

CST 347 ?? Real Time OS

# CST 347 – Real Time OS

Oregon TECH

## Lecture 17 – Memory Management I

Troy Scevers



## Introduction

# Introduction

STORAGE → CPU  
CANNOT EXEC FROM IT.

- Program must be brought (from disk drive) into memory and placed within a process for it to be run
  - Embedded world could include flash drive
- Main memory and registers are only storage CPU can access directly
- Memory unit only sees a stream of address + read requests, or address/data + write requests

SEQUENCES

## Hardware Issues

# Hardware Issues *e.g. PIC32*

- Register access in one CPU clock (or less)
- Main <sup>USP815</sup>memory can take many cycles
  - Memory technology impacts (SRAM/DRAM/Flash)

10:33:46 AM

## Hardware Issues

# Hardware Issues

110 2 13  
as n.s.  
4 clocks / instr.

- Register access in one CPU clock (or less)
- Main memory can take many cycles
  - Memory technology impacts (SRAM/DRAM/Flash)

Address	Name	Hex	Decimal	Binary
1F08_4010	CHRECC	0x00000000	0	00000000 00000000 00000000
1F08_4000	CHECON	0x00000000	0	00000000 00000000 00000000
1F08_4090	CHREIT	0x00000000	0	00000000 00000000 00000000
1F08_4080	CHRELO	[CHECON] 16 0 4		
1F08_4080	CHREIS	CHECON - DCSL - PRETEN - PFMWS		
1F08_4030	CHRESK	0 - 00 - 11 - 010		
1F08_40C0	CHREFFAS			
1F08_4020	CHRETAG	0x00000000	0	00000000 00000000 00000000
1F08_4040	CHREW0	0x00000000	0	00000000 00000000 00000000
1F08_4050	CHREW1	0x00000000	0	00000000 00000000 00000000
1F08_4060	CHREW2	0x00000000	0	00000000 00000000 00000000
1F08_4070	CHREW3	0x00000000	0	00000000 00000000 00000000
1F08_4000	CHRECON	0x00000000	0	00000000 00000000 00000000

## Register 4-1: CHECON: Cache Control Register (Continued)

bit 2-0 PFMWS&lt;2:0&gt;: PFM Access Time Defined in terms of SYSCLK Wait states bits

111 = Seven Wait states

110 = Six Wait states

101 = Five Wait state

100 = Four Wait states

011 = Three Wait states

010 = Two Wait states

001 = One Wait state

000 = Zero Wait states

CPU Clock period @ 80 Mhz:  
37.5 ns versus 12.5 ns

## Architectural/Hardware Issues

## Architectural/Hardware Issues

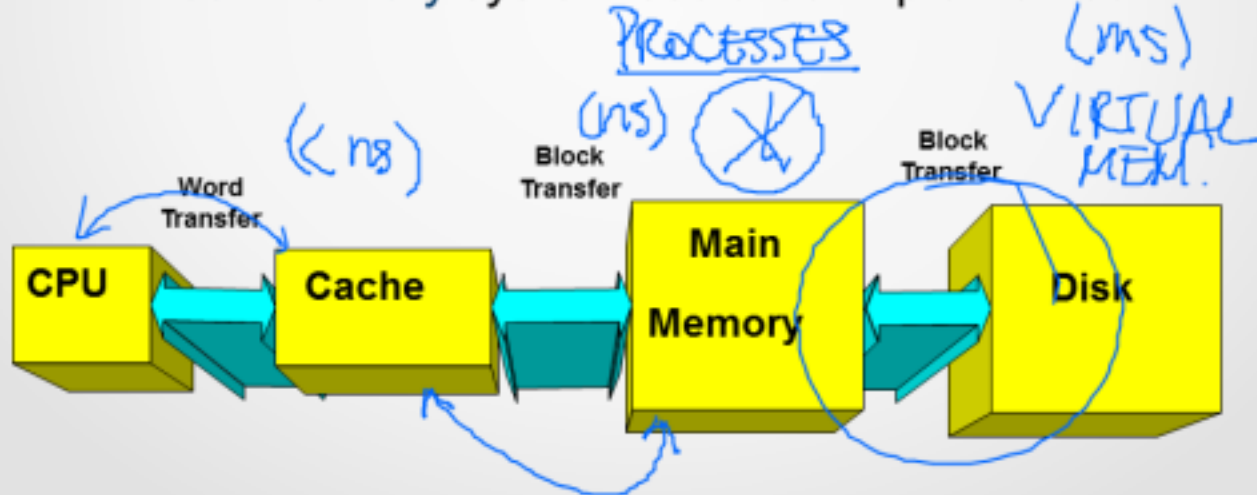
- Architectural effects are encountered
  - Cache sits between main memory and CPU/ registers
  - Virtual memory system could be implemented



