Fundamentals of System Hardware

Computer meaning :the computer is an electromechanical device which takes input does processing and produces output.

Types of Computers

Desktop,Laptop,Tablet et..

Inside A computer

Power supply: Provides clean conversion from line voltage to 12V and 5 V need inside the machine

Tertiary Storage: Cd-rom,DVD,Blue-Ray or even tape.This is the offline storage system

Secondary Storage:Hard driver or solid state this ithe permanent storage system of computer

Main memory(RAM):This is where code and data are stored when the coputer is shutdown this is lost.

Video Card/CPU:Stores information to display on the screen can do complex calculations related to decimals numbers.

Mainboard/Motherboard:Provides physical connectivity for all the devices included the system bus and all peripheral busses.If the CPU is brain this is the circulatory system.

CPU=Brain

All computers have

1-CPU,Central Processing unit which is brain of computer

2- Main memory (Ram)

Will have*

GPU, A video graphic controller where images can be rendered for display on a screen.

A network interface for communications.

Peripheral interfaces (USB,Thunderbolt,Firewiere,SCSI)

Communication between the devices

Internal communications in a machine is done by via ‘bus’

A bus is a physical pathway for communication between two or more devices

The system bus is the main pathway between the CPU and main memory but also carries data to and from input and output(IO) devices

Buses goes to CPU and Main memory RAM.

Main idea is communication and delivering datas.

THE CPU

The cpu is the brain of the computer

Its a single piece of silicion in the form of a chip

This is the only location where code is actually executed in the system

THE CPU ONLY RUNS MACHINE LANGUAGE CODE

The cpu operates on a ‘fetch-decode-execute’ cycle

Each type of CPU has its own set of instructions which it understands/

INTEL, can modifiy of these instructions.. FOR EXAMPLE

Each cpu has a small amount of memory ‘call registers’ which it uses to perform operations and store results.

A cpu may have a ‘cache’ memory to perform more quickly

MACHINE LANGUAGE

CPU understand like move add,substract,multiply,jump etc//

The designer of the cpu puts the capability to perform these operations in the physical chip

Instruction Set

The designers of the cpu create a set of instructions that cpu can perform

This is set of instructions usually as small as 100,can be represented by a numeric value

When cpu receives a particular instruction,It perform that task

Ex: Instruction OPCODE

ADD OxOO

Python ,does not execute your code.Python uses your code as a guide to executing instruction on the system

C++ takes your code and literally convert It in a process we call compilation to machine level code

Instruction set is like a contract between designer and machine

Fetch-Execute-Cycle

The processor does not have enough storage to keep an entire program we have a whoel long list of instructions that need to be executed in order to robot moves to in particular position the robot might not have all of those instructions for how to get here to the the corner

In cpu we can only store one or two instructions

The cpu performs a fetch to move the instruction from main memory inti the cpu specifically into an instruction register

It then decodes the instruction also moving it any additional data that might be necessary with that instruction

