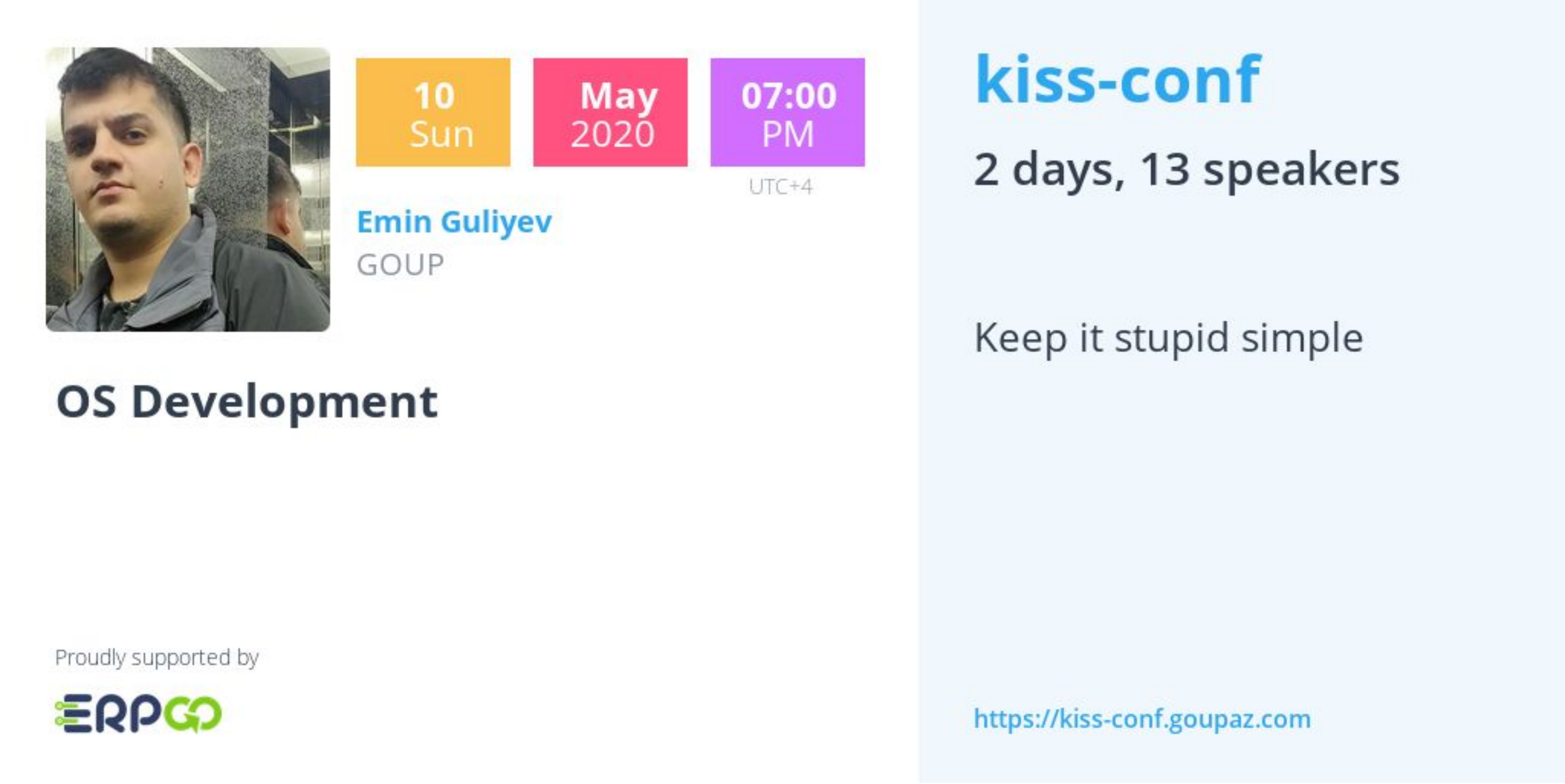


10 Sun

May 2020

07:00 PM

UTC+4

Emin Guliyev

GOUP

OS Development

Proudly supported by

ERPGO

kiss-conf

2 days, 13 speakers

Keep it stupid simple

https://kiss-conf.goupaz.com

Kiss.Conf 2020

Host: Sako M

OS development and concept

Emin Ghuliev, Goup

May 10, 2020 7:00-9:00 AM UTC

Emin Ghuliev

Compiler, microarchitecture and low-level stuff lover and hacker

Contact

Twitter: twitter.com/drm_gh

Education

Self-taught ninja

Experience

Current:

Maintainer @ GOUP

Past:

CERT Azerbaijan, APA Holding, e-Gov Development Center, Ensign

Overview

- Hardware components for Operating Systems
- Operating System concepts
- Development toolkit
- Safety in Operating system development (with Rust and Clang++)

Hardware components/unit

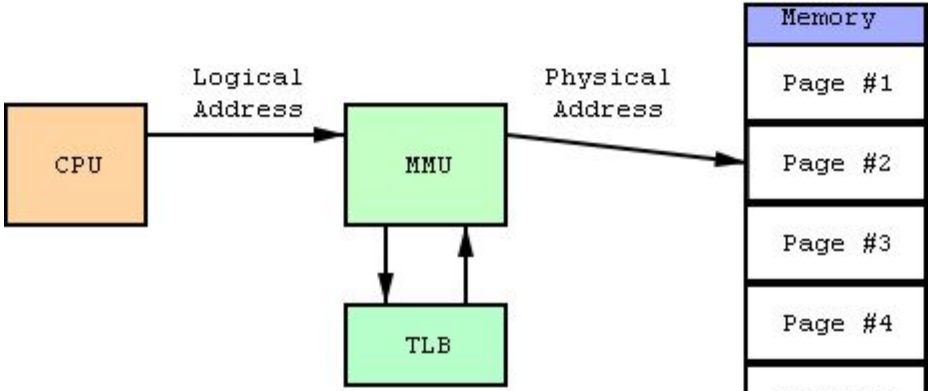
- MMU
- Interrupt Controller
- Clocks, Timers, Counters
- Input/Output
- Storage devices
- PCI/USB
- Network

OS components

- Bootloader
- Memory management (Paging, segmentation etc)
- Scheduler
- Multitasking/Multiprocessing
- System calls
- Device drivers

MMU

- A20 line (talk in bootloader slide)
- Segmentation
- Paging



TLB: Translation Look-aside Buffer
MMU: Memory Management Unit
CPU: Central Processing Unit

Interrupt Controller

- PIC
- APIC
- SMP (Symmetric multiprocessing)

