不需要设置一个修改位（脏位），并且发生替换时，直接将该cache行丢弃，不需要写回主存。写穿法的缺点：cache的写入时间将是访问主存的时间，从而大大降低写入cache的速度。写穿法较好地维护了cache与主存的一致性。

- **4.2 选择题**
  - （1）A
    - I：正确
    - II：正确
    - III：错误
    - IV：错误
    - 因此仅I和II

（1）[2010] 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Ⅳ　　D. 仅Ⅱ、Ⅲ和Ⅳ

（2）[2014] 某容量为 256MB 的存储器由若干 4M×8 位的 DRAM 芯片构成，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

  - （2）A
    - DRAM芯片容量=4Mx8位，地址线=22根，因为DRAM是采用地址复用技术，只需要一半的地址线=11根；数据线=8根，总数=11+8=19根

（3）[2009] 某计算机主存容量为 64KB，其中 ROM 区为 4KB，其余为 RAM 区，按字节编址。要用 2K×8 位的 ROM 芯片和 4K×4 位的 RAM 芯片来设计该存储器，则需要上述规格的 ROM 芯片数和 RAM 芯片数分别是_____。

A. 1、15　　B. 2、15　　C. 1、30　　D. 2、30

  - （3）D
    - ROM容量=4KB，RAM容量=64KB-4KB=60KB；ROM芯片的数量=4KB/(2Kx8位)=2片，RAM芯片的数量=60KB/(4Kx4位)=30片

- **4.2 选择题**

（4）[2010] 假定用若干个 2K×4 位的芯片组成一个 8K×8 位的存储器，则地址 0B1FH 所在芯片的最小地址是_____。

A. 0000H　　B. 0600H　　C. 0700H　　D. 0800H

- （4）D
  - 2K的地址范围：0000H～07FFH
  - 8K的地址范围：0000H～07FFH， 0800H～0FFFH， 1000H～17FFH， 1800H～1FFFH
  - 0B1FH在0800H～0FFFH范围内，其最小地址是0800H

（5）[2018] 假定 DRAM 芯片中存储阵列的行数为 r、列数为 c，对于一个 2K×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

- （5）C
  - A：行地址=11根（r=2048=2^{11}），列地址=0根（c=1=2^0），DRAM的地址线=max(行地址，列地址)=11根
  - B：行地址=6根（r=64==2^6），列地址=5根（c=32=2^5），DRAM的地址线=max(行地址，列地址)=6根
  - C：行地址=5根（r=32=2^5），列地址=6根（c=64=2^6），DRAM的地址线=max(行地址，列地址)=6根
  - D：行地址=0根（r=1=2^0），列地址=11根（c=2048=2^{11}），DRAM的地址线=max(行地址，列地址)=11根
  - B、C的地址线相同，但是C的行数少，因此C的刷新开销小（r=32、c=64）

（6）[2019] 假定一台计算机采用 3 通道存储器总线，配套的内存条型号为 DDR3-1333，即内存条接插的存储器总线的工作频率为 1333MHz、总线宽度为 64 位，则存储器总线的总带宽大约是_____。

A. 10.66GB/s　　B. 32GB/s　　C. 64GB/s　　D. 96GB/s

- （6）B
  - 存储器总线的带宽=总线宽度/时钟周期=总线宽度x工作频率=64位x1333MHz=10.664GB/s
  - 因为是3通道，因此存储器总线的带宽=3x10.664=31.992GB/s≈32GB/s

- **4.2 选择题（续）**

（7）[2015] 某计算机使用 4 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

– （7）D
  - 存储体0的地址为：8000、8004、8008；存储体1的地址为：8001、8005；存储体2的地址为：8002、8006；存储体3的地址为：8003、8007
  - 主存地址序列为：8005、8006、8007、8008、8001、8002、8003、8004、8000
  - A：8004和8008虽然都在存储体0，但是相差4个序列，因此不会冲突
  - B：8002和8007分别在存储体2和存储体3中，因此不会冲突
  - C：8001和8008分别在存储体1和存储体0中，因此不会冲突
  - D：8000和8004都在存储体0，并且是先后访问，因此会发生冲突

（8）[2015] 下列存储器中，在工作期间需要周期性刷新的是_____。

A. SRAM　　B. SDRAM　　C. ROM　　D. FLASH

– （8）B
  - SRAM：不需要刷新；SDRAM：需要刷新；ROM：不需要刷新；FLASH：不需要刷新

（9）[2011] 下列各类存储器中，不采用随机存取方式的是_____。

A. EPROM　　B. CDROM　　C. DRAM　　D. SRAM

– （9）B
  - EPROM、DRAM、SRAM都属于随机存取方式的存储器；CDROM是光盘，不属于随机存取方式的存储器

- **4.2 选择题（续）**

（10）[2012] 下列关于闪存（Flash Memory）的叙述中，错误的是_____。

A. 信息可读可写，并且读、写速度一样快

B. 存储元由 MOS 管组成，是一种半导体存储器

C. 掉电后信息不丢失，是一种非易失性存储器

D. 采用随机访问方式，可替代计算机外部存储器

– （10）A
  - A：错误，Flash Memory的读、写速度不一样（写入时必须先擦除原有数据，写速度比读速度要慢）
  - B：正确
  - C：正确
  - D：正确

	NAND Flash	NOR Flash
芯片容量	<32GBit	<1GBit
访问方式	顺序读写	随机读写
接口方式	任意I/O口	特定完整存储器接口
读写性能	读取快（顺序读） 写入快 擦除快（可按块擦除）	读取快（RAM方式） 写入慢 写入慢
使用寿命	百万次	十万次
价格	低廉	高昂

- **4.2 选择题（续）**

（11）[2017] 某 C 语言程序段如下：

```
for(i=0; i<=9; i++)
{
  temp=1;
  for(j=0; j<=i;
     j++)temp * =a[j];
  sum + =temp;
}
```

下列关于数组 a 的访问局部性的描述中，正确的是_____。

A. 时间局部性和空间局部性皆有　　B. 无时间局部性，有空间局部性

C. 有时间局部性，无空间局部性　　D. 时间局部性和空间局部性皆无

– （11）A
  - 数组a中通常包含若干个数据，这些数据在内存中是顺序存放的，因此具有空间局部性；数组a通常在程序的循环体中被执行若干次，因此还具有时间局部性

（12）[2009] 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 32B，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. 1　　C. 4　　D. 6

– （12）C
  - cache共有16块，二路组相联，因此cache有16/2=8组；主存块大小=32B，因此块内偏移=5位
  - $129_{十进制}$号主存单元的地址=1000 0001$_{二进制}$=100 00001$_{二进制}$，其对应的主存块号=$100_{二进制}$=4，将载入cache的第4组

- **4.2 选择题（续）**

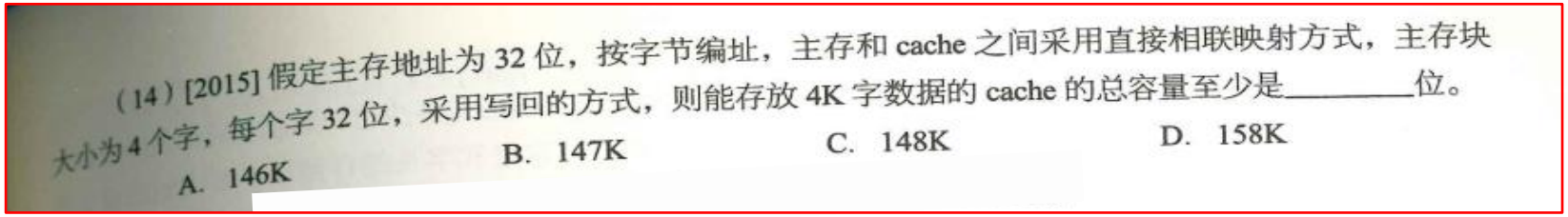
（13）[2012] 假设某计算机按字编址，cache 有 4 行，cache 和主存之间交换的块大小为 1 个字。若 cache 的内容初始为空，采用二路组相联映射方式和 LRU 替换策略。访问的主存地址依次为 0、4、8、2、0、6、8、6、4、8 时，命中 cache 的次数是_____。

A. 1　　B. 2　　C. 3　　D. 4

- （13）A
  - cache有4行，二路组相联，因此cache有4/2=2组
  - cache与主存之间交换的块大小为1个字，计算机按字编址，主存地址即为主存的块号
  - 命中次数=1

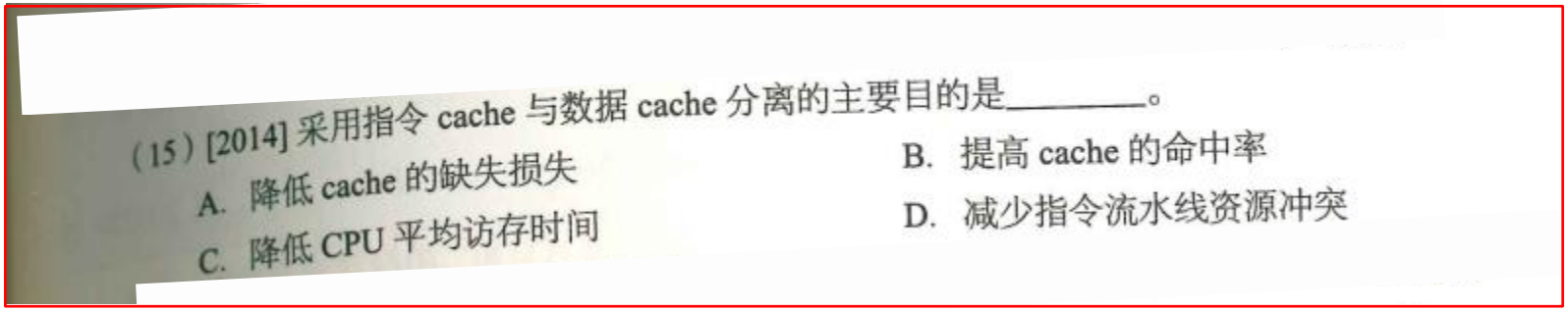
主存块号	0	4	8	2	0	6	8	6	4	8
对应cache组号	0	0	0	0	0	0	0	0	0	0
对应cache行号	0或1	0或1	0或1	0或1	0或1	0或1	0或1	0或1	0或1	0或1
cache 第0组、第0行	0^0	0^1	8^0	8^1	0^0	0^1	8^0	8^1	4^0	4^1
cache 第0组、第1行		4^0	4^1	2^0	2^1	6^0	6^1	6^0	6^1	8^0
cache 第1组、第0行										
cache 第1组、第1行										
命中情况	载入	载入	替换	替换	替换	替换	替换	命中	替换	替换

- **4.2 选择题（续）**


（14）[2015] 假定主存地址为 32 位，按字节编址，主存和 cache 之间采用直接相联映射方式，主存块大小为 4 个字，每个字 32 位，采用写回的方式，则能存放 4K 字数据的 cache 的总容量至少是_____位。

