Computer Architecture (263-2210-00L), Fall 2017
HW 2: Caches, Memory Parallelism, DRAM Fundamentals

Instructor: Prof. Onur Mutlu
TAs: Hasan Hassan, Arash Tavakkol, Mohammad Sadr, Lois Orosa, Juan Gomez Luna

Assigned: Tuesday, Oct 10, 2017
Due: **Tuesday, Oct 24, 2017**

- **Handin - Critical Paper Reviews (1).** You need to submit your reviews to `https://safari.ethz.ch/review/architecture/`. Please check your inbox. You should have received an email with the password you can use to login to the paper review system. If you have not received any email, please contact comparch@lists.ethz.ch. In the first page after login, you should click in "Architecture - Fall 2017 Home", and then go to "any submitted paper" to see the list of papers.
- **Handin - Questions (2-8).** Please upload your solution to the Moodle (`https://moodle-app2.let.ethz.ch/`) as a single PDF file. **Please use a typesetting software (e.g., LaTeX) or a word processor (e.g., MS Word, LibreOfficeWriter) to generate your PDF file. Feel free to draw your diagrams either using an appropriate software or by hand, and include the diagrams into your solutions PDF.**

1 Critical Paper Reviews [150 points]

Please read the following handout on how to write critical reviews. We will give out extra credit that is worth 0.5% of your total grade for each good review.

- Lecture slides on guidelines for reviewing papers. Please follow this format. `https://safari.ethz.ch/architecture/fall2017/lib/exe/fetch.php?media=onur-comparch-f17-how-to-do-the-paper-reviews.pdf`
- Some sample reviews can be found here: `https://safari.ethz.ch/architecture/fall2017/doku.php?id=readings`

(a) Write a one-page critical review for **two** of the following papers:

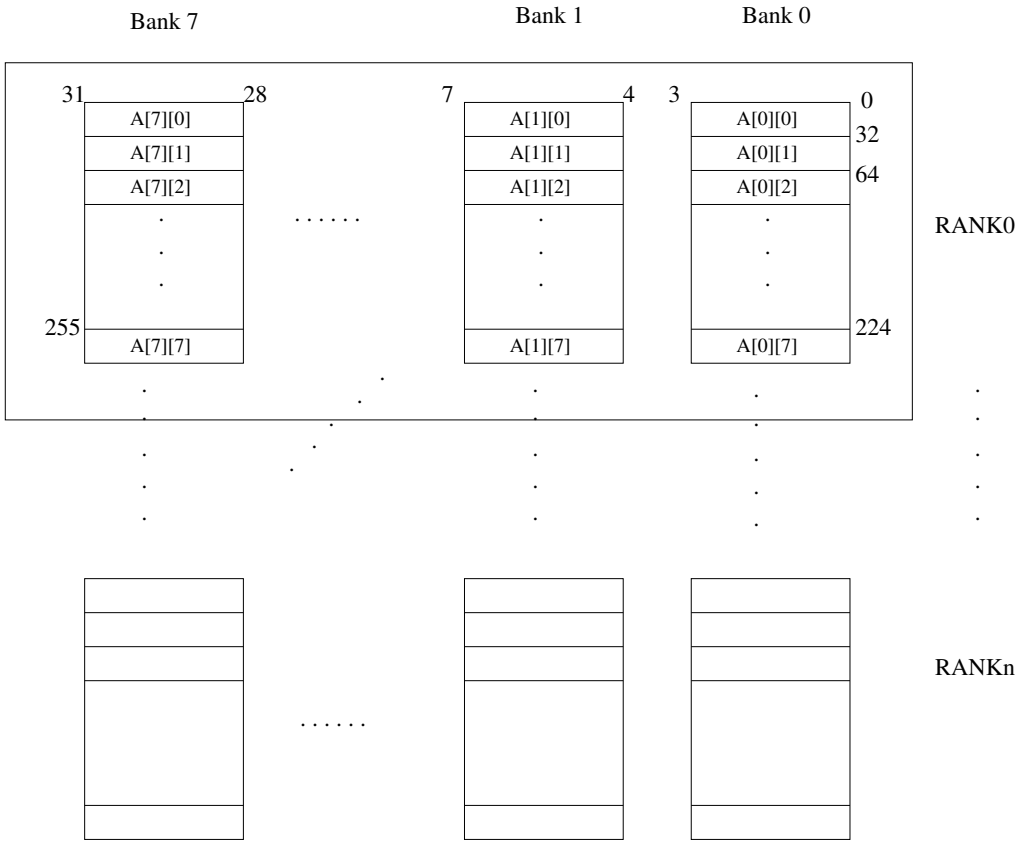
- M.K. Qureshi, D.N. Lynch, O. Mutlu, Y.N. Patt. "A Case for MLP-Aware Cache Replacement". ISCA 2006 `https://people.inf.ethz.ch/omutlu/pub/qureshi_isca06.pdf`
- D. Lee, Y. Kim, V. Seshadri, J. Liu, L. Subramanian, and O. Mutlu, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013 `https://people.inf.ethz.ch/omutlu/pub/tldram_hpca13.pdf`
- V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M.A. Kozuch, O. Mutlu, P.B. Gibbons, T.C. Mowry, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology, MICRO 2017 `https://people.inf.ethz.ch/omutlu/pub/ambit-bulk-bitwise-dram_micro17.pdf`

