|  |  |  |  |
| --- | --- | --- | --- |
| Instructor |  | Due Date |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Part | **1** | **2** | **3** | **4** | Total |
| *Maximum Points* | **25** points | **25** points | **25** points | **25** points | **100**G101010 pointsG |
| ***Your Score*** |  |  |  |  |  |

**Textbook Reading Assignment**

Thoroughly read Chapter(s) 1 in your Computer Architecture and Organization textbook.

**Part 1 Glossary Terms - An Introduction to Computer Architecture and Organization**

Define, in detail, each of these glossary terms from the realm of computer architecture and organization, in general. If applicable, use examples to support your definitions. Consult your notes or course textbook(s) as references or the Internet by visiting Web sites such as:

[**http://www.bing.com**](http://www.bing.com) or [**http://www.webopedia.com**](http://www.webopedia.com/)

**(a) Cloud Computing**

|  |
| --- |
| **A form of distributed computing, cloud computing is done by connecting to a network of virtual machines & web servers and enables people and businesses to deliver software as a service (SaaS) in a distributed way (i.e., to a client’s web browser anywhere in the world). Amazon web services, Microsoft Azure, and Google Cloud Platform are the three most widely distributed and used cloud computing platforms.** |

**(b) ENIAC**

|  |
| --- |
| **ENIAC (Electronic Numerical Integrator and Computer) was the first all-electronic, general-purpose computer. It was funded by the army to be used for calculating the trajectory of ballistics, which would take days. The ENIAC could generate ballistics tables in minutes.** |

**(c) High - Level Programming Language**

|  |
| --- |
| **A high-level programming language, described as the level between assembly language and the user in the computer hierarchy (37 Null et al.), is a human-readable and -writable programming language that compiles down to an assembly language. Examples include C++, Java, Python, JavaScript, etc.** |

**(d) nano**

|  |
| --- |
| **A nano in the context of computer organization and architecture is a prefix used to denote a billionth (10^-9) of some unit of measurement, typically in the context of how fast an operation can take place. For example, a computer that can execute a billion floating point operations, or “flops”, per second, could be said to complete one flop per nanosecond.**  **Also, “nano” is used to describe the scale of the elements of structures on a modern, complex integrated chip.** |

**(e) Rock’s Law**

|  |
| --- |
| **Rock’s law states that the cost of a semiconductor chip manufacturing plant doubles every four years (making chips get more expensive), and is an economic corollary to Moore’s Law, which states that the number of transistors in a circuit doubles every two years (chips get faster). Designing and developing a complex integrated circuit alone costs tens of millions of dollars, the infrastructure needed to produce them is even more expensive.  The idea is that eventually, we won’t be able to make faster chips because the costs would be prohibitive, but we’re not there yet.** |

**Part 2 Exercises - An Introduction to Computer Architecture and Organization**

Provide a brief but complete answer for each of these exercises.

**(1)** In what ways are hardware and software different? In what ways are they the same?

|  |
| --- |
| **“Software” is any set of instructions that tells hardware (the processors, control units, circuits, logic gates, etc.) what to do. They are different in the same way that a driver is different than the car they are driving. But they are fundamentally the same in that any task done by software can also be done using hardware, and any task done by hardware can be simulated by software (“principle of equivalence of hardware and software”). In other words, I can create a machine that runs on purely mechanical components to make an LED blink every 2 seconds, and I can simulate the logic of this program by writing down its instructions on a piece of paper.** |

**(2)** By what order of magnitude is something that runs in nanoseconds faster than something that runs in milliseconds?

|  |
| --- |
| **Nanoseconds (billionths) are two orders of magnitude lower than milliseconds (thousandths). You could say that a nanosecond is a million times faster than a millisecond.** |

**Part 3 Exercises - An Introduction to Computer Architecture and Organization**

**(1)** Suppose you are ready to purchase a new computer for personal use. First, take a look at advertisements from various magazines and newspapers and list terms you do not quite understand. Look these terms up and give a brief written explanation. Decide what factors are important in your decision as to which computer to buy and list them. After you select the system you would like to buy, identify which terms refer to hardware and which refer to software.

