高等计算机体系结构，2020年春季

作业5： Cache和Memory

主讲教师：栾钟治

助讲教师：杨海龙；助教：孙庆骁

作业下发时间：2020年5月08日

作业回收时间：2020年6月05日

# Cache 10分

下面给出了运行在带数据cache的处理器上的程序所生成的四种不同的地址序列，同时给出了每种序列的cache命中率。假设cache在每个序列开始时是空的，请回答该处理器数据cache的下述参数分别是多少：

(a) 相联度(1, 2 还是4路)

**答：4路**

**对于序列2来说，0, 512, 1024, 1536是这个序列中被重用的块，并且可能在第二次访问的时候命中块，在第二次访问的时候命中的次数为3次。假设块大小为8** ，**cache大小为256或者512，并且这些访问序列都映射到0组单元上，关联度为1或2时候，命中率均小于0.33，因此只有4成立。**

(b) 块大小(1, 2, 4, 8, 16 还是32 字节)

**答： 8字节**

**对于序列1来说6个访问只有2个中，那么假设块大小为8，命中率为33.3%成立。对于其他大小的块大小要么大于要么小于0.33，只有8成立。**

(c) cache总容量(256还是 512 字节)

**答：256字节**

**对于序列3无论替换策略是什么，假如是cache容量为512字节(以4路相联和8块大小来说)，那么命中率为4/9，所以cache容量为256字节成立。**

(d) 替换策略(LRU 还是 FIFO)

**答：LRU**

**假如是FIFO策略，那么命中率是3/8，而LRU策略命中率为1/4。**

假设：所有的访存都是单字节的访问，所有的地址都是字节地址。

|  |  |  |
| --- | --- | --- |
| 序列 | 地址序列 | 命中率 |
| 1 | 0, 2, 4, 8, 16, 32 | 0.33 |
| 2 | 0, 512, 1024, 1536, 2048, 1536, 1024, 512, 0 | 0.33 |
| 3 | 0, 64, 128, 256, 512, 256, 128, 64, 0 | 0.33 |
| 4 | 0, 512, 1024, 0, 1536, 0, 2048, 512 | 0.25 |

# 内存的交叉存取 20分

**2.1** 一台机器有4 KB的主存，由1个通道、1个rank和N(N>1)个bank构成。系统没有虚拟存储。

1) 数据采用cache块交叉存取策略进行交叉存取，即连续的cache块对应到连续的bank上；

2) cache块大小为32字节，bank的1行有128字节；

3) 采用打开行策略，即行缓冲中的行在被访问后继续保持在行缓冲中，直到有别的行被访问；

4) 行缓冲命中指访问的行存在于行缓冲中，行缓冲缺失指访问的行不在行缓冲中。

(a) 某个程序在这台机器上执行，访问以下字节时(数字表示字节的位置，比如320表示第320个字节)发生片上cache缺失而需要访存：0, 32, 320, 480, 4, 36, 324, 484, 8, 40, 328, 488, 12, 44, 332, 492，若行缓冲命中率为0，即所有访问的行都不在行缓冲中，请问bank数N的最小值是多少?

**答：当N=2时，bank中储存块的情况如下：(表格里面1表示一个cache块大小)**

**Bank 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | **4** | **6** | **8** |
| **10** | **12** | **14** | **16** |
| **18** | **20** | **22** | **24** |

**…**

**Bank 0**

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **3** | **5** | **7** |
| **9** | **11** | **13** | **15** |
| **17** | **19** | **21** | **23** |

…

**因此访问顺序为第1块->第2块->第11块->第16块->第1块->第2块->第11块->第16块…如此往复循环，从而导致行缓冲命中率为0。访问0时寻找bank 0中的1块，bank 0行缓冲未命中将第一行送入行缓冲，访问32时寻找bank 1中的2块， bank 1行缓冲未命中将第一行送入行缓冲，访问320时寻找bank 0中的11块，bank 0行缓冲未命中将第二行送入行缓冲，访问32时寻找bank 1中的16块，bank 1行缓冲未命中将第二行送入行缓冲，如此往复循环行缓冲命中率为0，所以bank数N的最小值是2。**

(b) 如果对于同一个序列，行缓冲命中率是75%，请问bank数N的最小值是多少?

**答：当N=4时，bank中储存块的情况如下：**

**Bank 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **4** | **8** | **12** | **16** |

**…**

**Bank 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **3** | **7** | **11** | **15** |

**…**

**Bank 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | **6** | **10** | **14** |

**…**

**Bank 0**

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **5** | **9** | **13** |

…

**因为1,2,11,16块分别处于不同的行中，由上一题分析可以知道，只有第一个周期行缓冲未命中，其余三个周期都是命中的，因此行缓冲命中率是75%， bank数N的最小值是4。**

(c) i) 对于同一序列，行缓冲的命中率能达到100%吗? 请解释原因

**答：可以的，只有四个高速缓存块被映射到不同的行，并且仅当包含每个缓存块的对应行已经打开时，行缓冲区的命中率才可以为100％。**

ii) 如果能达到, 最少需要多少bank才能够获得100%的行缓冲命中率?