2 Main Memory Organization and Interleaving [150 points]

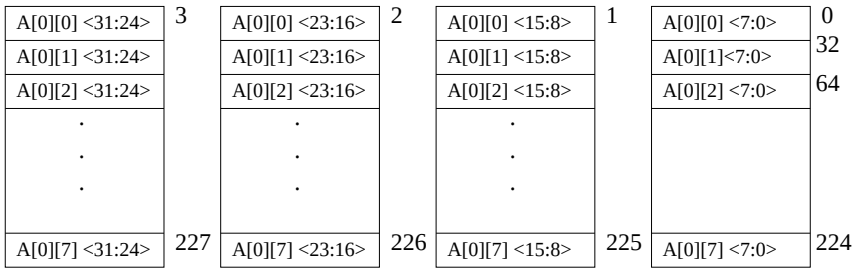
Consider the following piece of code:

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

The figure below shows an 8-way interleaved, byte-addressable memory. The total size of the memory is 4KB. The elements of the 2-dimensional array, A, are 4-bytes in length and are stored in the memory in column-major order (i.e., columns of A are stored in consecutive memory locations) as shown. The width of the bus is 32 bits, and each memory access takes 10 cycles.



A more detailed picture of the memory chips in Bank 0 of Rank 0 is shown below.



(a) Since the address space of the memory is 4KB, 12 bits are needed to uniquely identify each memory location, i.e., Addr[11:0]. Specify which bits of the address will be used for:

- Byte on bus
  Addr [_____ : _____]
- Interleave/Bank bits
  Addr [_____ : _____]
- Chip address
  Addr [_____ : _____]
- Rank bits
  Addr [_____ : _____]

(b) How many cycles are spent accessing memory during the execution of the above code? Compare this with the number of memory access cycles it would take if the memory were not interleaved (i.e., a single 4-byte wide array).

(c) Can any change be made to the current interleaving scheme to optimize the number of cycles spent accessing memory? If yes, which bits of the address will be used to specify the byte on bus, interleaving, etc. (use the same format as in part a)? With the new interleaving scheme, how many cycles are spent accessing memory? Remember that the elements of A will still be stored in column-major order.

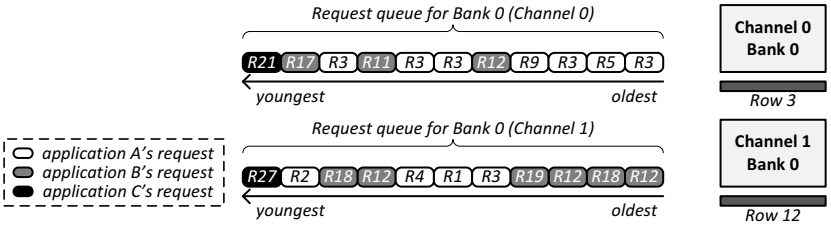
(d) Using the original interleaving scheme, what small changes can be made to the piece of code to optimize the number of cycles spent accessing memory? How many cycles are spent accessing memory using the modified code?

3 Memory Scheduling [150 points]

To serve a memory request, the memory controller issues one or multiple DRAM commands to access data from a bank. There are four different DRAM commands.

