1

Main Memory

COS 450 - Fall 2018

2

What is **Main** Memory

...fetch and store from CPU to memory

3

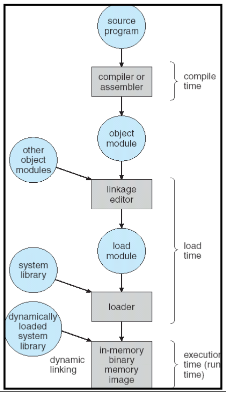
a process in memory...

When does a variable get an address?

- compile-time - absolute
- load-time - relocatable
- run-time - dynamic

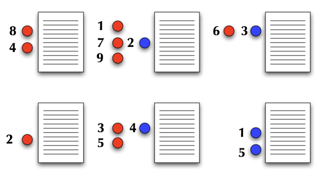
4

Life of a Program



5

Multiprocess Environment



How is memory

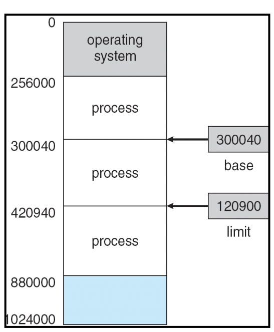
organized and **protected**?

6

Simple Base + Limit CPU register solution

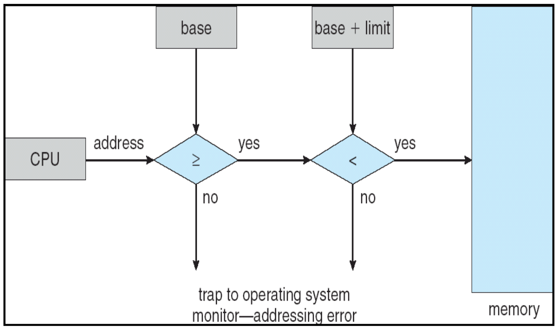
protection of processes from each other

relocation of code for execution



7

Base + Limit Registers



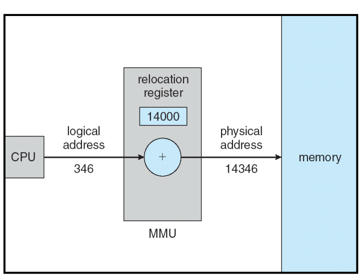
8

Logical Address Space

address relocation defines a **Logical** address space as well as a **Physical** address space.

9

Memory Management Unit



hardware to **map** logical to physical addresses

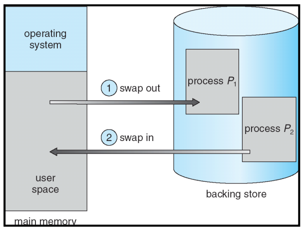
10

Not enough RAM?

Dynamic Loading

Dynamic Linking & Shared Libraries

Swapping



11

Allocation Strategies

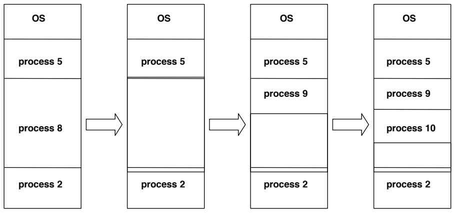
there are two basic strategies;

Contiguous

Non-Contiguous

12

Contiguous Allocation



Finding Space

13

to determine where to place a process we can use;

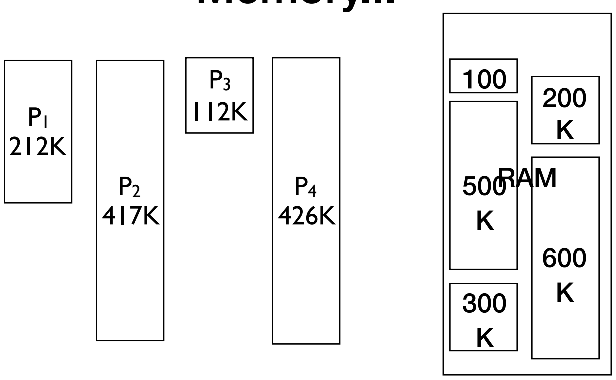
First-Fit

Best-Fit

Worst-Fit

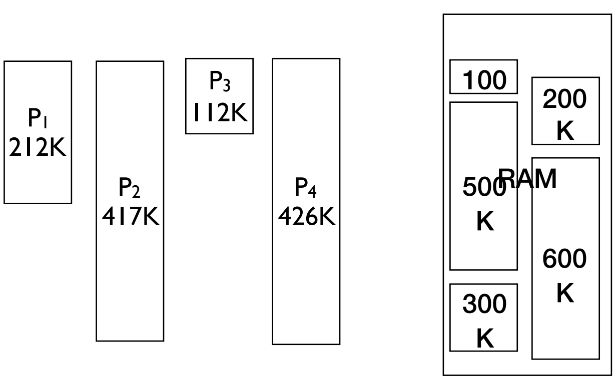
Processes that need Memory...