A. 146K　　B. 147K　　C. 148K　　D. 158K

– （14）C
  - 主存地址=区地址+行索引+块内偏移=32位
  - cache的数据容量=4K字，块大小=4字，因此cache有1K=1024行，行索引=10位
  - 1个块=4个字=4x32位=16字节，块内偏移=4位
  - 因此区地址=32-10-4=18位
  - 因为采用写回的方式，因此cache每一行包括：主存数据块副本、区地址、有效位、脏位，即：4x32位+18位+1位+1位=148位
  - 因此cache总容量=1024行x148位=148K位

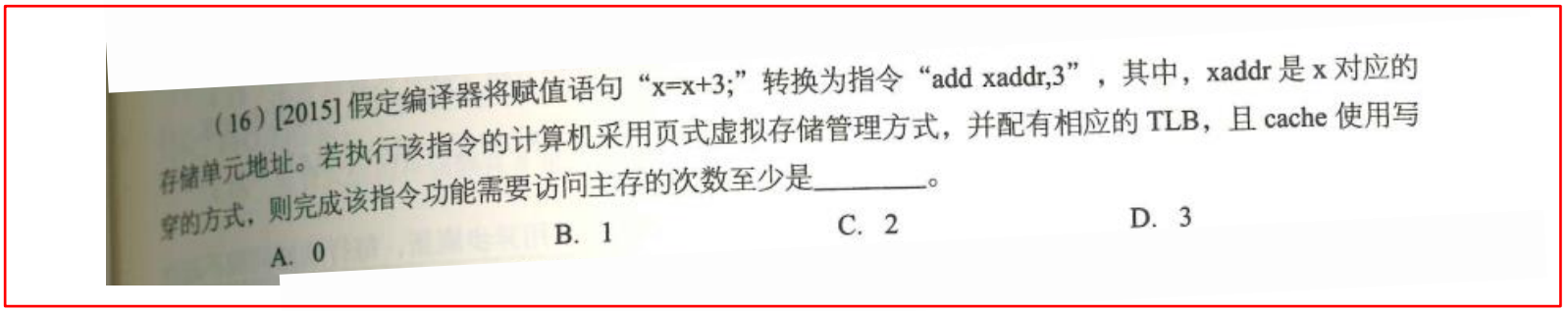

（15）[2014] 采用指令 cache 与数据 cache 分离的主要目的是_____。

A. 降低 cache 的缺失损失　　B. 提高 cache 的命中率

C. 降低 CPU 平均访存时间　　D. 减少指令流水线资源冲突

– （15）D
  - 指令cache和数据cache分离，CPU可以同时访问指令cache（取指令）和数据cache（取数据或写回数据），从而减少指令流水线 的资源冲突

- **4.2 选择题（续）**

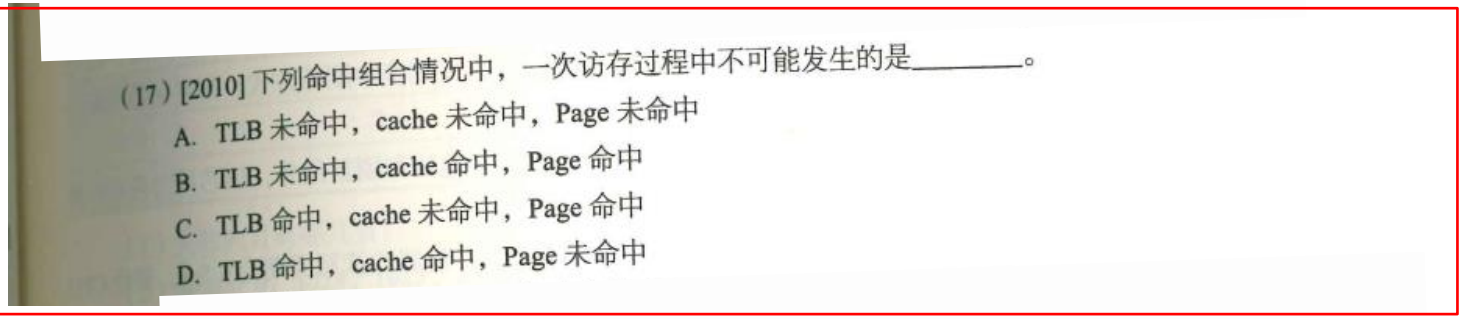


（16）[2015] 假定编译器将赋值语句“x=x+3;”转换为指令“add xaddr,3”，其中，xaddr 是 x 对应的存储单元地址。若执行该指令的计算机采用页式虚拟存储管理方式，并配有相应的 TLB，且 cache 使用写穿的方式，则完成该指令功能需要访问主存的次数至少是_____。

A. 0　　B. 1　　C. 2　　D. 3

  - （16）B
    - 当采用页式虚拟存储器时，并配有TLB，最短的路径是TLB命中、cache命中，此时虽然不需要访问存储器，但是因为cache使用写穿的方式，“add xaddr, 3”指令需要完成写cache操作（x+3 -> x），同时要执行写存储器的操作（写穿方式），因此访问主存的次数至少是1次

- **4.2 选择题（续）**


（17）[2010] 下列命中组合情况中，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

– （17）D
  - A：第5种情况
  - B：第3种情况
  - C：第2种情况
  - D：第7种情况，不可能

序号	TLB	页	cache	可能性	说明
1	命中	命中	命中	可能	TLB命中则页一定命中，页载入主存和数据块载入cache并不同步，所以cache有可能命中，也有可能缺失
2	命中	命中	缺失		
3	缺失	命中	命中	可能	TLB缺失后还可以访问慢速页表（慢表），页载入主存和数据块载入cache并不同步，所以cache有可能命中，也有可能缺失
4	缺失	命中	缺失		
5	缺失	缺失	缺失	可能	这是最糟糕的情况，虚存系统初始化时常见
6	命中	缺失	缺失	不可能	页缺失说明页不在主存中，TLB中一定没有对应页表项，TLB不可能命中
7	命中	缺失	命中		
8	缺失	缺失	命中	不可能	页缺失说明数据也不在主存中，所以cache不可能命中

- **4.2 选择题（续）**

（18）[2013] 某计算机主存地址空间大小为 256MB，按字节编址。虚拟地址空间大小为 4GB，采用页式存储管理方式，页面大小为 4KB，TLB（快表）采用全相联映射，有 4 个页表项，内容如表 4.12 所示。

表 4.12 4 个页表项的内容

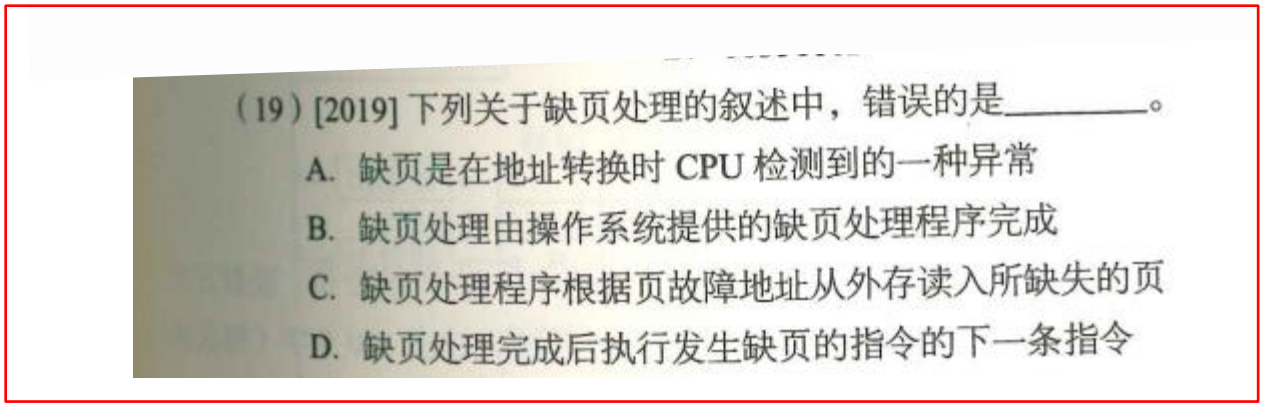
有效位	标记	页框号	…
0	FF180H	0002H	…
1	3FFF1H	0035H	…
0	02FF3H	0351H	…
1	03FFFH	0153H	…

则对虚拟地址 03FFF180H 进行虚实地址转换的结果是_____。

A. 0153180H　　B. 0035180H　　C. TLB 缺失　　D. 缺页

- （18）A
  - 主存地址空间大小=256MB，主存地址=28位
  - 虚拟地址空间大小=4GB，虚拟地址=32位
  - 页面大小=4KB，页偏移=12位
  - 物理地址 = 实页号（页框号）+ 页偏移，因此：页框号=28-12=16位
  - 虚拟地址 = 虚页号 + 页偏移，因此：虚页号=32-12=20位
  - 虚拟地址03FFF180H=虚页号 + 页偏移=03FFFH + 180H
  - 虚页号=03FFFH，查页表，得到页框号=0153H，有效位=1，表示命中，对应的主存地址= 0153H + 180H = 0153180H

- **4.2 选择题（续）**


（19）[2019] 下列关于缺页处理的叙述中，错误的是_____。

A. 缺页是在地址转换时 CPU 检测到的一种异常

B. 缺页处理由操作系统提供的缺页处理程序完成

C. 缺页处理程序根据页故障地址从外存读入所缺失的页

D. 缺页处理完成后执行发生缺页的指令的下一条指令

  - （19）D
    - A：正确
    - B：正确
    - C：正确
    - D：错误：缺页处理程序返回到原来的进程，驱使引起缺页的指令重新启动

- **4.3** 简答题
  - （1）计算机系统中采用层次化存储体系结构的目的是什么？
  - 答：
    - 存储系统层次结构利用程序局部性的原理，从系统级角度将速度、容量、成本各异的存储器有机组合在一起，全方位优化存储系统的各项性能指标；上层存储器可以为下层存储器做缓冲，将经常使用数据的副本调度到上层，这样CPU只需要访问上层快速的小容量存储器，即可获得大部分数据。
  - （2）为什么在存储器芯片中设置片选输入端？
  - 答：
    - 单片存储芯片的容量有限，通常需要将多片存储芯片按照一定的方式组织起来，并与CPU连接，从而获得一个大容量的存储器。通过设置片选端，可以由地址线经译码电路得到片选信号，从而选择不同的存储器芯片。
  - （3）动态MOS存储器为什么要刷新？如何刷新？
  - 答：
    - 因为DRAM电容上的电荷会逐渐泄露，数据只能保存较短的时间，为避免数据丢失，必须定期采用类似读操作的方式对存储单元补充电荷，这个过程称为刷新。
    - 动态MOS存储器采用类似读操作的方式进行刷新。DRAM的刷新是按行进行的，每次完成1行的刷新，刷新操作只需要给出存储器的行地址，不需要给出存储器的列地址。常见的刷新方式：集中刷新方式；分散刷新方式；异步刷新方式。
  - （4）试述多体交叉存储器的设计思想和实现方法。
  - 答：
    - 高位多体交叉存储器的结构与存储器字扩展完全相同，也称为顺序编址模式；高位地址经过译码器用于选择不同的存储体，低位地址同时送各个存储体。
    - 低位多体交叉存储器的结构与存储器字扩展也基本相同，只是低位地址经过译码器用于选择不同的存储体，而高位地址同时送各个存储体；低位多体交叉也称为交叉编址模式。低位多体交叉存储器的优点：由于程序具有局部性和连续性的特点，如果将程序的指令和数据分布在不同的存储体中，可以实现多个存储体并行工作。
  - （5）为什么说cache对程序员是透明的？
  - 答：
    - cache的数据查找、地址映射、替换策略、写入策略都是由cache控制器实现的，不需要操作系统干预。对程序员来说，在编写程序时，是按照主存的地址进行访存的，感觉不到计算机中有cache存在。因此cache对程序员来说是透明的。