Then executes that instruction

This process repeats with the next instruction in the sequence

Meaning the cpu can process millions of instructions per second

Memory

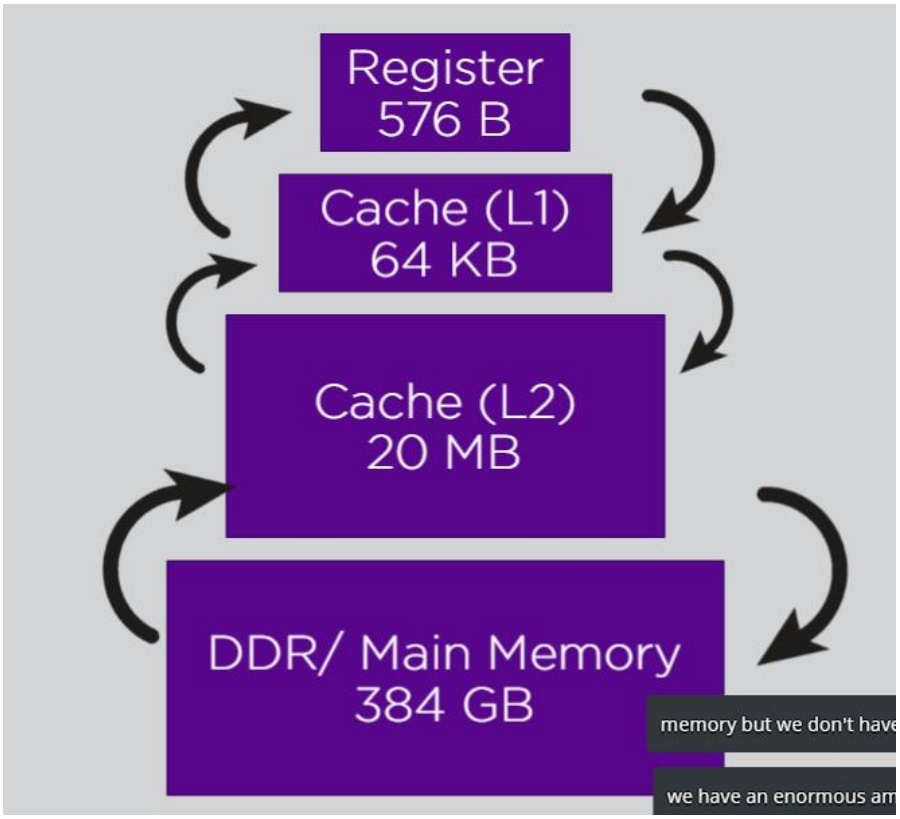
The instruction and all data has to come from somewhere

In order for code to be executed,it has to be in a register built into the cpu

Why not just store everything registers. It′ s expensive

IDEA:Main memory can be a short term storage solution which can feed information continuously to the CPU In the form of instructions It can also storage things much more longer than registers .Main memory is slower than registers but we have more of it

The hierarchy



Registers are small portions.they are fast.

RAM

Random Access Memory

We can access any place in the same amount of the time.

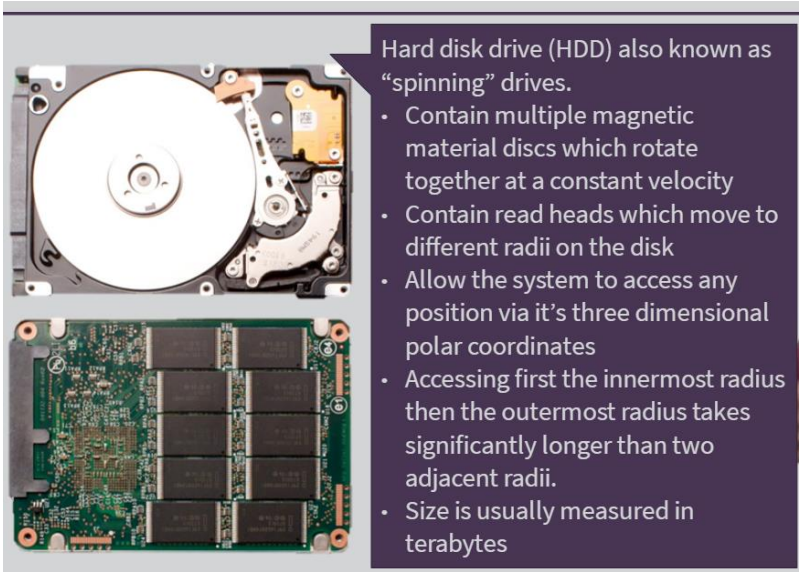
Each individual bytes has addresses.

Integer Is 4 bytes.