14



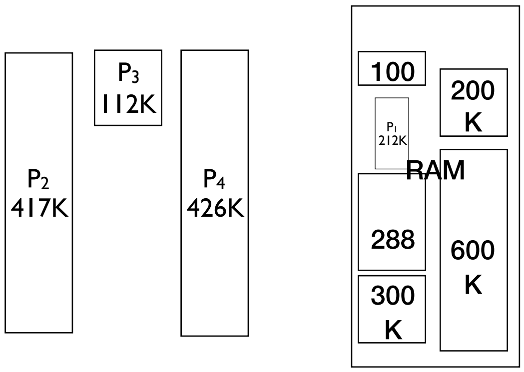
First-Fit Allocation

15-1



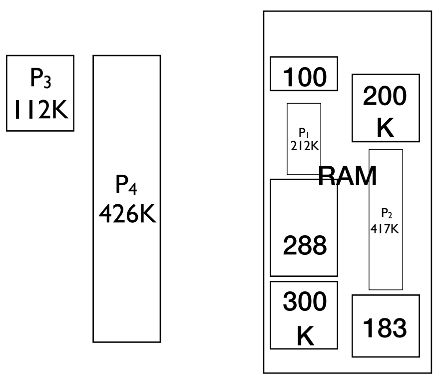
15-2

First-Fit Allocation



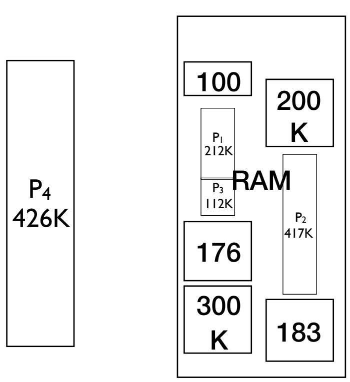
15-3

First-Fit Allocation



15-4

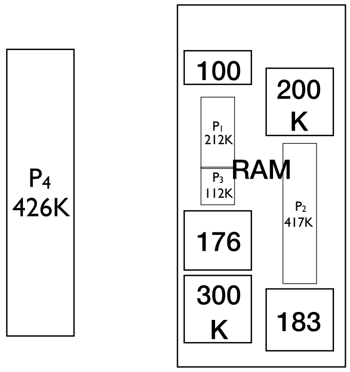
First-Fit Allocation



15-5

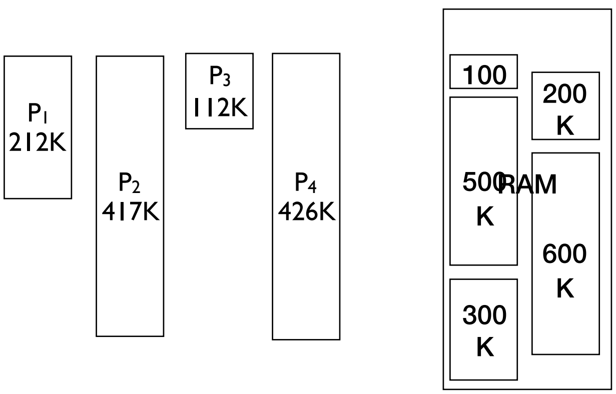
First-Fit Allocation

P4 does not fit!



