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

**MA5 – Parallelism and Pipelining**

**and Writing in the Discipline (WiD)**

**Activities**

COMP256 – Computing Abstractions

Dickinson College

Spring 2023

Prof. Grant Braught

**Name:**

The main part of this homework will be the first part the Writing in the Discipline (WiD) assignment for this course. In class we used a plumber and his toolbox as a metaphor to explain the concept of processor cache. For the WiD assignment you will create your own metaphor for processor cache and use it to explain caching concepts such a cache hit/miss and the principles of spatial and temporal locality.

After completing a draft of your WiD assignment there are a few questions that ask you to think about today’s topic. In class we introduced some additional performance improvements based upon new ideas for CPU design. These included instruction pipelining, super scalar processors and multicore CPUs. While the implementation of each of these improvements is mind-bendingly complex, underneath it all is really just a collection of millions of transistors arranged into more complicated logic circuits. But rather than trying to understand these topics at that level of abstraction, we have considered them at a higher level of abstraction using metaphors. If you are particularly interested in these topics there are some videos and readings linked at the end of this assignment.

**Writing in the Discipline (WiD):**

COMP256 is part of the Writing in the Discipline (WiD) thread for the computer science major. In computer science the WiD goals for the major are divided across the core courses in the major. In each course you will complete a writing assignment and add it to your WiD portfolio. By completing all of the WiD assignments across all of the core computer science courses and collecting that work into your WiD portfolio you will satisfy the College’s WiD graduation requirement.

Each of the core courses has its own WiD learning goal that is related to the content of the course. For COMP256 the WiD learning goal is:

* “Be able to use metaphor and analogy to explain complex technical concepts.”

This is a suitable goal for COMP256 as metaphors and analogies are themselves forms of abstraction.

The WiD component of COMP256 is spread across three assignments.

* In this assignment you will create a first draft of your writing assignment.
* In the second assignment (in a few weeks) you will read several of your peer’s drafts and provide feedback. In turn, you will also receive feedback from your peers.
* The third, and final assignment (a few weeks later) you will revise your draft to address the feedback received from your peers.

**A Processor Cache Metaphor:**

In class we used a plumber and his toolbox as a metaphor to explain the concept of processor cache. Your WiD assignment is to invent a new metaphor of your own and use it explain caching and the associated concepts and principles.

1. Write a complete polished draft that fully addresses the prompt given below:

**In about 500-800 words, develop a metaphor of your own that can be used to explain caching.** You must clearly identify and explain how the elements of your metaphor play the roles of main memory, cache, registers and ALU. In addition, you must use your metaphor to explain in detail the ideas of cache hits and cache misses and how the concepts of spatial and temporal locality contribute to cache efficiency.

**\*\*\* You must invent your own metaphor. \*\*\***

**\*\*\* You may not use the Plumber metaphor for this assignment. \*\*\***

2. When you have a complete polished draft do the following:

a. Ensure that your name DOES NOT appear in the document.

* + This is important because we will use a *double-blind* feedback process through which you will provide feedback to your classmates on their work and receive feedback on your work. The process is called double-blind because you will not know who you are providing feedback to, and you also will not know who has provided feedback to you. This is a process that is frequently used in academic and scientific publication to help remove bias.

b. Convert your draft to PDF and name the resulting PDF file as follows:

* + ***username*-256WiD-draft.pdf**
		- Replace *username* with your Dickinson username. That way I will be able to tell which file is yours. But I will also be able to easily rename everyone’s files to be anonymous before I share them with your peers.

3. When you have converted your draft to a pdf document with the filename as described above you will submit it to your WiD repository on GitHub. To do this:

 **a. Create a folder named COMP256 in your WiD repository on GitHub.**

**b. Upload your pdf document to the COMP256 folder you just created.**

If you find it helpful, the following screencast demonstrates how to create a directory and upload a document into your WiD repository:

* <https://web.microsoftstream.com/video/f3f79772-ba76-48f7-8f92-bc4514a85322> (2:13)

**Processor Pipelining:**

🔑 4. In class the idea of processor pipelining was introduced using the process of doing laundry as a metaphor. The idea “pipelining” work is not unique to computer processors and has been used in many other real-world applications. Describe another situation in which “pipelining” has been used to improve the performance of a system.

5. We have seen that while pipelining can improve the performance of a processer, *pipeline hazards* can interfere and limit the improvement that is possible. In class we discussed the three types of pipeline hazards, resource hazards, control hazards and data hazards. Using your example from the prior question, give and briefly explain an example of one of these types of hazards.

**Super Scalar Processors:**

🔑 6. The key idea behind super-scalar processing is having multiple functional units, where each unit can be completing a task in parallel (i.e. simultaneously) with the others. There are often multiple copies of multiple different functional units. For example, there may be several ALUs that can perform integer two’s complement operations and several more ALUs that can perform floating point arithmetic. Like caching and pipelining the ideas behind super-scalar processing are not unique to computer processors. Thus, there are many examples that can be used as metaphors here as well.