- **4.3 简答题（续）**
  - （6）直接相联映射方式下，为什么不需要使用替换算法？
  - 答：
    - 直接相联映射方式，主存的某一块将映射到cache的固定块；当cache满时，只能替换掉该主存块对应的cache块，因此不需要替换算法。
  - （7）为什么要考虑cache的一致性？
  - 答：
    - 因为cache是主存一部分内容的副本，在写入cache时，需要保证cache与主存数据的一致性；即cache的内容改变了，主存相应的内容也要改变。cache的写入策略主要有：写回法、写穿法。采用写回法，当写入cache时，只修改cache的内容，并不立即修改主存的内容；只有当该cache行被替换出去时，才将脏数据（cache被修改后的数据称为脏数据）写回主存。写穿法也称为直写法，当写入cache时，同时将该数据块也写回主存。写穿法的优点：cache的每行不需要设置一个修改位（脏位），并且发生替换时，直接将该cache行丢弃，不需要写回主存。操作的速度。
  - （8）替换算法有哪几种？它们各有何优点？
  - 答：
    - 常用的替换算法有4种：先进先出算法（FIFO），最不经常使用算法（LFU），近期最少使用算法（LRU），随机替换算法。
    - FIFO算法按照数据块进入cache的先后决定替换的顺序，在需要进行替换时，选择最先被载入cache的行（数据块）进行替换。FIFO算法的优点：系统开销较小。FIFO算法的缺点：没有考虑程序访问的局部性，可能会把一些需要经常使用的块（如循环程序块）也作为最早进入cache的块而替换掉，从而导致cache的命中率不高。
    - LFU算法将访问次数最少的cache行（数据块）替换掉。LFU算法的不足：计数器记录的是cache工作后的历史访问统计情况，并不能严格反映近期访问情况。
    - LRU算法的优点：保护了刚载入cache的新数据，符合cache的工作原理（程序局部性），使cache具有较高的命中率；LRU算法的缺点：需要快速比较多个cache行的计数器值，硬件实现比较复杂。
    - 随机替换就是在需要进行替换时，从所有可替换的行中，随机选取一行进行替换。优点是实现最容易，而且替换速度也比前面三种算法快；缺点是随机替换的数据，很可能马上又要使用，从而降低cache的命中率和工作效率。

- **4.4** 对于32KB容量的存储器，若按16位字编址，其地址寄存器应是多少位？数据寄存器是多少位？

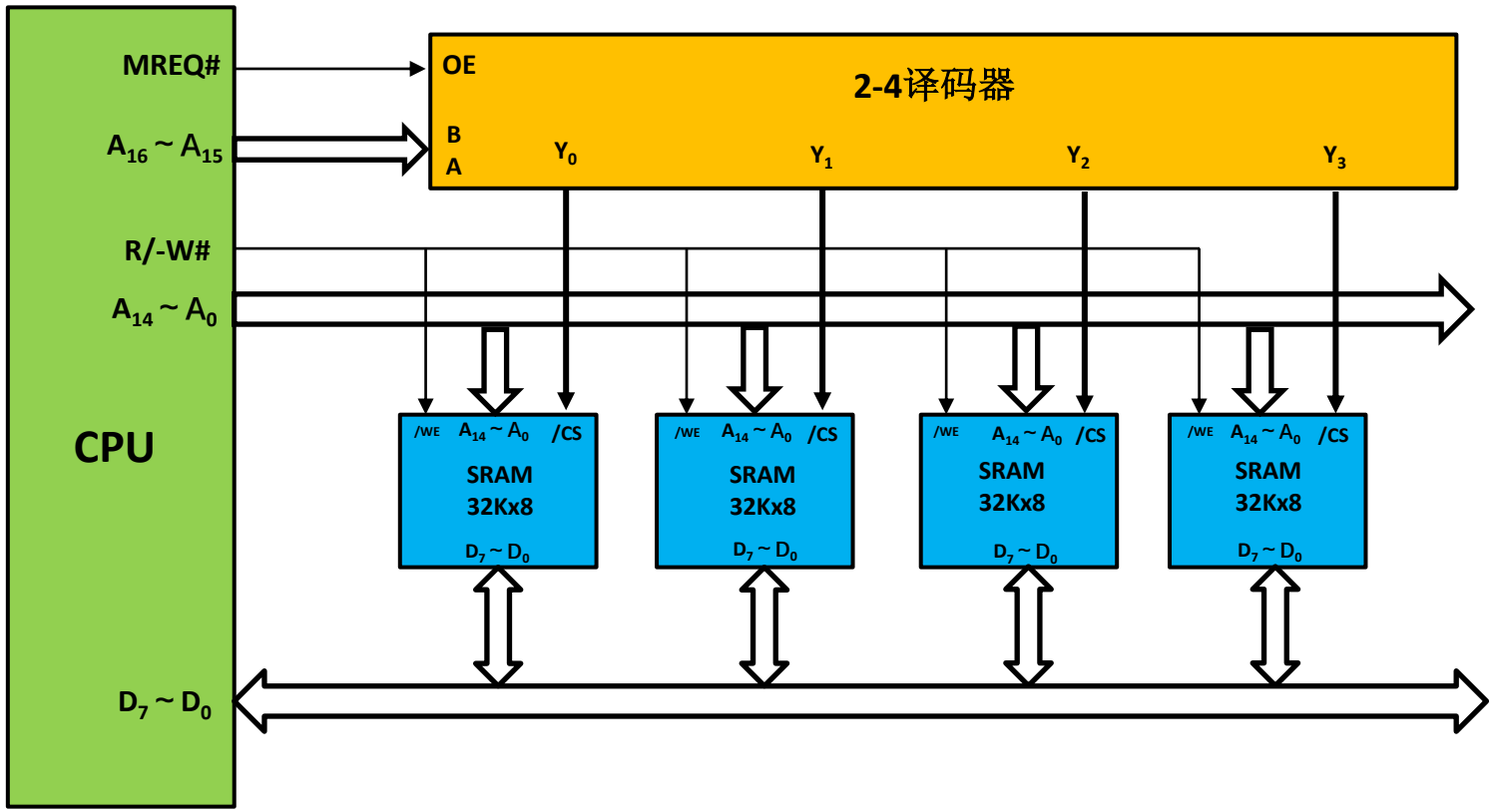
- 解：
  - 32KB/16位=16K=2^{14}
  - 地址寄存器=14位
  - 数据寄存器=16位

- **4.5（例题）　用4个32Kx8位SRAM存储芯片，可设计出哪几种不同容量和字长的存储器？画出相应设计图，并完成与CPU的连接。**

- **解：**
  - **（1）字扩展：32K -> 128K**
    - **4片32Kx8位 -> 128Kx8位**
  - **（2）位扩展：8位 -> 32位**
    - **4片32Kx8位 -> 32Kx32位**
  - **（3）字位扩展：32K -> 64K，8位 -> 16位**
    - **4片32Kx8位 -> 64Kx16位**

