ENGINEERING

UIC

INTERNSHIP REPORT 2019

CO-OP AIDE / GPIP Intern
Electrical & Computer Engineering

The University of Illinois at Chicago

Academic Supervisor
Professor
Zhichun Zhu
zzhu@uic.edu

Supervisor/Mentor
Ph.D. Student
Muhammad Rafique
mrafiq2@uic.edu

Submitted By
Jim Palomo
jpalom6@uic.edu

Date of Submission: August 7, 2019

UNIVERSITY OF ILLINOIS
AT CHICAGO

ELECTRICAL
AND
COMPUTER
ENGINEERING
COLLEGE OF
ENGINEERING
UIC

Preface & Acknowledgement

For 8 weeks from June 20, 2019, to August 7, 2019, I did an internship at the University of Illinois at Chicago, a public research university located in the heart of Chicago. The University of Illinois at Chicago is a Research I university. At this internship, I worked under the COE Department of Electrical and Computer Engineering. This was my very first internship and relatable work experience in the field.
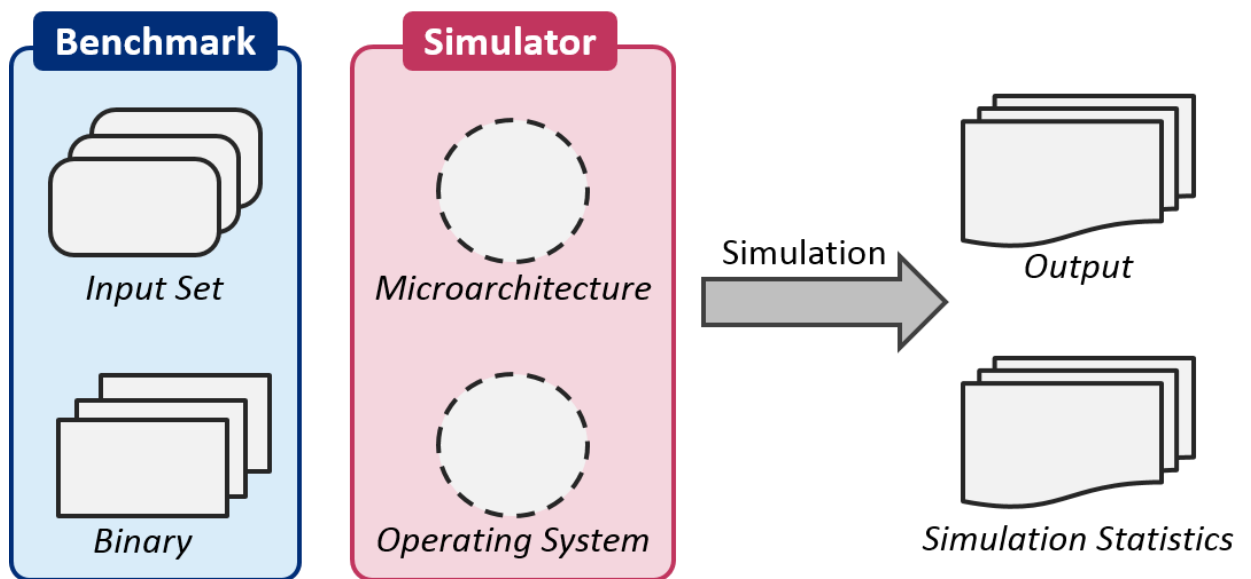

Figure 0.1 - gem5 diagram for simulated output, copied from [1]

The assigned project that I worked on is an open simulation platform for computer system architecture, called gem5. The main objective of gem5 is to create simulated computers and analyze data that is outputted. The possibilities that gem5 produces allows for computer architecture researchers is endless. Data that is gathered includes but is not limited to runtimes, microarchitecture analysis, and event-driven simulations. Computer architecture is a subfield in computer engineering and is something that was both familiar and new to me. Over the course of my internship, I gained valuable knowledge in the field of computer architecture and in turn enhanced my professional skills. I learned the

necessary tools to approach this project by self-learning and with the assistance of my mentor.

I truly value the opportunity that I was given and those who have helped me get here. I am thankful for Muhammad Rafique, my supervisor/mentor at this internship. Mr. Rafique has guided me throughout this internship. Mr. Rafique is currently pursuing a Ph.D. in computer engineering. Mr. Rafique's research topic correlates with computer architecture and thus, he works on gem5 simulations. Mr. Rafique gave me the necessary tools to assist me with this project. Mr. Rafique gave me beneficial one-to-one meetings that ended up fixing issues that I had with my code, an insight in the field and from a pursuing Ph.D. student, and much more. In regards to thank those that have helped me along, I can not forget to acknowledge Professor Zhichun Zhu, a professor who specializes in her research on computer architecture, parallel and distributed computer, and performance modeling and evaluation. Ms. Zhu has a Ph.D. in CS from the College of William and Mary and B.S. in CE from Huazhong University of Science and Technology. Ms. Zhu was the individual who assigned me to Muhammad and the project at hand.

