

计算机组成与系统结构

第三章 多层次的存储器 (3)

吕昕晨

lvxinchen@bupt.edu.cn

网络空间安全学院

多层次的存储器——SRAM





SRAM存储器

- SRAM存储器结构与操作
- SRAM实例：Cache
 - Cache基本原理
 - Cache地址映射
 - Cache读写操作
 - Cache替换策略

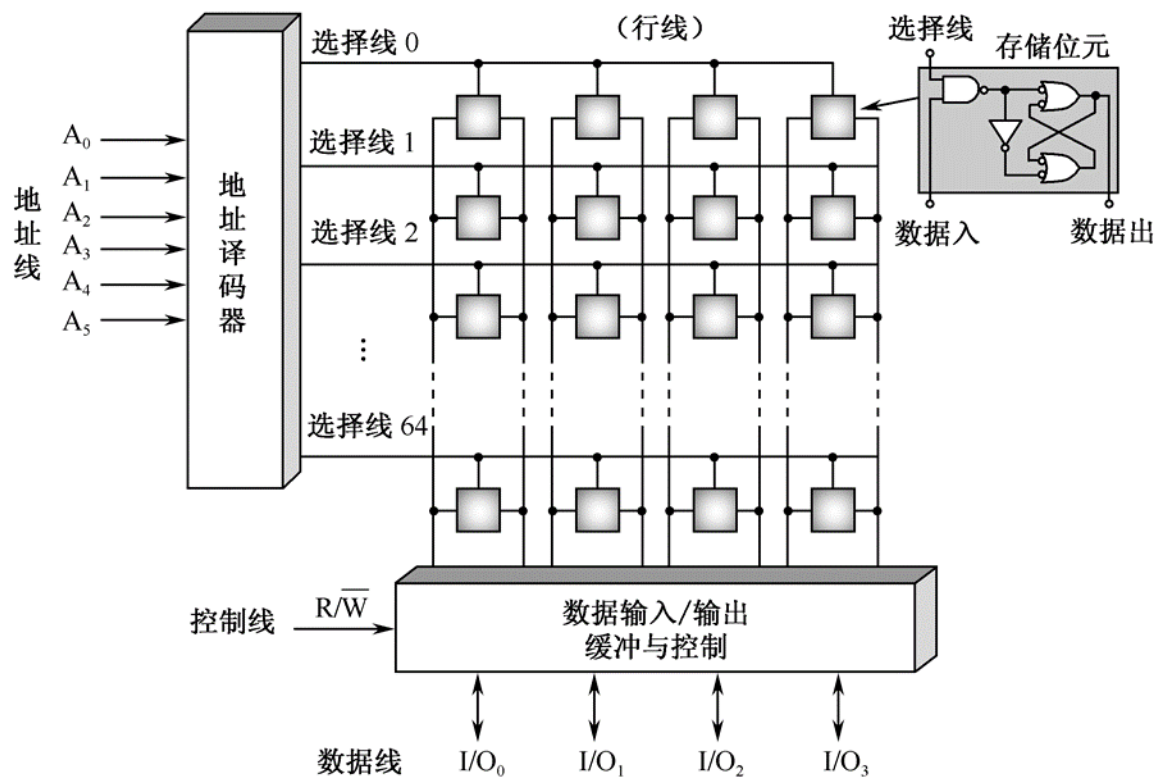
基本的静态存储元阵列



1、存储位元（触发器）

2、三组信号线

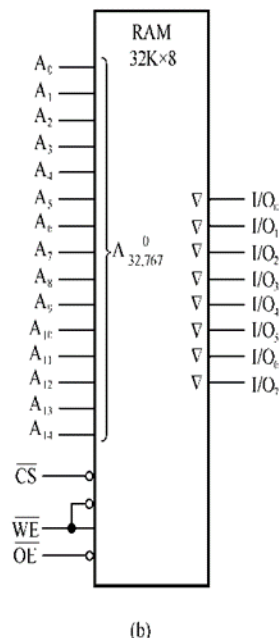
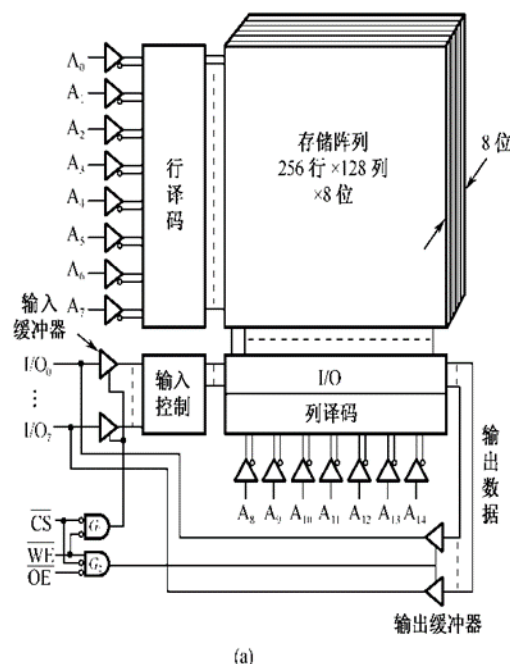
- 地址线
- 数据线
- 行线
- 列线
- 控制线



基本的SRAM逻辑结构



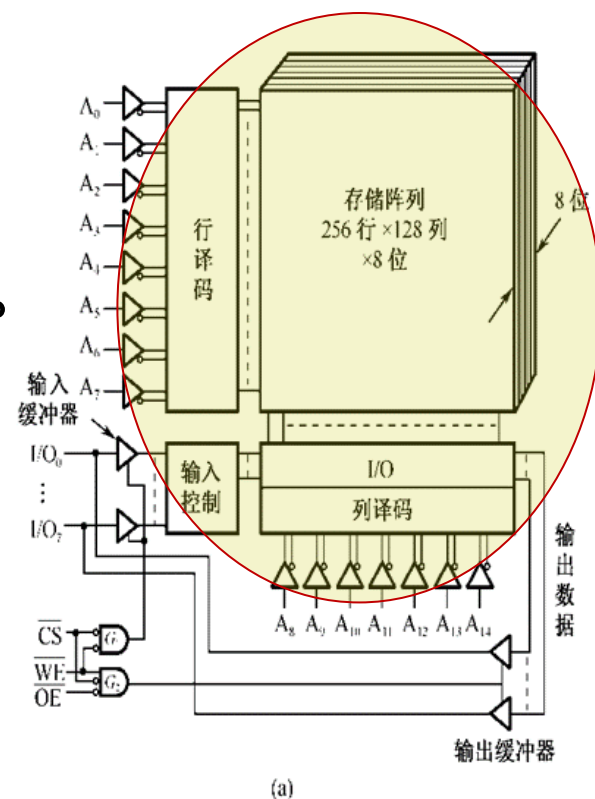
- SRAM大多采用双译码方式（效率），以便组织更大的存储容量
- 采用了二级译码
 - 地址分成两部分（行、列译码）
- 右图为SRAM对应逻辑图
 - 地址线
 - 输入输出控制
 - 片选与使能信号



