Dillon Irish

4/29/15

Computer Architecture Term Paper

Scott Overholser

The topic of computer architecture is a complex one. How a computer works, what components are essential, and how these components interact are all questions that fall under this topic. In order to answer these, we break the layers of a computer into separate abstractions. This is similar to the way we view and discuss the universe. The universe, overall, is very complex, but by breaking it into separate layers (planets, solar systems, galaxies) it becomes much easier to digest. When talking about computer architecture we break it into 9 layers of abstraction (from lowest to highest: physics, devices, analog circuits, digital circuits, logic, micro-architecture, architecture, operating system, and application software), we focused on 6 of these layers heavily in class.

Moving from the lowest (most basic) to the highest (more complex) layers, we began in with devices. This layer sits right above, and allows us to ignore, physics. Devices are the actual electrical instruments used the allow the system functionality. An example of these instruments are transistors and vacuum tubes. Devices are connected by terminals which allow us to measure the relationship between devices in terms of voltage and current.

Right above devices is analog circuits. In the analog circuit layer we begin to combine devices into components that allow the increase and decrease of voltage. This is the first layer where input and output voltages begin to matter, thus is the beginning of logic in a computer. Closely tied to the analog circuit, is digital circuits. In this layer we use logic gates (AND, OR, NOT, XAND, etc) to input and output discrete ranges. With the introduction of discrete ranges we began to measure discrete states (HIGH/LOW, On/Off, 1/0). The gates implemented in the layer (AND, OR, NOT, XOR, etc) allow use to began doing simple binary

calculations (addition, subtraction). It's not till the next layer that we see much more complex calculations.

In the logic layer we use complex structures made up of many gates to create things like adders and memory. With the introduction of subtraction and addition in the previous layer, we move onto boolean algebra. With the use of axioms and theorems we are able to make complex calculations, simpler and faster.

Above logic, we have the micro-architecture layer. Combinational and sequential logic resides here. The main purpose of of this layer is to connection between logic and architecture level much smoother. "[Micro-architecture] combines logic elements to execute the instructions defined by the architecture" (Harris & Harris, 5).

Lastly we have Architecture, the highest layer of abstraction we covered in class. "[Architecture] describes a computer from the programmer's perspective" (Harris & Harris, 5). Assembly language resides here, and is the lowest level programmers can interact with without resorting to coding in binary.

Each of these 6 layers individual is their own separate topic, but taken together they make up computer architecture. But just knowing these layers doesn't describe how a computer operates. They are a few key components that must be covered. Similar to the layers of abstraction, I will be covering them from lowest (most basic) to highest (more complex).

Lets begin with number systems. The purpose of a computer is just that, to compute. With the use of number systems this is possible. Humans, having 10 digits on their hands, typically think in base 10 or decimal number systems, but computers systems are not able to do this. Devices that make up one of the lowest levels of abstraction can only exist in one of two states, not ten. Thus computers use a base 2 or binary number system. Each bit of a binary number can represent a 0 or 1. Using these numbers, a computer is able to add and subtract binary numbers as well as hold negatives. Higher levels of abstraction may find

these large lines of 1’s and 0’s difficult to deal with, and are prone to making errors when using them. In this case we use base 16 or hexadecimal numbers. Not only are these numbers shorter in length, but can be handed much easier by both the computer as well as the user. Programmers are easily able to break hexadecimal numbers down into different sections (byte, nibble, word, etc.). Through the use of these number systems a computer is able to do its basic computing.

Just like in decimal calculations, binary numbers are able to use algebra. This is called boolean algebra. Similar to decimal algebra, boolean algebra is based on set axioms and theorems and allows large complicated calculations to be shorted down and simplified. Sense calculations in a computer are done through gates (AND, OR, NOT, XOR, etc) by using boolean algebra we are able to greatly simplify large circuits. Smaller and simpler circuits means a quicker system.

The digital building blocks of a computer system implement boolean algebra. These building blocks are based on combinational and sequential logic (formed in logic level) and implemented by the use of multiple combinational and sequential circuits (digital circuits). Combinational logic is used on circuits that only rely on current input, thus they require no memory. The most commonly used combinational circuit in a computer is the multiplexer. A multiplexer chooses its output from multiple inputs based on a select signal. Another example of a combinational circuit is the decoder. Decoders take N input and have 2^N outputs. Depending on the input combination, the output is chosen. Sequential Logic on the other hand is dependent on both current and prior input values from previous circuits. Sense is needs to remember previous inputs, these kinds of circuits require memory. While there two main different types of sequential circuits (latches and flip-flops) they both serve the same general purpose. They have no input, but two different outputs. This means they can exist in two different states, this is known as a bistable element. Essential these circuits are memory, and allow a state to be saved by the computer system.

By putting together multiple circuits we are able to construct components that make up a computer. Some examples of these would be the computer processor, memory, or the gpu. In order for these systems to communicate with one another and share data, the bus is implemented. The bus acts as a superhighway of data that connects all the separate components of the computer. In doing so, it is able to connect layers of abstraction into one computer architecture.

Moving into the architecture layer we have instructions. Instructions are used to tell the computer what to do. Even the highest level of abstraction (software applications) are boiled down to these simple instructions like add, subtract, and jump. These instructions indicate what needs to be done and who to perform that action on. The who, in these instructions, can come from memory, a register, the stack, or from within the instruction itself. Registers are sets of data holding places that are part of a process. They can hold many kinds of data such as instructions and storage addresses. They are usually big enough to hold a word and are the fastest way to access data. The stack is memory that is allocated for each function. Implemented as a last-in-first-out queue, it is used to hold local variables for functions. Through instructions we are able to connect lower layers of abstraction to the higher ones to create one cohesive unit that is working to accomplish a single goal.

The topic of computer architecture is a broad and complex one. By viewing the layers of abstraction as well as the important components and concepts that dwell within each layer, we can begin to comprehend how the architecture of a computer is built and operated.