I have gained a profound amount of knowledge from this internship. The knowledge that I acquired will benefit myself and those around me for the professional world. Moreover, I would like to give my gratitude to those that have had an impact on offering this opportunity. Without them I would not have had this opportunity: Yossi Hernadez, Mona Hurt, Rosemarie Coppola-Conroy, College of Engineering, Department of Electrical and Computer Engineering, and the University of Illinois at Chicago.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

COE   College of Engineering

ECE   Electrical & Computer Engineering

CS   Computer Science

CE   Computer Engineering

UIC   University of Illinois at Chicago

OOP   Object-Oriented Programming

CPU   Central Processing Unit

JSON   JavaScript Object Notation

gem5 Resources

Contributors	https://dl.acm.org/citation.cfm?doid=2024716.2024718
Book	http://learning.gem5.org/book/index.html
My gem5 Repository	https://github.com/JimPalomo/gem5-dev

Chapter 1

Introduction

When individuals think about computers, they think of the box-shaped pc. However, the concept of computers is everywhere, right in the palm of our hand. Many individuals carry around a computer known as a smartphone. This is a miniaturized computer that is more powerful than the computers in the 1960s that launched the Apollo-11 to the moon [2]. The reason for the technological evolution is due to Moore's Law, the principle for which the overall processing power of computers doubles every two years [3]. The way Moore's Law works is due to computational research on the architecture of computers. Computer architecture research is conducted in many ways. However, nowadays research the architecture of computers is done through simulations using software.

1.1 Introduction to gem5

The software that was used throughout my internship at UIC is gem5, a full-blown computer system simulator platform which includes processor microarchitecture. This software is mainly used by researchers in the computer architecture subfield. There is a distinguishable amount of capabilities that gem5 possesses. gem5 allows for components to be created, extended, rearranged, and parameterized. Moreover, gem5 allows the ability to simulate a series of timed events known as event-driven programming. gem5 has the option to simulate more than one computer [4], [5].

1.2 History of gem5

The history of gem5 was not one but two separate stories combined. gem5 was created by two different companies, Michigan m5 & Wisconsin GEMS. Together these two companies formed the simulation platform known as gem5. Key contributors include but

are not limited to Nathan Binkert, Bradford Beckmann, Gabriel Black. A full comprehensive list of contributors is available under gem5 resources [6].

Chapter 2: Getting Started

2.1 Learning Python & C++

The way I learned the operations of gem5 did not start by diving straight into gem5. I began by learning the fundamentals of Python and C++. This lasted from June 20th - July 2nd, almost 2 weeks. Within this course of time, I covered the very basics, syntax, and how information is passed for both of these programming languages. The following are basic concepts that I covered: variables, data types, strings, user input, arrays, functions, return, conditionals, loops, 2D arrays & nested loops, and pointers. The main objective was to learn these languages before starting gem5 but I had to gain a deeper understanding of the infrastructure of gem5. Therefore, I also had to learn about object-oriented programming for both languages. The following are OOP concepts that I covered: classes & objects, constructor & destructors, object functions, getters & setters, inheritance, access modifiers, member initialization, composition, this, operator overloading, protected members, decorators (classmethod, staticmethod, etc), dunder methods, and polymorphism.

When learning these languages, it was not always a smooth learning experience. After learning the basics from functions, loops, pointers, and more, I had some issues moving forward. The challenges that I faced existed when I was working with object-oriented programming. OOP is a topic that I have never covered so learning every detail was necessary to be able to understand gem5. Within OOP, I had a hard time covering the specific uses of public, private, and protected access modifiers. Access modifiers are used with inheritance by which a derived class obtains the base class's attributes such as functions and variables. To comprehend this issue, I used online resources such as

YouTube. The channels that I obtained information from was from Corey Schafer & Bucky Roberts (thenewboston). Both of these channels are beginner-friendly and help explain concepts clearly. I learned Python from Corey Schafer and C++ from Bucky Roberts [7], [8].

From their lessons, I figured out the specific uses of access modifiers. When using the public modifier, methods and variables can be accessed by the base class, subclass, and outside the class. In other words, the instances can be accessed anywhere under the public access modifier. Furthermore, private access modifiers can only be accessed by the base class. The subclass and outside the class do not have the authorization to use the instance methods or variables. Protected methods or variables can only be accessed by the base class and derived subclass through inheritance.