- `ACTIVATE`: Loads the row (that needs to be accessed) into the bank's row-buffer. This is called *opening* a row. **(Latency: 15ns)**
- `PRECHARGE`: Restores the contents of the bank's row-buffer back into the row. This is called *closing* a row. **(Latency: 15ns)**
- `READ/WRITE`: Accesses data from the row-buffer. **(Latency: 15ns)**

The following figure shows the snapshot of the memory request buffers (in the memory controller) at t_0. Each request is color-coded to denote the application to which it belongs (assume that all applications are running on separate cores). Additionally, each request is annotated with the address (or index) of the row that the request needs to access (e.g., *R3* means that the request is to the 3^{rd} row). Furthermore, assume that all requests are read requests.



A memory request is considered to be *served* when the `READ` command is complete (i.e., 15ns after the request's `READ` command has been issued). In addition, each application (A, B, or C) is considered to be **stalled** until *all* of its memory requests (across all the request buffers) have been served.

Assume that, initially (at t_0) each bank has the 3^{rd} and the 12^{th} row loaded in the row-buffer, respectively. Furthermore, no additional requests from any of the applications arrive at the memory controller.

3.1 Application-Unaware Scheduling Policies

(a) Using the **FCFS** scheduling policy, what is the **stall time** of each application?

(b) Using the **FR-FCFS** scheduling policy, what is the stall time of each application?

(c) What property of memory references does the *FR-FCFS* scheduling policy exploit? (Three words or less.)

(d) Briefly describe the scheduling policy that would **maximize** the *request throughput* at any given bank, where request throughput is defined as the number of requests served per unit amount of time. (Less than 10 words.)

3.2 Application-Aware Scheduling Policies

Of the three applications, application C is the least memory-intensive (i.e., has the lowest number of outstanding requests). However, it experiences the largest stall time since its requests are served only after the numerous requests from other applications are first served. To ensure the shortest stall time for application C, one can assign its requests with the highest priority, while assigning the same low priority to the other two applications (A and B).

(a) **Scheduling Policy X:** When application C is assigned a high priority and the other two applications are assigned the same low priority, what is the stall time of each application? (Among requests with the same priority, assume that *FR-FCFS* is used to break ties.)

Can you design an even better scheduling policy? While application C now experiences low stall time, you notice that the other two applications (A and B) are still delaying each other.

(b) Assign priorities to the other two applications such that you minimize the average stall time across all applications. Specifically, list all **three** applications in the order of highest to lowest priority. (Among requests with the same priority, assume that *FR-FCFS* is used to break ties.)

(c) **Scheduling Policy Y:** Using your new scheduling policy, what is the stall time of each application? (Among requests with the same priority, assume that *FR-FCFS* is used to break ties.)

(d) Order the four scheduling policies (FCFS, FR-FCFS, X, Y) in the order of lowest to highest average stall time.

4 Main Memory Potpourri [100 points]

A machine has a 4 KB DRAM main memory system. Each row is refreshed every 64 ms.

(a) The machine's designer runs two applications A and B (each run alone) on the machine. Although applications A and B have a similar number of memory requests, application A spends a surprisingly larger fraction of cycles stalling for memory than application B does? What might be the reasons for this?

(b) Application A also consumes a much larger amount of memory energy than application B does. What might be the reasons for this?

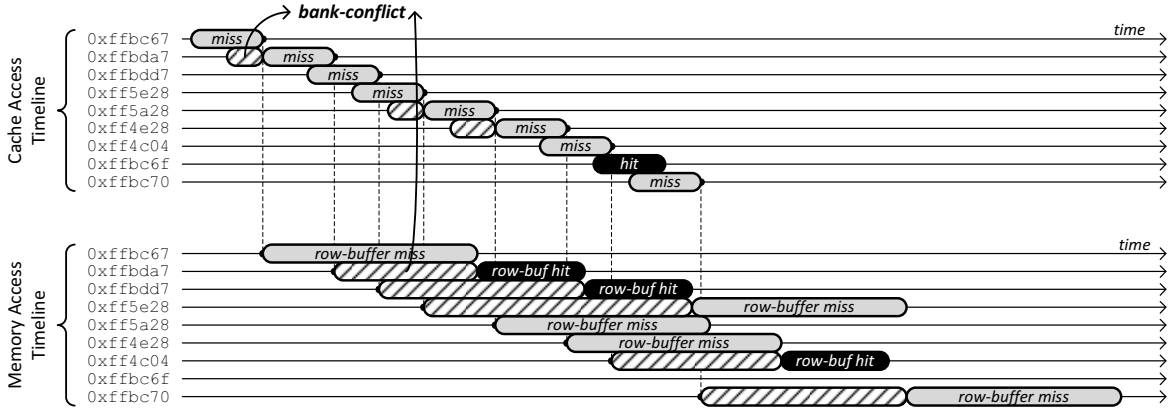
(c) When applications A and B are run together on the machine, application A's performance degrades significantly, while application B's performance doesn't degrade as much. Why might this happen?