When you turn off your computer,everything in ram is lost

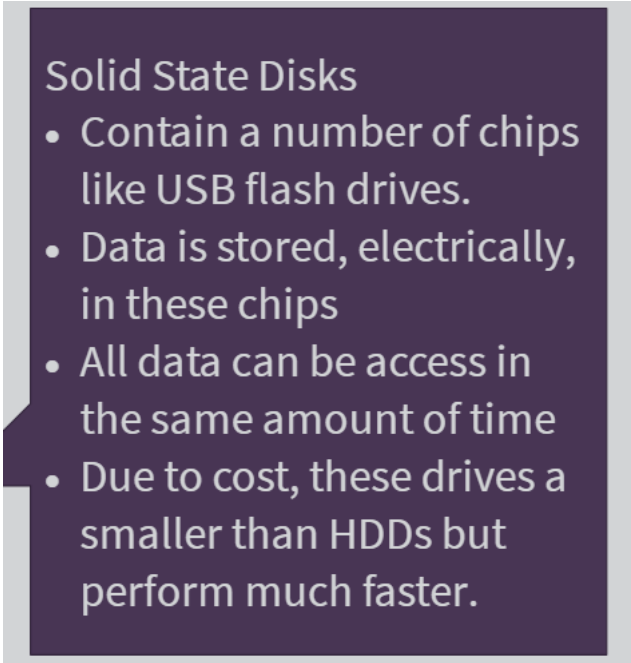
When you run a program all the machine language instructions are brought into RAM and one by one.Pulled into the CPU by the fetch and execute cycle

Secondary Stage



*HDD=Much capacity but slower

Google stores their data on HDD because they think capacity is important than speed for example



They are read only

SSD is accessible in random time

Reading is faster than HDD

It costs more than Hdd**

For COST THEY ARE EXPENSIVE THAN HDD

Introduction to Operating Systems

Operation system: a program that controls execution of application programs and acts as interface between applications and computer hardware.

Software which manages the system

Runs on the same processor as the user′ s program code.

Does not include applications

Layers of interaction

Users use applicatons to interact with operating system

The Kernel

The core component of the OS

Responsible for managing system resources

Assist applicatons with performing work

Nothing runs without the kernel*

Fundamentally,the kernel is the operating system

What is in the kernel?

The OS as a Resource Manager

Every application you open needs memory (RAM)

The government example

Government is resource manager

Governments gets resources,tax money

Governments spends tax money on services like fire fighters

Government needs buildings to operate and people to run it

They all cost money

Government takes some of the resources (money) to pay for its operating costs

The money spent on running the government is considerednecessary waste to the people

OLDEN DAYS

The computer ran only one program at a time.

The one program had complete access to all the system resources

The OS was only responsible for getting programgs ready to run

Requires JCL: Job control language

The mainframe operator (a human) decided which order to run programs in

When one program finished the os was ready with the next wanted to run

This is called batch multiprogramming

TODAY

Today we have lots of processing power and lots of resource enough to run multiple programs at the same time

We divide memory to run multiple programs

THE OS becomes resources

The OS decides which programs how much they get memory

THE OS manages allocation of resources

The Os decides which programs can run and when

The os will stop and restart running programs

This is called time sharing system

Monitoring running programs

Keep track off all running programs in the system in order to manage

Resource allocation

Scheduling

Authorization

The same program could be loaded multiple times*

So we need a way to track multiple instances of the same program running on a system

