Operating Systems

Memory Management (Lecture-4)

Monsoon 2014, IIIT-H, Suresh Purini

First Design Goal

(Goal 1, aka G1): Every process should get the illusion that it has the entire address space available.

- Physical address space: The address space supported by the hardware.
- Logical/virtual address space: A process's view of its own memory
- Which is bigger, physical or virtual address space?
  A. Physical address space
  B. Virtual address space
  C. It depends on the system
- Reading Assignment: Physical Address Extension (PAE)

Three Techniques

1. Segmentation
2. Paging
3. Segmented Paging

Review

1. How much of G1 is achieved?
2. What are the issues?
   - Relocation, isolation and program memory allocation (size??)

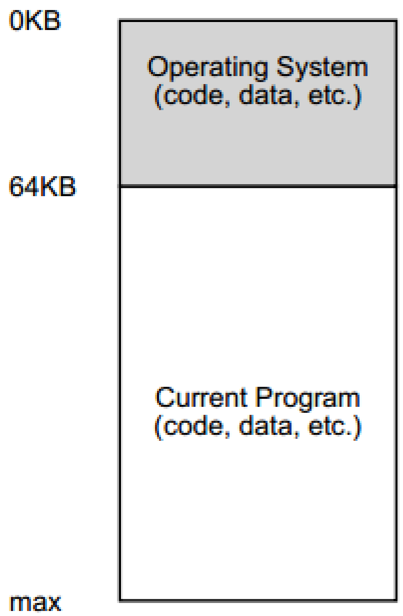


Figure 13.1: **Operating Systems: The Early Days**

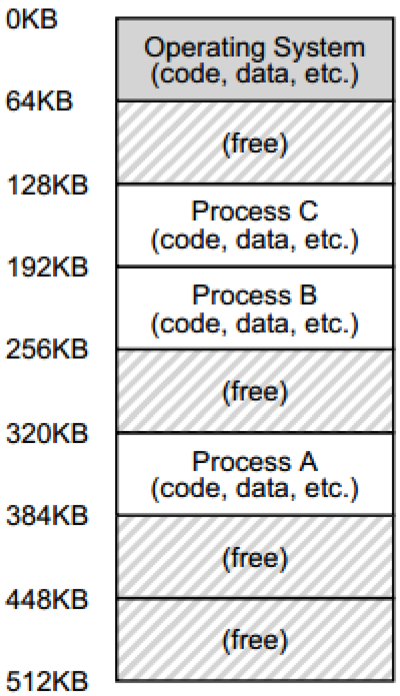
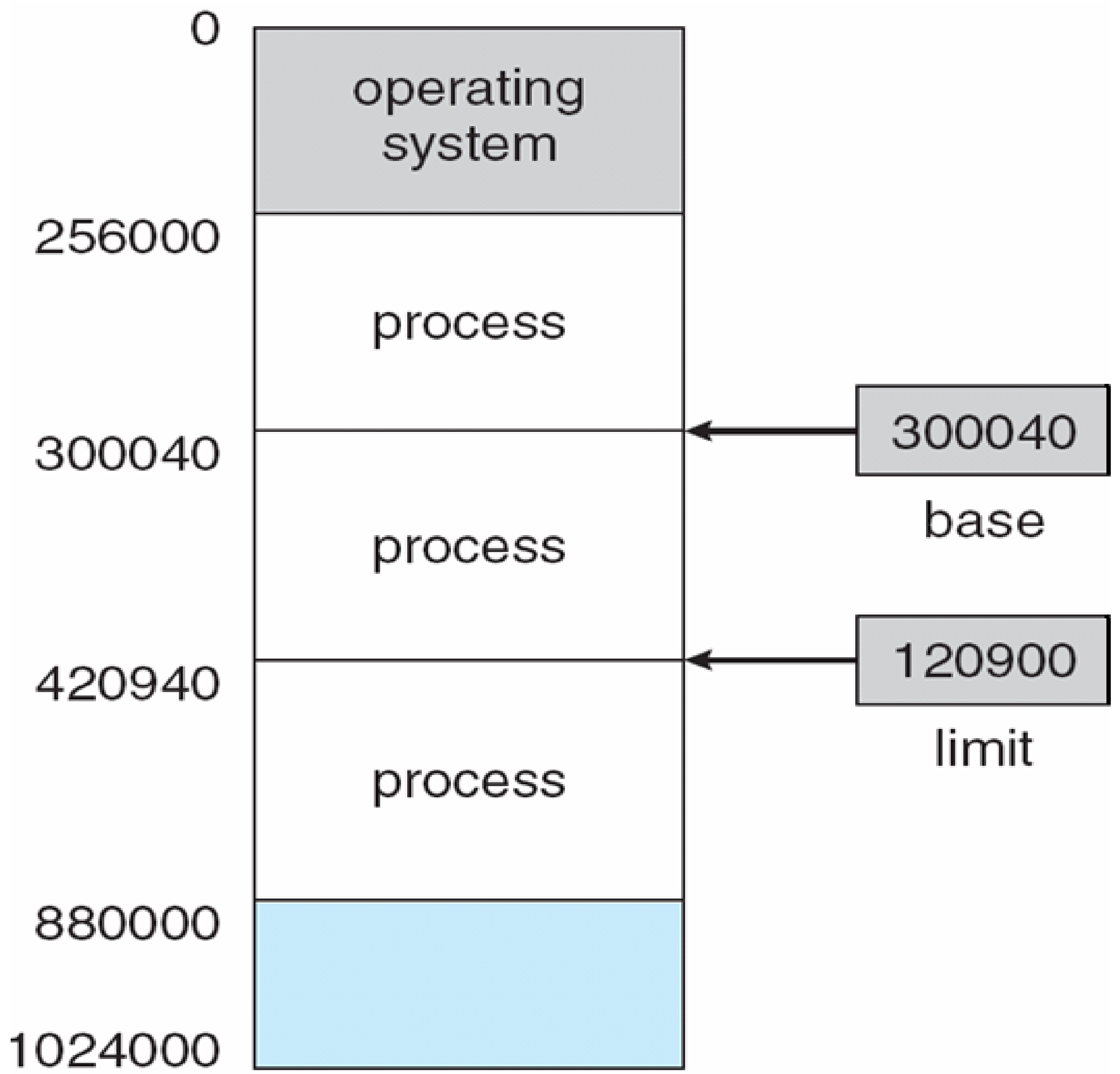


Figure 13.2: **Three Processes: Sharing Memory**

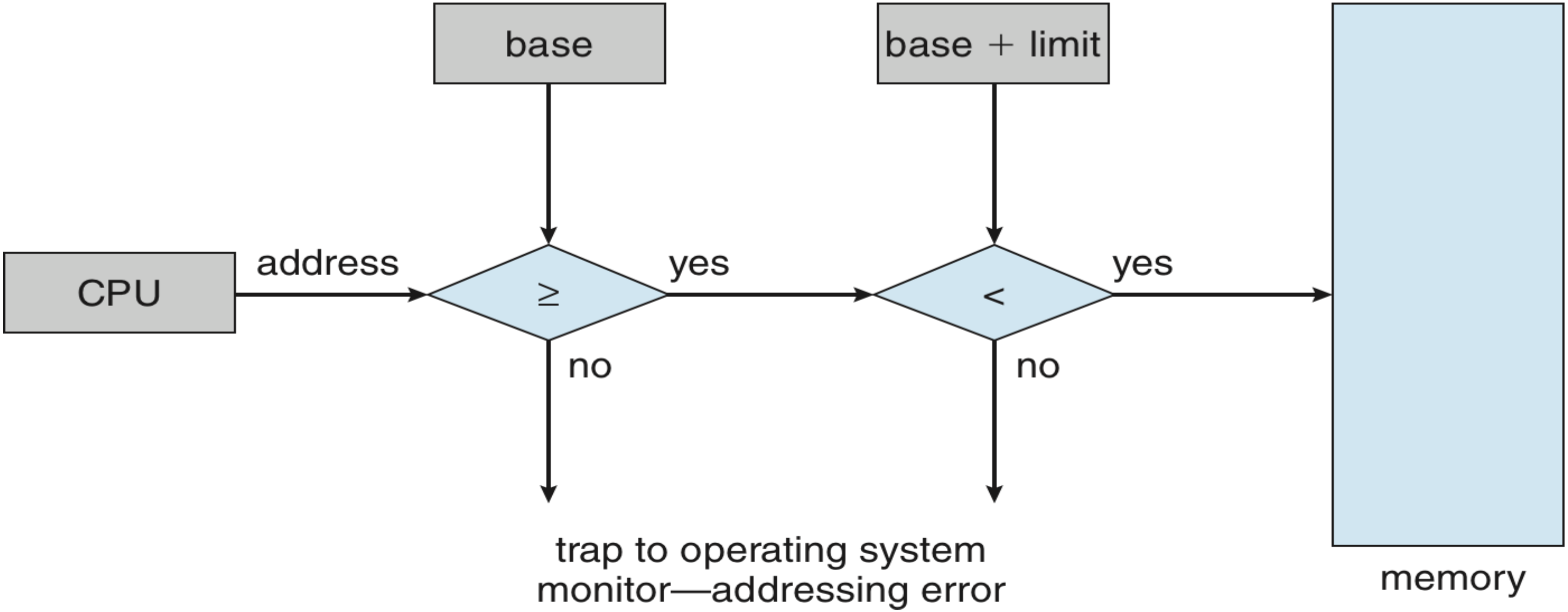
Basic Segmentation

- A pair of base and limit registers defines the logical address space.
  - Store and retrieve them during context switches



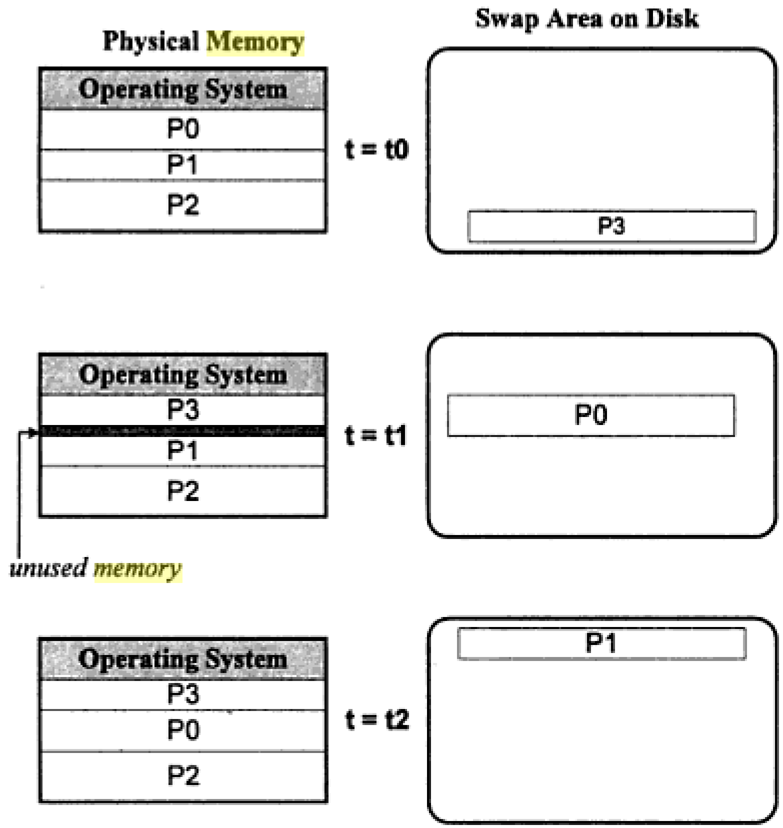
Basic Segmentation

- What issues are resolved and what are remaining?