基本的SRAM逻辑结构



- 存储体 ($256 \times 128 \times 8$)
 - 通常把各个字的同一个字的同一位集成在一个芯片 ($32K \times 1$) 中, 32K位排成 256×128 的矩阵。8个片子就可以构成32KB
- 地址译码器
 - 采用双译码的方式
 - $A_0 \sim A_7$ 为行地址译码线
 - $A_8 \sim A_{14}$ 为列地址译码线



SRAM读/写时序



- 时序过程

- ① 地址信号

- 选中存储单元

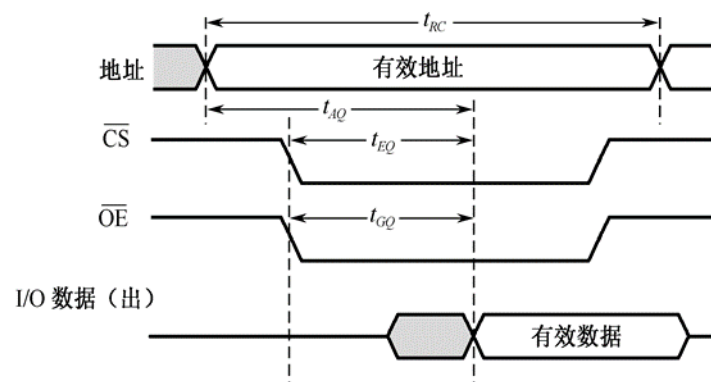
- 维持有效

- ② 片选信号

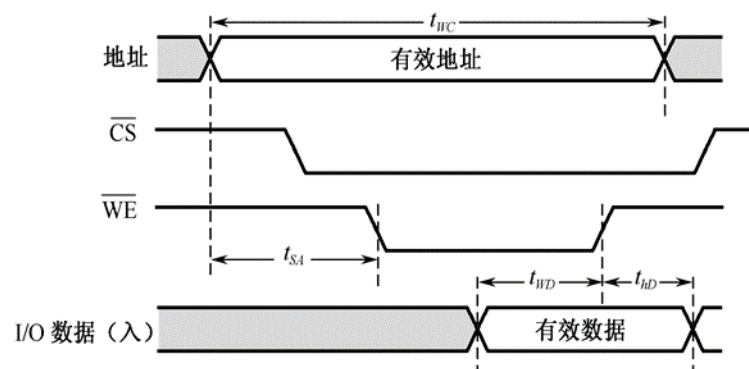
- ③ 读出、写入信号

- 控制读出、写入操作

- ④ 数据信号



(a) 读周期 (\overline{WE} 高)

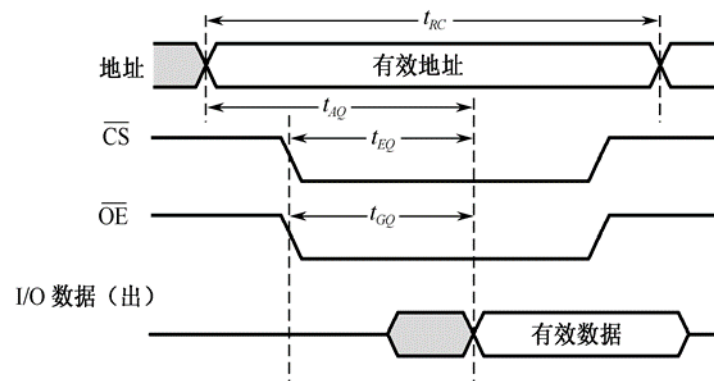


(b) 写周期 (\overline{WE} 低)

DRAM/SRAM时序

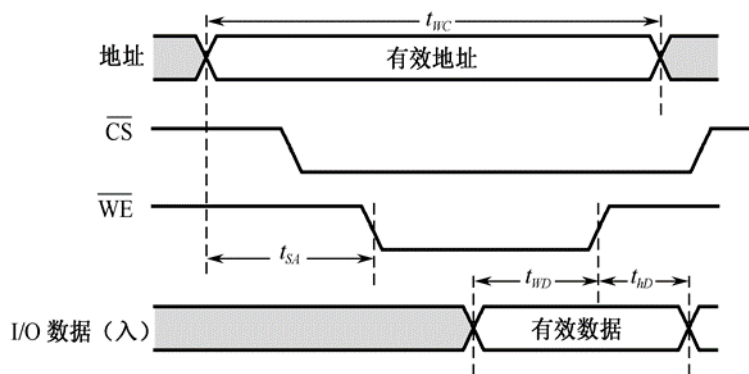
时序过程

- ① 地址信号
- ② 片选信号
- ③ 读出、写入信号
- ④ 数据信号

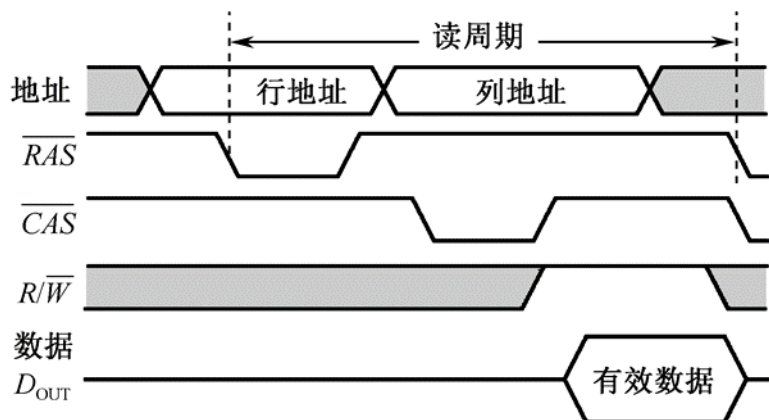


SRAM

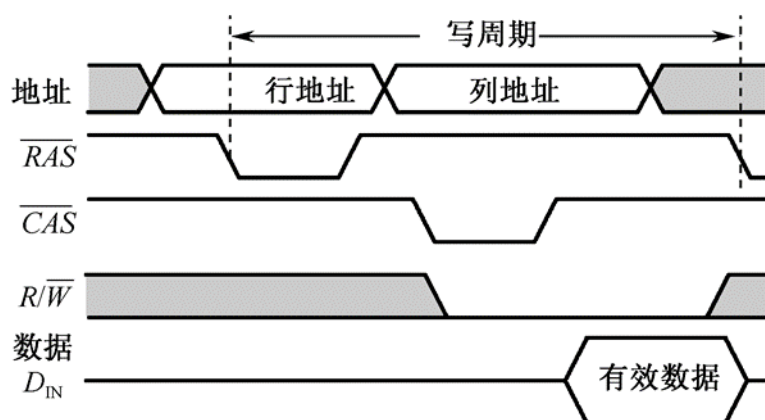
(a) 读周期 (\overline{WE} 高)



