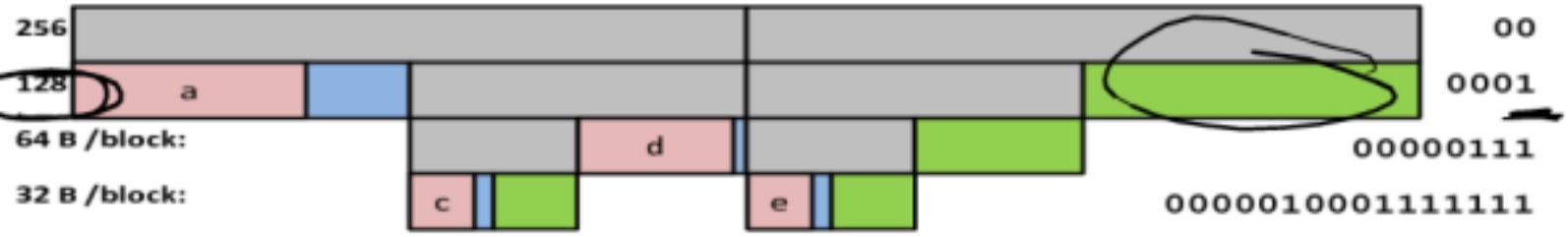
Buddy System and Segmentation

CS 241

More Buddy System

malloc(100)

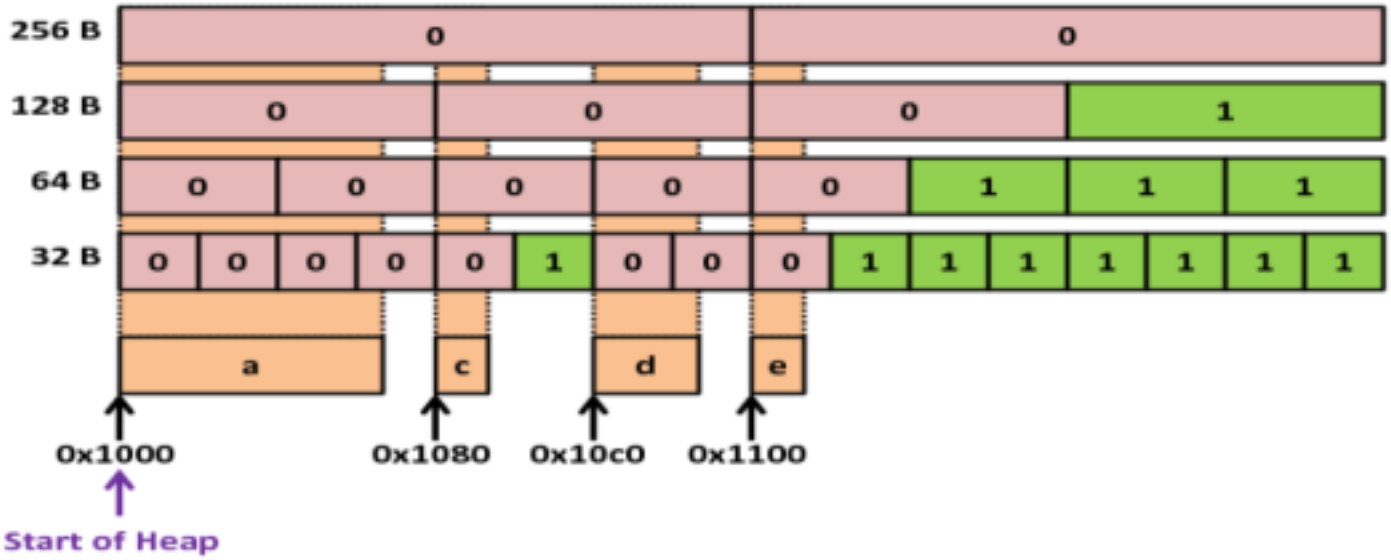
- Last Lecture:



- Optimization:
  - Multiple levels may each have their own bit masks.
    - **Advantage**: Speeds up finding open space
    - **Disadvantage**: Increased overhead

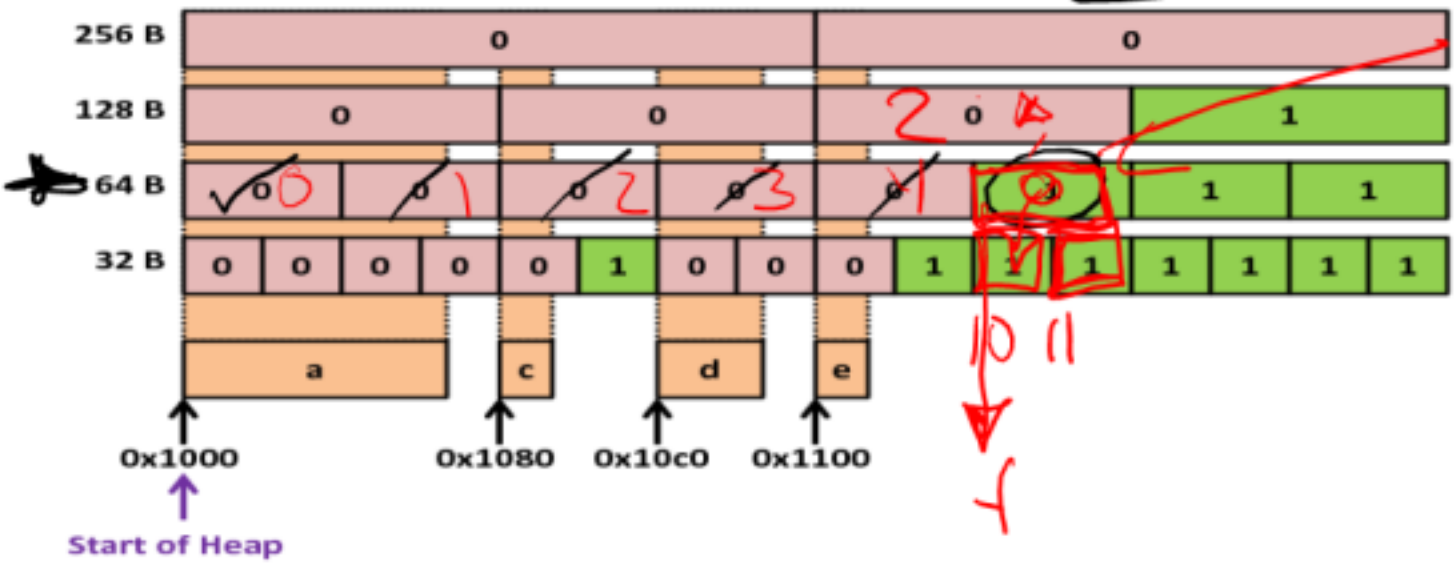
Buddy System

- Represented as bitmaps:



