<https://compas.cs.stonybrook.edu/~nhonarmand/courses/fa17/cse306/schedule.html>

^ class almost exactly like ours, better slides

# Exam 1 Review Sheet

Table of Contents

[Exam 1 Review Sheet 1](#_Toc526980765)

[01-Intro 2](#_Toc526980766)

[02-Virtualization CPU 3](#_Toc526980767)

[03-Scheduling 4](#_Toc526980768)

[04-Virtualizing Memory 7](#_Toc526980769)

[05-VM Paging 10](#_Toc526980770)

[06-VM TLB 11](#_Toc526980771)

[07-VM smaller page tables 11](#_Toc526980772)

[08-VM beyond physical 11](#_Toc526980773)

[09-Threads 11](#_Toc526980774)

[Notes From Review Day 11](#_Toc526980775)

## 01-Intro

**Operating System:** interface between hardware and application \* Software that makes hardware useful for applications.   
  
**Operating System** provides:

* **Abstraction**
  + Makes different devices look the same
  + Higher level functionality
* Abstraction in what form???
  + CPU: Processes
  + Memory: Address spaces
  + Disk: Files
* Resource Management: Share resources well
  + Advantages:
    - Efficient Use of resource
    - Fair Use of resource
    - Protect one application from another
* Challenges: Policy & Mechanism

**Three Pieces**

* **Virtualization**: Make every process think it has the CPU to itself
* **Concurrency**: OS must handle simultaneous processes/events. Easier if they are independent. Trickier if they interact
* **Persistency**: information is accessed permanently. Provide abstraction so that the application doesn’t need to know where the data is stored. Performance and Handle Failures.

**Design Goals:**

* **Efficiency: (low level view)** A Hardware management library
* **Better Usability: (high level view)** Physical machine to an abstract one**.**
* **Protection**: Processes are separate from another

**Other Design Goals**

* **Reliability**
* **Security**
* **Mobility**

## 02-Virtualization CPU

Process is a program in execution.   
Program is **STATIC**. Process is **DYNAMIC.**

## 03-Scheduling

**Definitions and Acronyms**

Time slice – length of time process has the CPU

longer = worse response time & few context switches

shorter = better response time & more context switches

Interactive Jobs – need I/O operations and care about response time

Batch Jobs – only care about turnaround time

**Formulas**

Turnaround Time = Completion Time – Arrival Time

Response Time = Time of First Run – Arrival Time

**Concepts**

Sharing the CPU

Why?

How?

Must have a mechanism which is the dispatcher, and a policy which is the scheduler.

FIFO – first in first out, also called FCFS, first come first served

SJF – shortest job first

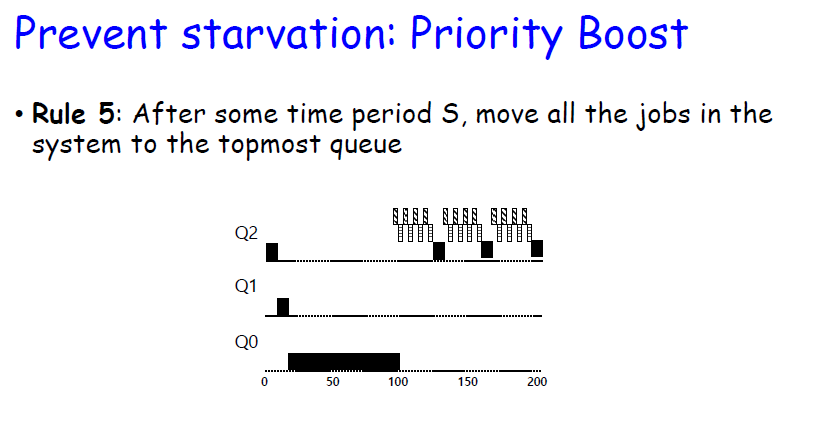
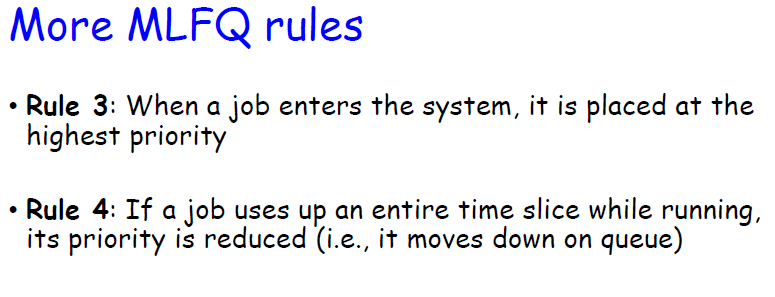
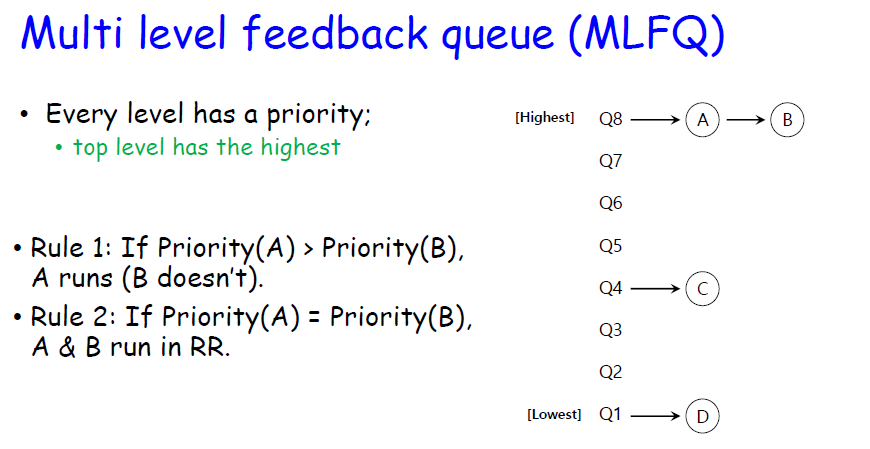
STCF – shortest time to completion first