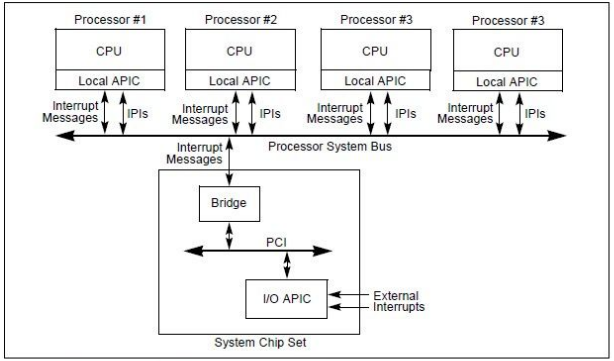
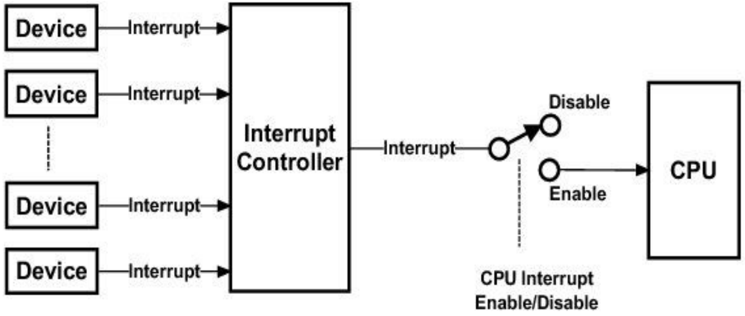


Figure 10-2. Local APICs and I/O APIC When Intel Xeon Processors Are Used in Multiple-Processor Systems



Interrupt Controller types

- Exception
- Interrupt Request (IRQ) or Hardware Interrupt
- Software Interrupt

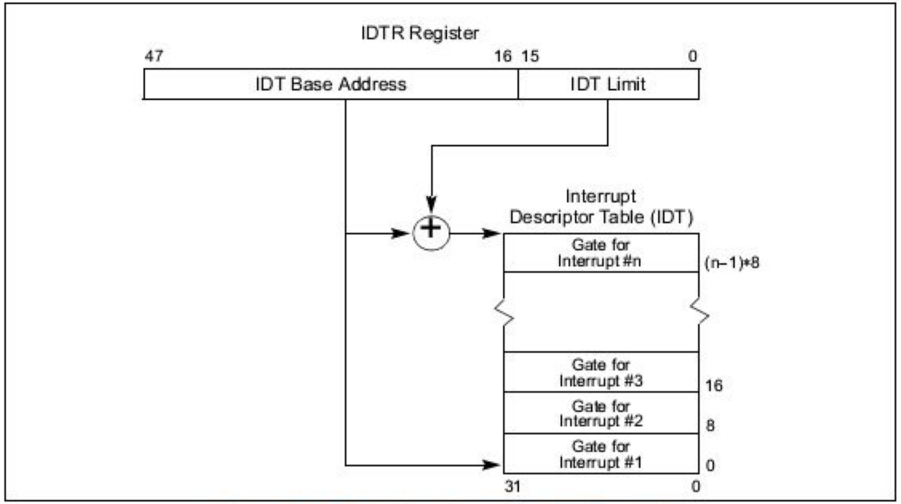
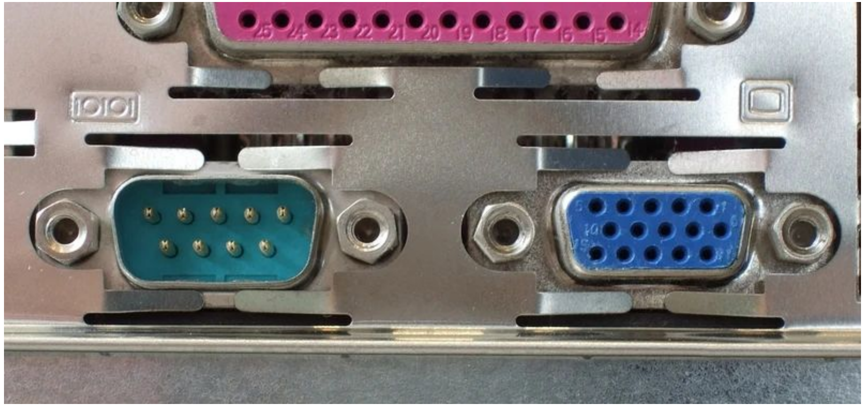


Figure 6-1. Relationship of the IDTR and IDT

Timers

Input/Output (PS2/Serial)

