Paging

Sridhar Alagar

Review: Match the following

Description

1. exactly one process uses RAM at a time
2. add per-process starting location to virtual addr to obtain physical addr
3. dynamic approach that verifies address is in valid range
4. several base+bound pairs per process

Name of approach (covered previous lecture):

Segmentation

Time Sharing

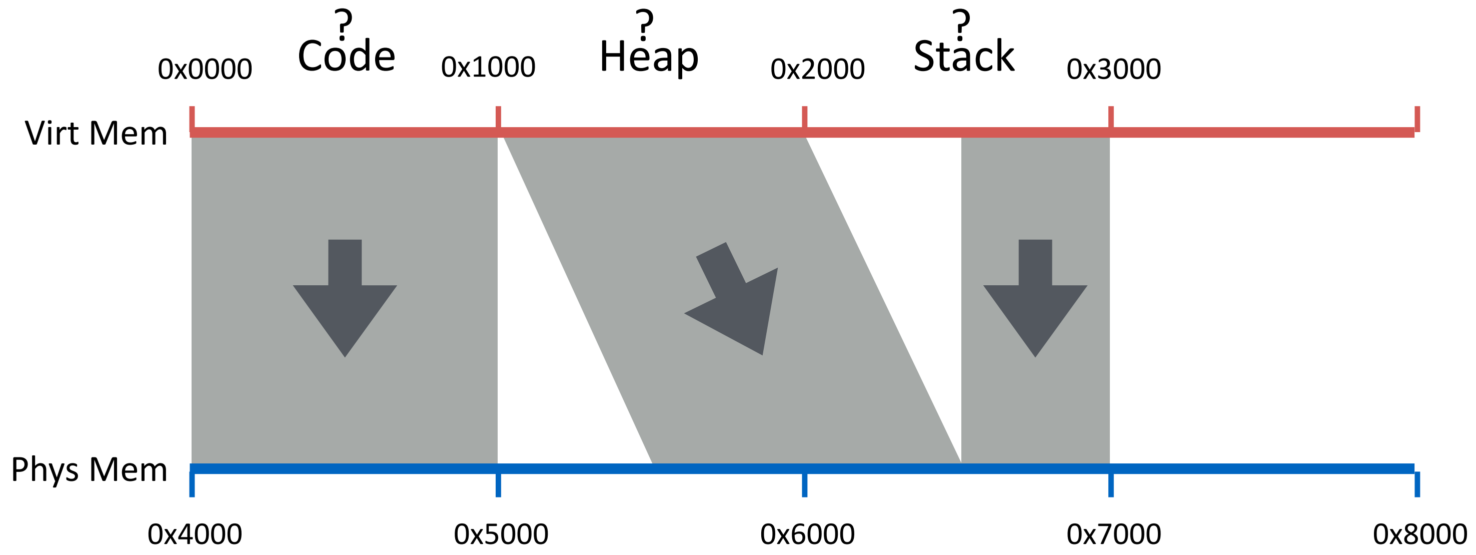
Base

Base+Bounds

Review: Segmentation

Assume 14-bit virtual addresses, high 2 bits indicate segment

Segments:
0=>code
1=>heap
2=>stack.



Where does segment table live?

All registers, MMU

Seg	Base	Bounds
0	0x4000	0xfff
1	0x5800	0xfff
2	0x6800	0x7ff

Review: Translation

```
0x0010: movl   0x1100, %eax
0x0013: addl   $0x3, %eax
0x0019: movl   %eax, 0x1100
```

%eip: 0x0010

Seg	Base	Bounds
0	0x4000	0xfff
1	0x5800	0xfff
2	0x6800	0x7ff

Physical Memory Accesses?

