University of BRISTOL

Overview of Computer Architecture

Further Execution and Operating Systems

Carl Henrik Ek - carlhenrik.ek@bristol.ac.uk

December 2, 2019

http://carlhenrik.com

Reminder of the Taught Unit

Today Further Execution and Operating Systems

Friday Operating Systems

Monday Introduction to High Level Languages

Friday High Level Languages

Reminder of the Practical

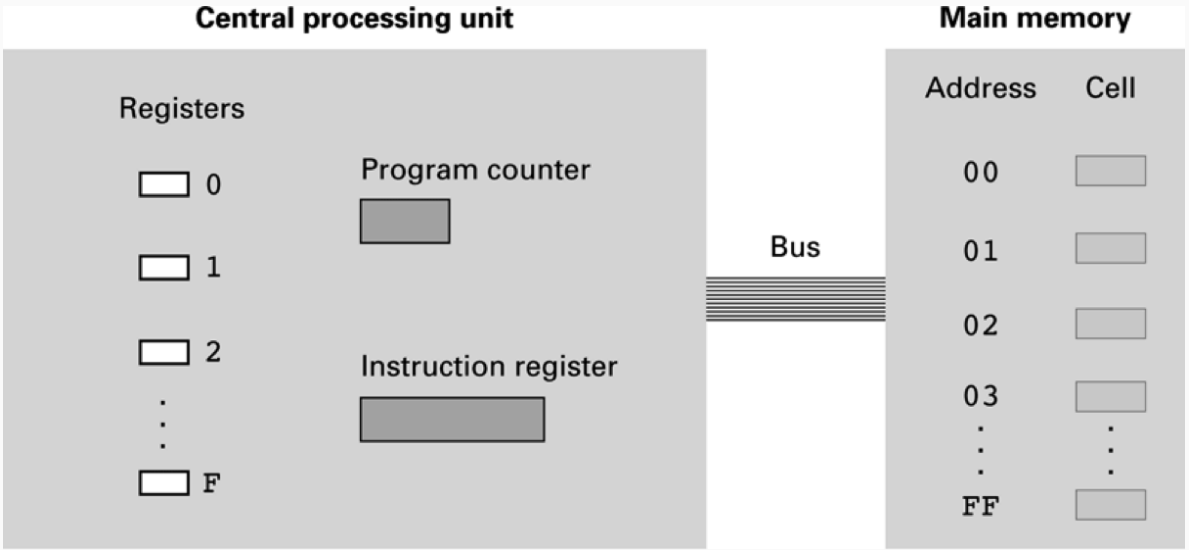
Today Continue with Gaussian elemination

Monday High Level Languages

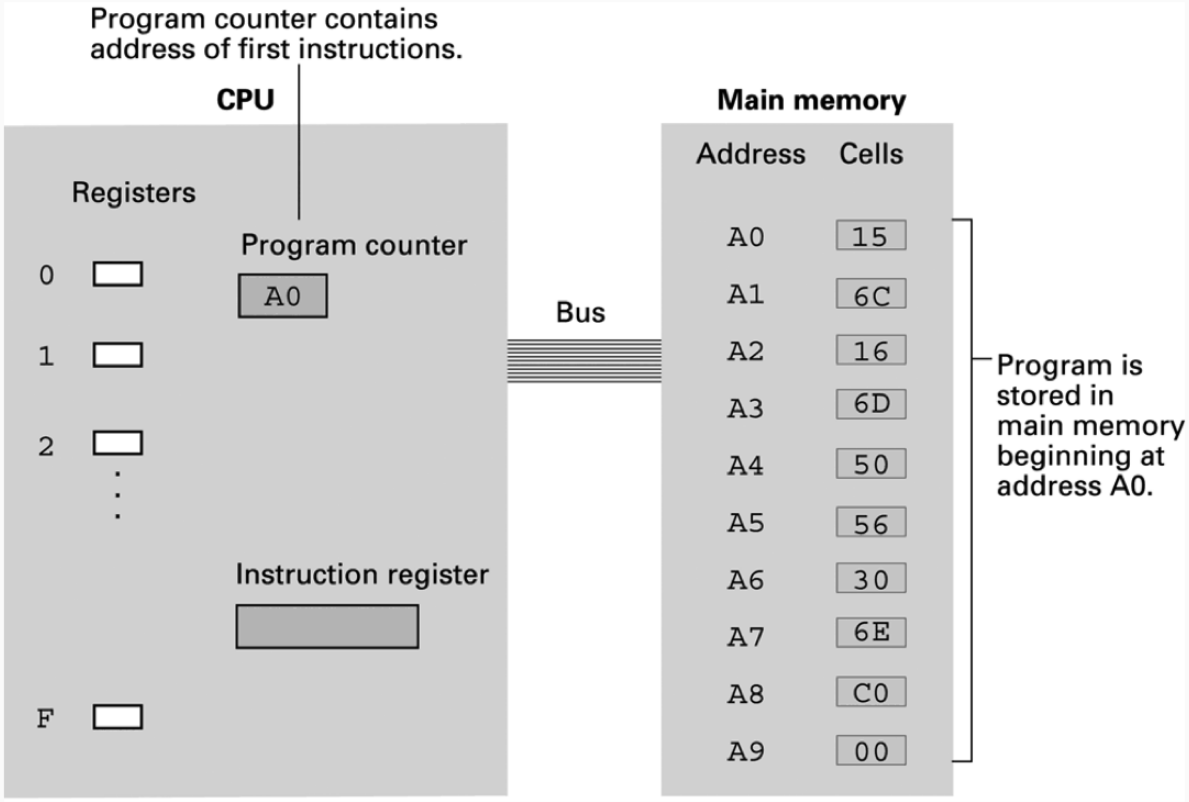
Test

Monday 16th of December

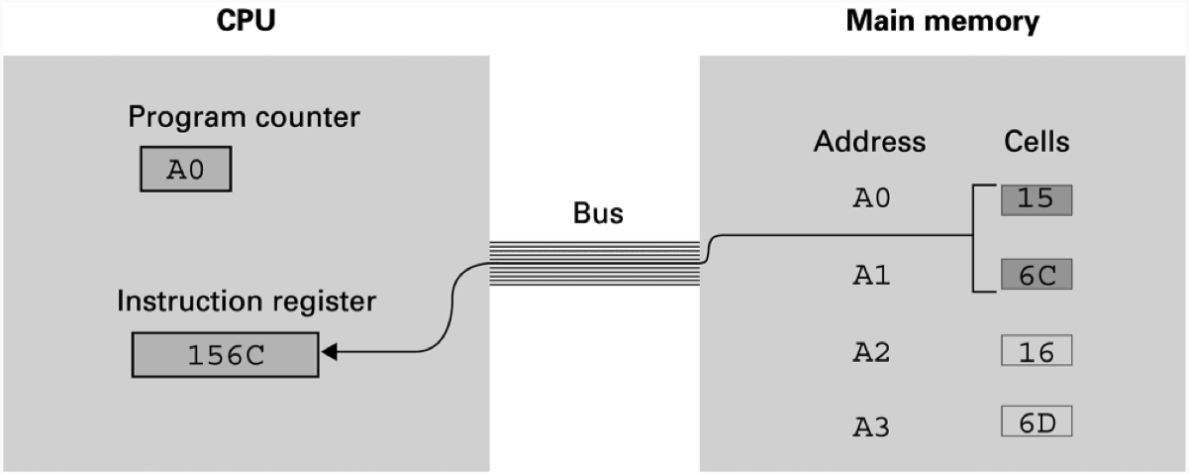
Execution


Central processing unit
Main memory
Registers
0
1
2
F
Program counter
Instruction register
Bus
Address
Cell
00
01
02
03
FF