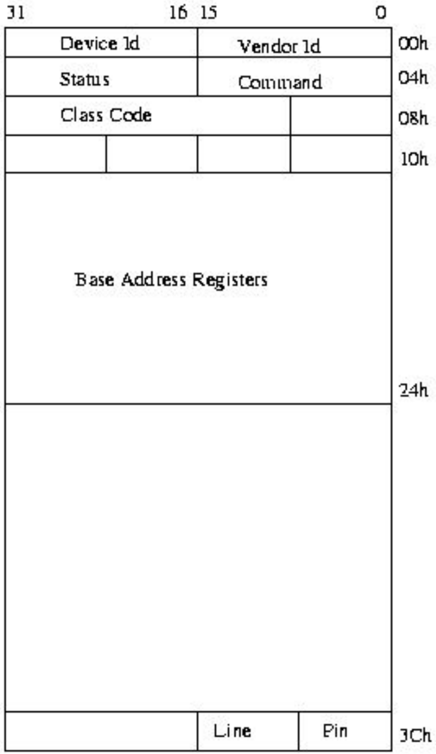


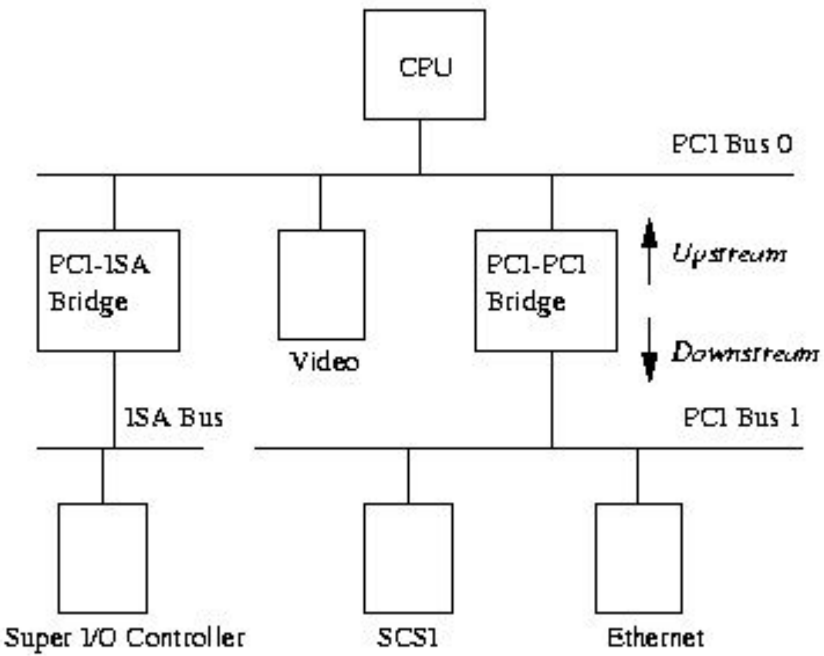
Storage Devices

- SATA
- ATAPI
- AHCI
- DMA

PCI/PCIe/USB

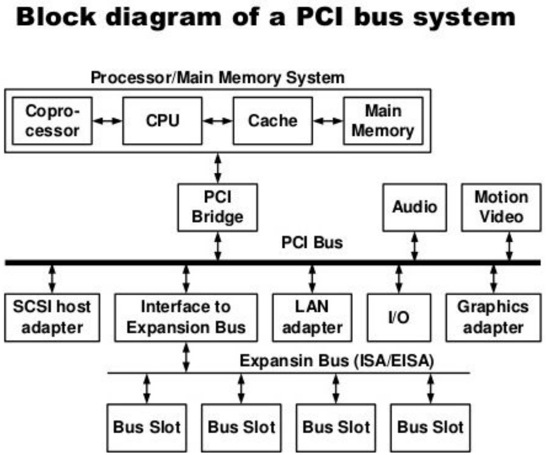
- PCI configuration space





Device drivers

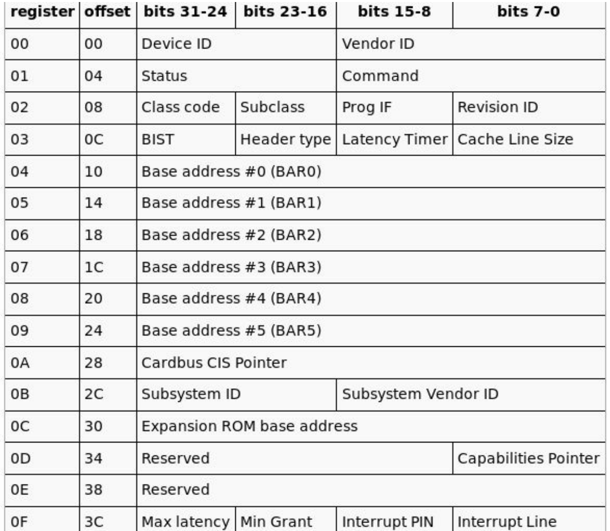
- PCI
    - First stage: PCI device discovery and initialization
    - Then OS will enumerate PCI buses to discovery devices.



Device drivers

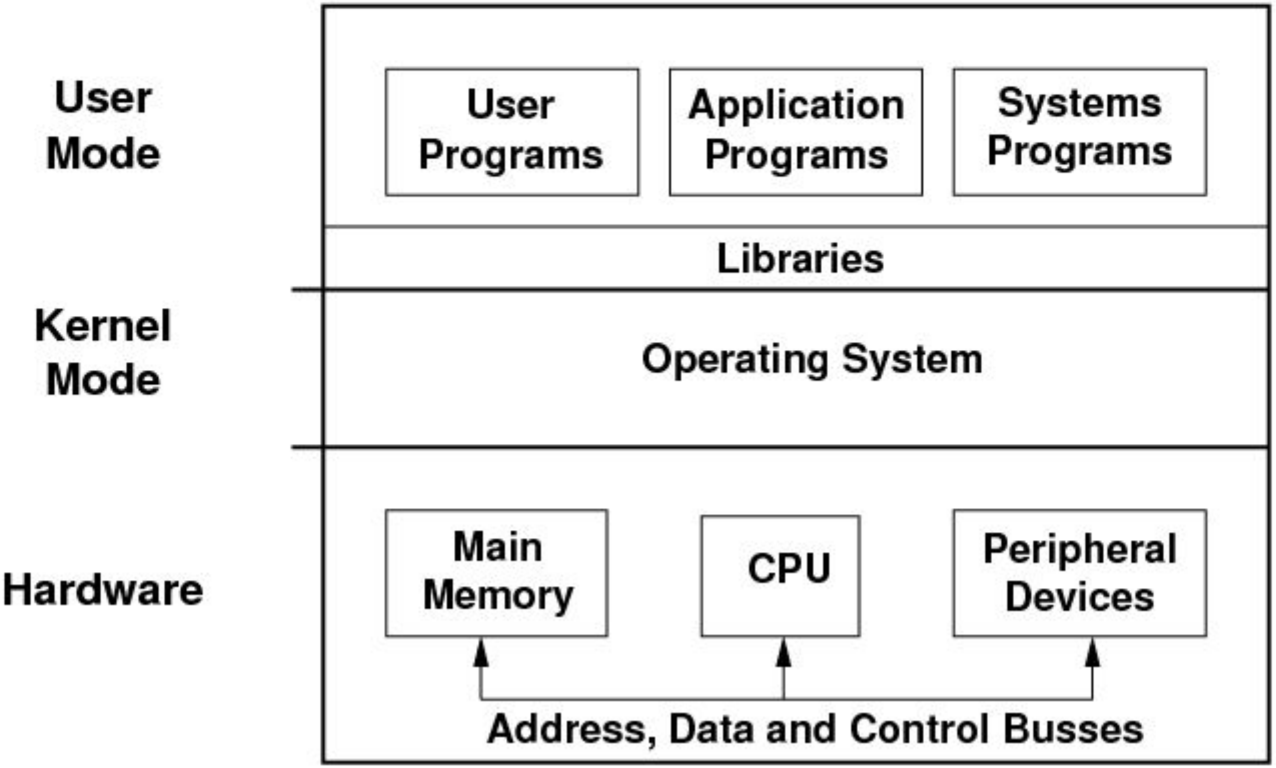
- PCI
  - The PCI specification provides for totally software driven initialization and configuration of each device (or target) on the PCI Bus via a separate Configuration Address Space.

