# Computer Architecture (263-2210-00L), Fall 2018
# HW 1: Fundamentals, Memory Hierarchy, Caches

SOLUTIONS

Instructor: Prof. Onur Mutlu
TAs: Juan Gómez Luna, Hasan Hassan, Minesh Patel, Jeremie Kim, Mohammed Alser, Giray Yaglikci, Can Firtina, Geraldo Francisco de Oliveira

Assigned: Wednesday, Sep 26, 2018
Due: **Wednesday, Oct 10, 2018**

- **Handin - Critical Paper Reviews (1).** You need to submit your reviews to `https://safari.ethz.ch/review/architecture18/`. Please, check your inbox, you should have received an email with the password you should use to login. If you didn't receive any email, contact comparch@lists.ethz.ch. In the first page after login, you should click in "Architecture - Fall 2018 Home", and then go to "any submitted paper" to see the list of papers.
- **Handin - Questions (2-8).** You should upload your answers to the Moodle Platform (`https://moodle-app2.let.ethz.ch/mod/assign/view.php?id=274671`) as a single PDF file.

## 1 Critical Paper Reviews [300 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/fall2018/lib/exe/fetch.php?media=onur-comparch-f18-how-to-do-the-paper-reviews.pdf`
- Some sample reviews can be found here: `https://safari.ethz.ch/architecture/fall2018/doku.php?id=readings`

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

- Moscibroda and Mutlu, "Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems," in Proceedings of the USENIX Security, 2007. `https://people.inf.ethz.ch/omutlu/pub/mph_usenix_security07.pdf`
- Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," in Proceedings of the International Symposium on Computer Architecture, 2012. `https://people.inf.ethz.ch/omutlu/pub/raidr-dram-refresh_isca12.pdf`
- Kim et al., "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors", in Proceedings of the International Symposium on Computer Architecture, 2014. `https://people.inf.ethz.ch/omutlu/pub/dram-row-hammer_isca14.pdf`

(b) (Optional) Write a one-page critical review for the following paper:

- Patt, "Requirements, Bottlenecks, and Good Fortune: Agents for Microprocessor Evolution," in Proceedings of the IEEE, 2001. `https://safari.ethz.ch/architecture/fall2018/lib/exe/fetch.php?media=patt_ieee2001.pdf`

## 2 DRAM Refresh [150 points]

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?

$2^{25}$ or $32M$ refreshes.

There are $2^{23}$ refreshes per channel, because there are 16 refreshes/row in 1.024 seconds, and $2^{15} * 8 * 2 = 2^{21}$ rows/channel. With 4 channels, this yields $2^{25} = 32M$ refreshes. (Also accepted 32768000 refreshes if the assumption was made that $32K = 32000$ rather than $2^{15}$.)

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

4.096% utilization.
Each refresh contributes $5ns$ on one command bus. Overall there are $2^{25} * 5$ command bus-nanoseconds used, but there are four command buses, hence $2^{25} * 5ns/4$ of time for which each command bus is occupied. Over $1.024s$ total, this gives $2^{25} * 5ns/(4 * 1.024s) = 4.096\%$.

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

0: refreshes use only the command bus. They do not transfer data.

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

2.048% utilization.

In each bank, each of $2^{15}$ rows is refreshed 16 times, occupying the bank for a total of $2^{15} * 16 * 40ns$ of time, over $1.024s$ of total time, hence 2.048% utilization.

(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?

8391040 refreshes.

There are $(2^{21} - 2^9 - 2^5)$ rows that refresh 4 times (at 256ms intervals), $2^9$ rows that refresh 8 times (at 128ms intervals), and $2^5$ rows that refresh 16 times (at 64ms intervals). Hence, $4 * (2^{21} - 2^9 - 2^5) + 8 * (2^9) + 16 * (2^5) = 8391040$.
We also accepted answers based on the assumption that the row counts given were either per channel or per bank, as long as the assumption was clearly stated.

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

1.0243% utilization.

Each refresh occupies $5ns$ on one command bus, and we must take total command bus time occupied over the period of time on all command buses, hence $5ns * 8391040/(4 * 1.024) = 1.0243\%$.

