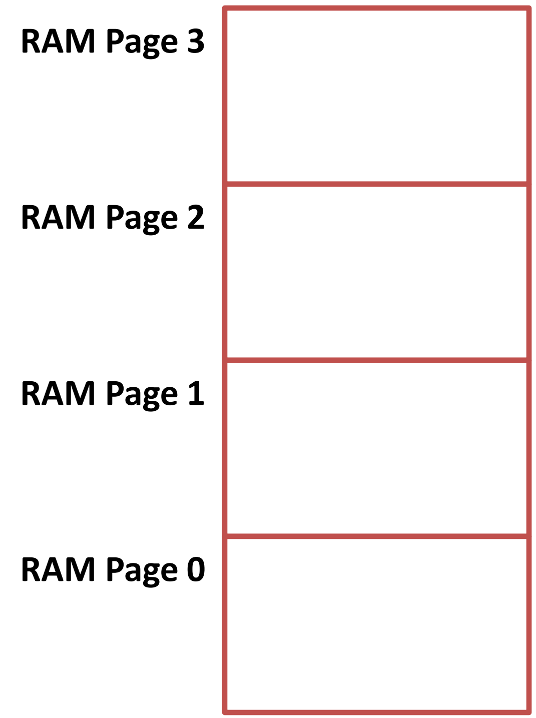
# Paging III

CS 241

Sept. 18, 2013

```
int subtract(int a, int *b) {
     int c = a - *b;
     return c;
}

int add(int a, int *b) {
     int c = a + *b;
     return c;
}

void main() {
     int a = 4;
     int *b = malloc(sizeof(int));
     *b = 7;
     int c = add(a, b);
     int d = subtract(c, b);
}
```

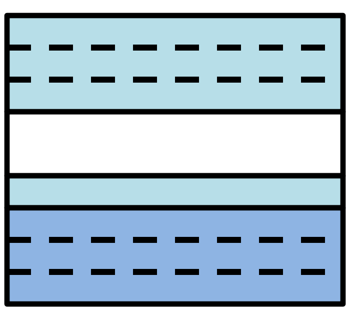
## RAM



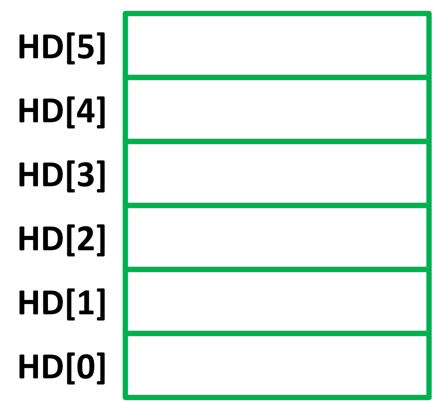

## Page Tabe

| | RES | DIRT | RW | NX | REF | PAGE |
|---|---|---|---|---|---|---|
| [100] | | | | | | |
| [99] | | | | | | |
| [98] | | | | | | |
| [3] | | | | | | |
| [2] | | | | | | |
| [1] | | | | | | |
| [0] | | | | | | |

