处理器体系结构

第五章 高速缓存存储器B

--Cache性能的评价和提高

(Micro-processor Architecture)

目录

5.2 Cache性能的评价和提高

5.2.1 Cache性能对CPU性能的影响

5.2.2 通过灵活的映射减少缺失

5.2.1 Cache性能对CPU性能的影响

CPU执行时间由两部分构成：CPU执行程序的时间、存储器阻塞时间（通常认为CPU访问Cache命中的时间是CPU执行程序时间的一部分）

CPU执行时间=（CPU执行程序周期数+存储器阻塞周期数）×时钟周期

存储器阻塞周期数=读阻塞周期数+写阻塞周期数

等待存储器周期数主要来自Cache缺失，因此：

读阻塞周期数=(读次数/指令数)×读缺失率×读缺失周期数

写阻塞周期数=(写次数/指令数)×写缺失率×写缺失周期数 + Write buffer stalls

其中，write buffer stalls可以通过增加 write buffer的深度来减少，所以通常可以忽略（如果不可忽略，则说明系统设计不好，需要更深的buffer或使用“写回”）

又因为在大多数“写直达”策略下，读写的缺失代价是一样的（都是从DRAM取block的时间），所以如果读写缺失率一致，则：

存储器阻塞周期数=(Cache访问次数/指令数)×缺失率×缺失周期数

5.2.1 Cache性能对CPU性能的影响

例1：一个程序的I-cache的缺失率为2%，D-cahce的缺失率为4%，假定处理器在没有任何存储器停顿时的CPI为2，发生缺失时处理时间为100个cycles，请问处理器从不发生cache miss性能是发生cache miss的性能的多少倍？ (假设load/store出现频率为36%)

解：设程序的指令条数为I，则发生指令缺失的时钟周期数为：

$$\text{Instruction miss cycles} = I \times 2\% \times 100 = 2 \times I;$$

发生数据缺失的时钟周期数为：

$$\text{Data miss cycles} = I \times 36\% \times 4\% \times 100 = 1.44 \times I$$

因此，总的存储器阻塞周期数为：

$$\text{memory stall cycles} = 1.44I + 2I = 3.44I$$

最终，不发生缺失和发生缺失所需的总周期数为：

不发生缺失的周期数=2I　　发生缺失的周期数=2I+3.44I=5.44I

性能比：5.44/2=2.72

若上面的例子中程序CPI由2改为1，请问存储器所占的时间比例是多少？

不发生缺失的周期数=I　　发生缺失的周期数=I+3.44I=4.44I

性能比：4.44

而对一个CPI = 2的处理器，其存储器占的时间比例为3.44/5.44=63%

对一个CPI=1的处理器，其存储器占的时间比例为3.44/4.44=77%

5.2.1 Cache性能对CPU性能的影响

例2：在例1例子基础上（假定处理器在没有任何存储器停顿时的CPI为2），时钟频率提高1倍，请问处理器提高1倍频率后的性能是原来不提高的多少倍？（在cache miss的情况下）

解：不论处理器的速度多快，存储器访存的绝对时间不会改变：这样原来访存的100个周期，由于时钟频率的提高就变成了200个周期。

$$\text{memory stall cycles} = [(2\% \times 200) + 36\% \times (4\% \times 200)] \times I = 6.88I$$

$$\text{总执行周期数} = 6.88I + 2I = 8.88I$$

$$\text{性能提高倍数：} \quad 5.44I/(8.88I/2) = 1.23$$

也就是说，在考虑存储器的情况下，时钟频率提高1倍后，其性能仅提高了1.2倍，并不是所想象的2倍

Amdahl定律：仅仅提高CPU速度（不论降低CPI还是提高频率）而不改善存储系统，则存储器阻塞的时间占比会增加

5.2.1 Cache性能对CPU性能的影响

频率越高，CPI越低，在评价处理器的性能时，忽视cache的行为，其风险就越大。

回忆：增加Cache有助于提高命中率

问题：为什么不做一个1G的Cache?