256-byte Configuration Space registers



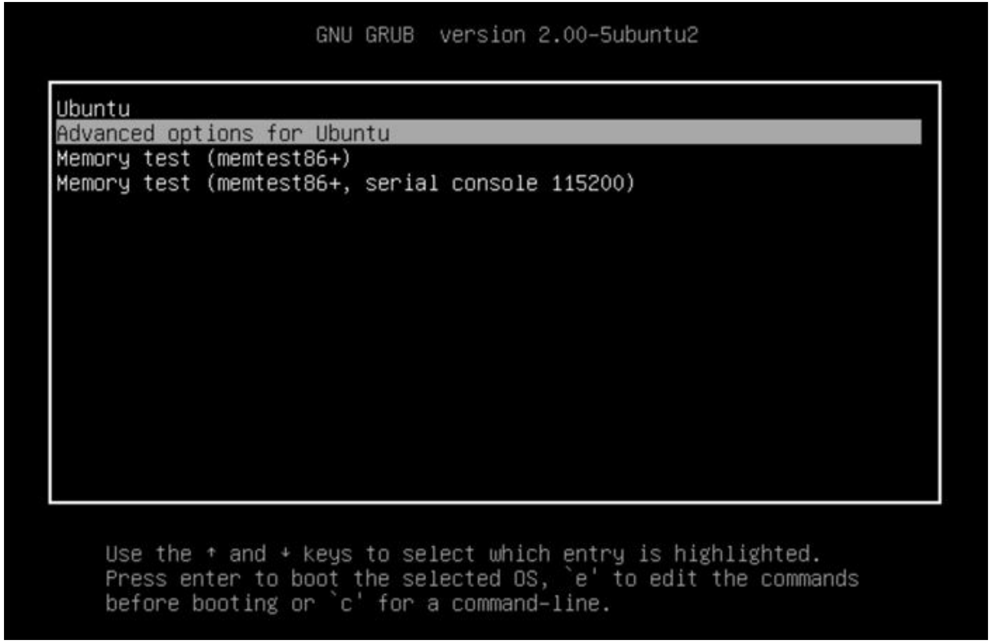
register	offset	bits 31-24	bits 23-16	bits 15-8	bits 7-0
00	00	Device ID		Vendor ID	
01	04	Status		Command	
02	08	Class code	Subclass	Prog IF	Revision ID
03	0C	BIST	Header type	Latency Timer	Cache Line Size
04	10	Base address #0 (BAR0)			
05	14	Base address #1 (BAR1)			
06	18	Base address #2 (BAR2)			
07	1C	Base address #3 (BAR3)			
08	20	Base address #4 (BAR4)			
09	24	Base address #5 (BAR5)			
0A	28	Cardbus CIS Pointer			
0B	2C	Subsystem ID		Subsystem Vendor ID	
0C	30	Expansion ROM base address			
0D	34	Reserved			Capabilities Pointer
0E	38	Reserved			
0F	3C	Max latency	Min Grant	Interrupt PIN	Interrupt Line

OS Components



Bootloader

- x86 processor will begin executing the instructions at address FFFF:0000 (Real mode)
- Physical address = (A * 0x10) + B (we can use just 16 bits of memory 2^16 = 64 kib)



```
                    GNU GRUB  version 2.00-5ubuntu2

 Ubuntu
 Advanced options for Ubuntu
 Memory test (memtest86+)
 Memory test (memtest86+, serial console 115200)

     Use the ↑ and ↓ keys to select which entry is highlighted.
     Press enter to boot the selected OS, `e' to edit the commands
     before booting or `c' for a command-line.
```

MMU (Memory Management Unit) Interaction

- Paging (Protections CPL0 - CPL3)
- Segmentation
  - GDT
  - LDT
- Memory Allocator Kernel Based (e.g buddy allocator, slub allocator)
- Memory Allocators User based (Heap)

Memory Management Unit

- Paging - divide memory into fixed-sized pages
- Segmentation - divide memory into segments
- MMU gives protection and limit mechanism
- MMU translates virtual address to physical address

Segmentation

- In real mode we use a logical address in the form A:B to address memory. This is translated into a physical address using the equation:
  - Physical address = (A * 0x10) + B
- In protected mode A selector represents an offset into a system table called the Global Descriptor Table (GDT).

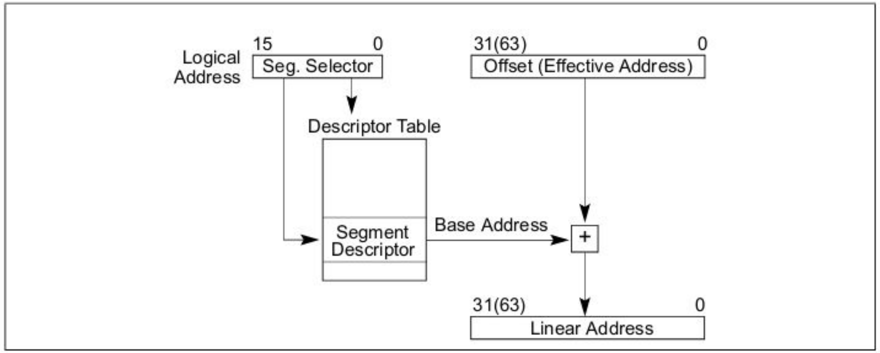
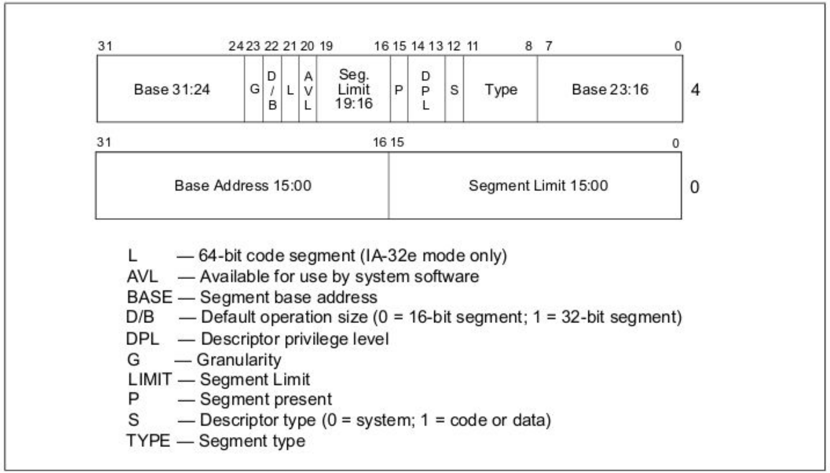