(d) The designer decides to use a smarter policy to refresh the memory. A row is refreshed only if it has not been accessed in the past 64 ms. Do you think this is a good idea? Why or why not?

(e) The refresh energy consumption when application B is run, drops significantly when this new refresh policy is applied, while the refresh energy when application A is run reduces only slightly. Is this possible? Why or why not?

5 Banks [150 points]

A processor's memory hierarchy consists of a small SRAM L1-cache and a large DRAM main memory, both of which are banked. The processor has a 24-bit physical address space and does not support virtual memory (i.e., all addresses are physical addresses). An application has just started running on this processor. The following figure shows the timeline of memory references made by that application and how they are served in the L1-cache or main memory.



For example, the first memory reference made by the application is to byte-address `0xffbc67` (assume that all references are byte-sized reads to byte-addresses). However, the memory reference misses in the L1-cache (assume that the L1-cache is intially empty). Immediately afterwards, the application accesses main memory, where it experiences a row-buffer miss (initially, assume that all banks in main memory each have a row opened that will never be accessed by the application). Eventually, the cache block (and only that cache block) that contains the byte-address `0xffbc67` is fetched from memory into the cache.

Subsequent memory references may experience *bank-conflicts* in the L1-cache and/or main memory, if there is a previous reference still being served at that particular bank. Bank-conflicts are denoted as hatched shapes in the timeline.

The following table shows the address of the memory references made by the application in both hexadecimal and binary representations.

Hexadecimal	Binary
`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`

From the above timelines and the table, your job is to answer questions about the processor's cache and main memory organization. Here are some assumptions to help you along the way.

- Assumptions about the L1-cache
  - Block size: ? (Power of two, greater than two)
  - Associativity: ? (Power of two, greater than two)
  - Total data-store size: ? (Power of two, greater than two)
  - Number of banks: ? (Power of two, greater than two)
  - Initially empty
- Assumptions about main memory
  - Number of channels: 1
  - Number of ranks per channel: 1
  - Number of banks per rank: ? (Power of two, greater than two)
  - Number of rows per bank: ? (Power of two, greater than two)
  - Number of cache-blocks per row: ? (Power of two, greater than two)
  - Contains the entire working set of the application
  - Initially, all banks have their 0^{th} row open, which is never accessed by the application

5.1 First, let's cover the basics

(a) Caches and main memory are sub-divided into multiple banks in order to allow parallel access. What is an alternative way of allowing parallel access?

(b) A cache that allows multiple cache misses to be outstanding to main memory at the same time is called what? (Two words or less. Hint: It's an adjective.)

(c) While cache misses are outstanding to main memory, what is the structure that keeps bookkeeping information about the outstanding cache misses? This structure often augments the cache.

(d) Which is larger, an SRAM cell or a DRAM cell?

(e) What is the number of transistors and/or capacitors needed to implement each cell, including access transistor(s)?

5.2 Cache and memory organization

NOTE: For the following questions, assume that all offsets and indexes come from contiguous address bits.

(a) What is the L1-cache's block size in bytes? Which bit positions in the 24-bit physical address correspond to the cache block offset? (The least-significant bit in the physical address has a bit position of 0.)

(b) How many banks are there in the L1-cache? Which bit positions in the 24-bit physical address correspond to the L1-cache bank index? (The least-significant bit in the physical address has a bit position of 0.)

(c) How many banks are there in main memory? Which bit positions in the 24-bit physical address correspond to the main memory bank index? (The least-significant bit in the physical address has a bit position of 0.)

(d) What kind of interleaving is used to map physical addresses to main memory?