1) Fetch instruction at logical addr 0x0010
   - Physical addr: 0x4010

Exec, load from logical addr 0x1100
   - Physical addr: 0x5900

2) Fetch instruction at logical addr 0x0013
   - Physical addr: 0x4013

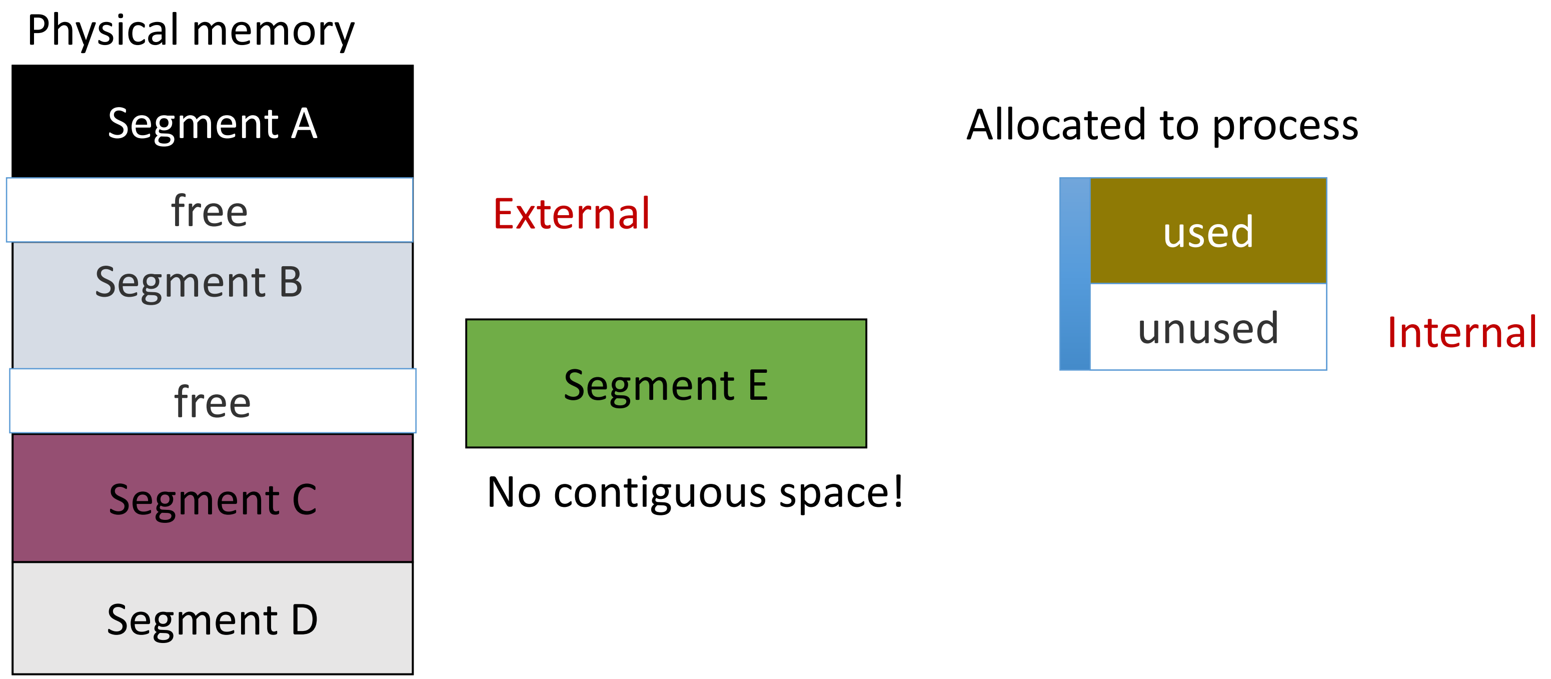
Exec, no load

3) Fetch instruction at logical addr 0x0019
   - Physical addr: 0x4019

Exec, store to logical addr 0x1100
   - Physical addr: 0x5900

Problem with Segmentation

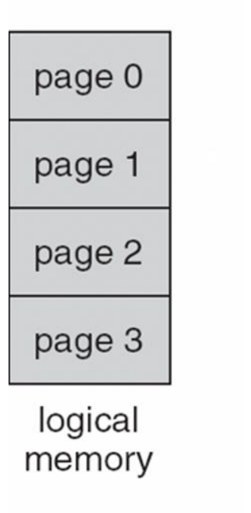
- Fragmentation: Memory that cannot be allocated
  - allocated memory is too big and unused by the process (internal)
  - free memory holes are too small and scattered (external)