Swapping

- What if there is not enough main memory to allocate for all the processes?
- What problem still persists?



Improved Segmentation

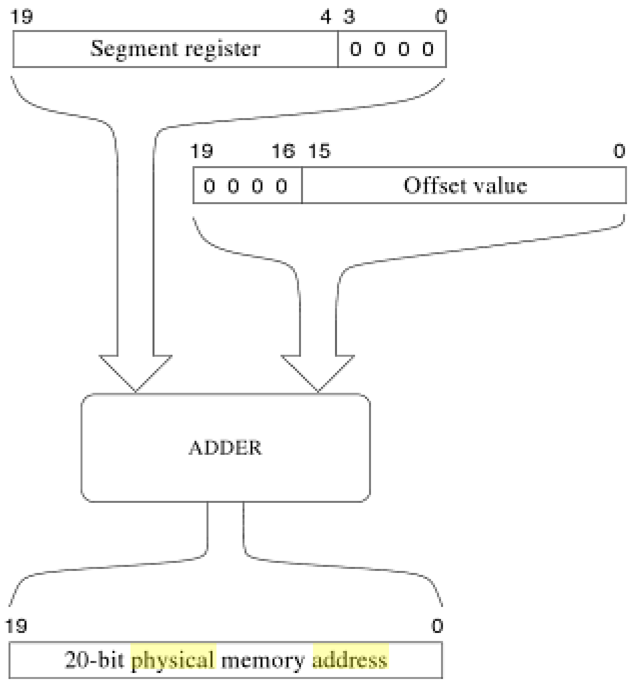
- Divide the address space of a process into regions/segments
  - Like stack, code, heap, text etc.
- Memory allocation for each region need not be contiguous
- For each region there is a base and bounds register pair
- Two approaches
  - Implicit
  - Explicit

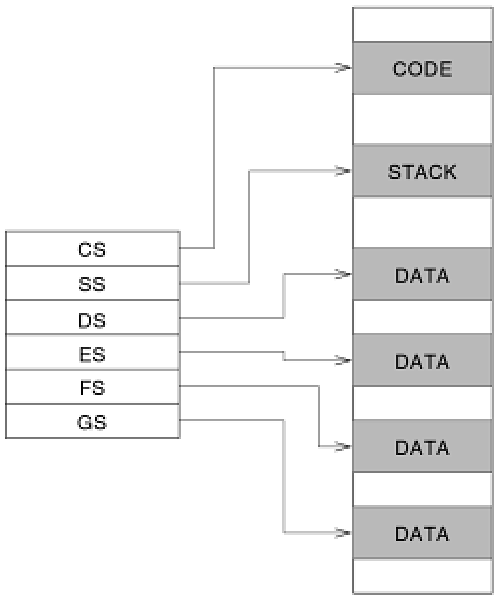
Explicit Segmentation – Intel x86 Real Mode

- Two modes of Intel Processors: Real (supports 8086) and Protected (default)
- Real mode

Can we run Linux in real mode?

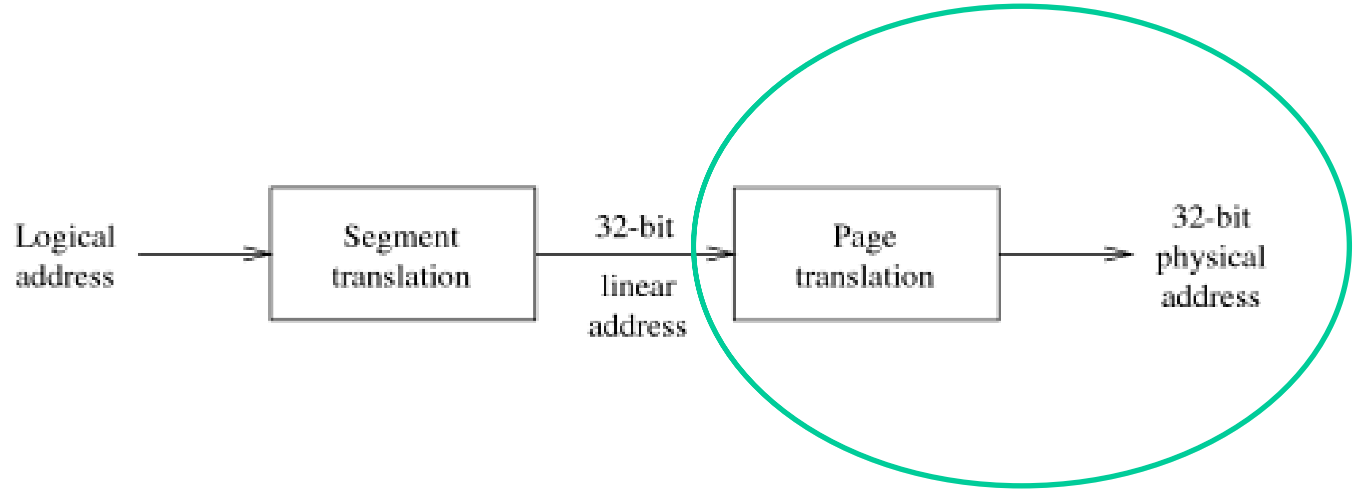
  - 20 address lines and hence the size of address space is 1 MB
  - Six 16-bit segment registers: CS, SS, DS, ES, FS, GS
  - Segment sizes are exactly 64K. No paging.





X86 – Real Mode Memory Management

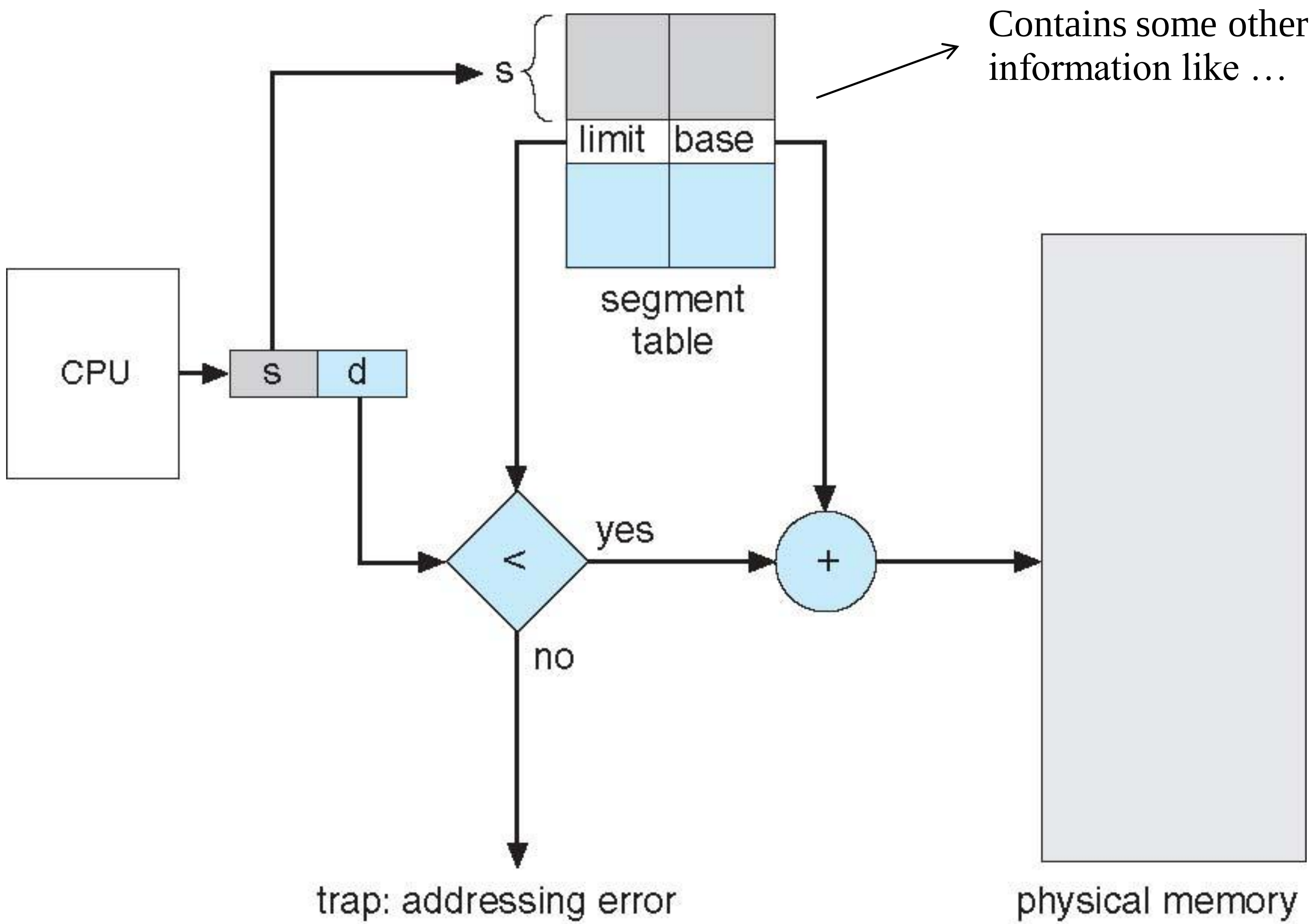
This part isn't there.