(b) 写周期 (\overline{WE} 低)



(a) 读周期



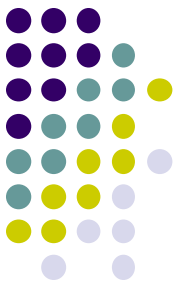
(b) 写周期

DRAM



SRAM存储器与Cache

- SRAM存储器结构与操作
- SRAM实例：Cache
 - Cache基本原理
 - Cache地址映射
 - Cache读写操作
 - Cache替换策略

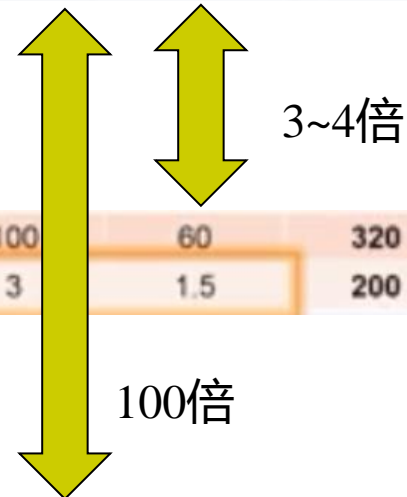


回顾：CPU、SRAM与DRAM速率

CPU		1980	1990	2000	2010	2010:1980
	Name	8080	386	Pentium II	Core i7	/
	Clock rate(MHz)	1	20	600	2,500	2,500
	Cycle time(ns)	1,000	50	1.6	0.4	2,500
	Cores	1	1	1	4	4
	Effective Cycle time(ns)	1,000	50	1.6	0.1	10,000

SRAM	\$/MB	19,200	320	100	60	320
	access time(ns)	300	35	3	1.5	200

DRAM	\$/MB	8,000	100	1	0.06	130,000
	access time(ns)	375	100	60	40	9
	typical size(MB)	0.064	4	64	8,000	125,000

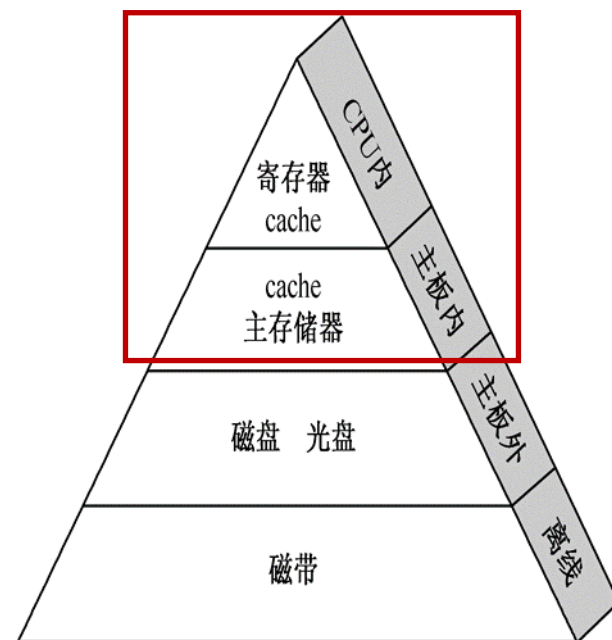


Cache整体目标



解决CPU和主存速度不匹配问题

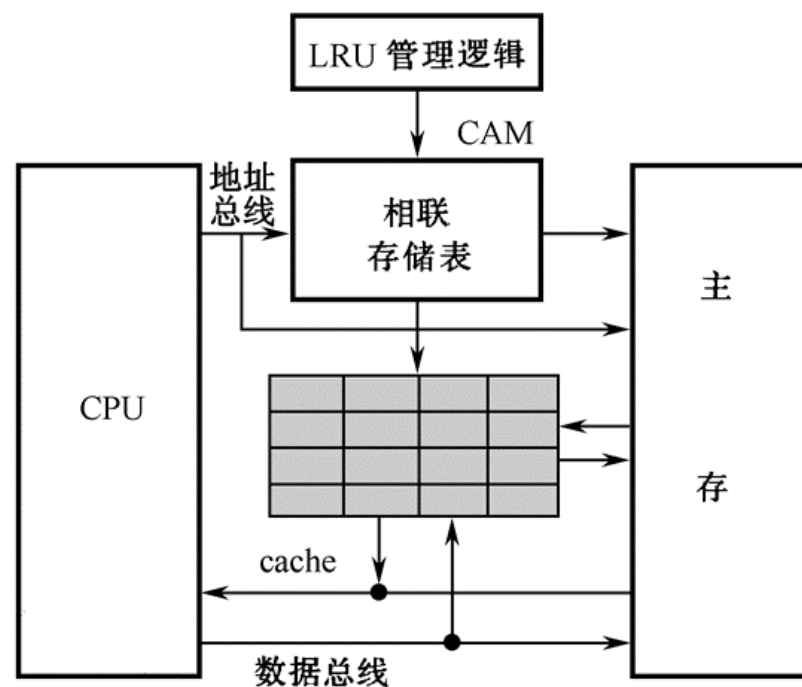
- 一般采用高速的SRAM构成。
- CPU和主存之间的速度差别很大采用两级或多级Cache系统
- 早期的一级Cache在CPU内，二级在主板上
- 现在的CPU内带L1 Cache和L2 Cache
- 全由硬件调度，对用户透明





Cache逻辑结构与功能

- 组成结构
 - Cache存储表
 - 相联存储表
 - 控制器
- 功能
 - 地址映射
 - 替换策略
 - 写一致性
 - 性能评价



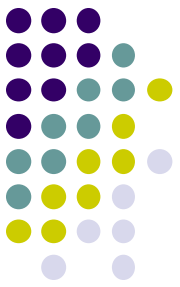


Cache基本原理 (1)

- Cache基本原理——程序局部性原理
 - 经验性结论
 - 程序在时间与空间上都表现局部性
- 时间局部性
 - 最近被访问存储单元很快还会被访问
- 空间局部性
 - 正在被访问存储单元附近单元很快会被访问

```
for (i=0; i<1000; i++)  
    for (j=0; j<200; j++)  
        sum += a[i][j];
```