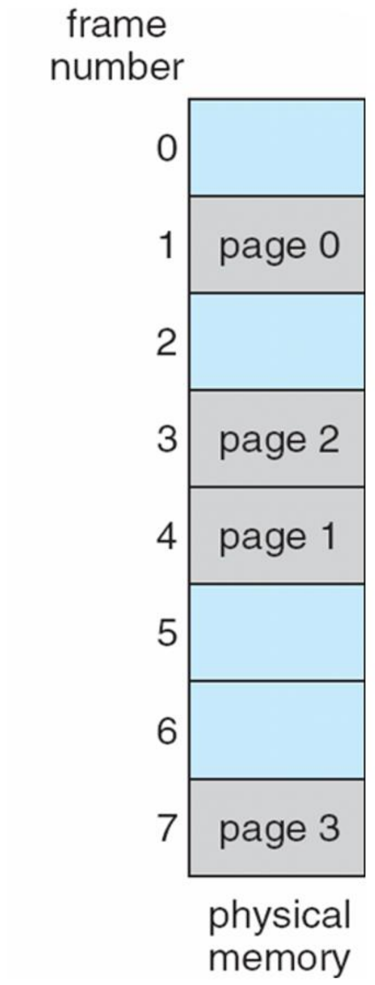


Paging

- Divide physical memory into fixed-sized blocks called physical **frames**
  - size is power of 2, between 512 bytes and 16 Mbytes
- Divide virtual memory into blocks of <u>same</u> size called virtual **pages**
- size of frame = size of a page