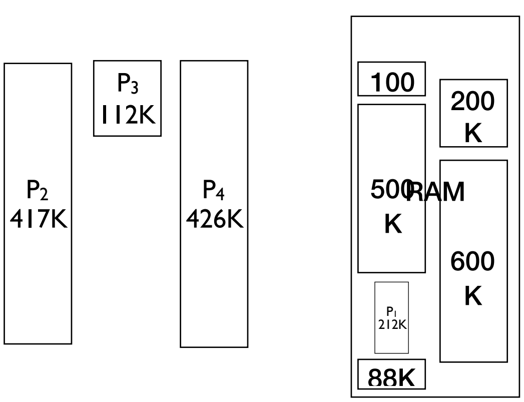
16-1

Best-Fit Allocation (work)



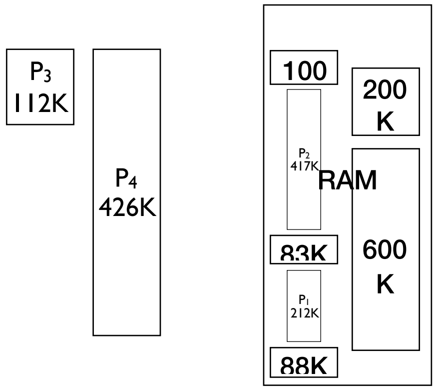
16-2

Best-Fit Allocation (work)



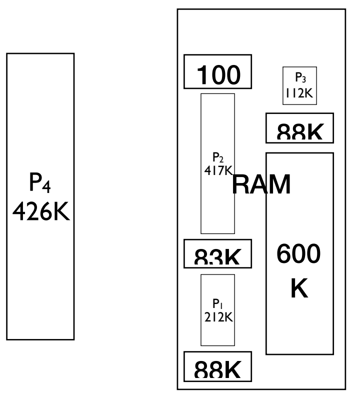
16-3

Best-Fit Allocation (work)



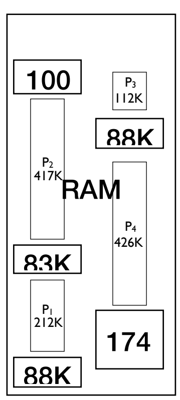
16-4

Best-Fit Allocation (work)



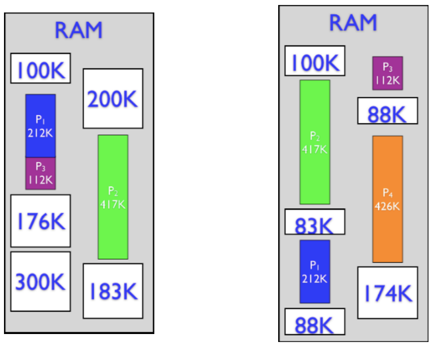
16-5

Best-Fit Allocation (work)



17

Comparison

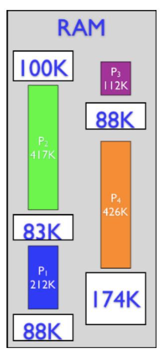


18

Fragmentation

There is enough memory for a 200K process...

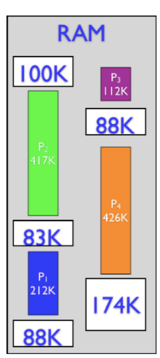
...just not all together



19

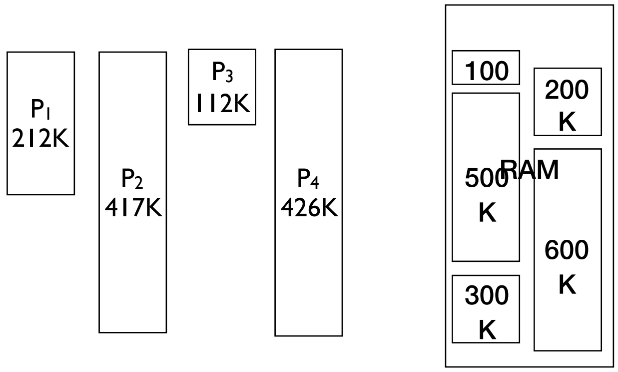
Fragmentation

Little chunks of memory that **cannot be used**



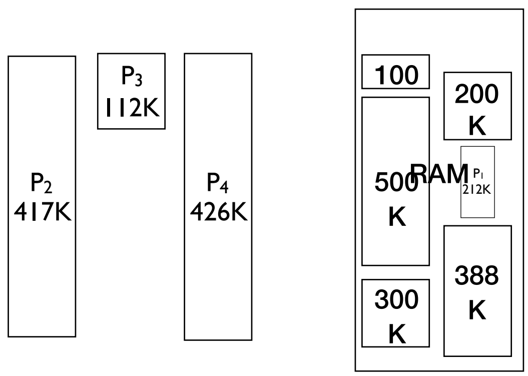
20-1

Worst-Fit Allocation (work)