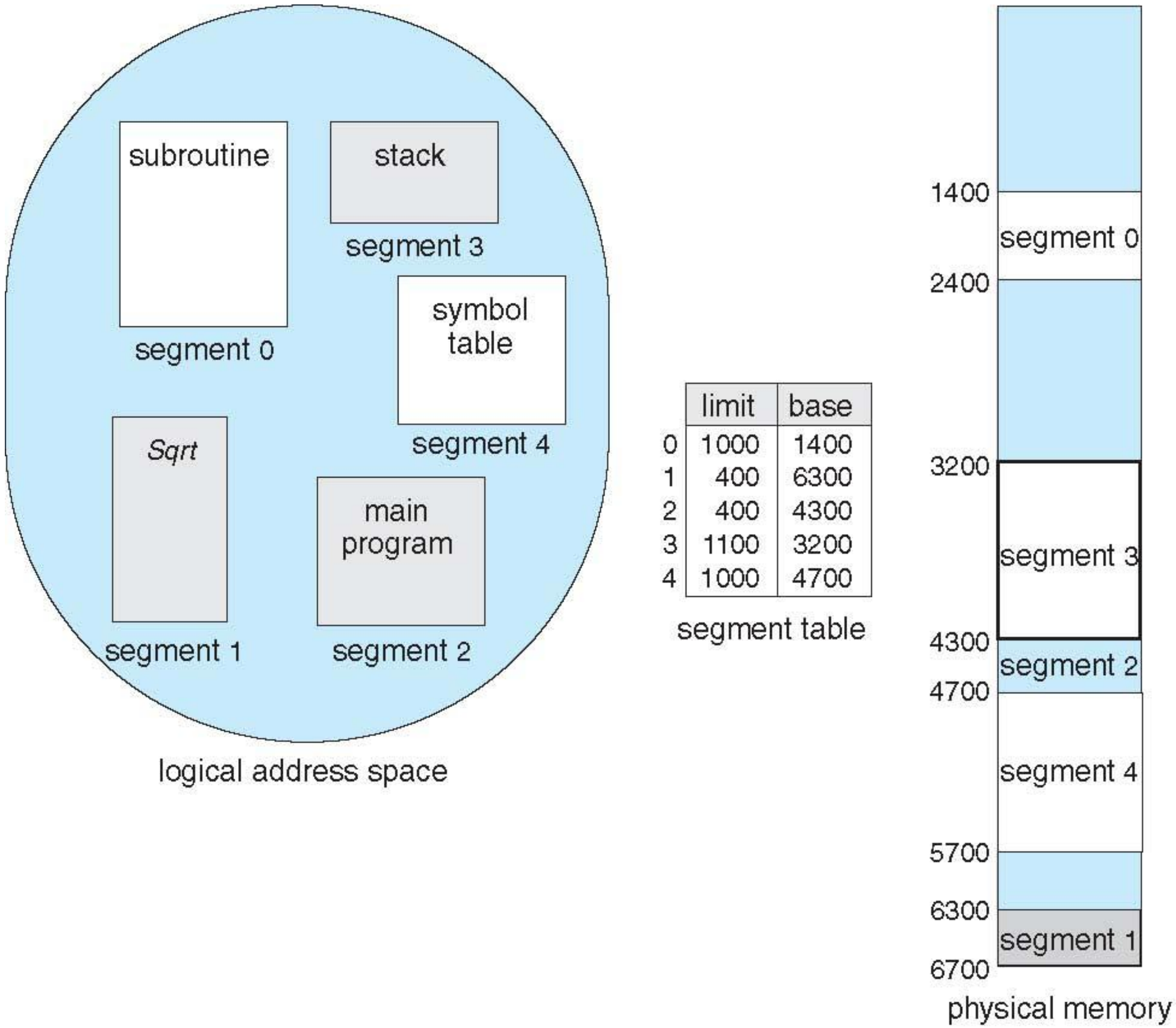
Implicit Segmentation

- Logical address implicitly contains the segment-number and the offset within.
- Segment table – maps logical address to physical address
- Two important registers
  - Segment-table base register (STBR) points to the segment table's location in memory
  - Segment-table length register (STLR) indicates number of segments used by a program