## Architectural/Hardware Issues

# Architectural/Hardware Issues

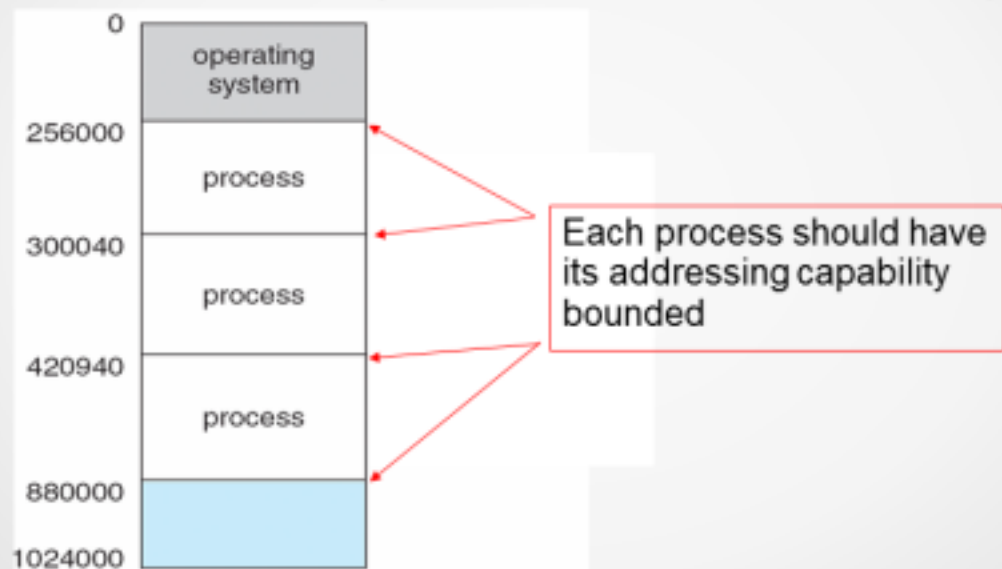
- Architectural effects are encountered
  - Cache sits between main memory and CPU/ registers
  - Virtual memory system could be implemented