典型程序段

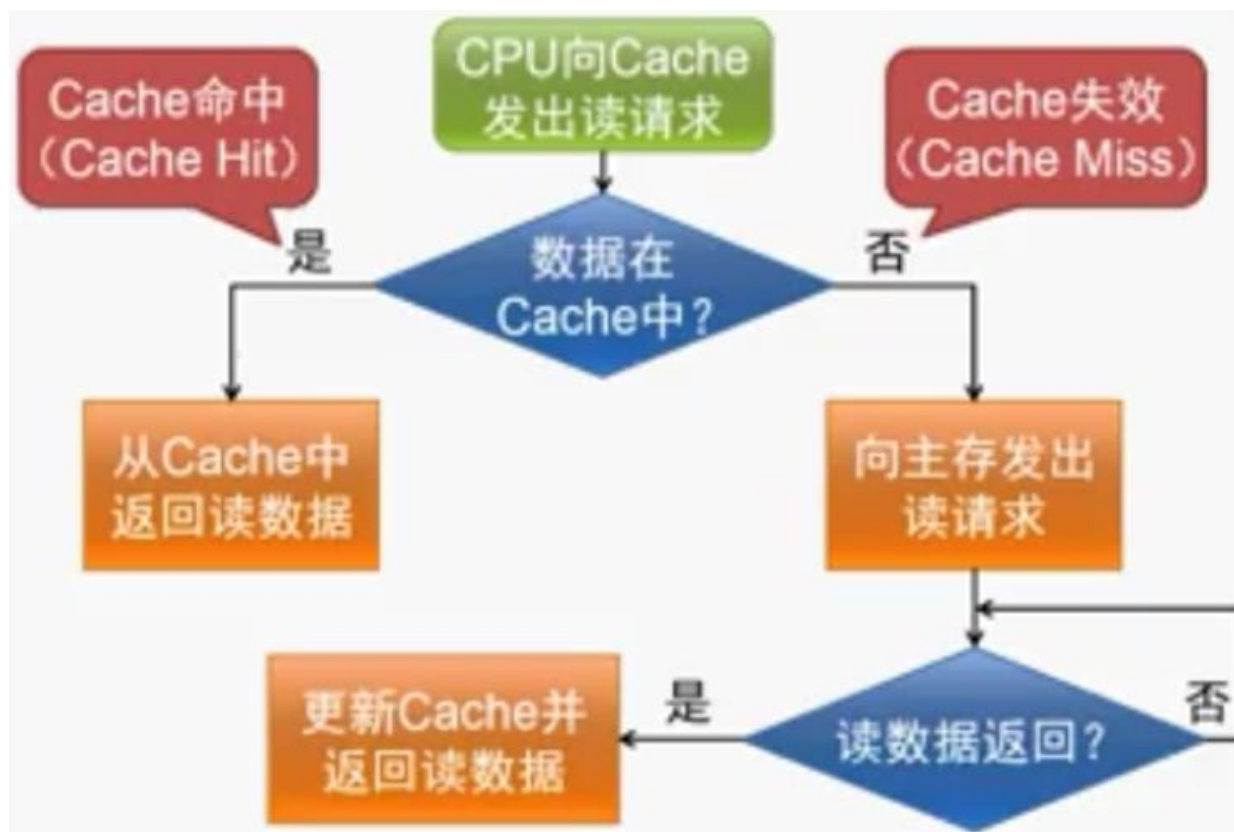


Cache基本原理 (2)

- Cache对**空间局部性**利用
 - 从主存中取回数据时，同时取回与对应位置相邻的主存单元数据
 - 以**数据块**为单位进行数据交换（行）
- Cache对**时间局部性**利用
 - 优先保存**近期频繁**被访问主存单元数据
 - 替换策略：LRU、LFU



Cache访问流程





Cache性能指标

命中率

$$h = \frac{N_e}{N_e + N_m}$$

平均访问时间

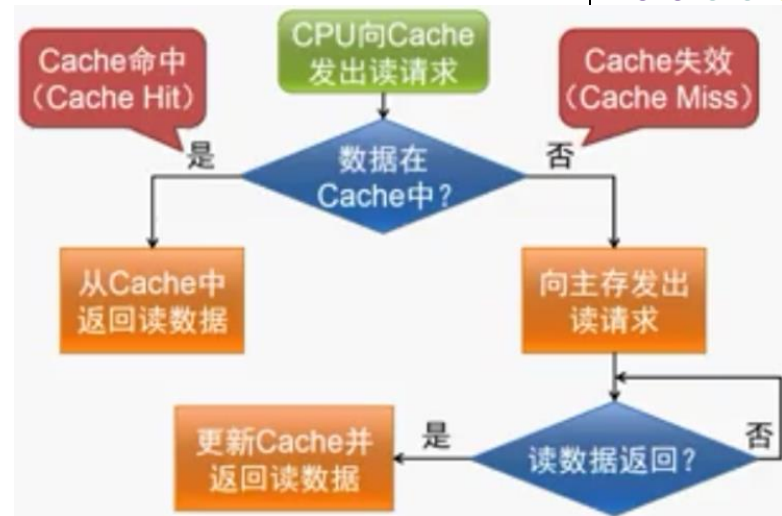
$$t_a = ht_c + (1 - h)t_m$$

访问效率

$$e = \frac{t_c}{t_a} = \frac{1}{r + (1 - r)h}$$

内存与Cache
的速度比

$$r = t_m / t_c$$





Cache命中率与效率关系

- 访存时间对比
 - CPU v.s. Cache: 3倍
 - CPU v.s. 内存: 100倍
- 平均访存时间
 - 命中率 $h=95\%$
$$3 * 0.95 + 0.05 * 100 = 7.85$$
 - 命中率 $h=99\%$
$$3 * 0.99 + 0.01 * 100 = 3.97$$
- 命中率提高4%，性能提升1倍
- cache的命中率应尽可能接近于1

		1980	1990	2000	2010	2010:1980
CPU	Name	8080	386	Pentium II	Core i7	/
	Clock rate(MHz)	1	20	600	2,500	2,500
	Cycle time(ns)	1,000	50	1.6	0.4	2,500
	Cores	1	1	1	4	4
	Effective Cycle time(ns)	1,000	50	1.6	0.1	10,000

	1980	1990	2000	2010	2010:1980	
DRAM	\$/MB	8,000	100	1	0.06	130,000
	access time(ns)	375	100	60	40	9
	typical size(MB)	0.064	4	64	8,000	125,000



CPU执行一段程序时，cache完成存取的次数为1900次，主存完成存取的次数为100次，已知cache存取周期为50ns，主存存取周期为250ns

1) Cache命中率?

A 0.8

B 0.85

C 0.9

☒ D 0.95

$$h = \frac{N_e}{N_e + N_m}$$

$$h = 1900 / (1900 + 100) \\ = 0.95$$



CPU执行一段程序时，cache完成存取的次数为1900次，主存完成存取的次数为100次，已知cache存取周期为50ns，主存存取周期为250ns

2) 平均访存时间?