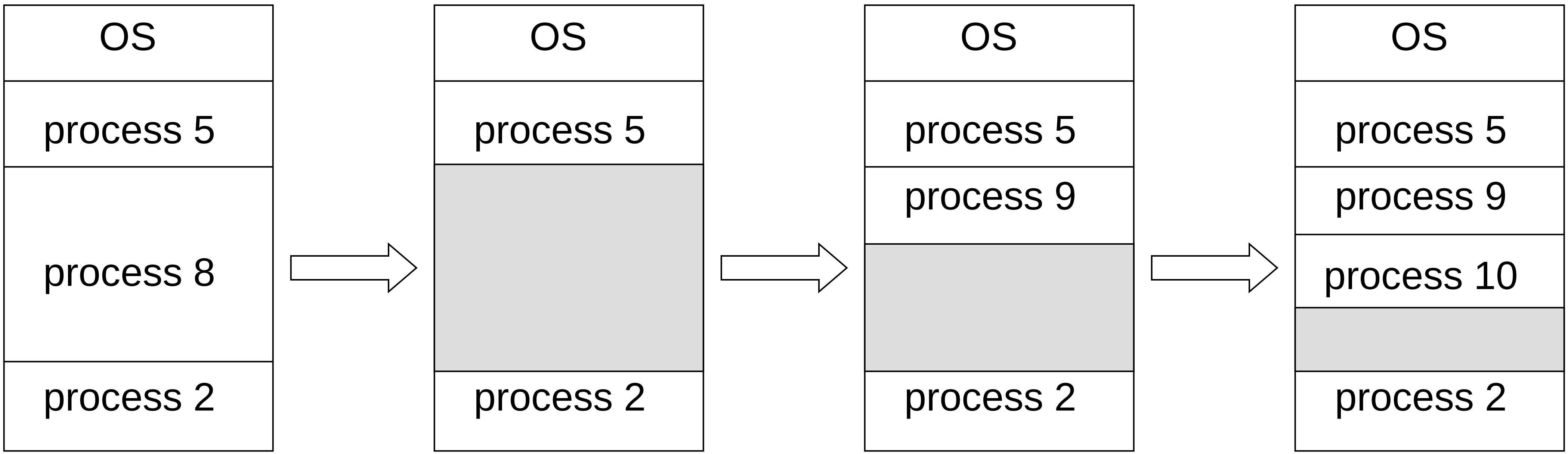
Implicit Segmentation Hardware



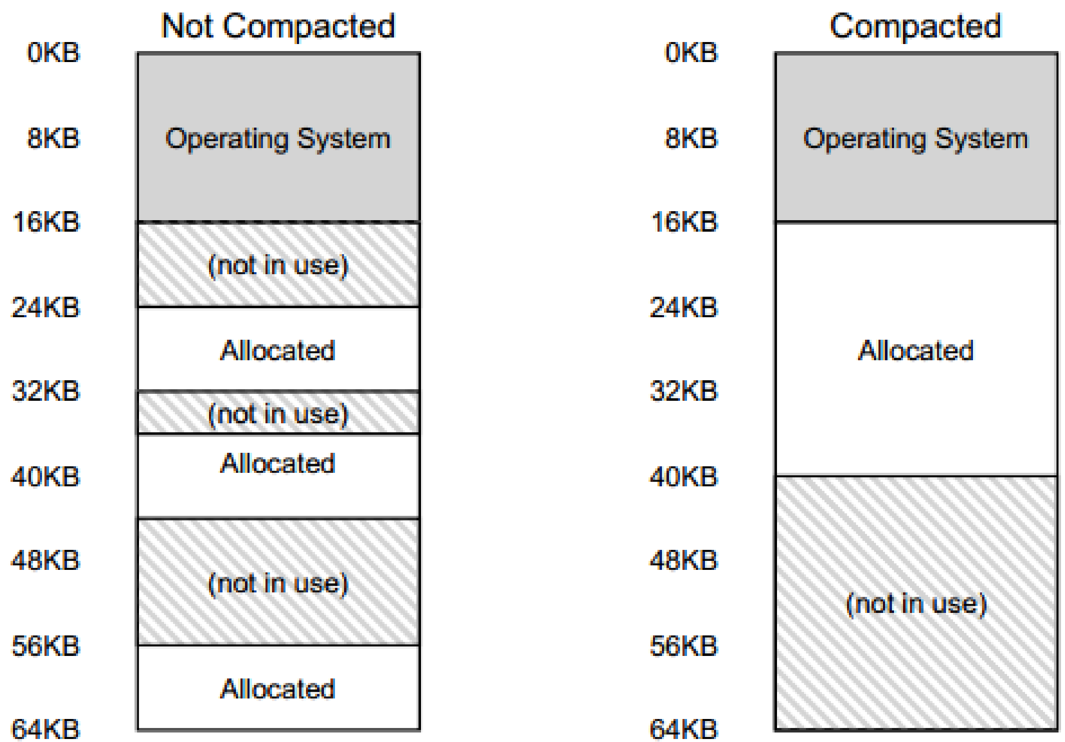
Example of Segmentation



Dynamic Memory Management



External Fragmentation and Compaction

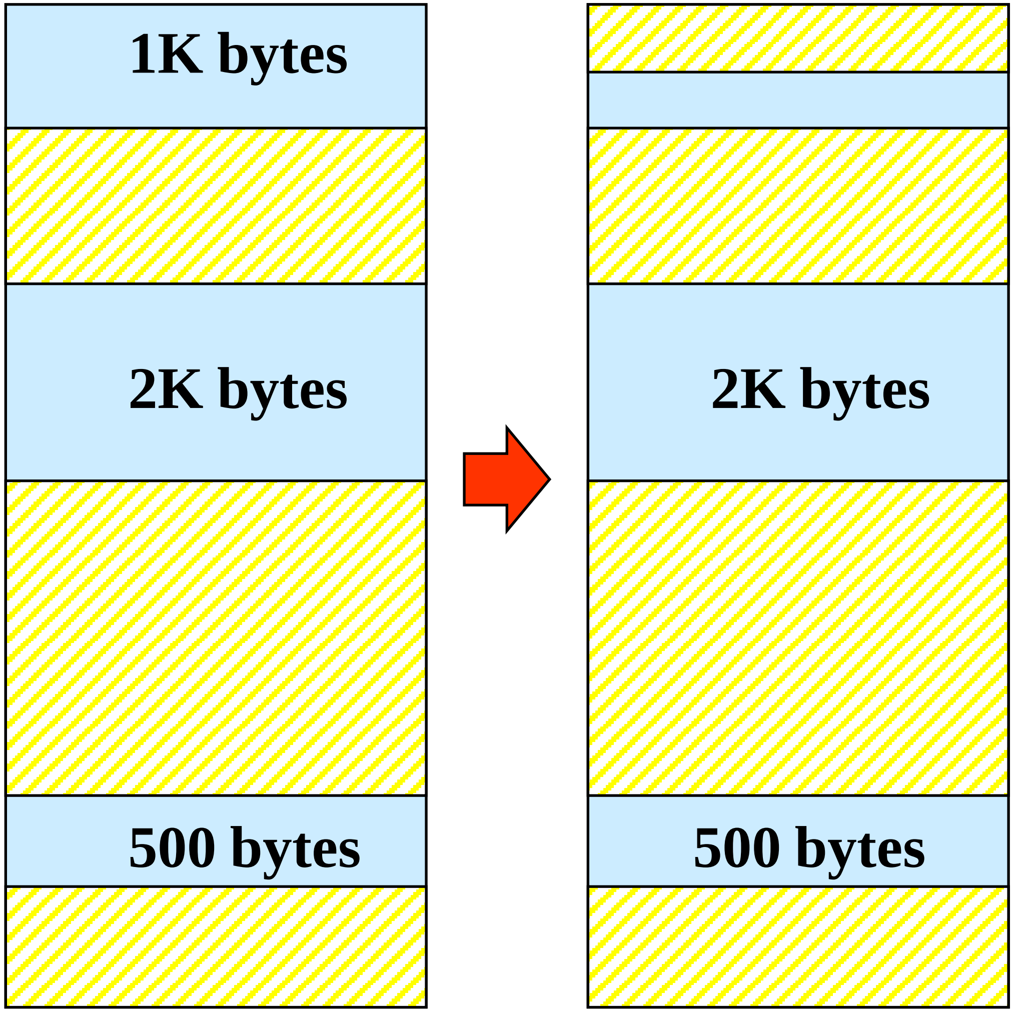


Dynamic Allocation Strategies

- 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 Allocation

To allocate n bytes, use the first available free block such that the block size is larger than n.



To allocate 400 bytes, we use the 1st free block available

Rationale & Implementation

- Simplicity of implementation
- Requires:
  - Free block list sorted by address
  - Allocation requires a search for a suitable partition
  - De-allocation requires a check to see if the freed partition could be merged with adjacent free partitions (if any)

Advantages

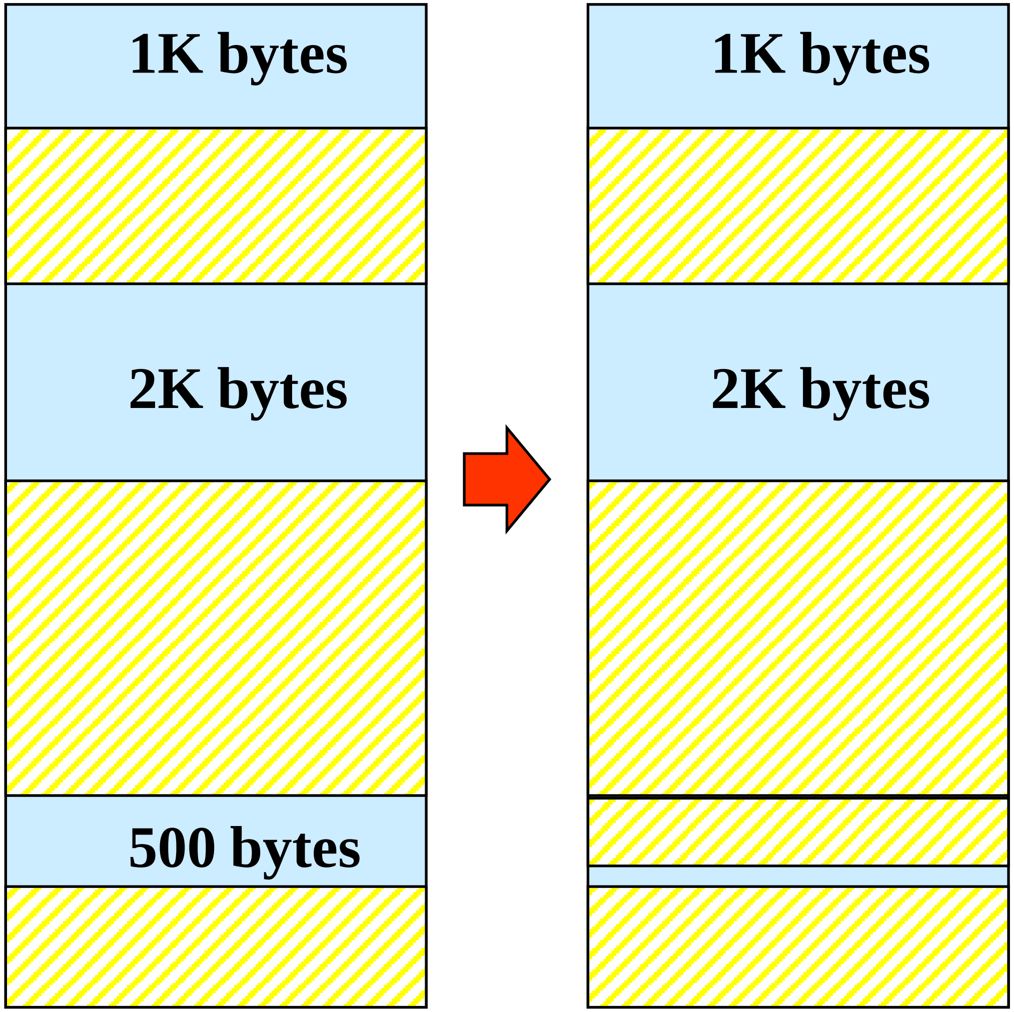
- Simple
- Tends to produce larger free blocks toward the end of the address space

Disadvantages

- Slow allocation
- External fragmentation

Best Fit Allocation

To allocate *n* bytes, use the *smallest* available free block such that the block size is larger than *n.*



To allocate 400 bytes, we use the 3rd free block available (smallest)

Rationale & Implementation

- To avoid fragmenting big free blocks
- To minimize the size of external fragments produced
- Requires:
  - Free block list sorted by size
  - Allocation requires search for a suitable partition
  - De-allocation requires search + merge with adjacent free partitions, if any

Advantages

- Works well when most allocations are of small size
- Relatively simple

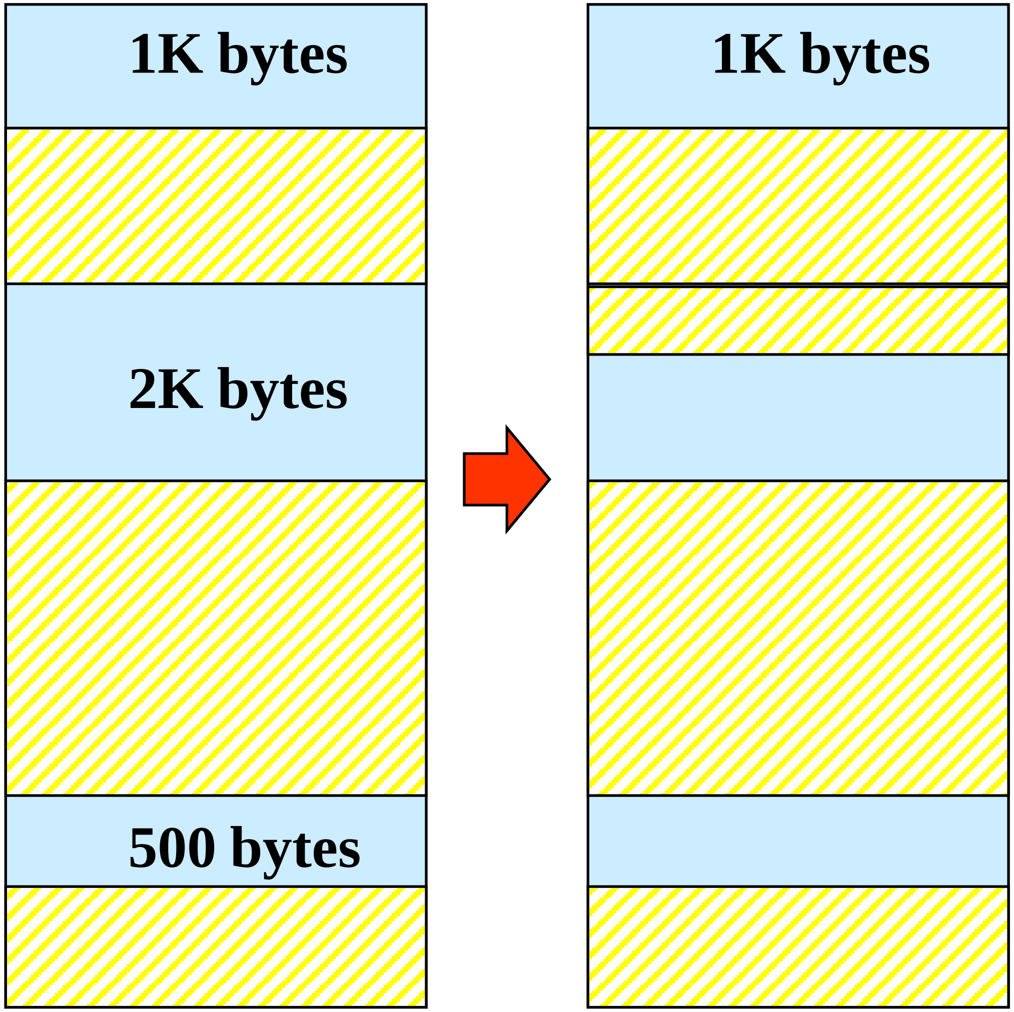
Disadvantages

- External fragmentation
- Slow de-allocation
- Tends to produce many useless tiny fragments (not really great)

- Doug Lea's malloc "In most ways this malloc is a best-fit allocator"

Worst Fit Allocation

To allocate *n* bytes, use the *largest* available free block such that the block size is larger than *n.*



To allocate 400 bytes,
we use the 2nd free block
available (largest)

Rationale & Implementation

- To avoid having too many tiny fragments
- Requires:
  - Free block list sorted by size
  - Allocation is fast (get the largest partition)
  - De-allocation requires merge with adjacent free partitions, if any, and then adjusting the free block list