Modifier	Class	Subclass	World
Private	Y	N	N
Protected	Y	Y	N
Public	Y	Y	Y

Table 0.1 - Table of access modifiers & their capable inheritance, copied from [9]

Chapter 3: Starting gem5

3.1 Simple configuration script

Eventually, after establishing the fundamentals of Python, C++, and object-oriented programming, I was able to begin teaching myself gem5. Before I could move further into the development of gem5, I had to set up gem5. I was able to gather the required dependencies and build a development environment with the help of Mr. Rafique. Once the setup was finished, I worked on creating a simple configuration script. In this script, I made a single-core computer. The simple configuration script was a program that is created to introduce new users to gem5. I constructed numerous components for the simulation. First of all, I created a system object and began establishing the clock domain by editing the frequency & voltage the clock domain receives. The frequency that I set the clock domain was 1 GHz. I then developed the system's memory to 512 MB, a memory bus, CPU, memory controller, threads, interconnects, and the simulated test file.

The system memory was set to timing mode which is one of the three modes that are used in gem5. Timing mode simulates the most realistic output data by the idea of detailed access. The other two modes are atomic and functional. Atomic is used for experimentation by accessing ports and completing requests faster. Finally, the functional mode is used for debugging purposes. In general, those who use gem5 use timing mode as a default setting [10].

The instructions that were used to set up gem5 can be found under gem5 resources Once I finished creating the simple configuration script, after a few fixed typos, I ran the program by rebuilding gem5. For future users, rebuilding gem5 is a common process when creating new configuration files/SimObjects. SimObjects are simulated objects that a user creates for testing. SimObject is a base class which individuals can use to derive a

subclass and create their own simulated objects. For the simple configuration script, I ran
a premade hello world file provided by gem5 and it ran as expected.


```
Global frequency set at 1000000000000 ticks per second
warn: DRAM device capacity (8192 Mbytes) does not match the address range assign
ed (512 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
Beginning simulation!
info: Entering event queue @ 0.  Starting simulation...
Hello world!
Exiting @ tick 454507000 because exiting with last active thread context
```

Figure 0.2 - Running simple configuration script, copied from [11]

Subsequently, after running the simple configuration script, a diagram of the simulated
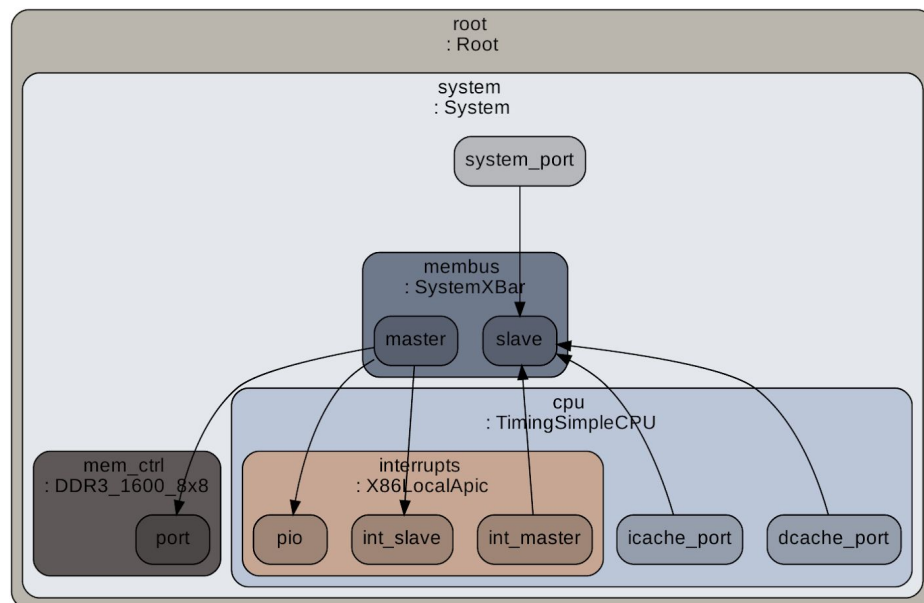architecture under the m5out directory. The diagram is shown in Figure 0.3 below:

**Figure 0.3 - Diagram of simulated architecture made from the simple configuration
file, copied from [12]**

3.2 Adding cache to the configuration script

I have only scratched the surface of gem5's capabilities thus far. To gain a deeper insight toward gem5, I continued learning through the provided gem5 book [13]. Therefore, I moved onto adding cache to the simple configuration script. There are different kinds of caches that were incorporated. The two caches that I assembled are a L1 cache (instruction, data) and a L2 cache. The difference between these caches is that the L1 cache has less storage capacity than that of L2 but is faster than the L2 cache. These caches are connected to memory buses that address and controls the flow of data. The purpose of these caches makes the computing power of the computer faster. Allowing data to travel at a faster rate due to it being a buffer between the CPU and RAM. In other words, caches allow for a faster data transfer than that of RAM. Based on what has been said, we can use Figure 0.2 and 0.5 to prove that adding caches improves the speed of which data is processed. Figure 0.2 shows the single-core computer's run time as 454507000 ticks compared to Figure 0.5 with a run time of 56435000 ticks. This was recorded when running the premade hello world file provided by gem5.
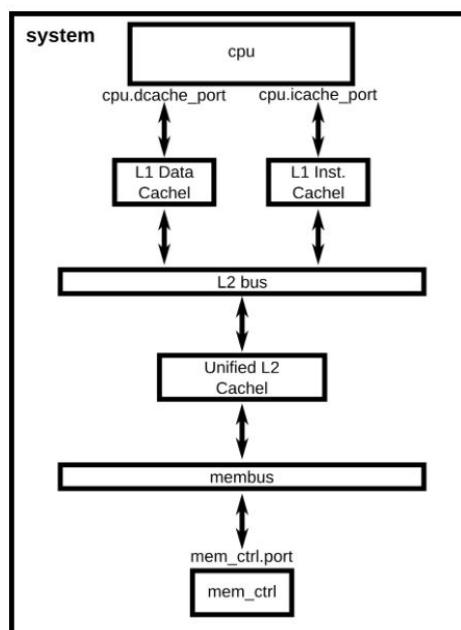

Figure 0.4 - System with two-level hierarchy, copied from [14]

11

```
Global frequency set at 1000000000000 ticks per second
warn: DRAM device capacity (8192 Mbytes) does not match the address range assign
ed (512 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
Beginning simulation!
info: Entering event queue @ 0.  Starting simulation...
Hello world!
Exiting @ tick 56435000 because exiting with last active thread context
```

Figure 0.5 - Running simple configuration file with caches, copied from [15]
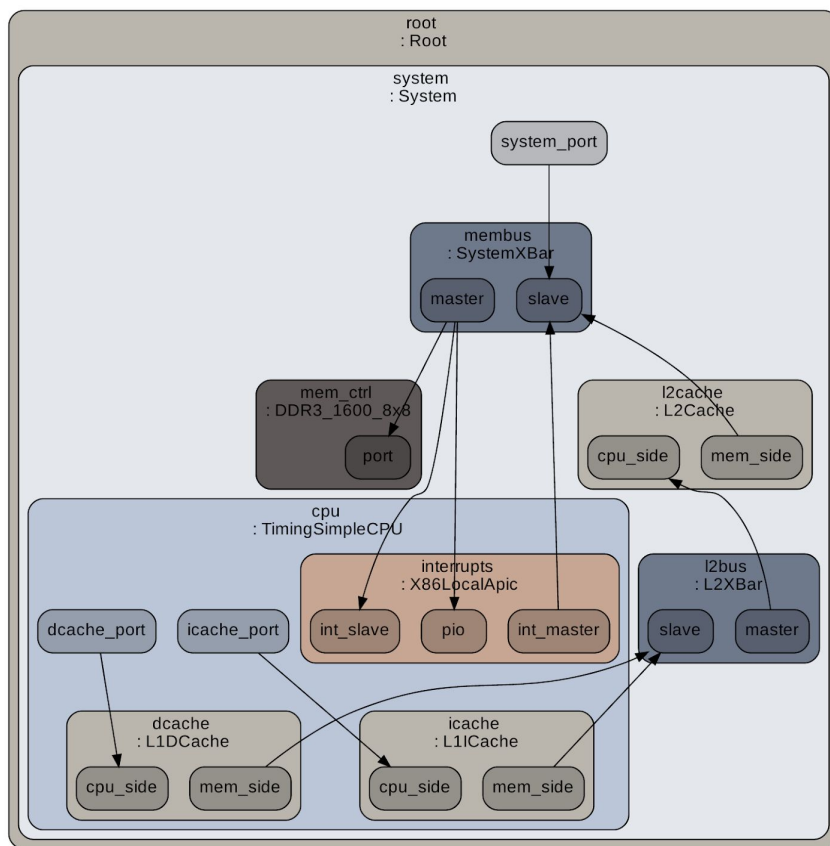

Figure 0.6 - Addition of a L1 & L2 Cache, copied from [16]

3.3 Statistics, outputs, and using default configuration scripts

The next topic that I covered did not require any coding but more of an understanding. To understand this topic, understanding gem5's purpose is required. The purpose of gem5 is to simulate computer systems and to analyze the data that is outputted. The data that is recorded can be found under m5out directory with files: config.ini, config.json, and stats.txt. Each of these files contains a purpose when it pertains to the use of data. For example, config.ini lists all SimObjects that have been created with their parameters. Config.json is similar to config.ini but in JSON format. Stat.txt contains the statics that the simulation recorded. These files are available for research use.