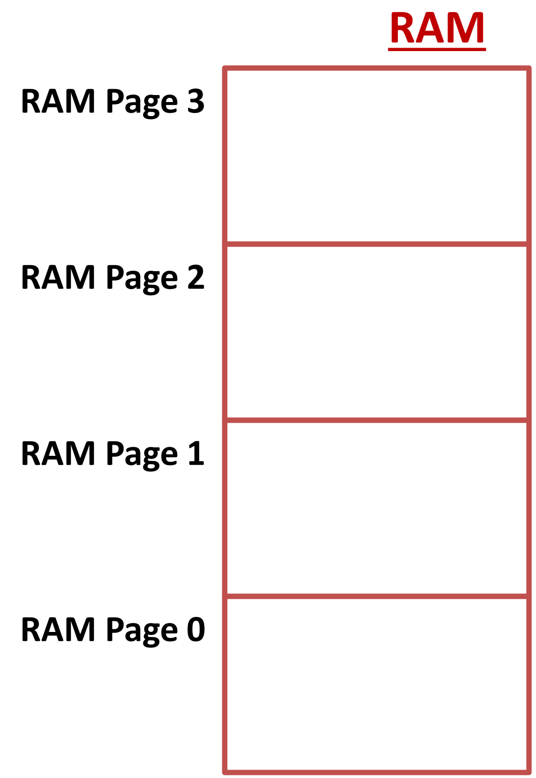
## Virtual Memory

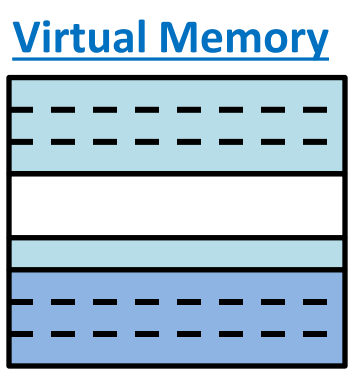


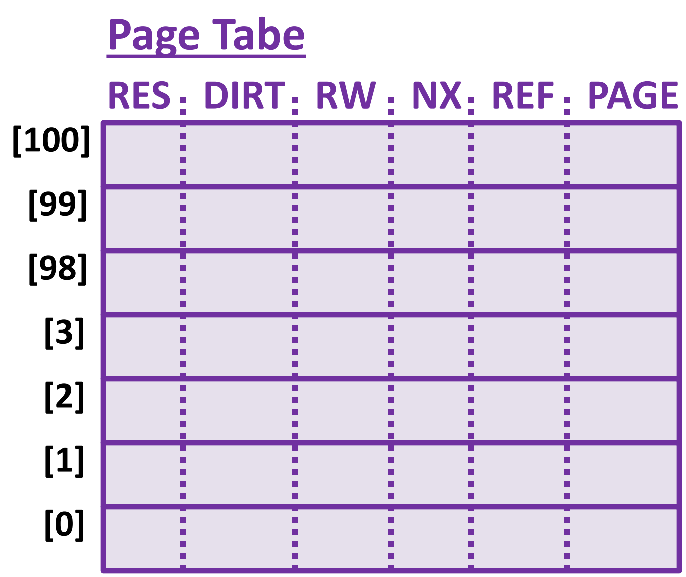
## Hard Drive

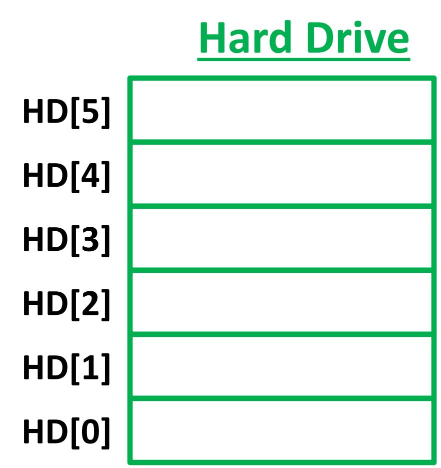


```c
int subtract(int a, int *b) {
     int c = a - *b;
     return c;
}

int add(int a, int *b) {
     int c = a + *b;
     return c;
}

void main() {
     char *b = malloc(sizeof(int));
     *(b + 1000) = 9;
}
```



# Segmentation Faults

- A “Seg Fault” occurs when an access is made to a virtual memory address that cannot be resolved.

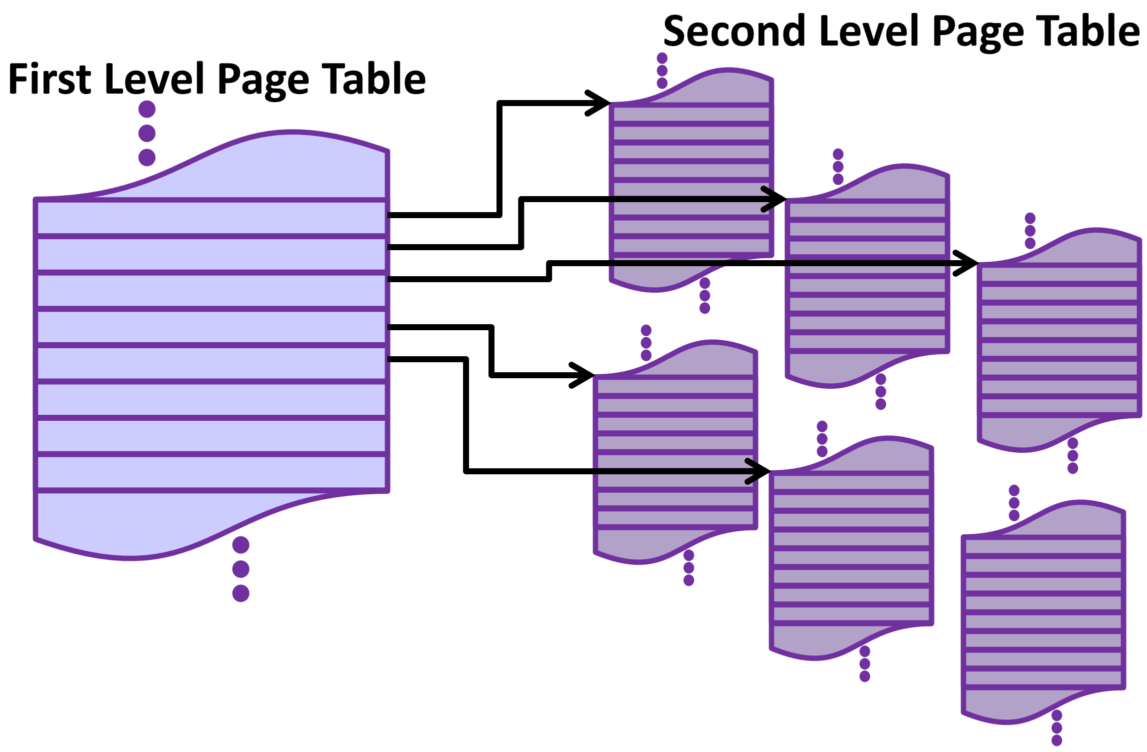
# x86 Page Table

- In x86:
  - Pages are 4 KB in size
  - Virtual Addresses are 32-bits
  - Each PTE is 4 B in size

- How large is the Page Table for each process?

# Multi-Level Page Table

- **Solution**: Create multiple levels of tables to look up a physical memory address.



# Multi-Level Page Tables

- Each virtual address can now be divided into (n+1) different pieces for an (n) level page table.
  - **Example:** Two Level Page Table:
    - First Level Page Number
    - Second Level Page Number
    - Page Offset

- Given
  - 32-bit Virtual Addresses
  - 4 KB Pages
  - 12-bit First Level Page Table Number

- What are the components of the address:

  **`0x48503423`**