《计算机组成原理》第八次作业

网络空间安全学院 信息安全 陆皓喆 2211044

5.1

本题考察矩阵计算中存储器的局部特性。下面的代码用C语言编写，同一行中的元素连续存放。假定每个字是32位整数。

```
for (I=0; I<8; I++)
    for (J=0; J<8000; J++)
        A[I][J]=B[I][0]+A[J][I];
```

5.1.1

题目

16字节的$cache$块中可以存放多少个32位的整数?

解答

可以存放2个32位的整数

5.1.2

题目

访问哪些变量会显示出时间局部性?

解答

访问I、J、$B[I][0]$会显示出时间局部性

5.1.3

题目

访问哪些变量会显示出空间局部性?

局部性同时受访问顺序和数据存放位置的影响。同样的计算也可以用下面的$MATLAB$语言编写，但与C语言代码不同的是，矩阵元素按列连续存放。

```
for I=1:8
    for J=1:8000
        A（I，J）=B（I，0）+A（J，I）；
    end
end
```

解答

访问我们的$A[I][J]$会显示出空间局部性

5.1.4

题目

存放全部将被访问的32位矩阵元素需要多少16字节的$cache$块?

解答

如果是C语言的话，就是32008

如果是$MATLAB$语言的话，就是32004

代码引用矩阵a中的$8 \times 8000 = 64000$个整数，每个16字节块有两个整数，我们需要32000个块。

代码还引用矩阵b的八行中的第一个元素。$Matlab$以列为主顺序存储矩阵数据;因此，所有8个整数都是连续的，并放在4个块中。C按行序存储矩阵数据;因此，每行的第一个RST元素位于不同的块中。

5.1.5

题目

访问哪些变量会显示出时间局部性?

解答

访问变量$A[J][I]$和$B[I][0]$会显示时间局部性

5.1.6

题目

访问哪些变量会显示出空间局部性?

解答

访问变量I、J、$B[I][0]$会显示出空间局部性

5.2

$cache$为处理器提供了一个高性能的存储器层次结构，因此十分重要。下面是一个32位存储器地址引用的列表，给出的是字地址。

$$0x03, 0xb4, 0x2b, 0x02, 0xbf, 0x58, 0xbe, 0x0e, 0xb5, 0x2c, 0xba, 0xfd$$

5.2.1

题目

已知一个直接映射$cache$，有16个块，块大小为1个字。对于每次访问，请标识出二进制地址、标记以及索引。假设$cache$最开始为空，那么请列出每次访问是命中还是缺失。

解答

字地址	二进制地址	标记	索引	命中/缺失
0x03	00000011	0	3	缺失
0xb4	10110100	b	4	缺失
0x2b	00101011	2	b	缺失
0x02	00000010	0	2	缺失
0xbf	10111111	b	f	缺失
0x58	01011000	5	8	缺失
0xbe	10111110	b	e	缺失
0x0e	00001110	0	e	缺失
0xb5	10110101	b	5	缺失
0x2c	00101100	2	c	缺失
0xba	10111010	b	a	缺失
0xfd	11111101	f	d	缺失

5.2.2

题目

已知一个直接映射$cache$，有8个块，块大小为2个字。对于每次访问，请标识出二进制地址、标记以及索引。假设$cache$最开始为空，那么请列出每次访问是命中还是缺失。

解答

字地址	二进制地址	标记	索引	命中/缺失
0x03	00000011	0	1	1 缺失
0xb4	10110100	b	2	0 缺失
0x2b	00101011	2	5	1 缺失
0x02	00000010	0	1	0 命中

字地址	二进制地址	标记	索引	命中/缺失
0xbf	10111111	b	7	1 缺失
0x58	01011000	5	4	0 缺失
0xbe	10111110	b	6	0 命中
0x0e	00001110	0	7	0 缺失
0xb5	10110101	b	2	1 命中
0x2c	00101100	2	6	0 缺失
0xba	10111010	b	5	0 缺失
0xfd	11111101	f	6	1 缺失

