go up stack, now @ OS

still use registers nowadays %eip & %rip used
32 bit 64 bit

sometimes won't distinguish

processor does only "one thing"

who gives it instructions?

CPU != computer

need an OS to pass instructions

OS is just another program
linux is just hella C files

run objdump -> hella ft's

program live in disk & loaded into memory

8 bytes memory : [0x0, 0x8)
32 bytes : [0x0, 0x20)
4 GB : [0x0, 0x1 0000 0000)

program first onto memory from disk, then get instr. from program counter register %rip

put main() to CPU

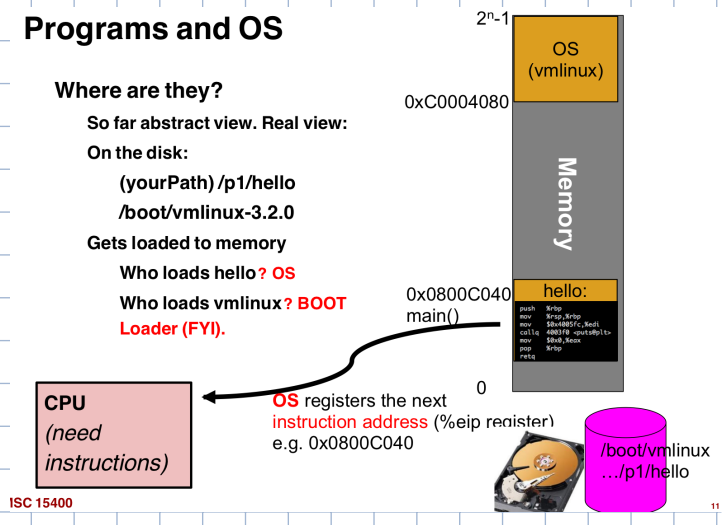
what loads OS to memory?
done by hardware bootloader firmware

what about infinite loops?
then print "A"
wait until loop finish then print? no
can run multiple programs simultaneously

program interrupted every 10ms

network "interrupt"

OS deals w/ interruptions



divide by zero
CPU can catch error, run handler

fn's to deal w/hardware
OS

need mechanism to break flow

OS exceptions
① exception handlers (by developers)
② exception table (hardware)

Exceptions

```
TimerInterrupt() { // every 10 ms
  ScheduleProcesses();
  Ex: P1 just ran, so now P2's turn;
}

DivideByZeroError() {
  // e.g. currentP just did x/0
  kill(currentProcess);
  popUpWindow("hey something wrong");
  scheduleProcesses();
}

Syscall_handler (int arg) {
  if (arg==READ) sys_read(...);
  ...
}

networkInterrupt() {
  // read data from network card
}
```

C code

OS has fn's w/memory

waiting to be called by...?

Exception/Interrupt Table

Built inside the CPU

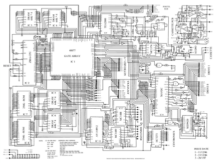
256 entries

Initialization

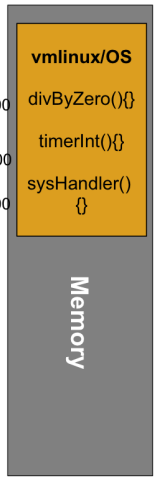
The boot loader boots the OS

OS initialization code install addresses of exception handlers to exception table

Exc#	Exception handler addresses
0	0xC0003000
...	...
128	0xC0001000
...	...
200	0xC0002000



Excep-tion #	Description	Actual Functions
0	Divide Error	DivideByZeroError( )
128	Syscall	SyscallHandler()
200	Timer interrupt	TimerInterrupt()
(See Fig. 8.9 in CSAPP 15400		



hardware table

agree on what each exception num means

Exc#

OS exception: both asynchronous & synchronous