ETH 263-2210-00L Computer Architecture, Fall 2020
HW 1: DRAM Refresh (SOLUTIONS)

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, Sep 24, 2020
Due: **Thursday, Oct 08, 2020**

- **Handin - Critical Paper Reviews (1).** You need to submit your reviews to `https://safari.ethz.ch/review/architecture20/`. Please, check your inbox, you should have received an email with the password you should use to login. If you did not receive any email, contact comparch@lists.inf.ethz.ch. In the first page after login, you should click in "Computer Architecture Home", and then go to "any submitted paper" to see the list of papers.
- **Handin - Questions (2-6).** You should upload your answers to the Moodle Platform (`https://moodle-app2.let.ethz.ch/course/view.php?id=13549`) as a single PDF file.

1. Critical Paper Reviews [1000 points]

Please read the guidelines for reviewing papers and check the sample reviews. We also assign you a **required reading** for this homework. You may access them by *simply clicking on the QR codes below or scanning them.* We will give out extra credit that is worth 0.5% of your total grade for each good review. If you submit 5 or 6 paper reviews, you will receive 250 or 500 BONUS points on top of 1000 points you may get from paper reviews, respectively (i.e., each additional submission is worth 250 BONUS points).

Guidelines   Sample reviews   Required Reading

Write an approximately one-page critical review for the following required reading (i.e., Paper #1) **and** *at least* 3 of the remaining 5 papers (i.e., papers from #2 to #6). A review with bullet point style is more appreciated. Try not to use very long sentences and paragraphs. Keep your writing and sentences simple. Make your points bullet by bullet, as much as possible.

1. (REQUIRED) Onur Mutlu, "Main Memory Scaling: Challenges and Solution Directions", Invited Book Chapter in More than Moore Technologies for Next Generation Computer Design, pp. 127-153, Springer, 2015. `https://people.inf.ethz.ch/omutlu/pub/main-memory-scaling_springer15.pdf`
2. 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`
3. 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`
4. Liu et al., "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms", in Proceedings of the International Symposium on Computer Architecture, 2013.
`https://people.inf.ethz.ch/omutlu/pub/dram-retention-time-characterization_isca13.pdf`
5. 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`
6. Kim et al., "Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques", in Proceedings of the International Symposium on Computer Architecture, 2020. `https://people.inf.ethz.ch/omutlu/pub/Revisiting-RowHammer_isca20.pdf`

2. Main Memory [150 points]

Answer the following questions for a machine that has a 4 GB DRAM main memory system and each row is refreshed every 64 ms.

(a) During standalone runs of two applications (A and B) on the machine, you observe that application A spends a significantly larger fraction of cycles stalling for memory than application B does while both applications have a similar number of memory requests. What might be the reasons for this?

A large number of application A's memory requests might be causing row-buffer conflicts, whereas application B's memory requests have better row-buffer locality. Hence, application A's requests might take longer time to serve and it spends more time stalling for memory.
Note: This question is open ended. There can be other correct solutions.

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

A row-buffer conflict consumes more energy than a a row-buffer hit. A row-buffer conflict requires a precharge, an activate and a read/write, whereas a row-buffer hit only requires a read/write.
Hence, application A can consume more memory energy, if it observes higher number of row-buffer conflicts.
Note: This question is open ended. There can be other correct solutions.

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

When the applications run together, they interfere with each other. Hence, both applications' performance degrades when they run together. However, if a memory scheduler that favors row-buffer hits over row-buffer conflicts (like FR-FCFS) is used, it would favor application B's requests over application A's requests. Therefore, application A's performance can degrade more.
Note: This question is open ended. There can be other correct solutions.

(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?

This can reduce refresh energy significantly if a large fraction of the rows in memory contain data and are accessed (within the 64 ms refresh window), as these rows do not have to be refreshed explicitly. However, if only a small number of rows contain data and only these rows are accessed, this policy will not provide much reduction in refresh energy as a large fraction of rows are still refreshed at the 64 ms rate.
One should consider the area, performance, and energy overheads of the required additional storage space to keep track of the row accesses.
Note: This question is open ended. There can be other correct solutions.

(e) When this new refresh policy is applied, the refresh energy consumption drops significantly during a run of application B. In contrast, during a run of application A, the refresh energy consumption reduces only slightly. Why might this happen?

This is possible. If application B has a large working set, it could access a large fraction of the memory rows (within the 64 ms refresh window) and hence these rows do not have to be refreshed explicitly. On the other hand, application A could have a much smaller working set and hence a large fraction of rows still have to be refreshed at the 64 ms rate.
Note: This question is open ended. There can be other correct solutions.