5.2.3

题目

对已知的访问优化$cache$的设计。这里有三种直接映射$cache$设计方案，每个容量都为8个字：

- $C1$块大小为1个字。
- $C2$块大小为2个字。
- $C3$块大小为4个字。

解答

字地址	二进制地址	标记	C1	C2	C3
0x03	00000011	0x00	3 缺失	1 缺失	0 缺失
0xb4	10110100	0x16	4 缺失	2 缺失	1 缺失
0x2b	00101011	0x05	3 缺失	1 缺失	0 缺失
0x02	00000010	0x00	2 缺失	1 命中	0 缺失
0xbf	10111111	0x17	7 缺失	3 缺失	1 缺失
0x58	01011000	0x0b	0 缺失	0 缺失	0 缺失
0xbe	10111110	0x17	6 缺失	3 命中	1 命中
0x0e	00001110	0x01	6 缺失	3 缺失	1 缺失
0xb5	10110101	0x16	5 缺失	2 命中	1 缺失
0x2c	00101100	0x05	4 缺失	2 缺失	1 缺失
0xba	10111010	0x17	2 缺失	1 缺失	0 缺失
0xfd	11111101	0x1F	5 缺失	2 缺失	1 缺失

$Cache$ 1丢失率= 100%

$Cache$ 1总周期= $12\times25+12\times2=324$

$Cache$ 2丢失率=$10/12=83\%$

$Cache$ 2总周期= $10\times25+12\times3=286$

$Cache$ 3丢失率=$11/12=92\%$

$Cache$ 3总周期=$11\times25+12\times5=335$

所以$Cache$ 2性能最好。

5.4

题目

5.3节展示了对直接映射$cache$进行索引的典型方法，即（块地址）mod（$cache$中的块数）。假设有一个64位地址的$cache$，其中含有1024个块，考虑一种不同的索引方法：块地址的$[63:54]$位与块地址的$[53:44]$位进行异或运算。能否使用该方法来索引直接映射$cache$？如果可以，请解释原因并讨论$cache$可能需要的修改。如果不行，请解释原因。

解答

我们可以使用该方法来索引我们的直接映射。

为了实现直接映射的缓存，我们只需要一个函数，它将地址作为输入并产生一个10位的输出。尽管以这种方式实现缓存是可能的，但并不清楚这种实现是否有益，因为缓存需要更大的标签，而且可能会有更多的冲突遗漏。

5.5

对于一个32位地址的直接映射$cache$，下面的地址位用来访问$cache$。

标记	索引	偏移量
31~10	9~5	4~0

5.5.1

题目

$cache$块大小是多少（单位为字）？

解答

每个缓存块由四个8字节的字组成。总偏移量为5位。这5位中的3位是单词偏移集(将偏移集转换成8字节的单词)。剩下的两位是块偏移集。2位允许我们枚举$2^2=4$个$word$。

5.5.2

题目

$cache$有多少项?

解答

我们有5个索引位，这告诉我们在$cache$中有$2^5 = 32$行。

5.5.3

题目

这样的$cache$实现时所需的总位数与数据存储位数之间的比率是多少？下表记录了从上电开始的$cache$访问的字节地址。

十六进制	00	04	10	84	E8	A0	400	1E	8C	C1C	B4	884
十进制	0	4	16	132	232	160	1024	30	140	3100	180	2180