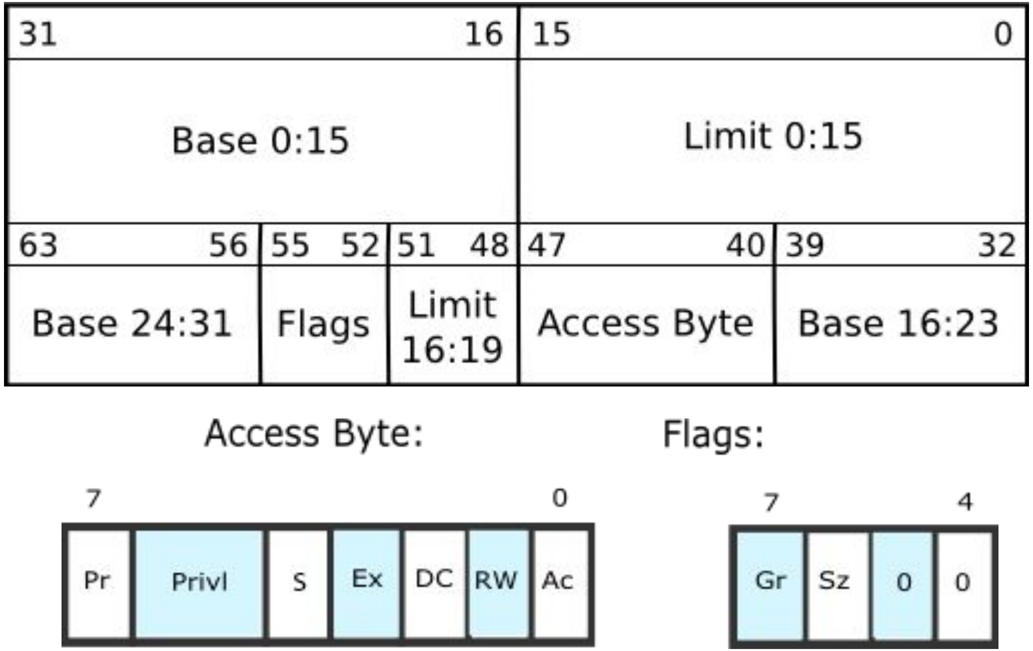
Figure 3-5. Logical Address to Linear Address Translation



L — 64-bit code segment (IA-32e mode only)
AVL — Available for use by system software
BASE — Segment base address
D/B — Default operation size (0 = 16-bit segment; 1 = 32-bit segment)
DPL — Descriptor privilege level
G — Granularity
LIMIT — Segment Limit
P — Segment present
S — Descriptor type (0 = system; 1 = code or data)
TYPE — Segment type

Figure 3-8. Segment Descriptor

Descriptor fields



31 16	15 0
Base 0:15	Limit 0:15

63 56	55 52	51 48	47 40	39 32
Base 24:31	Flags	Limit 16:19	Access Byte	Base 16:23

Access Byte:

7						0
Pr	Privl	S	Ex	DC	RW	Ac

Flags:

7			4
Gr	Sz	0	0

Segment descriptors

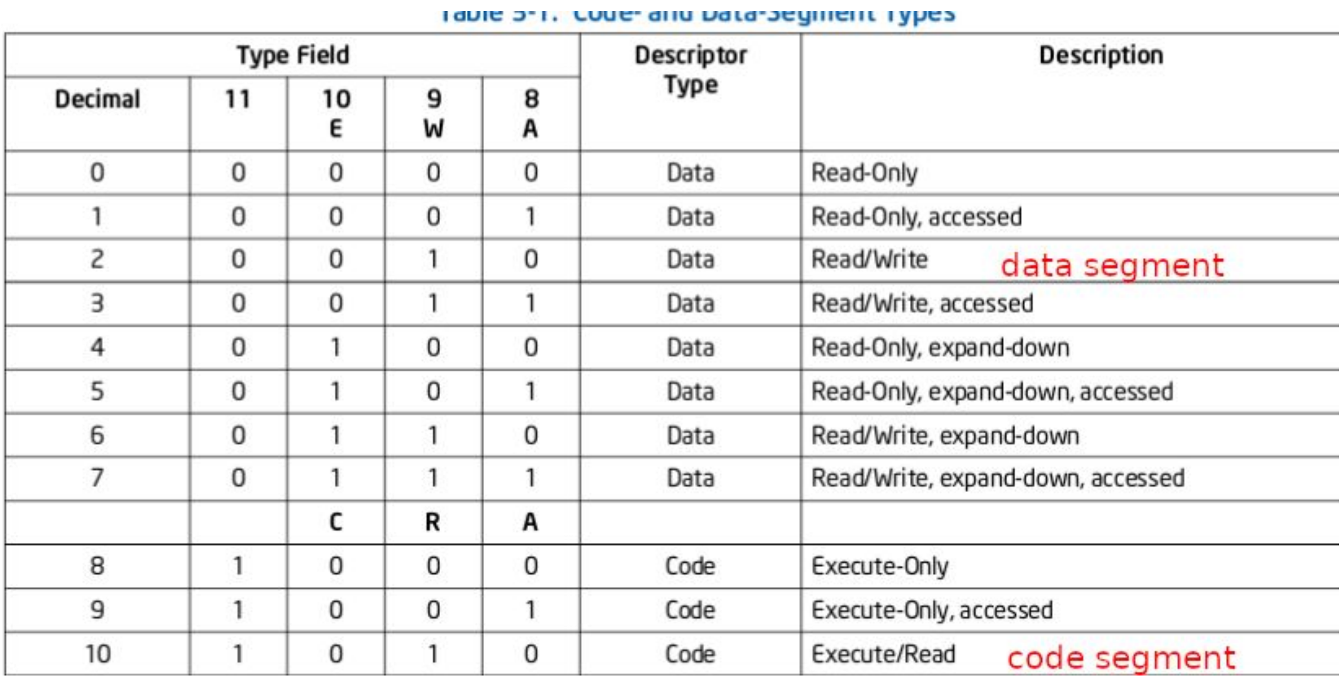


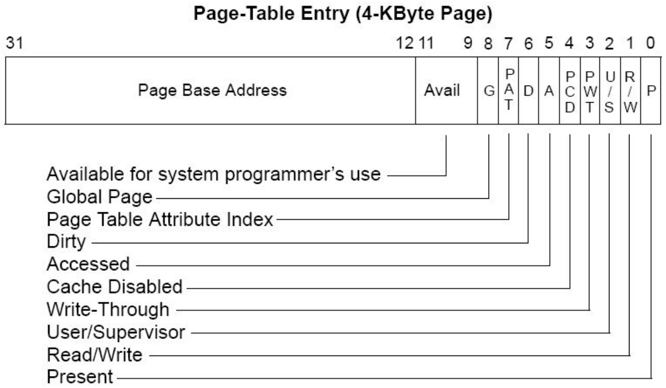
Table 3-1. Code- and Data-Segment Types

Type Field					Descriptor Type	Description
Decimal	11	10 E	9 W	8 A		
0	0	0	0	0	Data	Read-Only
1	0	0	0	1	Data	Read-Only, accessed
2	0	0	1	0	Data	Read/Write data segment
3	0	0	1	1	Data	Read/Write, accessed
4	0	1	0	0	Data	Read-Only, expand-down
5	0	1	0	1	Data	Read-Only, expand-down, accessed
6	0	1	1	0	Data	Read/Write, expand-down
7	0	1	1	1	Data	Read/Write, expand-down, accessed
		C	R	A		
8	1	0	0	0	Code	Execute-Only
9	1	0	0	1	Code	Execute-Only, accessed
10	1	0	1	0	Code	Execute/Read code segment

