**Score: \_\_\_\_\_**

**MA4 – Memory Hierarchy & Processor Cache**

**Activities**

COMP256 – Computing Abstractions

Dickinson College

Spring 2023

Prof. Grant Braught

**Name:**

Today’s class began talking about performance improvements to the stored program architecture. In particular, today focused on the different types of memory that are used in modern machines and how and why they are arranged into a hierarchy. In addition, the concept of caching and how it applies to the processor cache were introduced. In the following activities you will apply some of the ideas presented, fill in some additional details about the size, speed and cost of elements of the memory hierarchy. You will also gain some additional insight into why the processor cache works as well as how big of a performance improvement it actually provides.

They are not required viewing, but if you would like some additional perspective on today’s topic or some more information as you work through the exercises the following are good resources:

* Matthew Watkins’ video *Memory Hierarchy Introduction* gives an introduction to the memory hierarchy including discussions of cost/performance tradeoffs and locality of reference:
	+ <https://www.youtube.com/watch?v=_kZY4orPQW0> (10:08)
* Carrie Ann discusses memory and processor cache in the first 5:35 seconds of the Advanced CPU Designs video from the Crash Course Computer Science series:
	+ <https://www.youtube.com/watch?v=rtAlC5J1U40> (5:35)

**Memory Units:**

1. In class we saw that memory and storage sizes are sometimes specified using powers of 10 (e.g. Disk Space) and sometimes using powers of 2 (e.g. Main memory and Cache). These two different values are approximately the same, but not exactly the same.

a. Complete the table below with the values for these units using both powers of 10 and powers of 2. Use the first row as an example.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  | **Memory Size** | **As a power of 10** | **As a whole number** | **As a power of 2** | **As a whole number** |  |
|  | **Kilobyte (KB)** | 103 | 1,000 | 210 | 1,024 |  |
|  | **Megabyte (MB)** |  |  |  |  |  |
|  | **Gigabyte (GB)** |  |  |  |  |  |
|  | **Terabyte (TB)** |  |  |  |  |  |
|  |  |  |  |  |  |  |

b. It is worth noting that because of the differences above, you will sometimes see abbreviations such as MiB (mebibyte) or GiB (gibibyte) to specifically refer to the value that is a power of 2.

Are MiB’s and GiB’s larger or smaller than MB and GB?

🔑 2. It is helpful to have an idea of the relative size of the different measurements of memory size. Complete the table below to get a feel for these sizes. **Express your answers both as a power of 2 and as the whole number value of that power of 2**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  | **As a power of 2** | **As a whole number** |  |
|  | How many KB in a MB? |  |  |  |
|  | How many MB in a GB? |  |  |  |
|  | How many GB in a TB? |  |  |  |
|  | How many MB in a TB? |  |  |  |
|  | How many KB in a TB? |  |  |  |
|  |  |  |  |  |

**The Memory Hierarchy:**

🔑 3. Tradeoffs between capacity, cost and speed affect the amount of each type of memory or storage that a computer system will have at each level of the memory hierarchy. Complete each of the sections below that highlight these tradeoffs.

a. Fill in the middle column of the table below with “are faster than” or “are slower than” to compare the typical speeds of the different types of memory/storage.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | Registers |  | Main Memories |  |
|  | Caches |  | Registers |  |
|  | Hard Disk Drives |  | Main Memories |  |
|  | Solid State Drives |  | Hard Disk Drives |  |
|  |  |  |  |  |

b. Repeat part a, but this time fill in the center column with “costs more than” or “costs less than” to compare the typical cost per bit of the different types of memory/storage.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | Registers |  | Main Memories |  |
|  | Caches |  | Registers |  |
|  | Hard Disk Drives |  | Main Memories |  |
|  | Solid State Drives |  | Hard Disk Drives |  |
|  |  |  |  |  |

c. Repeat part a, but this time fill in the center column with “are larger than” or “are smaller than” to compare the typical capacities of the different types of memory/storage.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | Registers |  | Main Memories |  |
|  | Caches |  | Registers |  |
|  | Hard Disk Drives |  | Main Memories |  |
|  | Solid State Drives |  | Hard Disk Drives |  |
|  |  |  |  |  |

4. You may have noticed the terms SRAM and DRAM in the memory hierarchy diagrams in the class slides. SRAM and DRAM are two different ways to construct memory using electronic components. SRAM stands for Static RAM and DRAM stands for Dynamic RAM. While we do not yet know exactly what these things are, we can infer a few things about them based on what we do know and where they are used in the memory hierarchy. Complete the table below by filling in the cells in the table below with SRAM or DRAM.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Is faster |  |  |
|  | Is less expensive |  |  |
|  | Is used for larger memories |  |  |
|  |  |  |  |

🔑 5. We have seen that different types of memory/storage in the hierarchy can be classified along different dimensions. For example, by persistence (volatile or non-volatile), or by access type (random, direct, or sequential).