A PROCESS: A program in a running state

Loaded into main memory

Scheduled

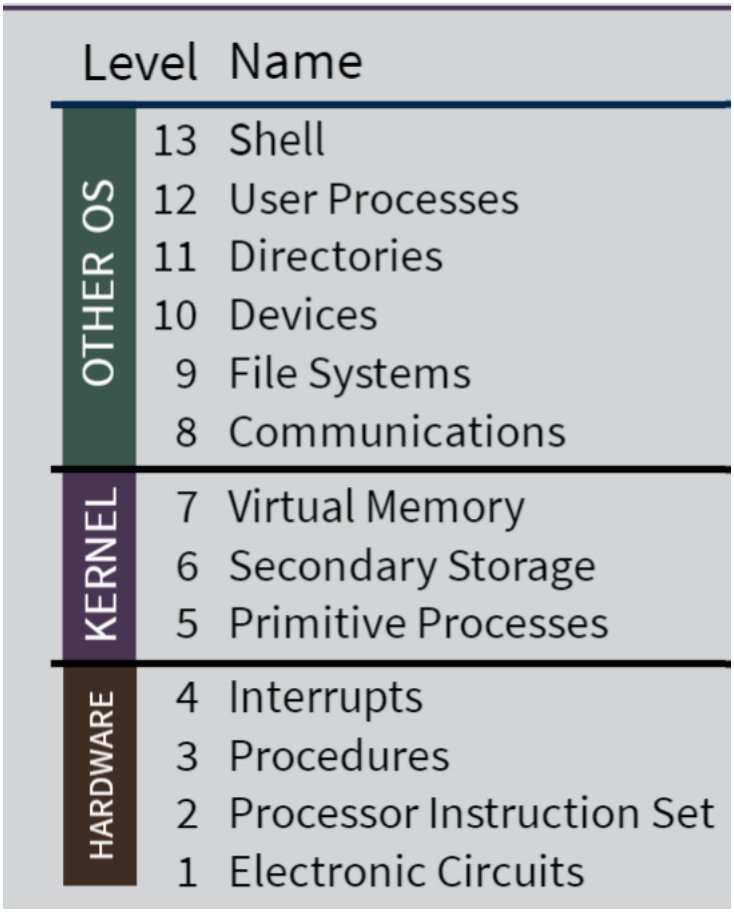
A PROCESS has:

Access to files

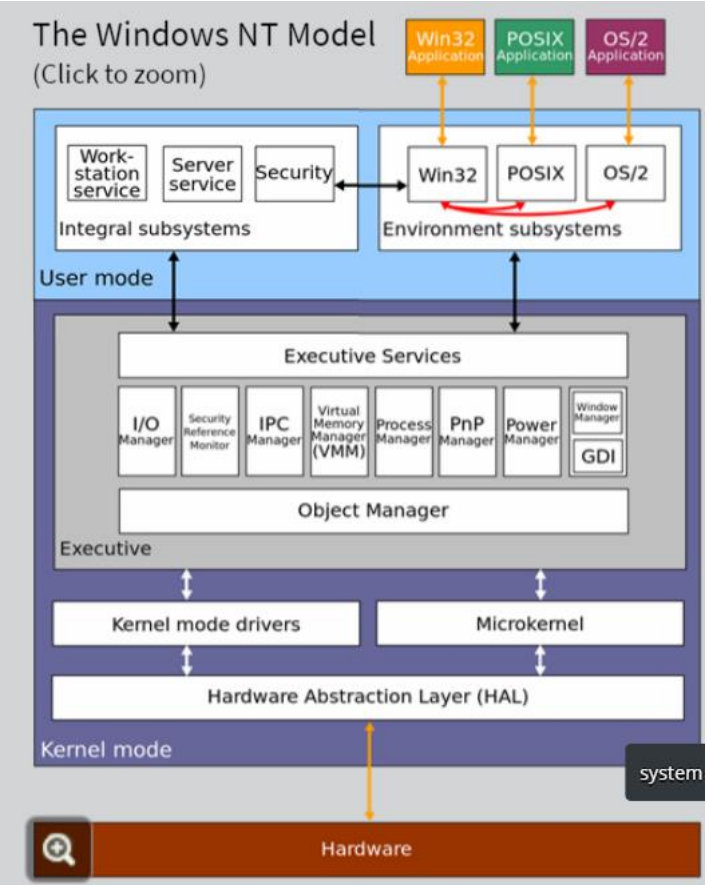
Access to networking connections

Code,the code is what′ s run.

OS LEVELS



The Windows Model



User applciations like Mozilla etc,they do onot have direct access to the KERNEL.

THE HAL

The hal is a layer which can provide the kernel with a set of functions to call which program the hardware properly.

