What is Computer Architecture?

Charles Bailey

04/14/2015

Computer Architecture is a required course Computer Science students must take to graduate from Westminster College. As the class has progressed, I have learned more about what Computer Architecture is. Computer Architecture can best be described as

“a specification detailing how a set of software and hardware technology standards interact to form a computer system or platform. In short, computer architecture refers to how a computer system is designed and what technologies it is compatible with.

As with other contexts and meanings of the word architecture, computer architecture is likened to the art of determining the needs of the user/system/technology, and creating a logical design and standards based on those requirements.” (Computer Architecture, techopedia).

Many of the lab exercises and homework that have been assigned have dealt with how computer systems interact with each other or with certain technologies like the Stelaris Launchpad. To gain a better understanding of what Computer Architecture is, we must look at Von Neumann architecture, which is what modern computers are modeled after.

Mathematician and physicist John Von Neumann described the Von Neumann architecture. The basic idea behind the architecture is having a processor with an arithmetic logic unit, registers, a control unit with an instructor register and program counter, memory for both data and instructions, external mass storage, and input and output mechanisms. It is often a stored-program computer that has an instruction fetch and a data operation that cannot run at the same time due to sharing a common bus. A bus is a communication system in a computer that will transfer data between computers. The Von Neumann architecture also implements the concept of the stored – program concept. The stored program concept involves programs written in machine language in a series of 32 bit numbers that represent instructions that can be stored in memory. The advantage of the stored program concept is that the hardware of the computer does not have to reconfigure or rewrite the hardware to run a new program. Instead, the program is written to memory, which allows a processor to execute instructions from a program in memory. Even a complex program can be simplified to memory reads and instruction executions. When a stored program is fetched from memory by the processor, the instructions are decoded and executed by the digital hardware. Without the Von Neumann architecture and the stored program, modern computers would not exist and they would be very limited in practical use.

One of the very first concepts of Computer Architecture that was covered was number systems. While humans are accustomed to working with decimal or base 10, many Computer Systems work with binary or hexadecimal numbers. The binary system allows a bit to have two states, (like true or false, on or off). One of the disadvantages of using binary numbers is that long numbers have to be used to represent larger values such as 4,294,967,295, which is 1111.1111.1111.1111.1111.1111.1111.1111 in binary, that can be tedious and be prone to error. Hexadecimal or base 16 is also used since it can represent larger numbers using more symbols. Chapter 1 also introduced the concept of digital abstraction. Digital systems will represent information discrete-valued variables that have a finite number of distinct values. Computers use a binary representation to represent high and low voltages, with 1 indicating high voltage, and 0 indicating a low voltage. This idea of representing values with 1’s and 0’s is Boolean logic and is useful since the computer does not have to worry about if the values represent a specific voltage, rotating gear, or hydraulic fluid level.

One of the first labs that we did involved learning about the boot loader. The boot loader of the TI Stelaris Launchpad downloads code to the Flash Memory of a device without using a debug interface. In a regular computer, the boot loader loads the operating system of a computer after the computer is turned on. This lab was important to understanding Computer Architecture because a boot loader with faulty design limitations can hamper a user’s ability to use an operating system or to use certain programs available on an operating system. The lab 1 report required us to describe what a register is. The primary purpose of registers is to store the addresses of instructions or data for later use. Interrupts were also introduced to us in this lab. The interrupts are signals to the processor from software or hardware that indicates an event needs immediate attention. In the lab, when an interrupt occurred, the processor had to use code from the Launchpad to handle the interrupt or terminate the process. The processor will record the cause of an interrupt and the value of the PC or program counter. The program counter is important because it serves as a register that that contains the location of the instruction being performed at the time. If an interrupt occurs, then it prevents the current process from running and for the user to wait for the exception handler to resolve the interrupt. The stack pointer is also an important concept in computer architecture because it stores the address of the instruction or program that was last used. Stacks work in a top down order, with the most recent address at the top and older processes at the bottom. The processes that are used first get pushed to the bottom and other programs can use the most recent address of the process.

Memory has been a very important topic discussed in Computer Architecture. Without memory, the computer would not be able to store programs or even know how to resolve interrupts, exceptions, and other issues. Registers are one of the components of memory, as they store the addresses of variables, instructions, or processes and provide a faster method of accessing those memory addresses instead of having to look in the entire main memory. Memory also stores vector tables that were introduced in lab 3. The vector tables have the addresses of interrupts stored, so that a processor can easily access the information and handle the interrupt. Lab 3 taught people about how a processor will store an interrupt number in a register, and then the processor will use the memory address to find the instructions needed to handle or stop the process after the interrupt has occurred. When the exception number was 15 shown in the Interrupt Service Handler, the interrupt meant that Systic was enabled, which had an automatic number of seconds to use instructions to resolve the interrupt.

Chapter Two introduced Combinational Logic Design, which very much involves Boolean equations and Boolean algebra. Since computers have binary logic that deals with high and low voltage, 1’s and 0’s, they often have Boolean equations. As discussed earlier, the advantage of a binary number system in computers is that the system does not have to worry about using 10 distinct values to represent a voltage and there are only two symbols to worry about. Boolean functions are also easier to use to distinguish the absence or presence of a current in the circuits of a computer. We can still represent the same numbers in binary that we could in decimal, with the only difference being with how humans see a number written out. Lab 4 introduced the breadboard and using wires to represent various functions of Boolean algebra. When the breadboard was configured properly, we accurately represented the Boolean table for a nand gate, and for a nor gate.

The last chapter that we covered and discussed was Chapter 6, which covered most of what Architecture is. Computers use assembly language, the human readable representation of what the computer’s native language is. Most programmers will write code in a high level language like C, C++, Java, Python, or Ruby, which is easy enough for humans to read and give instructions to a computer. High level languages use if/else statements, for loops, while loops, array indexing, and function calls. When a compiler compiles the code of a program, it translates the high level code into assembly language for the computer to get the instructions and execute the program. The computer will use registers for operands like a, b, or c in a program, but will need to represent them with a physical location to retrieve the binary data. Chapter 6 also talks about the stored program, which was talked about in an earlier paragraph. Assembly language allows a computer to store the data of a program within registers and allow the computer to understand the instructions of a high level program in a binary format. Without assembly language to give the computer an understanding of how to implement the instructions of a program, computers would have to operate with a different system than binary and require humans to do a lot of the work of counting and translating binary numbers.

Computer Architecture is very much about knowing the functionality, organization, and implementation of computer systems in hardware and how they meet the needs of a computer. For computer scientists, it gives them some background and exposure to Computer Engineering, as computer engineers will more often than not work with the hardware of a computer, while computer scientists focus more on software and its theoretical aspects. Computer Architecture gives us a chance to see how software interacts with the hardware, as many computer science students study aspects more applicable to how software operates and the theoretical aspects of how to use it. Having an understanding of how a computer system is designed and what components are used can enable a computer scientist to write more efficient code that better uses the components of computer hardware. Many of the systems covered like the processor and registers are important components for any software as they implement the instructions and save the data for a user, and allow multiple programs to be used in a computer. While we briefly talked about power consumption and performance, understanding how these issues affect a user is useful in developing better architecture for a computer to better serve the users. While a programmer may not need to understand how the hardware or the software components that talk to the hardware work, knowing Computer Architecture can help them know about errors like memory overflow and also help a programmer with debugging an issue.

Bibliography

"What Is Computer Architecture? - Definition from Techopedia."*Techopedias*. N.p., n.d. Web. 27 Apr. 2015.