## OS Management of Hardware

# OS Management of Hardware

- All processes only execute out of memory

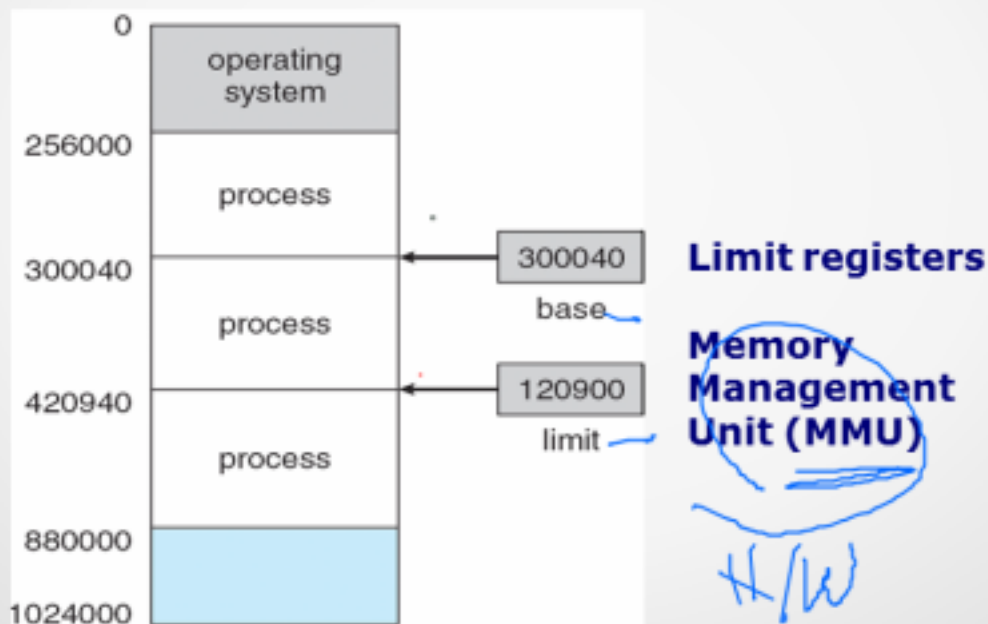




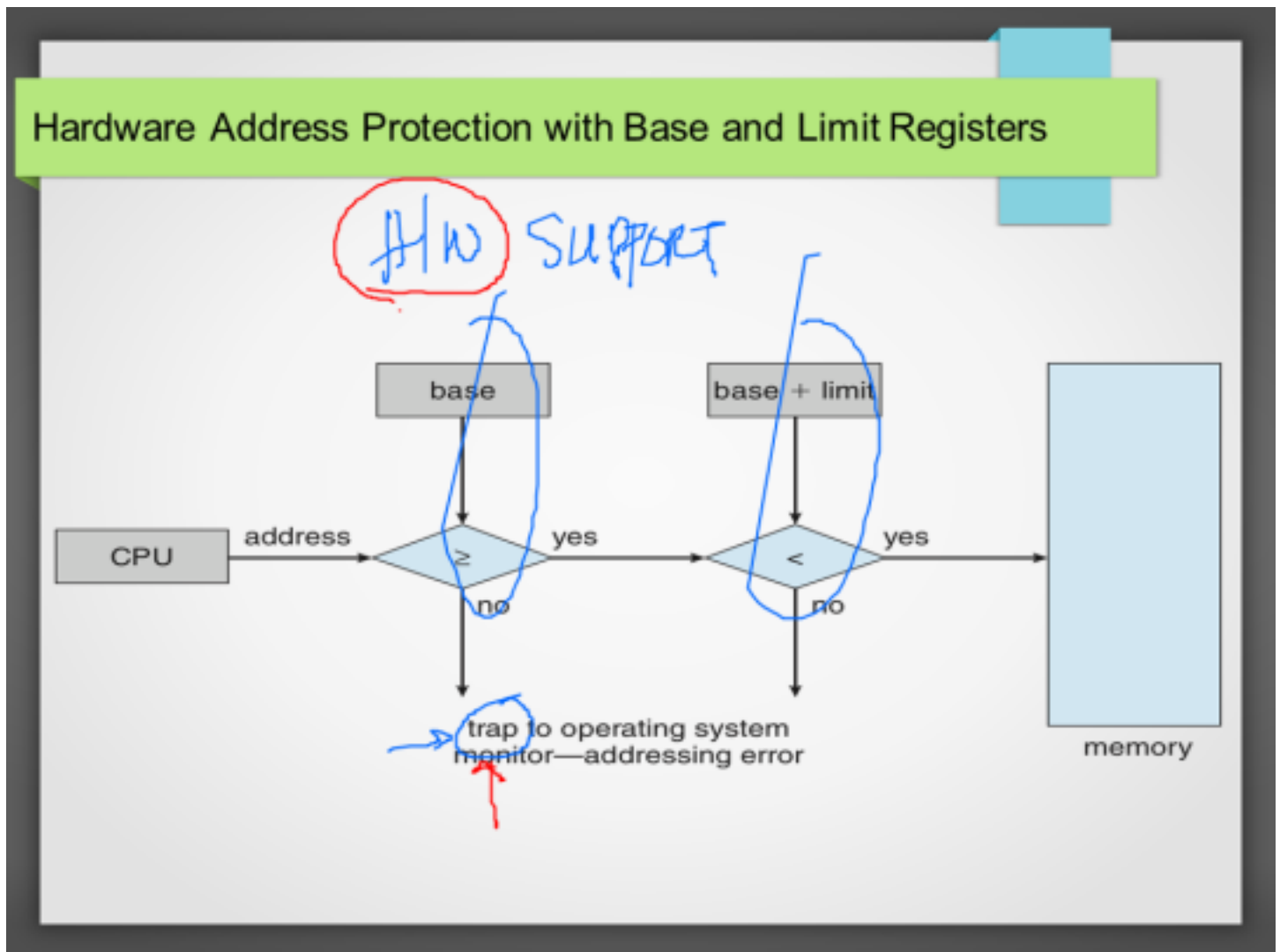
OS Management of Hardware

# OS Management of Hardware

- All processes only execute out of memory



Hardware Address Protection with Base and Limit Registers



## Address Binding

# Address Binding

- Program cannot always know available run-time address range (LINKING/LOADING)
- Addresses represented differently over lifetime of program (VIRTUAL MEMORY)
  - Source code uses symbolic references (labels)
  - Compiled code binds symbolic references to relocatable addresses
  - Linker/loader binds relocatable addresses to physical addresses

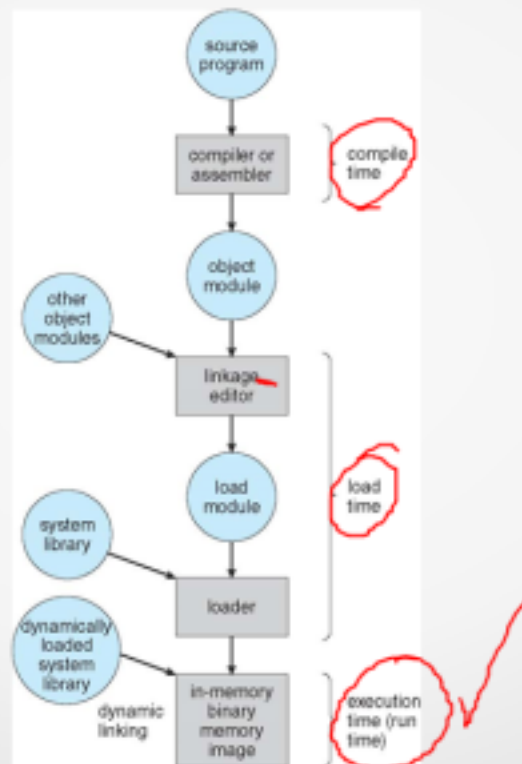