Classify each of the following types of memory or storage as **volatile** or **non-volatile**, whichever fits best. **Give a brief justification for each of your answers.**

 a. Main Memory (RAM)

 b. A Computer Backup Tape

 c. A classroom whiteboard

 d. The notes that you take during a class

🔑 6. Now, classify each of the following types of memory accesses as **random access**, **direct access** or **sequential access**, whichever fits best. **Give a brief justification for each of your answers.**

 a. Retrieving the value from a CPU Register for a computation

 b. Reading a file from a Computer Backup Tape

 c. Getting the book “Frankenstein” from a shelf in the library

 d. Finding the 2 of diamonds in a shuffled deck of cards.

 e. Thinking about a memory in your mind.

7. The type of memory access being performed will affect the speed at which the requested information can be accessed. Let’s assume that there is a particular type of storage that can read data using any of the three types of accesses (random, direct or sequential).

a. On average which type of access will take the least amount of time to read the data? Briefly explain your answer.

b. On average which type of access will take the most amount of time to read the data? Briefly explain your answer.

**Processor Cache:**

🔑 8. In class we used a plumber as a metaphor to explain the concept of processor cache. This question refers to that metaphor.

a. Fill in the second column in the table below with the element of the metaphor that corresponds to the corresponding element of a computing system.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Main Memory |  |  |
|  | Cache |  |  |
|  | CPU Registers |  |  |
|  | ALU |  |  |
|  |  |  |  |

b. Explain in a few sentences how the plumber can use the metaphor element that you identified as the cache in part a to leverage ***temporal locality*** to work more quickly.

c. Explain in a few sentences how the plumber can use the metaphor element that you identified as the cache in part a to leverage ***spatial locality*** to work more quickly.

9. Consider the following Java function:



The execution of this code will benefit from the presence of a processor cache due both to spatial and temporal locality. Fill in the “Locality Type” column below to with either “Temporal” “Spatial” to indicate the type of locality that would apply to the access.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Access** | **Type of Locality** |  |
|  | The access to values[i] on line 4 |  |  |
|  | The access to max on line 4 |  |  |
|  | The access to values.length in the loop condition on line 4 |  |  |
|  | The access to i in the loop update (i++) on line 4 |  |  |
|  | The access to values[i] on line 5 |  |  |
|  |  |  |  |

**The Cache Abstraction is Imperfect Too:**

With cache, like all of other abstractions we have studied, the idea is that we can pay attention to the important information, while forgetting about (or never knowing) the unimportant details. Here the important details are the memory accesses. We can simply think of accessing the main memory as getting a value from an address without worrying about or even knowing what the cache is doing. The hardware provides an abstraction that hides from us all of the details of checking if a value is in cache, dealing with cache misses and the copying of surrounding data and instructions to the cache. But as we’ve seen before, our abstractions are often imperfect and some of the “unimportant” details show through. The next question explores this with respect to processor cache.

10. Examine the program at:

* [https://repl.it/@braughtg/2dArrayCache#Main.java](https://repl.it/%40braughtg/2dArrayCache#Main.java)

a. This program declares a 2d array. How many rows and columns are in the array?

b. This program has two functions downEachColumn and acrossEachRow. These functions both have the same effect on the values in the array. As their names suggest, they just do their work in a slightly different order.

Study the functions and then complete the table below indicating the number of times that the arr[row][col]++; statement is executed by each of these functions.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Function** | **Number of times arr[row][col]++;****is executed.** |  |
|  | downEachColumn |  |  |
|  | acrossEachRow |  |  |
|  |  |  |  |

c. Which function, downEachColumn or acrossEachRow do you think will require less time to do its work? Or will they require the same amount of time. Briefly justify your answer.

d. Run the program a few times. Which function accomplishes its work more quickly? Approximately how many times faster is it than the other?

Surprised? I know I was!!

Now… of course the question is why? To understand that, recall that Java stores 2d arrays as a 1d array of 1d arrays as shown here:



Image from <https://www.therevisionist.org/software-engineering/java/tutorials/2-dimensional-array/>

Note that arrays are stored in contiguous memory locations, that is if x[0] is stored at address 1000 then x[1] would be stored at address 1001. Similarly, if x[0][0] is stored at address 2000, then x[0][1] would be stored at address 2001. Similarly for x[1][0] and x[1][1], and so on.

🏆 11. Now, thinking about how arrays are stored and how cache works explain based on the principles of locality why the function you identified in #10d is faster than the other one even though they do the same amount of work.

Optional: To help me improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.