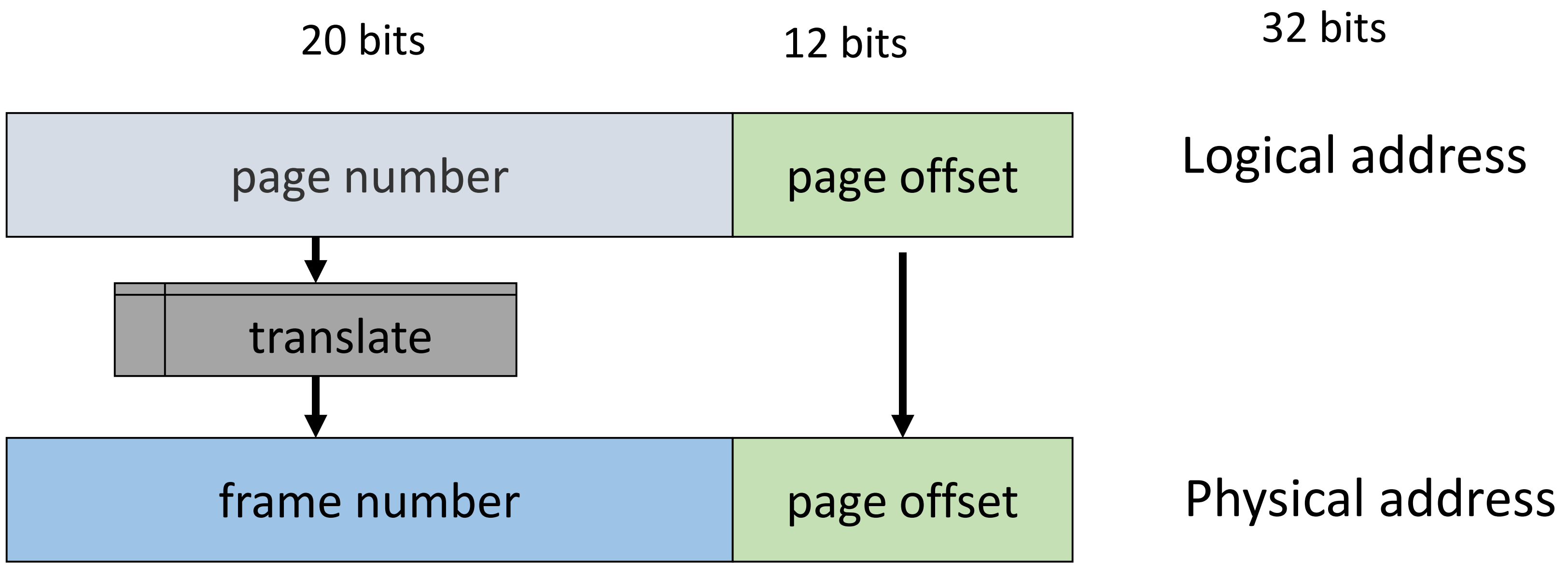
Logical and Physical views





How to translate VA to PA?

- Designate few higher order bits for page number
- Designate the lower order bits for offset within the page



No addition needed; just append bits correctly…

Quiz: Determine lower order bits

Given known page size, how many bits are needed in address to specify offset in page?

Page Size	Low Bits (offset)
16 bytes	4
1 KB	10
1 MB	20
512 bytes	9
4 KB	12

Quiz: Determine higher order bits

Given number of bits in virtual address and bits for offset,
how many bits for virtual page number?

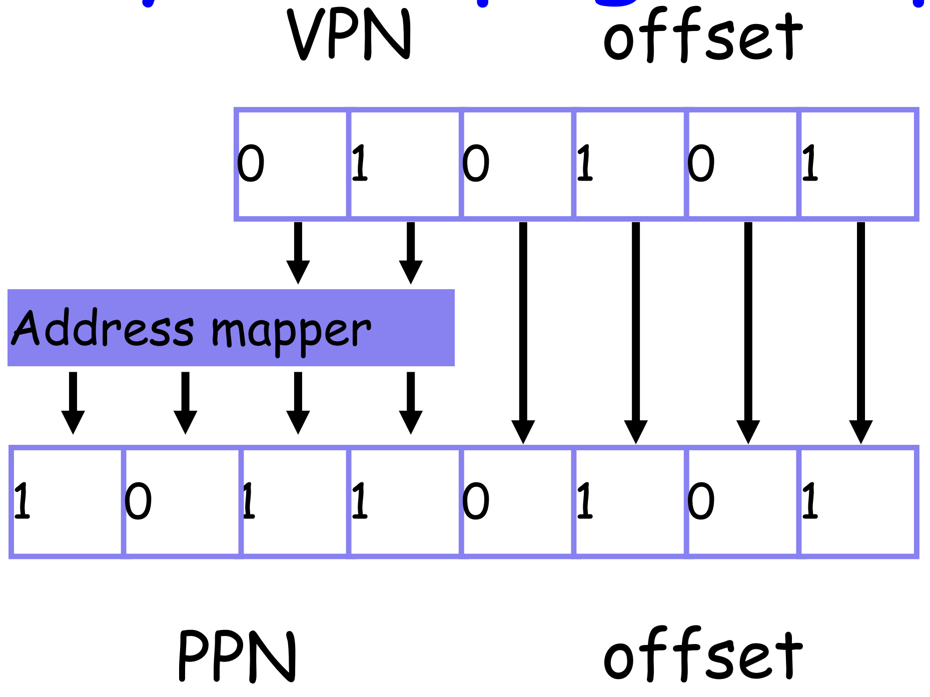
Page Size	Low Bits (offset)	Virt Addr Bits	High Bits (vpn)
16 bytes	4	10	6
1 KB	10	20	10
1 MB	20	32	12
512 bytes	9	16	7
4 KB	12	32	20

Quiz: Address format

Given number of bits for vpn, how many virtual pages can there be in an address space?