Paging

- Paging is a system which allows each process to see a full virtual address space without actually requiring the full amount of physical memory to be available or present

PTE (Page table entry)



Paging process

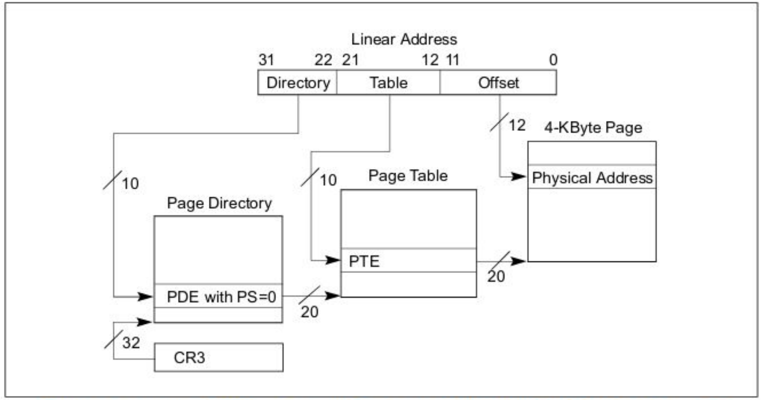
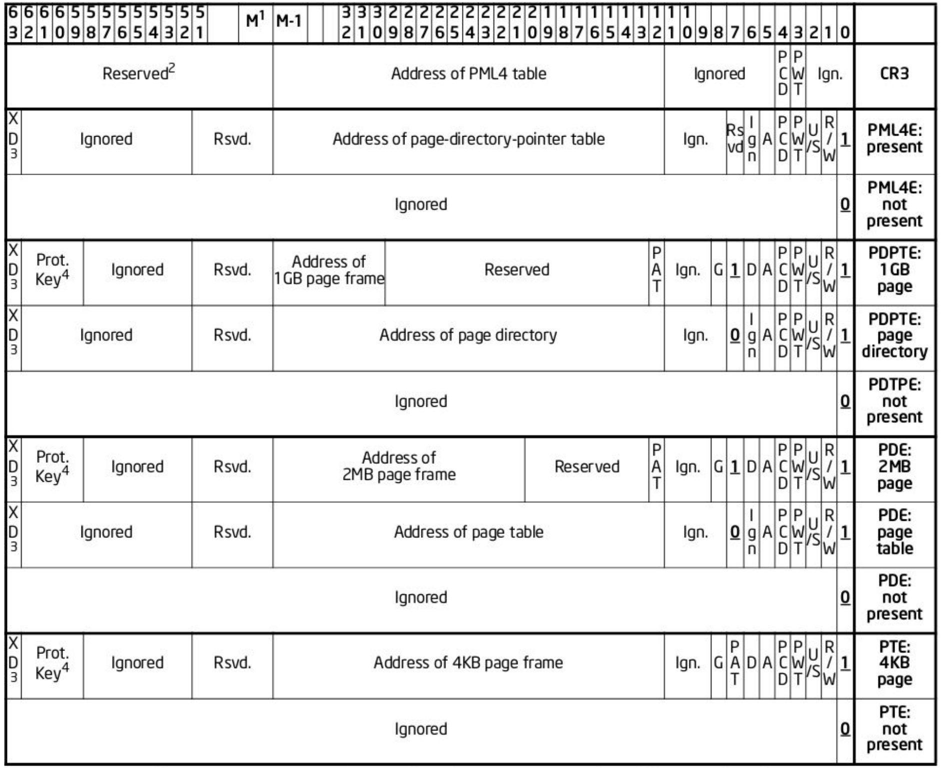


Figure 4-2. Linear-Address Translation to a 4-KByte Page using 32-Bit Paging

Big picture (Paging)



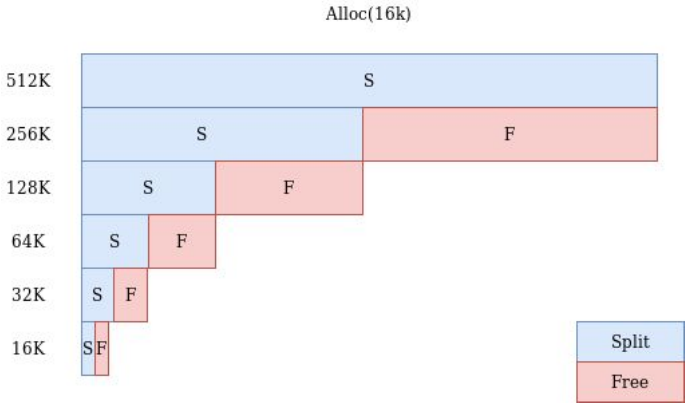
63	62 61 60 59	58 57 56 55 54 53 52	51 M[1]	M-1 32 31 30	29 28 27 26 25 24 23 22 21	20 19 18 17 16 15 14 13	12	11 10 9	8	7	6	5	4	3	2	1	0	
Reserved[2]				Address of PML4 table				Ignored					PCD	PWT	Ign.			**CR3**
XD3	Ignored		Rsvd.	Address of page-directory-pointer table				Ign.		Rsvd	Ign	A	PCD	PWT	U/S	R/W	1	**PML4E: present**
Ignored																	0	**PML4E: not present**
XD3	Prot. Key[4]	Ignored	Rsvd.	Address of 1GB page frame	Reserved		PAT	Ign.	G	1	D	A	PCD	PWT	U/S	R/W	1	**PDPTE: 1GB page**
XD3	Ignored		Rsvd.	Address of page directory				Ign.		0	Ign	A	PCD	PWT	U/S	R/W	1	**PDPTE: page directory**
Ignored																	0	**PDTPE: not present**
XD3	Prot. Key[4]	Ignored	Rsvd.	Address of 2MB page frame		Reserved	PAT	Ign.	G	1	D	A	PCD	PWT	U/S	R/W	1	**PDE: 2MB page**
XD3	Ignored		Rsvd.	Address of page table				Ign.		0	Ign	A	PCD	PWT	U/S	R/W	1	**PDE: page table**
Ignored																	0	**PDE: not present**
XD3	Prot. Key[4]	Ignored	Rsvd.	Address of 4KB page frame				Ign.	G	PAT	D	A	PCD	PWT	U/S	R/W	1	**PTE: 4KB page**
Ignored																	0	**PTE: not present**

Figure 4-11. Formats of CR3 and Paging-Structure Entries with 4-Level Paging

Kernel Physical allocators

- Buddy allocator
  - The buddy allocator works by repeatedly splitting memory blocks in half to create two smaller "buddies" until we get a block of the desired size.

