**CS570 “Advanced Computer Architecture”**

**Homework 7**

1. **MemoryInterleaving (30points)**

*A machine has a main memory of 4 KB, organized as 1 channel, 1 rank and N banks (where N > 1). The system does not have virtual memory.*

* *Data is interleaved using a cache block interleaving policy, as described in lecture, where consecutive cache blocks are placed on consecutive banks.*
* *The size of a cache block is 32 bytes. Size of a row is 128 bytes.*
* *An open row policy is used, i.e., a row is retained in the row-buffer after an access, until an access to another row is made.*
* *A row-buffer hit is an access to a row that is present in the row-buffer. A row-buffer miss is an access to a row that is not present in the row-buffer.*

1. **For a program executing on the machine, accesses to the following bytes miss in the on- chip caches and go to memory.**

**0, 32, 320, 480, 4, 36, 324, 484, 8, 40, 328, 488, 12, 44, 332, 492**

**The row-buffer hit rate is 0%, i.e., all accesses miss in the row-buffer. What is the minimum value of N - the number of banks?**

* 2 banks
* Size of a cache block is 32 bytes. Therefore, the cache block access sequence corresponding to the given byte access sequence is 0, 1, 10, 15, 0, 1, 10 ,15, 0, 1, 10, 15, 0, 1, 10, 15.
* When the number of banks is 1, all cache blocks are mapped to the same bank. Cache block 0, cache block 1, cache block 10 and cache block 15 are mapped to rows 0, 0, 2 and 3 respectively. Therefore, when cache block 1 is accessed right after cache block 0, it would hit in the row-buffer as row 0 would already be in the row-buffer. Hence, row-buffer hit rate would not be 0%.
* When the number of banks is 2, the 4 cache blocks 0, 1, 10 and 15 map to different rows and banks - (bank 0, row 0), (bank 1, row 0), (bank 0, row 1) and (bank 1, row 1) respectively. Hence, the access sequence would be (bank 0, row 0), (bank 1, row 0), (bank 0, row 1), (bank 1, row 1) (repeated four times). Therefore, rows 0 and 1 on each bank are accessed in an alternating fashion, resulting in a 0% row-buffer hit rate.

**b)  For the same sequence, if the row-buffer hit rate for the same sequence were 75%, what would be minimum value of N - the number of banks?**

* 4 banks
* When the number of banks is 1, cache blocks 0, 1, 10 and 15 map to rows 0, 0, 2 and 3 respectively (as we saw in part a). Thus, the row access sequence is 0, 0, 2, 3 (repeated four times). Three out of four accesses are row-buffer conflicts, hence the row-buffer hit rate is only 25%. For all other number of banks, the four cache blocks 0, 1, 10 and 15 map to different rows.
* Hence, assuming the rows containing the four cache blocks are not already open in the row-buffer, the maximum achievable hit rate is 75%, as the first access to each cache block would result in a (compulsory) row-buffer miss. This maximum hit rate of 75% can be achieved only if there are no row-buffer conflicts. Thus, rows containing the four cache blocks should map to different banks.
* Therefore, the minimum number of banks required to achieve this hit rate is 4.

**c)  Could the row-buffer hit rate for the sequence be 100%? Why or why not? Explain. If yes, what is the minimum number of banks required to achieve a row-buffer hit rate of 100%?**

* The four cache blocks are mapped to different rows. Hence the row-buffer hit rate can be 100% only if the row containing each cache block is already open.
* 4 banks is sufficient to achieve this, if the four rows containing the four cache blocks are already open (at each of the four banks).

2. **MemoryInterleavingII (20points)**

*A DRAM main memory system has N banks in one rank on one channel. A row is 256 bytes and a cache block is 64 bytes. Data is* ***row-interleaved across banks*** *using the following physical address bit scheme: (Row: Rank: Column: Byte Offset).*

*An open row policy is used, i.e., a row is retained in the row-buffer after an access until an access to another row is made. Initially, row 1024 is open in all banks in all channels.*

1. **Determine the total number of banks (across all channels) in the system given that the row-buffer hit rate on the following sequence of cache block numbers is 33.3% (1/3) Sequence:  
   0, 4, 8, 16, 32, 64, 128, 256, 128, 64, 32, 16, 8, 4, 0.**

25 = 32 banks.

1. **For what total number of banks in the system will the row-buffer hit rate of this sequence be 7/15?**

27 = 128 banks.

3. **MemoryBasics (15points)**

**a)  Which one needs refresh, SRAM or DRAM? Why?**

**SRAM**

SRAM (static RAM) is random access memory (RAM) that retains data bits in its memory as long as power is being supplied. Unlike dynamic RAM (DRAM), which stores bits in cells consisting of a capacitor and a transistor, SRAM ***does not have to be periodically refreshed***.

**DRAM**

Dynamic random access memory (DRAM) is a type of semiconductor [memory](https://searchstorage.techtarget.com/definition/memory-card) that is typically used for the data or program code needed by a computer processor to function. DRAM is a common type of random access memory (RAM) that is used in personal computers (PCs). A DRAM storage cell is dynamic, meaning that ***it needs to be refreshed*** or given a new [electronic charge](https://whatis.techtarget.com/definition/charge-electric-charge) every few [milliseconds](https://whatis.techtarget.com/definition/millisecond) to compensate for charge leaks from the capacitor.

