Mode Bit: 0 / 1 (kernel / user)
DMA Direct Memory Access
Does direct addressing of memory imply that the O.S is not involved?

An OS is a program that runs other programs.

Where is the source code?

- **Multprogramming** increases CPU utilization by ensuring the CPU always has something to execute
- Several jobs are kept in memory
- Memory Hierarchy (not all jobs can be 'in' memory at one time)
- Hence the use of a job pool!
- OS picks one of the jobs in memory to execute.
- When a job is waiting for an I/O Operation, OS switches to another job
- Security

If several jobs are kept in memory, does that mean all of them are running at the same time? No!

What security vulnerabilities should we prevent? If we have job 2 as Skype and job 4 as firefox, skype shouldn't be able to access the memory that firefox uses.

The OS handles where application data is placed when being moved in/out of memory, cache, etc, and where the evicted data needs to go.

Three ways that programs invoke system calls

- GUI: Graphical User Interface
- batch: scripting
- Command Line

OS thus needs to provide:

Program Execution: Load program into memory and run item

I/O operations: Send data to screen, speaker, to/from file, etc.

File systems: Create, delete, add to file, etc.

How would one process communicate with another?

- Shared Memory
- Message passing (semaphores)

These services do not necessarily guarantee fair use of resources.

- **Resource Allocation**
  - Scheduling algorithms
- **Accounting**
  - Which user/process uses how much of a specific resource
  - VM is *basically* accounting
  - Programs need time, space, and fair access.
- **Protection and Security**
  - Make sure data from one process/user is not accessible to another process/user

The command interpreter has no knowledge of what a command does, but instead launches a system program

Located in `usr/include/x86_64-linux-gnu/sys`

How are parameters passed to the OS?

1. Registers
2. Save parameters in block or table (memory) and pass via a register