Advantages

- Works best if allocations are of medium sizes

Disadvantages

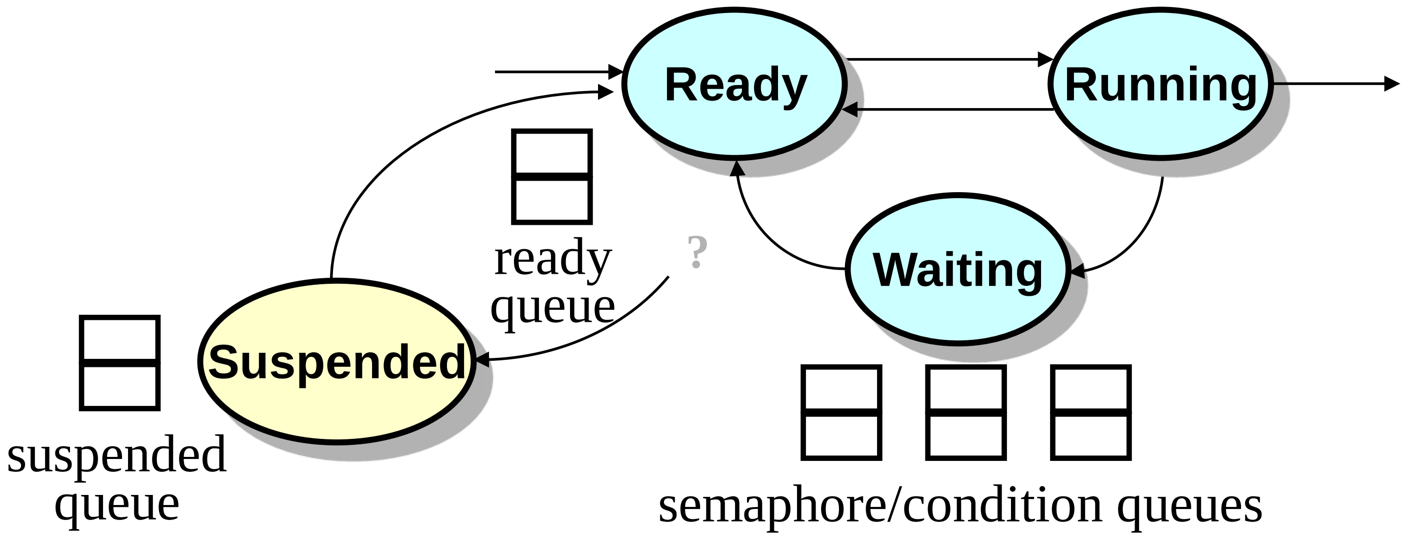
- Slow de-allocation
- External fragmentation
- Tends to break large free blocks such that large partitions cannot be allocated

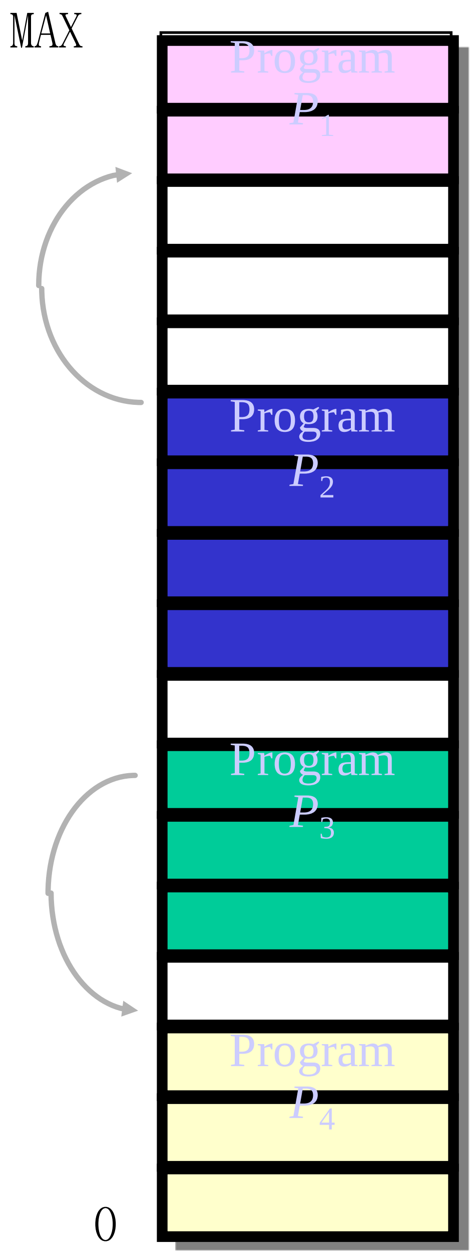
Dynamic Allocation of Partitions
Eliminating Fragmentation

Compaction

Relocate programs to coalesce holes

- **Swapping**
  - Preempt processes & reclaim their memory





Summary of Segmentation

Pros

1. Process and kernel isolation
2. Dynamic relocation
3. Segment sharing
4. ………

Cons

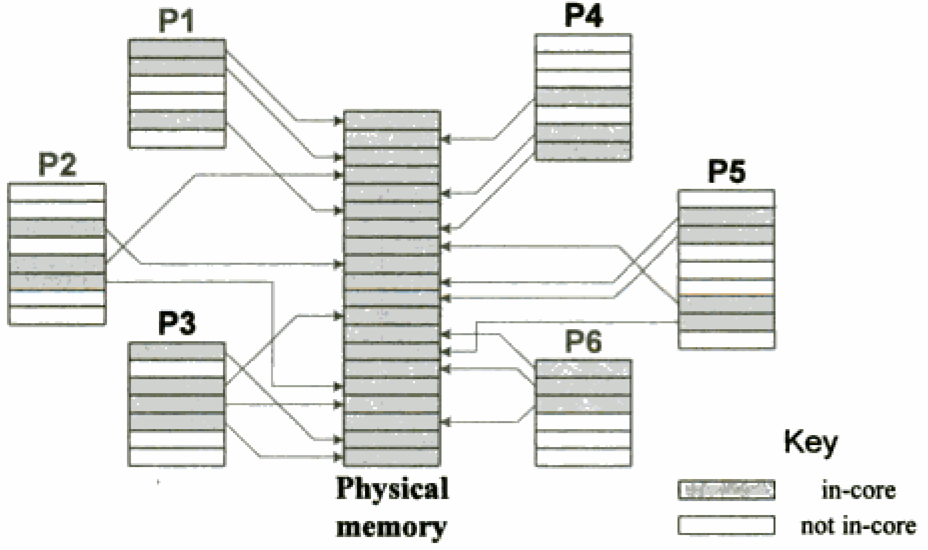
1. External fragmentation
2. Granularity of memory allocation, swapping, …
3. …….

Paging

Paging

Key Ideas

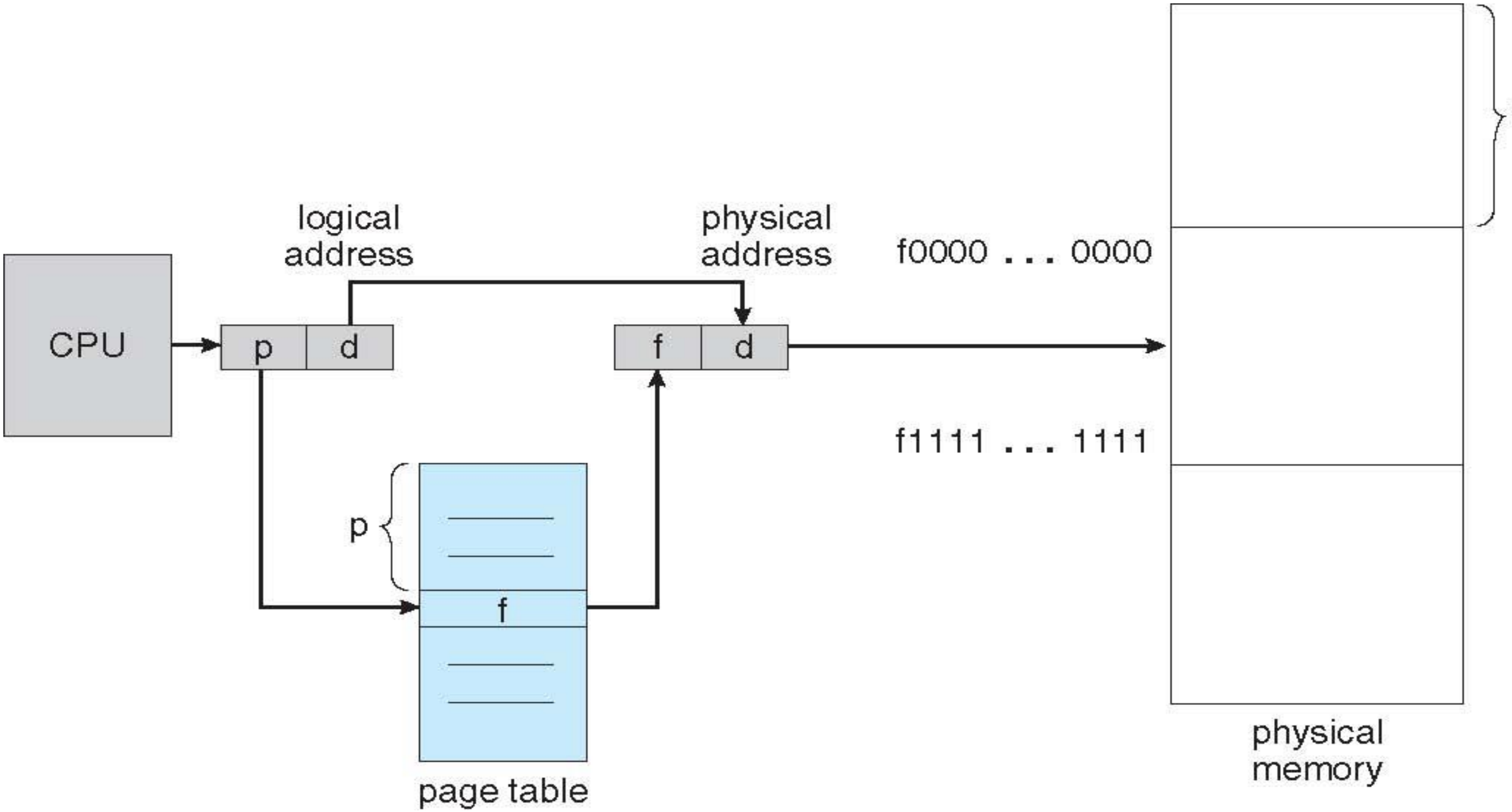
1. Physical address space of a process can be noncontiguous
2. Divide logical memory into blocks of same size called **pages**
3. Divide physical memory into fixed-sized blocks called **frames**
   - Size is on the order of KBs (compare it with cache line size)
4. Set up a **page table** to translate logical to physical addresses



Address Translation Mechanism

Question: Is the base address of the page table virtual?