|  |
| --- |
| **Looking at the new Macbook with the M1 chip. I have no idea how the M1 chip is different than a regular Intel chip. I’m going to focus on the chip itself and look up terms I’m unfamiliar with on Wikipedia to keep this in scope. This will be more of a silicon chip research assignment, since I have no idea how chips are made.   From: https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/  The M1 chip is a low power silicon chip with 16 billion transistors and delivers 3.5x faster CPU speeds than the previous generation mac. It has a “unified memory architecture for dramatically improved performance and efficiency” and is “built using cutting-edge 5-nanometer process technology”. I think that’s enough to unpack.**   * **Low power silicon chip: A chip is a set of electronic circuits on a piece of semiconducting material, typically silicon. Semiconducting materials have useful properties such as passing current more easily in one direction than another, and variable resistance, which make them suitable replacements for vacuum tubes for passing electronic pulses through a circuit. M1 is a “low power” silicon chip because its small design (nanometers) allows for low power dissipation.** * **16 billion transistors: a transistor is a semiconductor device used to amplify or switch electrical signals and power. 16 billion transistors on a chip would require having elements on the nano-scale.** * **Unified memory architecture: “the methods used to implement electronic computer data storage” (wiki). Apple’s M1 chip has a “unified” architecture in which the CPU and graphics processor share the same pool of memory, so there’s no back-and-forth I/O of system memory and graphics memory, it’s all in the same place. The memory bandwidth in an M1 Max Macbook is almost as fast as a Playstation 5.** * **“5 nanometer process technology”: this is sort of mind-blowing. The MOSFET (metal–oxide–semiconductor field-effect transistor) scale of semiconductor fabrication gets smaller every year. In 2001, elements of structures on a semiconductor were being made on a 130nm scale, in 1981, it was an order of magnitude smaller at 1.5 micrometers. More transistors = more speed.** |

**(2)** Suppose a transistor on an integrated circuit chip were 2 microns in size. According to Moore's Law, how large would that transistor be in 2 years?   
 How is Moore's law relevant to programmers?

|  |
| --- |
| **In 2 years, that 2µm sized transistor would be 1µm (a size we reached in 1984).  Moore’s law is relevant to programmers because it forces us to adapt. As transistors get smaller, new technologies become possible, new applications are possible, more JavaScript frameworks and NPM packages are created, and we’ve got to learn how to use them in order to stay relevant and keep our jobs.   However, we are reaching the end of Moore’s law. Atoms are anywhere between 0.1 to 0.5 nanometers in diameter. Apple’s M1 chip is made on a 5nm MOSFET scale. In the next couple of years, we should have chips on the 2.5 nanometer scale, and by the end of the decade we’ll be designing transistor elements at a near-atomic scale.   This means that, barring some breakthrough in quantum computing, hardware will soon reach its zenith, which means that software and the way we write it will be frozen.   However, with the development of cloud computing infrastructure, there’s still an incredible amount of potential for software growth and innovation. The hardware may not get better but there’ll be more of it, and it’ll get more accessible.   Imagine an array of 1nm MOSFET chips that you could dispatch an expensive multi-process (a highly localized weather model simulation for example) and return to a client (a browser in a smartphone for instance). What would take many hours if ran on software on a laptop would take only seconds if ran using multi-processing in the cloud.** |

**Part 4 Exercises - An Introduction to Computer Architecture and Organization**

**(1)** Under the von Neumann architecture, a program and its data are both stored in memory. It is therefore possible for a program, thinking a memory location holds a piece of data when it actually holds a program instruction, to accidentally ( or on purpose ) modify itself.

What implications does this present to you as a programmer?

|  |
| --- |
| **To me, this implies that when writing a program I need to be careful when assigning pointers / references in my code to avoid modifying certain functions or methods I had created.   For example, if I wrote a class in Python that had a “foo” method, I would need to be careful to avoid using “self.foo” when working within other class methods.   This would be at a higher level, at a lower level, allocating memory addresses manually would be possible but sounds like a bad idea. Though at an even lower level, e.g. working with a chip’s machine instruction set on a breadboard, I imagine it would be necessary.** |

**(2)** What are the challenges facing organizations that wish to move to a Cloud platform? What are the risks and benefits?

|  |
| --- |
| **I’m assuming this is an organization whose primary product is software that is either downloaded or physically distributed, and is meant to run natively on a user’s computer.   The main challenge for moving to the cloud would involve re-writing that software to run on the cloud. There would need to be additional software (a web server) written to manage user I/O, and frontend software written to run it in the client (e.g. a browser). Other challenges include making sure that the software-on-the-cloud is accessible in many regions, and the resources used to access and run it are not exhausted (via load balancing and distributed databases). There’s also the overhead cost of using resources on a cloud network such as AWS.   The risks are as numerous as the benefits. There could be outages, data breaches, IP theft, etc. However, in exchange for the added risk, the additional resources, scalability, and accessibility that cloud computing provides means organizations can quickly roll out powerful software on a global scale.** |