

Homework：存储2

本此作业统一以 $K = 10^3, M = 10^6, G = 10^9$ 为计量单位。以后如果遇到类似的题目请提前查看/询问/确定这件事。

请直接用 Markdown 题目源文件填充答案，最后统一提交 PDF 格式，比如使用 Typora 导出。

T2

下面的表给出了一些不同的高速缓存的参数。你的任务是填写出表中缺失的字段。其中 m 是物理地址的位数，C 是高速缓存大小（数据字节数），B 是以字节为单位的块大小，E 是相联度，S 是高速缓存组数，t 是标记位数，s 是组索引位数，而 b 是块偏移位数。

m	C	B	E	S	t	s	b
32	1024	4	4	64	24	6	2
32	1024	32	2	16	23	4	5
32	2048	8	1	256	21	8	3

T3

假设我们有一个具有如下属性的系统:

- 内存是字节寻址的。
- 内存访问是对 1 字节字的（比如访问一个 char）。
- 地址宽 12 位。
- 高速缓存是两路组相联的($E=2$)，块大小为 4 字节($B=4$)，有 4 个组($S=4$)。

高速缓存的内容如下，所有的地址、标记和值都以十六进制表示:

组索引	标记	有效位	字节1	字节2	字节3	字节4
0	00	1	40	41	42	43
	83	1	FE	97	CC	D0
1	00	1	44	45	46	47

组索引	标记	有效位	字节1	字节2	字节3	字节4
	83	0	54	55	56	57
2	00	1	48	49	4A	4B
	40	0	21	22	23	24
3	FF	1	9A	D0	03	EE
	00	0	A1	A2	A3	A4

对于下面每个内存访问，当他们顺序执行时，指出高速缓存是否命中，如果命中且操作前的数可从已有信息判断，请给出。

操作	地址	命中	值（或未知）
读	0x834	miss	未知
写	0x836	miss	未知
读	0xFFFF	hit	EE

T4

仔细阅读下面的程序，根据条件回答下列各题：

- 地址宽度为 10
- 数组的起始地址为 0b0001000000（即二进制表示）
- Block size = 4 Byte，Set = 4，两路组相连（B = 4, S = 4, E = 2）
- 替换算法为 LRU (最近最少使用)

```
#define LENGTH 8
void clear44(char array[LENGTH][LENGTH]) {
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            array[i][j] = 0;
}
```

4.1.1

以上程序会发生几次缓存miss？

答：4次。每次访问 array[i][0] 有一次冷不命中，然后载入 array[i][0] ~ array[i][3]，后三次 hit。共 miss 4 次。

4.1.2

如果 LENGTH = 16，那么会发生几次缓存miss？

答：还是 4 次，cache 和 array 地址是对齐的，miss原因同上，只有 4 次冷不命中。

4.1.3

如果 LENGTH = 17，那么会发生几次缓存miss？

答：

- i = 0 时，array[0][0] 会 miss 一次，其余 hit。
- i = 1 时，array[1][0] 会 miss 一次，array[1][3] 会 miss 一次，共 2 次。
- i = 2 时，array[2][0] 会 miss 一次，array[2][2] 会 miss 一次，共 2 次。
- i = 3 时，array[3][0] 会 miss 一次，array[3][1] 会 miss 一次，共 2 次。

综上，共有 7 次 miss。

4.1.4

请画出在 LENGTH = 17 时，程序执行结束时 set0 和 set1 的高速缓存状态，假设一开始全空。

可以用 Array[0][0] ~ Array[0][3] 的形式表示 Data 段落，有效位为 0 的行留空，每个 Set 内的顺序无所谓

SetID	Tag	Data
0	0000 0110	array[1][15] ~ array[2][1]
0	0000 0111	array[2][14]~array[3][0]
1	0000 0111	array[3][1]~array[3][4]
1	0000 0110	array[2][2]~array[2][5]

4.2

修改条件为

- 地址宽度为 10
- 数组的起始地址为 0b0010000000（即二进制表示）

- Cache 的容量为 16 Byte , Block size = 4 Byte , 全相联
- 替换算法为 LRU

4.2.0

Tag 的位数是多少？

答：8 位。 $m = 10, s = 0, b = 2$, 得 $t = m - s - b = 8$ bits.