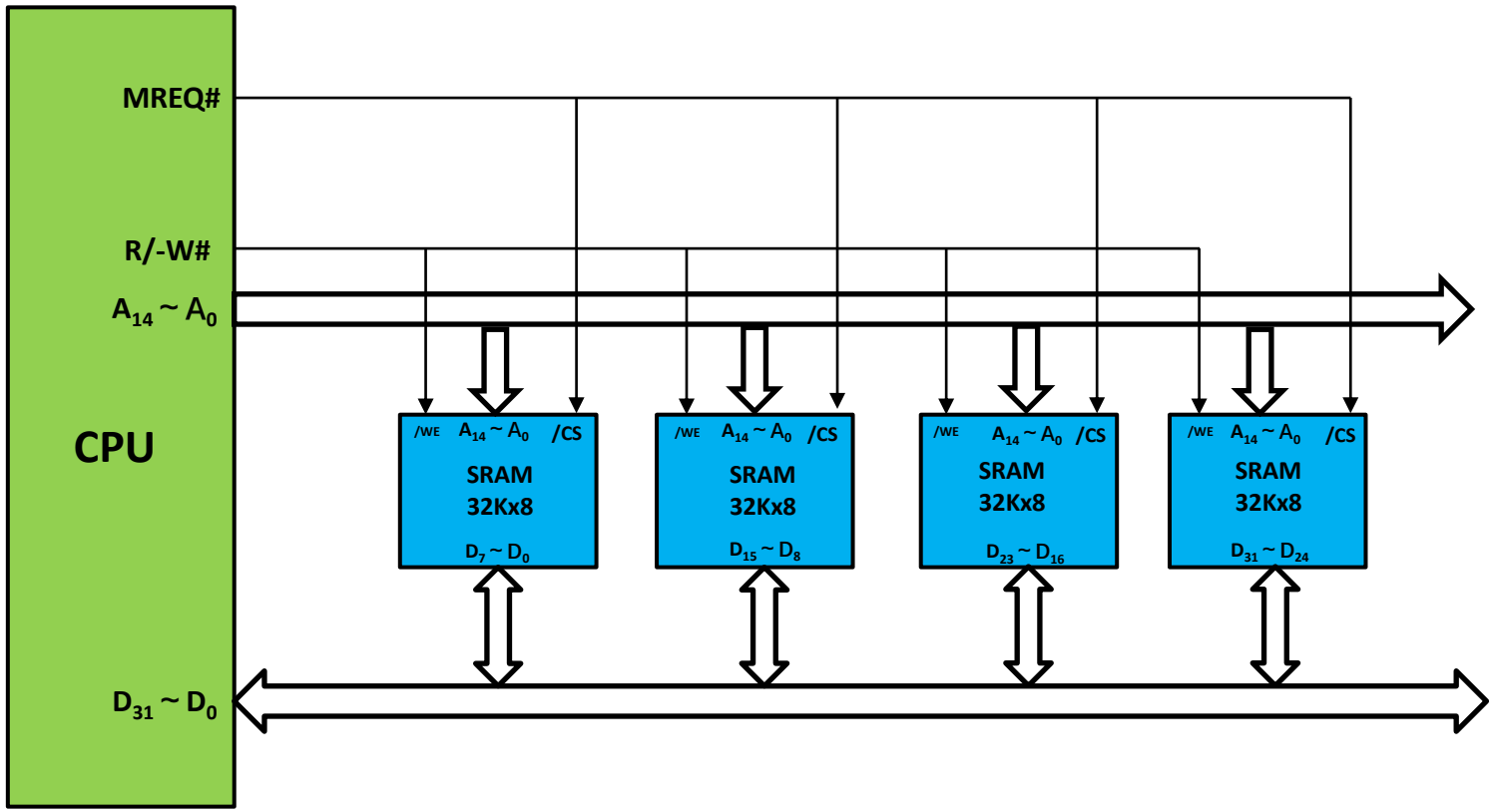
（1）字扩展：32K -> 128K

4片32Kx8位 -> 128Kx8位



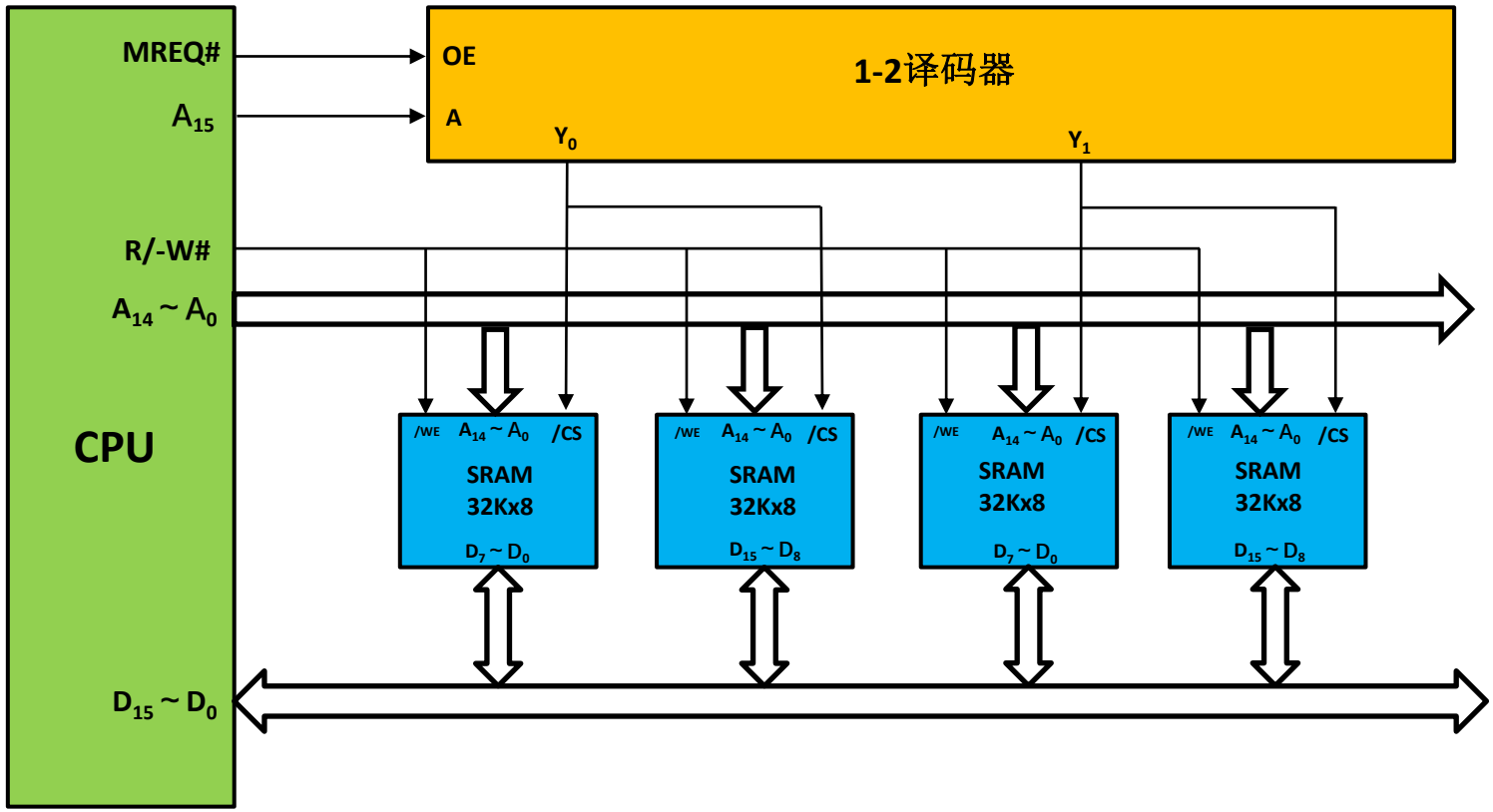
（2）位扩展：8位 -> 32位

4片32Kx8位 -> 32Kx32位



（3）字位扩展：32K -> 64K，8位 -> 16位

4片32Kx8位 -> 64Kx16位

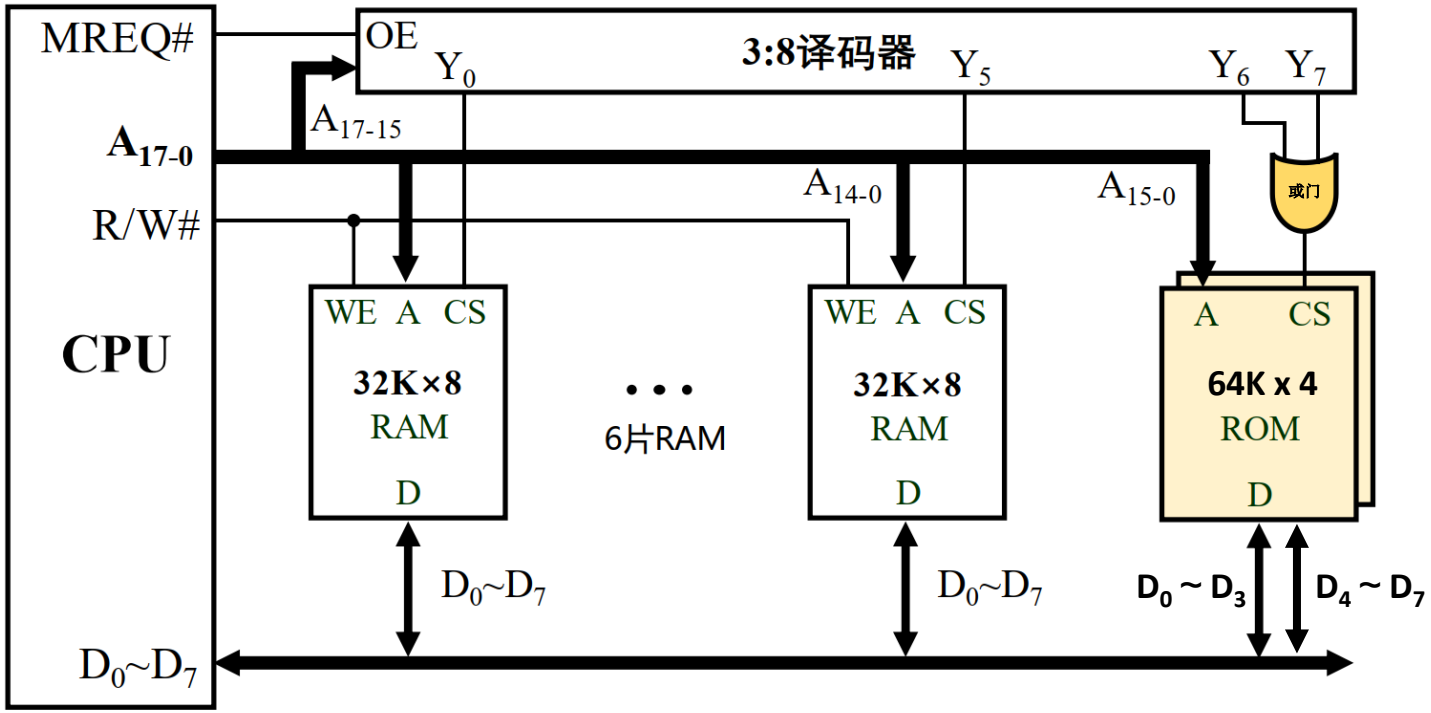


- 4.6　用32Kx8位RAM芯片和64Kx4位ROM芯片，设计256Kx8位存储器。其中，从30000H到3FFFFH的地址空间为只读存储区，其他为可读、可写存储区。完成存储器与CPU的连接。
- 解：
  - 从30000H到3FFFFH的地址空间为只读存储器；因此，ROM的容量=64KB，需要2片64Kx4位ROM芯片
  - RAM的容量=256KB-64KB=192KB，需要192KB/32KB=6片32Kx8位RAM芯片
  - RAM采用位扩展，ROM采用字扩展
  - 32Kx8位的RAM芯片，地址线=15根（A_{14}～A_0），数据线=8根（D_7～D_0）
  - 64Kx4位的ROM芯片，地址线=16根（A_{15}～A_0），数据线=4根（D_3～D_0）
  - 256KB的地址空间为：00000H～3FFFFH，其中00000H～2FFFFFH为RAM空间，30000H～3FFFFH为ROM空间
  - 00 0000 0000 0000 0000 ～ 00 0111 1111 1111 1111　接第1片RAM
  - 00 1000 0000 0000 0000 ～ 00 1111 1111 1111 1111　接第2片RAM
  - 01 0000 0000 0000 0000 ～ 01 0111 1111 1111 1111　接第3片RAM
  - 01 1000 0000 0000 0000 ～ 01 1111 1111 1111 1111　接第4片RAM
  - 10 0000 0000 0000 0000 ～ 10 0111 1111 1111 1111　接第5片RAM
  - 10 1000 0000 0000 0000 ～ 10 1111 1111 1111 1111　接第6片RAM
  - 11 0000 0000 0000 0000 ～ 11 1111 1111 1111 1111　接2片ROM芯片