Execution

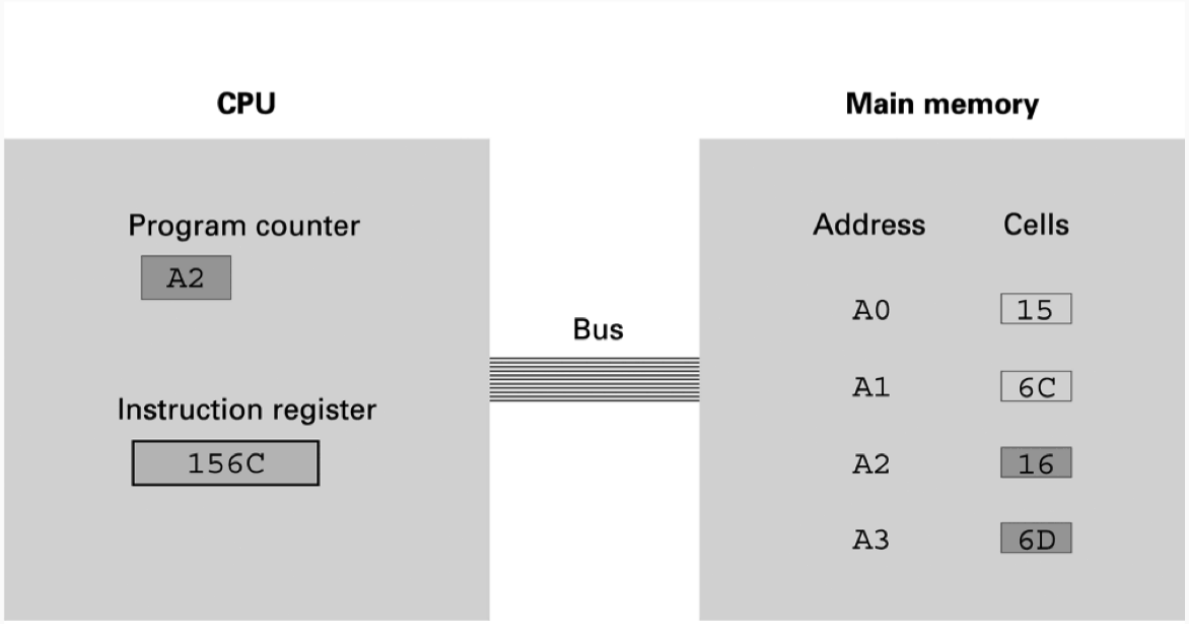

Program counter contains address of first instructions.
CPU
Main memory
Registers
Program counter
A0
0
1
2
F
Bus
Address
Cells
A0 15
A1 6C
A2 16
A3 6D
A4 50
A5 56
A6 30
A7 6E
A8 C0
A9 00
Instruction register
Program is stored in main memory beginning at address A0.

Execution



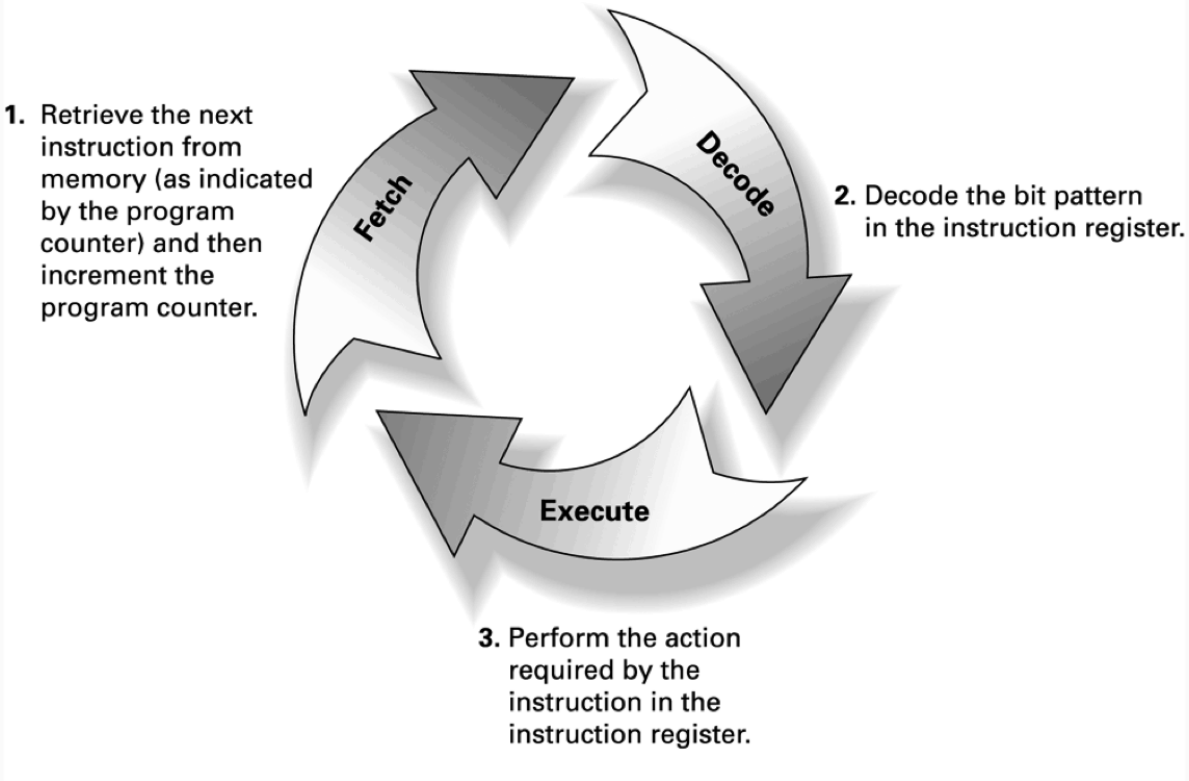
a. At the beginning of the fetch step the instruction starting at address A0 is retrieved from memory and placed in the instruction register.

Execution



b. Then the program counter is incremented so that it points to the next instruction.

Execution


1. Retrieve the next instruction from memory (as indicated by the program counter) and then increment the program counter.
Fetch
Decode
2. Decode the bit pattern in the instruction register.
Execute
3. Perform the action required by the instruction in the instruction register.