**答：如果包含四个高速缓存块的四行已经打开（在四个存储区中的每个存储区），则4个存储区就足以实现此目的。**

**2.2** 一个DRAM主存储系统由1个通道、1个rank和N个bank构成。Bank一行256字节，一个cache块64字节。数据采用跨bank的行交叉存取方式组织，物理地址的分配方案如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 行 | Bank | 列 | BiB(Bytes in Block) |

采用打开行策略，即行缓冲中的行在被访问后继续保持在行缓冲中，直到有别的行被访问。初始时，所有bank的第1024行打开。

(a) 当有如下的cache块访问序列时，如果系统的行缓冲命中率为33.3% (即1/3)，请问系统中共有多少个bank:

0, 4, 8, 16, 32, 64, 128, 256, 128, 64, 32, 16, 8, 4, 0

**答：假如采用bank的行交叉存取方式组织，bank为32。**

**由256÷64=4可得一行有4个Cache块，因此考虑访问序列0, 4, 8, 16, 32, 64, 128, 256分别位于第1，2，3，5，9，17，33，65行，故不可能位于同一bank中的相同行，因此在第一次访问时不可能命中。因此后面的序列128，64，32，16，8，4，0中命中了5次，有两次未命中。若0，4，8，16，32，64，128，256所在行分别位于不同的bank中，那么128，64，32，16，8，4，0应当全部命中，现在出现了两次未命中，说明在0，4，8，16，32，64，128，256序列中，应当有且只有两次行缓冲在填充后修改的情况出现。**

**所以考虑以1，2，3，5，9，17，33，65为访问序列行数，那么假设N为bank数量，计算访问序列mod N之后结果相同出现两次。可知N=32成立，即以N为步长的等差数列与1，2，3，5，9，17，33，65的重合数字有且只有2个。**

**当N=32时，读取33行后替换bank 0 行缓冲中1，读取65行后替换bank 0 行缓冲中33，因此128，64，32，16，8，4，0序列只有128和0未命中，命中率即为5/15。**

**其余情况N=65-1=64,明显不成立，N=17-1=16,1-17-33-65均冲突也不成立，N<16以后的方案由于数列的冲突性，均无法满足要求。**

**答:假如采用块交叉存取策略（上课压根没讲块和行交叉存取区别）：**

**由256÷64=4可得一行有4个Cache块，因此考虑访问序列0, 4, 8, 16, 32, 64, 128, 256, 128, 64, 32, 16, 8, 4, 0。假设有N个bank块那么每个访问的块位于bank块的计算方式是X mod N = Y,Y出现相同的四次时候，bank块就要划分到下一行。**

**考虑N=1时，明显不正确。**

**N=2时，命中序列为~~0~~, 4, ~~8, 16, 32, 64, 128, 256, 128, 64, 32, 16, 8, 4~~, 0，准确率为2/15，不正确。**

**N=3时，命中序列为~~0, 4, 8, 16, 32, 64, 128, 256, 128, 64, 32, 16, 8, 4, 0~~，准确率为0/15，不正确。**

**N=4时，命中序列为~~0~~, 4, 8, ~~16, 32, 64, 128, 256, 128, 64, 32, 16, 8,~~ 4, 0，准确率为4/15，不正确。**

**Bank 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **3** | **7** | **11** | **15** |
| **19** | **23** | **27** | **31** |

**…**

**Bank 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | **6** | **10** | **14** |
| **18** | **22** | **26** | **30** |

**…**

**Bank 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **5** | **9** | **13** |
| **17** | **21** | **25** | **29** |

**…**

**Bank 0**

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **4** | **8** | **12** |
| **16** | **20** | **24** | **28** |
| **32** |  |  |  |

…

**N=5时，命中序列为~~0, 4, 8, 16, 32, 64, 128, 256,~~ 128, 64, 32, 16, ~~8, 4~~, 0，准确率为5/15，正确。简单来说，0, 4, 8, 16, 32位于不同的bank，64和128和前面4和8的bank的下面某行，因此替换时候会不命中。**

**Bank 4**

|  |  |  |  |
| --- | --- | --- | --- |
| **4** | **9** | **14** | **19** |
| **24** | **29** | **34** | **39** |

**…**

**Bank 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **3** | **8** | **13** | **18** |
| **23** | **28** | **33** | **38** |

**…**

**Bank 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | **7** | **12** | **17** |
| **22** | **27** | **32** | **37** |

**…**

**Bank 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **6** | **11** | **16** |
| **21** | **26** | **31** | **36** |

**…**

**Bank 0**

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** | **10** | **15** |
| **20** | **25** | **30** | **35** |
| **40** |  |  |  |

(b) 如果行缓冲命中率是7/15，请问系统中共有多少个bank?