3. DRAM Refresh - Utilization [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^{23} refreshes per channel; $2^{25} = 32M$ across 4 channels.

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

$4.096\% = (32M \times 5ns/(4 \times 1.024s))$

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

0

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

$2.048\% = (32K \times 16 \times 40ns/1.024s)$

(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?

$((2^15 - (2^9 + 2^5) \times 1) + (2^5 \times (\frac{256}{64}) + (2^9 \times (\frac{256/128}{)})$ row refreshes per 256ms.
$\times 4$ for 1.024s. $\times 8$ for number of banks per rank $\times 2$ for ranks per channel $\times 4$ for channels per DRAM

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

$1.0243\% = 5e^{-9} * (2^5 \times 16 + 2^9 \times 8 + (2^{21} - 2^9 - 2^5) \times 4)/(4 \times 1.024)$

(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?

$0.5121\% = 8 * (5e^{-9}) \times (2^5 \times 16 + 2^9 \times 8 + (2^{21} - 2^9 - 2^5) \times 4)/(64 \times 1.024)$

(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)

(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?

false positives: 8192 + 2 (8192)

What command bus utilization results from this refresh scheme?

$5e^{-9} \times ((2^5 + 2) \times 16 + (2^9 + 2^{13}) \times 8 + (2^{21} - 2^9 - 2^5 - 2 - 2^{13}) \times 4)/(4 \times 1.024)$

What data bus utilization results from this refresh scheme?

0%

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

$0.5141\% = 8 * (5e^{-9}) \times ((2^5 + 2) \times 16 + (2^9 + 2^{13}) * 8 + (2^{21} - 2^9 - 2^5 - 2 - 2^{13}) \times 4)/(64 \times 1.024)$

4. RowHammer [150 points]

4.1. RowHammer Attacks

In order to characterize the vulnerability of your DRAM device to RowHammer attacks, you must be able to induce RowHammer errors. Assume the following about the target system:

- The CPU has a single in-order processor, and does not implement virtual memory.
- The physical memory address is 16 bits.
- The DRAM subsystem consists of two channels, four banks per channel, and 64 rows per bank.
- The memory controller employs open-page policy.
- The DRAM modules and the memory controller do not employ any remapping or scrambling schemes for the physical address.
- All the cells in the DRAM subsystem are equally vulnerable to RowHammer-induced errors.

You implement codes based on instructions shown in Table 1.

Instruction	Description	Functionality
`B LABEL`	Unconditional Branch	`PC = LABEL`
`STORE IMM, Rs`	Store word to memory	`MEM[IMM] = Rs`
`CLFLUSH IMM`	Cache line flush	`Flush cache line containing IMM`

Table 1. Instruction Descriptions.

(a) You run Code 1 below, but you cannot observe any errors in the target system. You figured out that the number of activations is much lower than your expectation. Give reason(s) as to why Code 1 cannot introduce a sufficient amount of activations.

Code 1

```
LOOP:
  STORE 0x8732, R0
  CLFLUSH 0x8732
  B LOOP
```

All of reads in Code 1 are to the same row in DRAM, and the memory controller minimizes the number of DRAM commands by opening and closing the row just once, while issuing many column reads.

(b) You try Codes 2a, 2b, and 2c, but find that *only one of them can induce RowHammer errors* in your DRAM subsystem. Which code segment is the one that can induce RowHammer errors? Justify your answer.

Code 2a

```
LOOP:
   STORE 0x8732, R0
   STORE 0x98CD, R1
   CLFLUSH 0x8732
   CLFLUSH 0x98CD
   B LOOP
```

Code 2b

```
LOOP:
   STORE 0xF1AB, R0
   STORE 0x0054, R1
   CLFLUSH 0xF1AB
   CLFLUSH 0x0054
   B LOOP
```

Code 2c

```
LOOP:
   STORE 0x2B97, R0
   STORE 0xDA68, R1
   CLFLUSH 0x2B97
   CLFLUSH 0xDA68
   B LOOP
```

(a) In order to introduce enough activations, two `STORE` instructions should access *different rows in the same bank.*

(b) Three code segments are identical except for the memory addresses, so we can assume that only one code segment has two `STORE` instructions whose destination addresses are assigned to the same bank (but different rows).

(c) Since the DRAM subsystem 1) consists of 8 banks and 2) employs no address remapping/scrambling schemes, two addresses assigned the same bank should satisfy a condition C: they have at least three same bit values at the same position.