Page Size	Low Bits (offset)	Virt Addr Bits	High Bits (vpn)	Virt Pages
16 bytes	4	10	6	64
1 KB	10	20	10	1 K
1 MB	20	32	12	4 K
512 bytes	9	16	7	32
4 KB	12	32	20	1 MB

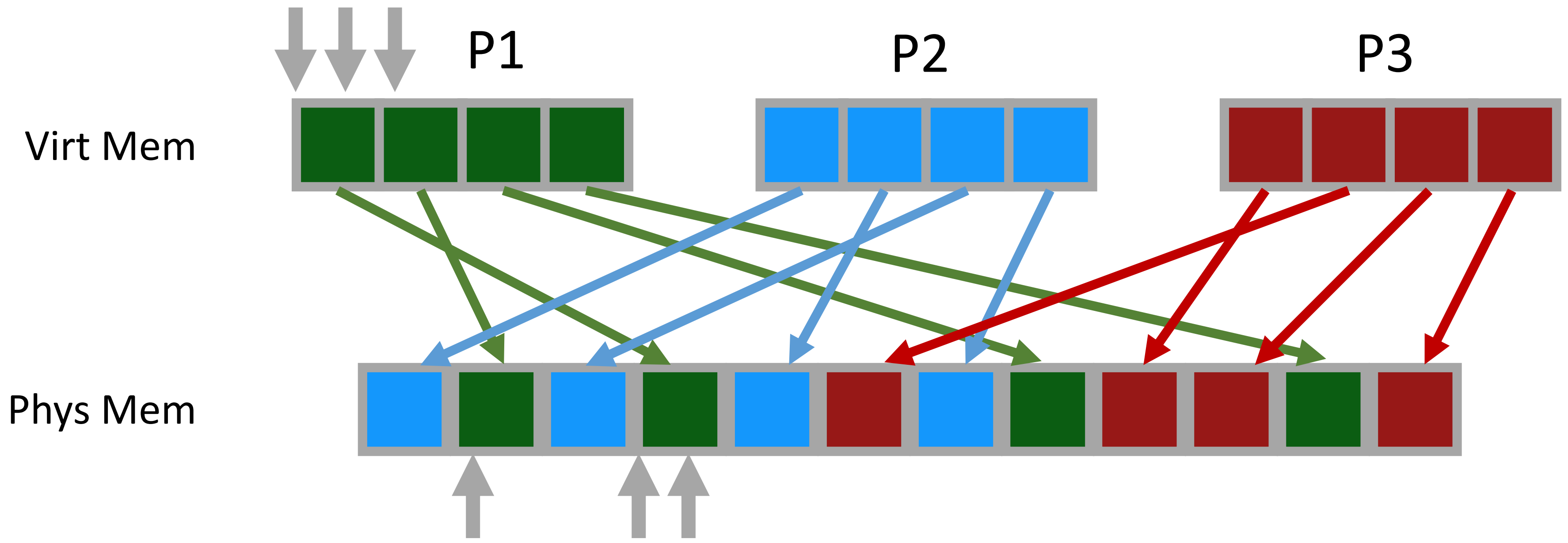
Virtual to Physical page mapping



Number of bits in virtual address format does not need to equal number of bits in physical address format

- How should OS translate VPN to PPN?
  - For paging, OS needs more general mapping mechanism