1. Cache一般做在片上，会增加成本
2. Cache过大会增加命中时间（先前的分析没有考虑）

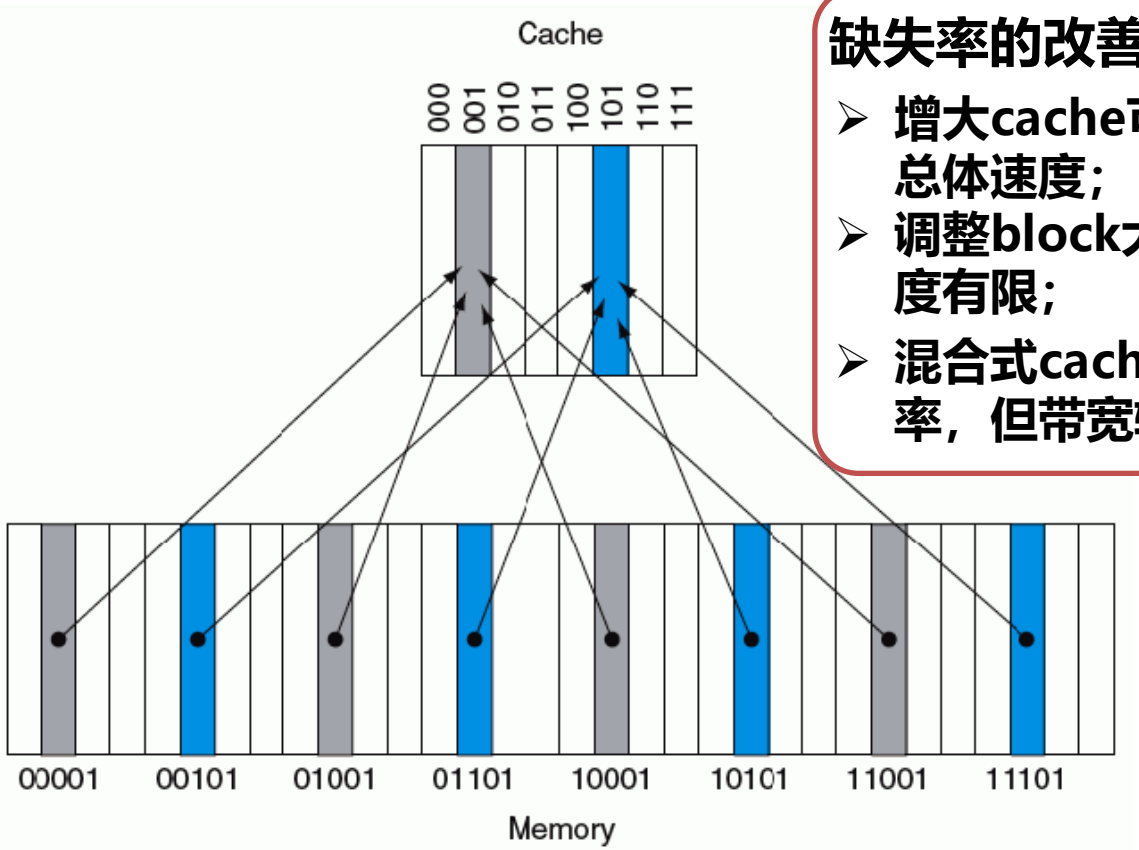
命中时间增加会导致时钟周期变长（降低频率），或增加流水深度（多个周期访问命中的Cache），复杂度进一步提升，最终可能降低处理器性能

->越少越快！

目录

5.2.2 通过灵活的映射减少缺失

回顾：**直接映射**，direct-mapped cache，一个块只能放在cache中一个位置



缺失率的改善措施：

- 增大cache可以降低缺失率，但可能反而降低总体速度；
- 调整block大小可以改善缺失率，但改善的幅度有限；
- 混合式cache比分离式cache具有更低的缺失率，但带宽较小，且不利于流水线，得不偿失；

问题：很容易产生缺失，比如交替访问00001和01001时会一直缺失

->有没有新的映射方法，降低2个block争夺cache中的同一个位置的概率?