解答

总位数与数据存储位数之间的比率是1.21。缓存总共存储32行$\times$ 4个字/块$\times$ 8个字节/字=1024字节=8192位。

除数据外，每行包含54个标记位和1个有效位。因此，所需的总比特数=
$8192 + 54 \times 32 + 1 \times 32 = 9952$位

5.5.4

题目

对于每个访问，请列出：

- 标记、索引、偏移；
- 是否命中或缺失；
- 哪些字节被替换（如果有）。

解答

字节地址	二进制地址	标记	索引	偏移	命中/缺失	被替换的字节
0x00	0000 0000 0000	0x0	0x00	0x00	缺失	
0x04	0000 0000 0100	0x0	0x00	0x04	命中	
0x10	0000 0001 0000	0x0	0x00	0x10	命中	
0x84	0000 1000 0100	0x0	0x04	0x04	缺失	
0xe8	0000 1110 1000	0x0	0x07	0x08	缺失	
0xa0	0000 1010 0000	0x0	0x05	0x00	缺失	
0x400	0100 0000 0000	0x1	0x00	0x00	缺失	0x00-0x1F
0x1e	0000 0001 1110	0x0	0x00	0x1e	缺失	0x400-0x41F
0x8c	0000 1000 1100	0x0	0x04	0x0c	命中	
0xc1c	1100 0001 1100	0x3	0x00	0x1c	缺失	0x00-0x1F
0xb4	0000 1011 0100	0x0	0x05	0x14	命中	
0x884	1000 1000 0100	0x2	0x04	0x04	缺失	0x80-0x9f

5.5.5

题目

命中率是多少?

解答

$$\frac{4}{12} = 33\%$$

命中率为33%

5.5.6

题目

列出$cache$的最终状态，每个有效项表示为<索引，标记，数据>的形式。

解答

$cache$的最终状态是:

$< 0, 3, Mem[0xC00] - Mem[0xC1F] >$
$< 4, 2, Mem[0x880] - Mem[0x89f] >$
$< 5, 0, Mem[0x0A0] - Mem[0x0Bf] >$
$< 7, 0, Mem[0x0e0] - Mem[0x0ff] >$

5.6

回忆一下两个写策略和写分配策略，它们结合起来既可以在一级$cache(L1)$中实现，也可以在二级$cache(L2)$中实现。假定一级和二级$cache$的选择如下：

L1	L2
写直达，写不分配	写回，写分配

5.6.1

题目

在存储器层次结构中的不同层使用缓冲区（$buffer$）来降低访问延迟。对这个给定的配置，列出$L1\ cache$与$L2\ cache$之间，以及$L2\ cache$与存储器之间可能需要的缓冲区。

解答

$L1\ cache$的写丢失损失较小，而$L2\ cache$的写丢失损失较大。在$L1$和$L2$缓存之间设置一个$write\ buff$，可以隐藏$L2\ cache$的写缺失延迟。当替换一个$dirty\ block$时，$L2\ cache$将受益于$write\ buff$，因为新块将在$dirty\ block$物理写入内存之前被读入。

5.6.2

题目

描述处理$L1\ cache$写缺失的过程，考虑涉及的组件以及替换一个脏块的可能性。

解答

在$L1$写丢失时，字直接写入$L2$，而不将其块放入$L1\ cache$。如果这导致了$L2$丢失，那么它的块必须被带到$L2\ cache$中，可能取代脏块，脏块必须首先被写到内存中。

5.6.3

题目

对于一个多级不包含（$exclusive$）$cache$（一个块只能存放在一个$cache$层次中，要么在$L1\ cache$中，要么在$L2\ cache$中）配置，描述处理$L1\ cache$写缺失的过程，考虑涉及的组件以及替换一个脏块的可能性。

解答

在$L1$写失败后，该块将驻留在$L2$中，而不是$L1$中。在同一块上的后续读错误将要求将$L2$中的块写回内存，转移到$L1$，并在$L2$中失效。

5.10

本题将研究不同容量对整体性能的影响。通常来说,$cache$访问时间与$cache$容量成比例。假设访问主存需要$70ns$,并且在所有指令中有36%的指令需要访问数据存储器。下表是$P1$和$P2$两个处理器各自的$L1$ $cache$数据。

	L1 cache容量	L1 cache缺失率	L1 cache命中时间
$P1$	$2KB$	8.0%	$0.66ns$
$P2$	$4KB$	6.0%	$0.90ns$

5.10.1

题目

假定$L1$ $cache$的命中时间决定了$P1$和$P2$的周期时间,它们各自的时钟频率是多少?