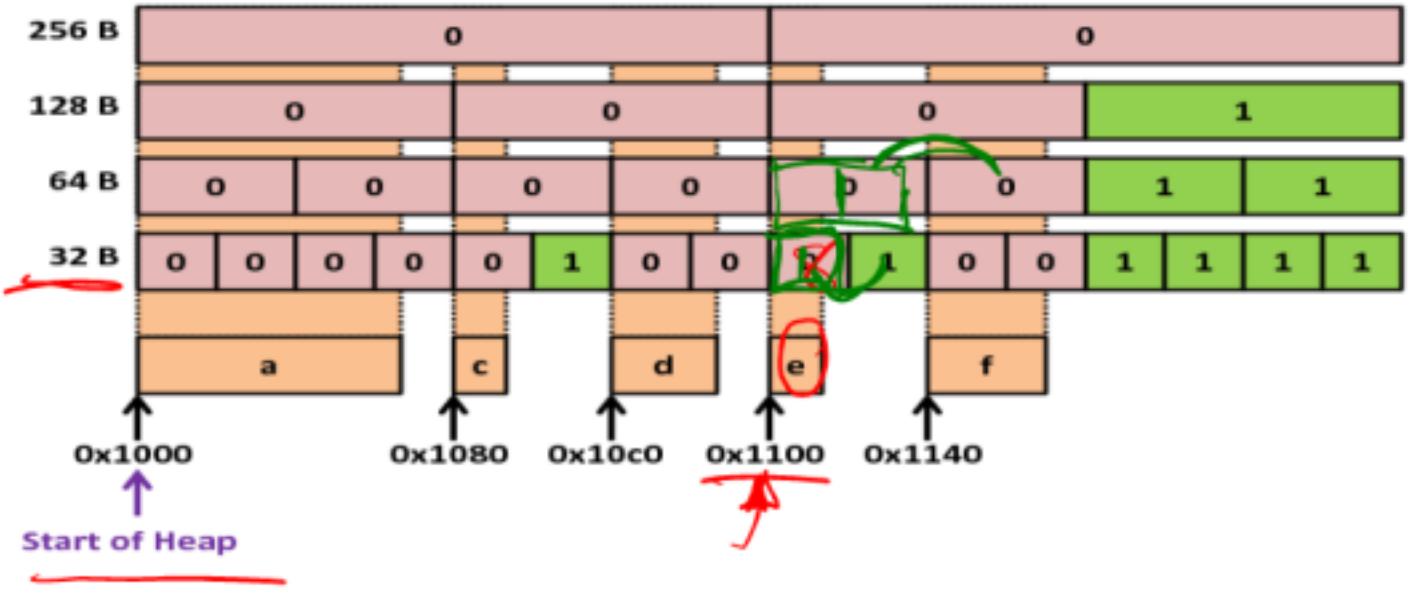
Buddy System

- Request: `f = malloc(56)` → 00111000



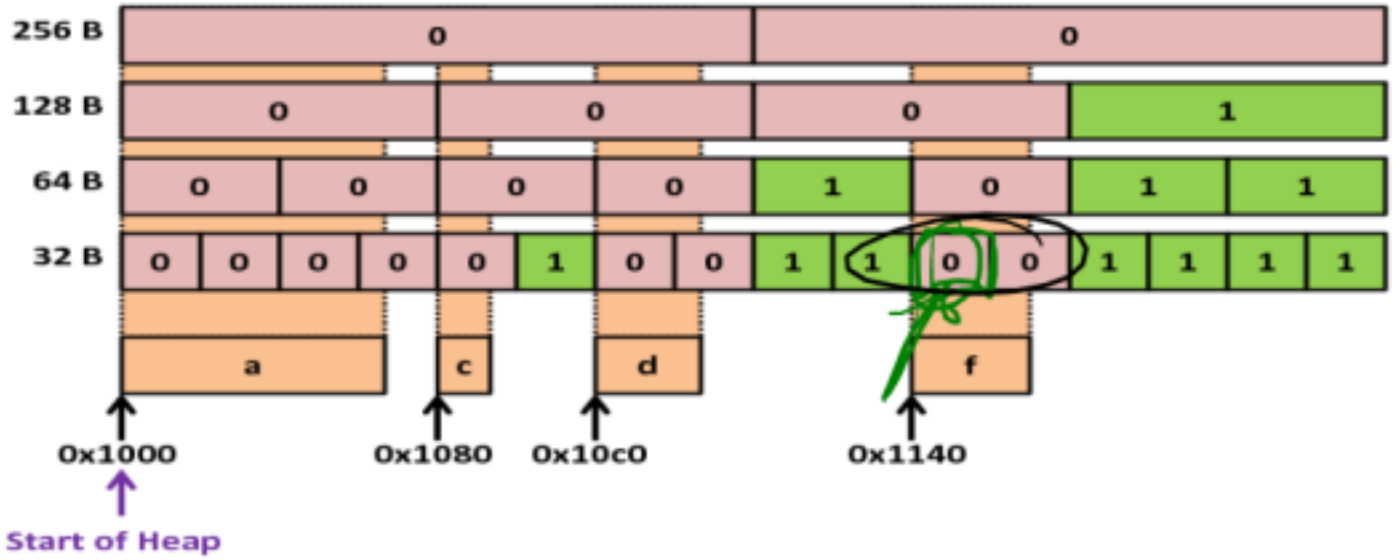
Buddy System

- Request: `free(e)`



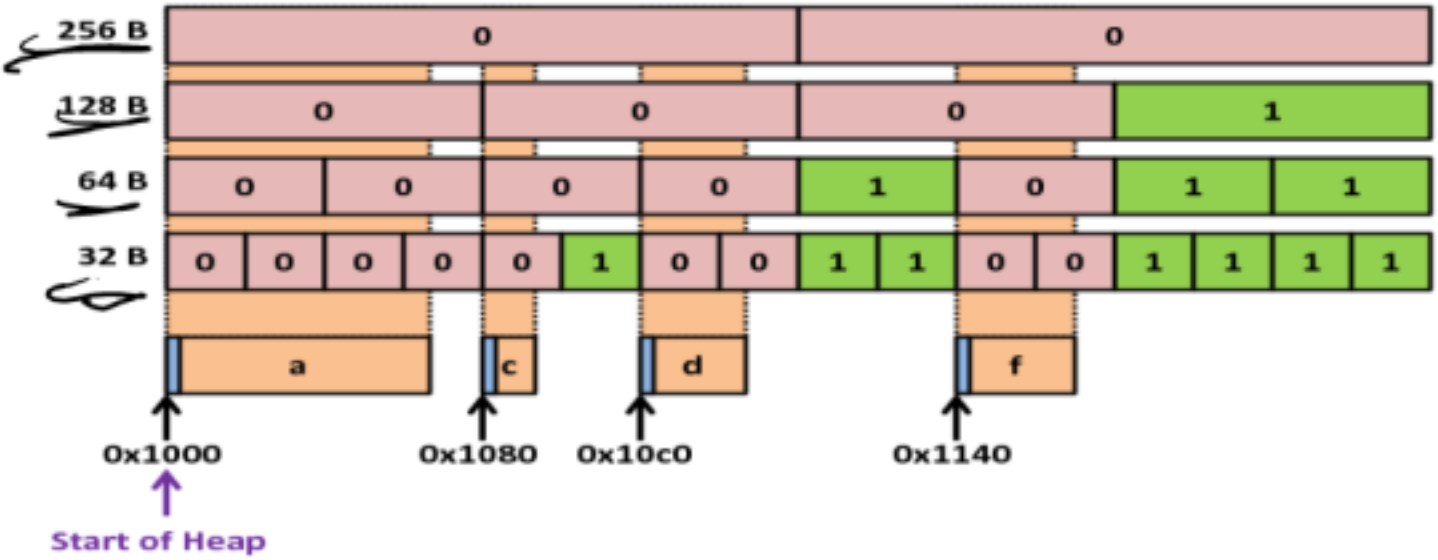
Buddy System

- Request: `free(f)`



Buddy System