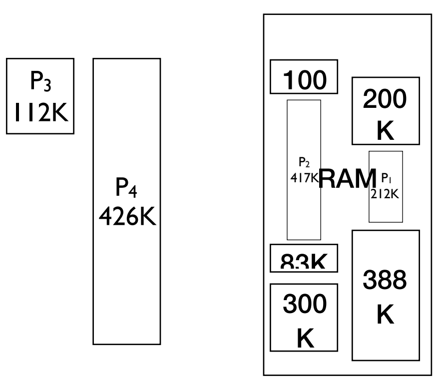
20-2

Worst-Fit Allocation (work)



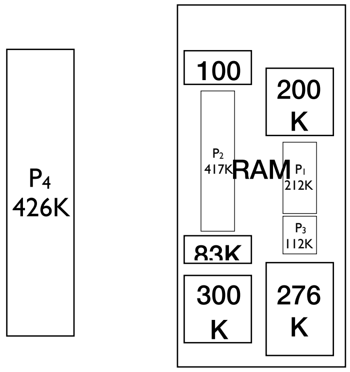
20-3

Worst-Fit Allocation (work)



Worst-Fit Allocation (work)

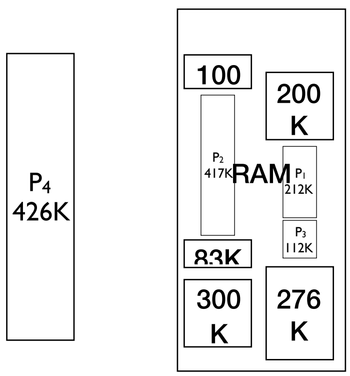
20-4



Worst-Fit Allocation (work)

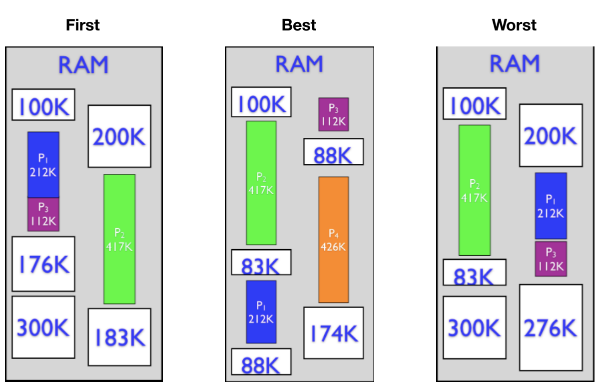
20-5

P4 does not fit!



Comparison

21



22

Contiguous Allocation

Which is the “best”?

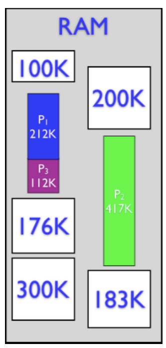
First-Fit - incomplete allocation

Best-Fit - seems good... expensive

Worst-Fit - incomplete allocation

23-1

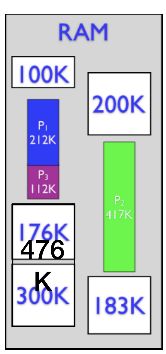
Compaction



Shuffle allocations around to remove small bits of free space... expensive.

23-2

Compaction



Shuffle allocations around to remove small bits of free space... expensive.