**b)  What are the typical page table entry fields and what is the function of each field?**

* The map from virtual addresses (VA) to physical addresses (PA) is the Page Table.
* The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.
* The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.
* The Logical address Space is also splitted into fixed-size blocks, called **pages**.
* Page Size = Frame Size

Page table has *page table entries* where each page table entry stores a frame number and optional status (like protection) bits. Many of status bits used in the virtual memory system. The most important thing in PTE is frame Number. ***Page table entry has the following information.***

1. **Frame Number –** It gives the frame number in which the current page you are looking for is present. The number of bits required depends on the number of frames. Frame bit is also known as address translation bit.
2. **Present/Absent bit –** Present or absent bit says whether a particular page you are looking for is present or absent. In case if it is not present, that is called Page Fault. It is set to 0 if the corresponding page is not in memory. Used to control page fault by the operating system to support virtual memory. Sometimes this bit is also known as **valid/invalid** bits.
3. **Protection bit –** Protection bit says that what kind of protection you want on that page. So, these bit for the protection of the page frame (read, write etc).
4. **Referenced bit –** Referenced bit will say whether this page has been referred in the last clock cycle or not. It is set to 1 by hardware when the page is accessed.
5. **Caching enabled/disabled -** this bit **enables or disable** caching of the page. Some times we need the fresh data. we want the information has to be consistency, which means whatever information user has given, CPU should be able to see it as first as possible. That is the reason we want to disable caching
6. **Modified bit –** Modified bit says whether the page has been modified or not.

**c)  Where does page table locate? How to solve the slow page table lookup problem?**

The page table of the process is held in the kernel space. The kernel may have several page tables in RAM, but only one is the active page table. In x86 CPUs, it's the page table pointed by register CR3.

The page table lookup may fail for two reasons and:

* The lookup may fail if there is no translation available for the virtual address, meaning that virtual address is invalid. This will typically occur because of a programming error, and the operating system must take some action to deal with the problem. On modern operating systems, it will cause a *segmentation fault* signal being sent to the offending program.
* The lookup may also fail if the page is currently not resident in physical memory. This will occur if the requested page has been [moved out](https://en.wikipedia.org/wiki/Paging) of physical memory to make room for another page. In this case the page is paged out to a secondary store located on a medium such as a hard disk drive. *When this happens the page needs to be taken from disk and put back into physical memory. A similar mechanism is used for memory-mapped file, which are mapped to virtual memory and loaded to physical memory on demand*.

**d)  When accessing the DRAM, does the access time remain unchanged?**

Yes the access time remain unchanged, random access allows the PC processor to access any part of the memory directly rather than having to proceed sequentially from a starting place. RAM is located close to a computer's processor and enables faster access to data than storage media such as HDD and SSD.

**e)  What does the bank access latency consist of (Hint: multiple cases to discuss)?**

Access latency is the time that elapses from when a node sends a request for a file until it receives the complete file. DRAM Bank latency consists of

* Simple CAS (column address strobe) if row is “open” OR
* RAS (row address strobe) + CAS if array precharged OR
* PRE + RAS + CAS (worst case).

4. **MemoryOrganization(35points)**

a.

* Byte on bus Addr[1:0]
* Interleave bits Addr[4:2]
* Chip address Addr[7:5]
* Rank bits Addr[11:8]
* 577 Cycles.
* The first 8 memory accesses, A[0][0] to A[0][7], must occur sequentially with no overlap since they are all acesses to the same bank. Thus, it would take 80 cycles for the 1st 8 memory accesses, with the 8th access starting in cycle 70. Since the 8th and 9th memory accesses, A[0][7] and A[1][0], respectively, are to different banks, the accesses can overlap, and the 9th access can start in cycle 71 (70 cycles for the 1st 7 accesses plus 1 additional cycle of the 8th access).
* Continuing with this logic, the access to A[2][0] could start in cycle 142 (71x2). Finally, the access to A[7][0] could start in cycle 497 (71x7). Now all that remains are 8 more memory accesses, all to the same bank (A[7][0] to A[7][7]). This takes another 80 cycles, bringing the total to 577 cycles (497 + 80).
* If the memory were not interleaved, all 64 memory accesses must happen sequentially with no overlap, so it would take a total of 640 cycles (64\*10). Therefore, we do gain some benefit from this interleaving scheme, but not that much.
* Yes, a change can be made. The new bits are:
  + - Byte on bus Addr[1:0]
    - Interleave bits Addr[4:2]
    - Chip address Addr[11:9]
    - Rank bits Addr[8:5]
* 73 Cycles. With the new interleaving scheme, consecutive memory accesses are to either to different banks of the same rank, or to different ranks all together. In both cases, the consecutive accesses can start immediately after each other. Therefore the latency of all memory accesses would be hidden except the first access.
* Total number of cycles = 10 + 63 = 73

d.

Only one line of code needs to be changed.Alternatively, you could keep that line the same, but swap the variable (i/j) of the inner and outer loops as shown below.

sum=sum+A[i][j];  
to  
sum = sum + A[j][i];

**Original code**:

for(i=0;i<8;++i){  
 for(j=0;j<8;++j){  
 sum=sum+A[i][j];  
 }  
 }

**New code:**

for(j=0;j<8;++j){  
 for(i=0;i<8;++i){  
 sum=sum+A[i][j];  
 }  
 }

* 87 Cycles.
* Now consecutive memory accesses are to different banks, so the accesses can overlap. The 1st access, A[0][0], would begin at cycle 0, the 2nd, A[1][0], at cycle 1, and so on. The 8th access, A[7][0], would start at cycle 7.
* However, the 9th access, A[0][1], cannot start at cycle 8. It would have to wait 2 more cycles for the 1st access to finish since it is on the same bank as the 1st access; therefore, it would start at cycle 10. Continuing this logic, the access to A[0][2] would start at cycle 20, and finally, the access to A[0][7] would start at cycle 70.
* Now, all that is left are 8 accesses, but they are all to different banks so they can start 1 cycle after each other. The access to A[1][7] would begin at cycle 71, A[2][7] at 72, and finally A[7][7], the last memory access, would begin at cycle 77 and, therefore, end at cycle 87