– 选用1个3-8译码器，将地址线的高3位（A_{17}～A_{15}）接3-8译码器；译码器的输出为：Y_7～Y_0

– 其中Y_5～Y_0分别接6个RAM芯片， Y_7～Y_6通过一个或门同时接2个ROM芯片

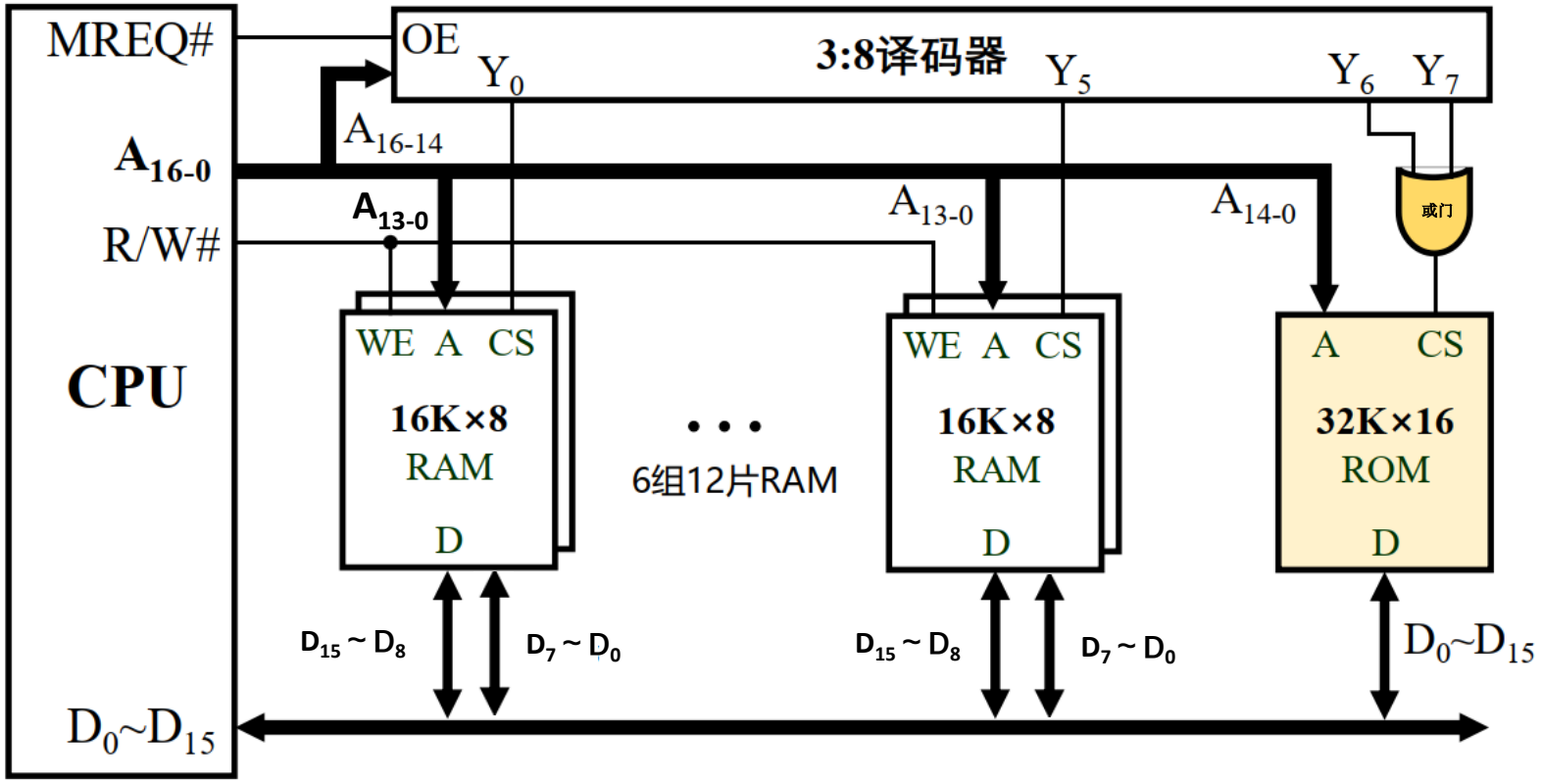
– 地址线的低15位同时接6个RAM芯片，低16位同时接2个ROM芯片；8位数据线同时接6个RAM芯片，数据线的高4位、低4位分别接2个ROM芯片



- **4.7（例题）　某计算机字长为16位，主存容量为128Kx16位，请用16Kx8位SRAM芯片和32Kx16位ROM芯片，为该计算机设计一个主存储器。要求18000H到1FFFFH为ROM区，其余为RAM区。画出存储器与CPU的连接。**

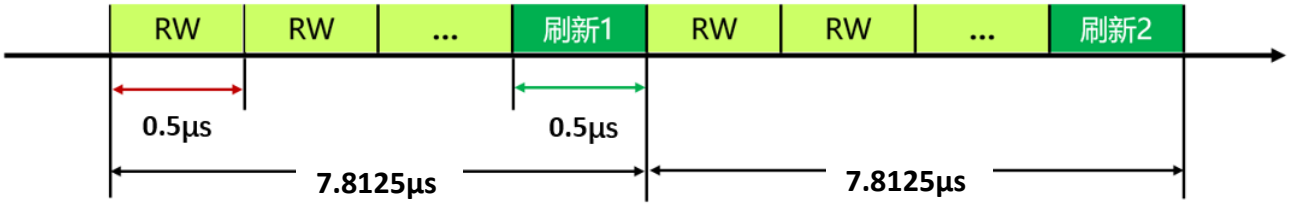
- **解：**
  - **18000H到1FFFFH为ROM区；因此，ROM的容量=32Kx16位，需要1片32Kx16位ROM芯片**
  - **RAM区的容量=128Kx16位 - 32Kx16位=96Kx16位，需要(96Kx16)/(16Kx8)=12片16Kx8位RAM芯片**
  - **RAM采用字位同时扩展，16K -> 96K，8位 -> 16位**
  - **32Kx16位的ROM芯片，地址线=15根（A_{14}～A_0），数据线=16根（D_{15}～D_0）**
  - **16Kx8位的RAM芯片，地址线=14根（A_{13}～A_0），数据线=8根（D_7～D_0）**
  - **128Kx16位的地址空间为：00000H～1FFFFH，其中00000H～17FFFFH为RAM空间，18000H～1FFFFH为ROM空间**
  - **0 0000 0000 0000 0000 ～ 0 0011 1111 1111 1111　接第1、2片RAM**
  - **0 0100 0000 0000 0000 ～ 0 0111 1111 1111 1111　接第3、4片RAM**
  - **0 1000 0000 0000 0000 ～ 0 1011 1111 1111 1111　接第5、6片RAM**
  - **0 1100 0000 0000 0000 ～ 0 1111 1111 1111 1111　接第7、8片RAM**
  - **1 0000 0000 0000 0000 ～ 1 0011 1111 1111 1111　接第9、10片RAM**
  - **1 0100 0000 0000 0000 ～ 1 0111 1111 1111 1111　接第11、12片RAM**
  - **1 1000 0000 0000 0000 ～ 1 1111 1111 1111 1111　接1片ROM芯片**

- 选用1个3-8译码器，将地址线的高3位（A_{16}～A_{14}）接3-8译码器；译码器的输出为：Y_7～Y_0
- 其中Y_5～Y_0分别接6个RAM芯片，Y_7～Y_6通过一个或门接ROM芯片
- 地址线的低14位同时接12个RAM芯片，低15位接1个ROM芯片；16位数据线的高8位接6个RAM芯片、低8位接另外6个RAM，16位数据线直接接1个ROM芯片



- **4.8　用64Kx1位的DRAM芯片构成1Mx8位的存储器，若采用异步刷新，每行刷新间隔不超过2ms，则产生刷新信号的间隔时间是多少？假设读写周期为0.5μs，若采用集中刷新方式，则存储器刷新一遍最少需要多少个读写周期？CPU的“死”时间是多少？**

- **解：**
  - **DRAM芯片为64Kx1位，64K=65536=256X256，即存储体为256行、256列**
  - **采用异步刷新方式，刷新间隔=2ms/256行=7.8125 μs，即每隔7.8125 μs刷新一次，每次的刷新时间=读写周期= 0.5 μs**



异步刷新方式

  - **采用集中刷新方式，存储器刷新一遍需要256个读写周期（存储体有256行）**
  - **此时，CPU的“死”时间=256 x 0.5 μs =128 μs；死区率= 128 μs /2ms = 6.4 %**

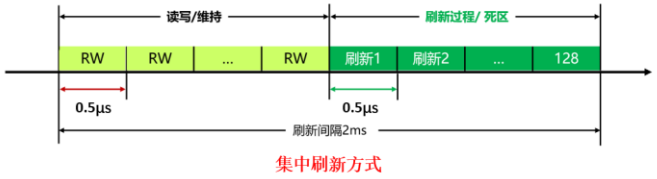
- **4.9　设有某动态RAM芯片，容量为64Kx1位，除电源线、接地线和刷新线外，该芯片的最小引脚数量是多少？**

- **解：**
  - **64Kx1位，地址线=16根（2^{16}=64K），因为是DRAM，采用地址复用技术（行地址、列地址复用），实际只需要8根地址线（一半的地址线）**
  - **数据线=1根**
  - **此外，DRAM芯片还需要/RAS和/CAS控制信号，共2根**
  - **因此该芯片的最小引脚数量是8+1+2=11根**

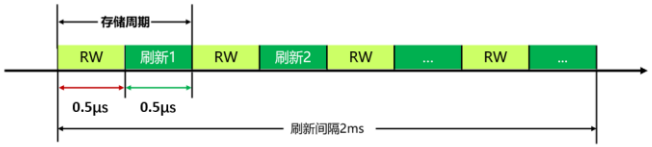
- **4.10　用16Kx1位的DRAM芯片构成64Kx8位的存储器，设存储器的读写周期为0.5μs，要使CPU在1μs内至少访问存储器一次，采用哪种刷新方式比较合适？若每行刷新间隔不超过2ms，该方式下刷新信号的产生周期是多少？**

- **解：**
  - 因为集中刷新方式有“死区”，因此不能采用集中刷新方式；而采用异步刷新方式或分散刷新方式比较合适。

  - **DRAM芯片为16Kx1位，16K=16384=128X128，即存储体为128行、128列；采用异步刷新方式，刷新间隔=2ms/128行=15.625 μs，即每隔15.625 μs刷新一次，每次的刷新时间=读写周期= 0.5 μs；因此异步刷新方式下刷新信号的产生周期是15.625 μs**

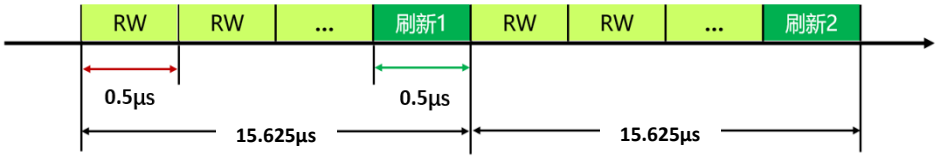


集中刷新方式



分散刷新方式

异步刷新方式



异步刷新方式

- **4.11　设cache的容量为2^{14}块，每块是一个32位字，主存容量是cache容量的256倍，其中有表4.13所示的数据（地址和数据均采用十六进制表示）。分别采用全相联映射、直接相联映射、四路组相联映射方式，将主存中这些数据载入cache后，cache各块中的数据内容及相应的标志是什么？**