Two addresses in each code are:

- **Code 2a**
  $\underline{\texttt{100}}\texttt{0 0111 0011 0010 (0x8732)}$
  $\underline{\texttt{100}}\texttt{1 1000 1100 1101 (0x98CD)}$
- **Code 2b**
  $\texttt{1111 }\underline{\texttt{000}}\texttt{1 1010 1011 (0xF1AB)}$
  $\texttt{0000 }\underline{\texttt{000}}\texttt{0 0101 0100 (0x0054)}$
- **Code 2c**
  $\texttt{0010 }\underline{\texttt{101}}\texttt{1 1001 0111 (0x2B97)}$
  $\texttt{1101 }\underline{\texttt{101}}\texttt{0 0110 1000 (0xDA68)}$

We can observe

(a) Two paired addresses in every code segment have only three same bit values at the same position, i.e., satisfy C.

(b) The position of the same bit values in Code 2b and Code 2c is the same, but different from Code 1. Therefore, if Code 2b can induce RowHammer errors, Code 2c should also be able toinduce errors and Code 2a should not. On the other hand, if Code 2a can induce RowHammer errors, neither Code 2b or Code 2c can induce errors.

Since only one code segment can induce RowHammer errors, Code 2a is the one able to induce RowHammer errors.

4.2. RowHammer Mitigation Mechanisms

To identify a viable RowHammer mitigation mechanism for your system, you compare the two following mitigation mechanisms:

- **Mechanism A.** The memory controller maintains a counter for every row, which increments every time the corresponding row is activated. If the counter value for a row exceeds a threshold value T, the memory controller activates the row's two adjacent rows and resets the counter.
- **Mechanism B.** Each time a row is closed (or precharged), the memory controller flips a biased coin with a probability p of turning up heads, where $p << 1$. If the coin turns up heads, the memory controller activates one of its adjacent rows where either of the two adjacent rows are selected with equal probability ($p/2$).

(a) You set T for Mechanism A to 164 K based on the value of the Maximum Activation Count (MAC, i.e., the maximum number of times a row can be activated without inducing RowHammer errors in its adjacent rows) reported by the DRAM manufacturer. Calculate the number of bits required for counters in a memory controller which manages a single channel, 2 ranks per channel, 8 banks per rank, and 2^{15} rows per bank.

To count values up to 164 K, we need at least 18 bits ($2^{18} > 164K$) per counter.
$\therefore 18\ [\frac{bit}{row}] \times (2^{15})\ [\frac{row}{bank}] \times 16\ [bank] = 9 \times 2^{20}$ bits $= 9$ Mib

(b) How does the answer to (a) change when both the number of rows per bank and the number of banks per chip are doubled?

It will increase to 36 Mib with 2x rows and 2x banks.

(c) You profile the memory access pattern of the target system, and observe that the same pattern repeats exactly every 64 ms (the current refresh interval). Table 2 shows the number of activations for each row within a 64-ms time interval in a descending order. Given values $T = 164$ K for Mechanism A and $p = 0.001$ for Mechanism B, calculate the expected number of additional activations within a 64-ms time interval under each technique.

Row Index	**# of ACTs**
0x7332F	73 K
0x1802C	64 K
0x03F05	32 K
0x5FF02	10 K
...	...
Total	480 K

Table 2. Number of Activations for Each Row.

Mechanism A introduces no additional row activation, since no row is activated more than the threshold.

On the other hand, for Mechanism B, the number of additional activations can be modeled as a random variable X that is binomially-distributed with parameters $B(480,000,\ p)$.
$\therefore$ # of additional activations $= E(X) = 480,000 \times 0.001 = 480$

(d) How does the answer to (c) change when both the number of rows per bank and the number of banks per chip are doubled? Assume that the memory access pattern does not change.

The performance overhead only depends on the access pattern, so it will not change.

(e) What is the *common challenge* to implement the above mechanisms in the commodity systems?

This question is open ended. There could be other possible right answers.

Both Mechanisms require the information about exact mappings between row address and physical row, which is unlikely disclosed by manufacturers.

(f) How can you address the common challenge?

This question is open ended. There could be other possible right answers.

Possible solutions

- Reverse engineering
- Implementing the techniques inside DRAM modules where the mapping function is managed.
- For counter-based approach, we can block future activations on a row (instead of refreshing its adjacent rows), if the counter value of a row exceeds the threshold value.