- **Buddy System requires knowledge of level/size of allocation.**

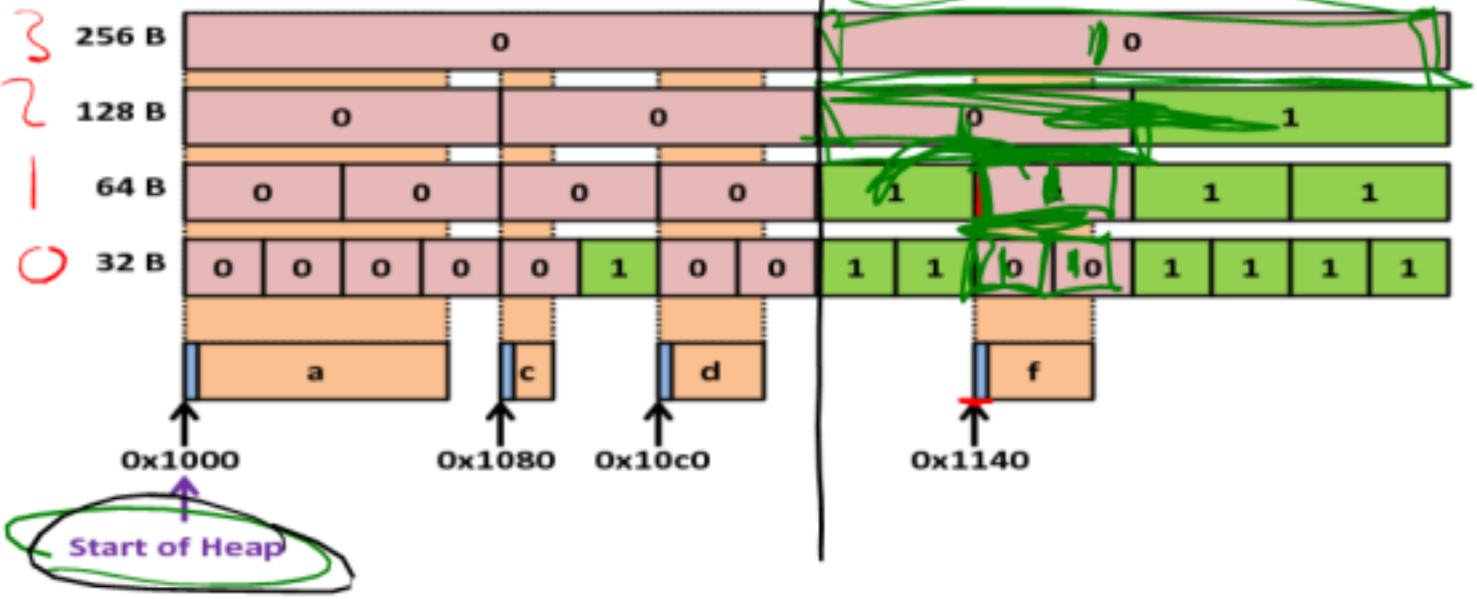


$32 = 100000$

$64 = 1000000$

Buddy System

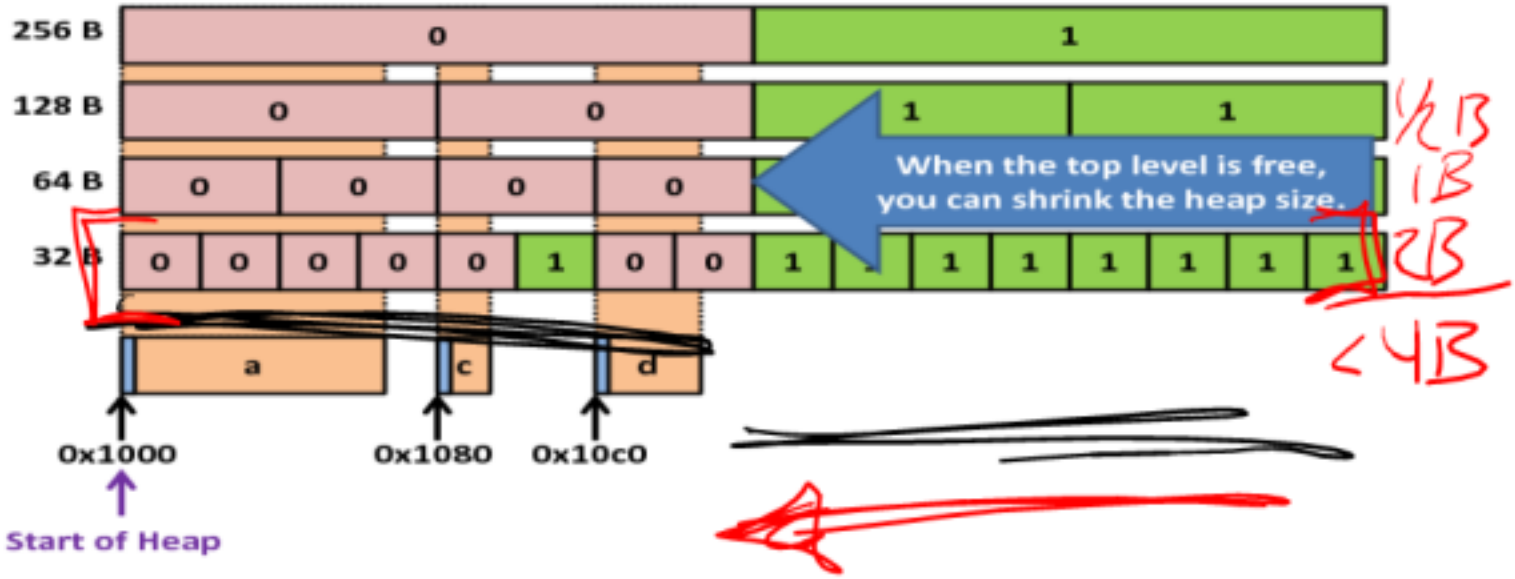
- Request: `free(f)`



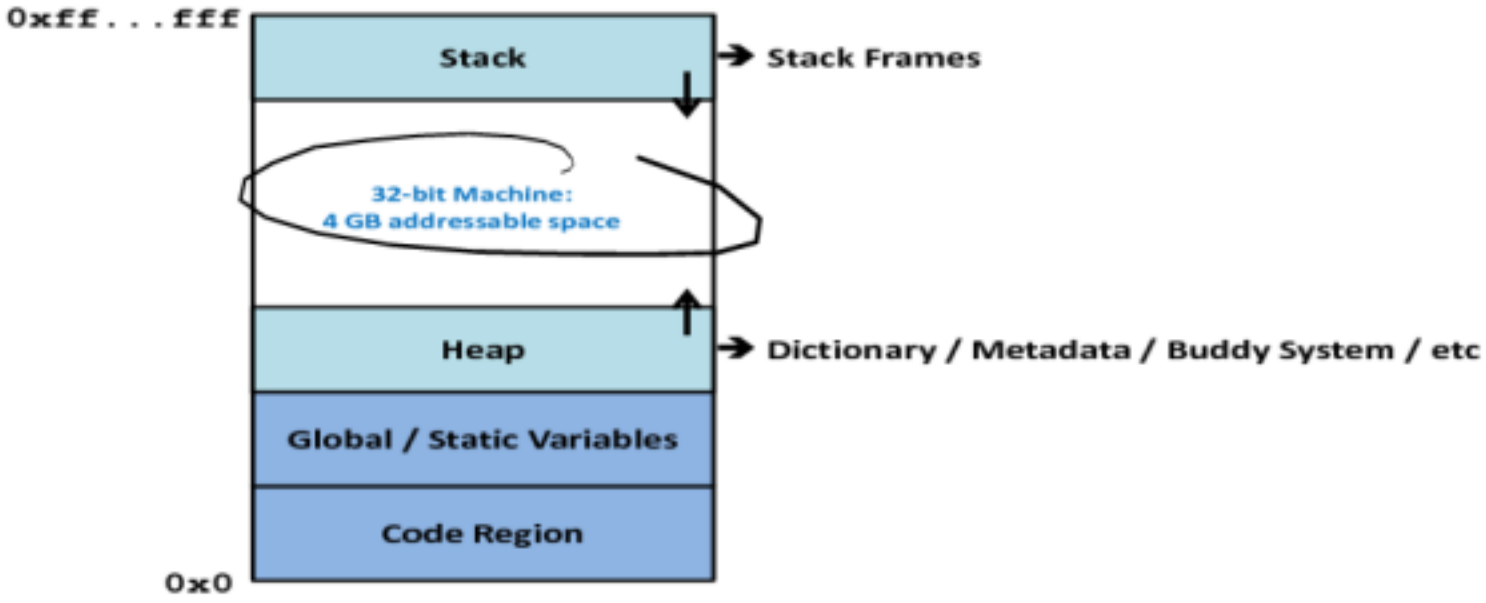
Buddy System

- Result:

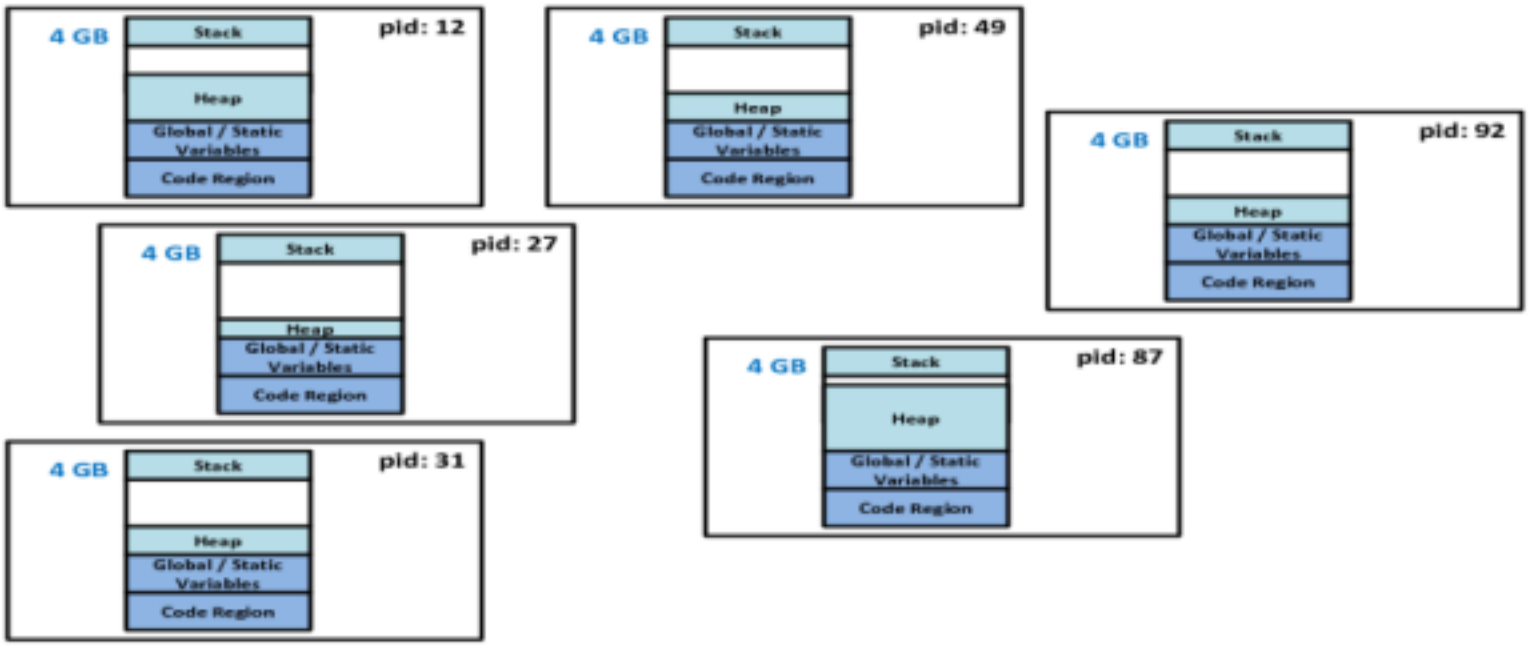
sbrk(0)



Up to now...


0xff...fff
Stack
Stack Frames
32-bit Machine:
4 GB addressable space
Heap
Dictionary / Metadata / Buddy System / etc
Global / Static Variables
Code Region
0x0

Every process has their own virtual memory space..



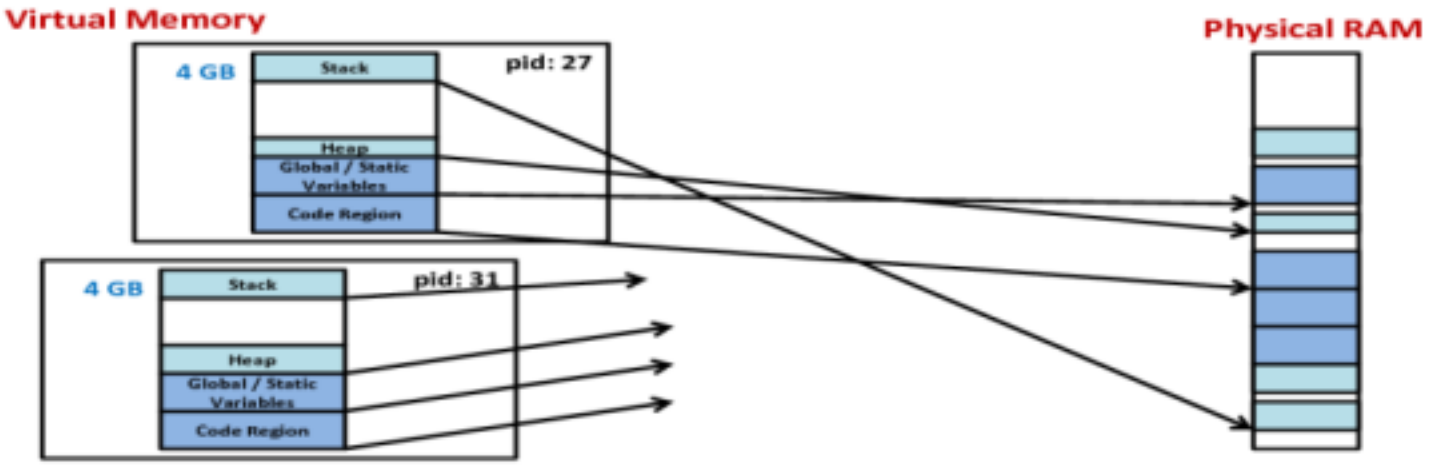
Virtual Memory Translation

- Every virtual address **must** to be translated into a physical address before data is accessed.

- **int k = j + 3;**
  - To look up the data stored in virtual address **&j**...
    1. Translate virtual address **&j** into a physical address
    2. Fetch that data from the location in RAM
    3. Return the data to the process

Solution #1: Segmentation

- **Segmentation**: Place each logical piece of a program's memory into RAM at an offset.