All computer have Timer,System bus,Memory Access Routine

Changes to hardware only require changing the hal,not rewriting the whole OS

Windows Device Drivers

Devices drivers are karnel layer software written by companies that design hardware

They provide functions for the kernel to call in order to access the hardware

Poor softwares could be blue screens

They were the cause of frequent blue screening before windows hardware Quality labs

UNIX

A multi user,multi tasking OS

Designed to allow users to manage their own tasks

They upload operating system.They also give the source code

You can play with mainframe.

Released into public domain as open source software

Comes in many different flavors: AIX,Linux,Solaris

There are other distributions

BSD(Berkeley Software Distribution)

Processes and Threads Lecture 3

Process:A running program in a system state

Includes code,data,context

Is stored in sequential memory space

*process is created by OS to keep track of

State of the running program

Resources assigned to the running program.

State: A condition that the process will spend a significant amount of time in

Suspension:There may be process that don’t have to load main memory

The process is completely removed from main memory

Process is stored on secondary storage for future return to the point we left off

Frees main memory for other process

Controlled by medium-term scheduling algorithm

The process will not be aware of the suspension

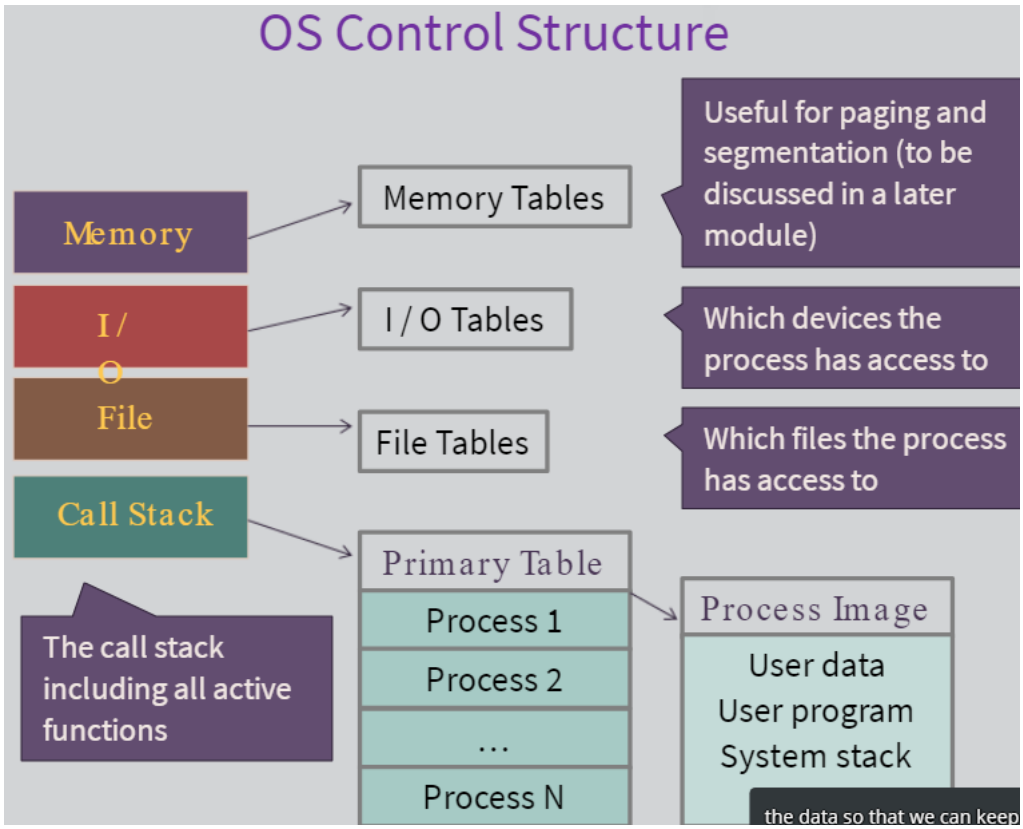
Some reasons for suspension

- Debugging
- Long term delay
- Freeing main memory

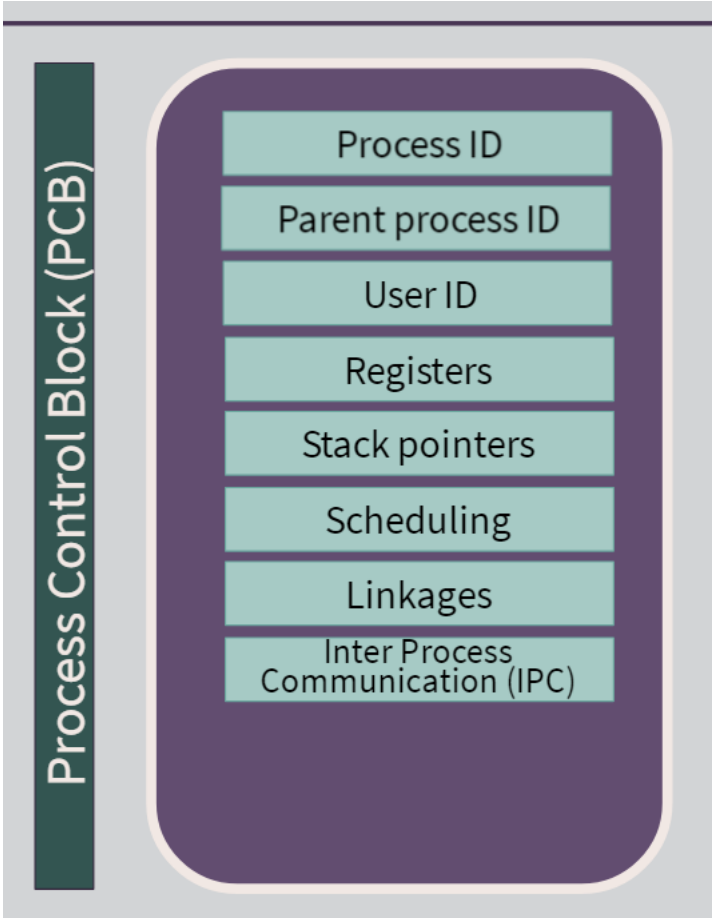
Process Image-THE PCB

Process control block

Includes all the information the OS needs to run and control process



Contents of a PCB



Modes: Operation systems restricts Kernel mode.

Kernel mode:code can execute any part of the system

User mode:cannot directly access system hardware cannot run some cpu instructions cannot access any memory outside of its own

Switching:

Easy to go from kernel to user