解答

$P1$的时钟频率是$1.515GHz$

$P2$的时钟频率是$1.11GHz$

5.10.2

题目

$P1$和$P2$各自的$AMAT$(平均存储器访问时间)分别是多少?

解答

对于$P1$，所有内存访问至少需要一个周期(访问$L1$)。8%的内存访问额外需要$70ns$访问主存储器。这是$70/0.66 = 106.06$个周期。然而，我们不能分割周期;因此，我们必须四舍五入到107个周期。因此，平均内存访问时间为$1 + 0.08 \times 107 = 9.56$周期，即$6.31ps$。

对于$P2$，一次主存访问需要$70ns$。即$70/0.90 = 77.78$个周期。因为我们不能除以周期，所以我们必须四舍五入到78个周期。因此，平均内存访问时间为$1 + 0.06 \times 78 = 5.68$周期，即$6.11ps$。

综上所述:

我们的$P1$的$AMAT$是$6.31ns$或者9.56 $cycles$

我们的$P2$的$AMAT$是$5.11ns$或者5.68 $cycles$

5.10.3

题目

假定在没有任何存储器阻塞时基本的CPI为1.0,$P1$和$P2$各自的总CPI分别是多少?哪个处理器更快?(当我们说"基本CPI为1.0"时,是指指令在一个时钟周期完成,除非访问指令或数据时发生$cache$缺失。)
对下面三个问题,我们考虑在P1中增加$L2\ cache$,以弥补$L1\ cache$的容量限制。在解决这些问题时,依然使用上表中$L1\ cache$的容量和命中时间。$L2\ cache$缺失率是它的局部缺失率。

L2 cache容量	L2 cache缺失率	L2 cache命中时间
$1MB$	95%	$5.62ns$

解答

对于$P1$，每条指令至少需要一个周期。此外，8%的指令在指令缓存中丢失，并导致107个周期的延迟。

此外，36%的指令是数据访问。这36%中有8%是缓存丢失，这又增加了107个周期。

$$1+0.08\times 107+0.36\times 0.08\times 107=12.64$$

时钟周期为$0.66ps$，每条指令需要$8.34ns$。

使用相同的逻辑，我们可以看到$P2$的CPI为7.36，平均仅为$6.63ns$/指令。

综上所述:

$P1$的CPI是12.64，每条指令需要$8.34ns$

$P2$的CPI是7.36，每条指令需要$6.63ns$

5.10.4

题目

增加$L2\ cache$后,$P1$的$AMAT$是多少?有了$L2\ cache$,$AMAT$是更好还是更差了?

解答

$L2$访问需要9个周期($5.62/0.66$四舍五入到下一个整数)。

所有内存访问至少需要一个周期。8%的内存访问在$L1$缓存中丢失，并进行$L2$访问，这需要9个周期。95%的$L2$访问都是失败的，需要107个周期的内存查找。

$$1+0.08[9+0.95\times 107]=9.85$$

所以我们的$AMAT$应该是$9.85\ cycles$

$AMAT$应该是更差了

5.10.5

题目

假定在没有任何存储器阻塞时基本的CPI为1.0,增加$L2\ cache$后,$P1$的总的CPI是多少?

解答

我们可以这样计算:

$$AMAT + memory \times (AMAT - 1)$$

我们代入我们的数据，得到最后的结果:

$$the\ CPI\ for\ P1\ with\ an\ L2\ cache = 9.85 \times 0.36 \times 8.85 = 13.04$$

5.10.6

题目

要使有$L2\ cache$的$P1$比没有$L2\ cache$的P1更快，$L2\ cache$的缺失率应该是多少?