A 55

B 60

C 65

D 70

$$t_a = ht_c + (1 - h)t_m$$

$$t_a = 0.95 * 50 + 0.05 * 250 \\ = 60$$



例题：多级cache计算

现有一处理器，假设其基本CPI为1.0（所有访问在第一级cache中命中），时钟频率5GHz。

设平均每条指令在第一级cache中产生的缺失率为2%。假定访问一次主存储器的时间为100ns，其中包括所有缺失处理。

若增加一个二级cache，命中或缺失的访问时间都为5ns，且容量大到可使必须访问主存的缺失率降为0.5%，问处理器速度提高多少。



例题：多级cache计算

解：

CPI（每条指令周期数）：周期数/指令条数

主存周期数： $100\text{ns}/0.2\text{ns}=500$ 周期

总的CPI=基本CPI + 存储器中停顿时钟周期

1) 只有一级Cache： $1 \text{ (CPI)} + 500 \times 0.02 = 11$

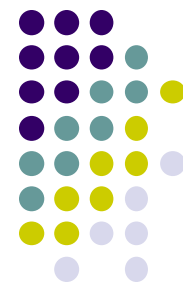
2) 有两级Cache： $1 + 0.02 \times 25 \text{ (5/0.2)} + 500 \times 0.005 = 4$

后者是前者CPU性能的： $11.0 \div 4.0 = 2.8$ 倍

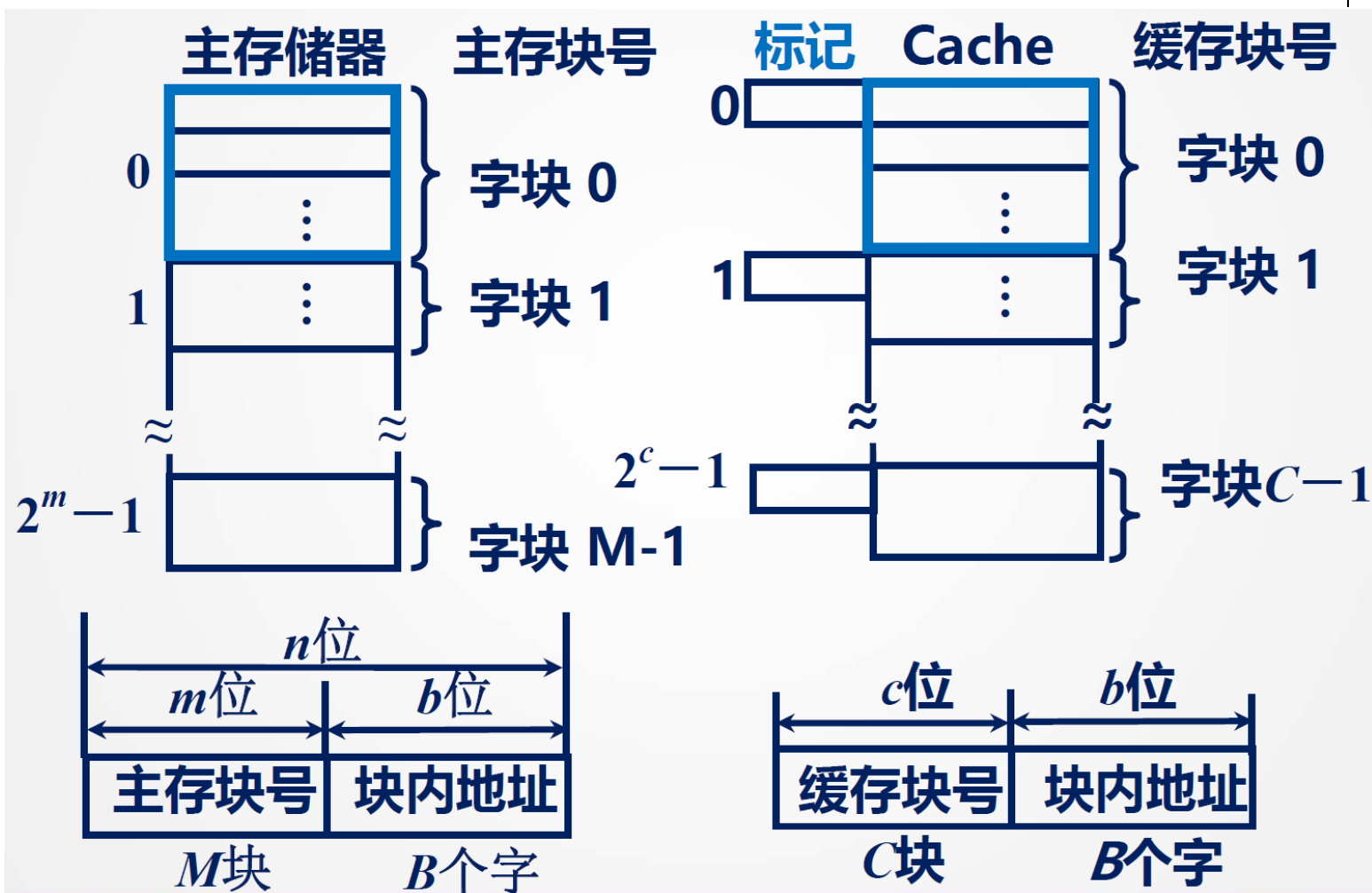


SRAM存储器与Cache

- SRAM存储器结构与操作
- SRAM实例：Cache
 - Cache基本原理
 - Cache地址映射（重点）
 - Cache读写操作
 - Cache替换策略



