Ýmir Þórleifsson

HW 1

1.1.1

Abstraction and virtualization.

Justify or refute the statements below. Give supporting examples, if appropriate.

a) Abstraction can be used without virtualization.

Yes. Because abstraction is a software type that allows the user to easily interact with the computer on a higher level. Can be done without virtualization.

b) Virtualization can be used without abstraction.

Generally no. Virtualization is usually used to simulate items, like CPU and storage. This is mostly used to help with a smoother experience along with abstraction. But if one chooses it should be possible to use virtualization while only dealing only with low level commands, without abstraction.

c) Abstraction and virtualization can be used together.

Yes. I.e. when time-sharing (multitasking) we allow each program (abstraction) to have their own CPU (virtualization).

1.1.2

Multiprogramming and time-sharing.

a) What do multiprogramming and time-sharing have in common? What are the differences between the two concepts?

Multiprogramming and time-sharing are both keeping several programs active in memory and switch execution among the different programs to maximize the use of the CPU. When one program enters an I/O-bound phase and does not need much CPU computation, other programs can still access it.

The difference between these concepts is that time-sharing switches periodically among all active computations to guarantee acceptable response times to the user. By using virtualization, creating a virtual CPU for each computation. While multiprogramming only switches execution on an I/O-bound phase.

1.2.1

a) What do interrupts and traps have in common? What are the differences between the two concepts?

An interrupt diverts execution to the kernel. A trap is a type of interrupt, that is triggered from an error or supervisor call instruction. So they both divert to the kernel.

The difference is how they are triggered, interrupts are triggered from hardware signals, while traps are software signals.

1.2.2

a) Is multiprogramming possible without interrupts?

No. Because how would the CPU know when to switch between programs? Interrupts are used to tell the kernel to switch execution to another program.

b) Is time-sharing possible without interrupts?

No. Because the user expects each program to respond in an appropriate amount of time. This cannot be done if only one program would be able to run at a time, i.e. no jumping between execution of programs (with timeout interrupts).