5.2.2 通过灵活的映射减少缺失

全相联cache（fully associative cache）：

- **一个block可以放在cache中的任意位置；**
- **因为block放在任意位置，所以在找block时必须将地址与所有的tag对比**

优点：由于block可以放置在任意位置，所以不容易产生缺失

缺点：硬件成本过高，且会增加命中时间，只适合小容量的cache，实际处理器中很少使用

5.2.2 通过灵活的映射减少缺失

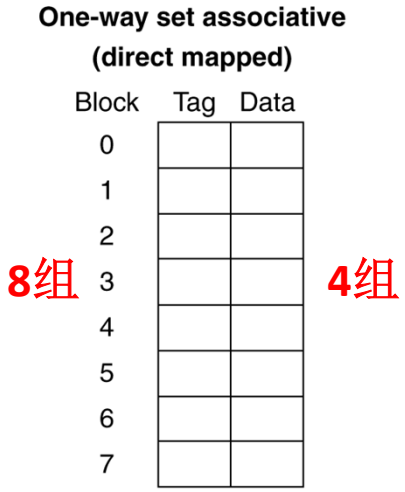
组相联cache（set associative cache）：

- **介于直接映射和全相联映射之间的一种方法**
- **先把cache中分成若干组，每组可有若干个位置（至少应为2）可放Block**
- **在组相联cache中，如果每组有n个位置可放Block，就称为n路组相联cache**

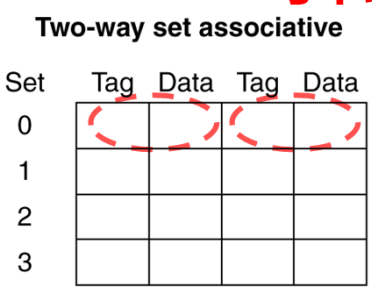
5.2.2 通过灵活的映射减少缺失

思考：我们可以把每一种放置方法看作是组相联的变体

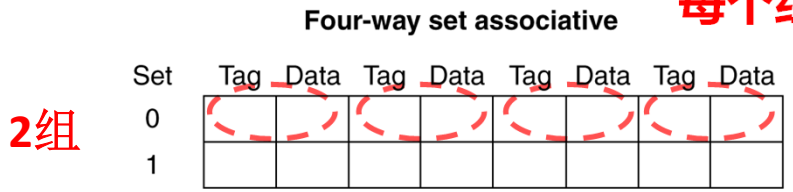
- **直接映射就是1路组相联映射，即每组仅有一个位置可放Block，每一个block就是一组**
- **全相联cache，相当于cache中仅有1组**



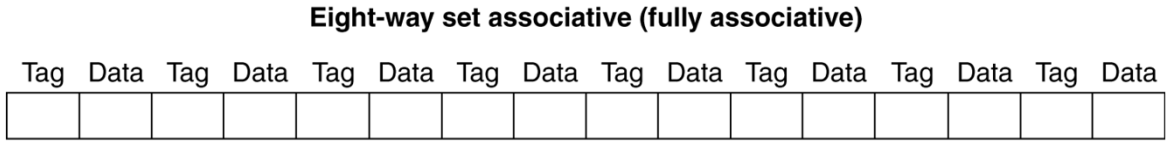
8组



4组



2组



1组

直接映射简单直观，但命中率低；全相联成本高，命中时间变长，适合小容量cache；

5.2.2 通过灵活的映射减少缺失

例：有1个cache，每个cache包含4个Block，每个Block包含一个word，若该cache采用全相联，2路组相联和直接映射来组织cache，请问当Blocks访问序列是0，8，0，6，8，会有多少miss和hit产生?

直接映射

访问地址
0 mod 4=0
8 mod 4=0
0 mod 4=0
6 mod 4=2
8 mod 4=0

1
2
3
4
5

Hit Or Miss	contents of cache blocks after reference			
	0	1	2	3
miss	Memory[0]			
miss	Memory[8]			
miss	Memory[0]			
miss	Memory[0]		Memory[6]	
miss	Memory[8]		Memory[6]	

- 发生5次Miss
- 蓝色表示添加
- 黑色表示老的项
- 空白格表示无数据

5.2.2 通过灵活的映射减少缺失

例：有1个cache，每个cache包含4个Block，每个Block包含一个word，若该cache采用全相联，2路组相联和直接映射来组织cache，请问当Blocks访问序列是0，8，0，6，8，会有多少miss和hit产生?

