David Hatcher

CDA4205

Dr. Karam

8/31/2020

Homework 1

1. Moore's Law is the idea that the density of transistors on chips will double nearly every year originally, it was eventually changed to 18 months, in turn doubling the performance of chips every 18 months. Gordan Moore was the originator. For future processors it means to keep up with Moore's Law the size of the transistors will need to be cut in half every 18 months. Computer architects use it by planning the size of their next generation of chips.
2. A transistor is basically a switch that can be turned on or off by an electrical current that, when "on" allows electricity to pass through it. The development of transistors affected computer by making them be able to run faster and have a smaller profile by being the first non-mechanical switch. It replaced the vacuum tube.
3. Abstraction with respect to computer architecture is the idea that the different levels of computing do not necessarily known by the other people using them, such as a person writing a program in python does not need to known how the gates on their processor work. This is used to break up the levels of concern when developing either software or parts for computers, which is helpful because no one person needs to have a fully complete understanding of how a computer works from the physics of the electrons to the higher-level code they could write. An example of how I've used abstraction in the past is while writing a project for one of my programming classes, instead of creating one method that would accomplish the entirety of the project I broke it up into many different methods that would accomplish a small part of it so that none of the different methods had to "know" what the other were doing but just ran off of the assumption that they accomplished what they were supposed to.
4. Application design/requirements tends to be related to a specific problem requirement while the hardware design tends to be focus on versatility to allow for many platforms to be run off it. The demands of software can influence hardware design by providing suggestions on how to improve architecture, for example if you were building a application for a computer that needed to pull data from a TCP connection constantly it might be a good idea to move the component responsible for that TCP connection closing to the processor to cut down on the transfer time between those components. Hardware design can affect applications in the same way such as if the hardware is built do be able to do one thing efficiently then the software developers have to take those things into account while building the software. As to which should come first, you obviously can't run software on hardware that doesn't exists by that doesn't mean you can't start building software for hardware that's currently in development, so realistically they can both happen at the same time.
5. Is it necessary for a computer programming to understand computer architecture? My answer is no, because of the whole idea of abstraction, just because someone doesn't understand how the physics work for electrons moving through a transistor doesn't mean they can't write code to make an application work on those same transistors. However, having an understanding of the different levels of abstraction for computer architecture may help a programmer create

more efficient code as they might have a better understanding of what actions on the platform they're using will run more efficiently.

6. Some of the challenges faced in modern semiconductor manufacturing are producing products are the current sizes that are able to function properly, as once you get to a certain size you run into issues where the electricity will start to bleed out of the circuits and end up in places you don't want it. Another challenge is the shear cost of R&D for new methods of creating the chips when the sizes get to small for the current manufacturing methods to handle. These relate to Moore's law because the best way to improve performance of chips is to increase the density of the transistors on the chips which means that the transistors must become smaller and smaller over time. What I found most surprising was the number of steps and processes that have to be completed to create these chips and how they're still relatively cheap when compared to the costs of the materials and R&D.

Intro to RSIC-V