Binding of Instructions and Data to Memory

## Binding of Instructions and Data to Memory

- Address binding of instructions and data to memory addresses can happen at three different stages
  - **Compile time**: If memory location known a priori, absolute code can be generated; must recompile code if starting location changes
  - **Load time**: Must generate relocatable code if memory location is not known at compile time
  - **Execution time**: Binding delayed until run time if the process can be moved during its execution from one memory segment to another
    - Need hardware support for address maps (e.g., base and limit registers)

## Multistep Processing of a User Program

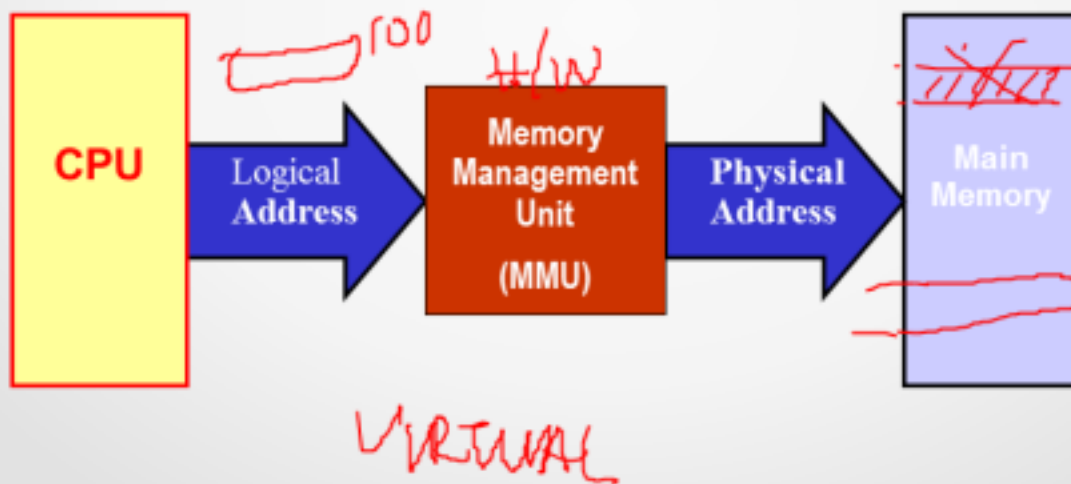
# Multistep Processing of a User Program



