1- introduction and os overview

operating system= low level software that manages computer software and hardware

↳ abstraction= reduces complexity between user application and computer

↳ arbitration= manages access to resources

↳ concurrency= gives every application the illusion of having its own cpu by switching between them

↳ virtual memory= gives every application the illusion of infinite memory

* in first computers there were no operating system, computer only ran one program at time, no need
* throughput= measure of how many units of info a system can process in a given amount of time

control flow= the order in which individual statements, func calls or instructions are executed

↳ jumps and branches
↳ call and return
} two mechanism for changing control flow with react to chance in program state

exceptional control flow= an exception is an interrupt change in control flow in response to some change in the system state. (ctrl-c, timer, division by zero, data arriving from a disk...)

↳ low level mechanisms=

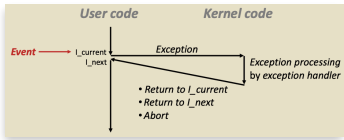
↳ exceptions= implemented using hardware and OS software, control is transfered to the OS kernel

↳ higher level mechanisms=

↳ process context switch= OS software + timer

↳ signals = OS software

↳ nonlocal jumps= C runtime library - setjmp(), longjmp()



exception tables - interrupt vector = an array of pointers to functions to handle specific exceptions

① asynchronous exceptional control overflow/interrupts

↳ caused by events external to the processor (can happen any time)

↳ handler returns to next instruction

↳ examples: timer interrupt, i/o interrupt from external device (ctrl-c, arrival of data)

② synchronous exceptional control overflow

↳ caused by events that occur as a result of executing an instruction

↳ traps=	↳ faults=	↳ aborts=
• intentional	• unintentional	• unintentional
• returns the next inst	• possibly recoverable	• unrecoverable
• ex: system calls, break point traps, special instructions	• reexecute current or abort • ex: page fault (recoverable) protection faults (unrecoverable) floating point exceptions	• aborts • ex: illegal inst., parity error, machine check

page fault= when a software program attempts to access a memory block not currently stored in RAM + physical memory
- after exception, page is copied from disk to memory in os kernel
- return and reexecute the current instruction

invalid memory reference= after page fault exception, if invalid address is detected, SIGSEGV signal is sent to user program → segmentation fault

privileged instructions= the inst. that are only executed in kernel mode
- user applications do not include, only system call code include

system calls (syscall)= function that a user program uses to ask the OS for a particular service
- read, write, open, close, stat, fork, execve, _exit, kill
- these functions can directly access hardware in a controlled way
- process control, file management, device management, information maintance, communication
- Common system API: POSIX API (linux, unix, macos x), Win32 API (windows)

System program= link between the user interface and system calls → rm, ls, gcc, ssh, vim...