Logically subdivide memory block power-of-two blocks

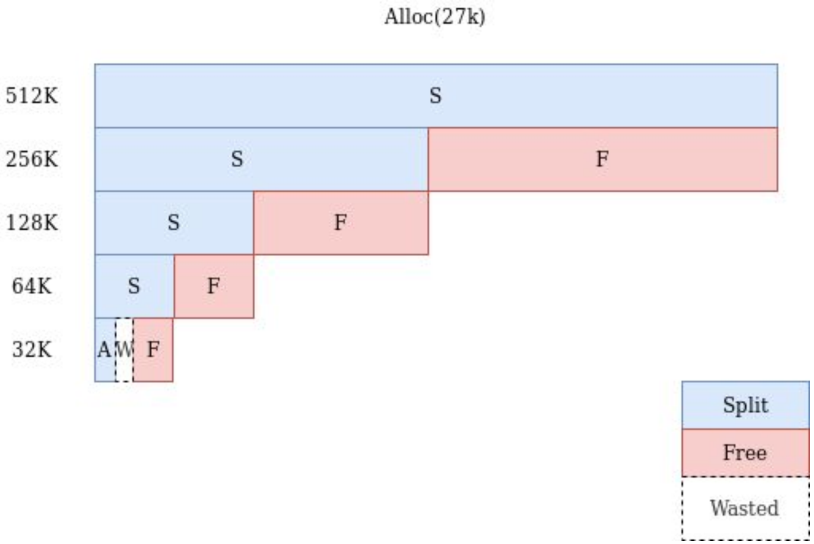


If we need 16K, we split the 64K block into two 32K and then split one of those into 32 K.

Internal Fragmentation in Memory allocators

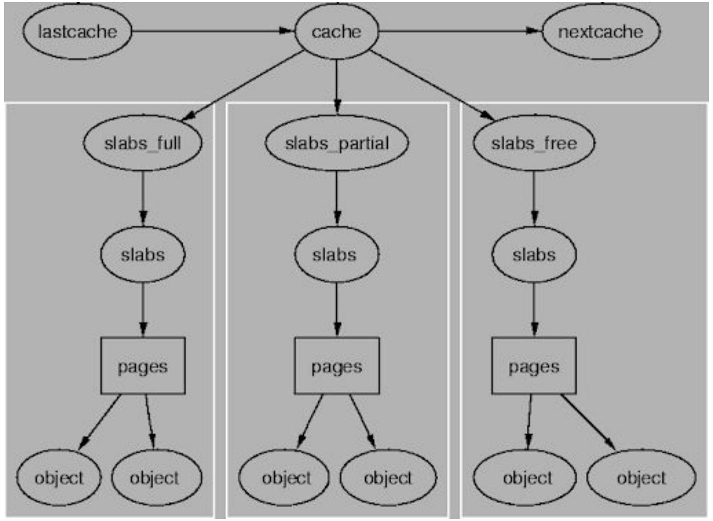
- Internal fragmentation will cause wasted memory inside block

27K is allocated 5K is wasted



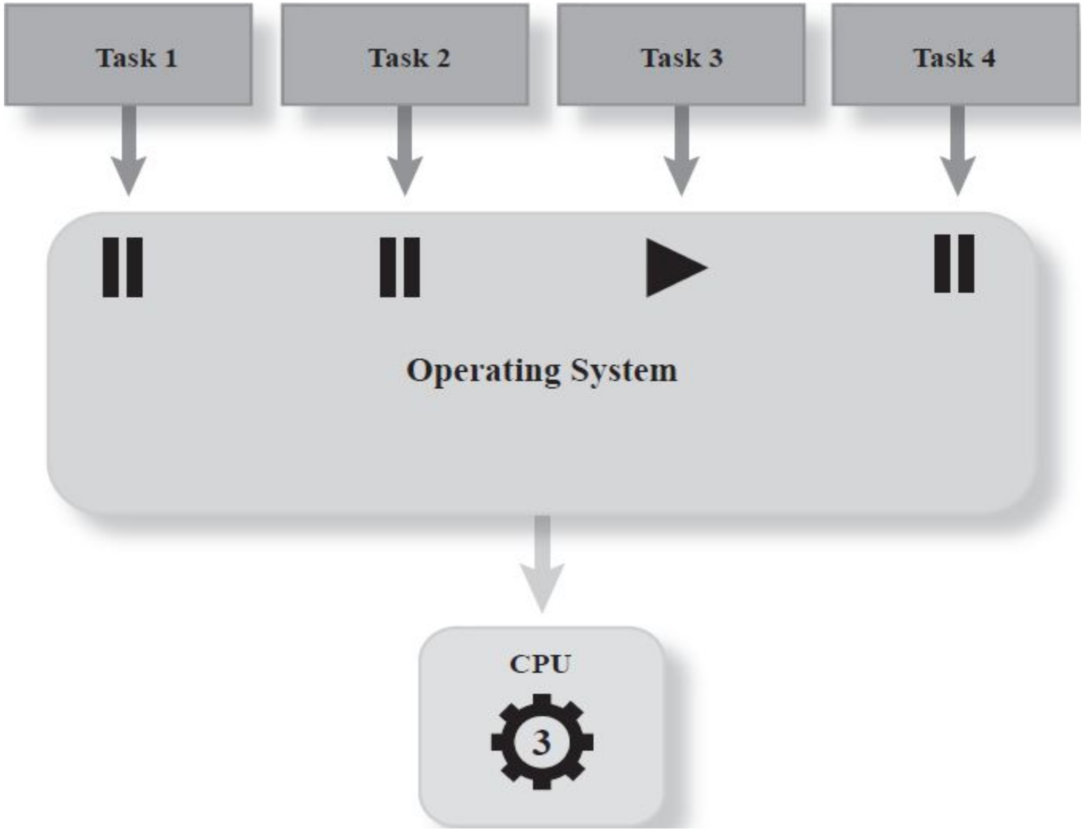
Kernel Slab allocator

- It used to reduce fragmentation. The technique is used to retain allocated memory that contains a data object of a certain type for reuse upon subsequent allocations of objects of the same type