**答：假如采用bank的行交叉存取方式组织，bank为128。**

**如上所示，0，4，8，16，32，64，128，256块分别位于第1，2，3，5，9，17，33，65行，故不可能位于同一bank中的相同行，因此在第一次访问时不可能命中。因此128，64，32，16，8，4，0中命中了7次，没有未命中。所以分析方法同上，以1，2，3，5，9，17，33，65为起始，以N为步长的等差数列与1，2，3，5，9，17，33，65的重合数字有且只有0个。所以保证序列对N求余数互不相等即可，那么8个数字8个余数要求N>8，因此：**

**N=9,65和2 mod 9相同,不成立。**

**N=10, 33和3 mod 9相同,不成立。**

**N=11,成立，余数都不相同。**

**所以系统至少要有11个bank。假如需要采用2次方数量，那么需要大于65-1=64的数量，及128个bank才可以。**

**答：假如采用块交叉存取策略：**

**上面已经考虑0同行下一位4，考虑8、16、32三种。**

**Bank 0：0-8-16-24情况不成立。**

**Bank 0：0-16-32-48，命中率6/15情况不成立。**

**Bank 0：0-32-64-96，命中率7/15情况，成立。因为0-32-64-96—128—256处于同一个bank中，所以命中率为(2+2+3)/15=7/15。**

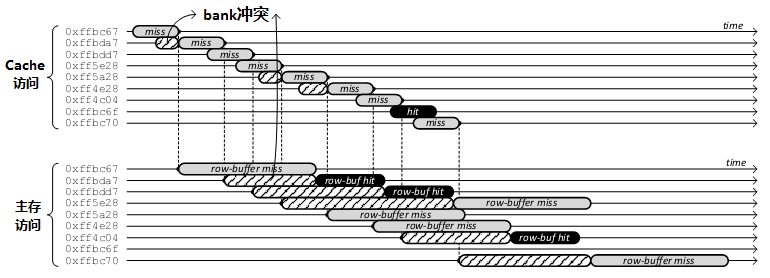
**所以bank有32个。**

# Bank 25分

一个处理器的分层存储结构由一个小的SRAM L1-cache和一个大的DRAM主存储器组成，SRAM和DRAM都被划分成bank。处理器有24位物理地址空间，并且不支持虚拟存储(即所有地址都是物理地址)。某个应用开始在这个处理器上运行，下图显示了在时间尺度上应用对存储系统引用的过程(包括在L1-cache和主存中)。

例如, 应用对存储第一次引用的字节地址是0xffbc67(假设所有的引用都是对按字节编址的内存地址的按字节读取)，但是这次引用在L1-cache中不命中 (假设L1-cache初始时为空)。紧接着，应用访问主存，这会经历一次行缓冲的不命中(初始时，假设主存的所有bank都打开一个永远不会被任何应用访问的行)。最后，包含字节地址0xffbc67的cache块 从主存取到cache中。

随后的内存引用可能会经历L1-cache和/或主存的bank冲突(当某个特定的bank还在提供之前的某个引用时)。



下表用16进制和2进制分别给出了该应用对存储系统引用的地址序列。

|  |  |
| --- | --- |
| 16进制 | 2进制 |
| ffbc67 | 1111 1111 1011 1100 0110 0111 |
| ffbda7 | 1111 1111 1011 1101 1010 0111 |
| ffbdd7 | 1111 1111 1011 1101 1101 0111 |
| ff5e28 | 1111 1111 0101 1110 0010 1000 |
| ff5a28 | 1111 1111 0101 1010 0010 1000 |
| ff4e28 | 1111 1111 0100 1110 0010 1000 |
| ff4c04 | 1111 1111 0100 1100 0000 0100 |
| ffbc6f | 1111 1111 1011 1100 0110 1111 |
| ffbc70 | 1111 1111 1011 1100 0111 0000 |

请分析上面的图和表，回答下列有关处理器上cache和主存的组织相关的问题，以下是一些假设：

1) L1-cache的假设

块大小: ? (2的幂, 大于2)

相联度: ? (2的幂, 大于2)

数据存储的大小: ? (2的幂, 大于2)

Bank数: ? (2的幂, 大于2)

初始时为空

2) 主存的假设

通道数: 1

每通道rank数: 1

每rank的bank数: ? (2的幂, 大于2)

每bank的行数: ? (2的幂, 大于2)

每行的cache块数: ? (2的幂, 大于2)