(e) To fully support a 24-bit physical address space, how many rows must each main memory bank have? Which bit positions in the 24-bit physical address correspond to the main memory row index? (The least-significant bit in the physical address has a bit position of 0.)

(f) Each cache block within a row is called a *column*. How many columns are there in a single row? Which bit positions in the 24-bit physical address correspond to the main memory column index? (The least-significant bit in the physical address has a bit position of 0.)

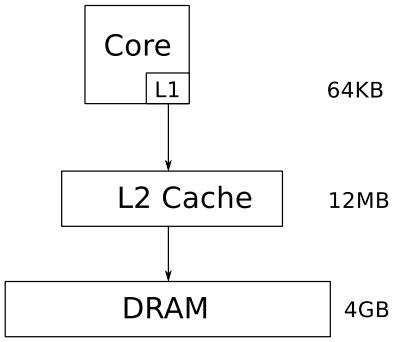
6 Memory Hierarchy [100 points]

Assume you developed the next greatest memory technology, MagicRAM. A MagicRAM cell is non-volatile. The access latency of a MagicRAM cell is 2 times that of an SRAM cell but the same as that of a DRAM cell. The read/write energy of MagicRAM is similar to the read/write energy of DRAM. The cost of MagicRAM is similar to that of DRAM. MagicRAM has higher density than DRAM. MagicRAM has one shortcoming, however: a MagicRAM cell stops functioning after 2000 writes are performed to the cell.

(a) Is there an advantage of MagicRAM over DRAM other than its density? (Please do not repeat what is stated in the above paragraph.) Explain.

(b) Is there an advantage of MagicRAM over SRAM? Explain.

(c) Assume you have a system that has a 64KB L1 cache made of SRAM, a 12MB L2 cache made of SRAM, and 4GB main memory made of DRAM.



Assume you have complete design freedom and add structures to overcome the shortcoming of MagicRAM. You will be able to propose a way to reduce/overcome the shortcoming of MagicRAM (note that you can design the hierarchy in any way you like, but cannot change MagicRAM itself).

(i) Does it makes sense to add MagicRAM anywhere in this memory hierarchy given that you can potentially reduce its shortcoming?

(ii) If so, where would you place MagicRAM? Describe it in terms of the figure above and describe why you made this choice.

If not, why not? Explain below clearly and methodically.

(d) Propose a way to reduce/overcome the shortcoming of MagicRAM by modifying the given memory hierarchy. Be clear in your explanations and illustrate with drawings to aid understanding.

7 Instruction and Data Caches [100 points]

Consider the following loop is executed on a system with a small instruction cache (I-cache) of size 16 B. The data cache (D-cache) is fully associative of size 1 KB. Both caches use 16-byte blocks. The instruction length is 4 B. The initial value of register `$1` is 40.

```
Exit: lw   $6, X($1)
      addi $6, $6, 1
      sw   $6, Y($1)
      subi $1, $1, 4
      beq  $1, $0, Exit
      j    Loop
      ...
```

(a) Compute I-cache and D-cache miss rates, considering:

- `X` and `Y` are different arrays.
- `X` and `Y` are the same array.

(b) Compute the average number of cycles per instruction (CPI), using a baseline ideal CPI (ideal caches) equal to 2, and a miss latency equal to 10 clock cycles.

(c) A compiler could unroll this loop for optimization. How would this affect CPI?

(d) How would the previous results change with a 32-byte I-cache?

8 Prefetching [100 points]

Suppose you have designed the next fancy hardware prefetcher for your system. You analyze its behavior and find the following:

(a) The prefetcher successfully prefetches block A into the cache before it is required by a load instruction. The prefetched block evicts a never-to-be-used block from the cache, so it does not cause cache pollution. Furthermore, you find that the prefetch request does not waste bus bandwidth needed by some other request.

(b) The prefetcher successfully prefetches block B into the cache before it is required by a load instruction. The prefetched block evicts a never-to-be-used block from the cache, so it does not cause cache pollution. Furthermore, you find that the prefetch request does not waste bus bandwidth needed by some other request.

Upon further analysis, you find that the prefetching of block A actually reduced execution time of the program whereas prefetching of block B did not reduce execution time significantly. Describe why this could happen. Draw two execution timelines, one with and one without the prefetcher, to illustrate the concept.