5. VRL: Variable Refresh Latency [150 points]

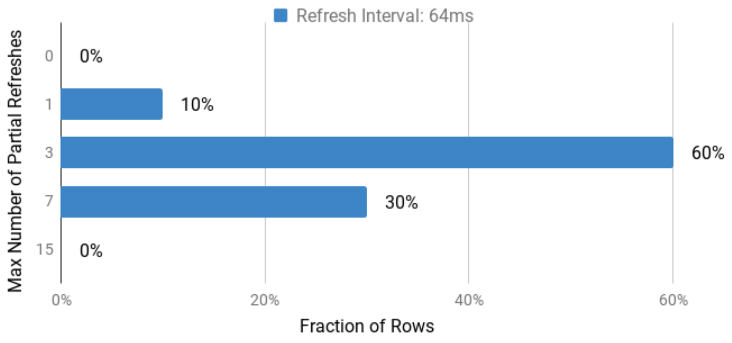
In this question, you are asked to evaluate Variable Refresh Latency[1], which is based on two key observations:

- First, a cell's charge reaches 95% of the maximum charge level in 60% of the nominal latency value during a refresh operation. In other words, the last 40% of the refresh latency is spent to increase the charge of a cell from 95% to 100%.
- Second, a 100% charged cell reliably operates even after several 95% charge restorations, but it needs to be fully charged again after a finite number of 95% charge restorations. This finite number varies from cell to cell.

Based on these observations, the paper defines two types of refresh operations: (1) *full refresh* and (2) *partial refresh*. Full refresh uses the nominal latency and restores the cell charge to 100%, while the latency of partial refresh is only 60% of the nominal value and it restores 95% of the initial charge level. The **key idea** of the paper is to apply a *full refresh* operation **only when necessary** and use *partial refresh* operations at all other times.

(a) Consider a case in which:

- Each row must be refreshed every 64 ms (i.e., the refresh interval is 64 ms).
- Row refresh commands are evenly distributed across the refresh interval. In other words, all rows are refreshed exactly once in *any* given 64 ms time window.
- You are given the following plot, which shows *the distribution of the maximum number of partial refreshes* across all rows of a particular bank.
- We define T as the time that a bank is busy serving the refresh requests in a window of 64ms. if all rows are always fully refreshed (baseline).



How much time does it take (in terms of T) for a bank to refresh all rows within a refresh interval after applying Variable Refresh Latency?

Full refresh latency = T, partial refresh latency = 0.6T.

10% of the rows are fully refreshed at every other interval:
$0.1 \times (0.5 \times 0.6T + 0.5 \times T)$
60% of the rows are fully refreshed after every three partial refresh:
$0.6 \times (0.75 \times 0.6T + 0.25 \times T)$
30% of the rows are fully refreshed after every seven partial refresh:
$0.3 \times (0.875 \times 0.6T + 0.125 \times T)$

Then, new refresh latency of a bank would be 0.695T.

[1]Das et al., *"VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency."* In Proceedings of the 55th Annual Design Automation Conference (DAC), 2018.

(b) You find out that you can relax the refresh interval of the baseline before applying Variable Refresh Latency as follows:

- 90% of the rows are refreshed at every 128ms; 10% of the rows are refreshed at every 64ms.
- Refresh commands are evenly distributed in time.
- All rows are always fully refreshed.
- A row's refresh operation costs $0.2/N$ $ms.$, where N is the number of rows in a bank.

We define *refresh overhead* as the fraction of time that a bank is busy, serving the refresh requests over a very large period of time. Calculate the refresh overhead for the baseline with the relaxed refresh interval.

At every 128ms:

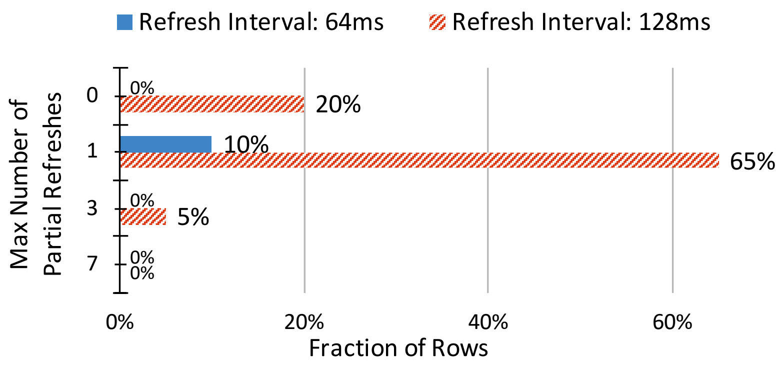
10% of the rows are refreshed twice, 90% of the rows are refreshed once.