解答

因为两个版本$P1$的时钟周期时间和内存指令的百分比是相同的，所以只关注$AMAT$就足够了。

我们需要的是:

$$AMAT\ with\ L2 < AMAT\ with\ L1\ only$$
$$1 + 0.08[9 + m \times 107] < 9.56$$

当我们的$m < 0.916$的时候会发生

5.10.7

题目

要使有$L2\ cache$的$P1$比没有$L2\ cache$的$P2$更快，$L2\ cache$的缺失率应该是多少?

解答

我们希望$P1$的每条指令的平均时间小于$6.63ns$。这意味着我们需要:

$$(CPI\_P1 \times 0.66) < 6.63$$

因此，我们需要$CPI_P1 < 10.05$

我们已知:

$$CPI\_P1 = AMAT\_P1 + 0.36(AMAT\_P1–1)$$

因此我们需要:

$$AMAT\_P1 + 0.36(AMAT\_P1 - 1) < 10.05$$

这个情况发生在$AMAT_P1 < 7.65$

所以，我们最后需要解出：

$$1 + 0.08[9 + m \times 107] < 7.65$$

解出来是：

$$m < 0.693$$

这就是说，我们的缺失率最大是69.3%

5.11

本题研究了不同$cache$设计的效果，特别比较5.4节中的相联$cache$和直接映射$cache$。对于这些练习题，参照下面的字地址序列：

$$0x03, 0xb4, 0x2b, 0x02, 0xbe, 0x58, 0xbf, 0x0e, 0x1f, 0xb5, 0xbf, 0xba, 0x2e, Oxce$$

5.11.1

题目

请给出一个三路组相联、块大小为2个字、总容量为48个字的$cache$的大体组织结构。你给出的大体结构应类似于图$5-18$，但请明确给出标记和数据字段的宽度。

解答

缓存中的每一行总共有六个单词(三种方式各两个)。总共有$48/6 = 8$条线。

5.11.2

题目

跟踪练习题$5.11.1$中$cache$的行为，假设采用真正的LRU替换策略。对于每次访问，请指出：

- 二进制字地址
- 标记
- 索引
- 偏移
- 该访问是命中还是缺失
- 每个访问被处理后$cache$每一路中的标记是什么

解答

字地址	二进制地址	标记	索引	偏移	命中/缺失	第一路	第二路	第三路
0x03	0000 0011	0x0	1	1	缺失	T(1)=0		

字地址	二进制地址	标记	索引	偏移	命中/缺失	第一路	第二路	第三路
0xb4	1011 0100	0xb	2	0	缺失	T(1)=0 T(2)=b		
0x2b	0010 1011	0x2	5	1	缺失	T(1)=0 T(2)=b T(5)=2		
0x02	0000 0010	0x0	1	0	命中	T(1)=0 T(2)=b T(5)=2		
0xbe	1011 1110	0xb	7	0	缺失	T(1)=0 T(2)=b T(5)=2 T(7)=b		
0x58	0101 1000	0x5	4	0	缺失	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5		
0xbf	1011 1111	0xb	7	1	命中	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5		
0x0e	0000 1110	0x0	7	0	缺失	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	
0x1f	0001 1111	0x1	7	1	缺失	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1
0xb5	1011 0101	0xb	2	1	命中	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1

字地址	二进制地址	标记	索引	偏移	命中/缺失	第一路	第二路	第三路
0xbf	1011 1111	0xb	7	1	命中	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1
0xba	1011 1010	0xb	5	0	缺失	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=2 T(5)=b	T(7)=1
0x2e	0010 1110	0x2	7	0	缺失	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=2 T(5)=b	T(7)=1
0xce	1100 1110	0xc	7	0	缺失	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=2 T(5)=b	T(7)=c