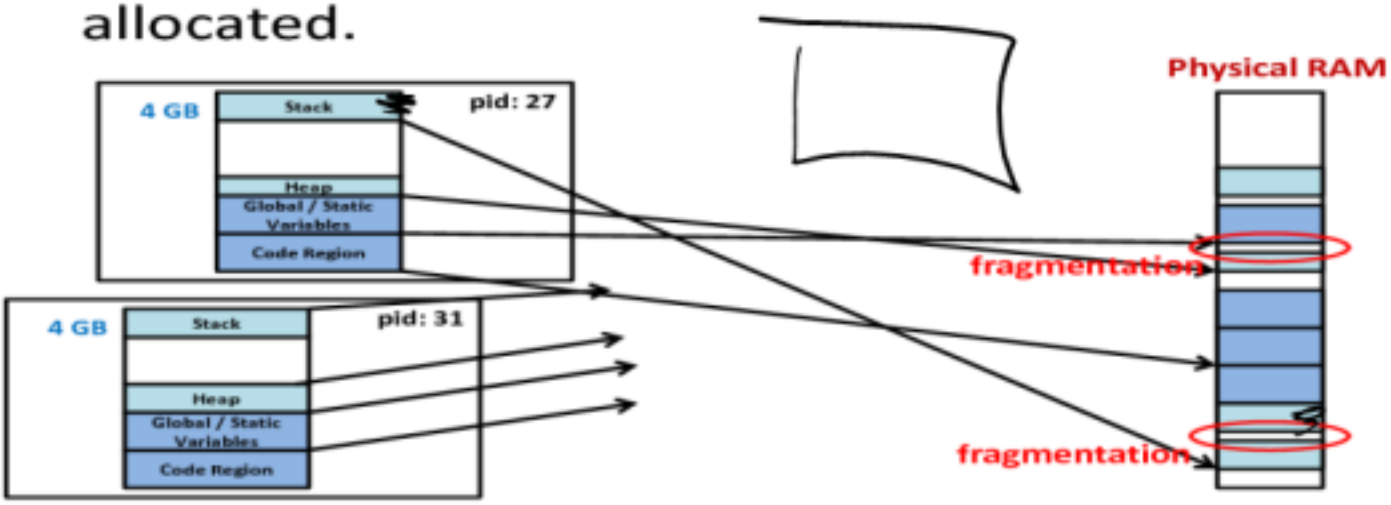


Memory Segmentation

- **Advantages?**
  - Easy
  - Fast
- **Disadvantages?**
  - Limited RAM

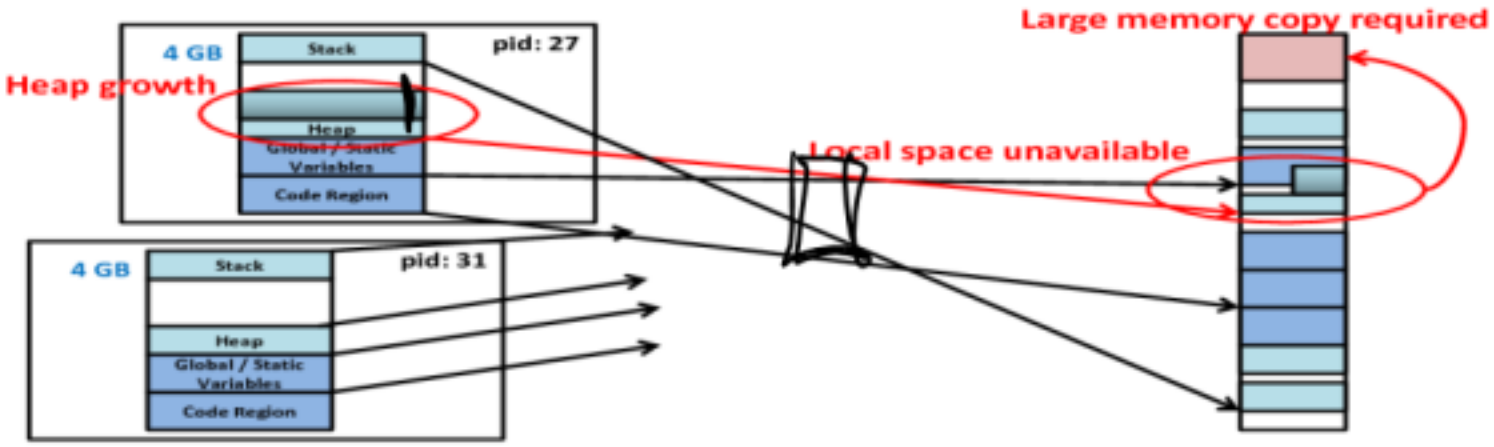
Segmentation Disadvantage #1

- **External Fragmentation:** Physical RAM will develop small holes over time, unable to be allocated.



Segmentation Disadvantage #2

- **Storage Growth:** When memory usage grows, physical RAM must be allocated. If space is not available nearby, a large copy is required.



Paging and Page Replacement

CS 241

Paging

- Motivation:
  - Segmentation maps entire regions of memory to RAM, what happens if we scale this down?

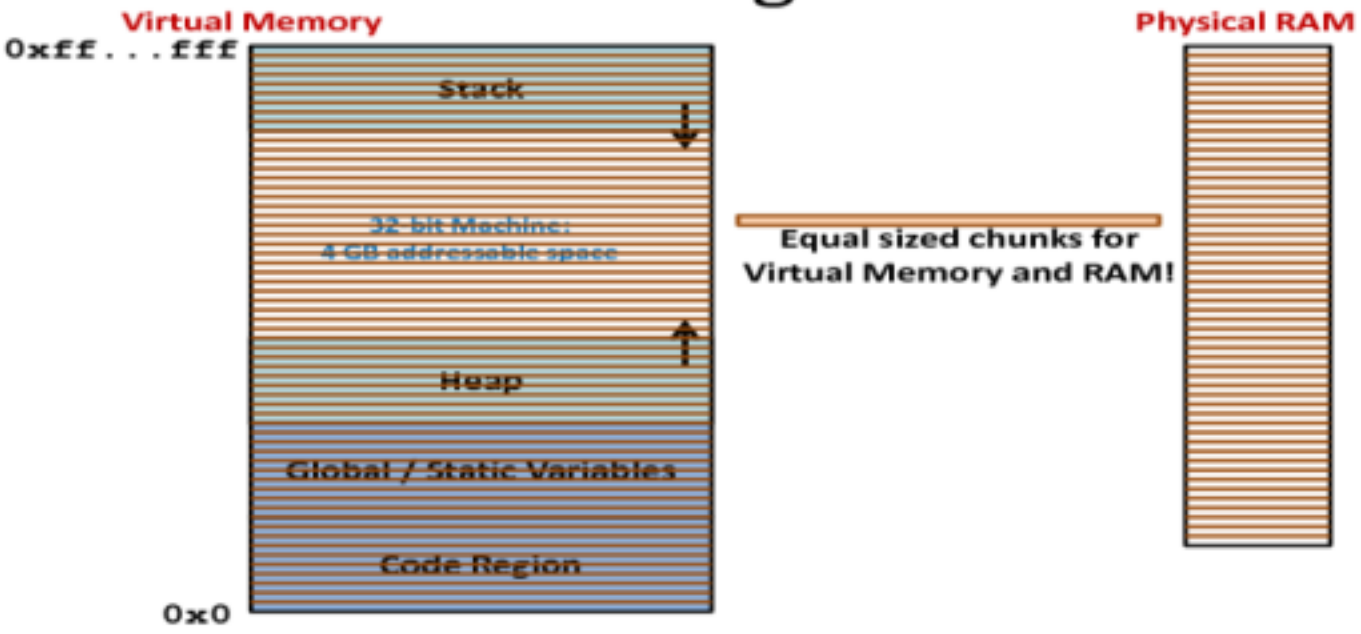
- Answer:
  - Paging

Page

- Divide up **both** the **virtual address space** and the **physical RAM** into equal size chunks.
  - Each of these equal size chunks are called a **page**.

- Pages must always be of size $\mathbf{2^N}$.
  - Eg: 1 KB, 2 KB, 4 KB, 8 KB, etc.

