EI338 Homework 1

Computer system engineering & Deadline: 2019-9-20 Thursday 24:00

September 17, 2020

1. Read the introduction of the OS Textbooks (Operating System Concepts (any edition is ok)) and write down what you learned from the introduction and what you want to learn in this course.

   What I have learned from OS Introduction.

   - The overall computer system can be roughly divided into four parts, the hardware, the operating system, the application programs and a user. The fundamental goal of computer systems is to execute programs and to make solving user problems easier. The definition of operating system remains blurred, but we can here divide it into the always-running kernal, middleware frameworks and system programs.
   - In a typical PC computer system, various device controllers are connected through a common bus, and they are managed by the device drivers in the operating systems. CPU and device controllers execute in parallel, the driver knows the status of the controller through *interrupts*.
   - The design of a complete storage system must balance all the factors related to storage topics such as the trade-off between speed and size, the characteritic (e.g. volatile or not) of every hierachy, and the communication between layers.
   - A large portion of operating system code is dedicated to managing I/O, to avoid the high overhead when used for bulk data movement, *direct memory access* is used.
   - A comuter system can be organized in various ways for specific purposes such as single-processor systems, multiprocessor systems, and clustered systems.
   - Operating systems start running by an initial program called *bootstrap program*, then the kernel gets loaded and excuting. After that, system daemons and applications are run. Events are almost always signaled by the occurrence of an interrupt, by hardware or through exception and system call.
   - The operating system switches to and executes another process to implement multiprogramming or multitasking, where such techniques as CPU scheduling, dual-mode, timer are involved
   - An operating system is a resource manager, it manages processes, memory, file-system, mass-storage, cache and I/O systems.
   - *Protection* is the mechanism for controlling the access of processes or users to the resources defined by the computer system. *Security* is implemented to defend a system from external and internal attacks.
   - *Virtualization* is a technology that allows us to abstract the hardware of a single computer into several different execution environments.
   - Distributed systems are gaining increasing importance. There are several categories of networks according to its size. A network operating system is an operating system that provides features such as file sharing across the network, along with a communication scheme that allows different processes on different computers to exchange messages.
   - Several data structures useful and common in the kernal, such as stacks, queues, trees, hash maps and bitmaps.
   - Operating systems can be seen in various computing environments such as PC, client-server, peer-to-peer, cloud computing, and embedded systems. There are also a number of free and open-source operating systems ready to use and study. Note there is a difference between the notion of free and open-source.

   What I want to learn from OS part of the course

   - How the **basic** parts of an operating system are organized and **implemented**, and why they are designed in this way.
   - Important concepts in the operating system such as process, thread, and file systems.
   - Most importantly, less taxonomy and terminologies, more **coding and practice**

2. Read the introduction of the CA Textbooks (Computer Architecture: A Quantitative Approach (any edition is ok)) and write down what you learned from the introduction and what you want to learn in this course.

   What I have learned from CA introduction

   - With technological improvements, a new set of architectures with simper instructions, RISC, raised the performance of computers. Two critical performance techniques are instruction-level parallelism and use of caches.
   - The cost-performance improvement leads to new classes of computers, as well as the dominance of microprocessor-based computers across the entire range of computer design. Parallelism at multiple levels, including instruciton-level, task-level, thread-level and request-level, is becoming the driving force of computer design across all classes of computers.
   - The definition of computer architecture should not only be limited to instruction set design but also applied to other challenges when implementing the design, such as microarchitecture and hardware.
   - Typical ISAs are 80x86, ARM and MIPS, which vary in many aspects such as memory addressing rules, addressing modes, operands, operations, control flow instructions and encodings.
   - The architecture is largely affected and driven by trends in technology. For example, bandwidth has outpaced latency across latency across all technologies, feature size is going down while transistor performance is getting more complex to evaluate, subject to various emerging factors such as wire delays.
   - Power and energy should also be considered carefully, with techniques such as sleeping idle devices, dynamic voltage-frequency scaling, designs for typical use, and overclocking.
   - An understanding of cost and its factors is essential for computer architects. Some major factors that influence the cost of a computer includes the cost of water, the reliance of the die, testing, etc.
   - Benchmarks are used to measure the performance. There are many benchmarks for specific use. A benchmark suite is recommended. To normalize execution times, we can use a reference computer to rate others by ratio. To compare between several benchmark programs, we should take geometric mean.
   - Some quantitative principles of computer design involves taking advantage of parallelism, using locality, focusing on common use.

   What I want to learn from CA part of the course

   - A deeper insight into the instruction set, since we have learnt the basic semantics of MIPS instruction during the Computer Compostion Course, e.g. why they are designed this way.
   - Formal methods in quanfifying all the measures, with concrete examples.
   - The relationship and boundary between the CA and OS design, since the course will cover both topics.