5.11.3

题目

请给出一个全相联、块大小为1个字、总容量为8个字的$cache$的大体组织结构。你给出的大体结构应类似于图$5-18$，但请明确给出标记和数据字段的宽度。

解答

结构大概是：标记字段为8位，为块地址宽度，内容为字长度（64或32位）

5.11.4

题目

跟踪练习题$5.11.3$中的$cache$行为，假设使用真正的LRU替换策略。对于每次访问，请指出：

- 二进制字地址
- 标记
- 索引
- 偏移

- 该访问是命中还是缺失
- 每个访问被处理后$cache$中的内容是什么

解答

因为这个缓存是完全关联的，并且只有一个单词块，所以没有索引，也没有偏移。因此，单词$address$相当于tag。

字地址	二进制地址	标记	缺失/命中	内容
0x03	0000 0011	0x03	缺失	3
0xb4	1011 0100	0xb4	缺失	3, b4
0x2b	0010 1011	0x2b	缺失	3,b4, 2b
0x02	0000 0010	0x02	缺失	3, b4, 2b, 2
0xbe	1011 1110	0xbe	缺失	3, b4, 2b, 2, be
0x58	0101 1000	0x58	缺失	3, b4, 2b, 2, be, 58
0xbf	1011 1111	0xbf	缺失	3, b4, 2b, 2, be, 58, bf
0x0e	0000 1110	0x0e	缺失	3, b4, 2b, 2, be, 58, bf, e
0x1f	0001 1111	0x1f	缺失	b4, 2b, 2, be, 58, bf, e, 1f
0xb5	1011 0101	0xb5	缺失	2b, 2, be, 58, e, 1f, b5, b5
0xbf	1011 1111	0xbf	命中	2b, 2, be, 58, e, 1f, b5, bf
0xba	1011 1010	0xba	缺失	2, be, 58, e, 1f, b5, bf, ba
0x2e	0010 1110	0x2e	缺失	be, 58, e, 1f, b5, bf, ba, 2e
0xce	1100 1110	0xce	缺失	58, e, 1f, b5, bf, ba, 23, ce

5.11.5

题目

请给出一个全相联、块大小为2个字、总容量为8个字的$cache$的大体组织结构。你给出的大体结构应类似于图$5-18$，但请明确给出标记和数据字段的宽度。

解答

标记字段为8位，为块地址宽度，内容为2个字长度（64或128位）

5.11.6

题目

跟踪练习题$5.11.5$中的$cache$行为，假设采用真正的LRU替换策略。对于每次访问，请指出：

- 二进制字地址
- 标记
- 索引
- 偏移
- 该访问是命中还是缺失
- 每个访问被处理后$cache$中的内容是什么

解答

因为这个缓存是完全关联的，所以没有索引。(按数据访问顺序显示的内容。顺序并不意味着物理位置)

Word Address	Binary Address	Tag	Offset	Hit/Miss	Contents
0x03	0000 0011	0x01	1	M	[2,3]
0xb4	1011 0100	0x5a	0	M	[2,3], [b4,b5]
0x2b	0010 1011	0x15	1	M	[2,3], [b4,b5], [2a,2b]
0x02	0000 0010	0x01	0	H	[b4,b5], [2a,2b], [2,3]
0xbe	1011 1110	0x5f	0	M	[b4,b5], [2a,2b], [2,3], [be,bf]
0x58	0101 1000	0x2c	0	M	[2a,2b], [2,3], [be,bf], [58,59]
0xbf	1011 1111	0x5f	1	H	[2a,2b], [2,3], [58,59], [be,bf]
0x0e	0000 1110	0x07	0	M	[2,3], [58,59], [be,bf] , [e,f]
0x1f	0001 1111	0x0f	1	M	[58,59], [be,bf], [e,f], [1e,1f]
0xb5	1011 0101	0x5a	1	M	[be,bf],[e,f], [1e,1f], [b4,b5]
0xbf	1011 1111	0x5f	0	H	[e,f], [1e,1f], [b4,b5], [ba,bb]
0xba	1011 1010	0x5d	0	M	[1e,1f], [b4,b5], [b3,bf], [ba,bb]
0x2e	0010 1110	0x17	0	M	[b4,b5], [b3,bf], [ba,bb], [2e,2f]