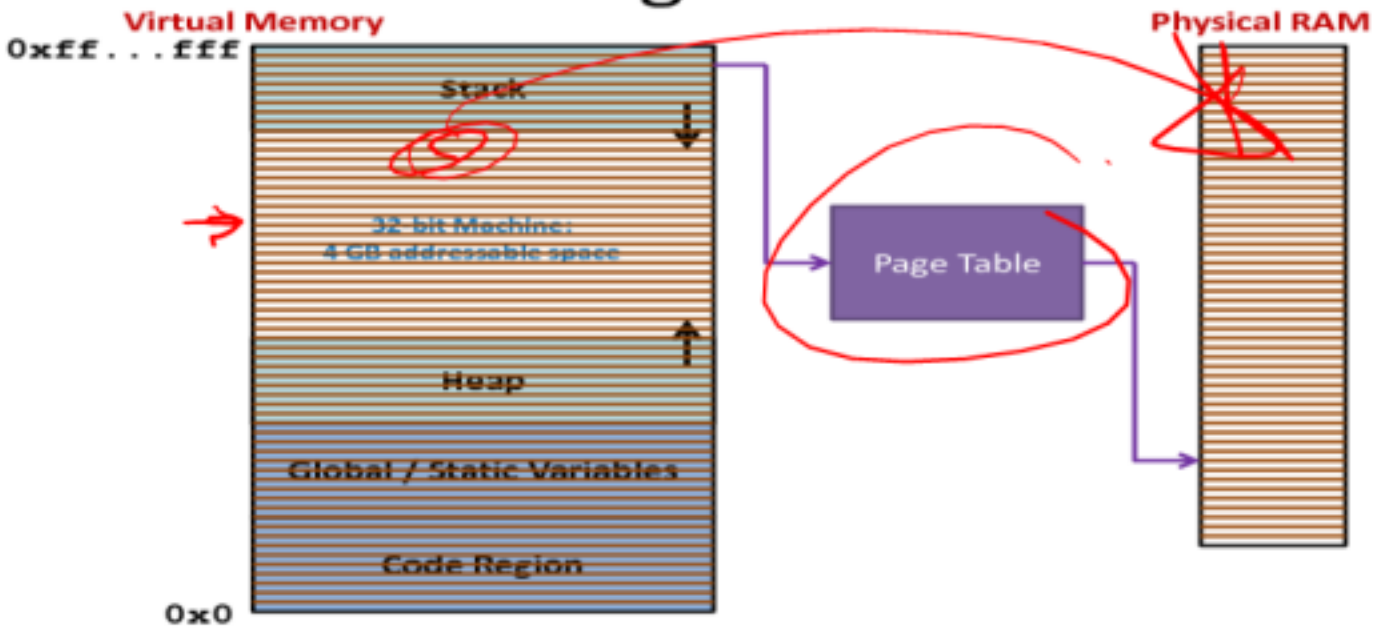
Page


Virtual Memory
Physical RAM
0xff...fff
Stack
32 bit Machine:
4 GB addressable space
Heap
Global / Static Variables
Code Region
0x0
Equal sized chunks for
Virtual Memory and RAM!

Page Table

- A **page table** provides a mapping between each **virtual page** and the **physical page** in RAM.

Page Table


Virtual Memory
0xff . . . fff
Stack
32 bit Machine:
4 GB addressable space
Heap
Global / Static Variables
Code Region
0x0
Page Table
Physical RAM

How does the Page Table work?

- Given a size of a page, you can always divide a virtual memory address into two pieces:
  - **Page Number / Page Table Index**: What virtual page are we on?
  - **Page Offset**: How far are we inside the page?

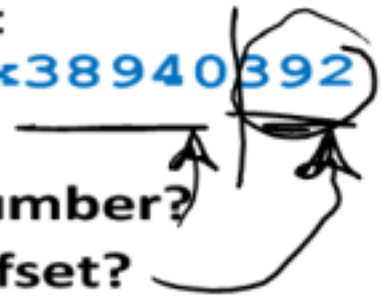
How does the Page Table work?

- Example
  - **4 KB** pages

$= 2^{12}$ BYTES $\rightarrow$ 12 BITS

- Virtual Address:

0x38940392



- **Virtual Page Number?**
- **Virtual Page Offset?**

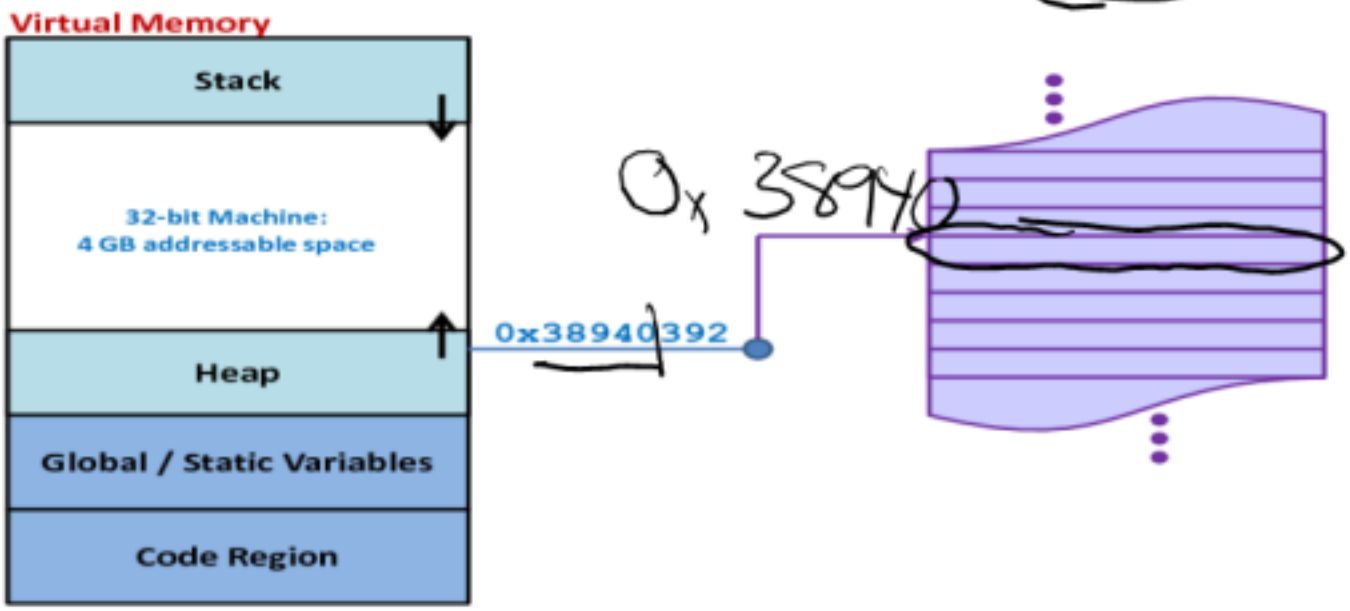
How does the Page Table work?