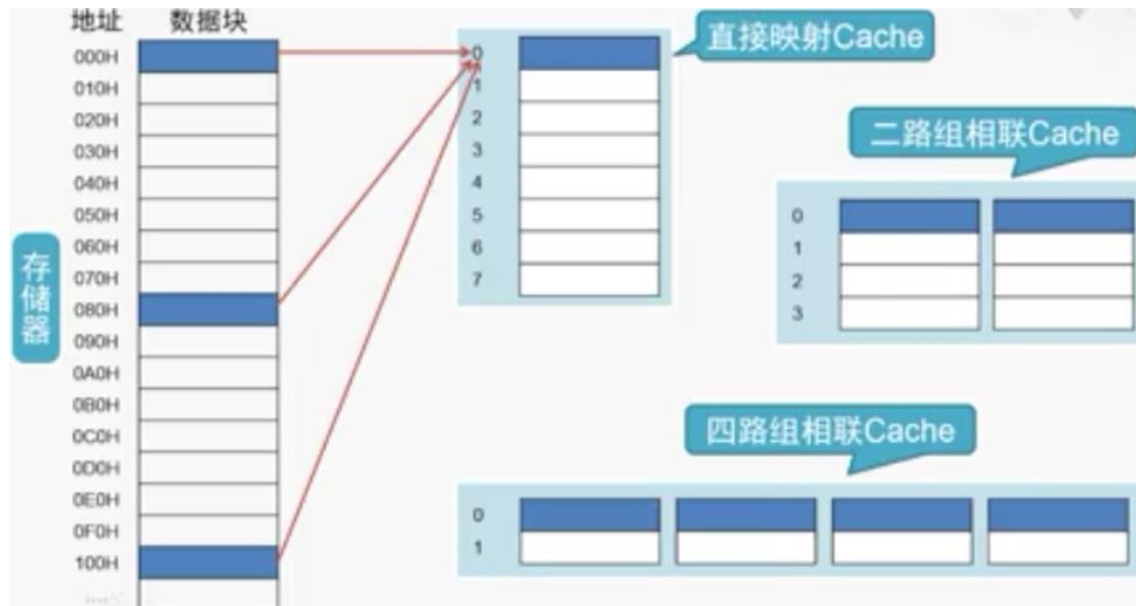
Cache—主存的编址 (行/数据块)





Cache地址映射

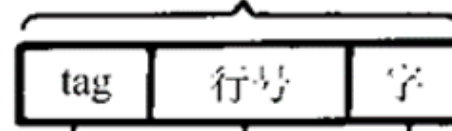
- Cache大小：C个数据块
 - 内存M数据块， $M \rightarrow C$ 映射
 - 直接映射： $C * 1$
 - 全相联： $1 * C$
 - 组相联： U （组数） $* V$ （组内块数） $= C$



1 直接地址映射

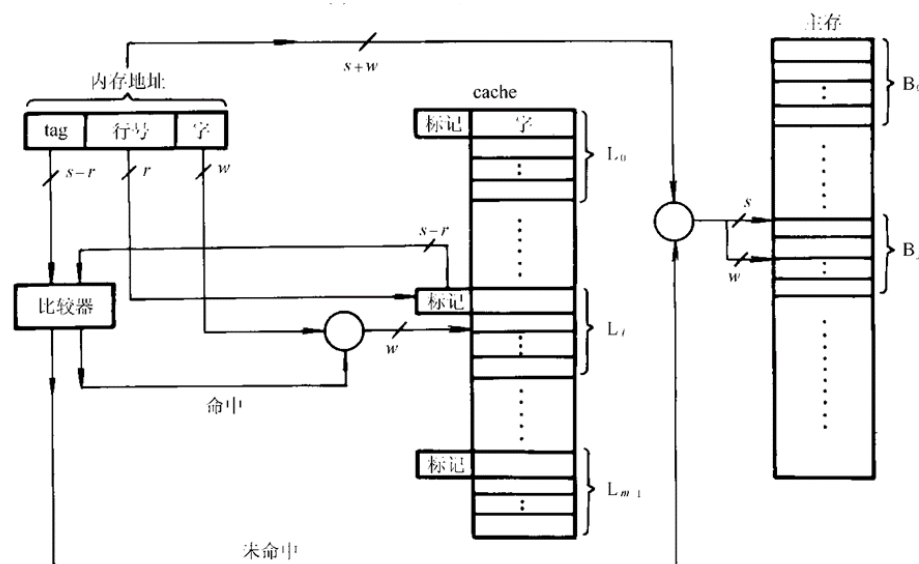


内存地址

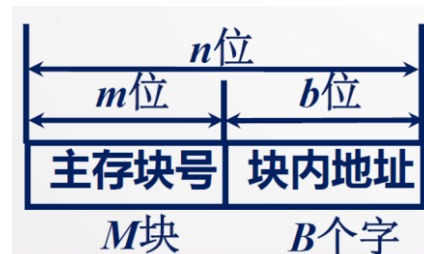


地址映射方式：根据行号进行寻址

- 特点
 - 优点：比较电路少，硬件实现简单，命中时间短，Cache地址为主存地址的低几位，不需变换
 - 缺点：冲突概率高
- 应用场合
 - 适合大容量Cache
 - 散列范围更广
 - 从而减少冲突概率

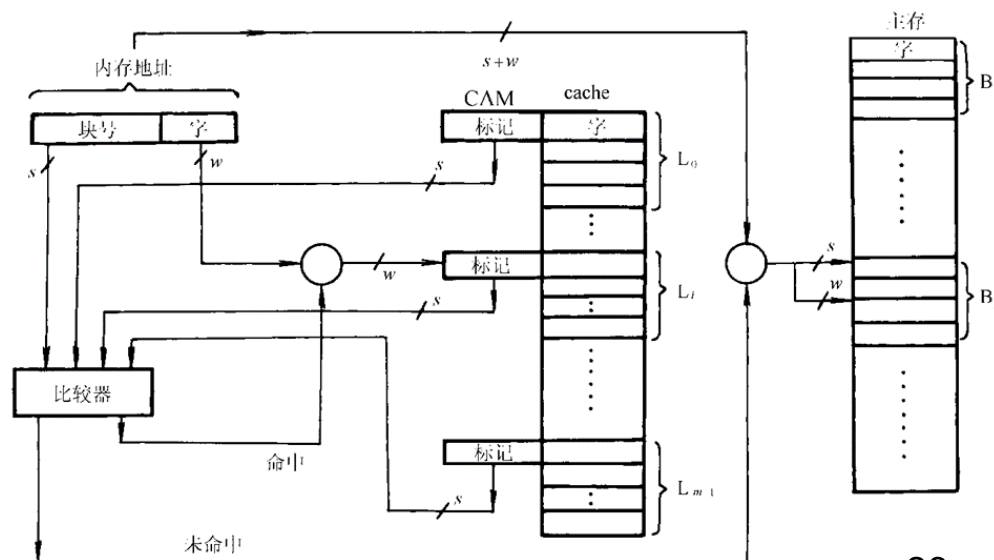


2 全相联地址映射



地址映射方式：与内存位置无关，对比所有标记与主存块号

- 特点：
 - 优点：冲突概率小，Cache的利用高。
 - 缺点：比较器难实现，需要一个访问速度很快代价高的相联存储器
- 应用场合：
 - 适用于小容量Cache