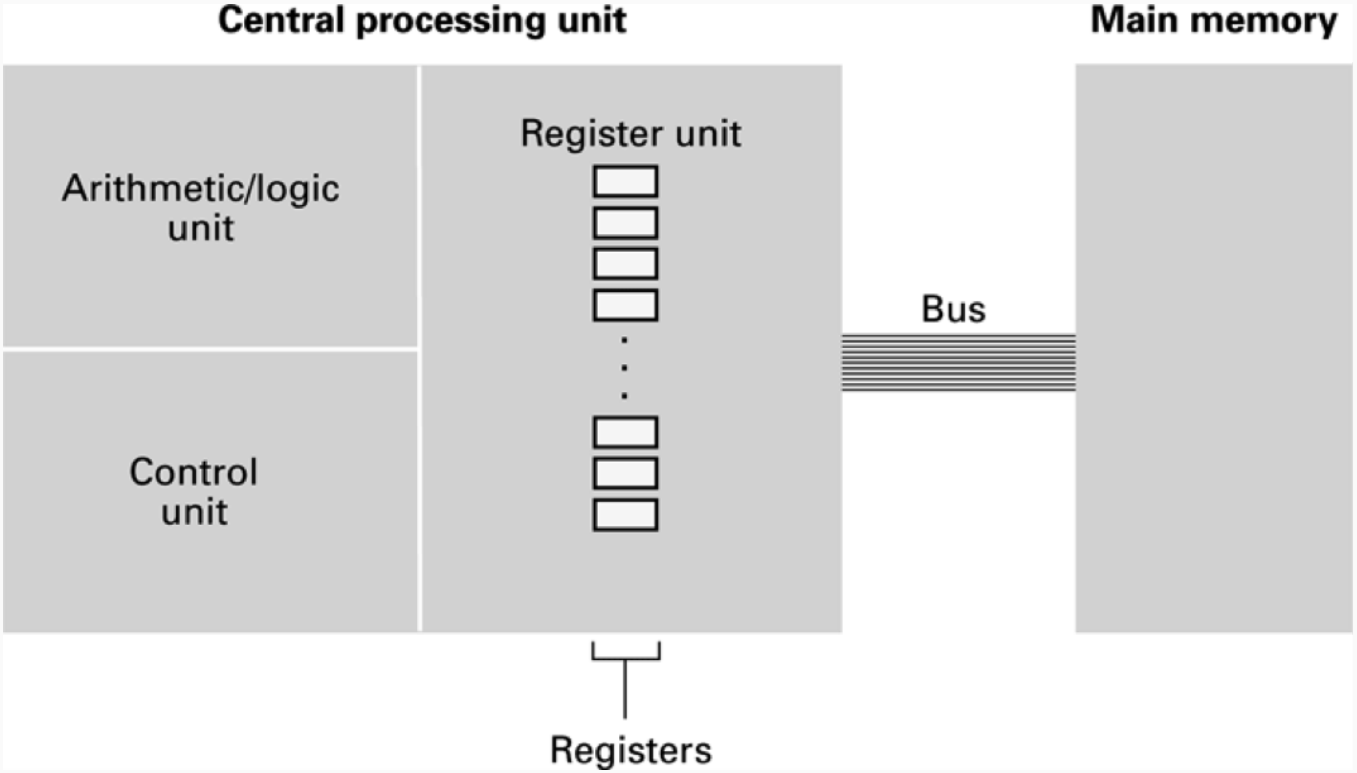
a) Potential energy

(b) Kinetic energy

Context Switch

Context Switch

CPU


Central processing unit
Main memory
Arithmetic/logic unit
Control unit
Register unit
Bus
Registers

Context Switch

Code

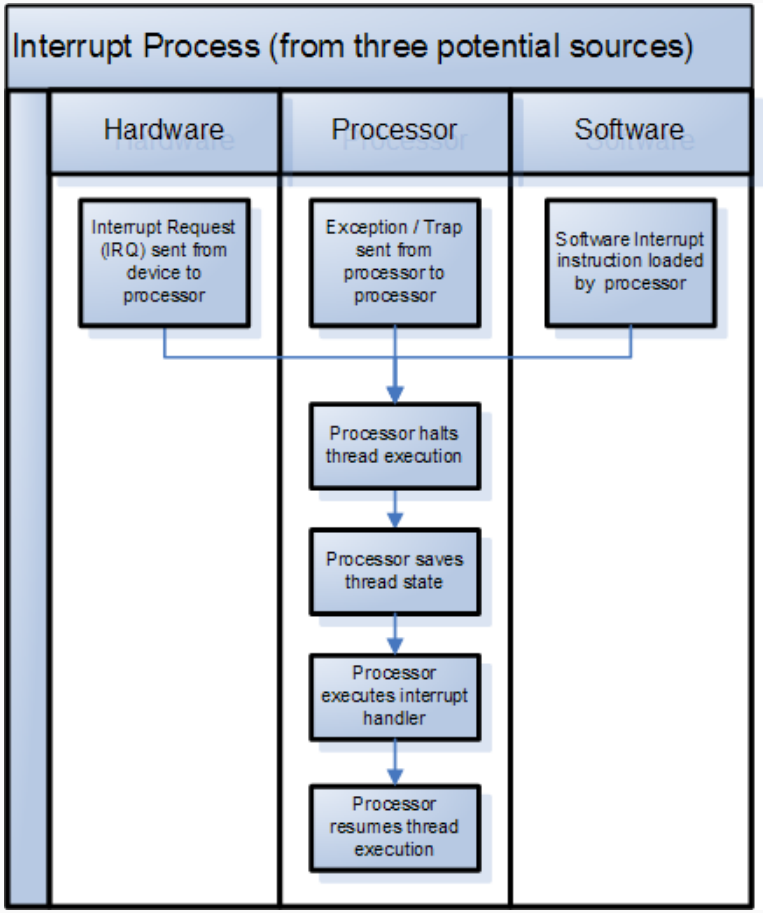
```
_context1
        mov         sp, #0xffabcdefgh
        ldmia         sp!, {r0-r16}

        b         _context1

_context2
        mov         sp, #0xabcdefghff
        ldmia         sp!, {r0-r16}

        b         _context2
```

Key Press

Vertical Blanking

Interrupts

- Software `instruction`
- Processor `TRAP`
- Hardware `IRQ`



Software Interrupts

Code

```
swi         0x11
```

Software Interrupts

Code

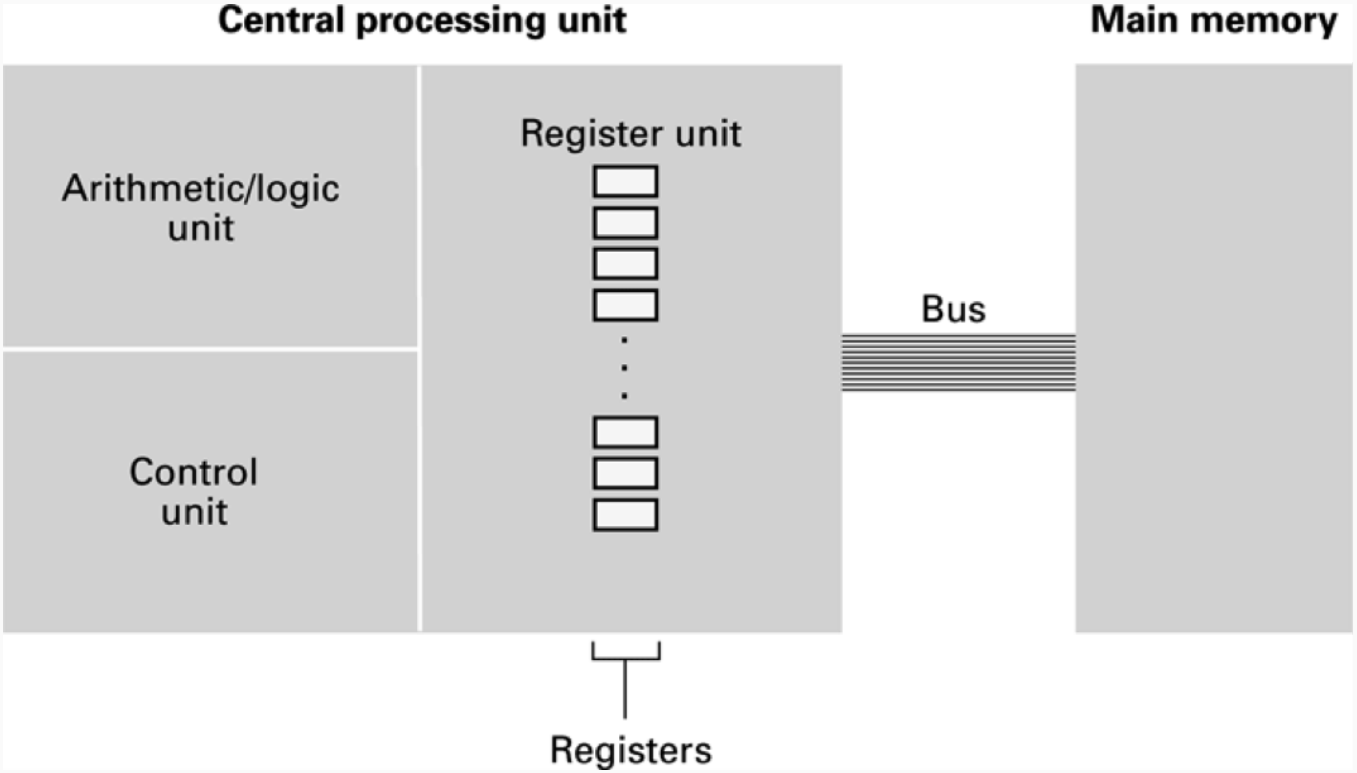
```
> cat /usr/include/asm-generic/unistd.h
> cat /usr/include/arm-linux-gnueabihf/asm/unistd.h
> #define __NR_restart_syscall (__NR_SYSCALL_BASE+ 0)
> #define __NR_exit            (__NR_SYSCALL_BASE+ 1)
> #define __NR_fork            (__NR_SYSCALL_BASE+ 2)
> #define __NR_read            (__NR_SYSCALL_BASE+ 3)
> #define __NR_write           (__NR_SYSCALL_BASE+ 4)
```