Logical vs. Physical Addresses

## Logical vs. Physical Addresses

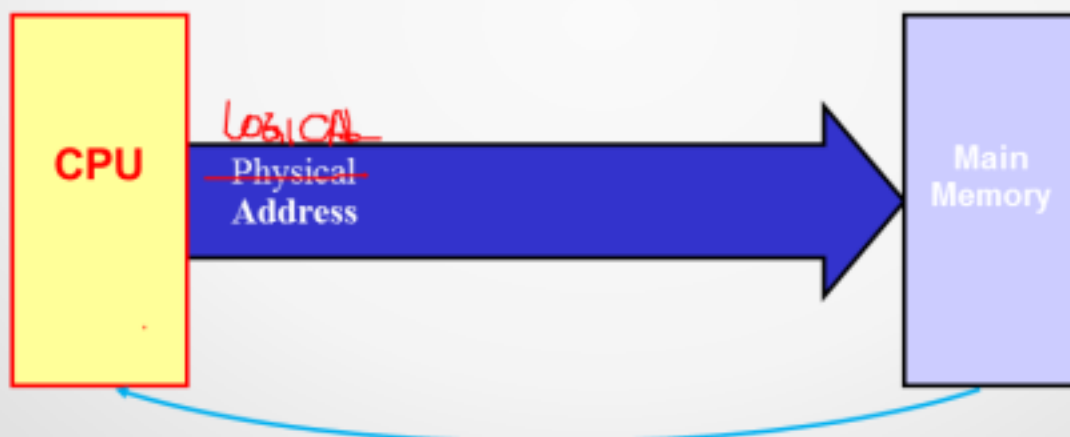
- Logical Address – Generated by the CPU
- Physical Address – Seen by memory system
- MMU required



## Logical vs. Physical Addresses

## Logical vs. Physical Addresses

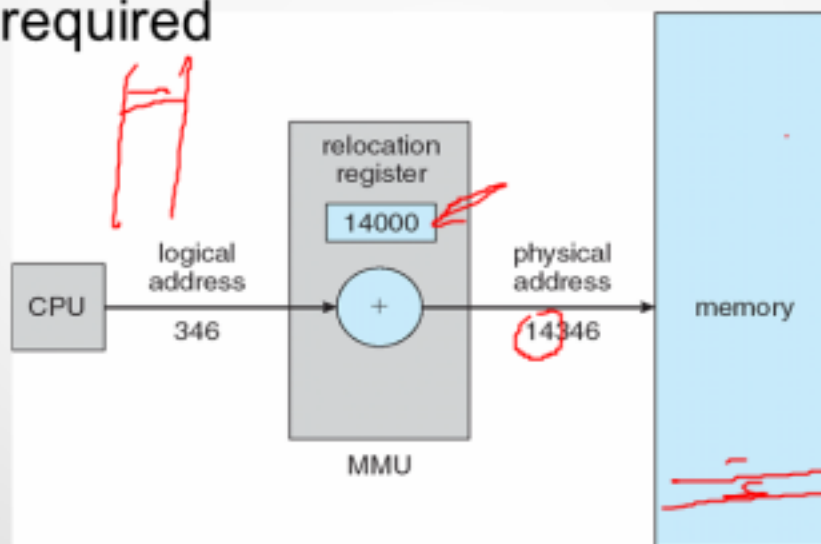
- Logical Address – Generated by the CPU
- Physical Address – Seen by memory system
- MMU required



## Logical vs. Physical Addresses

# Logical vs. Physical Addresses

- Logical Address – Generated by the CPU
- Physical Address – Seen by memory system
- MMU required