Word Address	Binary Address	Tag	Offset	Hit/Miss	Contents
0xce	1100 1110	0x67	0	M	[be,bf], [ba,bb], [2e,2f], [ce,cf]

5.11.7

题目

使用最近使用（$Most\ Recently\ Used, MRU$）替换策略重做练习题5.11.6

解答

按数据访问顺序显示的内容。顺序并不意味着物理位置。

Word Address	Binary Address	Tag	Offset	Hit/Miss	Contents
0x03	0000 0011	0x01	1	M	[2,3]
0xb4	1011 0100	0x5a	0	M	[2,3], [b4,b5]
0x2b	0010 1011	0x15	1	M	[2,3], [b4,b5], [2a,2b]
0x02	0000 0010	0x01	0	H	[b4,b5], [2a,2b], [2,3]
0xbe	1011 1110	0x5f	0	M	[b4,b5], [2a,2b], [2,3], [be,bf]
0x58	0101 1000	0x2c	0	M	[b4,b5], [2a,2b], [2,3], [58,59]
0xbf	1011 1111	0x5f	1	H	[b4,b5], [2a,2b], [2,3], [be,bf]
0x0e	0000 1110	0x07	0	M	[b4,b5], [2a,2b], [2,3], [e,f]
0x1f	0001 1111	0x0f	1	M	[b4,b5], [2a,2b], [2,3], [1e,1f]
0xb5	1011 0101	0x5a	1	H	[2a,2b], [2,3], [1e,1f], [b4,b5]
0xbf	1011 1111	0x5f	1	M	[2a,2b], [2,3], [1e,1f], [ba,bb]
0xba	1011 1010	0x5d	0	M	[2a,2b], [2,3], [1e,1f], [ba,bb]
0x2e	0010 1110	0x17	0	M	[2a,2b], [2,3], [1e,1f], [2e,2f]

Word Address	Binary Address	Tag	Offset	Hit/Miss	Contents
0xce	1100 1110	0x67	0	M	[2a,2b], [2,3], [1e,1f], [ce,cf]

5.11.8

题目

使用优化替换策略（例如缺失率最低的策略）重做练习题5.11.6

解答

因为这个缓存是完全关联的，所以没有索引。

Word Address	Binary Address	Tag	Offset	Hit/Miss	Contents
0x03	0000 0011	0x01	1	M	[2,3]
0xb4	1011 0100	0x5a	0	M	[2,3], [b4,b5]
0x2b	0010 1011	0x15	1	M	[2,3], [b4,b5], [2a,2b]
0x02	0000 0010	0x01	0	H	[2,3], [b4,b5], [2a,2b]
0xbe	1011 1110	0x5f	0	M	[2,3], [b4,b5], [2a,2b], [be,bf]
0x58	0101 1000	0x2c	0	M	[58,59], [b4,b5], [2a,2b], [be,bf]
0xbf	1011 1111	0x5f	1	H	[58,59], [b4,b5], [2a,2b], [be,bf]
0x0e	0000 1110	0x07	0	M	[e,f], [b4,b5], [2a,2b], [be,bf]
0x1f	0001 1111	0x0f	1	M	[1e,1f], [b4,b5], [2a,2b], [be,bf]
0xb5	1011 0101	0x5a	1	H	[1e,1f], [b4,b5], [2a,2b], [be,bf]
0xbf	1011 1111	0x5f	1	H	[1e,1f], [b4,b5], [2a,2b], [be,bf]
0xba	1011 1010	0x5d	0	M	[1e,1f], [b4,b5], [ba,bb], [be,bf]
0x2e	0010 1110	0x17	0	M	[1e,1f], [b4,b5], [2e,2f], [be,bf]

Word Address	Binary Address	Tag	Offset	Hit/Miss	Contents
0xce	1100 1110	0x67	0	M	[1e,1f], [b4,b5], [ce,cf], [be,bf]