2路组相联

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

替换规则：**the least recently used block**

- 发生4次Miss,1次hit
- 蓝色表示添加
- 黑色表示老的项
- 空白格表示无数据

5.2.2 通过灵活的映射减少缺失

例：有1个cache，每个cache包含4个Block，每个Block包含一个word，若该cache采用全相联，2路组相联和直接映射来组织cache，请问当Blocks访问序列是0，8，0，6，8，会有多少miss和hit产生?

全相联

Block address		Hit/miss	Cache content after access			
0		miss	**Mem[0]**			
8		miss	Mem[0]	**Mem[8]**		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	**Mem[6]**	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

- 发生3次Miss,2次hit
- 蓝色表示添加
- 黑色表示老的项
- 空白格表示无数据

5.2.2 通过灵活的映射减少缺失

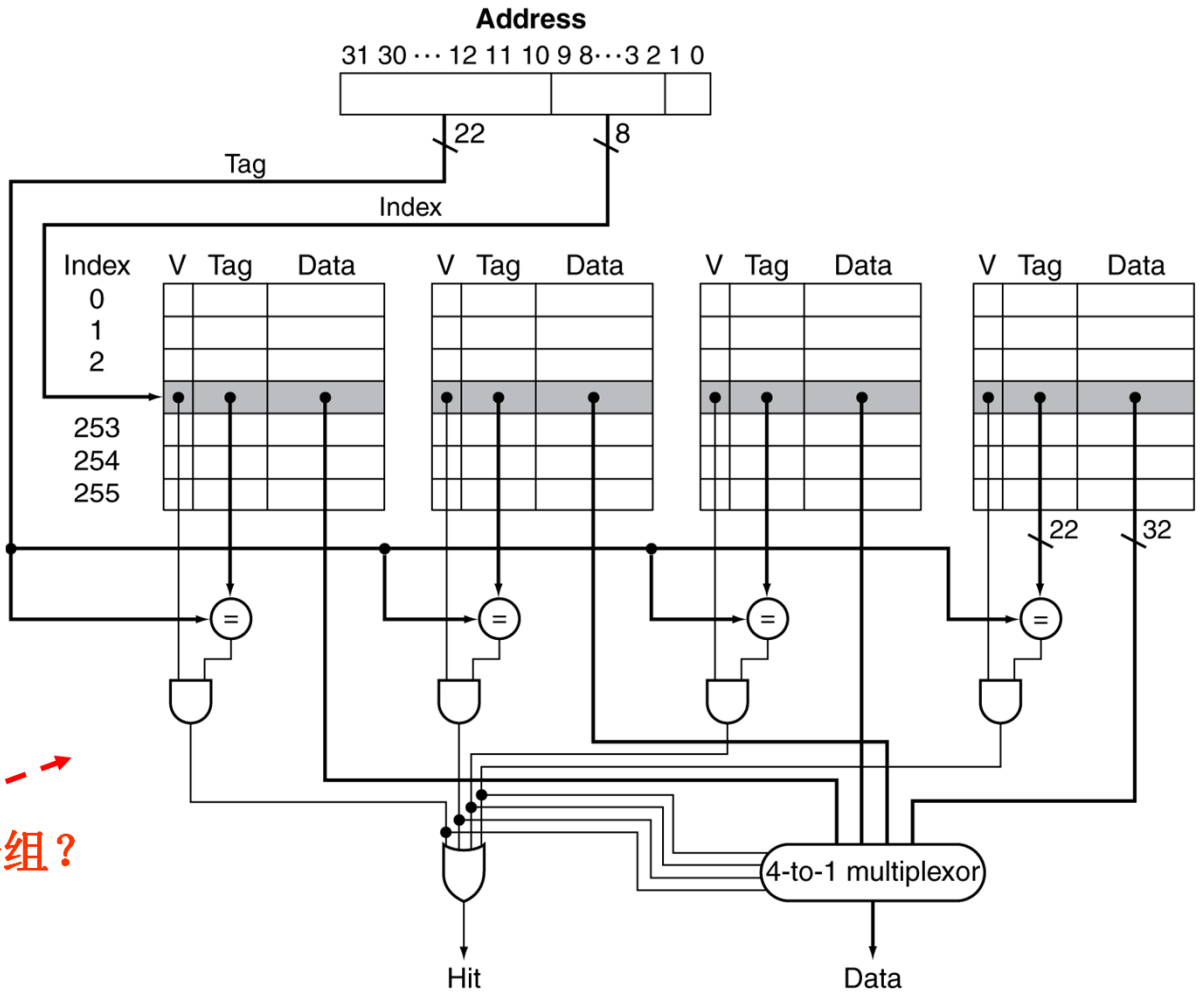
下图显示的是一个64KB data Cache，每个Block拥有16个word，关联映射从1路（直接映射）到8路组相联映射，运行的程序是SPEC2000 Benchmarks