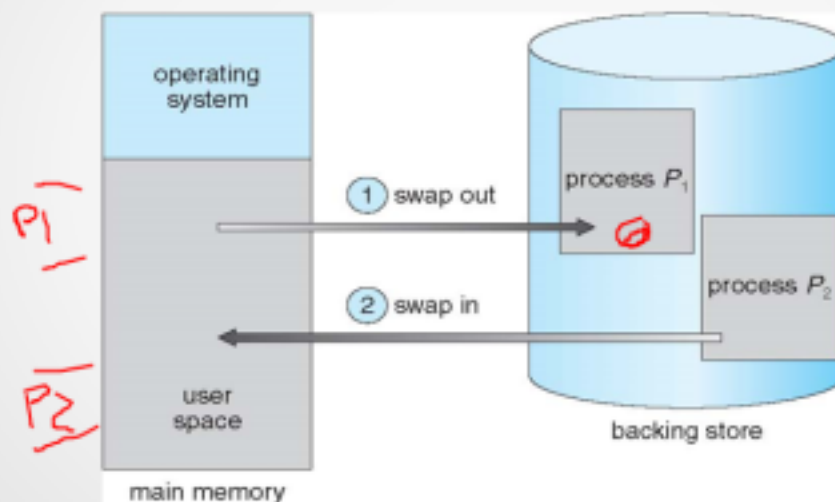




## Swapping

# Swapping

- Memory requirements exceed physical memory



- Time to context switch is very high

## Memory Allocation

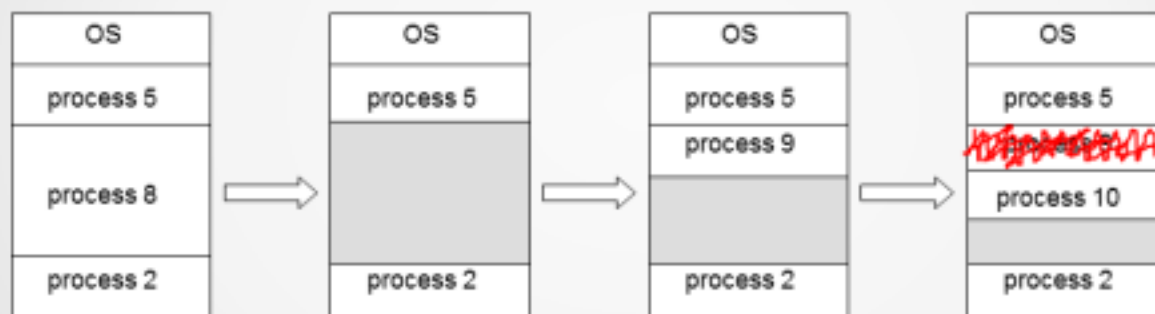
# <sup>RAM</sup>Memory Allocation

- Single partition
- Multiple partition

## Memory Allocation

# Memory Allocation

- Single partition
- Multiple partition



- Must handle fragmentation

## Dynamic Storage-Allocation Problem

# Dynamic Storage-Allocation Problem

How to satisfy a request of size  $n$  from a list of free holes?

- First-fit: Allocate the first hole that is big enough
- Best-fit: Allocate the smallest hole that is big enough; must search entire list, unless ordered by size
  - Produces the smallest leftover hole
- Worst-fit: Allocate the largest hole; must also search entire list
  - Produces the largest leftover hole

First-fit and best-fit better than worst-fit in terms of speed and storage utilization

## Fragmentation

# Fragmentation

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used
- First fit analysis reveals that given  $N$  blocks allocated,  $0.5 N$  blocks lost to fragmentation
  - $1/3$  may be unusable -> **50-percent rule**

Fragmentation (Cont.)

## Fragmentation (Cont.)

- Reduce external fragmentation by **compaction**  
*"DEFRAG"* →
- Shuffle memory contents to place all free memory together in one large block
- Compaction is possible *only* if relocation is dynamic, and is done at execution time
- I/O problem
  - Latch job in memory while it is involved in I/O
  - Do I/O only into OS buffers
- Now consider that backing store has same fragmentation problems