ETH 263-2210-00L Computer Architecture, Fall 2020
Discussion Session 2

Instructor: Prof. Onur Mutlu
TAs: Mohammed Alser, João Dinis Ferreira, Rahul Bera, Geraldo Francisco De Oliveira Junior, Can Firtina, Juan Gómez Luna, Jawad Haj-Yahya, Hasan Hassan, Konstantinos Kanellopoulos, Jeremie Kim,Nika Mansouri Ghiasi, Haiyu Mao, Lois Orosa Nogueira, Jisung Park, Minesh Patel, Gagandeep Singh, Kosta Stojiljkovic, Abdullah Giray Yaglikci

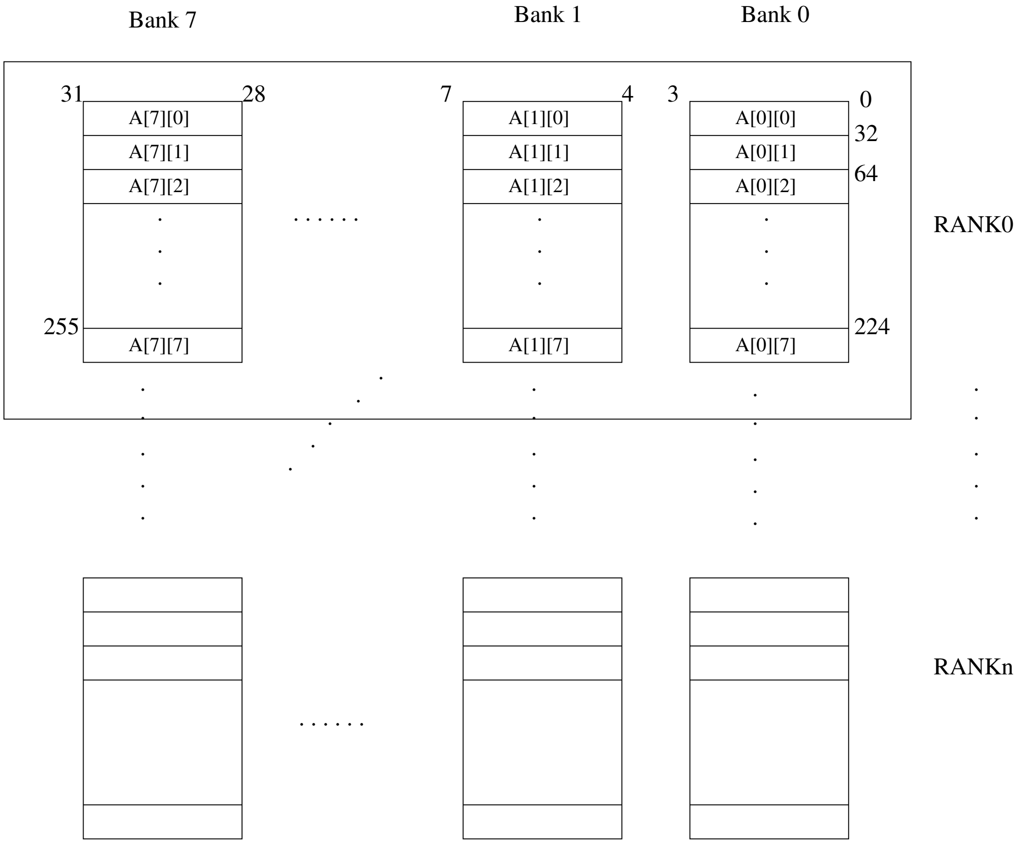
Given: Thursday, Dec 10, 2020

1 Main Memory Organization and Interleaving

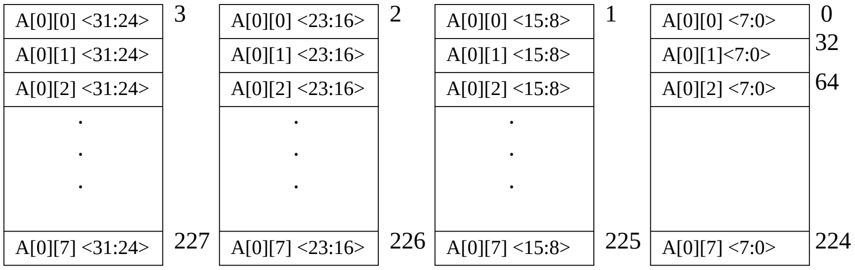
Consider the following piece of code:

```
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:

(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?