Describe in a few sentences an example where the ideas behind super-scalar processing have been used in non-computing applications.

**Multicore Processors:**

We briefly discussed the multicore processor below in today’s class:



The following questions dig a little deeper into the operation of multicore processors.

🏆7. In a multicore processor each processor core is essentially a full CPU with its own registers, ALU and control unit. The control unit in each core of course has its own program counter and its own instruction register. Thus, as each core goes through its fetch/decode/execute cycle it can be fetching instructions from different parts of the main memory. Thus, the different cores could be running different parts of the same program or each core could be running a different program.

Use these ideas to explain why it is advantageous for each core to have its own independent L2 cache as pictured above. Hint: Remember that cache works based on the principle of locality.

🏆8. From your answer to the previous question, it should not be that surprising that each core has its own independent L1 cache as well. This is just a smaller faster more expensive cache that is located physically closer to the registers than the L2 cache, and thus can be accessed even faster.

But notice that the L1 cache for each processor has been divided into two separate caches, one for data and one for instructions. These caches behave the same as the other caches in that they hold copies of things we have used recently or that we are likely to need soon. The difference is that instruction cache will only hold program instructions and the data cache will only hold data that is used by running programs.

Explain why it is advantageous to have separate caches for instructions and data? Hint: If it’s about cache… it’s always about locality!!

**The Complexity/Cost vs Performance Tradeoff:**

All of the performance enhancing designs that we have learned about require a mind-bending amount of extra circuitry to make them work. For example, pipelining requires extra circuits to synchronize the stages and account for hazards. Super-scalar processors require circuits that can decide when to reorder instructions and then how to put the results back together again. Multi-core processors require duplication of the circuits that implement register banks, ALUs and control units. All of this extra circuitry adds both complexity and cost to the systems that use these designs.

However, in return for all the added complexity and cost the system is able to provide greater performance. This creates a ***Complexity/Cost vs Performance trade off*** where greater performance can be achieved by adding complexity and cost, or conversely cost and complexity can be reduced by sacrificing performance.

🔑 9. Briefly explain in a few sentences of your own words how processor cache is also an example of the Complexity/Cost vs Performance tradeoff.

**Final Thoughts:**

We can now see that even thinking about the fetch/decode/execute cycle of a machine is an abstraction. If we think about instructions being fetched, decoded and executed one-by-one, as is done by the Knob & Switch computer, we have a way of understanding what the result of running a machine language program will be. That relevant information allows us to be able to write or study a machine language program and know what it will do. What is not relevant at that level of abstraction is exactly how the program is actually executed. It may be run though a 3-stage or a 15-stage pipeline. Its instructions may be delayed or even executed out of order to minimize pipeline hazards. It may even be that multiple instructions are run in parallel on multiple functional units in a super scalar processor, or on multiple processor cores. On a modern machine, it is likely that all of those things are happening! It is really quite astonishing to imagine all of that happening each time you run a program. But thankfully, the abstraction allows us to forget about those details and just imagine our program running in a simple fetch/decode/execute cycle.

**Going Above and Beyond:**

Today’s class just scratched the surface of some of the ways that processor designs have leveraged new design ideas and taken advantage of the unimaginably large number of transistors that Moore’s law has made available. We touched a little more on that in today’s activities but many of these topics are beyond the scope of this course. None of the following is required, but if you these topics interesting you might find the following sources interesting as well. You might also consider seeking out a course on Computer Architecture.

Carrie Anne’s provides additional perspective on the performance improvements that we have seen, as well as a few others in the video below that is part of the Crash Course Computer Science series:

* *Advanced CPU Designs*
	+ <https://www.youtube.com/watch?v=rtAlC5J1U40> (12:22)

When multiple cores and multiple level caches are used together an interesting problem called Cache Coherence arises due to write operations. For example, imagine one core writes the value of a variable to its L1 data cache. Now imagine another core is executing a different part of that same program. How will it know that this variable has been updated? Ensuring that all cores can efficiently access up to date information is the Cache Coherence problem. It is not required, but if you are curious there are a few good videos on Cache Coherence from the Georgia Tech High Performance Computer Architectures course:

* *Cache Coherence Problem*
	+ <https://www.youtube.com/watch?v=TMJj015C93A> (3:21)
* *How to Get Cache Coherence*
	+ <https://www.youtube.com/watch?v=In8RZg345pM> (5:18)

If thinking about design improvements like these is something you find fascinating, you might enjoy reading the article below by Jason Robert Carry Patterson of Lighterra. It discusses many of the biggest ideas in processor design and is, though not an easy read, written at a level that is pretty accessible.

* *Modern Processors: A 90-Minute Guide!*
	+ [http://www.lighterra.com/papers/modernmicroprocessors](http://www.lighterra.com/papers/modernmicroprocessors/)

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.