包含应用的整个工作集

初始时，所有的bank打开第0行，应用永远不会访问该行

注意: 对于以下问题，假设所有的偏移量和索引来自连续的地址位

**答：首先将上图转换成以下表格。**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 地址 | 十进制地址 | 相对地址 | Cache | Row buffer | 冲突 |
| 0xffbc67 | 16759911 | a | miss | miss | - |
| 0xffbda7 | 16760231 | a+320 | miss | hit | Cache&Mem |
| 0xffbdd7 | 16760279 | a+368 | miss | hit | Mem |
| 0xff5e28 | 16735784 | b | miss | miss | Mem |
| 0xff5a28 | 16734760 | b-1024 | miss | miss | Cache |
| 0xff4e28 | 16731688 | b-4096 | miss | miss | Cache |
| 0xff4c04 | 16731140 | b-4644 | miss | hit | Mem |
| 0xffbc6f | 16759919 | a+8 | hit | - | - |
| 0xffbc70 | 16759920 | a+9 | miss | miss | Mem |

(a) L1-cache的块大小是多少字节? 24-bit物理地址中哪些位是cache块偏移量? (物理地址的最低位为0位)

**答：假设块大小是16字节，那么16759911 mod 16 = 1047494 余数 7，因为访问地址16759911，拿取bank块是1047494且访问的是块中间的地址，所以把1047494 - 1047494 +16字节这个块放入Cache并且访问a+8是hit的状态，但是a+9却是miss的，因为已经超出块的范围了。**

**而假设块大小为8，由上面分析看出来是错误的。**

**且由上面可以分析出Cache采用的是因此L1-cache块大小为16字节，第0-3位是块内偏移量。**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 地址 | 块号 | 相对地址 | Cache | Row buffer | 冲突 |
| 0xffbc67 | 1047494 | a | miss | miss | - |
| 0xffbda7 | 1047514 | a+20 | miss | hit | Cache&Mem |
| 0xffbdd7 | 1047517 | a+23 | miss | hit | Mem |
| 0xff5e28 | 1045986 | a-1508=b | miss | miss | Mem |
| 0xff5a28 | 1045922 | b-64 | miss | miss | Cache |
| 0xff4e28 | 1045730 | b-256 | miss | miss | Cache |
| 0xff4c04 | 1045696 | b-290 | miss | hit | Mem |
| 0xffbc6f | 1047494 | a | hit | - | - |
| 0xffbc70 | 1047495 | a+1 | miss | miss | Mem |

**因为a+320，a+368均发生主存bank冲突，对比a,a+230是相隔20块，2+368是相隔23块，而题目已经说明了每行Cache块数大于2且为2的幂数，那么可以判断主存不是采用块交叉存取的方式，否则20和23块的命中是不会发生冲突的，因为位于不同的bank上互相不干扰。**

**b-64没有发生主存bank冲突，且b-256也没有发生主存bank冲突，两者位于不同的行，可以判断主存采用行交叉存取的方式，b-290超出行缓冲的范围之后发生主存冲突，因此可以确定主存采用行交叉存取。**

**因为a和a+20发生冲突的话Cache bank数应该为20的约数（因为此时a正处于被占用的状态），所以cache bank为4，如下所示a和a+20是冲突状态，且每行的cache块数为4也成立，后面的b,b-64,b-256发生cache冲突符合假设。所以我们可以推断cache是采用块交叉存储，因此4-5位为cache bank的索引位。**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **bank 0** | | | | **bank 1** | | | | **bank 2** | | | | **bank 3** | | | |
| **1** | **5** | **9** | **13** | **2** | **6** | **10** | **14** | **3** | **7** | **11** | **15** | **4** | **8** | **12** | **16** |
| **17** | **21** | **25** | **29** | **18** | **22** | **26** | **30** | **19** | **23** | **27** | **31** | **20** | **24** | **28** | **32** |

**因为在a的Row buffer没有命中后，a+20,a+23都命中了，所以主存bank每行的块数是大于23的，有因为b没有命中，b-64也没有命中，所以主存bank每行的块数小于等于64，再由b-256没有命中，b-290命中可以推断每行块数大于34，因此可以推断主存bank每行的块数为64。**

**由b,b-64,b-256是可以并行访存的（主存采用行交叉存取方式,没有引起主存bank冲突），而a,a-1508是不可以并行访存的，所以1508÷64 ≈ 24，所以主存的bank数大于4且是24约数，所以主存bank为8,** **10-12位为主存bank索引。**

(b) L1-cache有多少bank? 24-bit物理地址中哪些位是L1-cache的bank索引? (物理地址的最低位为0位)

**答：由上面分析可以得，L1-cache有4个bank，4-5位为cache bank的索引位。**

(c) 主存中有多少bank? 24-bit物理地址中哪些位是主存的bank索引? (物理地址的最低位为0位)

**答：由上面分析可以得，主存由8个bank,10-12位为主存索引位。**

(d) 物理地址向主存映射时用了什么样的交叉存取方案?