24

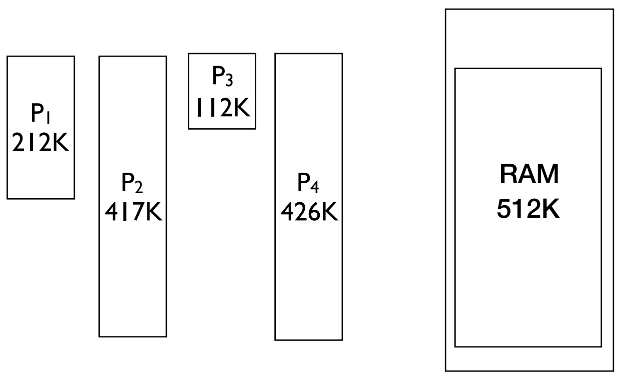
Non-Contiguous Allocation

Paged memory

Segmented memory

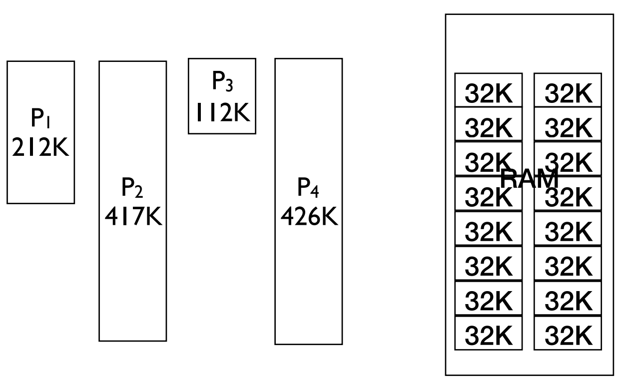
25-1

Paged Memory Allocation



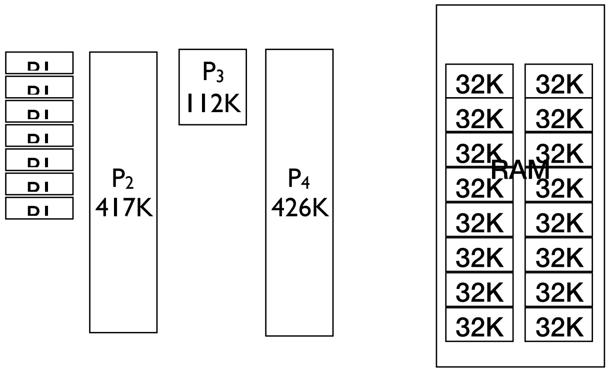
25-2

Paged Memory Allocation



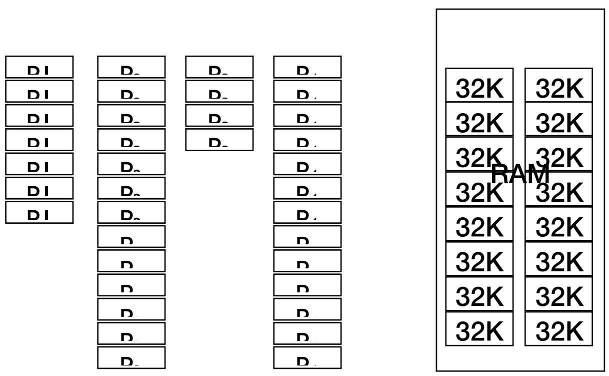
25-3

Paged Memory Allocation



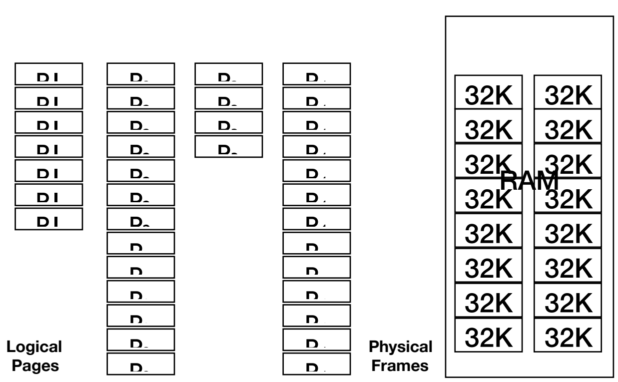
25-4

Paged Memory Allocation



26-1

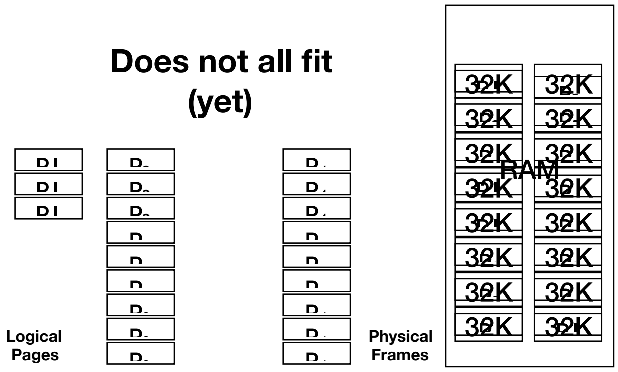
Paged Memory Allocation



26-2

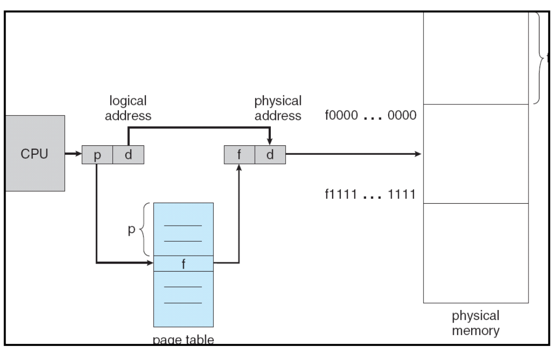
Paged Memory Allocation

Does not all fit (yet)