Multitasking/Scheduling

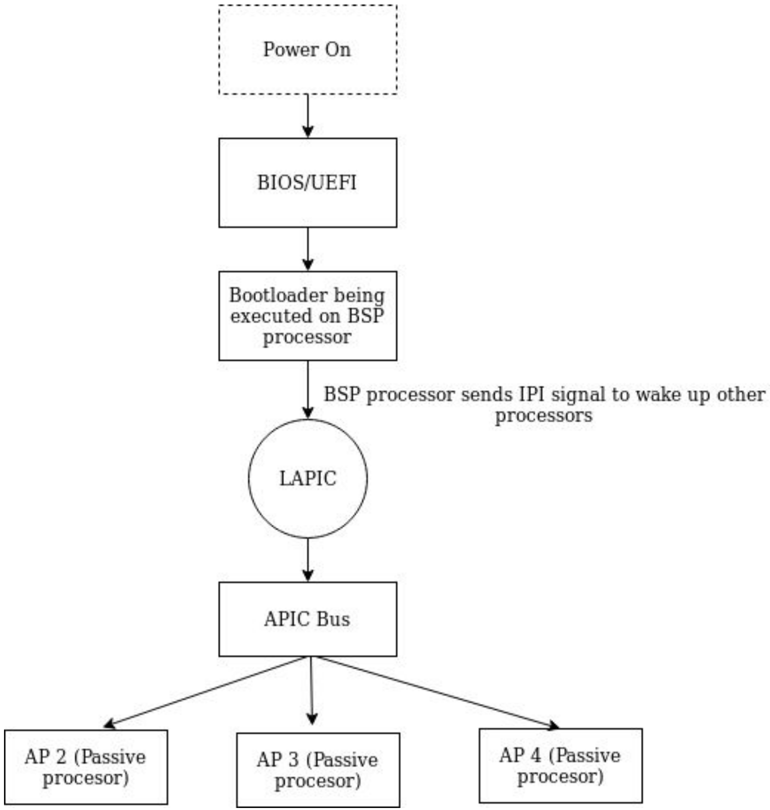
Multitasking - Preemption



Multitasking in the wild

- Task scheduling
  - An interrupt is scheduled with timer to allow the operating system kernel to switch between processes when their time slices expire
  - Task scheduler will set rate to fire interrupt by Timer controller

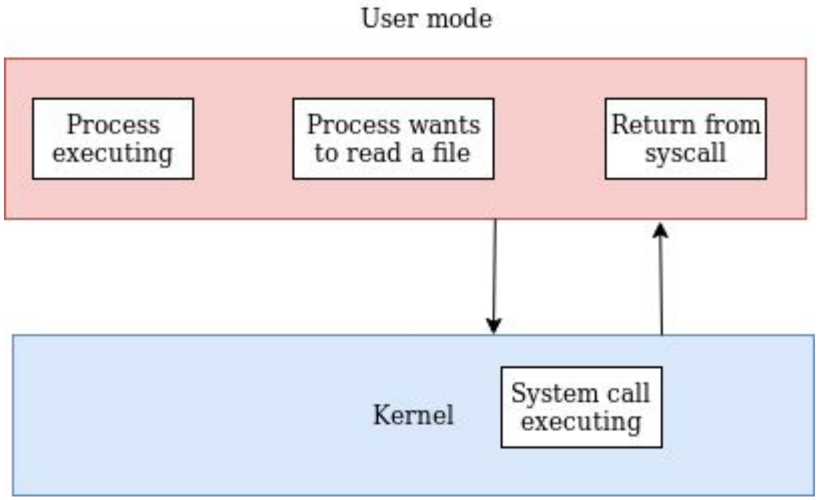
Multiprocessor



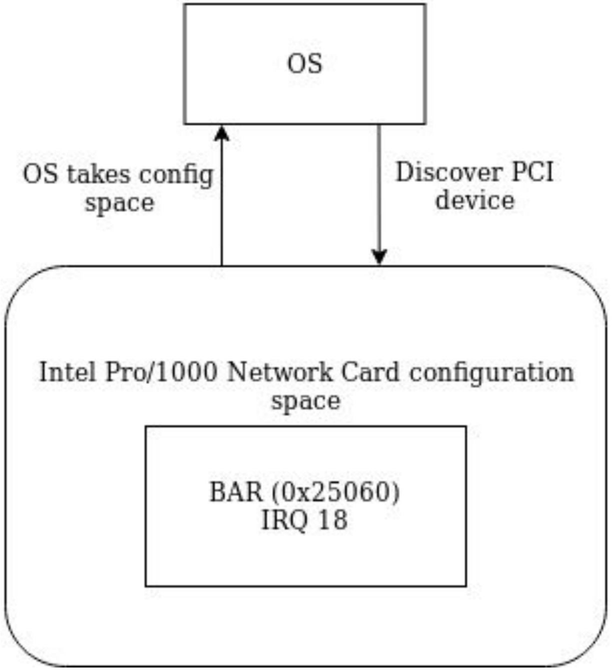
Power On
BIOS/UEFI
Bootloader being executed on BSP processor
BSP processor sends IPI signal to wake up other processors
LAPIC
APIC Bus
AP 2 (Passive procesor)
AP 3 (Passive procesor)
AP 4 (Passive procesor)

System calls

- Syscalls
  - is the programmatic way in which a process requests a service from the kernel of the operating system on which it is executed.



Device communication with OS



OS

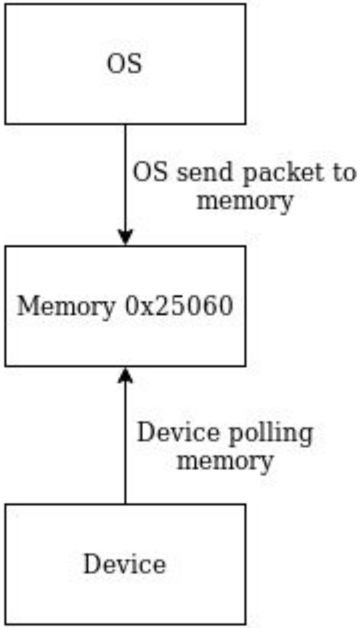
OS takes config space

Discover PCI device

Intel Pro/1000 Network Card configuration space

BAR (0x25060)
IRQ 18

Next time



OS

OS send packet to memory

Memory 0x25060

Device polling memory

Device

Q&A Discussion

https://bit.ly/3akb2Iq