**答：行交叉存取方案。**

(e) 为了支持24-bit的物理地址空间，主存的每一个bank需要多少行? 24-bit物理地址中哪些位是主存的行索引? (物理地址的最低位为0位)

**答：主存bank一行有64×16=1024个字节，需要10个位编码，共有8个bank，需要3个位编码，故于11个位将用于编码行索引，即每一个bank需要2^24÷1024÷8=2048行，物理地址中的13-23总共11位用于行索引。**

(f) 在一行中的每个cache块被称为列，一行中有多少列? 24-bit物理地址中哪些位是主存的列索引? (物理地址的最低位为0位)

**答：一行中有64列，列索引为4-9位。**

# 内存调度 25分

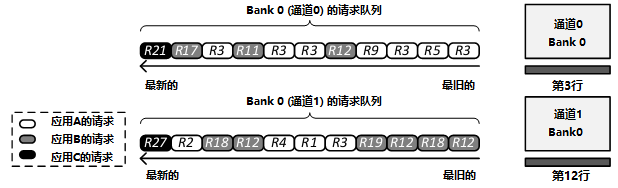
为了响应访存请求，内存控制器会发射1条或多条DRAM命令以从bank访问数据。有4种不同的 DRAM命令。

1) 激活(ACTIVATE): 取被访问的行装入bank的行缓冲。 这一操作也被称为打开行(延迟: 15ns)

2) 预充电(PRECHARGE): 将bank的行缓冲中的内容存回行。这一操作也被称为关闭行 (延迟: 15ns)

3) 读/写: 从行缓冲中访问数据(延迟: 15ns)

下图显示了在时刻t0时内存控制器中的内存请求缓冲的快照。每一个请求按照颜色的不同代表了其所属的不同应用(假设所有的应用运行在独立的核上)。同时，每个请求标注了它要访问的行地址(或索引)，例如R3表示请求的是第3行。另外，假设所有的请求都是读请求。



访存请求在读命令完成后被响应(即读命令被发射15ns之后)，每个应用(A、B或C)停顿直到它所有访存请求被响应为止。

假设初始时(t0时)，每个bank的第3行和第12行分别被取出并存入行缓冲，没有任何其他应用的请求到达内存控制器。

**4.1** 非应用感知的调度策略

(a) 使用先来先服务调度策略(FCFS)，每个应用的停顿时间是多少?

**通道0:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **开始时刻** | **访存序列** | **行缓冲** | **命中情况** | **结束时刻** | **DEAM命令** |
| **0** | **R3** | **R3** | **hit** | **15** | **读/写** |
| **15** | **R5** | **R3** | **miss** | **60** | **激活+预充电+读/写** |
| **60** | **R3** | **R5** | **miss** | **105** | **激活+预充电+读/写** |
| **105** | **R9** | **R3** | **miss** | **150** | **激活+预充电+读/写** |
| **150** | **R12** | **R9** | **miss** | **195** | **激活+预充电+读/写** |
| **195** | **R3** | **R12** | **miss** | **240** | **激活+预充电+读/写** |
| **240** | **R3** | **R3** | **hit** | **255** | **读/写** |
| **255** | **R11** | **R3** | **miss** | **300** | **激活+预充电+读/写** |
| **300** | **R3** | **R11** | **miss** | **345** | **激活+预充电+读/写** |
| **345** | **R17** | **R3** | **miss** | **390** | **激活+预充电+读/写** |
| **390** | **R21** | **R17** | **miss** | **435** | **激活+预充电+读/写** |

**通道1:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **开始时刻** | **访存序列** | **行缓冲** | **命中情况** | **结束时刻** | **DEAM命令** |
| **0** | **R12** | **R12** | **hit** | **15** | **读/写** |
| **15** | **R18** | **R12** | **miss** | **60** | **激活+预充电+读/写** |
| **60** | **R12** | **R18** | **miss** | **105** | **激活+预充电+读/写** |
| **105** | **R19** | **R12** | **miss** | **150** | **激活+预充电+读/写** |
| **150** | **R3** | **R19** | **miss** | **195** | **激活+预充电+读/写** |
| **195** | **R1** | **R3** | **miss** | **240** | **激活+预充电+读/写** |
| **240** | **R4** | **R1** | **hit** | **285** | **激活+预充电+读/写** |
| **285** | **R12** | **R4** | **miss** | **330** | **激活+预充电+读/写** |
| **300** | **R18** | **R12** | **miss** | **375** | **激活+预充电+读/写** |
| **375** | **R2** | **R18** | **miss** | **420** | **激活+预充电+读/写** |
| **420** | **R27** | **R2** | **miss** | **465** | **激活+预充电+读/写** |

**应用A在通道0中345时刻完成，在通道1中420时刻完成，因此应用A在t0+420时刻完成。**

**应用B在通道0中390时刻完成，在通道1中375时刻完成，因此应用A在t0+390时刻完成。**

**应用C在通道0中435时刻完成，在通道1中465时刻完成，因此应用A在t0+465时刻完成。**

(b) 使用行缓冲优先加先来先服务的调度策略(FR-FCFS)，每个应用的停顿时间是多少?