TRAP

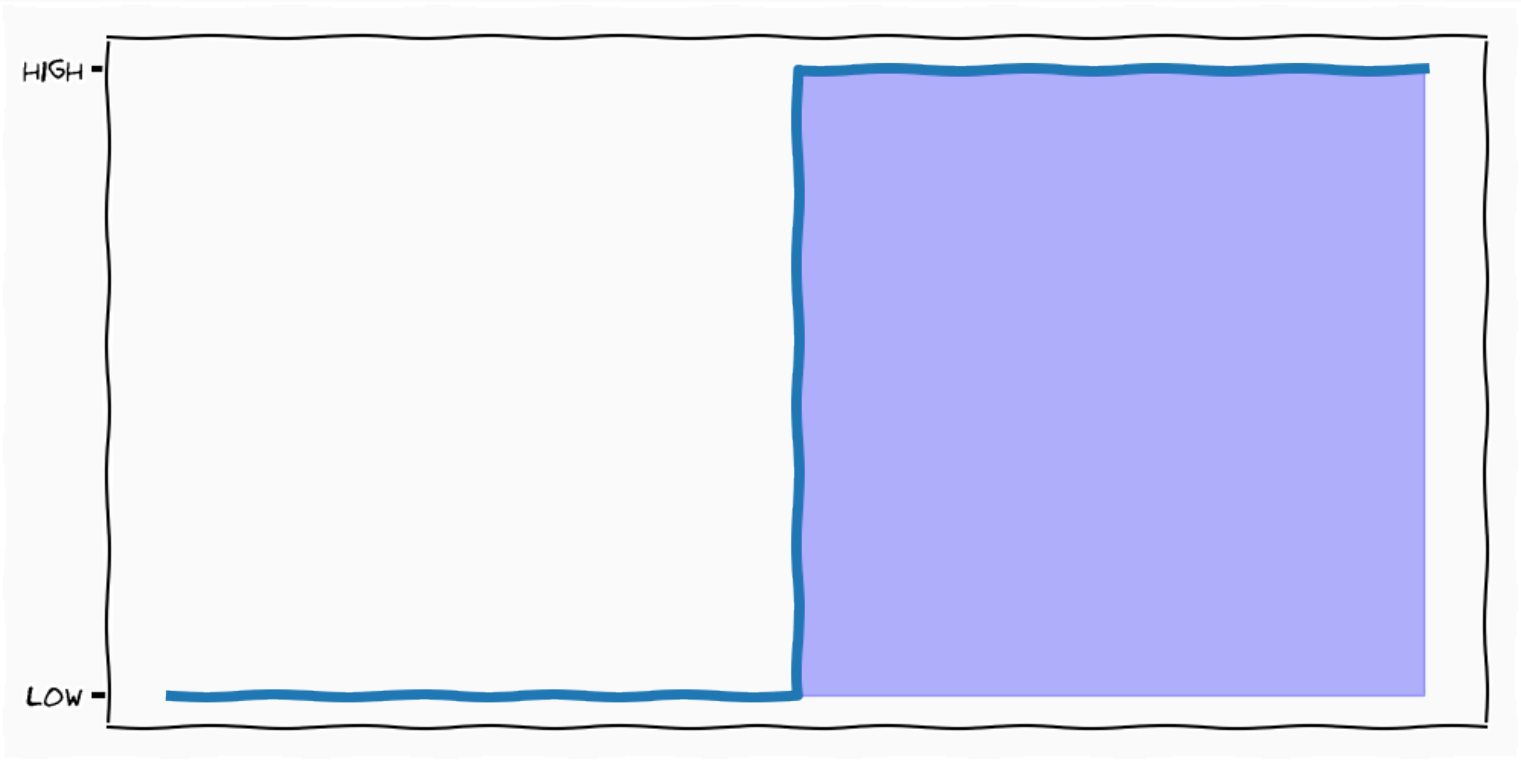
Code

```
mov         r0, #42
eor         r1, r1, r1
sdiv         r2, r1, r0
```

CPU


Central processing unit
Main memory
Arithmetic/logic unit
Control unit
Register unit
Bus
Registers

IRQ



- ARM has 72 dedicated IRQ lines
- x86 has 16 dedicated IRQ lines
- =cat *proc/interrupts*

Interrupt is Triggered

1. Interrupt is triggered

Interrupt is Triggered

1. Interrupt is triggered
2. Disable Interrupts

Interrupt is Triggered

1. Interrupt is triggered
2. Disable Interrupts
3. Processor halts execution

Interrupt is Triggered

1. Interrupt is triggered
2. Disable Interrupts
3. Processor halts execution
4. Save state of current context

Interrupt is Triggered

1. Interrupt is triggered
2. Disable Interrupts
3. Processor halts execution
4. Save state of current context
5. Call Interrupt Handler

Interrupt is Triggered

1. Interrupt is triggered
2. Disable Interrupts
3. Processor halts execution
4. Save state of current context
5. Call Interrupt Handler
6. Restore state of previous context

Interrupt is Triggered

1. Interrupt is triggered
2. Disable Interrupts
3. Processor halts execution
4. Save state of current context
5. Call Interrupt Handler
6. Restore state of previous context
7. Enable Interrupts