2 Main Memory Potpourri

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?

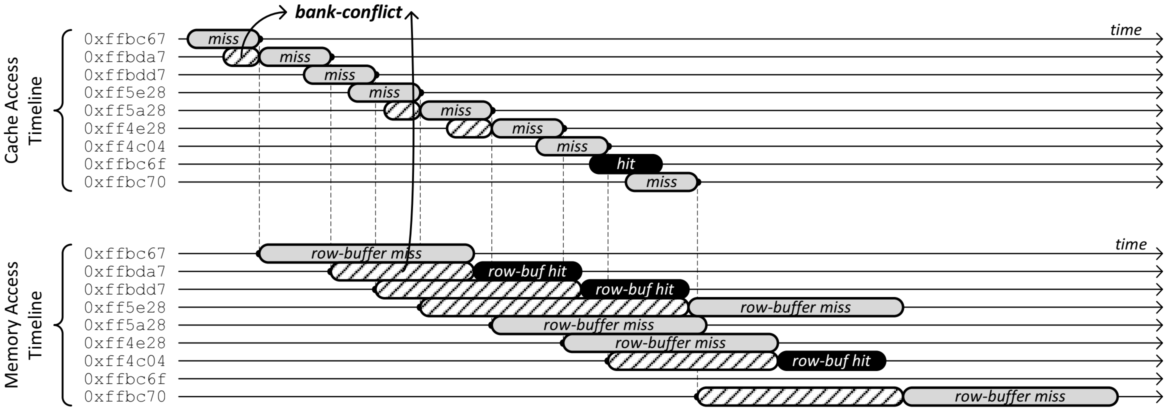
(c) When applications A and B are run together on the machine, application A's performance degrades significantly, while application B's performance doesn't degrade as much. Why might this happen?

(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?

3 Banks

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.

(a) Memory address table

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

3.1 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)?

3.2 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.)

4 Memory Hierarchy

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.

5 Prefetching

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.

6 DRAM Refresh

A memory system is composed of eight banks, and each bank contains 32K rows. The row size is 8 KB.

Every DRAM row refresh is initiated by a command from the memory controller, and it refreshes a single row. Each refresh command keeps the command bus busy for 5 ns.

We define *command bus utilization* as a fraction of the total time during which the command bus is busy due to refresh.

The retention time of each row depends on the temperature (T). The rows have different retention times, as shown in the following Table 1:

Retention Time	Number of rows
$(128 - T)$ ms, $0°C \leq T \leq 128°C$	2^8 rows
$2 * (128 - T)$ ms, $0°C \leq T \leq 128°C$	2^{16} rows
$4 * (128 - T)$ ms, $0°C \leq T \leq 128°C$	all other rows
$8 * (128 - T)$ ms, $0°C \leq T \leq 128°C$	2^8 rows

Table 1: Retention time

6.1 Refresh Policy A

Assume that the memory controller implements a refresh policy where all rows are refreshed with a fixed refresh interval, which covers the worst-case retention time (Table 1).

(a) What is the maximum temperature at which the DRAM can operate reliably with a refresh interval of 32 ms?

(b) What command bus utilization is directly caused by DRAM refreshes (with refresh interval of 32 ms)?

6.2 Refresh Policy B

Now assume that the memory controller implements a refresh policy where all rows are refreshed only as frequently as required to correctly maintain their data (Table 1).