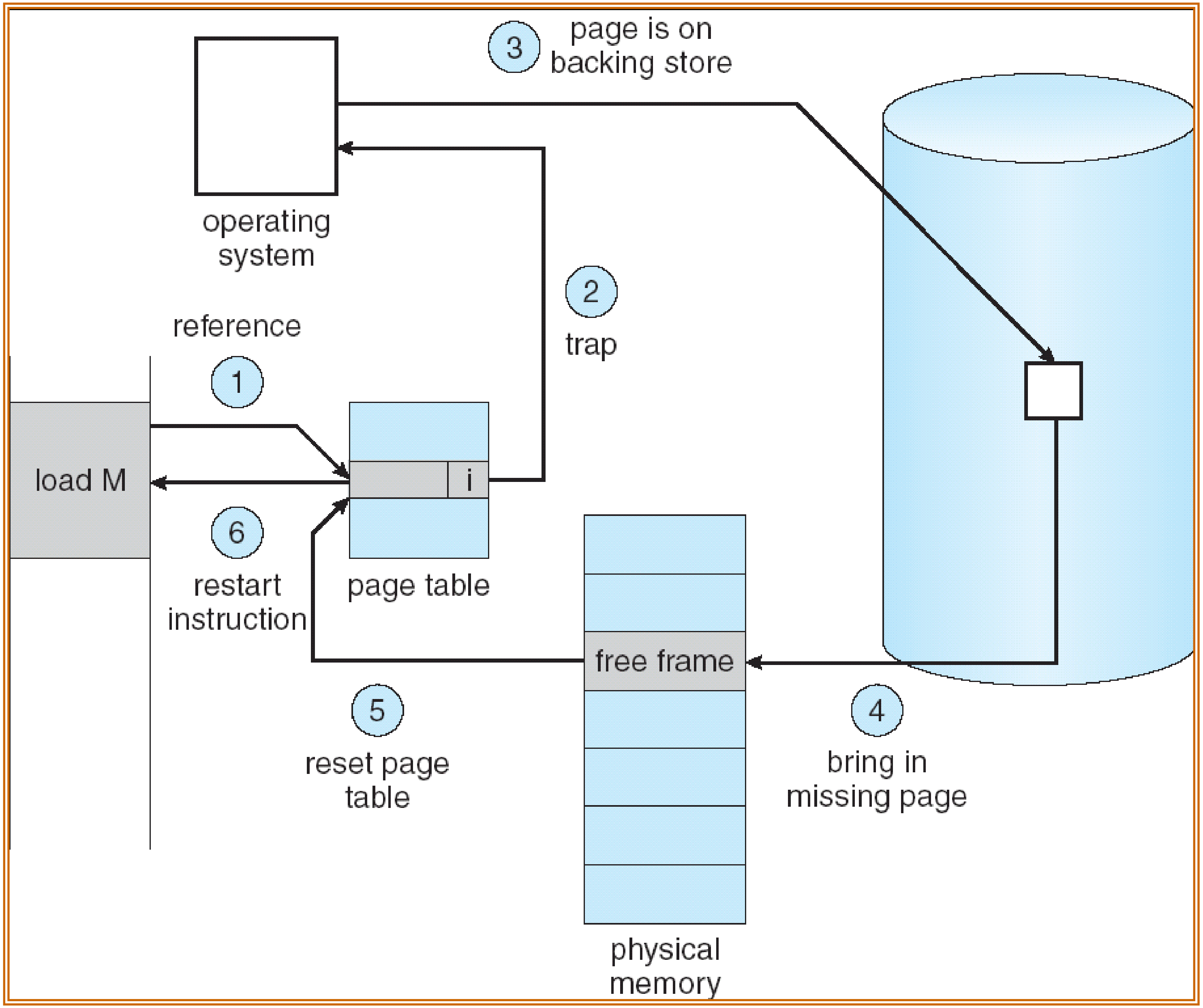
- Page Table Base Register holds the physical address (Check CR3 in x86).



Paging Advantages

1. What happens to external fragmentation? (turns internal)
   - On an average P/2 bytes per process gets wasted, where P is the page size.
   - Why not decrease page size?
2. Page sharing
3. Dynamic relocation
4. Process and kernel isolation
5. Is it necessary to have all process pages to be in physical memory?
   - Quick program start time, CoW, accommodate more active processes, …

Steps in Handling a Page Fault

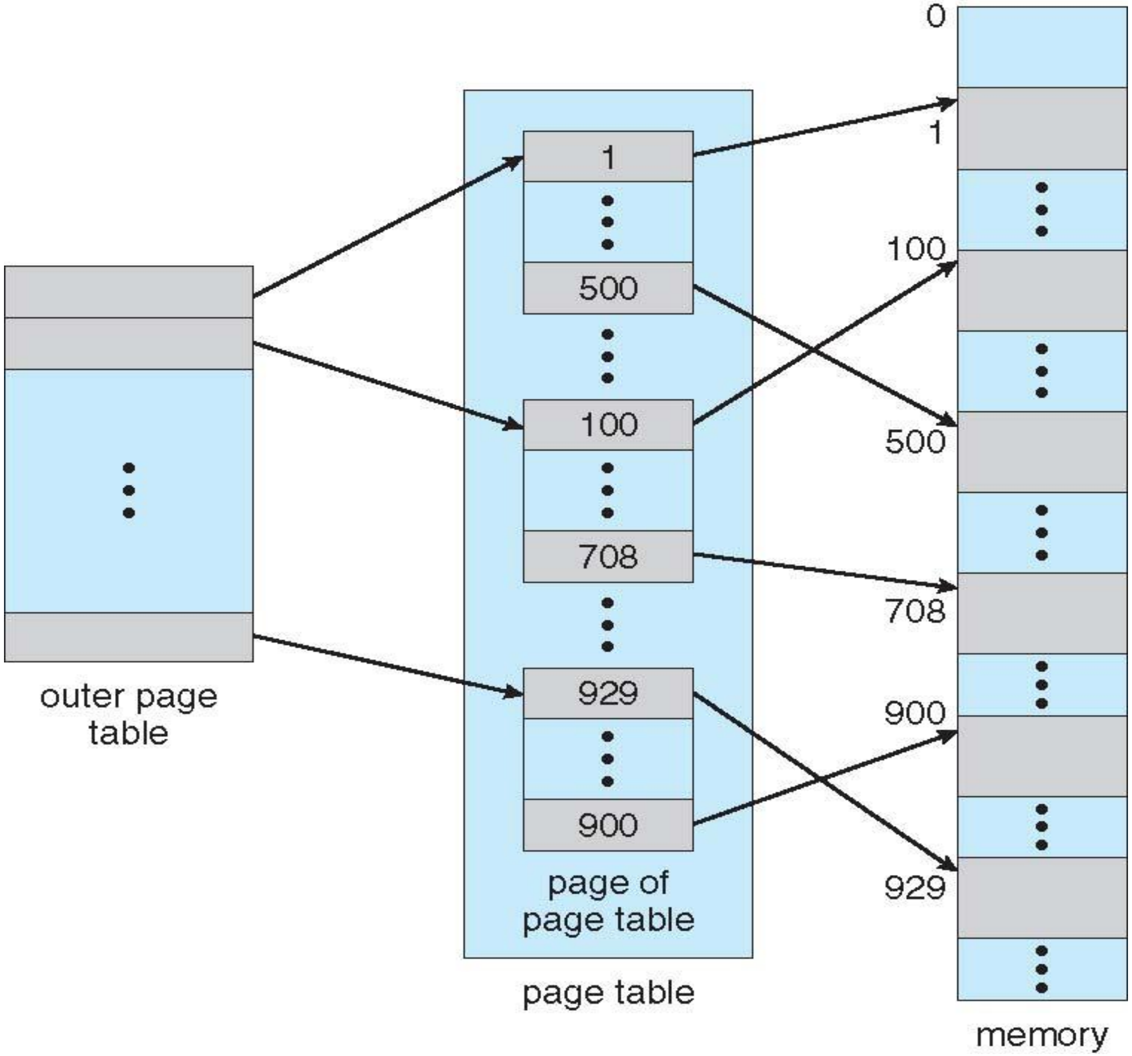


LW R1, ++SP . Weird things can happen. Read about Precise Exception Handling.

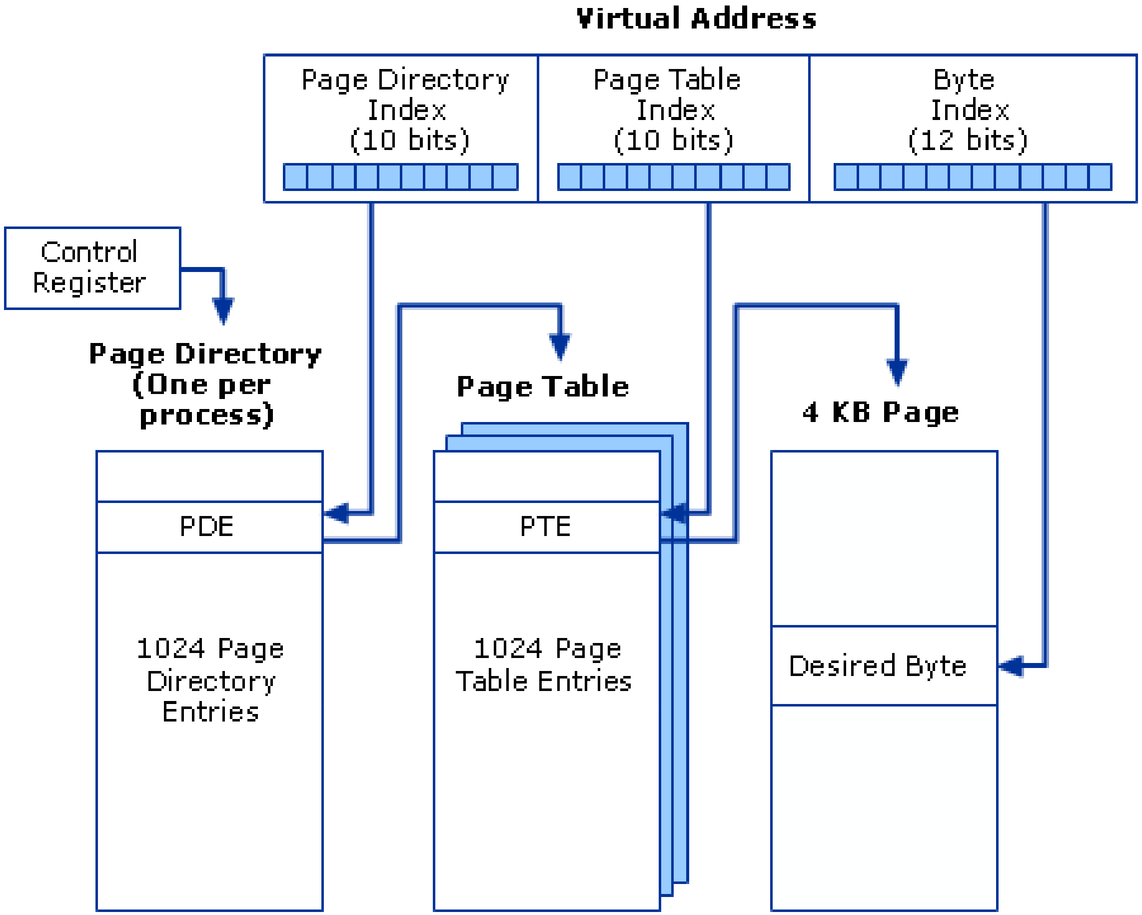
Paging

Question: 4 GB virtual address space and 4 KB page size. How many page table entries per each process?