Total time spent for refresh in a 128 ms. interval is $(0.9N + 2 \times 0.1N) \times 0.2/N = 0.22ms.$

Then refresh overhead is 0.22/128

(c) Consider a case where:

- 90% of the rows are refreshed at every 128ms; 10% of the rows are refreshed at every 64ms.
- Refresh commands are evenly distributed in time.
- You are given the following plot, which shows *the distribution of the maximum number of partial refreshes* across all rows of a particular bank.
- Fully refreshing a row costs $0.2/N\ ms.$, where N is the number of rows in a bank.
- *Refresh overhead* is defined as the fraction of time that a bank is busy, serving the refresh requests over a very large period of time.



Calculate the refresh overhead and compare it against the baseline configuration (the previous question). How much reduction do you see in the performance overhead of refreshes?

Full refresh of a row costs 0.2/N ms. Then, partial refresh of a row costs 0.12/N ms

At every $4 \times 128\ ms$:

- 20% of the rows are refreshed for 4 times:
  4 times *fully refreshed* and 0 times *partially refreshed.*
- 10% of the rows are refreshed for 8 times:
  4 times *fully refreshed* and 4 times *partially refreshed.*
- 65% of the rows are refreshed for 4 times:
  2 times *fully refreshed* and 2 times *partially refreshed.*
- 5% of the rows are refreshed for 4 times:
  1 time *fully refreshed* and 3 times *partially refreshed.*

Total time spent for refresh is:

$= (0.2N \times 4 + 0.1N \times 4 + 0.65N \times 2 + 0.05N \times 1) \times 0.2/N$
$+ (0.2N \times 0 + 0.1N \times 4 + 0.65N \times 2 + 0.05N \times 3) \times 0.12/N$

$= (0.8 + 0.4 + 1.3 + 0.05) \times 0.2 + (0.4 + 1.3 + 0.15) \times 0.12$
$= 2.55 \times 0.2 + 1.85 \times 0.12$
$= 0.51 + 0.222 = 0.732$ ms.

Then, refresh overhead is: $0.732/(4 \times 128)$

So, the reduction is $1 - (\frac{0.732}{4\times128})\ /\ \frac{0.22}{128}\ \approx\ 16.8\%$.

6. BONUS: DRAM Refresh - Energy [150 points]

A new supercomputer has a DRAM-based memory system with the following configuration:

- The total capacity is 1 ExaByte (EB).
- The DRAM row size is 8 KiloByte (KB).
- 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 every 64 ms. (Note: For each calculation in this question, you may leave your answer in simplified form in terms of powers of 2 and powers of 10.)

(a) How many DRAM rows does the memory system have?

Capacity $= 2^{60}$ and row size $= 2^{13}$, so there are 2^{47} rows.

(b) According to the specs of this hypothetical DRAM device, the memory controller should issue a refresh command (REF) at every 7.8us. How many rows is refreshed when a REF command is issued?

Because each row needs to be refreshed every 64ms, all rows need to be refreshed within 64ms. As the DRAM device receives only $64ms/7.8us = 8205$ REF commands in a window of 64ms, each REF command should refresh $2^{47}/8205$ rows.

(c) What is the power consumption of DRAM refresh? (Hint: you will need to figure out how much current the DRAM device draws during refresh operations. You can find useful information in the technical note by Micron[2]. Use the current (IDD) numbers specified in Micron's datasheet[3].
Clearly state all the assumptions and show how you derive the power numbers. You are welcome to use other datasheets as well. Make sure you specify how you obtain the power numbers and show your calculations and thought process.)

Let $P_{refresh}$ be the power consumption of the DRAM device while performing only refresh operations. To find that number, we can use $P_{refresh} = IDD5B * V_{DD}$, where $IDD5B$ is the average current consumption while the DRAM device is being continuously refreshed. Both $IDD5B$ and V_{DD} can be found in the datasheet. There are many different $IDD5B$ parameters to pick from in the datasheet depending on the types of DRAM we are assuming. If we assume DDR3-1066 with $IDD5B = 160mA$ (as can be seen in Table 19 in the datasheet), $P_{refresh} = 160mA * 1.5V = 240mW$.

(d) What is the total energy consumption of DRAM refresh during a refresh cycle? And during a day?

$Energy = Power * Time$. $Power$ is $P_{refresh}$ from part (c).
If $Time$ is 64ms (i.e., a refresh cycle), $Energy$ is $E = 240mW * 64ms = 1.54 * 10^{-2}J$.
If $Time$ is a day (i.e., $T = 24hrs * 60mins * 60secs$), $Energy$ is $E = 2.07 * 10^{4}J$.

[2] Micron, "TN-47-04: Calculating Memory System Power for DDR2 Introduction"
[3] Micron, "1Gb: x4, x8, x16 DDR3 SDRAM Features"