表4.13　主存数据分布情况

地址	数据	地址	数据
000000	87568536	01FFFC	4FFFFC68
000008	87792301	FFFFF8	01BF2460
010004	9ABEFCD0		

- 解：
  - cache的容量为2^{14}块，cache块地址=14位
  - 每块是一个32位字=4B，块内偏移=2位（2^2=4）
  - cache地址=14+2=16位

  - 主存容量是cache的256倍，主存块为256x 2^{14}块= 2^{22}块，主存块地址=22位
  - 主存地址=主存块地址+块内偏移=22+2=24位

  - （1）全相联映射：
    - 主存地址=主存块地址（tag）+块内偏移（offset）=22位 + 2位
    - cache每一行的内容=有效位+主存块地址（tag）+数据块副本
    - 例如表4.13中的第一个数据块：主存地址=000000H，主存块地址=主存地址/4=000000H，数据块副本=87568536H，有效位=1，假设映射到cache的第0行；其他行类似得到

全相联映射方式cache每行的内容

cache行	有效位	主存块地址（tag）（22位）	数据块副本（32位）
0	1	000000	87568536
1	1	000002	87792301
2	1	004001	9ABEFCD0
3	1	007FFF	4FFFFC68
4	1	3FFFFE	01BF2460

- 4.11　设cache的容量为2^{14}块，每块是一个32位字，主存容量是cache容量的256倍，其中有表4.13所示的数据（地址和数据均采用十六进制表示）。分别采用全相联映射、直接相联映射、四路组相联映射方式，将主存中这些数据载入cache后，cache各块中的数据内容及相应的标志是什么？

表4.13　主存数据分布情况

地址	数据	地址	数据
000000	87568536	01FFFC	4FFFFC68
000008	87792301	FFFFF8	01BF2460
010004	9ABEFCD0		

- 解（续）：
  - cache的容量为2^{14}块，cache块地址=14位
  - 每块是一个32位字=4B，块内偏移=2位（2^2=4）
  - cache地址=14+2=16位
  - 主存容量是cache的256倍，主存块为256x 2^{14}块= 2^{22}块，主存块地址=22位
  - 主存地址=主存块地址+块内偏移=22+2=24位
  - （2）直接相联映射：
    - 主存地址=区地址（tag）+行索引（index）+块内偏移（offset）= 8位 + 14位 + 2位
    - cache每一行的内容=有效位+区地址（tag）+数据块副本
    - 例如表4.13中的第一个数据块：主存地址=000000H，区地址=00H，行索引=0000H，数据块副本=87568536H，有效位=1，映射到cache的第0000H行；其他行类似得到

直接相联映射方式cache每行的内容

	cache行	有效位	区地址（tag）（8位）	数据块副本（32位）
000000H=00 0000 0	0000	1	00	87568536
000008H=00 0002 0	0002	1	00	87792301
010004H=01 0001 0	0001	1	01	9ABEFCD0
01FFFCH= 01 3FFF 0	3FFF	1	01	4FFFFC68
FFFF8H = FF 3FFE 0	3FFE	1	FF	01BF2460

- **4.11　设cache的容量为2^{14}块，每块是一个32位字，主存容量是cache容量的256倍，其中有表4.13所示的数据（地址和数据均采用十六进制表示）。分别采用全相联映射、直接相联映射、四路组相联映射方式，将主存中这些数据载入cache后，cache各块中的数据内容及相应的标志是什么？**

表4.13　主存数据分布情况

地址	数据	地址	数据
000000	87568536	01FFFC	4FFFFC68
000008	87792301	FFFFF8	01BF2460
010004	9ABEFCD0		

- 解（续）：
  - cache的容量为2^{14}块，cache块地址=14位
  - 每块是一个32位字=4B，块内偏移=2位（2^2=4）
  - cache地址=14+2=16位
  - 主存容量是cache的256倍，主存块为256x 2^{14}块= 2^{22}块，主存块地址=22位
  - 主存地址=主存块地址+块内偏移=22+2=24位
  - （3）四路组相联映射：
    - 主存地址=标记（tag）+组索引（index）+块内偏移（offset）= 10位 + 12位 + 2位
    - cache每一行的内容=有效位+标记（tag）+数据块副本
    - 例如表4.13中的第一个数据块：主存地址=000000H，标记=000H，组索引=000H，数据块副本=87568536H，有效位=1，映射到cache的第000H组；其他行类似得到

四路组相联映射方式cache每行的内容

	cache组	有效位	标记（tag）（10位）	数据块副本（32位）
000000H=000 000 0	000	1	000	87568536
000008H=000 002 0	002	1	000	87792301
010004H=004 001 0	001	1	004	9ABEFCD0
01FFFCH= 007 FFF 0	FFF	1	007	4FFFFC68
FFFF8H = 3FF FFE 0	FFE	1	3FF	01BF2460

- **4.12** 某计算机的cache由64个存储块构成，采用四路组相联映射方式，主存包含4096个存储块，每块由128个字组成，访问地址为字地址。（1）主存地址和cache地址各有多少位？（2）按照题干条件中的映射方式，列出主存地址的划分情况，并标出各部分的位数。

- 解：
  - （1）
    - cache由64个存储块构成，cache的块地址=6位（2^6=64）；采用四路组相联映射方式，cache有16组，组地址=4位
    - 每块由128个字组，块内偏移=7位（2^7=128）
    - cache地址=6+7=13位
    - 主存包含4096个存储块，主存块地址=12位（2^{12}=4096）
    - 主存地址=主存块地址+块内偏移=12+7=19位
  - （2）
    - 四路组相联映射：主存地址=标记（tag）+组索引（index）+块内偏移（offset）= 8位 + 4位 + 7位

- **4.13** 某计算机的主存容量为4MB，cache容量为16KB，每块包含8个字，每字为32位，映射方式采用四路组相联。设cache的初始状态为空，CPU依次从主存第0、1、2、…、99号单元读出100个字（每次读一个字），并重复此操作10次，替换算法采用LRU算法。（1）求cache的命中率。（2）若cache比主存快10倍，分析采用cache后存储访问速度提高了多少？

- 解：
  - （1）
    - 主存容量为4MB，因此，主存地址=22位；cache容量为16KB，因此，cache地址=14位
    - 每块包含8个字，每字为32位，因此，每块=8x32=32B，块内偏移=5位
    - 主存块地址=22-5=17位，共2^{17}=131072块
    - cache块地址=14-5=9位，共2^9=512块；映射方式采用四路组相联，因此，cache有512/4=128组
    - 因为每块包含8个字，因此，主存第0、1、2、…、99号单元，分布在主存的第0、1、2、…、12块中（13个块）；13个主存块没有超出cache的组，因此不需要执行替换
    - 第1次读100个字时，只有读第0、8、16、24、32、40、48、56、64、72、80、88、96号单元时不命中（13次），其余87次都命中；后面重复操作时，全部命中；因此，1000次访问中，只有13次不命中，其余987次都命中，命中率=987/1000=98.7%

  - （2）
    - 采用cache后的平均访问时间：$t_a=ht_c+(1-h)t_m$；$t_m=10t_c$
    - 因此，$t_a=ht_c+(1-h)10t_c=0.987t_c+0.013x10t_c=1.117t_c$
    - 不采用cache的访问时间=$10t_c$
    - 用cache后存储访问速度提高了：$10t_c / 1.117t_c \approx$ 8.95倍

主存数据块

	第0字	第1字	第2字	第3字	第4字	第5字	第6字	第7字
第0块	0	1	2	3	4	5	6	7
第1块	8	9	10	11	12	13	14	15
第2块	16	17	18	19	20	21	22	23
第3块	24	25	26	27	28	29	30	31
第4块	32	33	34	35	36	37	38	39
第5块	40	41	42	43	44	45	46	47
第6块	48	49	50	51	52	53	54	55
第7块	56	57	58	59	60	61	62	63
第8块	64	65	66	67	68	69	70	71
第9块	72	73	74	75	76	77	78	79
第10块	80	81	82	83	84	85	86	87
第11块	88	89	90	91	92	93	94	95
第12块	96	97	98	99				
…								
…								
…								
第131071块								

- **4.14** 假定某数组元素按行优先顺序存放在主存中，则在以下两段伪代码A和B中，分析下列问题：（1）两段代码中对数组访问的时间局部性和空间局部性。（2）变量sum的时间局部性和空间局部性。（3）for循环体对指令访问的时间局部性和空间局部性。

- 解：
  - （1）
    - 程序A的数组a，其存储次序是按行优先顺序，访问顺序也是按行优先顺序，因此具有空间局部性；但是数组a的每个元素只使用一次，因此不具有时间局部性。
    - 程序B的数组a，其存储次序是按行优先顺序，但是访问顺序是按列优先顺序，因此不具有空间局部性；并且数组a的每个元素只使用一次，因此也不具有时间局部性。
  - （2）sum是单个变量，不存在空间局部性；但是变量sum在每次循环执行时都会用到，因此具有时间局部性。
  - （3）for循环体中的指令会被反复执行，因此具有时间局部性；此外for循环体中的指令序列通常会顺序执行，因此也具有一定的空间局部性。

程序A：

```
int sum_array_A(int a[M][N])
{
  int i,j,sum=0;
    for (i = 0; i < M; i++)
          for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

程序B：