User to kernel happens automatically upon certain events

When to process Switch

Interrupt: a hardware signal indicating that the hardware need servicing

TRAP:A condition which requires OS support

Blocking system call: A request from the process for OS support

A process change it’s situation for ex running to waiting situation

Multiple Processors:multiprocessing means these problems become more complex and happen more often

Two processors can run at same time*

Threads: Resource ownership and execution are two different issues

Resource ownership now bevomes the only concern of the process

Execution;scheduling and running parts yes there will be many of the process become threads

Multithreaded environnements

Process-Process control block

Memory allocation

Files

Linkages

Thread-Thread Control Block

Context(processor registers)

Stack(inc local variables)

Access to all of the resources of the thread

Execution of code is responsibility of Thread

Reasons for multithreading

Today all of applications are multithreading

When u use word, there are many threats u use.

Thread Concurrency and Deadlocks

Threads all share the resources of the process

Threads run as if they were separate program

Threads can run asynchronously

MEMORY MANAGEMENT

Reasons for memory management

In multiprogramming system,there will be never enough main memory

The Os will need to allocate memory to multiple programs in the sthe system

The Os will need to move parts between main memory and secondary memory

Memory managements Requirements

Relocation:A process could be loaded into any location in main memory

A process can be moved during its lifetime

Protection

A process should not be allowed to access memory outside of its allocation

The OS cannot pre screen the memory accesses,it must be done dynamically

Sharing

Some processes might use the same code,data

Logical organization

Modules,shared objects or dynamically linked libraries should be allowed

Logical vs Physical Addresses

Due to primarily to relocation a program will have to use a logical address to access memory