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

0

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

0.5121%.

We can take the total bank occupancy time (computed as $40ns$ multiplied by total number of refreshes) over total number of banks times the total period: $40ns * 8391040/(64 * 1.024) = 0.5121\%$.

(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?

4 Mbit (2 bits per row). Also accepted simply "2 bits per row."

(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?

8423823 refreshes.

First find the number of rows refreshed at each rate. At 64 ms, we have $2^5$ rows which actually require this rate, plus $2^{-20}$ out of the $2^{21} - 2^5$ other rows which will be false-positives. At 128 ms, we have $2^9$ rows which actually require this rate, plus $2^{-8}$ out of the $2^{21} - 2^9 - 2^5$ other rows. At 256 ms we have the rest of the rows. During the 1.024s, each bin is refreshed respectively 16, 8, and 4 times. Thus we have $16 * (2^5 + (2^{21} - 2^5) * 2^{-20}) + 8 * (2^9 + (2^{21} - 2^9 - 2^5) * 2^{-8}) + 4 * (2^{21} - 2^9 - 2^5 - (2^{21} - 2^9) * 2^{-20} - (2^{21} - 2^9 - 2^5) * 2^{-8}) = 8423823$ total refreshes.

What command bus utilization results from this refresh scheme?

1.028% utilization.

$5ns * 8423823/(4 * 1.024s) = 1.028\%$.

What data bus utilization results from this refresh scheme?

0

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

0.5141% utilization.

$40ns * 8423823/(64 * 1.024s) = 0.5141\%$.

## 3 Data Flow Programs [100 points]

The Fibonacci number $F_n$ is recursively defined as

$$F(n) = F(n-1) + F(n-2),$$

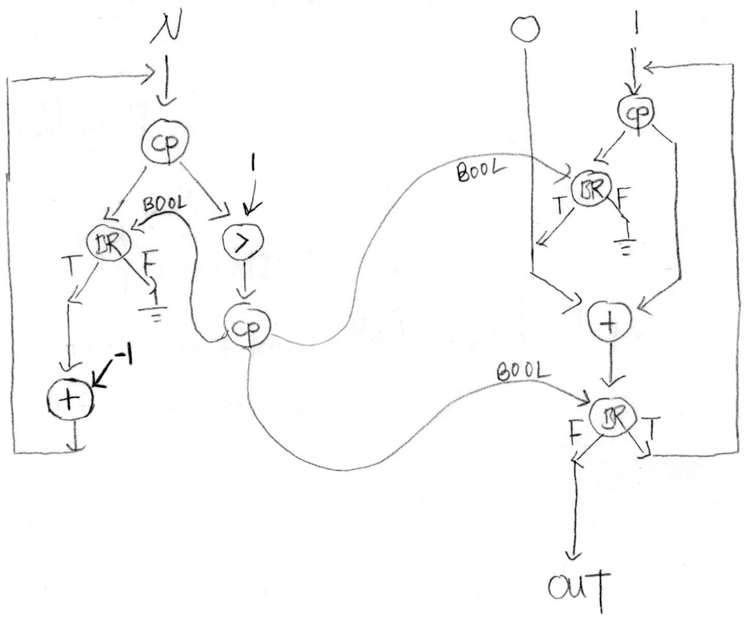
where $F(1) = 1$ and $F(2) = 1$. So, $F(3) = F(2) + F(1) = 1 + 1 = 2$, and so on. The Fibonacci number can be computed as $F($n$)$:

```
int fib(int n)
{
  int a = 0;
  int b = 1;
  int c = a + b;
  while (n > 1) {
    c = a + b;
    a = b;
    b = c;
    n--;
  }
  return c;
}
```

Draw the data flow graph for the `fib(n)` function. You may use the following data flow nodes in your graph:

- + (addition)
- > (left operand is greater than right operand)
- `Copy` (copy the value on the input to both outputs)
- `BR` (branch, with the semantics discussed in class, label the True and False outputs)

You can use constant inputs (e.g., 1) that feed into the nodes. Clearly label all the nodes, program inputs, and program outputs. Try to the use fewest number of data flow nodes possible.