Moreover, the proceeding section to statistics and outputs to understand the default configuration scripts. In the gem5 directory, there are two default configuration files, one for each simulation model. These modes are simulated emulation (se_mode) and full simulation (fe_mode). The files provided are se.py and fs.py which both contain a guide to simulating in each mode. These files contain a whole scope for each mode. The only action that the user would have to conduct is to change is the parameters. Using these files allow for quick access to simulating computer architecture; thus, concluding Part I from the gem5 book [13].

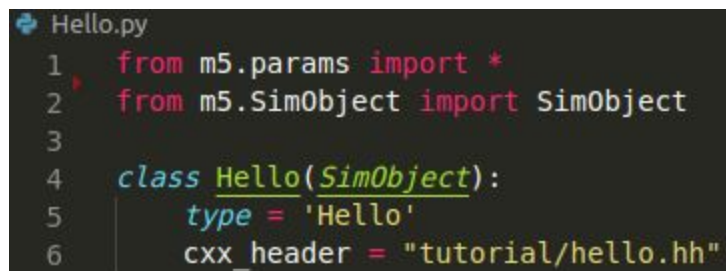# Chapter 4: Modifying and Extending gem5

4.1 Setting gem5 development environment with style guidelines

When continuing gem5, there was a point where I was given a guideline to follow for formatting purposes. Since I did not plan nor was it required for the internship to contribute to gem5, I did not follow the style guidelines. I received approval from my mentor, Mr. Rafique since my main objective was to learn gem5.

4.2 Creating a simple SimObject

By now, I am going to be referring to Part II of the gem5 book. This is the part where it is very important to pay attention to. When I worked on this part, I had an abundance of issues, most were minor but could make or break the simulation. My issues arose in the event-driven portion of Part II. In regards to creating a simple SimObject, I did not have any problems. SimObjects are very simple and straightforward to establish. However, knowing how to develop SimObjects requires an understanding of linker files.

In gem5, creating a SimObject requires creating five individual files. One file is created to instantiate the object which is known as a configuration file. The instantiation of a SimObject is done in python. In this file, the object is created with a specified name and C++ header (cxx_header), see Figure 0.7. The name of the header is used to create the header file in C++. The purpose of a C++ header file is to construct a class from SimObject which declares user-added functions and variables under access specifiers (public/private/protected), see Figure 0.8. On the other hand, implementation files are used to write the code for the functions declared in the header files, see Figure 0.9.

Figure 0.7 - Creating a configuration file, copied from [18]

Figure 0.8 - Creating a header file, copied from [19]

Figure 0.9 - Creating an implementation file, copied from [20]

After these three files are created, configuration, header, and implementation, there are

still two necessary files. One of those files is called a sconscript. Since gem5 is built on

SCons, an open-source build tool, a sconscript is necessary to compile the C++ files and

to parse the Python file [17]. Within the sconscript, the SimObject and implementation

files are stated. Once the following files above are created, gem5 will need to be rebuilt

with scons (e.g. scons build/X86/gem5.opt) to compile and link the files for gem5 to run,

see Figure 1.0. Finally, we can create the final file, another configuration file known as a

run script. This is the file that will be run on the terminal when testing the new SimObject. In this script, it is necessary to include an instantiation of the root object which is required for all gem5 instances. Then we instantiate the new SimObject that is created. Next, we call the instantiates on the m5 module to enable the simulation to run. At last, we can call this run script when building gem5 (e.g. build/X86/gem5.opt src/tutorial/run_hello.py), see Figure 1.1. A more in-depth explanation can be found in the gem5 booklet [13].

```
≡ SConscript
1    Import('*')
2
3    SimObject('Hello.py')
4    Source('hello.cc')
```

Figure 1.0 - Creating a Sconscript, copied from [21]

```
run_hello.py
1    import m5
2    from m5.objects import *
3
4    root = Root(full_system = False)
5
6    root.hello = Hello()
7
8    m5.instantiate()
9
10   print("Beginning simulation!")
11   exit_event = m5.simulate()
12   print('Exiting @ tick %i because %s' % (m5.curTick(), exit_event.getCause()))
```

Figure 1.1 - Creating a run script configuration file, copied from [22]

4.3 Debugging gem5

Debugging in programming is used to find errors in the code whether it be logical or computer compiled. In this section, I learned how to debug gem5 by creating debug flags.

Debug flags help identify errors in a simulation by knowing what the simulation is doing
at specific times. I had no issues developing the debug flags.