(a) How many refreshes are performed by the memory controller during a 1.024 second period? (with T=64°C)

6.3 Refresh Policy C

Assume that the memory controller implements an even smarter policy to refresh the memory. In this policy, the refresh interval is fixed, and it covers the worst-case retention time (64ms), as the refresh policy in part 6.1. However, as an optimization, a row is refreshed only if it has *not* been **accessed** during the past refresh interval. For maintaining correctness, if a cell reaches its maximum retention time without being refreshed, the memory controller issues a refresh command.

(a) Why does a row *not* need to be refreshed if it was accessed in the past refresh interval?

(b) A program accesses all the rows repeatedly in the DRAM. The following table shows the access interval of the rows, the number of rows accessed with the corresponding access interval, and the retention times of the rows that are accessed with the corresponding interval.

Access interval	**Number of rows**	**Retention times**
1ms	2^{16} rows	64ms, 128ms or 256ms
60ms	2^{16} rows	64ms, 128ms or 256ms
128ms	all other rows	128ms or 256ms

(c) What command bus utilization is directly caused by DRAM refreshes?

6.4 Refresh Policy D

Assume that the memory controller implements an approximate mechanism to reduce refresh rate using Bloom filters, as we discussed in class. For this question we assume the retention times in Table 1 with a constant temperature of $64°C$.

One Bloom filter is used to represent the set of all rows that require a 64 ms refresh rate.

The memory controller's refresh logic is modified so that on every potential refresh of a row (every 64 ms), the refresh logic probes the Bloom filter. If the Bloom filter probe results in a "hit" for the row address, then the row is refreshed. Any row that does not hit in the Bloom filter is refreshed at the default rate of once per 128 ms.

(a) The memory controller performs 2107384 refreshes in total across the channel over a time interval of 1.024 seconds. What is the false positive rate of the Bloom filter? (NOTE: False positive rate = Total number of false positives / Total number of accesses).

- Hint: $2107384 = 2^3 * (2^{18} + 2^{11} - 2^{10} + 2^9 - 2^8 - 1)$

7 DRAM Refresh - Utilization

A memory system has four channels, and each channel has two ranks of DRAM chips. Each memory channel is controlled by a separate memory controller. Each rank of DRAM contains eight banks. A bank contains 32K rows. Each row in one bank is 8KB. The minimum retention time among all DRAM rows in the system is 64 ms. In order to ensure that no data is lost, every DRAM row is refreshed once per 64 ms. Every DRAM row refresh is initiated by a command from the memory controller which occupies the command bus on the associated memory channel for 5 ns and the associated bank for 40 ns. Let us consider a 1.024 second span of time.

We define *utilization* (of a resource such as a bus or a memory bank) as the fraction of total time for which a resource is occupied by a refresh command.

For each calculation in this section, you may leave your answer in *simplified* form in terms of powers of 2 and powers of 10.

(a) How many refreshes are performed by the memory controllers during the 1.024 second period in total across all four memory channels?

(b) What command bus utilization, across all memory channels, is directly caused by DRAM refreshes?

(c) What data bus utilization, across all memory channels, is directly caused by DRAM refreshes?

(d) What bank utilization (on average across all banks) is directly caused by DRAM refreshes?

(e) The system designer wishes to reduce the overhead of DRAM refreshes in order to improve system performance and reduce the energy spent in DRAM. A key observation is that not all rows in the DRAM chips need to be refreshed every 64 ms. In fact, rows need to be refreshed only at the following intervals in this particular system:

Required Refresh Rate	Number of Rows
64 ms	2^5
128 ms	2^9
256 ms	all other rows

Given this distribution, if all rows are refreshed only as frequently as required to maintain their data, how many refreshes are performed by the memory controllers during the 1.024 second period in total across all four memory channels?

What command bus utilization (as a fraction of total time) is caused by DRAM refreshes in this case?

(f) What DRAM data bus utilization is caused by DRAM refreshes in this case?

(g) What bank utilization (on average across all banks) is caused by DRAM refreshes in this case?

(h) The system designer wants to achieve this reduction in refresh overhead by refreshing rows less frequently when they need less frequent refreshes. In order to implement this improvement, the system needs to track every row's required refresh rate. What is the minimum number of bits of storage required to track this information?