3 组相联地址映射

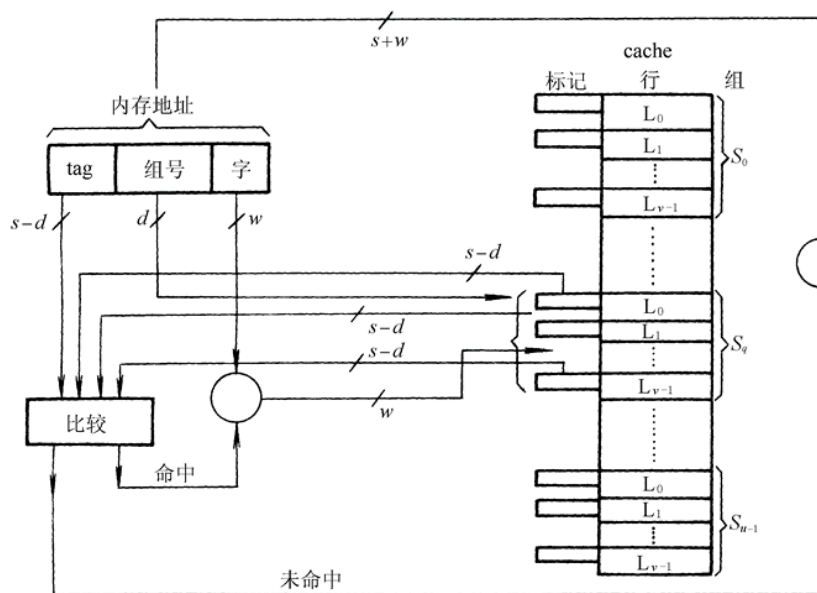
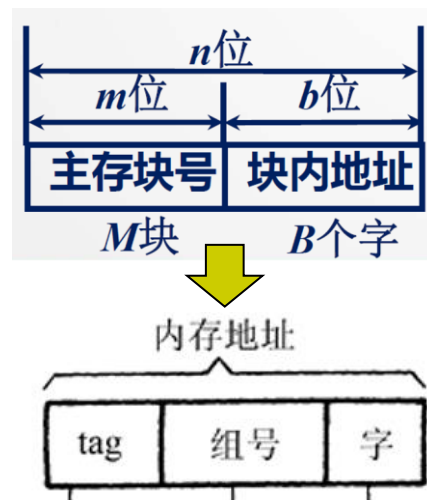
地址映射方式：根据组号进行寻址

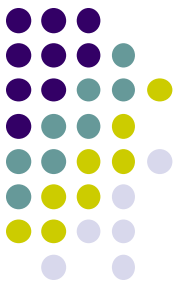
- 特点

- 优点：比全相联容易实现，冲突低
- 缺点：随组数增加更为复杂

- 组成结构

- $U \times V$ 数组 (mod U)
- V 一般为2的幂
- $V=1$ ，直接映射
- $U=1$ ，全相联





Cache地址映射 (步骤)

- 求解方法

- 确定字号 w

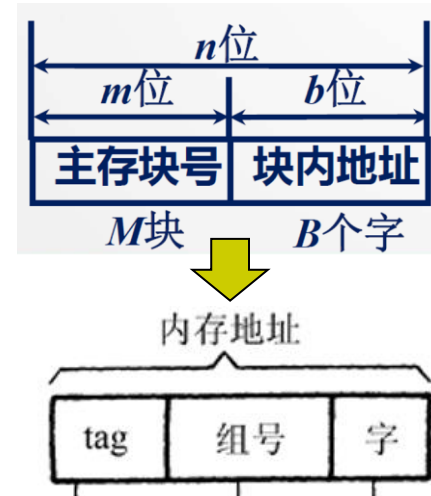
- 块内偏移量?
- 数据块 $\rightarrow X$ 字, $X=2^w$

- 确定行号 r

- 总共多少行?
- 直接映射N/组相联U, $N/U=2^r$

- 确定标记 $s-r$

- 多少个数据块被映射到同一行?
- 计算方法: 内存总数据块地址位 s -行号 r
- 内存大小 $\rightarrow Y$ 数据块, $2^s=Y$





地址映射例题 (2)

例8：一个组相联cache由64个行组成，每组4行。
主存包含4K个块，每块128字。请表示内存地址的格式

块大小 = 行大小 = $128 = 2^7$ $\therefore w = 7$

每组的行数 $k = 4$

组号 = $64/4=16=2^4$ $\therefore d = 4$

主存总体块数： $4K = 2^{12}$ $\therefore s = 12$

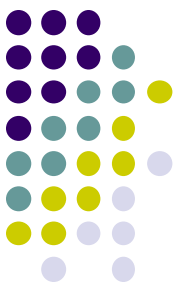
标记大小 $(s-d)$ 位 = $12-4 = 8$ 位

8位	4位	7位
标记 $s-d$	组号 d	字号 w



地址映射方式例题

- 主存容量1MB，字长8位，块大小16B，Cache容量64KB
- 1) 采用**直接映射**，给出[F0010H]对应标记为 [填空1]、行号为[填空2]、字号为 [填空3]。
- 2) 采用**二路组相联映射**，给出[F0010H]对应标记为 [填空1]、行号为[填空2]、字号为 [填空3]。
- 3) 采用**全相联映射**，给出[F0010H]对应标记为 [填空1]、字号为 [填空2]。



地址映射方式例题

- 主存容量1MB，字长8位，块大小16B，Cache容量64KB
- 1) 采用直接映射，给出[F0010H]对应标记为 [填空1]、行号为[填空2]、字号为 [填空3]。
 - 确定字号 w
 - 数据块 $\rightarrow 16$ 字， $X=2^w \quad \therefore$ 字号 $w = 4$
 - 确定行号 r
 - 行数 $N=64\text{KB}/16\text{B}=2^{12} \quad \therefore$ 行号 $r = 12$
 - 确定标记 $s-r$
 - 内存大小 $1\text{MB}/16\text{B}=2^{16} \quad \therefore s = 16$
 - 标记位数： $s-r=4$
 - F0010H=

1111

0000 0000 0001

0000

标记 行号 字号



- 32



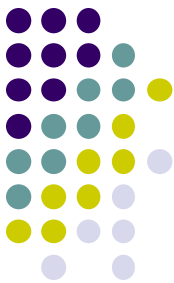
地址映射方式例题

- 主存容量1MB，字长8位，块大小16B，Cache容量64KB
- 3) 采用全相联映射，给出[F0010H]对应标记为 [填空1]、字号为 [填空2]。
 - 确定字号 w
 - 数据块 \rightarrow 16字， $X=2^w \quad \therefore$ 字号 $w = 4$
 - 确定标记（行号 $r=0$ ）
 - 内存大小 $1\text{MB}/16\text{B}=2^{16} \quad \therefore s = 16$
 - F0010H=

1111	0000	0000	0001
------	------	------	------

0000

标记字号



扩展：思考硬件—软件关系

□ 思考在Cache直接映射方式下，通过二重循环访问二维数组，如果内外层循环变量不同，对执行速度有何影响。

```
for(i=0; i<256; i++)  
{  
    for(j=0; j<256; j++)  
    {  
        sum += a[i][j];  
    }  
}
```