Figure 1.2 - Debug flags example running simple config script, copied from [23]

4.4 Event-driven programming

Event-driven programming is a notable feature that gem5 supports. When working with
event-driven programming, the simulation triggers events that are queued at specific
times. While I was working on event-driven programming, I encountered a lot of
problems. This is due to the addition of problems being stacked upon another. For
example, since I was unable to figure out the error after researching online, reviewing
through each file, looking over error codes, I started over numerous times. Due to me
starting over, I had multiple files that were duplicates of one another. Therefore, when I
compiled gem5, the error got longer and kept telling me the file/object has already been
created elsewhere. After talking with Mr. Rafique, I deleted the duplicate files and tried
to debug from the source files. After reviewing my source files, I had forgotten a few key
components. The components I forgot to add were a few debug flags that I did not
incorporate. Once the debug flags were implemented, my program worked as intended.

```
Beginning simulation!
info: Entering event queue @ 0.  Starting simulation...
    100: hello: Hello world! Processing the event! 9 left
    200: hello: Hello world! Processing the event! 8 left
    300: hello: Hello world! Processing the event! 7 left
    400: hello: Hello world! Processing the event! 6 left
    500: hello: Hello world! Processing the event! 5 left
    600: hello: Hello world! Processing the event! 4 left
    700: hello: Hello world! Processing the event! 3 left
    800: hello: Hello world! Processing the event! 2 left
    900: hello: Hello world! Processing the event! 1 left
   1000: hello: Hello world! Processing the event! 0 left
   1000: hello: Done firing!
```

Figure 1.3 - Event-driven programming, copied from [24]

4.5 Adding parameters to SimObjects

Furthermore, the next topic on the list that I covered is parameters. Parameters allow for additional attributes to be added to the simulation. For instance, the parameters that I added are a time-to-wait and number-of-fires. The time-to-wait parameter allows for a delay while the number-of-fires enables a set amount of events to be processed. I had very few issues when working with parameters. I just had to fix a few typos and make sure I had all the necessary files.

```
10000000: hello.goodbye_object: Saying goodbye to hello
10000000: hello.goodbye_object: Scheduling another fillBuffer in 152592 ticks
10152592: hello.goodbye_object: Processing the event!
10152592: hello.goodbye_object: Scheduling another fillBuffer in 152592 ticks
10305184: hello.goodbye_object: Processing the event!
10305184: hello.goodbye_object: Scheduling another fillBuffer in 152592 ticks
10457776: hello.goodbye_object: Processing the event!
10457776: hello.goodbye_object: Scheduling another fillBuffer in 152592 ticks
10610368: hello.goodbye_object: Processing the event!
10610368: hello.goodbye_object: Scheduling another fillBuffer in 152592 ticks
10762960: hello.goodbye_object: Processing the event!
10762960: hello.goodbye_object: Scheduling another fillBuffer in 152592 ticks
10915552: hello.goodbye_object: Processing the event!
10915552: hello.goodbye_object: Goodbye done copying!
```

Figure 1.4 - Adding parameters number-of-fires & time-to-wait, copied from [25]

4.6 Creating a memory system

Creating a memory system is done by developing the interaction of how information is sent. In computers, the information in the form of packets that are sent across ports. The two types of ports that are used in gem5 are called the master and slave ports. The master port sends requests to the slave port. Therefore, the slave port receives the request from the master port and then sends a response back to the master port. The use of these ports allows for information to be traveled throughout the computer. To demonstrate the travel of information through the computer we can visualize how a CPU works. A CPU is the central processing unit which processes all the data for a computer to work. For instance, a 1 GHz CPU can carry out one billion instructions per second [26]. The instructions are then delivered to and from memory by the master and slave ports. The packets of information gathered back from memory through the master and slave ports, is then sent to the CPU where the information is processed.
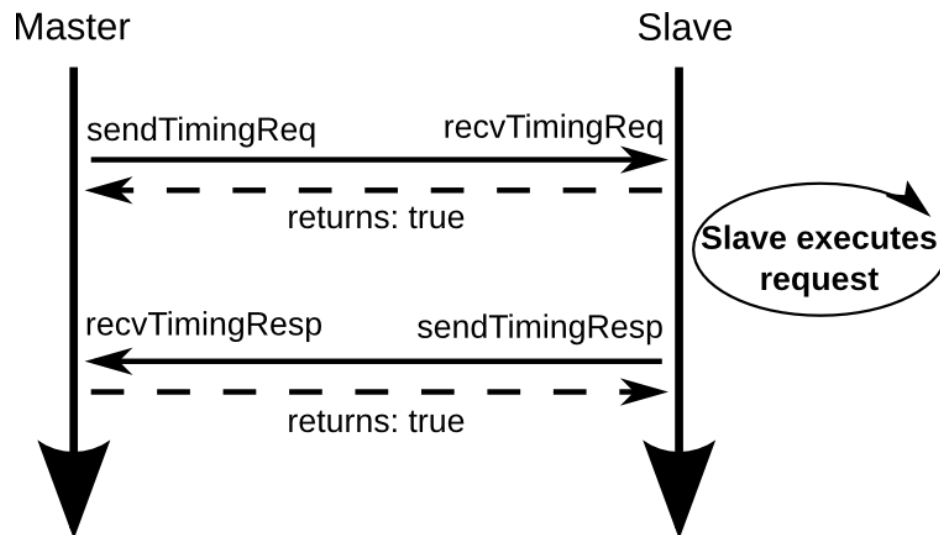

