

作业（存储器、高速缓存、虚拟内存）

一、存储器

1. 存储器容量为 64MB，字长 64 位，体（bank）数 $m = 8$ ，分别用顺序方式和交叉方式进行组织。存储周期 $T = 100\text{ns}$ ，数据总线宽度为 64 位，总线周期 $\tau = 10\text{ns}$ 。问顺序存储器和交叉存储器的带宽（单位：位/秒）各是多少？

答：

信息总量： $q = 64 * 8 = 512 \text{ bit}$

顺序存储器用时： $t_1 = mT = 8 * 100\text{ns} = 800\text{ns} = 8 * 10^{-7}\text{s}$

交叉存储器用时： $t_2 = T + (m - 1)\tau = 100 + 7 * 10 = 170\text{ns} = 1.7 * 10^{-7}\text{s}$

顺序存储器带宽： $W_1 = q/t_1 = 512/(8 * 10^{-7}) = 6.4 * 10^8 \text{ bit/s}$

交叉存储器带宽： $W_2 = q/t_2 = 512/(1.7 * 10^{-7}) = 3.01 * 10^9 \text{ bit/s}$

2. 用 $64\text{K} \times 1$ 位的 DRAM 芯片构成 $256\text{K} \times 8$ 位的存储器。要求：
- （1）计算所需芯片数。
 - （2）若采用异步刷新方式，每单元刷新闻隔不超过 2ms ，则产生刷新信号的间隔是多少时间？

答：

- （1）位拓展需要 $8/1 = 8$ ，字拓展需要 $256/64 = 4$ ，所以总共需要 $8 * 4 = 32$ 枚芯片。
- （2） $64\text{K} \times 1$ 位为 256×256 矩阵，采用异步式刷新，则产生刷新信号的间隔是 $2\text{ms}/256 = 7.81\mu\text{s}$ 。

二、高速缓存

1. 直接映射：

为了将内存数据块放置到高速缓存中，可以将内存地址分几部分看待：

Tag bits	Index bits	Offset bits
----------	------------	-------------

Offset bits 是块内偏移，Index bits 用于标记该内存数据块在高速缓存中的组号 (set number)，剩下的 tag bits 是该数据块的标记。

如果高速缓存采用直接映射 (direct mapped) 采用写直达 (write through) 的更新策略，那么高速缓存中每一行包含的内容为：cache data block, tag, valid bit，不需要包含 dirty bit.

根据已知信息，请填写下表中的空格部分：

Address size(bits) 内存地址的长度, 寻址到 byte	Cache size	Block Size 数据块大小	tag bits tag 位数	Index bits Index 位数	Offset bits 块内偏移位数	Bits per row 每一行的总位数
16	4KB	4B	4	10	2	$32+4+1$ $= 37$
32	32KB	16B	17	11	4	$16*8+17+1$ $= 146$
32	64KB	16B	16	12	4	$16*8+16+1$ $= 145$
64	2048KB	128B	43	14	7	1068

2、组相联映射：

假设某计算机的主存地址空间大小为 64MB，采用字节编址方式。其 cache 数据区容量为 4KB，采用 4 路组相联映射方式、LRU 替换和回写 (write back) 策略，块大小为 64B。请问：

(1) 主存地址字段如何划分？要求说明每个字段的含义、位数和在主存地址中的位置。

答：

cache 的划分为： $4KB = 2^{12}B = 2^4 \text{组} \times 2^2 \text{行/组} \times 2^6 B/\text{行}$

主存地址字段高 16 位为 tag，中间 4 位为 index，最低 6 位为 offset。

主存空间划分为： $64MB = 2^{26}B = 2^{16} \text{组群} \times 2^4 \text{块/组群} \times 2^6 B/\text{块}$

(2) 该 cache 的总容量（不仅包括数据区容量）有多少位？

答：

cache 有 2^6 行，每行有 16 位的 tag，1 位有效位，1 位 dirty，2 位 LRU，数据 64B，总容量为 $64 \times (16+1+1+2+64 \times 8) = 34048$ 位。

3、代码分析与高速缓存的性能：

一个二路组相联映射的高速缓存（2-way associative cache）容量为 128 bytes，每个高速缓存块大小为 32 字节 (32 bytes per block)。long long 型数据的长度为 8 个字节 (8 bytes)。假定 table 数组的起始地址为 0x0。以下代码的高速缓存失效率(miss rate)为多少？

```
int i;
int j;
long long table[4][8];
for (j = 0; j < 8; j++)
    for (i = 0; i < 4; i++)
```

{ table[i][j] = i + j; }

答：

128B = 4 * 32B = 4 * 4 * 8 B, 高速缓存中可以存 16 个 long long 型数据。

table[4][8] 中的 32 个数据在主存中按行存储, 即一行 8 个数据相邻。

共有 32 次访存请求, 前 4 次访存将每行前 4 个数据写入高速缓存的一个块, 后 12 次访存都会命中, 接下来 4 次访存会依次替换高速缓存中的 4 个块, 替换为每行后 4 个数据, 最后 12 次访存都会命中。

所以高速缓存失效率为 $8/32 \times 100\% = 25\%$ 。

4、平均存储器访问时间 (Average Memory Access Time: AMAT)

AMAT 是内存访问的平均 (expected) 时间, 可以用以下公式来估算:

$AMAT = hit_time + miss_rate \times miss_penalty$

- Hit_time: cache hit 时, 访问 cache 所花的时间
- Miss_rate: 高速缓存的失效率
- miss penalty: 当发生 cache miss 时, 需要花的额外的访存时间, 所以一次 cache miss 需要花费 (hit_time + miss_penalty) 的时间

假设高速缓存系统的属性如下, 求 AMAT 是多少?