4.2.1

原始程序会发生几次缓存miss？

答：4 次。只有访问 `array[i][0]` 产生 4 次冷不命中。

4.2.2

如果 `LENGTH = 16` , 那么会发生几次缓存miss？

答：4 次。只有访问 `array[i][0]` 产生 4 次冷不命中。

4.2.3

如果 `LENGTH = 17` , 那么会发生几次缓存miss？

答：7次。

- $i = 0$ 时 , 一次冷不命中。
- $i = 1$ 时 , 2 次 , `array[1][0]` 一次 , `array[1][3]` 一次。
- $i = 2$ 时 , 2 次 , `array[2][0]` 一次 , `array[2][2]` 一次。
- $i = 3$ 时 , 2 次 , `array[3][0]` 一次 , `array[3][1]` 一次。

综上 , 共 7 次。

4.2.4

请画出在 `LENGTH = 17` 时 , 程序执行结束时的高速缓存状态 , 假设一开始全空。

可以用 `Array[0][0] ~ Array[0][3]` 的形式表示 Data 段落 , 有效位为 0 的行留空

SetID	Tag	Data
0	0010 1001	<code>array[2][2] ~ array[2][5]</code>
1	0010 1100	<code>array[2][14] ~ array[3][0]</code>

SetID	Tag	Data
2	0010 1101	array[3][1] ~ array[3][4]
3	0010 1000	array[1][15] ~ array[2][1]

附：

$$\begin{aligned} & \text{当 } j=0,1,2,3 \text{ 时 } [1][15] \sim [2][1] \checkmark \\ & \text{当 } j=0,1,2,3 \text{ 时 } [0][10] \sim [0][13] \checkmark \\ & \text{当 } j=0,1,2,3 \text{ 时 } [2][10] \sim [2][13] \checkmark \\ & \text{当 } j=0,1,2,3 \text{ 时 } [1][10] \sim [1][13] \checkmark \end{aligned}$$

$$\begin{aligned} & \frac{1 \times 1 + j}{4} \\ & \frac{1 \times 1 + j}{4} = 5 \rightarrow \text{block } 1 \\ & [2][10] \sim [2][13] \end{aligned}$$

$$\begin{aligned} & \frac{3 \times 1 + 0}{4} \\ & \frac{3 \times 1 + 0}{4} = 5 \rightarrow \text{block } 1 \\ & \frac{3 \times 1 + 0}{4} \end{aligned}$$

$$[2][10] \sim [2][13]$$

$$[1][10] \sim [1][13]$$

$$\begin{aligned} & \text{当 } j=0,1,2,3 \text{ 时 } [0][10] \sim [0][13] \checkmark \\ & \text{当 } j=0,1,2,3 \text{ 时 } [1][10] \sim [1][13] \checkmark \\ & \text{当 } j=0,1,2,3 \text{ 时 } [2][10] \sim [2][13] \checkmark \\ & \text{当 } j=0,1,2,3 \text{ 时 } [1][10] \sim [1][13] \checkmark \end{aligned}$$

$$\begin{aligned} & [2][10] \sim [2][13] \\ & [0][10] \sim [0][13] \leftarrow \text{block } 0 \leftarrow \text{LRU} \\ & [0][10] \sim [0][13] \leftarrow \text{block } 12 \\ & [1][10] \sim [1][13] \leftarrow \text{block } 4 \leftarrow \text{LRU} \\ & [2][10] \sim [2][13] \leftarrow \text{block } 13 \\ & [1][10] \sim [1][13] \leftarrow \text{block } 5 \leftarrow \text{LRU} \\ & [1][15] \sim [2][1] \rightarrow \text{block } 8 \end{aligned}$$