Figure 1.5 - Master and slave ports, copied from [27]

When creating these ports in gem5, I had to create the requests and responses that the ports will get. This is shown in Figure 1.5 above. In the grand scheme, a master port

sends requests and receives responses while a slave port receives the requests (and executes them) and then sends the response back to the master port.

4.7 Creating a cache object

In this section, I created a cache object. This is similar to the cache object made from 3.2. A cache will enable the simulation to access data faster. This is due to quicker access to information than accessing from memory.

Figure 1.6 - Difference when adding a cache object, copied from [28]
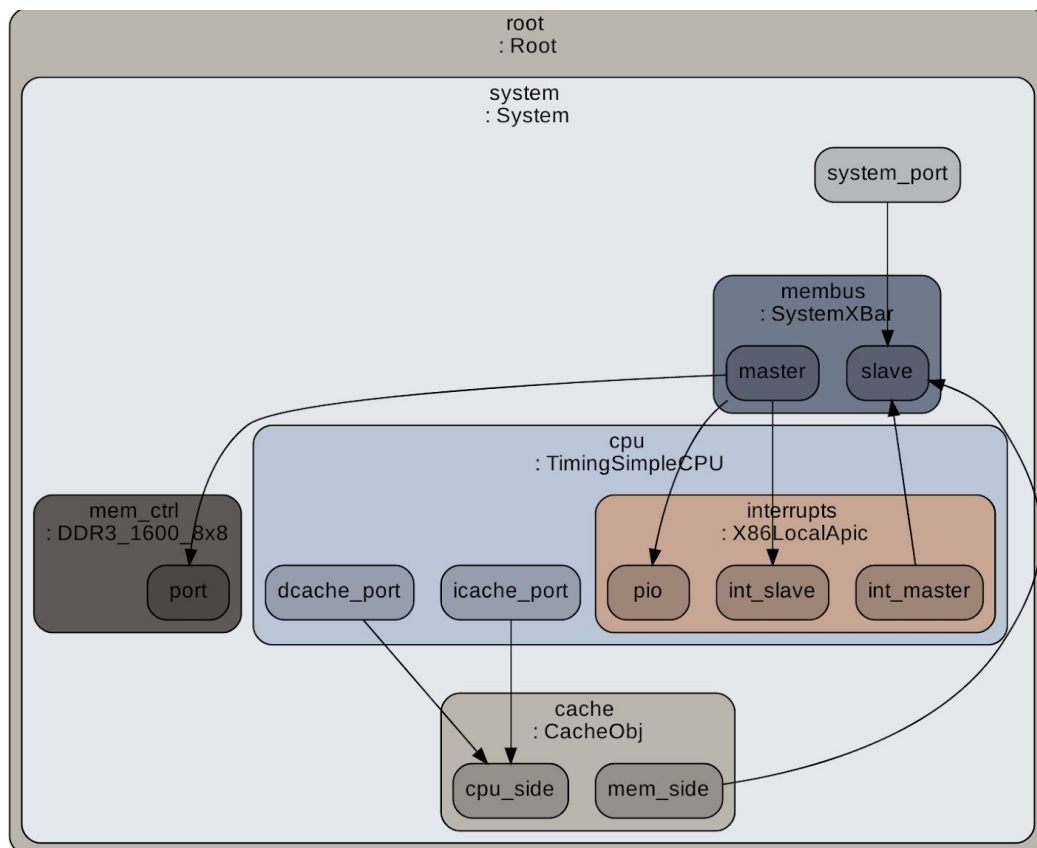

Figure 1.7 - Diagram of cache object, copied from [29]

Chapter 5: Full System Simulation

5.1 Full system mode

Full system simulation is sectioned in Part V which is the final part offered by gem5. My internship period was running out therefore, I moved over to Part V. Everything up to this point has been done using the simulated emulation mode. Full system simulation is a different mode that gem5 provides which was started in 3.3. In this mode, a full system is replicated and relies on the hardware that the simulation runs on. The full system simulation allows for a more accurate way of simulating the run time and processes. When using this mode, the user needs a disk image to run, in my case, I used the x86 Linux image. The disk image is used to run as the operating system for the simulation.