- Idea: Use two level paging. Further, page the page tables.



Intel Two Level Paging Structure



Reading Homework: Read about inverted page tables.

Intel Segmented Paging

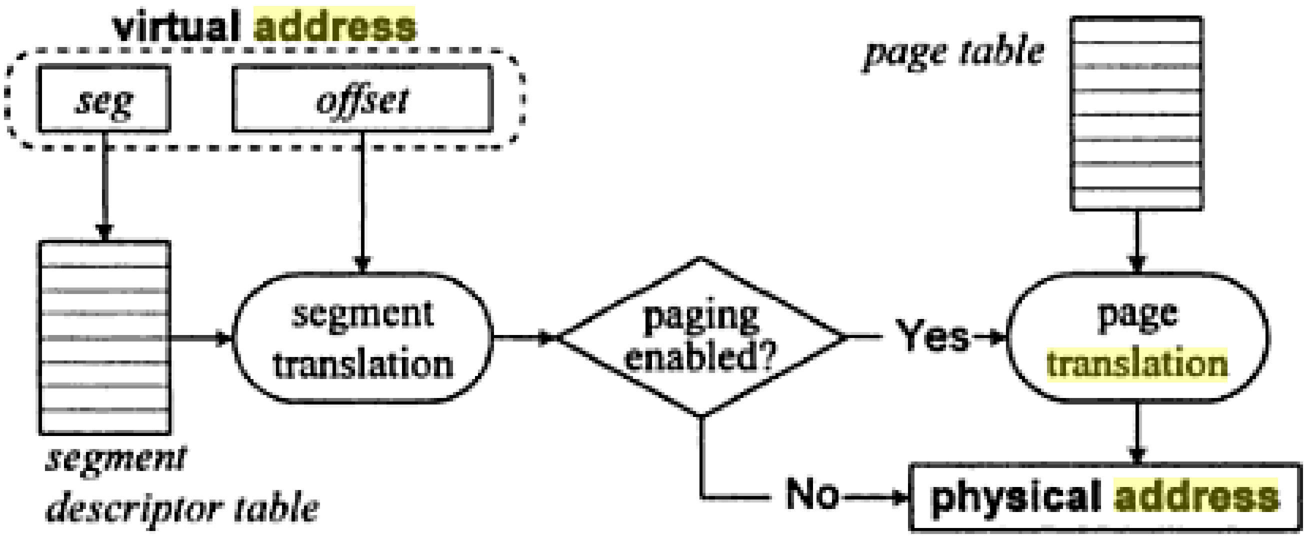
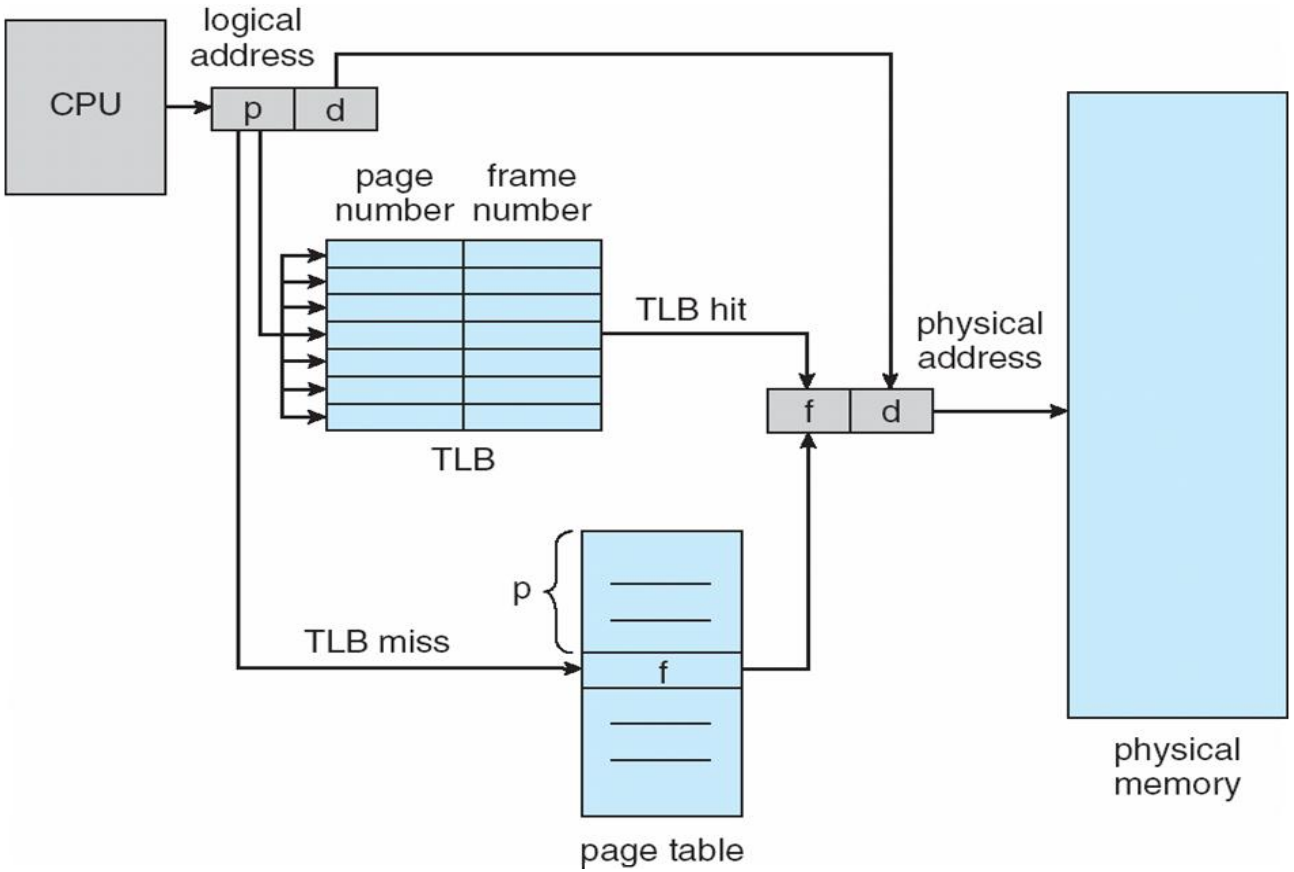


Figure 13-5. Address translation on the Intel 80x86.

Paging

Question: What could be the main problem with paging?

- It is expensive. Why?
- Idea: Use translation look-aside buffer (TLB)



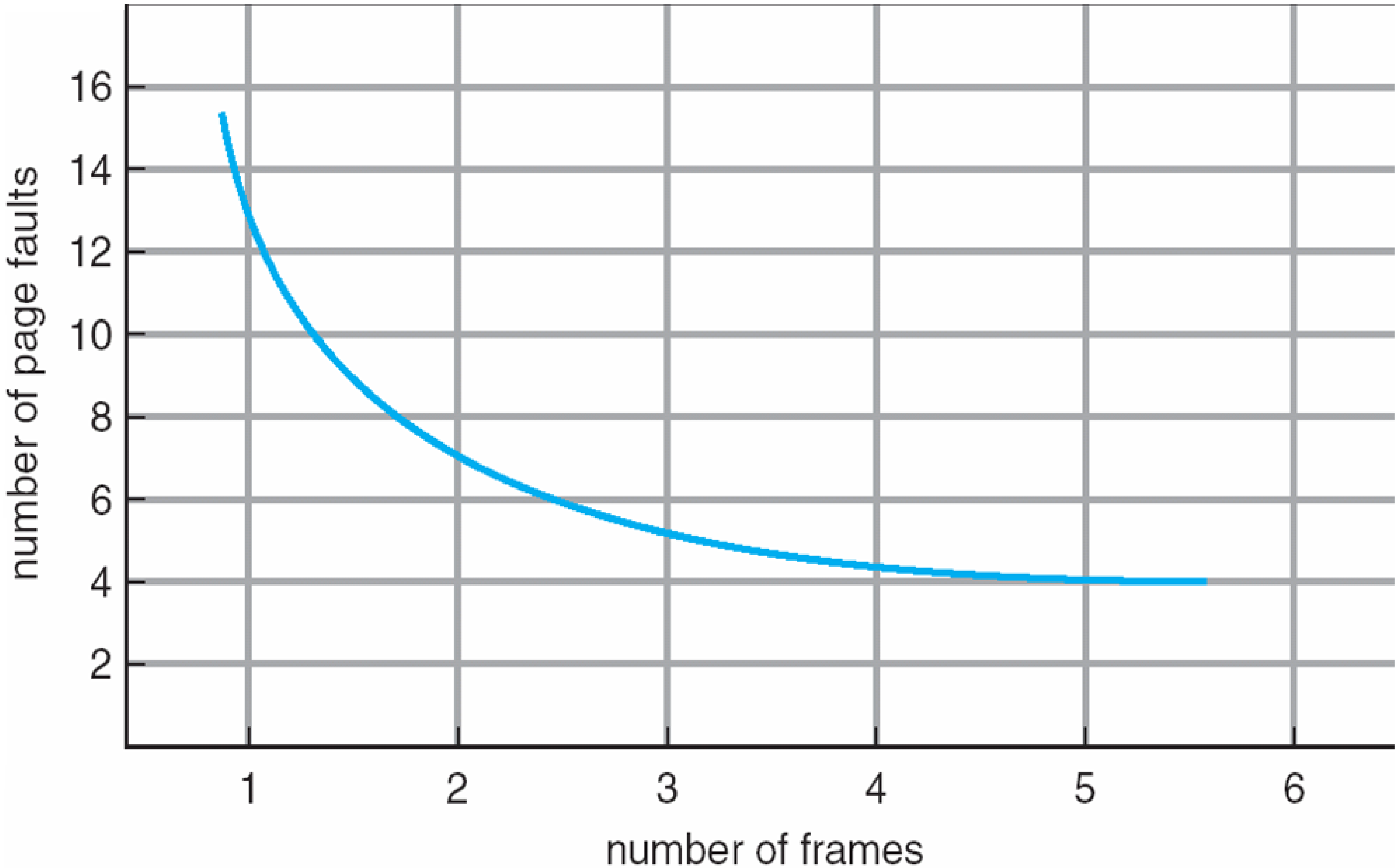
Reading Assignment: Read about design and implementation of Associative Memories.