Associativity	Data miss rate
1	10.3%
2	8.6%
4	8.3%
8	8.1%

表中可以看出从1路到2路，其缺失率从10.3%变到8.6%，缺失率降低了(10.3-8.6)/10.3=15%；而继续从2路到4路再到8路的组相联，性能改进微乎其微。

5.2.2 通过灵活的映射减少缺失

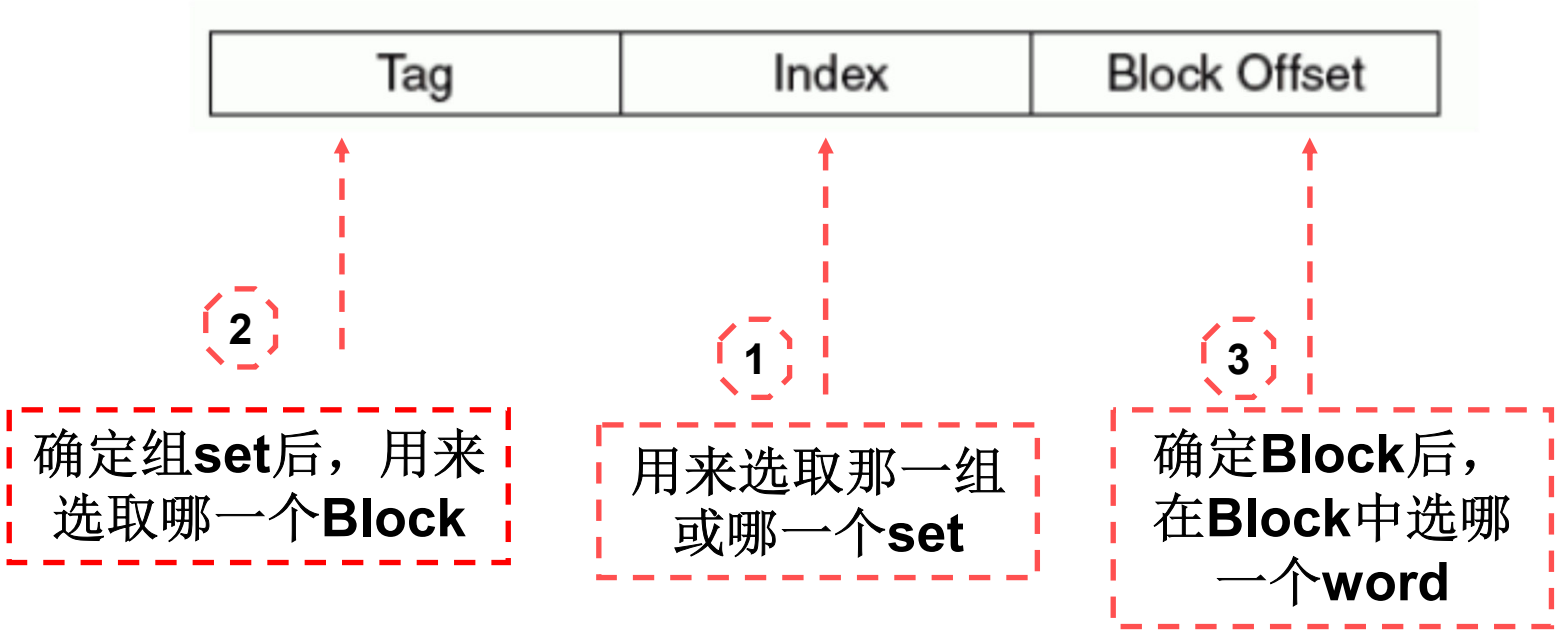
硬件实现：一个4路组相联cache的实现(每个Block有1个word)



