CS426 MidTerm Study Guide

Architecture

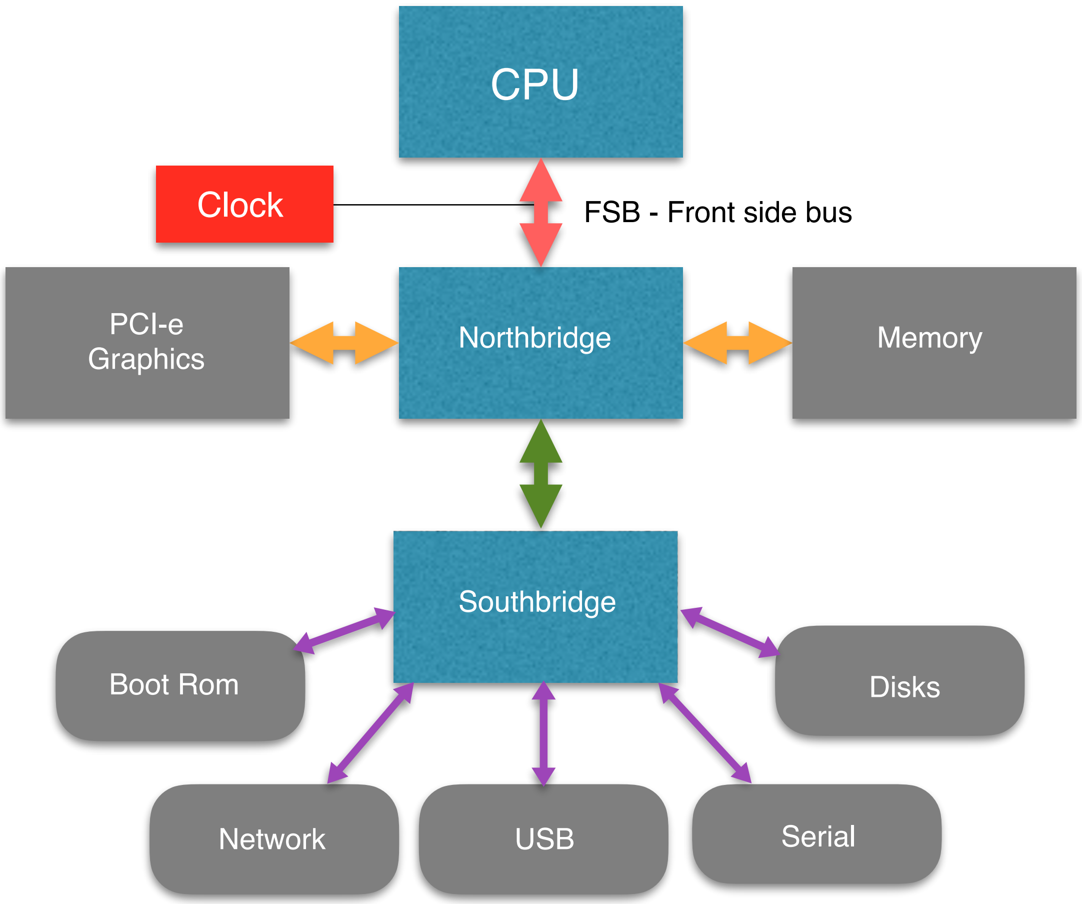
Operating systems exist to provide a uniform means of accessing hardware, and provide a platform upon which programs and applications may be built. The high-level design of these systems is known as an architecture.

Hardware Architecture

Hardware architecture is typically specified with a block diagram. This diagram show the hardware components, and the signal paths that connect them. On modern hardware, multiple cores and multiple processors are common. Because of speed requirements, data must be cached as close to the processor core that is using it as possible. Hardware architectures that support this are known as NUMA (Non-Uniform Memory Access) architectures.

Multiprocessor systems may be homogeneous, or heterogeneous. Most systems today, including mobile devices are both. They have homogeneous processor cores for applications, and homogeneous graphics processor cores for display.

A common architecture in use on Intel based systems, is the Northbridge/Southbridge architecture.



Things to note:

1. The clock is close to the processor.
2. The bus connected to the processor is known as the front side bus, FSB.
3. The Northbridge component is for very high speed peripherals, like memory and advanced graphics controllers.
4. The Southbridge component is for relatively lower speed peripherals. Note that keyboards, mice, and terminals are all serial devices usually.

On the exam be able to recreate this diagram and indicate what all the parts represent.

Software Architecture

An operating system has a **layered architecture**. It is the middle of a three layer architecture that has hardware as the lower layer, and applications as the upper layer. Communication across layers in a layered architecture is known as an **interface**. The interface to the operating system is accomplished via interrupts and/or software traps. The code running in the OS layer runs the hardware in a privileged mode that allows direct access to the hardware. User applications run in a mode that give limited access to computer resources. This allows an operating system to isolate processes from each other and provide a basis for implementing security systems.

User Processes

Application code runs as a process associated with a particular user.

Object Code File

The compiler produces an executable known as an object file. It contains the information needed to load a process into memory. There is a header that identifies the format of the file, and where to find map that points to each section in the file. You will need to know the sections of the object file as described on the educat page.

Processes

You will need to be able to draw and label the parts of a process that has been loaded into memory. You will also need to describe the basic components of a process control block. See Chapter 3.

You will need to know how to use the fork() system call. You will need to know how to detect errors. You will need to know that the parent process must either wait for the child process to die, or install a SIGCHLD signal handler to do the wait when it dies. You will need to know what will happen if you do not wait for a child process to die.

Threads

You will need to know the difference between user threads and kernel threads. You will need to be able to explain the one-to-one, many-to-one, and many-to-many threading models. You will need to know what parts of a process are shared by threads, and which parts are specific to a thread..

Coding

You will need to be able to declare a pointer to a function.
You will need to be able to code the forking of a process and check for errors appropriately.
You will need to be able to code a synchronization primitive like atomicOr or atomicAnd using an atomicCompareAndSwap call. These are called lock-free or wait-free synchronization algorithms and are used at the lowest levels of the operating system.