Context Switches, Caching and TLBs

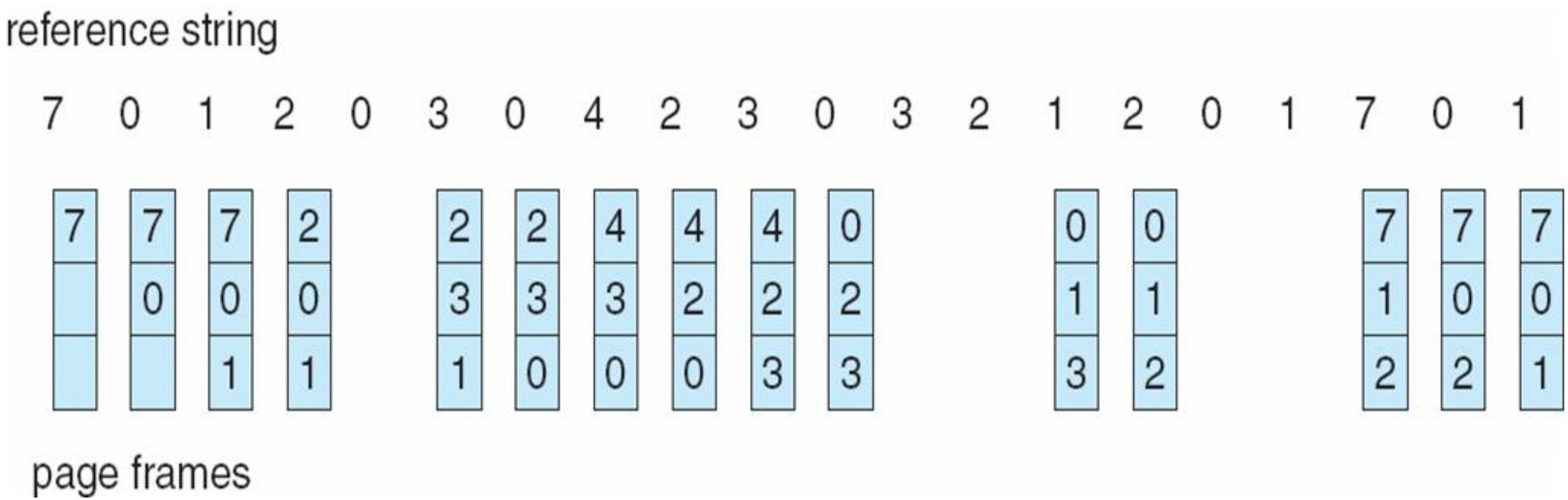
What’s the relation between them?

Page Replacement Algorithms

- Question: How to choose the victim page? (policy question)
- Objective: Minimize the number of page faults while being fair (local vs global).
- Basic Expectation: Larger the RAM size, smaller the number of page faults.



FIFO Page Replacement



Recall (Locality Hypothesis): Spatial and temporal locality properties of programs.

Belady's Anomaly

Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

3 frames :

1	1	4	5
2	2	1	3
3	3	2	4

9 page faults

4 frames:

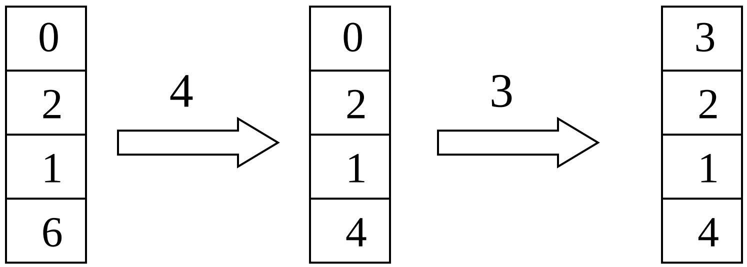
1	1	5	4
2	2	1	5
3	3	2	
4	4	3	

10 page faults

Optimal Page Replacement Algorithm

Idea: Replace the page which is used farthest in the future.

Reference String: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1



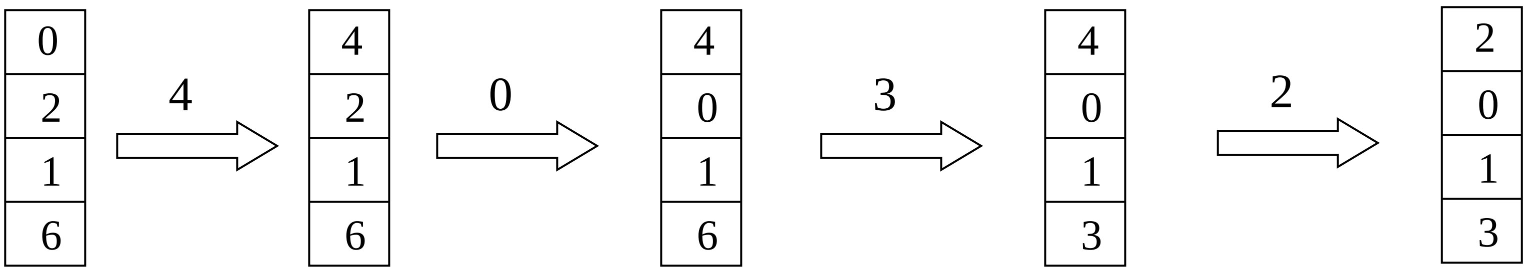
Fault Rate: 6 / 12 = 0.50

Read about competitive analysis of algorithms here.

http://www.eecs.berkeley.edu/~luca/cs261/lecture17.pdf

LRU Replacement

- Basic idea: Replace the page in memory that has not been accessed for the longest time.
- Optimal policy looking back in time as opposed to forward in time.

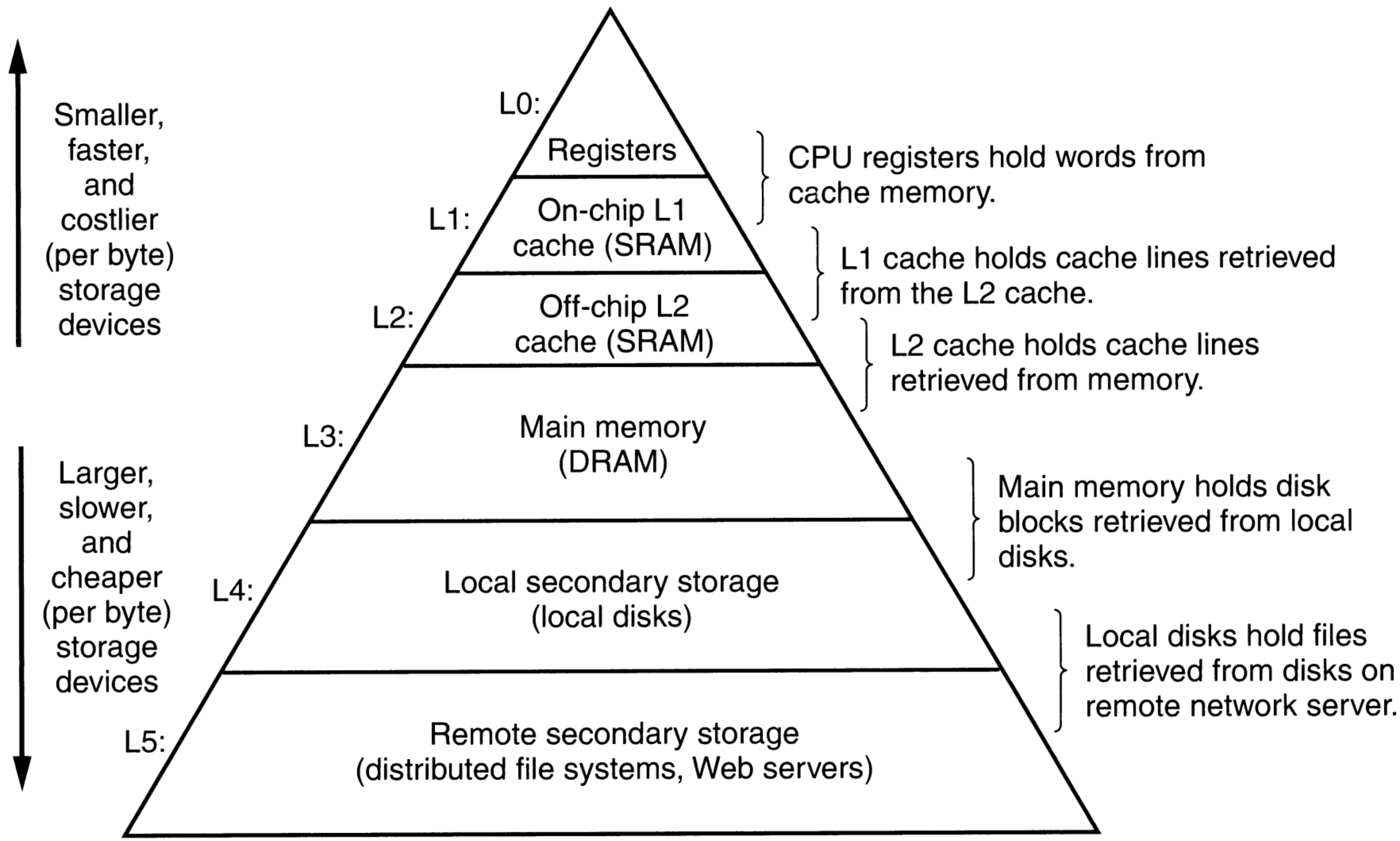


Working Sets and Thrashing

Page Daemon

Look at *kswapd* of Linux

Big Picture


Smaller, faster, and costlier (per byte) storage devices
Larger, slower, and cheaper (per byte) storage devices
L0:
Registers
L1:
On-chip L1 cache (SRAM)
L2:
Off-chip L2 cache (SRAM)
L3:
Main memory (DRAM)
L4:
Local secondary storage (local disks)
L5:
Remote secondary storage (distributed file systems, Web servers)
CPU registers hold words from cache memory.
L1 cache holds cache lines retrieved from the L2 cache.
L2 cache holds cache lines retrieved from memory.
Main memory holds disk blocks retrieved from local disks.
Local disks hold files retrieved from disks on remote network server.