- Example
  - **256 B** pages → $2^8 = 8$ BITS
- Virtual Address:

  0x38940392
- **Virtual Page Number?**
- **Virtual Page Offset?**

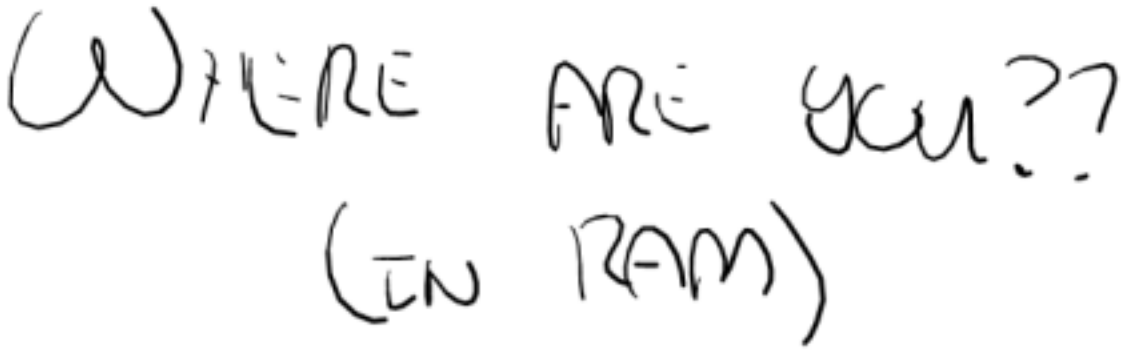
Page Table

Assume 4 KB pages.



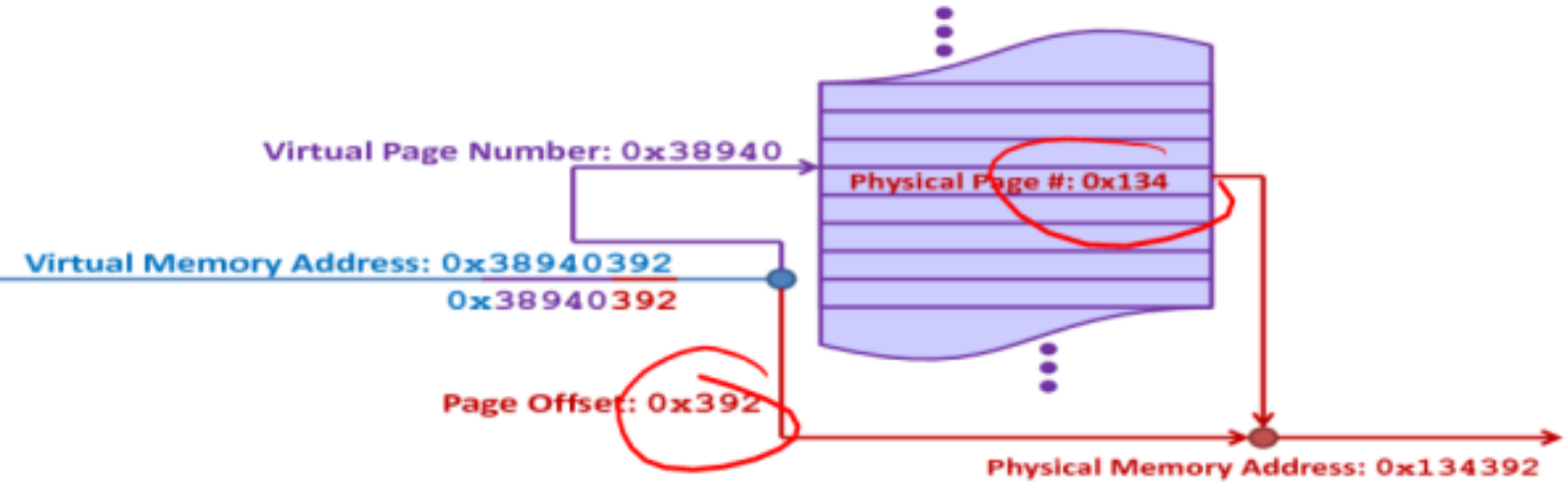
Page Table Entries

- What is the most important thing in each entry in the Page Table, or each **Page Table Entry (PTE)**?



Page Table

Assume 4 KB pages.



Modern Systems...

- A modern economy laptop may have 2 GBs of RAM...
  - ...may not be enough RAM for all those programs you run! :(

Resident Bit

- **Solution**: Allow the page table to translate both to RAM and to a hard drive!

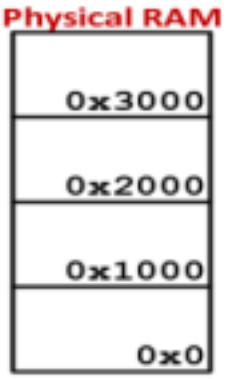
- How?
  - **Resident Bit**: A single bit in every page table entry.
    - **1**: **Resident in RAM**, use the physical page number to translate the virtual address to a physical address.
    - **0**: Not resident in RAM, but it is **on disk**. Instead of a physical page number, a disk page number of the data.

A Really Simple System...

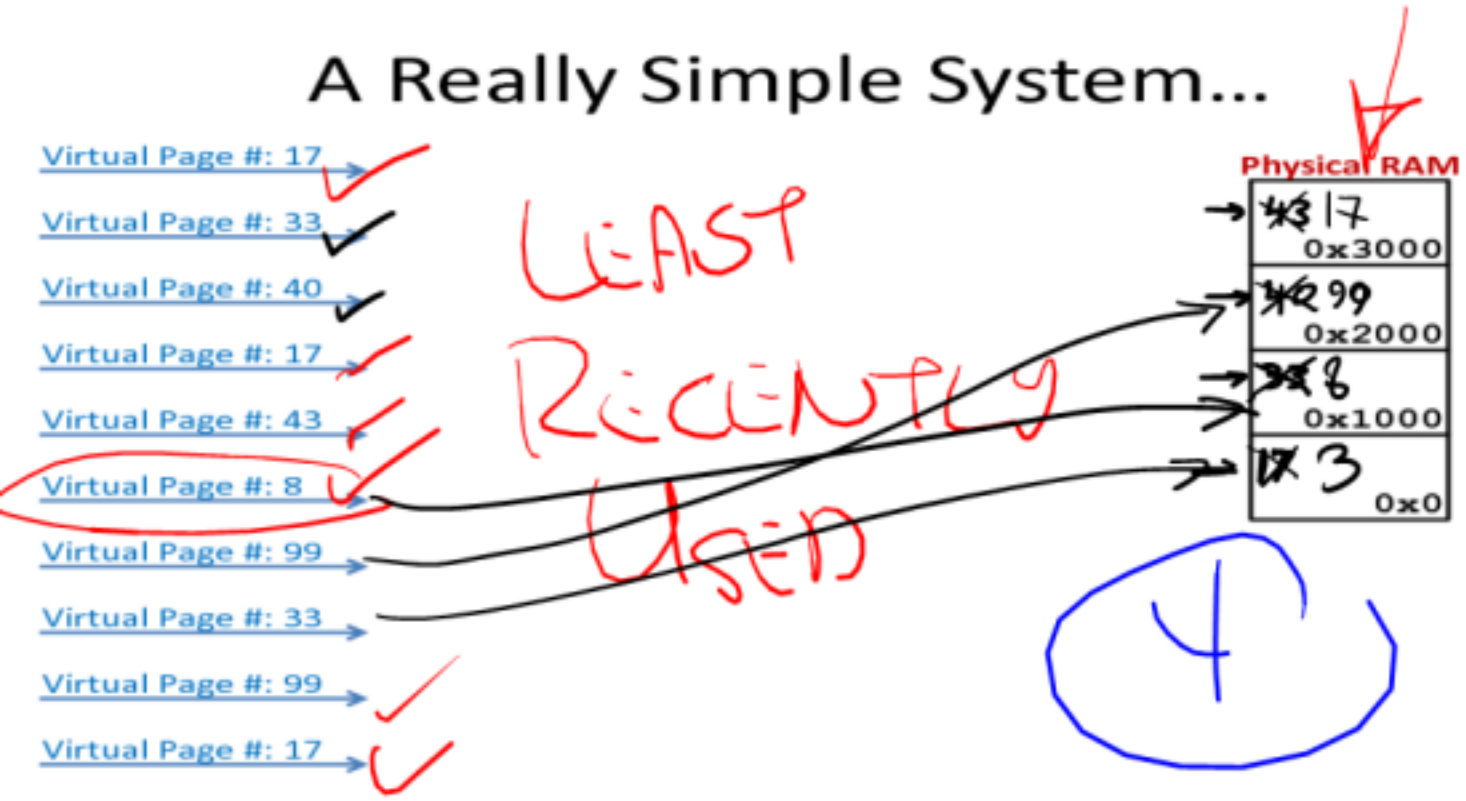
- For an example, we have a system with some really simple properties:
  - 4 KB pages
  - Only 16 KB of RAM
    - Eg: Only 4 total pages.