## 4 Caches [100 points]

A byte-addressable system with 16-bit addresses ships with a two-way set associative, writeback cache with perfect LRU replacement. The tag store (including the tag and all other meta-data) requires a total of 4352 bits of storage. What is the block size of the cache? Assume that the LRU information is maintained on a per-set basis as a single bit. (Hint: $4352 = 2^{12} + 2^{8}$ .)

We formulate two basic equations:

$$4352 = 2^{index} * (2 * (tag + dirty + valid) + LRU) \tag{1}$$

$$tag + index + offset = 16 \tag{2}$$

There is one dirty bit and one valid bit per block, and one LRU bit per set. So, now the Equation 1 looks like: $4352 = 2^{index} * (2 * (tag + 2) + 1) = (2^{index} * (2tag + 4)) + 2^{index}$

By using the hint ($4352 = 2^{12} + 2^{8}$), we get $2^{index} = 2^{8}$, so $index = 8$, and from $2^{index} * (2tag + 4) = 2^{12}$ we get $(2tag + 4) = 2^{4}$, so $tag = 6$.

By solving the Equation 2, we get $offset = 2$, so, the block size is $2^{2} =$ **4 bytes**

## 5 Reverse Engineering Caches [100 points]

You're trying to reverse-engineer the characteristics of a cache in a system so that you can design a more efficient, machine-specific implementation of an algorithm you're working on. To do so, you've come up with four patterns that access various *bytes* in the system in an attempt to determine the following four cache characteristics:

- Cache block size (8, 16, 32, 64, or 128 B)
- Cache associativity (1-, 2-, 4-, or 8-way)
- Cache size (4 or 8 KB)
- Cache replacement policy (LRU or FIFO)

However, the only statistic that you can collect on this system is cache hit rate after performing the access pattern. Here is what you observe:

| Access Pattern | Blocks Accessed (Oldest → Youngest) | | | | | | | | | Hit Rate |
|---|---|---|---|---|---|---|---|---|---|---|
| A | 0 | 4096 | 8192 | 12288 | 16384 | 4096 | 0 | | | 1/7 |
| B | 0 | 1024 | 2048 | 3072 | 4096 | 5120 | 6144 | 3072 | 0 | 1/9 |
| C | 0 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 4/9 |
| D | 128 | 1152 | 2176 | 3200 | 128 | 4224 | 1152 | | | 2/7 |

Based on what you observe, what are the following characteristics of the cache? (Be sure to justify clearly your answer for full credit.)

(a) Cache block size (8, 16, 32, 64, or 128 B)?

Let's focus on access pattern C for this part. In this access pattern, for a given cache block size, all bytes map to certain sets—regardless of cache associativity and regardless of cache size.
There is only one mapping that causes 4 accesses out of 9 to be hits: At a cache block size of 64 B, addresses 4, 8, 16, and 32 all hit in the block brought in by the access to address 0. The other accesses go to different cache blocks.

(b) Cache associativity (1-, 2-, 4-, or 8-way)?

Let's focus on access pattern A for this part. In this access pattern, for a given cache associativity, all bytes map to the *same* set—regardless of cache block size and regardless of cache size.
There is only one associativity that causes 1 access out of 7 to be a hit: An associativity of 4 allows address 4096 to be a hit. Any less, and no accesses would hit; any more, and address 0 would also hit.

(c) Cache size (4 or 8 KB)?

Let's focus on access pattern B for this part. In this access pattern, given a cache associativity of 4, and for a given cache size, all bytes map to the *same* set—regardless of cache block size.
There is only one cache size that maps blocks to sets in a way that causes 1 access out of 9 to be a hit: A cache size of 4 KB causes all of the blocks to map to the same set, generating a hit for address 3072. At 8 KB, the access to address 0 also hits in the cache.

(d) Cache replacement policy (LRU or FIFO)?