**通道0:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **开始时刻** | **访存序列** | **行缓冲** | **命中情况** | **结束时刻** | **DEAM命令** |
| **0** | **R3** | **R3** | **hit** | **15** | **读/写** |
| **15** | **R3** | **R3** | **hit** | **30** | **读/写** |
| **30** | **R3** | **R3** | **hit** | **45** | **读/写** |
| **45** | **R3** | **R3** | **hit** | **60** | **读/写** |
| **60** | **R3** | **R3** | **hit** | **75** | **读/写** |
| **75** | **R5** | **R3** | **miss** | **120** | **激活+预充电+读/写** |
| **120** | **R9** | **R5** | **miss** | **165** | **激活+预充电+读/写** |
| **165** | **R12** | **R9** | **miss** | **210** | **激活+预充电+读/写** |
| **210** | **R11** | **R12** | **miss** | **255** | **激活+预充电+读/写** |
| **255** | **R17** | **R11** | **miss** | **300** | **激活+预充电+读/写** |
| **300** | **R21** | **R17** | **miss** | **345** | **激活+预充电+读/写** |

**通道1:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **开始时刻** | **访存序列** | **行缓冲** | **命中情况** | **结束时刻** | **DEAM命令** |
| **0** | **R12** | **R12** | **hit** | **15** | **读/写** |
| **15** | **R12** | **R12** | **hit** | **30** | **读/写** |
| **30** | **R12** | **R12** | **hit** | **45** | **读/写** |
| **45** | **R18** | **R12** | **miss** | **90** | **激活+预充电+读/写** |
| **90** | **R18** | **R18** | **hit** | **105** | **读/写** |
| **105** | **R19** | **R18** | **miss** | **150** | **激活+预充电+读/写** |
| **150** | **R3** | **R19** | **miss** | **195** | **激活+预充电+读/写** |
| **195** | **R1** | **R3** | **miss** | **240** | **激活+预充电+读/写** |
| **240** | **R4** | **R1** | **miss** | **285** | **激活+预充电+读/写** |
| **285** | **R2** | **R4** | **miss** | **330** | **激活+预充电+读/写** |
| **330** | **R27** | **R2** | **miss** | **375** | **激活+预充电+读/写** |

**应用A在通道0中165时刻完成，在通道1中330时刻完成，因此应用A在t0+330时刻完成。**

**应用B在通道0中300时刻完成，在通道1中150时刻完成，因此应用A在t0+300时刻完成。**

**应用C在通道0中435时刻完成，在通道1中375时刻完成，因此应用A在t0+375时刻完成。**

(c) FR-FCFS利用的是内存引用行为的什么特征? (6个字 ☺)

**答：行缓冲局部性。**

(d) 请简要描述可以最大化请求吞吐量的调度策略，请求吞吐量的意思是每单位时间响应的请求数。 (10个字左右☺)

**答：优先调度使行缓冲命中的请求。**

**4.2** 应用感知的调度策略

图中的3个应用，应用C是内存密集程度最低的(即有最少的请求数)。然而，它经历了最长的停顿时间，因为它的请求响应晚于其它多个被优先服务的应用的请求。为了保证应用C的停顿时间最短，可以为它的请求分配最高优先级，而给应用A和B的请求分配同样的低优先级。

1. 调度策略 X: 当应用C分配高优先级并且应用A和B分配相同的低优先级，每个应用的停顿时间是多少? (对于相同优先级的请求，假设使用FR-FCFS策略)

**通道0:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **开始时刻** | **访存序列** | **行缓冲** | **命中情况** | **结束时刻** | **DEAM命令** |
| **0** | **R21** | **R3** | **miss** | **45** | **激活+预充电+读/写** |
| **45** | **R3** | **R21** | **miss** | **90** | **激活+预充电+读/写** |
| **90** | **R3** | **R3** | **hit** | **105** | **读/写** |
| **105** | **R3** | **R3** | **hit** | **120** | **读/写** |
| **120** | **R3** | **R3** | **hit** | **135** | **读/写** |
| **135** | **R3** | **R3** | **hit** | **150** | **读/写** |
| **150** | **R5** | **R3** | **miss** | **195** | **激活+预充电+读/写** |
| **195** | **R9** | **R5** | **miss** | **240** | **激活+预充电+读/写** |
| **240** | **R12** | **R9** | **miss** | **285** | **激活+预充电+读/写** |
| **285** | **R11** | **R12** | **miss** | **330** | **激活+预充电+读/写** |
| **330** | **R17** | **R11** | **miss** | **375** | **激活+预充电+读/写** |