RR – round robin

MLFQ – multi level feedback queue, top level has highest priority

Lottery - each process gets tickets, whoever wins lottery runs

|  |  |  |
| --- | --- | --- |
|  | pros | cons |
| FIFO | easy to implement | short jobs MIGHT have to wait a long time  bad response time |
| SJF | good turnaround of short jobs | short job that arrives AFTER a long job causes starvation  bad response time |
| STCF | good turnaround of differing arrival times | jobs that use I/O cause starvation  bad response time |
| RR | good response time  doesn't care about run time | does not have a mechanism to adjust to the process at hand |
| MLFQ | mimics SJF  good response time for interactive | many interactive jobs will starve the long jobs  It's possible game the scheduler by giving up CPU before time slice is done |
| Lottery | fair sharing  simple to implement |  |



## 04-Virtualizing Memory

https://compas.cs.stonybrook.edu/~nhonarmand/courses/fa17/cse306/slides/05-virtual\_memory.pdf

Really good power point that is exactly like our class ^

**Definitions and Acronyms**

Allocation: “to distribute”, sets limits for program where virtual memory is mapped to physical memory

MMU: Memory Management Unit, hardware implementation that maps a logical address to a physical address by ADDING the base register to the logical address

Segmentation: divide each process into separate boxes of memory needs, stack head data code, each can grow or shrink independently of the others

Fragmentation:

Dynamic Relocation: moving a currently executing process

**Formulas**

**Concepts**

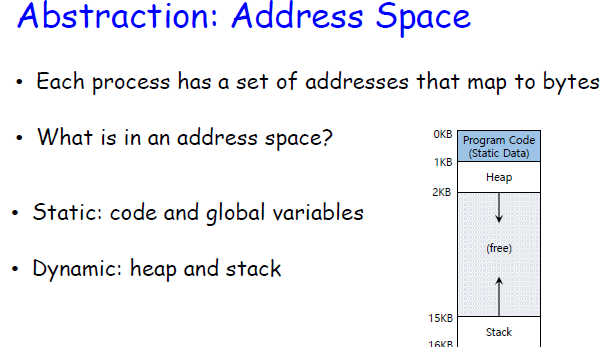
Virtualization

- only used when resource has to be shared

- creates isolation between entities sharing

- entities don’t know they are sharing, they believe they own the entire resource

Abstraction of Memory



What goes in the program code? Main() and all other programs

What goes in heap? Pointers to objects

What goes in (Static Data)? Variables for programs, like counter variable x for program Main()

Sharing Memory

Dynamic Relocation - move a currently running process to a new location of physical memory