Interrupt Vector Table

Interrupt ID	Address of Routine
IRQ 0	0x000fff00
IRQ 7	0x0ff0ff00

/proc/interrupts

Code

```
cat /proc/interrupts
```

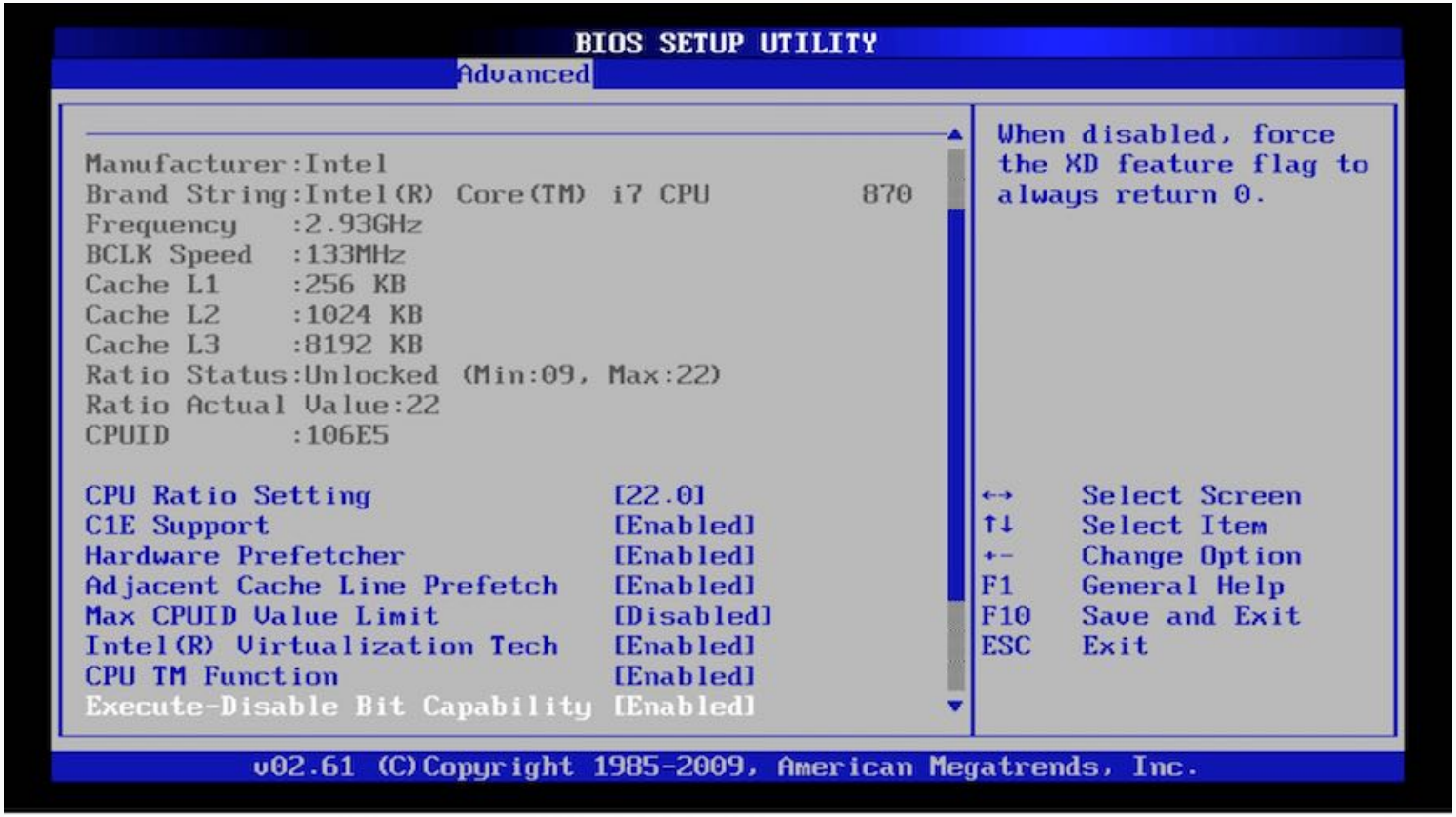
Operating Systems

Start

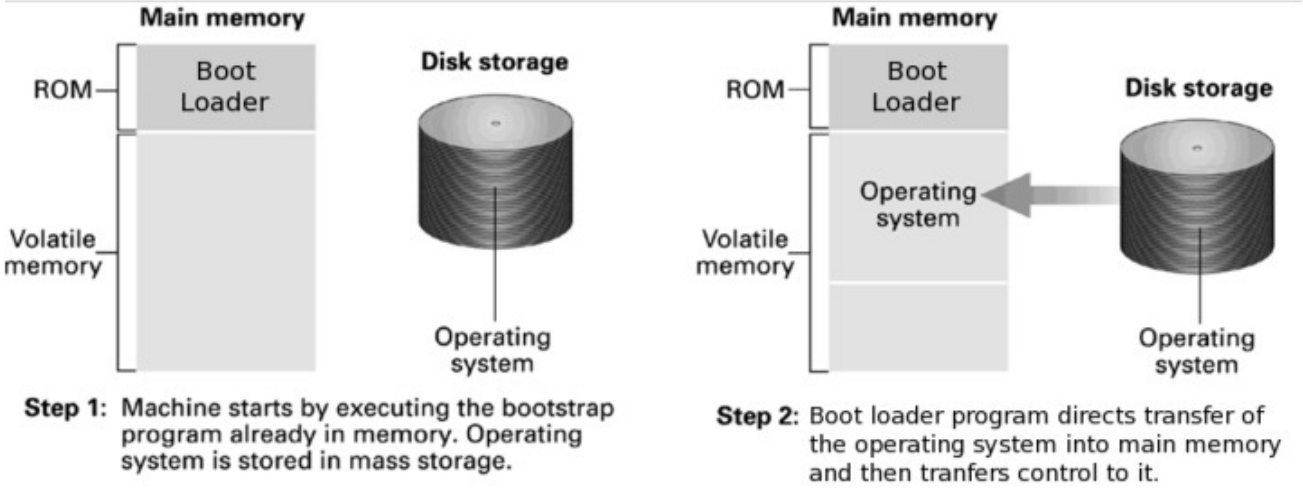
Bootstrapping[1]

[1] https://en.wikipedia.org/wiki/Baron_Munchausen

BIOS


BIOS SETUP UTILITY
Advanced
Manufacturer:Intel
Brand String:Intel(R) Core(TM) i7 CPU 870
Frequency :2.93GHz
BCLK Speed :133MHz
Cache L1 :256 KB
Cache L2 :1024 KB
Cache L3 :8192 KB
Ratio Status:Unlocked (Min:09, Max:22)
Ratio Actual Value:22
CPUID :106E5
CPU Ratio Setting [22.0]
C1E Support [Enabled]
Hardware Prefetcher [Enabled]
Adjacent Cache Line Prefetch [Enabled]
Max CPUID Value Limit [Disabled]
Intel(R) Virtualization Tech [Enabled]
CPU TM Function [Enabled]
Execute-Disable Bit Capability [Enabled]
When disabled, force the XD feature flag to always return 0.
↔ Select Screen
↑↓ Select Item
+- Change Option
F1 General Help
F10 Save and Exit
ESC Exit
v02.61 (C)Copyright 1985-2009, American Megatrends, Inc.

BIOS


Main memory
ROM
Boot Loader
Volatile memory
Disk storage
Operating system
Step 1: Machine starts by executing the bootstrap program already in memory. Operating system is stored in mass storage.
Main memory
ROM
Boot Loader
Operating system
Volatile memory
Disk storage
Operating system
Step 2: Boot loader program directs transfer of the operating system into main memory and then tranfers control to it.