27

Paging Hardware



28

Paging Details

Page table per process (PCB pointer)

Global free frame list

Shared Pages

Page tables in main memory

29

Fragmentation

Little **chunks of memory that cannot be used**

External Fragmentation (earlier)

Internal Fragmentation

30

Internal Fragmentation

We can only allocate fixed size chunks of memory (frames)

...on average 1/2 a page/frame per process is lost to fragmentation (so far).

31

Effective Access Time

To access memory with paging...

we need to access memory **twice!**

```
EAT = 2 * memory speed

EAT = 2 * 100ns = 200ns = 200ns
```

TLB (cache)

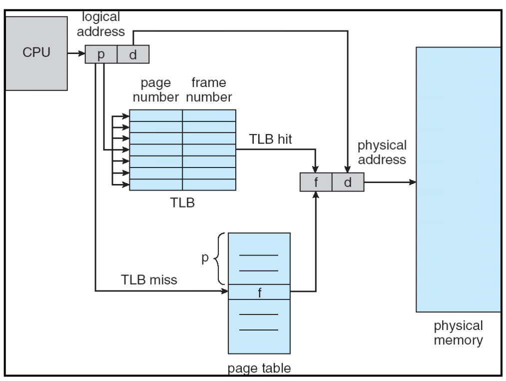
32

The **Translation Look-aside Buffer**

A **cache** specifically for page table entries

TLB Hardware

33



TLB Hit Time

34

When we have a TLB **hit…**

EAT = TLB-time + memory-speed

20ns + 100ns = 120ns

35

TLB Miss Time

When we have a TLB **miss…**

EAT = TLB-time + 2 * memory-speed

20ns + 2 * 100ns = 220ns

36-1

Effective Access Time

The TLB hit ratio is key,

```
EAT = (hit * 120ns) + (miss * 220ns)

EAT = (0.8 * 120ns) + (0.2 * 220ns)

     = 140ns

EAT = (0.98 * 120ns) + (0.02 * 220ns)

     = 122ns
```

36-2

Effective Access Time

The TLB hit ratio is key,

```
EAT = (hit * 120ns) + (miss * 220ns)

EAT = (0.8 * 120ns) + (0.2 * 220ns)

     = 140ns

EAT = (0.98 * 120ns) + (0.02 * 220ns)

     = 122ns
```

36-3

Effective Access Time

The TLB hit ratio is key,

```
EAT = (hit * 120ns) + (miss * 220ns)

EAT = (0.8 * 120ns) + (0.2 * 220ns)

     = 140ns

EAT = (0.98 * 120ns) + (0.02 * 220ns)

     = 122ns
```

37-1

Page Tables are Large

2^{32} (4G) logical address space with 2^{12} (4K) pages...

over 1 million pages

4 bytes each = 4MiB page table

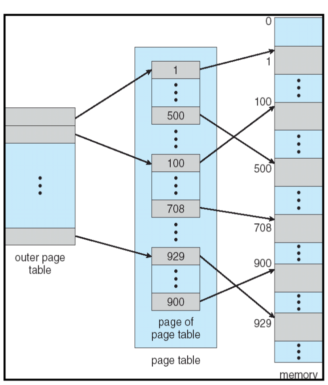
37-2

Page Tables are Large

2^{32} (4G) logical address space with 2^{12} (4K) pages...

over 1 million pages

4 bytes each = 4MiB page table



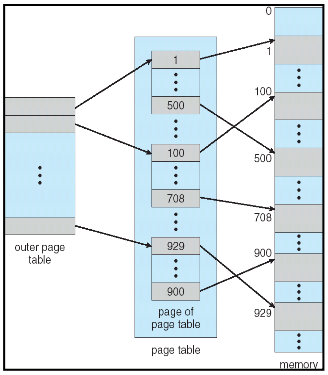
Page Tables are Large

37-3

Outer and Inner pages & tables

page number		page offset
p_1	p_2	d
1	1	1
2	0	0

Outer always in memory
Inner when needed



Segmentation

38

intel Example

39