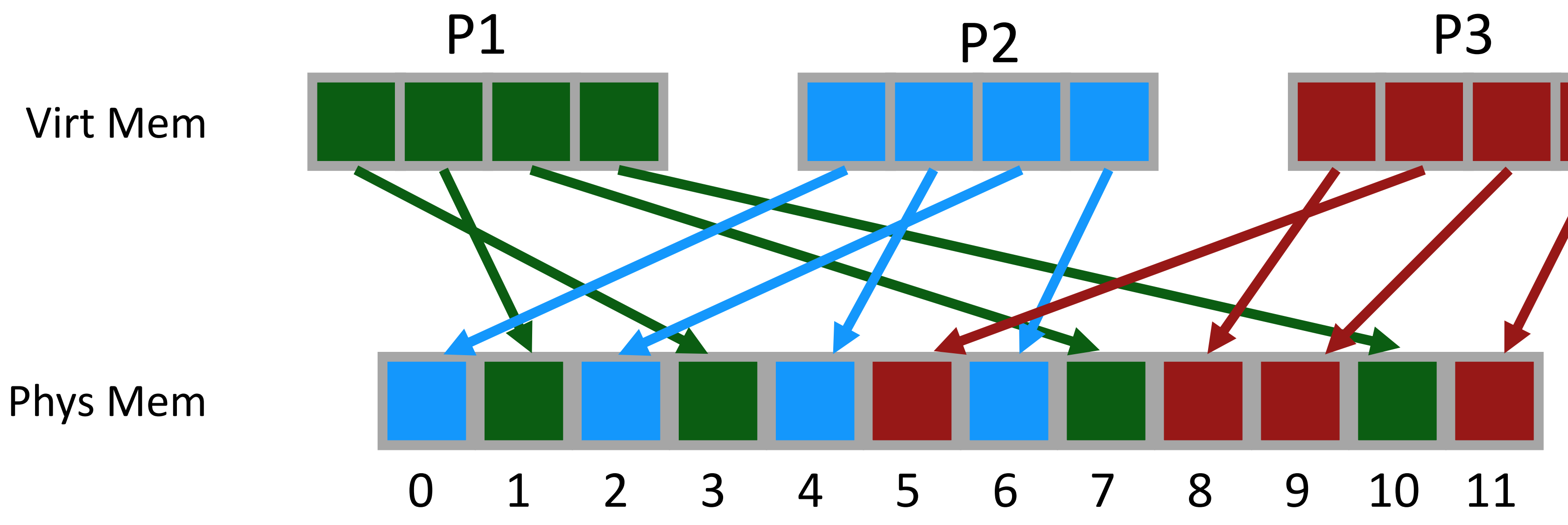
VPN to PPN mapping



What data structure is good for mapping?

Big array: pagetable

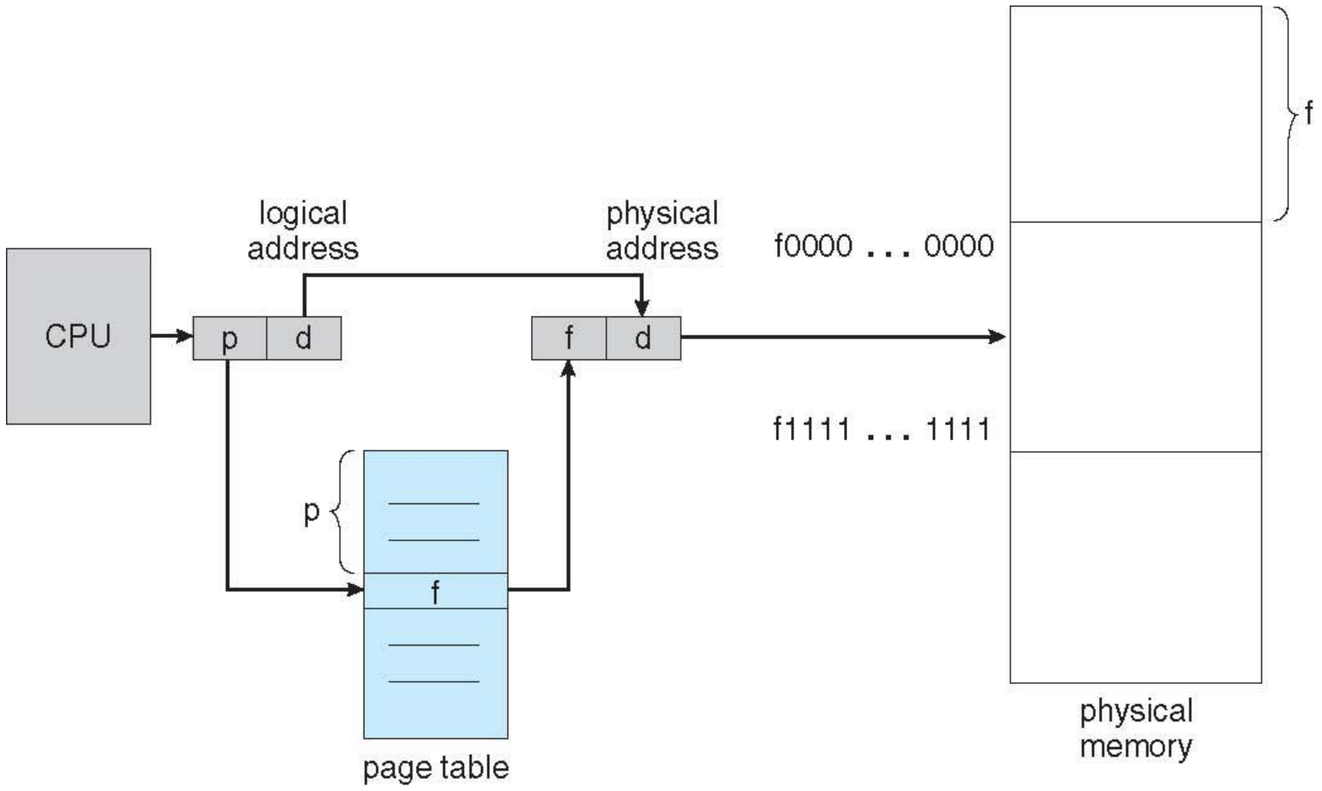
Quiz: Fill the page table?