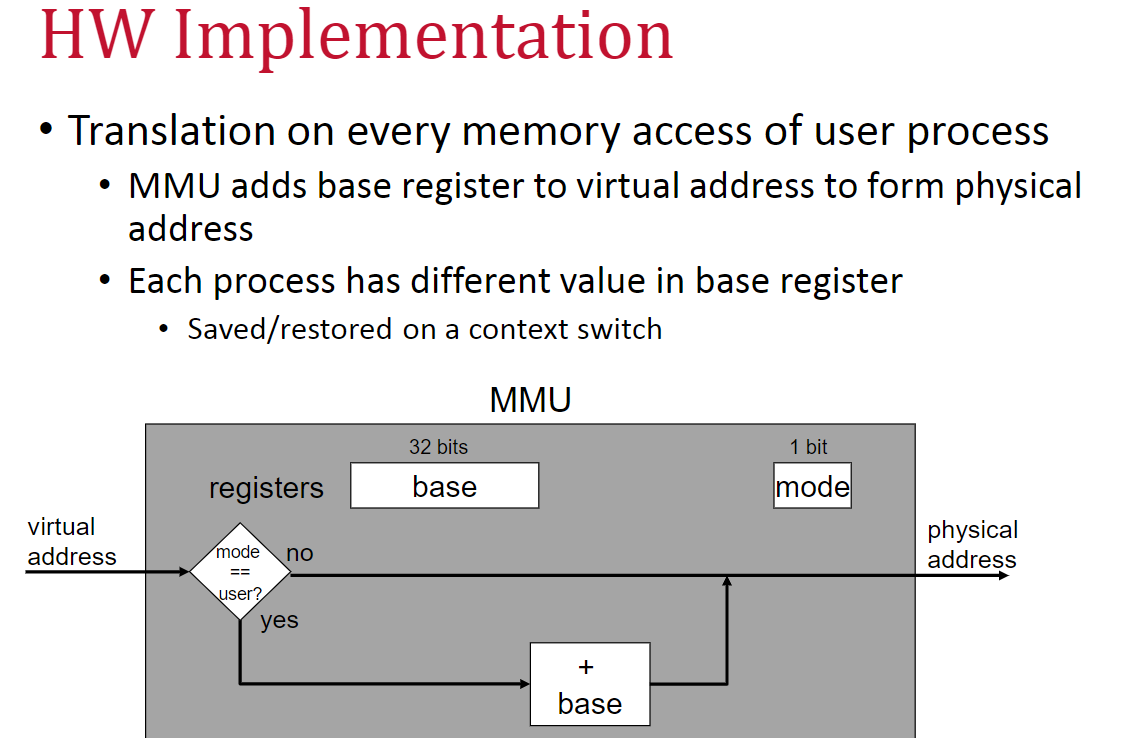
Does every process have its own unique value in its base register? Yes

Who modifies the base register for the process? OS

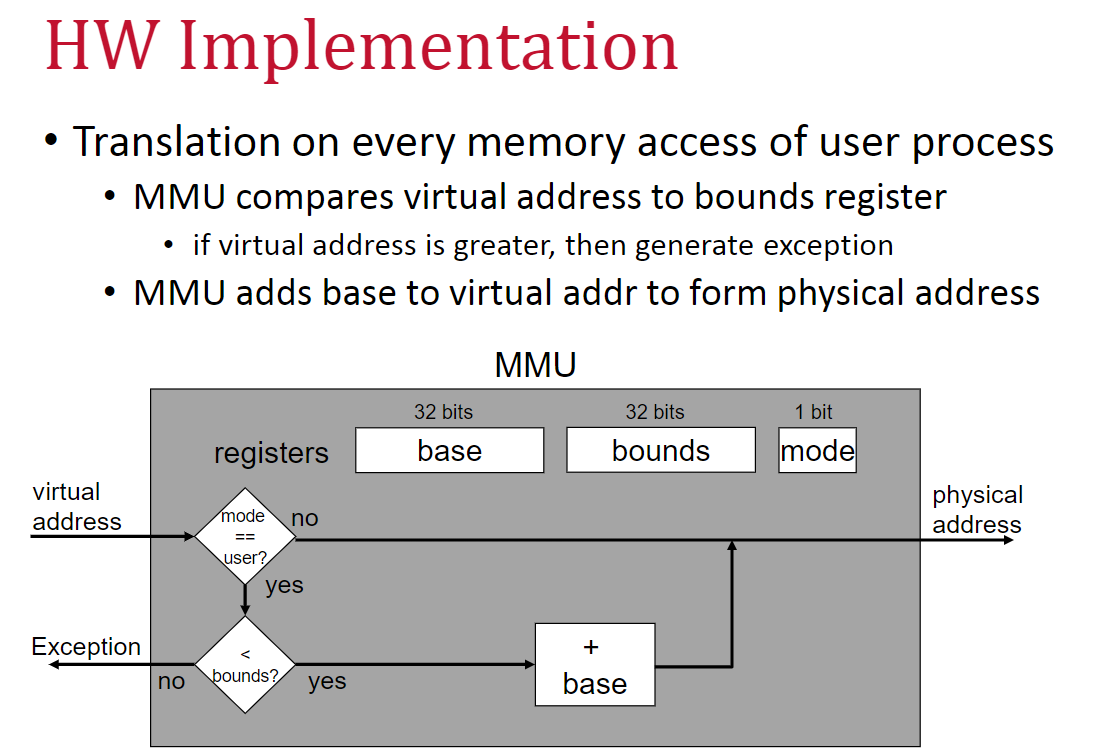
Who decides when to relocate? OS

Who generates the virtual address? The process itself

Who maps the virtual address to physical? Hardware, with the MMU



Add the bounds to that shit son!!