请问有多少个组？

5.2.2 通过灵活的映射减少缺失

在Cache中定位Block



NOTE：确定了set或组后，每一个set中有若干个tag（一个tag/Block），set中所有的tag的比较一定是并行进行的

增加关联性意味着set中Block的数目增加，相应的就需要更多的比较器来进行tag的比较。

5.2.2 通过灵活的映射减少缺失

例：假定cache的大小为4K blocks，每个block有4个word，32位的地址，请问在直接映射，2路组相联，4路组相联和全相联情况下，set的数目和tag所占bit数为多少?

解：因为每个Block有4个word，每个word有4个字节，在Block中寻址字节需要4个地址位。也就是说，32位的地址，还剩（32 - 4）=28位被用来作为tag和index。

直接映射：Set的个数与Block的个数相同，（相当于1路组相联）。 所以set的数目为4K。而4K个Blocks，需要地址位12 bit 。因此，每个tag位长为28-12=16 位，总共的tag的比特数为16×4K = 64K bits

2路组相联：每2个Blocks组成一个set，因此Set的数目为4K/2=2K个。识别2K个sets地址位长就为 $\log_2(2\times2^{10})$ =11bits.这时，tag的位长为28-11=17 bits，总的tag的比特数为17×2×2K = 68Kbits

4路组相联：每4个Blocks组成一个set，因此Set的数目为4K/4=1K个。识别1K个sets地址位长为10bits.这时，tag的位长为28-10=18 bits，总的tag的比特数为18×4×1K = 72Kbits

全相联：整个cache只有一个set，set有4K个Blocks，这时，检索set的地址位为0，tag的位长为28bits，整个的tag比特数为28×4K×1 = 112K bits

5.2.2 通过灵活的映射减少缺失

选择哪一个Block替换?

- **直接映射的cache中，由于每一个位置只有一个Block，因此，发生miss时，当前的Block无条件的将被替换掉。**
- **在组相联的cache中，每个set中有多个Blocks，发生Miss时，需要利用一种替换规则来选择一个Block被替换**
- **在全相联的cache中，所有的Blocks都是被替换的候选对象，更需要替换的规则。**
  - **最常用的替换机制就是最近最少使用算法LRU（The Least Recently used ）**
  - **在2路组相联的cache中，通常设置一个bit位，用来指明哪一个block被最近引用或访问**

本章小结

5 高速缓存存储器

- 存储器组织结构
- Cache的基本结构
- 改善Cache性能
- 改善DRAM性能
- Cache性能对CPU性能的影响
- 通过灵活的映射减少缺失

*关于组相联的补充说明

Note：组相联有两种实现方法：

1、四段分法，数据地址构成：　　区号+组号+组内块号+块内地址

2、三段分法，数据地址构成：　　Tag+Index+Block offset

本课程教材、考试采用：2、三段分法

课后作业

若DRAM按字节编址，且Cache具有8个block，每个block具有一个word，进行两路组相联，初始内容为空，采用LRU方式替换，则当访问主存地址0,16,32,8,0,24,32,24,16,32时：

（1）上述十次访问的主存块号分别是多少？（格式：X_X_X_X_X_X_X_X_X_X）

（2）命中Cache的次数是?

(a)0　(b)1　(c)2　(d)3　(e)4　(f)5　(g)6　(h)7　(i)8　(j)9

（3）请以“XXXXXXXXXX”的格式给出上述十次访问cache的结果，用M表示Miss，H表示Hit，例如：MHHMMHHMMH