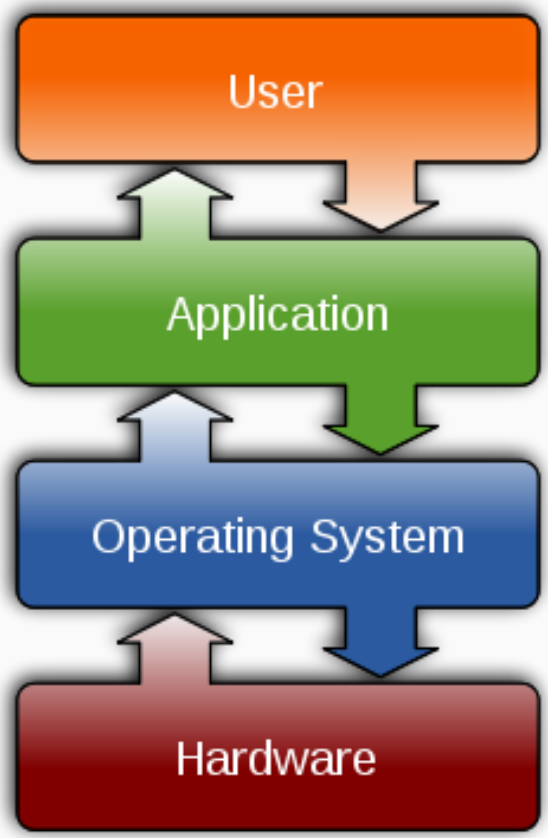
BIOS

- Checks hardware
- Sets up some interrupts
- Sets up boot device/priority
- Leave control to `boot loader`

Operating System

Tasks

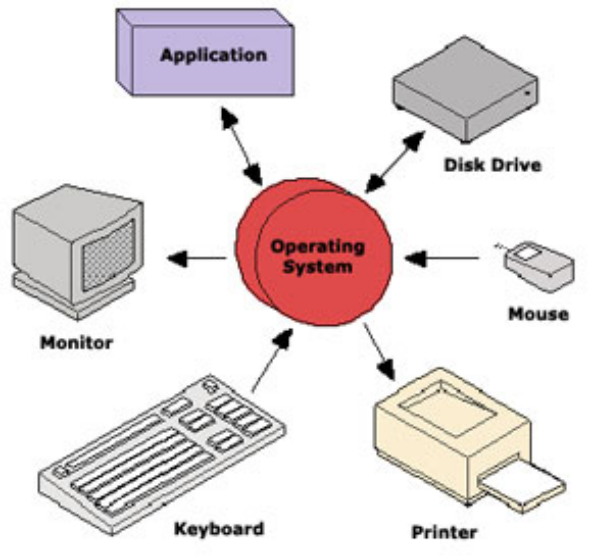
- Computer Startup
- Abstracts Hardware
- Manage Resources
- Provides Application Layer
- Schedule Applications



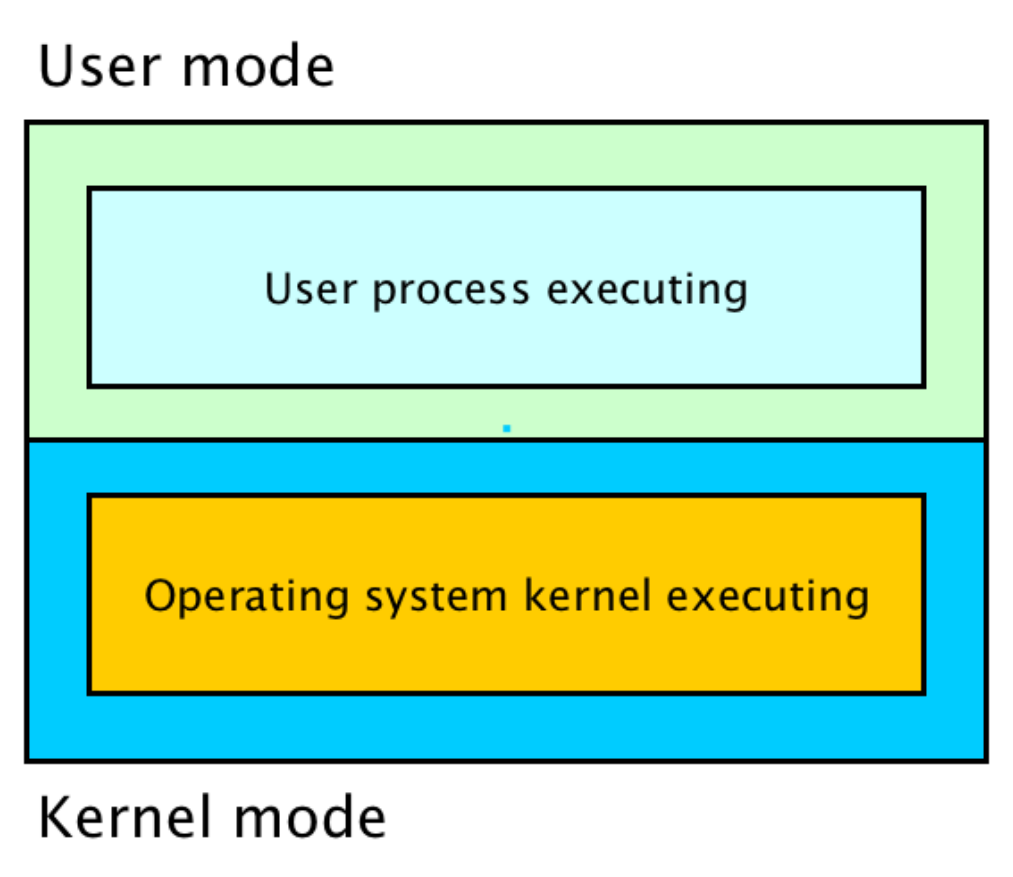
Operating System

Resource Management

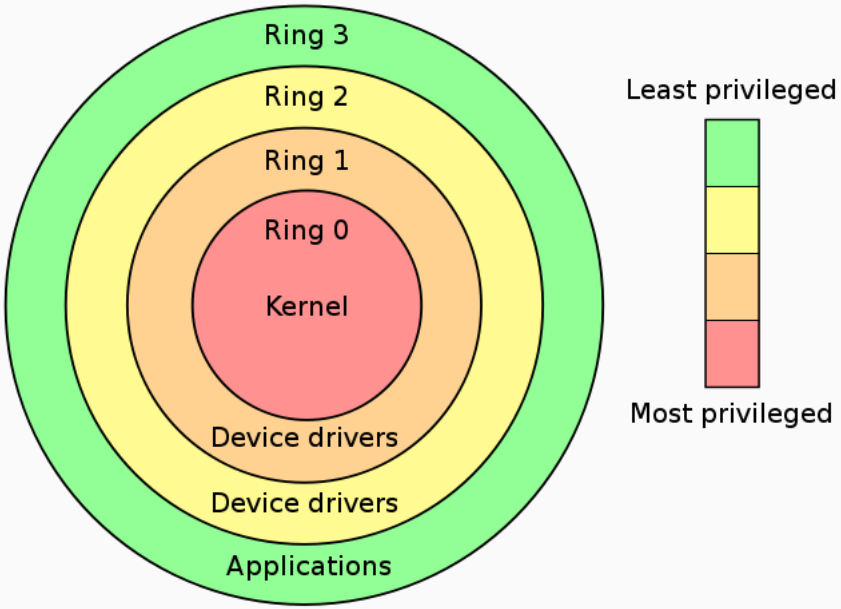
- Schedule Execution
- Coordinate Execution
- Manage Memory