- a) L1\$ hits in 1 cycle (local miss rate 25%)
- b) L2\$ hits in 10 cycles (local miss rate 40%)
- c) L3\$ hits in 50 cycles (global miss rate 6%)
- d) Main memory hits in 100 cycles (always hits)

答：

$$T1 = 1 + 25\% \times T2$$

$$T2 = 10 + 25\% \times 40\% \times T3$$

$$T3 = 50 + 6\% \times T4$$

$$T4 = 100$$

$$AMAT = T1 = 4.9 \text{ cycle}$$

41. 局部失效率

对于某一级Cache来说：

局部失效率 = 该级Cache的失效次数/到达该级Cache的访存次数

42. 全局失效率

对于某一级Cache来说：

全局失效率 = 该级Cache的失效次数/CPU发出的访存总次数

Global miss rate 和 Local miss rate 的定义请参考如下描述：

Global miss rate：

- the fraction of references that miss some level of a multilevel cache
- misses in this cache divided by the total number of memory accesses generated by the CPU

Local miss rate – the fraction of references to one level of a cache that miss

$$\begin{aligned}\text{Local Miss rate L2\$} &= \text{L2\$ Misses} / \text{L1\$ Misses} \\ &= \text{L2\$ Misses} / \text{total_L2_accesses} \cdot \text{L2\$ local miss rate}\end{aligned}$$

三、虚拟存储器（Virtual Memory）

1、

程序中使用的内存地址是虚拟地址，一个虚拟地址（VA）可以看作两部分：虚页号、页内偏移（page offset），如下图中标示：

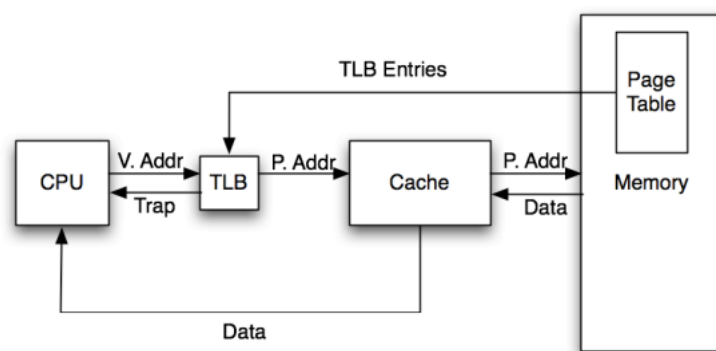
Virtual Page Number	Page Offset
---------------------	-------------

但事实上无论是数据还是指令都是存放在物理内存上的。一个物理地址（PA）也可以看作两部分：物理页号、页内偏移。如下图中标示：

Physical Page Number	Page Offset
----------------------	-------------

如果页大小是 4KB=4096 bytes，那么 page offset 就是 12 位。

将 VA 转换为 PA 会使用到快表（TLB）和页表（Page Table）。下图示意了 TLB 和页表（p 在内存访问时所处的位置。



每一个进程都有一个页表，页表存储在内存中，操作系统通过设置一个**专用寄存器**的值，告诉硬件**页表在内存中的起始地址**。每当切换进程时，操作系统会将要执行的进程的页表起始地址转载到这个专用寄存器中。

页表的结构一般如下：

The Page Table

Index = Virtual Page Number (VPN) (not stored)	Page Valid	Page Dirty	Permission Bits (read, write, ...)	Physical Page Number (PPN)
0				
1				
2				
...				
(Max virtual page number)				

每一个页表项（page table entry）除了记录虚页号（VPN）和物理页号（PPN）的映射关系之外，还设置了有效位、脏位和权限位：

- “page valid” 有效位：用于标记该虚页是否在内存中；
- “page dirty” 脏位：操作系统需要知道，是否将该内存页更新到磁盘上；
- “permission bits” 权限位：用于限制对该页进行某种操作。

快表（TLB）是页表的缓存（cache），假设 TLB 如果采用全相联映射（fully associative）机制，它的结构如下：

TLB Entry Valid	Tag = Virtual Page Number	Page Table Entry		
		Page Dirty	Permission Bits	Physical Page Number
...

回答问题：

1) 如果页表地址寄存器中装入了新的值，TLB 会发生什么操作？

答：

说明切换了进程，则页表会更新，原来 TLB 里的条目全部无效了。所以需要将 TLB 所有条目 Valid 位置为无效。

- 2) 某个处理器内存地址长度为 16 位，页大小为 256 byte，TLB 采用全相联映射，总共有 8 个 TLB 表项， 并采用 LRU 替换机制（LRU 位有 3 位，可以表示 8 个 TLB 表项的访问次序。如果 LRU 位的值为 0，表示该页最近刚刚被访问）。

假设当前进程初始时 TLB 的内容如下，并假设该进程访问的所有页既可以读也可以写。

Initial TLB

VPN	PPN	Valid	Dirty	LRU
0x01	0x11	1	1	0
0x00	0x00	0	0	7
0x10	0x13	1	1	1
0x20	0x12	1	0	5
0x00	0x00	0	0	7
0x11	0x14	1	0	4
0xac	0x15	1	1	2
0xff	0x16	1	0	3

假设现在空闲的物理页是：0x17, 0x18, 0x19;

如果接下来，用标记出的访问模式（读或者写）对以下内存地址进行访问：

Access pattern:

Read	0x11f0
Write	0x1301
Write	0x20ae
Write	0x2332
Read	0x20ff
Write	0x3415

请画出完成以上访问后，TLB 的 final state。

VPN	PPN	Valid	Dirty	LRU
0x01	0x11	1	1	5
0x13	0x17	1	1	3
0x10	0x13	1	1	6
0x20	0x12	1	1	1
0x23	0x18	1	1	2
0x11	0x14	1	0	4
0xac	0x15	1	1	7
0x34	0x19	1	1	0