**通道1:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **开始时刻** | **访存序列** | **行缓冲** | **命中情况** | **结束时刻** | **DEAM命令** |
| **0** | **R27** | **R12** | **miss** | **45** | **激活+预充电+读/写** |
| **45** | **R12** | **R27** | **miss** | **90** | **激活+预充电+读/写** |
| **90** | **R12** | **R12** | **hit** | **105** | **读/写** |
| **105** | **R12** | **R12** | **hit** | **120** | **读/写** |
| **120** | **R18** | **R12** | **miss** | **165** | **激活+预充电+读/写** |
| **165** | **R18** | **R18** | **hit** | **180** | **读/写** |
| **180** | **R19** | **R18** | **miss** | **225** | **激活+预充电+读/写** |
| **225** | **R3** | **R19** | **miss** | **270** | **激活+预充电+读/写** |
| **270** | **R1** | **R3** | **miss** | **315** | **激活+预充电+读/写** |
| **315** | **R4** | **R1** | **miss** | **360** | **激活+预充电+读/写** |
| **360** | **R2** | **R4** | **miss** | **405** | **激活+预充电+读/写** |

**应用A在通道0中240时刻完成，在通道1中405时刻完成，因此应用A在t0+405时刻完成。**

**应用B在通道0中375时刻完成，在通道1中225时刻完成，因此应用A在t0+225时刻完成。**

**应用C在通道0中45时刻完成，在通道1中45时刻完成，因此应用A在t0+45时刻完成。**

你能否设计一个更好的调度策略? 虽然应用C的停顿时间小了，但是应用A和B之间还是会互相影响。

(b) 为其它两个应用分配优先级，这样你可以最小化所有应用的平均停顿时间。请具体从大到小列出三个应用的优先级(对于相同优先级的请求，假设使用FR-FCFS策略)

**答:假设C>A>B，同理可计算得到，应用A在t0+240时刻完成，应用B在t0+405时刻完成。**

**假设C>B>A，同理可计算得到，应用A在t0+405时刻完成，应用B在t0+180时刻完成。**

**则C>B>A的优先级可以最小化所有应用的平均停顿时间。**

(c) 调度策略Y: 使用你的新调度策略，每个应用的停顿时间分别是多少? (对于相同优先级的请求，假设使用FR-FCFS策略)

**答：应用A在t0+405时刻完成，应用B在t0+180时刻完成，应用C在t0+45时刻完成。**

(d) 请将四种调度策略 (FCFS, FR-FCFS, X, Y)的平均停顿时间从大到小排列

**答：FCFS:平均 425 ns；FR-FCFS:平均 335 ns；X:平均 275 ns；;Y:平均 210 ns；**

**所以FCFS> FR-FCFS >X>Y**

# 分层存储体系结构 10分

假设你研究出了下一代的存储技术:“魔法RAM”。魔法 RAM的位元是非易失性的；它的访问延迟是SRAM的2倍，与DRAM相同 ；读/写时的能耗和成本与DRAM相当；比DRAM的密度更高。然而，魔法RAM有一个缺点: 每个位元在执行2000次写操作之后会停止运转。

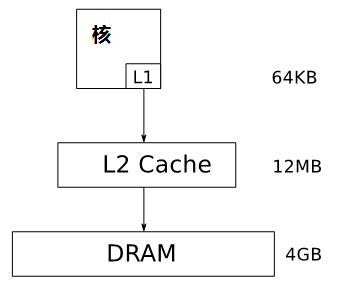
(a) 相比DRAM，魔法RAM除了密度高之外，还有什么优势么? 请解释。

**答：因为魔法RAM是非易失性的，所以它不需要刷新。 这样可以减少动态功率，总线利用率和存储区争用。 魔法RAM也是非易失性的，可以实现新用途或编程模型**

(b) 相比SRAM，魔法RAM有什么优势吗? 请解释。

**答：魔法RAM比SRAM具有更高的密度和更低的成本。**

(c) 假设一个系统有64KB SRAM的L1 cache 、12MB SRAM的L2 cache和4GB DRAM的主存。



假设你可以利用这个分层存储结构，经过自由地设计和增加任何结构以克服魔法RAM的缺陷(除了修改魔法RAM本身)