```
int sum_array_B(int a[M][N])
{
  int i,j,sum=0;
    for (i = 0; i < N; i++)
          for (j = 0; j < M; j++)
            sum += a[j][i];
    return sum;
}
```

- **4.15** 主存容量为**8MB**，虚存容量为**2GB**，分页管理时若页面大小为**4KB**，求出对应的**VPN**、**VPO**、**PPN**、**PPO**的位数。

- 解：
  - 虚拟地址（**VA**）= 虚拟页号（**VPN**）+ 虚拟页偏移（**VPO**）
  - 物理地址（**PA**）= 物理页号（**PPN** ）+ 物理页偏移（**PPO**）
  - 主存容量为**8MB**，因此，**PA=23**位
  - 虚存容量为**2GB**，因此，**VA=31**位
  - 分页管理时若页面大小为**4KB**，因此，**VPO=PPO=12**位
  - **VPN=VA-VPO=31-12=19**位
  - **PPN=PA-PPO=23-12=11**位

- 4.16 某页式虚拟存储器共8页，每页为1KB，主存容量为4KB，页表如表4.14所示。

表4.14 虚拟存储器页表

虚页号	0	1	2	3	4	5	6	7
实页号	3	2	1	2	3	1	0	0
载入位	1	1	0	0	1	0	1	0

- （1）失效的页有哪几页？
- （2）虚地址0、3028、1023、2048、4096、8000（均为十进制）的实地址分别是多少？

- 解：
  - （1）
    - 表4.14中载入位=0的为失效页，即虚页号为2、3、5、7的页
  - （2）
    - 虚拟地址（VA）= 虚拟页号（VPN）+ 虚拟页偏移（VPO）；物理地址（PA）= 物理页号（PPN ）+ 物理页偏移（PPO）
    - 每页为1KB，因此，VPO=PPO=10位；主存容量为4KB，因此，PA=12位
    - 页式虚拟存储器共8页，因此，VPN=3位；VA=VPN+VPO=3+10=13位；PPN=PA-PPO=12-10=2位

    ① 虚地址=0=000 00 0000 0000，因此，VPN=000=0，查表4.14，得到实页号=PPN=3=11，载入位=1，命中，实地址=11 00 0000 0000=C00H=3072

    ② 虚地址=3028= 010 11 1101 0100，因此，VPN=010=2，查表4.14，得到实页号=PPN=1=01，载入位=0，缺页

    ③ 虚地址=1023= 000 11 1111 1111，因此，VPN=000=0，查表4.14，得到实页号=PPN=3=11，载入位=1，命中，实地址=11 11 1111 1111=FFFH=4095

    ④ 虚地址=2048= 010 00 0000 0000，因此，VPN=010=2，查表4.14，得到实页号=PPN=1=01，载入位=0，缺页

    ⑤ 虚地址=4096=100 00 0000 0000，因此，VPN=100=4，查表4.14，得到实页号=PPN=3=11，载入位=1，命中，实地址=11 00 0000 0000=C00H=3072

    ⑥ 虚地址=8000= 111 11 0100 0000，因此，VPN=111=7，查表4.14，得到实页号=PPN=0=00，载入位=0，缺失

- 4.17 某计算机系统中有一个TLB和L1级数据cache，存储系统按字节编址，虚拟存储器容量为2GB，主存容量为4MB，页大小为128KB，TLB采用四路组相联方式，共有16个页表项。cache容量为16KB，每块包含8个字，每字为32位，映射方式采用四路组相联，回答下列问题：
- （1）虚拟地址中哪几位表示虚拟页号？哪几位表示页内地址？虚拟页号中哪几位表示TLB标记？哪几位表示TLB索引？
- （2）物理地址中哪几位表示物理页号？哪几位表示偏移地址？
- （3）为实现主存与数据cache之间的组相联映射，对该地址应进行怎样的划分？

- 解：
  - （1）
    - 虚拟地址（VA）= 虚拟页号（VPN）+ 虚拟页偏移（VPO）
    - 物理地址（PA）= 物理页号（PPN）+ 物理页偏移（PPO）
    - 虚拟存储器容量为2GB，因此，VA=31位
    - 主存容量为4MB，因此，PA=22位
    - 页大小为128KB，因此，VPO=PPO=17位
    - VPN=VA-VPO=31-17=14位
    - PPN=PA-PPO=22-17=5位
    - 虚拟页号=14位；页内地址=17位
    - TLB采用四路组相联方式，VPN=TLB标记+TLB索引
    - TLB共有16个页表项，四路组相联，有16/4=4组，因此，TLB索引=2位，TLB标记=14-2=12位
  - （2）
    - 物理页号=5位；偏移地址=17位
  - （3）
    - cache容量为16KB，因此，cache的地址=14位
    - 每块包含8个字，每字为32位，因此，块地址=3+2=5位
    - cache块地址=14-5=9位，共512块
    - cache采用四路组相联，因此cache有512/4=128组，组地址=7位
    - 主存地址=22位=标记+组地址+偏移地址=10位+7位+5位

- 4.18 某计算机采用页式虚拟存储管理方式，按字节编址，虚拟地址为32位，物理地址为24位，页大小为8KB；TLB采用全相联映射；cache数据区大小为64KB，按二路组相联方式组织，主存块大小为64B。存储访问过程的示意图如图4.57所示。请回答下列问题：
- （1）图中字段A～G的位数各是多少？TLB标记字段B中存放的是什么信息？
- （2）将块号为4099的主存块装入cache中时，映射的cache组号是多少？对应H字段的内容是什么？
- （3）cache缺失处理的时间开销大，还是缺页处理的时间开销大？为什么？
- （4）为什么cache可以采用写穿策略，而修改页面内容时总是采用写回策略？

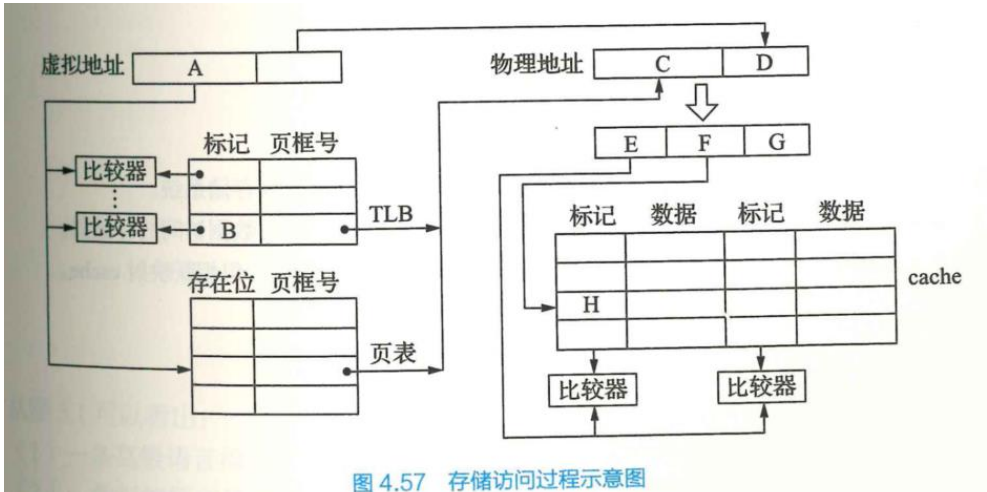


图 4.57 存储访问过程示意图

- 解：
  - （1）
    - 虚拟地址（VA）= 虚拟页号（VPN）+ 虚拟页偏移（VPO）
    - 物理地址（PA）= 物理页号（PPN ）+ 物理页偏移（PPO）
    - 页大小为8KB，因此，VPO=PPO=13位；VPN=VA-VPO=32-13=19位；PPN=PA-PPO=24-13=11位
    - A=VPN=19位，C=PPN=11位，D=PPO=13位
    - TLB采用全相联映射，因此，TLB中的标记=B=VPN=19位
    - 主存块大小为64B，因此偏移地址=6位=G
    - cache数据区大小为64KB，因此，cache有64KB/64B=1K=1024行， 二路组相联方式组织，cache有1024/2=512组
    - 因此，F=cache的组数=9位
    - E=24-9-6=9位
    - A=19位，B=19位，C=11位，D=13位，E=9位，F=9位，G=6位
    - TLB 中标记字段 B 的内容是虚拟页号（VPN），表示该 TLB 项对应哪个虚页的页表项。