Let's focus on access pattern D for this part. In this access pattern, given a cache associativity of 4, and given a cache size of 4 KB, for a given replacement policy, all bytes map to the same set—regardless of cache block size.
First, notice that the access to address 128 will be a hit: Under LRU, the next block to be evicted would be the one for address 1152; whereas, under FIFO, the next block to be evicted would be the one for address 128.
Second, notice that the access to address 4224 then evicts the respective block. The access for address 1152 would then miss under LRU, but hit under FIFO, for a total of two hits (including the access to address 128). FIFO is the only cache replacement policy that causes 2 accesses out of 7 to be hits.

## 6 Sectored Cache vs. Smaller Blocks [100 points]

After mounds of coffee and a good number of high-level simulation cycles, you narrowed down the design choices for the L1 cache of the next-generation processor you are architecting to the following two:

**Choice 1.** A sectored 64KB cache with 64-byte blocks and 8-byte subblocks

**Choice 2.** A non-sectored 64KB cache with 8-byte blocks

Assume the associativity of the two caches are the same.

(a) What is one definitive advantage of the sectored cache over the non-sectored one?

Smaller tag store size.

(b) What is one definitive advantage of the sectored cache over the non-sectored one?

Potentially better hit-rate if program does not have spatial locality.

(c) Which of the choices has the faster cache hit latency? Circle one:

**Choice 1** **Choice 2** **Not Enough Information**

Justify your choice, describing the tradeoffs involved if necessary.

The described sectored cache has fewer sets. Hence, the tag access time will be shorter, which may result in a shorter cache hit latency.

On the other hand, the described sectored cache must check more valid bits, and must mux out the appropriate data from a 64-byte-wide block, while the described non-sectored cache checks only one valid bit per tag and muxes data out of a smaller 8-byte block. Hence, the non-sectored cache may also have a shorter cache hit latency.

Without knowing exactly how long each of these components of the latency takes, we cannot know which choice of cache design has lower hit latency.

## 7 Memory Interleaving [100 points]

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.

(a) 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?

**Solution:**
**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) If the row-buffer hit rate for the same sequence were 75%, what would be minimum value of $N$ - the number of banks?

**Solution:**
**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) i) Could the row-buffer hit rate for the sequence be 100%? Why or why not? Explain.

**Solution:**
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.

ii) If yes, what is the minimum number of banks required to achieve a row-buffer hit rate of 100%?

**Solution:**
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).

## 8 Virtual Memory [50 points]

An ISA supports an 8-bit, byte-addressable virtual address space. The corresponding physical memory has only 128 bytes. Each page contains 16 bytes. A simple, one-level translation scheme is used and the page table resides in physical memory. The initial contents of the frames of physical memory are shown below.

| Frame Number | Frame Contents |
|---|---|
| 0 | Empty |
| 1 | Page 13 |
| 2 | Page 5 |
| 3 | Page 2 |
| 4 | Empty |
| 5 | Page 0 |
| 6 | Empty |
| 7 | Page Table |

A three-entry translation lookaside buffer that uses Least Recently-Used (LRU) replacement is added to this system. Initially, this TLB contains the entries for pages 0, 2, and 13. For the following sequence of references, put an 'H' below those that generate a *TLB hit* and write "PF" below those that generate a *page fault*. What is the hit rate of the TLB for this sequence of references? (Note: LRU policy is used to select pages for replacement in physical memory.)

References (to pages): 0, 13, 5, 2, 14, 14, 13, 6, 6, 13, 15, 14, 15, 13, 4, 3.

(a) At the end of this sequence, what three entries are contained in the TLB?

(b) What are the contents of the 8 physical frames?

References (to pages): 0, 13, 5, 2, 14, 14, 13, 6, 6, 13, 15, 14, 15, 13, 4, 3.

**Solution:**

| References (to pages): | 0 | 13 | 5 | 2 | 14 | 14 | 13 | 6 | 6 | 13 | 15 | 14 | 15 | 13 | 4 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | H | H | | | PF | H | | PF | H | H | PF | | H | H | PF | PF |

TLB Hit Rate = 7/16

(a) At the end of this sequence, what three entries are contained in the TLB?

**Solution:**
4, 13, 3

(b) What are the contents of the 8 physical frames?

**Solution:**
Pages 14, 13, 3, 2, 6, 4, 15, Page table