Page Tables:

P1	P2	P3
3	0	8
1	4	5
7	2	9
10	6	11

Paging hardware



Where will page table reside?

How big is page table?

- How big is a typical page table?
  - assume **32-bit** address space
  - assume 4 KB pages
  - assume 4 byte entries

- Final answer: **4 MB**
  - page table size = Num entries * size of each entry
  - num entries = num virtual pages = 2^(bits for vpn)
  - bits for vpn = 32– number of bits for page offset

    = 32 – lg(4KB) = 32 – 12 = 20
  - num entries = 2^20 = 1 MB
  - page table size = Num entries * 4 bytes = 4 MB

Where to store the page table?

- Store each page table in memory
  - Hardware finds page table base with register (e.g., CR3 on x86)

- What happens on a context-switch?
  - Save old page table base register in PCB of descheduled process
  - Change contents of page table base register to newly scheduled process

Other PTE bits

- What other info is in page table entries besides translation?
  - valid bit
  - protection bits
  - present bit (needed later)
  - reference bit (needed later)
  - dirty bit (needed later)
- Page table entries are just bits stored in memory
  - Agreement between hw and OS about interpretation

Memory access with pages

```
0x0010: movl   0x1100, %eax
0x0013: addl   $0x3, %eax
0x0019: movl   %eax, 0x1100
```

Assume PT is at phys addr 0x5000

Assume PTE's are 4 bytes

Assume 4KB page

How many bits for offset? 12

Simplified view of page table

2
0
80
99

Old: How many mem refs with segmentation?

5 (3 instrs, 2 movl)

Physical Memory Accesses with Paging?

1) Fetch instruction at logical addr 0x0010; vpn?

- Access page table to get ppn for vpn 0
- Mem ref 1: 0x5000
- Learn vpn 0 is at ppn 2
- Fetch instruction at 0x2010 (Mem ref 2)

Exec, load from logical addr 0x1100; vpn?

- Access page table to get ppn for vpn 1
- Mem ref 3: 0x5004
- Learn vpn 1 is at ppn 0
- Movl from 0x0100 into reg (Mem ref 4)

Page table is slow!!! Doubles memory references

Advantages of paging

- No external fragmentation
  - Any page can be placed in any frame in physical memory
- Fast to allocate and free
  - Alloc: No searching for suitable free space
  - Free: Doesn't have to coalesce with adjacent free space
  - Just use bitmap to show free/allocated page frames
- Simple to swap-out portions of memory to disk (later lecture)

Disadvantages of paging

- Internal fragmentation
  - wasted memory grows with larger pages

- Additional memory reference to page table
  - Page table stored in memory
  - MMU stores only PTBR

- Page table size can be high