- 4.18 某计算机采用页式虚拟存储管理方式，按字节编址，虚拟地址为32位，物理地址为24位，页大小为8KB；TLB采用全相联映射；cache数据区大小为64KB，按二路组相联方式组织，主存块大小为64B。存储访问过程的示意图如图4.57所示。请回答下列问题：
- （1）图中字段A～G的位数各是多少？TLB标记字段B中存放的是什么信息？
- （2）将块号为4099的主存块装入cache中时，映射的cache组号是多少？对应H字段的内容是什么？
- （3）cache缺失处理的时间开销大，还是缺页处理的时间开销大？为什么？
- （4）为什么cache可以采用写穿策略，而修改页面内容时总是采用写回策略？

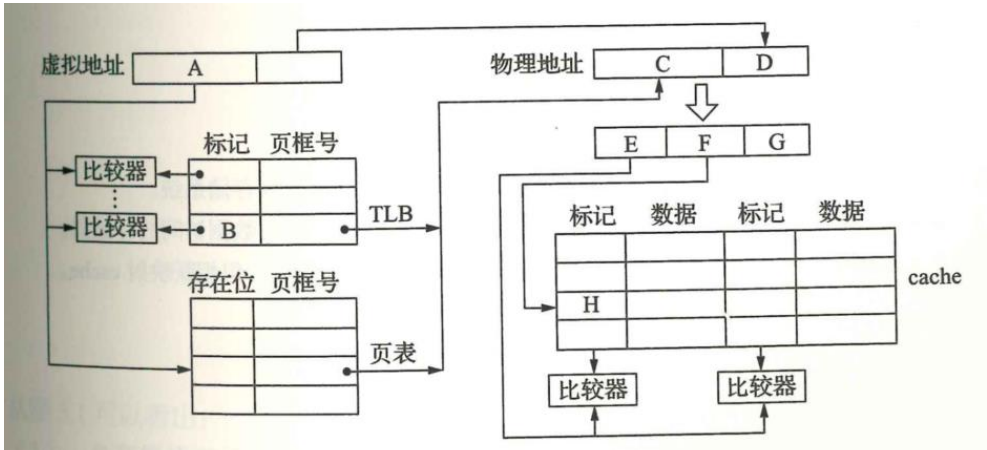


图 4.57 存储访问过程示意图

- 解（续）：
  - （2）
    - cache采用二路组相联映射方式，主存地址=标记（9位）+组索引（9位）+块内偏移（6位）=主存块地址（18位）+块内偏移（6位）
    - 块号为4099（十进制）的主存块，块地址=4099= 000001000 000000011
    - 组索引= 000000011=3
    - 标记= 000001000
    - 映射的cache组号是3；对应H字段的内容是000001000

- 4.18 某计算机采用页式虚拟存储管理方式，按字节编址，虚拟地址为32位，物理地址为24位，页大小为8KB；TLB采用全相联映射；cache数据区大小为64KB，按二路组相联方式组织，主存块大小为64B。存储访问过程的示意图如图4.57所示。请回答下列问题：
- （1）图中字段A～G的位数各是多少？TLB标记字段B中存放的是什么信息？
- （2）将块号为4099的主存块装入cache中时，映射的cache组号是多少？对应H字段的内容是什么？
- （3）cache缺失处理的时间开销大，还是缺页处理的时间开销大？为什么？
- （4）为什么cache可以采用写穿策略，而修改页面内容时总是采用写回策略？

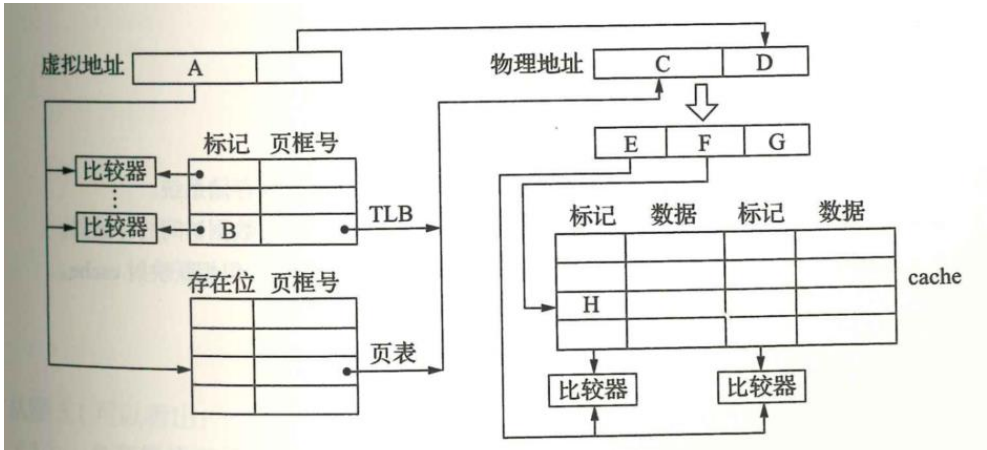


图 4.57 存储访问过程示意图

- 解（续）：
  - （3）
    - 因为缺页处理需要访问磁盘，而 cache 缺失只要访问主存，cache 缺失带来的开销小，而处理缺页的开销大。
  - （4）
    - cache可以采用写穿策略时，需要同时写cache和写主存。
    - 修改页面内容时（即虚拟存储器），如果采用写穿策略，则需要同时写主存和写磁盘。
    - 而写磁盘比写主存慢得多；所以 cache 可以采用写穿策略， 而虚存则应采用写回策略（不需要同时写磁盘）。

- 4.19　某计算机采用页式虚拟存储管理方式，按字节编址。CPU进行存储访问的过程如图4.58所示。请回答下列问题：
- （1）主存的物理地址占多少位？
- （2）TLB采用什么映射方式？TLB是用SRAM还是DRAM实现？
- （3）cache采用什么映射方式？若cache采用LRU替换算法和写回策略，则cache每行中除数据（Data）、tag和有效位外，还应有哪些附加位？cache总容量是多少？cache中有效位的作用是什么？
- （4）若CPU给出的虚拟地址为0008 C040H，则对应的物理地址是多少？是否在cache中命中？说明理由。若CPU给出的虚拟地址是0007 C260H，则该地址所在主存块映射到的cache组号是多少？

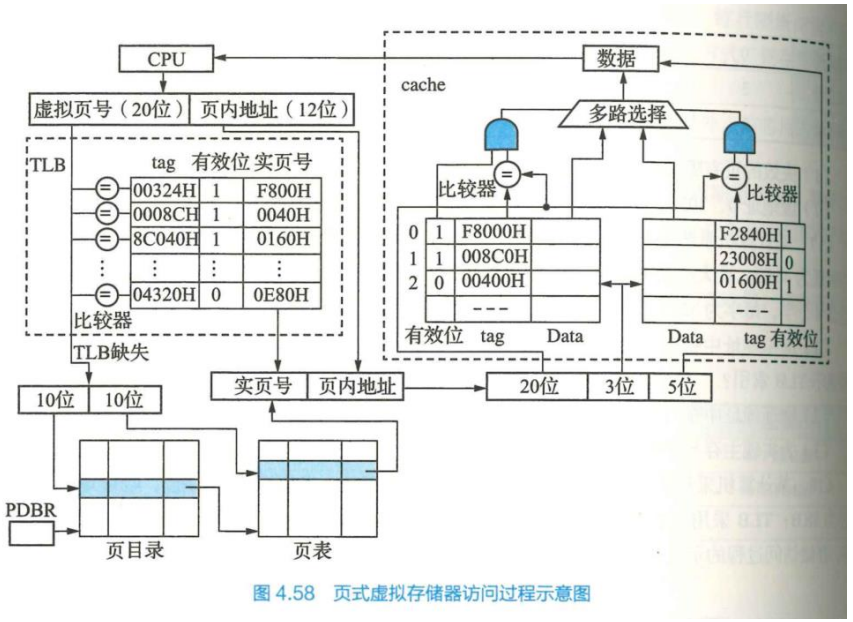


图 4.58　页式虚拟存储器访问过程示意图

- 解：
  - （1）
    - 主存物理地址=20+3+5=28位
  - （2）
    - TLB的tag=20位，与虚拟页号的位数相等，因此TLB采用全相联映射方式
    - TLB用SRAM实现

- 4.19 某计算机采用页式虚拟存储管理方式，按字节编址。CPU进行存储访问的过程如图4.57所示。请回答下列问题：
- （1）主存的物理地址占多少位？
- （2）TLB采用什么映射方式？TLB是用SRAM还是DRAM实现？
- （3）cache采用什么映射方式？若cache采用LRU替换算法和写回策略，则cache每行中除数据（Data）、tag和有效位外，还应有哪些附加位？cache总容量是多少？cache中有效位的作用是什么？
- （4）若CPU给出的虚拟地址为0008 C040H，则对应的物理地址是多少？是否在cache中命中？说明理由。若CPU给出的虚拟地址是0007 C260H，则该地址所在主存块映射到的cache组号是多少？

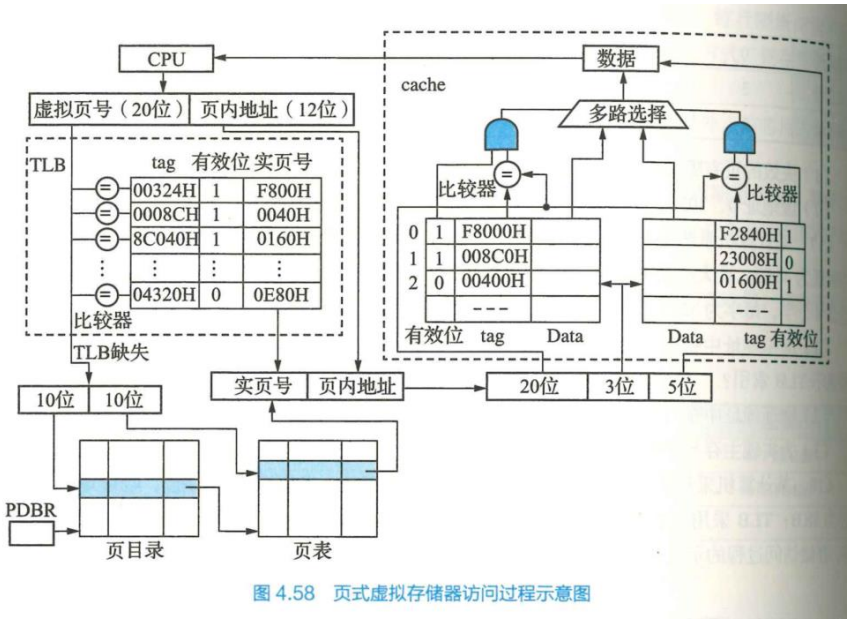


图 4.58 页式虚拟存储器访问过程示意图

- 解（续）：
  - （3）
    - cache采用二路组相联映射方式
    - cache采用LRU替换算法，因为是二路组相联映射方式，只需要在cache的每行中增加 1 位 LRU 位用于替换计数
    - cache采用写回策略，则还应有脏位（Dirty Bit）
    - 因此还应有2位附加位（LRU位、脏位）
    - cache有8组（图4.58中的3位），每组2行，共16行；每行包括：数据块、tag、有效位、LRU位、脏位
    - 主存物理地址=20+3+5=28位，因此cache的块内地址=5位，即数据块=32x8
    - tag=20位
    - cache总容量=16行x(32x8 + 20 + 1 + 1 +1)=16x279=4464=558字节
    - cache的有效位用来指出所在 cache 行中的信息是否有效

- 4.19　某计算机采用页式虚拟存储管理方式，按字节编址。CPU进行存储访问的过程如图4.57所示。请回答下列问题：
- （1）主存的物理地址占多少位？
- （2）TLB采用什么映射方式？TLB是用SRAM还是DRAM实现？
- （3）cache采用什么映射方式？若cache采用LRU替换算法和写回策略，则cache每行中除数据（Data）、tag和有效位外，还应有哪些附加位？cache总容量是多少？cache中有效位的作用是什么？
- （4）若CPU给出的虚拟地址为0008 C040H，则对应的物理地址是多少？是否在cache中命中？说明理由。若CPU给出的虚拟地址是0007 C260H，则该地址所在主存块映射到的cache组号是多少？

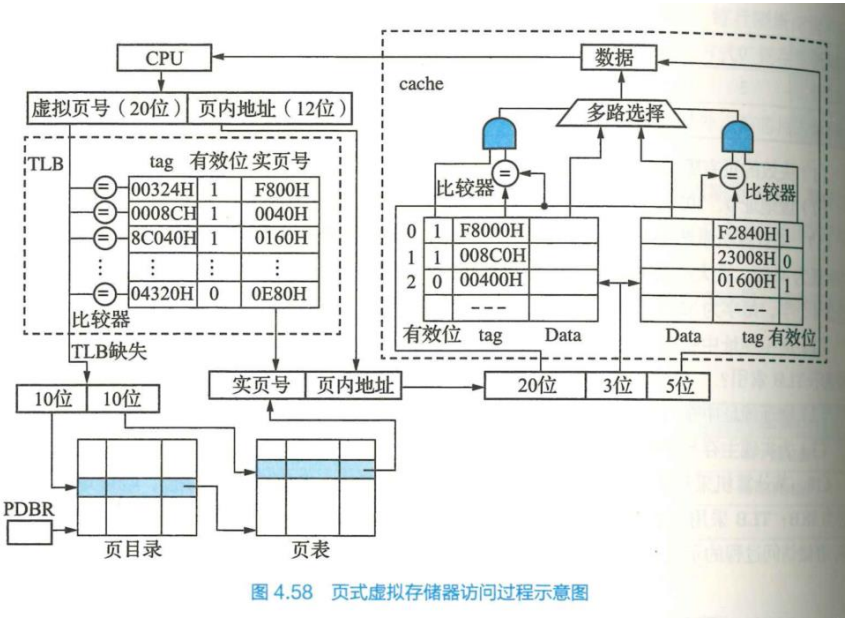


图 4.58　页式虚拟存储器访问过程示意图

- 解（续）：
  - （4）
    - 虚拟地址为0008 C040H= 0008C 040
    - 根据0008CH查TLB，得到实页号=0040H，并且有效位=1
    - 因此，物理地址=0040 040H
    - 物理地址=标记（20位）+组索引（3位）+块内偏移（5位）=0040 040H=0000 0000 0100 0000 0000 010 00000
    - 组索引=010=2，标记=00400H
    - 查二路组相联映射cache的第2组，有两个标记（00400H、01600H），有一个比较相等，但是其有效位=0，不命中
    - 虚拟地址为0007 C260H= 0007C 260=VPN（20位） + VPO（12位）
    - 主存地址=PPN（16位） + PPO（12位）= PPN（16位） + 260H = xxxx xxxx xxxx xxxx 0010 011 00000
    - 因此，组索引=011=3，该地址所在主存块映射到的cache组号是3

Thanks