(i) Assume that the system designer implements an approximate mechanism to reduce refresh rate using Bloom filters, as we discussed in class. One Bloom filter is used to represent the set of all rows which require a 64 ms refresh rate, and another Bloom filter is used to track rows which require a 128 ms refresh rate. The system designer modifies the memory controller's refresh logic so that on every potential refresh of a row (every 64 ms), it probes both Bloom filters. If either of the Bloom filter probes results in a "hit" for the row address, and if the row has not been refreshed in the most recent length of time for the refresh rate associated with that Bloom filter, then the row is refreshed. (If a row address hits in both Bloom filters, the more frequent refresh rate wins.) Any row that does not hit in either Bloom filter is refreshed at the default rate of once per 256 ms.

The false-positive rates for the two Bloom filters are as follows:

Refresh Rate Bin	False Positive Rate
64 ms	2^{-20}
128 ms	2^{-8}

The distribution of required row refresh rates specified in part (e) still applies.

How many refreshes are performed by the memory controllers during the 1.024 second period in total across all four memory channels?

What command bus utilization results from this refresh scheme?

What data bus utilization results from this refresh scheme?

What bank utilization (on average across all banks) results from this refresh scheme?

8 DRAM Scheduling and Latency

You would like to understand the configuration of the DRAM subsystem of a computer using reverse engineering techniques. Your current knowledge of the particular DRAM subsystem is limited to the following information:

- The physical memory address is **16 bits**.
- The DRAM subsystem consists of a single channel and 4 banks.
- The DRAM is byte-addressable.
- The most-significant 2 bits of the physical memory address determine the **bank**.
- The DRAM command bus operates at 500 MHz frequency.
- The memory controller issues commands to the DRAM in such a way that *no command* for servicing a *later* request is issued before issuing a READ command for the current request, which is the oldest request in the request buffer. For example, if there are requests A and B in the request buffer, where A is the older request and the two requests are to different banks, the memory controller does *not* issue an ACTIVATE command to the bank that B is going to access *before* issuing **a** READ **command** to the bank that A is accessing.

You **realize** that you can **observe** the memory requests that are waiting to be serviced in the request buffer. At a particular point of time, you take the snapshot of the request buffer and you observe the following requests in the request buffer.

Requests in the request buffer (in descending order of request age, where the oldest request is on the top):

time ↓

```
Read 0x4C80
Read 0x0140
Read 0x4EC0
Read 0x8000
Read 0xF000
Read 0x803F
Read 0x4E80
```

At the same time you **take** the snapshot of the request buffer, you start probing the DRAM command bus. You observe the DRAM command type and the cycle (relative to the first command) at which the command **is** seen on the DRAM command bus. The following are the DRAM commands **you observe** on the DRAM bus while the requests above are serviced.

```
Cycle 0 --- PRECHARGE
Cycle 6 --- ACTIVATE
Cycle 10 --- READ
Cycle 11 --- READ
Cycle 21 --- PRECHARGE
Cycle 27 --- ACTIVATE
Cycle 31 --- READ
Cycle 32 --- ACTIVATE
Cycle 36 --- READ
Cycle 37 --- READ
Cycle 38 --- READ
Cycle 42 --- PRECHARGE
Cycle 48 --- ACTIVATE
Cycle 52 --- READ
```

Answer the following questions using the information provided above.

(a) What are the following DRAM timing parameters used by the memory controller, in terms of nanoseconds?

i) ACTIVATE-to-READ latency

ii) ACTIVATE-to-PRECHARGE latency

iii) PRECHARGE-to-ACTIVATE latency

(b) What is the row size in bytes? Explain your answer.

(c) What is the status of the banks *prior* to the execution of any of the the above requests? In other words, which rows from which banks were open immediately prior to issuing the DRAM commands listed above? Fill in the table below indicating whether a bank has an open row, and if there is an open row, specify its address. If there is not enough information to infer the open row address, write *unknown*.