Isolating Memory

## 05-VM Paging

**Definitions and Acronyms:**

Internal Fragmentation: Process does not use all the space allocated to it.

External Fragmentation: There is adequate total space for a process BUT it is scattered in chucks throughout the virtual address space.

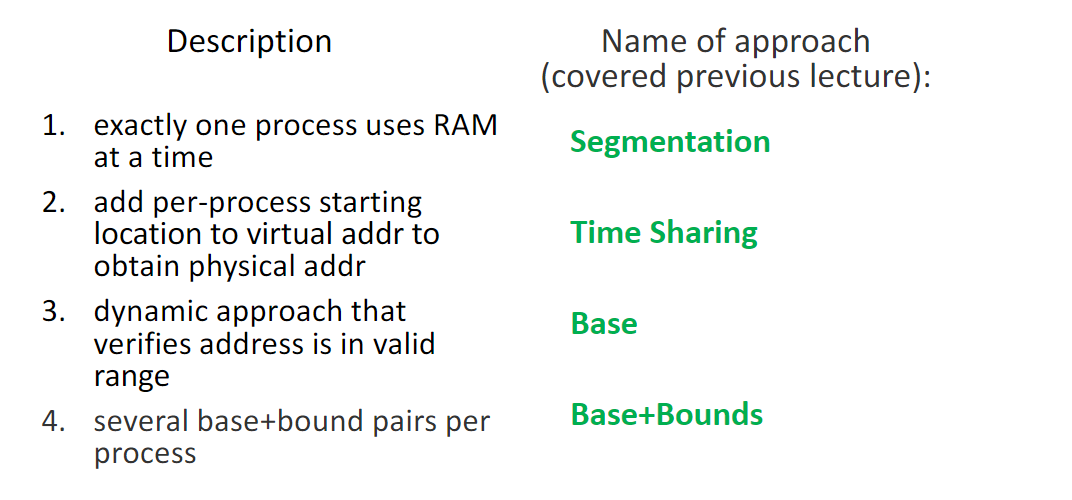
Paging: memory contains commonly accessed values, loaded for use immediately

**Formulas**

* **Segmentation Address Translation:**  *Physical Address = Offset + Base*

**Concepts**

* **Context Switching with base-and-bounds**
  + Base and Bounds are a part of the process’ context
  + Store the base-and-bound to old process control block (PCB)
  + Restore them from the new process’ PCB.
* **Base and Bounds**
  + **Advantages:** Easy to implement with Hardware. Dynamic Relocation. Protection
  + **Disadvantages:** Processes require contiguous memory, Internal Fragmentation, External Fragmentation.
* **Solution to Base and Bounds: SEGMENTATION** 
  + Address space for a process is divided into *segments*.
  + Every segment contains an element of address space such as heap, stack, code, data.
    - Segments are variable in size.
    - Independently placed in memory.
    - Each segment is protected
  + Advantages: Segment Sharing. Easier to relocate a segment versus the entire process/program.
  + Disadvantages: Fragmentation isn’t solved. Segment’s variable size can cause them to be large.



1. ---> **Time Sharing**
2. ---> **Base**
3. ---> **Base+Bounds**
4. ---> **Segmentation**

## 06-VM TLB

**Definitions and Acronyms**

**Formulas**

**Concepts**

## 07-VM smaller page tables

**Definitions and Acronyms**

**Formulas**

**Concepts**

## 08-VM beyond physical

**Definitions and Acronyms**

**Formulas**

**Concepts**

## 09-Threads

**Definitions and Acronyms**

**Formulas**

**Concepts**

## Notes From Review Day

Exam Review

Big picture

Virtualization

CPU

Memory

How fork works

Process creation

process deletion

waiting for child

Schedulers

How different one’s work

Comparison chart

turnaround

response time

calculation times

Paging

how paging work

translating virtual page to actual page

relationships between variables

if page size increases, what happens to page table

A modern OS virtualizes CPU by time sharing

A process is a program in execution

A thread is

You must change from user mode to kernel mode to execute syscall

More context switches, more overhead

Virtual page is an imaginary copy of the physical frame

Offset formula

Know what threads share and have of their own

Lexicographic - alphabetical

5 States in proc.h

EMBRYO, READY, RUNNING, BLOCKED, ZOMBIE

Know how to draw Gantt charts

Know definitions for scheduling

Know formulas for scheduling

Part 1

T/F

Part 2

program output

Part 3

virtual memory

Part 4

paging