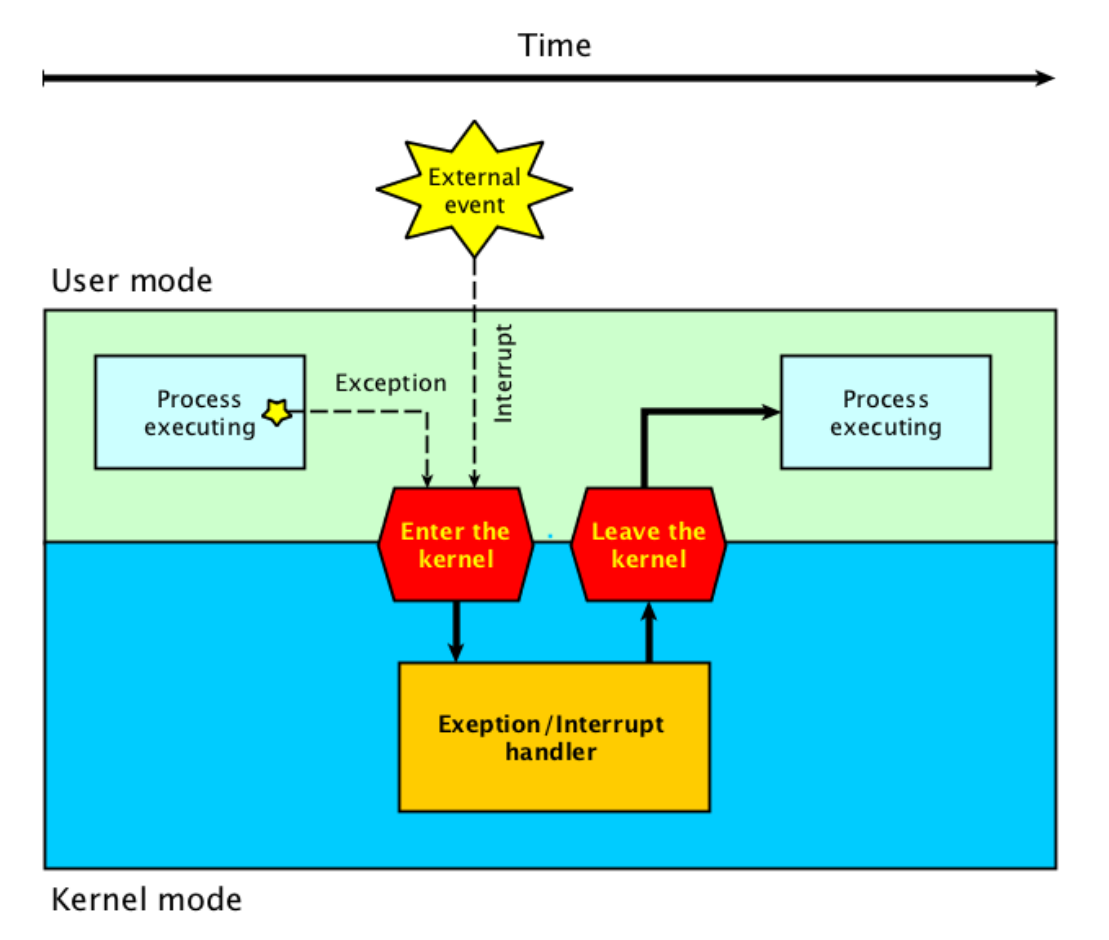
User/Kernel Mode



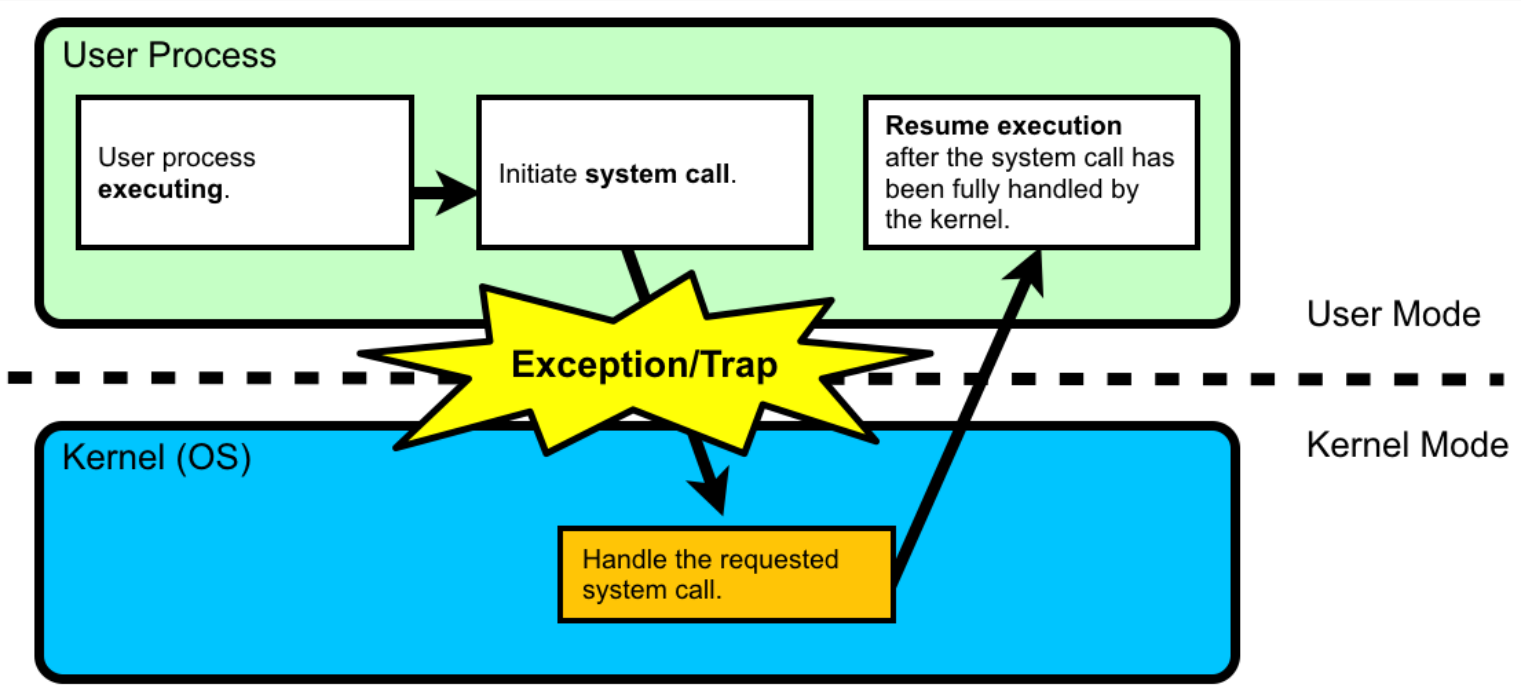
User/Kernel Mode



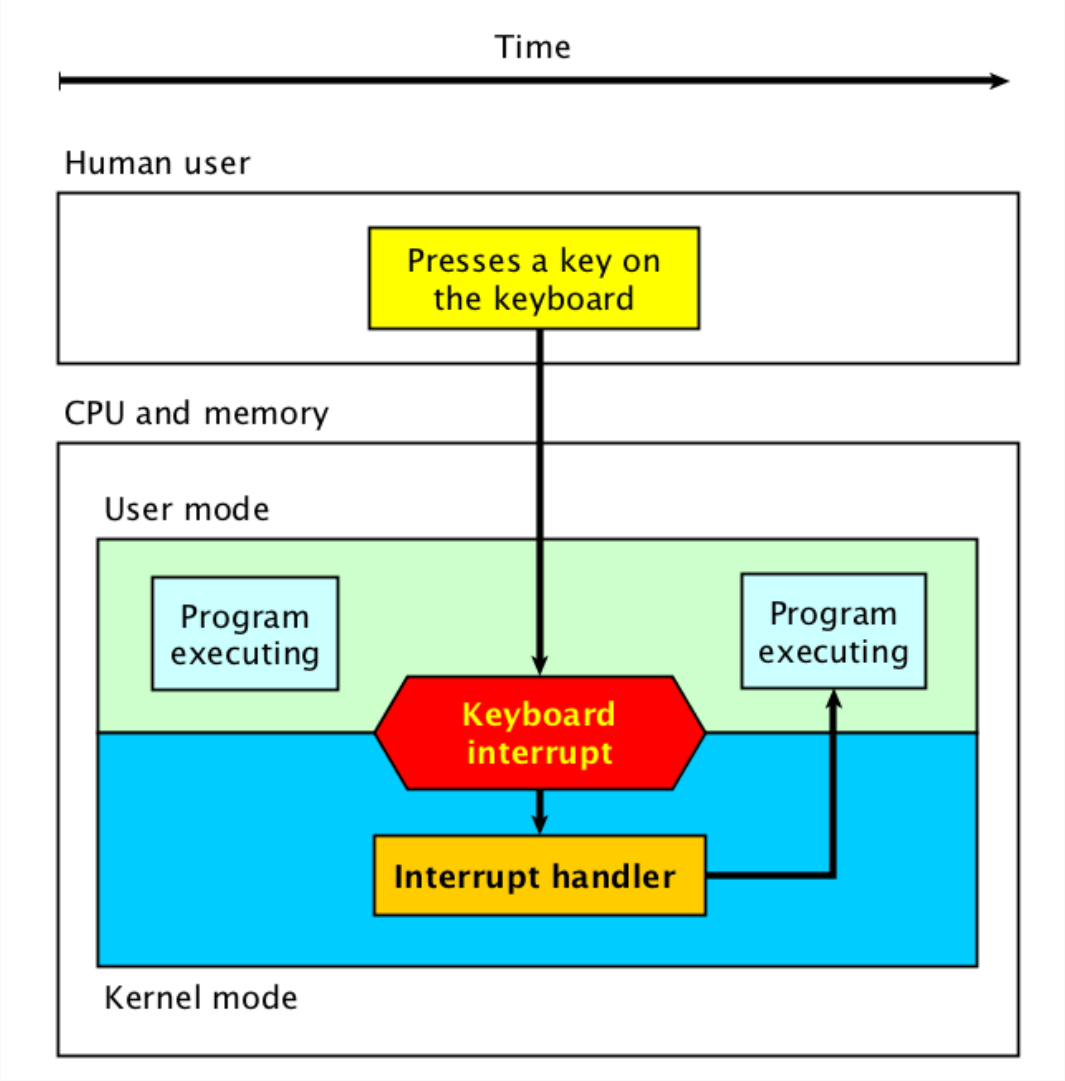
Interrupt


Time
External event
User mode
Process executing
Exception
Interrupt
Process executing
Enter the kernel
Leave the kernel
Exeption/Interrupt handler
Kernel mode

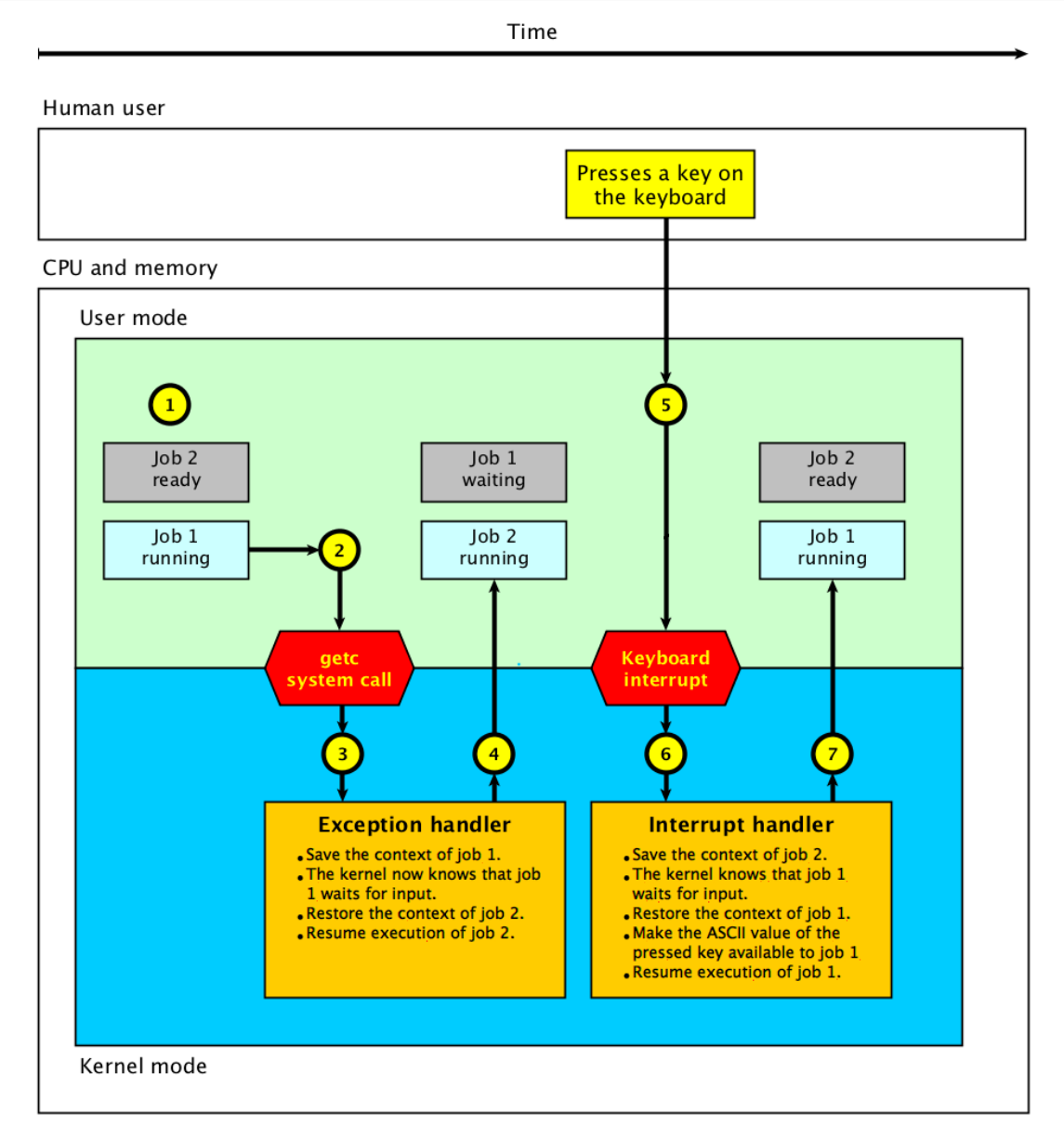
Interrupt


User Process
User process executing.
Initiate system call.
Resume execution after the system call has been fully handled by the kernel.
Exception/Trap
User Mode
Kernel Mode
Kernel (OS)
Handle the requested system call.

Keyboard


Time
Human user
Presses a key on the keyboard
CPU and memory
User mode
Program executing
Program executing
Keyboard interrupt
Interrupt handler
Kernel mode

Multiple Processes



Multiple Processes

Code

```
ps -e | wc -l
```

Operating Systems

- Program is static
- Execution is not

Summary

Summary

- Interrupts
  - allows to interrupt execution
  - software/trap/hardware
- Operating Systems
  - Booting
  - Resource Management

Next Time

- Operating Systems
  - Scheduling
  - Memory Management