	Open or Closed?	**Open Row Address**
Bank 0		
Bank 1		
Bank 2		
Bank 3		

(d) To improve performance, you decide to implement the idea of Tiered-Latency DRAM (TL-DRAM) in the DRAM chip. Assume that a bank consists of a single subarray. With TL-DRAM, an entire bank is divided into a near-segment and far-segment. When accessing a row in the near-segment, the ACTIVATE-to-READ latency *reduces* by 2 cycles and the ACTIVATE-to-PRECHARGE latency reduces by 5 cycles. When accessing a row in the far-segment, the ACTIVATE-to-READ latency *increases* by 1 cycle and the ACTIVATE-to-PRECHARGE latency increases by 2 cycles.

Assume that the rows in the near-segment have smaller row ids compared to the rows in the far-segment. In other words, physical memory row addresses 0 through $N-1$ are the near-segment rows, and physical memory row addresses N through $M-1$ are the far-segment rows.

If the above DRAM commands are issued 5 cycles faster with TL-DRAM compared to the baseline (the last command is issued in cycle 47), how many rows are in the near-segment? Show your work.

9 Memory Scheduling

In lectures, we introduced a variety of ways to tackle memory interference. In this problem, we will look at the Blacklisting Memory Scheduler (BLISS) to reduce unfairness. There are two key aspects of BLISS that you need to know.

- When the memory controller services η consecutive requests from a particular application, this application is blacklisted. We name this non-negative integer η the **Blacklisting Threshold**.
- The blacklist is cleared periodically every **10000** cycles starting at $t = 0$.

To reduce unfairness, memory requests in BLISS are prioritized in the following order:

- Non-blacklisted applications' requests
- Row buffer hit requests
- Older requests

The memory system for this problem consists of 2 channels with 2 banks each. Tables 1 and 2 show the memory request stream in the same bank for both applications at different times ($t = 0$ and $t = 10$). For both tables, a request on the left-hand side is older than a request on the right-hand side in the same table. The applications do not generate more requests than those shown in Tables 1 and 2. The memory requests are labeled with numbers that represent the row position of the data within the accessed bank. Assume the following for all questions:

- A row buffer *hit* takes **100 cycles**.
- A row buffer *miss* (i.e., opening a row in a bank with a closed row buffer) takes **200 cycles**.
- A row buffer *conflict* (i.e., closing the currently open row and opening another one) takes **250 cycles**.
- All row buffers are closed at time $t = 0$

Application A (Channel 0, Bank 0)								
Application B (Channel 0, Bank 0)	Row 2	Row 2	Row 2	Row 2	Row 2	Row 3	Row 3	Row 4

Table 2: Memory requests of the two applications at $t = 0$

Application A (Channel 0, Bank 0)	Row 3	Row 7	Row 2	Row 0	Row 5	Row 3	Row 8	Row 9
Application B (Channel 0, Bank 0)	Row 2	Row 2	Row 2	Row 2	Row 2	Row 3	Row 3	Row 4

Table 3: Memory requests of the two applications at $t = 10$. Note that none of the Application B's existing requests are serviced yet.

(a) Compute the slowdown of each application using the FR-FCFS scheduling policy after both threads ran to completion. We define:

$$slowdown = \frac{memory\ latency\ of\ the\ application\ when\ run\ together\ with\ other\ applications}{memory\ latency\ of\ the\ application\ when\ run\ alone}$$

Show your work.

(b) If we use the BLISS scheduler, for what value(s) of η (the Blacklisting Threshold) will the slowdowns of **both** applications be equal to those obtained with FR-FCFS?

(c) For what value(s) of η (the Blacklisting Threshold) will the slowdown of A be < 1.5?

(d) For what value(s) of η (the Blacklisting Threshold) will B experience the maximum slowdown it can possibly experience with the Blacklisting Scheduler?

(e) What is a simple mechanism (that we discussed in lectures) that we can use instead of BLISS to make the slowdowns of both A and B equal to 1.00?