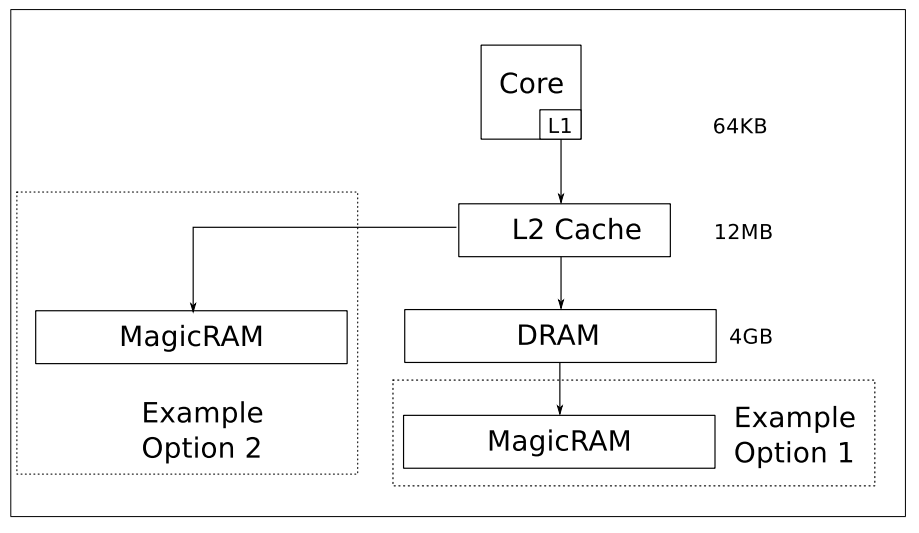
(i) 可能将魔法RAM加入这个分层存储结构以减小它的缺陷吗?

**答：可以。**

(ii) 如果可能, 魔法RAM该放到哪里? 根据上面的图来说明，并说明为什么选择放在这个位置。

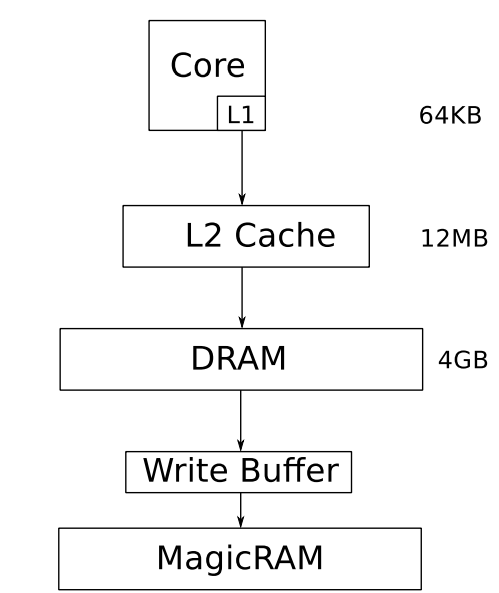
如果不可能，为什么? 请解释。

**答：将魔法RAM放在层次结构中的DRAM下面，并使用DRAM作为魔法RAM的缓存。 这样，DRAM执行写合并，因此魔法RAM不会很快磨损。 另一种选择是将魔法RAM放在DRAM的“旁边”（在同一或另一个通道上），然后将魔法RAM明确地用于只读数据。**



(d) 请给出一种通过修改这个分层存储结构以减少或克服魔法RAM缺陷的方法。请简单清晰地说明你的方法，可以利用图示来说明问题。

**答： 可以添加写合并/合并缓冲区。 存储器层次结构还可以执行耗损均衡，或者可以预测哪些数据不太可能被修改，而仅将该数据放入魔法RAM。**



# 虚存和cache 10分

一个2路组相联cache，采用写回策略和LRU替换策略，需要15x29 bit的标签存储来存储包括有效位、脏位以及LRU等标签信息。Cache虚拟索引，物理标签。虚拟地址空间1MB，页大小是2 KB，cache块大小是8字节。

(a)该cache的数据存储是多少字节？

**答：该cache是2路组相联， 所以每个组都有2个t大小的标签位，2个有效位，2个脏位，和1个LRU位（因为单个位足够为2路相联实现LRU算法）**

**所以假设i为索引位数。**

**则标签大小为2i×(2t+2+2+1)=15×29**

**所以t=5，i=9**

**所以数据存储为：29 ×(2×8)=8KB**

(b)虚拟索引中有多少位来自虚页号？

**答：1位。页面大小是2KB，因此页面的偏移位是11位(10:0)，cache块大小是3位(2:0),且虚拟索引为9位(11:3),因此虚拟索引的第一位(第11位)来自虚拟页码。**

(c)这个存储系统的物理地址空间有多大？

**答：页面偏移量是11位，与物理标签相同的物理帧号是5位，所以物理空间大小为211+5=216=64KB**