- **Q**: How many bits are in a physical memory address?

A Really Simple System...



A Really Simple System...

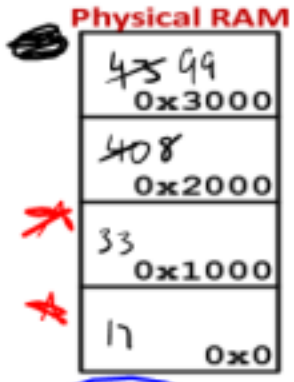


A Really Simple System...

Virtual Page #: 17
Virtual Page #: 33
Virtual Page #: 40
Virtual Page #: 17
Virtual Page #: 43
Virtual Page #: 8
Virtual Page #: 99
Virtual Page #: 33
Virtual Page #: 99
Virtual Page #: 17

OPTIMAL

- VERY HARD TO PREDICT



2

A Really Simple System...

A Really Simple System...

Virtual Page #: 17
Virtual Page #: 33
Virtual Page #: 40
Virtual Page #: 17
Virtual Page #: 43
Virtual Page #: 8
Virtual Page #: 99
Virtual Page #: 33
Virtual Page #: 99
Virtual Page #: 17

Most Recently Used

Physical RAM

43 8 99	0x3000
40	0x2000
33	0x1000
17	0x0

A Really Simple System...

A Really Simple System...

Virtual Page #: 17 →

Virtual Page #: 33 →

Virtual Page #: 40 →

Virtual Page #: 17 →

Virtual Page #: 43 →

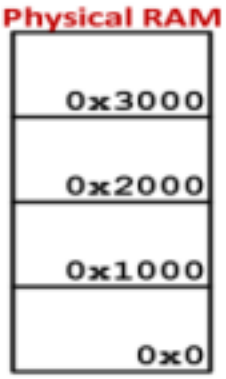
Virtual Page #: 8 →

Virtual Page #: 99 →

Virtual Page #: 33 →

Virtual Page #: 99 →

Virtual Page #: 17 →



A Really Simple System...

Virtual Page #: 17 →

Virtual Page #: 33 →

Virtual Page #: 40 →

Virtual Page #: 17 →

Virtual Page #: 43 →

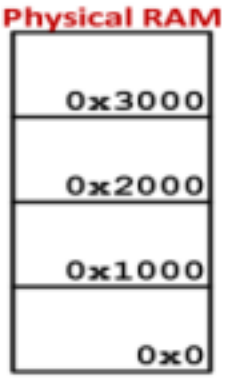
Virtual Page #: 8 →

Virtual Page #: 99 →

Virtual Page #: 33 →

Virtual Page #: 99 →

Virtual Page #: 17 →



A Really Simple System...

Virtual Page #: 17 →

Virtual Page #: 33 →

Virtual Page #: 40 →

Virtual Page #: 17 →

Virtual Page #: 43 →

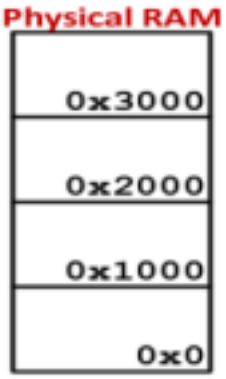
Virtual Page #: 8 →

Virtual Page #: 99 →

Virtual Page #: 33 →

Virtual Page #: 99 →

Virtual Page #: 17 →



A Really Simple System...

Virtual Page #: 17 →

Virtual Page #: 33 →

Virtual Page #: 40 →

Virtual Page #: 17 →

Virtual Page #: 43 →

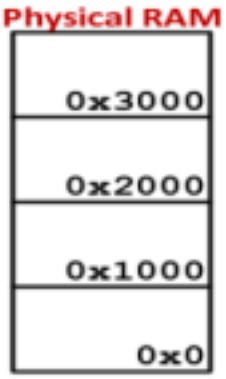
Virtual Page #: 8 →

Virtual Page #: 99 →

Virtual Page #: 33 →

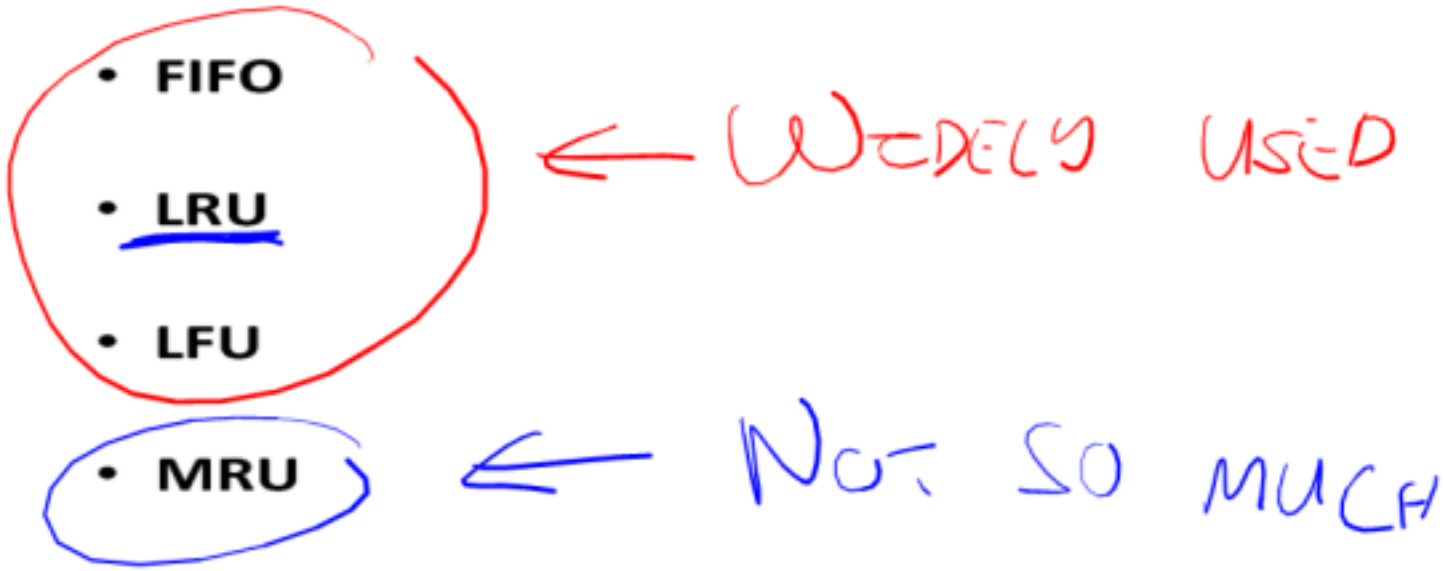
Virtual Page #: 99 →

Virtual Page #: 17 →



Five Page Replacement Algorithms

- **Optimal**
- **FIFO**
- **LRU**
- **LFU**
- **MRU**



Five Page Replacement Algorithms

- **Optimal**
- **FIFO**
- **LRU**
- **LFU**
- **MRU**