However, setting up this mode for simulations takes a considerable amount of time. For instance, it could be hours, days, or even weeks to start up the simulation. Therefore, I was unable to test this mode further. I waited for hours with little to no results. A diagram of the simulated system can be found on my website [30].

Conclusion

gem5 is a modular full computer system simulation platform for architecture research which also includes microarchitecture of a processor. The gem5 simulation platform is supported by contributing individuals and several companies: the National Science Foundation, AMD, ARM, Hewlett-Packard, IBM, Intel, MIPS, Sun, Lucent, and the Alfred P. Sloan Foundation [5].

The capabilities of gem5 have allowed for further research by those who are interested in computer architecture. The ability to simulate a full system by being able to establish connections, create components, and to control the flow of information all modularly makes gem5 very beneficial.

Overall, my internship experience at the University of Illinois at Chicago has given me a great deal of insight on gem5. I started from scratch by learning about two new programming languages. From there, I was able to start creating my very first configuration script for simulation. As the weeks passed, I got more in-depth knowledge of the workings of gem5. I was able to create different modular objects such as a memory system, cache, and other interconnects. Moreover, I gathered more of an understanding of how a computer works down to the smallest of components.

I would like to give another thank you to those who have helped me acquire this opportunity. Also, I would like to thank Mr. Muhammad Rafique and Ms. Zhichun Zhu for helping me throughout my internship.

References

[1]   "Simulation Process." GitHub,

        github.com/dependablecomputinglab/csi3102-gem5-profiling.

[2]   Puiu, Tibu. "Your Smartphone Is Millions of Times More Powerful That All of

        NASA's Combined Computing in 1969." ZME Science, 15 Feb. 2019,

        www.zmescience.com/research/technology/smartphone-power-compared-to-apoll

        o-432/.

[3]   Moore's Law, www.mooreslaw.org/.

[4]   "Introduction." gem5, gem5.org/Introduction.

[5]   "Main Page." gem5, gem5.org/Main_Page.

[6]   "The gem5 Simulator." ACM SIGARCH Computer Architecture News, ACM,

        dl.acm.org/citation.cfm?doid=2024716.2024718.

[7]   Roberts, Bucky. YouTube, www.youtube.com/user/thenewboston.

[8]   Schafer, Corey. "Corey Schafer." YouTube, www.youtube.com/user/schafer5.

[9]   "What Are the Differences between the Private, Public and Protected Access

        Specifiers in c?" Quora,

        www.quora.com/What-are-the-differences-between-the-private-public-and-protec

        ted-access-specifiers-in-c.

[10]   "General Memory System." gem5,

        gem5.org/General_Memory_System#Atomic.2FTiming.2FFunctional_accesses.

[11]   Palomo, Jim. Simple Configuration Script.

[12]   Palomo, Jim Diagram of Simple Configuration Script.

[13]   "gem5 Tutorial." gem5, learning.gem5.org/book/index.html.

[14]   Adding Cache to the Configuration Script. gem5,

        learning.gem5.org/book/part1/cache_config.html.

[15]   Palomo, Jim. Running Simple Configuration File with Caches.

[16]   Palomo, Jim. Diagram of the Addition of L1 & L2 Caches

[17]   "A Software Construction Tool." SCons, Scons Foundation, scons.org/.

[18]   Palomo, Jim. Hello Object File.

[19]   Palomo, Jim. Hello Header File.

[20]   Palomo, Jim. Hello Implementation File.

[21]   Palomo, Jim. Hello Sconscript File.

[22]   Palomo, Jim. Hello Run Script.

[23]   Palomo, Jim. Debug Flags for Simple Configuration Script.

[24]   Palomo, Jim. Event-Driven Programming.

[25]   Palomo, Jim. Adding Parameters: Number-of-Fires & Time-to-Wait.

[26]   "The CPU and the Fetch-Execute Cycle - Revision 2 - KS3 Computer Science -
        BBC Bitesize." BBC News, BBC,
        www.bbc.co.uk/bitesize/guides/zws8d2p/revision/2.

[27]   Learning gem5. gem5, learning.gem5.org/book/part2/memoryobject.html.

[28]   Palomo, Jim. Difference When Adding a Cache Object.

[29]   Palomo, Jim. Diagram of Cache Object.

[30]   Personal Portfolio. Jimppalomo.github.io.

[31]   gem5-dev repository. https://github.com/JimPalomo/gem5-dev.