方案1

```
for(j=0; j<256; j++)  
{  
    for(i=0; i<256; i++)  
    {  
        sum += a[i][j];  
    }  
}
```

方案2



SRAM存储器与Cache

- SRAM存储器结构与操作
- SRAM实例：Cache
 - Cache基本原理
 - Cache地址映射
 - Cache读写操作
 - Cache替换策略



Cache主体——SRAM矩阵

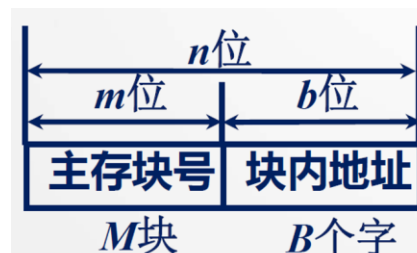
- 组织方式

- 按行组织

- 行数：16行

- 列数：数据16字节+有效位+标签

内存16bit: [2011H]



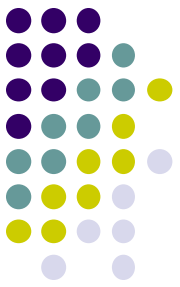
		有效位	标签	数据						
表项0		0								
表项1		0								
表项2		0								
表项3		0								
表项4		0								
表项5		0								
.....		0								
行号	表项15	0	标签	字节0	字节1	字节2	字节3	字节15	字号



Cache读操作 (1)

- 读操作程序示例 ↗ 行号1+标签20H
 - MOV AL, [2011H] 1. 未命中, 读2010H, 分配表项1
 - MOV BL, [4011H]
 - MOV DL, [401FH]

	有效位	标签	数据					
表项0	0							
表项1	0							
表项2	0							
表项3	0							
表项4	0							
表项5	0							
.....	0							
表项15	0		字节0	字节1	字节2	字节3	字节15



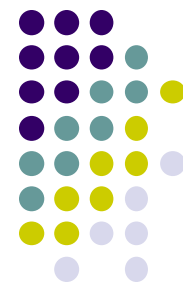
Cache读操作 (2)

- 读操作程序示例 ↗ 行号1+标签20H
 - MOV AL, [2011H] 1. 未命中, 读2010H, 分配表项1
 - MOV BL, [4011H] 2. 未命中, 读4010H, 替换表项1
 - MOV DL, [401FH]


表项0		有效位	标签	数据				
		0						

表项0		有效位	标签	数据				
		0						
表项0	1	20H	A0H	A1H	A2H	A3H	AFH
表项1	0							
表项2	0							
表项3	0							
.....								
表项15	0		字节0	字节1	字节2	字节3	字节15

表项15	0		字节0	字节1	字节2	字节3	字节15
------	---	--	-----	-----	-----	-----	-------	------



Cache读操作 (3)

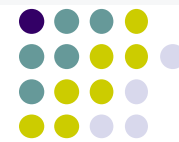
- 读操作程序示例  行号1+标签20H
 - MOV AL, [2011H] 1. 未命中, 读2010H, 分配表项1
 - MOV BL, [4011H] 2. 未命中, 读4010H, 替换表项1
 - MOV DL, [401FH] 3. 命中, 读4010H

		有效位	标签	数据				
表项0		0						
表项1		1	40H	B0H	B1H	B2H	B3H	BFH
表项2		0						
表项3		0						
.....								
表项15		0		字节0	字节1	字节2	字节3 字节15
.....								
表项15		0		字节0	字节1	字节2	字节3 字节15
表项15		0		字节0	字节1	字节2	字节3 字节15



Cache写操作策略

- Cache的内容只是主存部分内容的拷贝，它应当与主存内容保持一致。可选用如下三种写操作策略
 - **写回法**（效率高、控制复杂）
 - 命中时，只修改Cache内容
 - 替换时，写回至主存（判断是否修改）
 - **全写法**（效率低、控制逻辑简单）
 - 无论是否命中，同时修改Cache与内存
 - 写操作时延 = 访问主存时延
 - **写一次法**（混合方法）
 - 结合写回与全写法，与写回法基本一致，第一次命中时采用全写法；奔腾CPU策略



有如下程序段，Cache采用写回法，16*16字节结构

MOV [2011H], AL

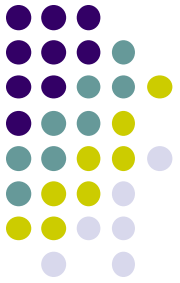
MOV [2011H], BL (AL ≠ BL)

MOV DL, [4011H]

执行第3条语句，Cache的操作包括

	有效位	标签	数据							
表项0	0									
表项1	0									
表项2	0									
表项3	0									
表项4	0									
表项5	0									
.....	0									
表项15	0		字节0	字节1	字节2	字节3	字节15		

- ☐ A 写[2011H]单元，读[4011H]单元至表项1
- ☒ B 写[2011H]单元，替换[4010H]数据块至表项1
- ☐ C 替换[4010H]数据块至表项1
- ☐ D 写[2011H]单元，替换[4011H]单元至表项40H



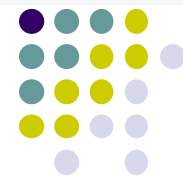
SRAM存储器与Cache

- SRAM存储器结构与操作
- SRAM实例：Cache
 - Cache基本原理
 - Cache读写操作
 - Cache地址映射
 - Cache替换策略

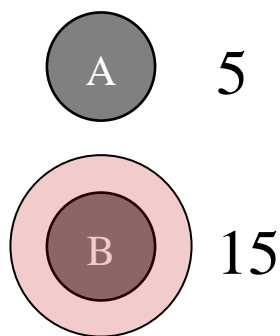


Cache替换策略（读未命中）

- LFU（最不经常使用，Least Frequently Used）
 - 被访问的**行计数器**增加1，换值小的行，不能反映近期Cache的访问情况（**计数器 = 最近访问次数/频率**）
- LRU（近期最少使用，Least Recently Used）
 - 被访问的**行计数器**置0，其他的计数器增加1，换值大的行，符合Cache的工作原理（**计数器 = 上次访问时间间隔**）
- 随机替换
 - 从特定的行位置中随机地选取一行换出即可
 - 这种策略在硬件上容易实现，且速度也比前两种策略快
 - 缺点是随意换出的数据很可能马上又要使用，从而降低命中率和Cache工作效率



二路组相联Cache采用LRU替换算法
某次读Cache未命中，需进行替换操作
对应组内块计数器计数器值分别为：5、15
Cache此时选择替换掉块计数器值为？



- LFU
 - 被访问的行计数器增加1，换值小的行，不能反映近期Cache的访问情况
- LRU
 - 被访问的行计数器置0，其他的计数器增加1，换值大的行，符合Cache的工作原理

总结