Logical addresses need to be converted dynamically to physical addresses

A simple solution to logical addresses could be the offset from the beginning of the program

The cpu′ s hardware memory management unit is responsible for converting during runtime the logical address to a physical address

Partitioning Strategies

Fixed partitioning

Dynamic partitioning

Buddy System

Simple paging

Simple segmentation

Fixed partitioning: main memory is divided into a static number of parts

A process will be allocated one part of equal or greater size than it needs

Equal sized partitioning

Causes internal fragmentation:wasted space inside the partition

Unequal sized partitioning

Needs to worry about which partition to place the process in

Dynamic Partitioning

Partitions are created for exactly as much memory as the process needs

Causes external fragmentation-wasted memory between allocations

Requires compaction,a costly process

OS data structures are complex because a process could start anywhere

Where does the OS place a partition

Best fit- the area with the size closest(causes external fragmentation)

FirstFit-Just choose the first spot in memory large enough

Nextfit-begin looking from the location of last allocation

Buddy System

A compromise between fixed and dynamic

Memory is divided and divided again in multiples of 2

2 MB of memory might be divided as such,if we need a100 kb allocation

The process would be given the first 128 KB space

If we have a large number of 100 kb processes we would further divided the larger partitions to accommodate the need

This makes OS structures easier because processes begin and end on a 2 n boundary

Paging

Memory is broken down into all equal sized(say 4K frames)

A process is broken town into same sized (4K) pages

Since this is ram,no frame is better suited for the task of storing the page than any other all have O(1) access time

Pages are loaded into non contiguous frames

The os records the frame number for each page in a page map trable stored in PCB

The hardware MMU needs to be able to query the PMT so the format as to be agreed on.

Benefits of Paging

No external fragmentation

A minimum of internal fragmentation

Easy protection

Easy relocation

Easy sharing

Converting logical to Physical

The logical address is relative to the start of the process

To calculate the page number,we do bit shifts because pages are always multiples of 2

To calculate the offset into the page,we can use WOR

The page number can be use an index (like in array) into the PMT to find the frame number in main memory

The cpu will then bit shift that frame number and add the offset to find the physical address

Segmentation

Each process is divided into unequal sized partitions

Logical address now needs to be segment offset

Storage is similar to Dynamic partitioning

Virtual Memory

Assumes paging or segmentation

Since the process will only ever access one memory location at a given time the rest of the memory of the process doesn′ t need tot be in main memory

We can swap pages of the process out onto the hardddrive and bring them back later if needed

Example the splash screen at startup of a process ient ever need again after the process is running

Benefits OF VM

The programmer perceives a much larger memory space

The system can remove unused pages

More processes can run in the system resulting in better performance

Lookup Problems

Now any memory access by the process requires that the MMU determine the physical address which requires a query to the PMT

Each memory access requires two memory access

To save time the CPU implements a translation LOOkaside Buffer which is a type of content addressable memory which stores a cache of those entries in the PMT which have been retrieved recently

Replacement Policy

When main memory runs out,The OS needs to remove some pages,this is called stealing

If we choose poorly the result will be worse performance

The optimal choice would be steal the page which is not going to be used for the longest period of time

Possible algorithms

Optinal not feasible

LRU Not feasible

FIFO Easy to implement but poor performance

Clock Easy to implement but requires use bits

Residents set managements

Two problems

How much main memory each process deserve

Smaller allocation means more processes

When its time to steal which process do we steal from

Local-the process page faulting will lose a page

Global- Any process can lose a page

Controlling page faults by resident set size

If a process fits entirely in main memory,it will not generate any page faults

If a process only has one frame,it will generate the maximul number of page faults

PFF(Page fault Frequency)Algorithm

Examine the time between the last page fault and the current one

If less than an preset F value,add a frame

If greater than the preset f value,discard all pages with use bit 0 reset all use bits to 0 and continue

Upper bounds and lower bounds woulbe better than just F

Unfortunately it performs poorly during locality shifts