10 Emerging Memory Technologies

Researchers at Lindtel developed a new memory technology, L-RAM, which is non-volatile. The access latency of L-RAM is close to that of DRAM while it provides higher density compared to the latest DRAM technologies. L-RAM has one shortcoming, however: it has limited endurance, i.e., a memory cell stops functioning after 10^6 writes are performed to the cell (known as cell wear-out).

(a) Lindtel markets a new computer system with L-RAM to have a lifetime of 2 years and the following specifications:

- 4 GBs of L-RAM as main memory with a *perfect* wear-leveling mechanism, i.e., writes are equally distributed over all the cells of L-RAM.
- The processor is in-order and there is no memory-level parallelism.
- It takes 4 ns to send a memory request from the processor to the memory controller and it takes 20 ns to send the request from the memory controller to L-RAM. The write latency of L-RAM is 40 ns.
- L-RAM is word-addressable. Thus, each write request writes 8 bytes to memory.

A student at ETH tests the lifetime of the system and finds that this new computer system *cannot* guarantee a lifetime of 2 years. She writes a program to wear out the entire L-RAM device as quickly as possible. How fast is she able to wear out the device? Show all work.

(b) L-RAM works in the multi-level cell (MLC) mode in which each memory cell stores 2 bits. The student decides to improve the lifetime of L-RAM cells by using the single-level cell (SLC) mode. When L-RAM is used in SLC mode, the lifetime of each cell improves by a factor of 10 and the write latency decreases by 75%. What is the lifetime of the system using the SLC mode, if we repeat the experiment in part (a), with all else remaining the same in the system? Show your work.

(c) Provide a mechanism that would increase the guaranteed lifetime of the computer system without changing the physical circuitry of L-RAM. From the baseline computer system in part (a), describe the changes required to guarantee a computer system lifetime of 2 years, with your mechanism. Be concrete and precise.

11 Prefetching II

An ETH student writes two programs A and B and runs them on 3 different toy machines, M1, M2, and M3, to determine the type of the prefetching mechanism used in each of these 3 machines. She observes programs A and B to have the following access patterns to cache blocks. Note that the addresses are *cache block addresses*, not byte addresses.

Program A: 27 accesses

```
a, a + 1, a + 2, a + 3, a + 4, a + 8, a + 16, a + 32, a + 64,
a, a + 1, a + 2, a + 3, a + 4, a + 8, a + 16, a + 32, a + 64,
a, a + 1, a + 2, a + 3, a + 4, a + 8, a + 16, a + 32, a + 64
```

Program B: 501 accesses

```
b, b + 2, b + 4, ...., b + 998, b + 1000
```

The student is able to measure the accuracy and coverage of the prefetching mechanism in each of the machines. The following table shows her measurement results:

	Machine M1		**Machine M2**		**Machine M3**	
	Coverage	Accuracy	Coverage	Accuracy	Coverage	Accuracy
Program A	6/27	6/27	0	0	1/3	9/26
Program B	499/501	499/501	0	0	499/501	499/500

The student knows the following facts about M1, M2, and M3 machines:

- The prefetcher prefetches into a fully-associative cache whose size is 8 cache blocks. The cache employs the FIFO (First-In First-Out) replacement policy.
- The prefetchers have large enough resources to detect and store access patterns.
- Each cache block access is separated long enough in time such that all prefetches issued can complete before the next access happens.
- There are 5 different possible choices for the prefetching mechanism:
  1) Markov prefetcher with a correlation table of 4 entries
  2) Markov prefetcher with a correlation table of 10 entries
  3) 1st-next-block prefetcher (degree = 1) – prefetches block N + 1 after seeing block N
  4) 4th-next-block prefetcher (degree = 1) – prefetches block N + 4 after seeing block N
  5) stride prefetcher
- None of the above-mentioned prefetchers employ confidence bits.
- The prefetchers start out with an empty table when each program A and B start execution.
- The prefetcher sends only one prefetch request after a program access (i.e., prefetch degree = 1).

Determine what type of prefetching mechanism each of the above-mentioned machines use:

Machine M1:

Machine M2:

Machine M3:

Extra space for explanation: