Recitation 6

Midterm Review!

Thread
Process
Corporate needs you to find the differences between this picture and this picture.
Linux kernel
They're the same picture.

Table of Contents

1. Process Groups and Signaling
2. Memory Management
3. Caches
4. Threads
5. Sample Problems

Some of these concepts are pretty straight forward...

BUT do you REALLY know them?

fork(2)

```
int main() {
  int i = 3;
  fork();
  i++;
  print(i);
}
```

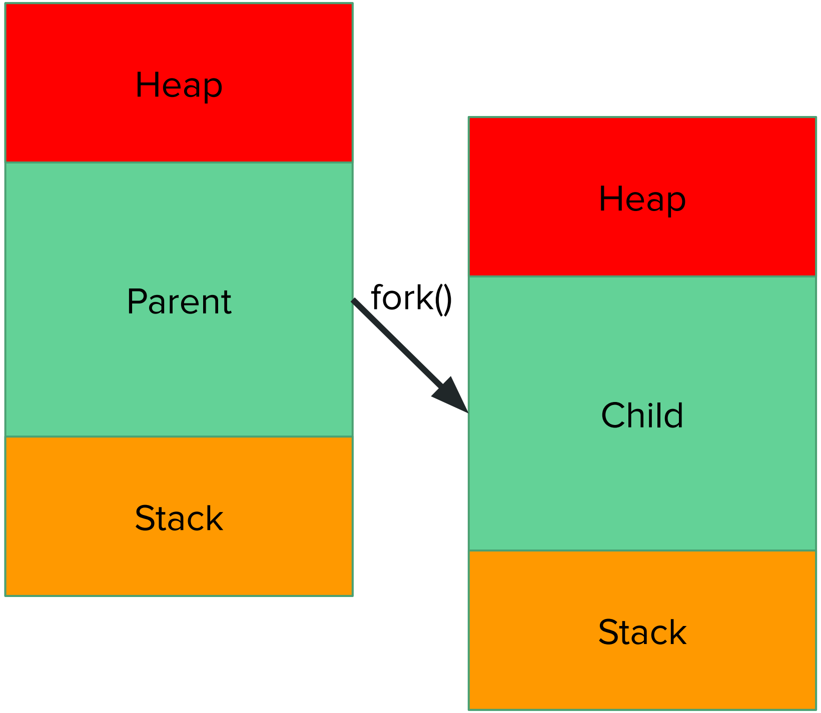
What is printed?

4

4

```
int main() {
  for (i is [1,3]) {
    pid = fork();
     if (pid == 0)
print("hi\n");
  }
}
```

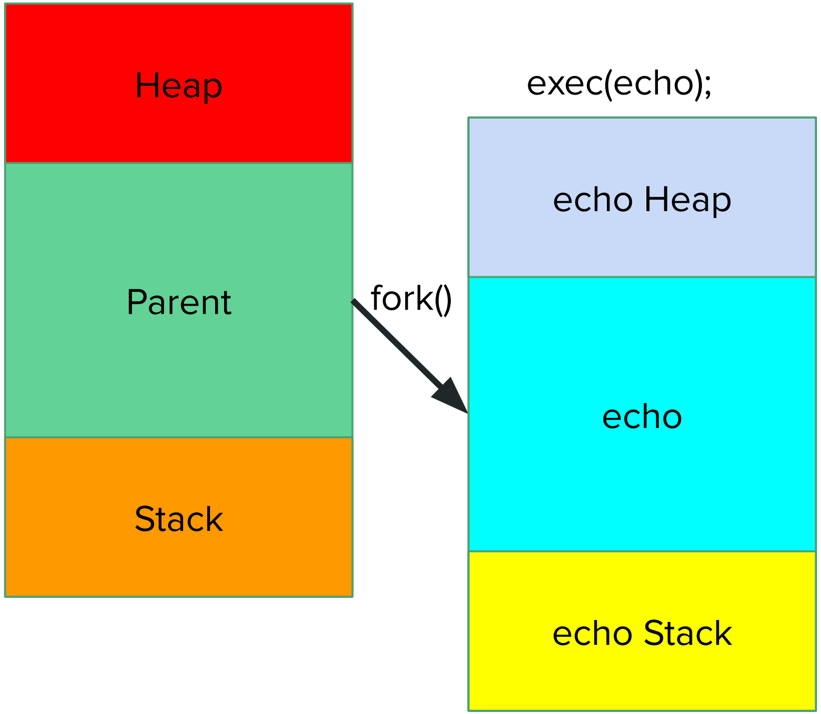
How many children? **7**



So how do we control behavior of children?

exec(3)

- Loads in a new program for execution
- **Resets** PC, SP, registers, memory for the new program to run
- Process EXITS after successful exec(3)



```
int main() {
  for (i is [1,10]) {
    pid = fork();
    if (pid == 0)
      execvp("echo", ["echo", "hi", NULL]);
  }
}
```

How many ‘hi’? **10**

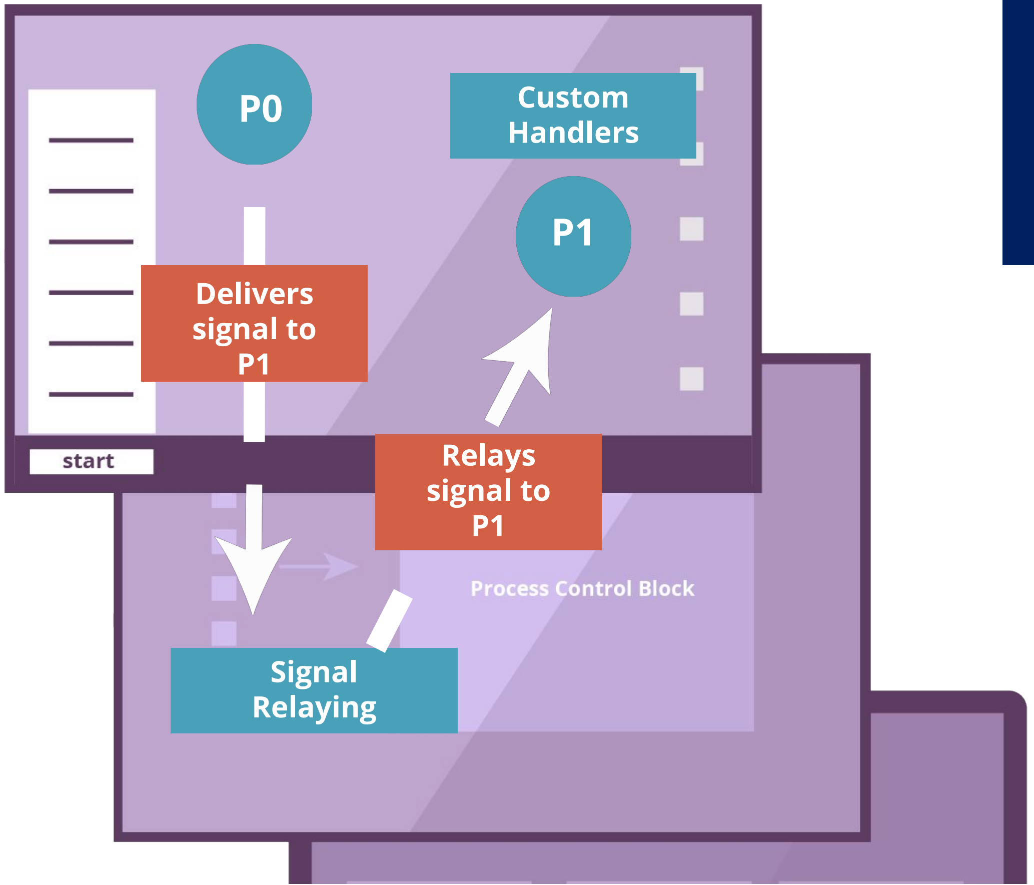
Inter-Process Signaling

User Space

When was this used in our projects?

alarm!

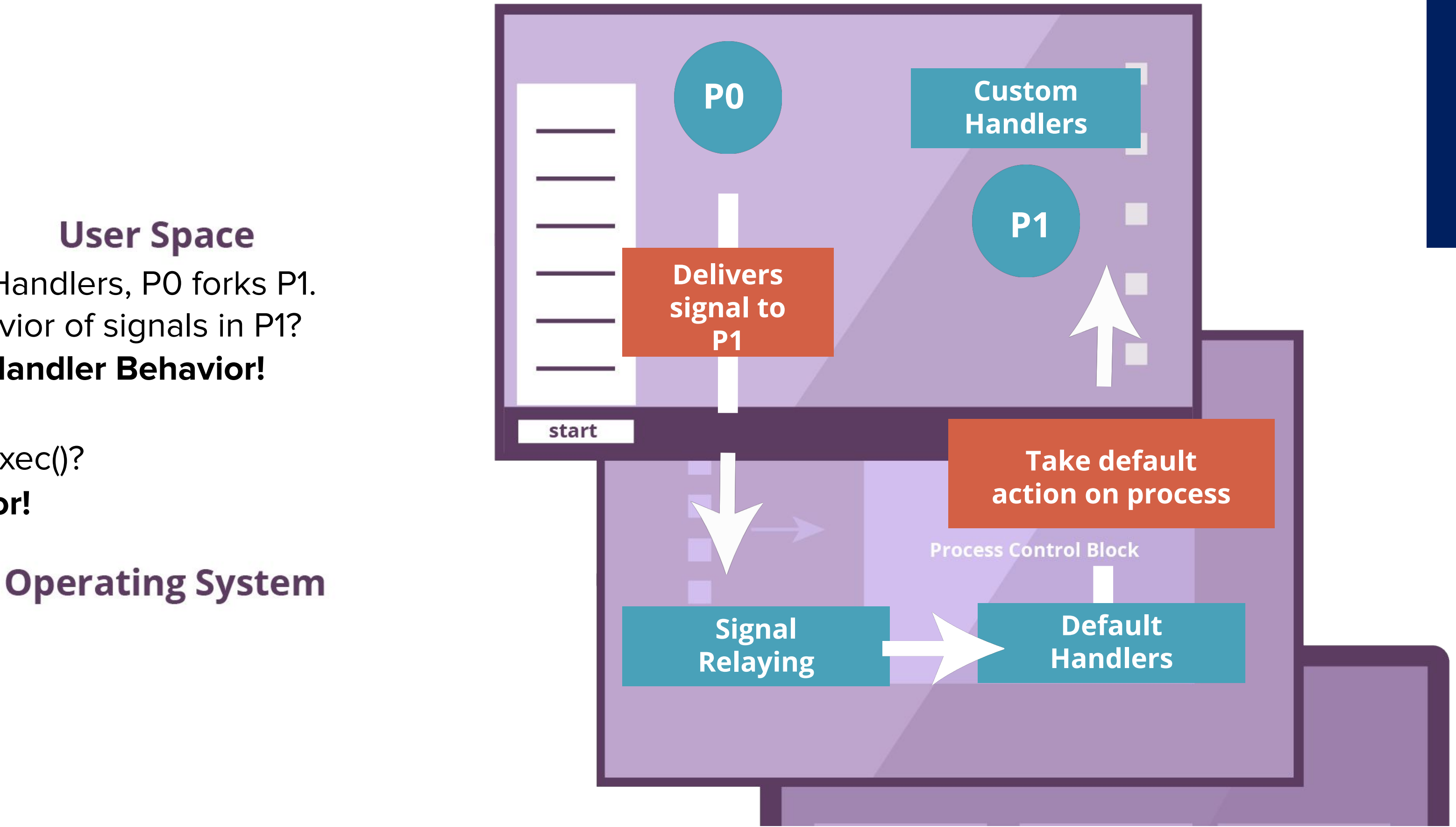
Operating System



Inter-Process Signaling

P0 had Custom Handlers, P0 forks P1.
What is the behavior of signals in P1?
Custom Signal Handler Behavior!

What if P1 had exec()?
Default Behavior!



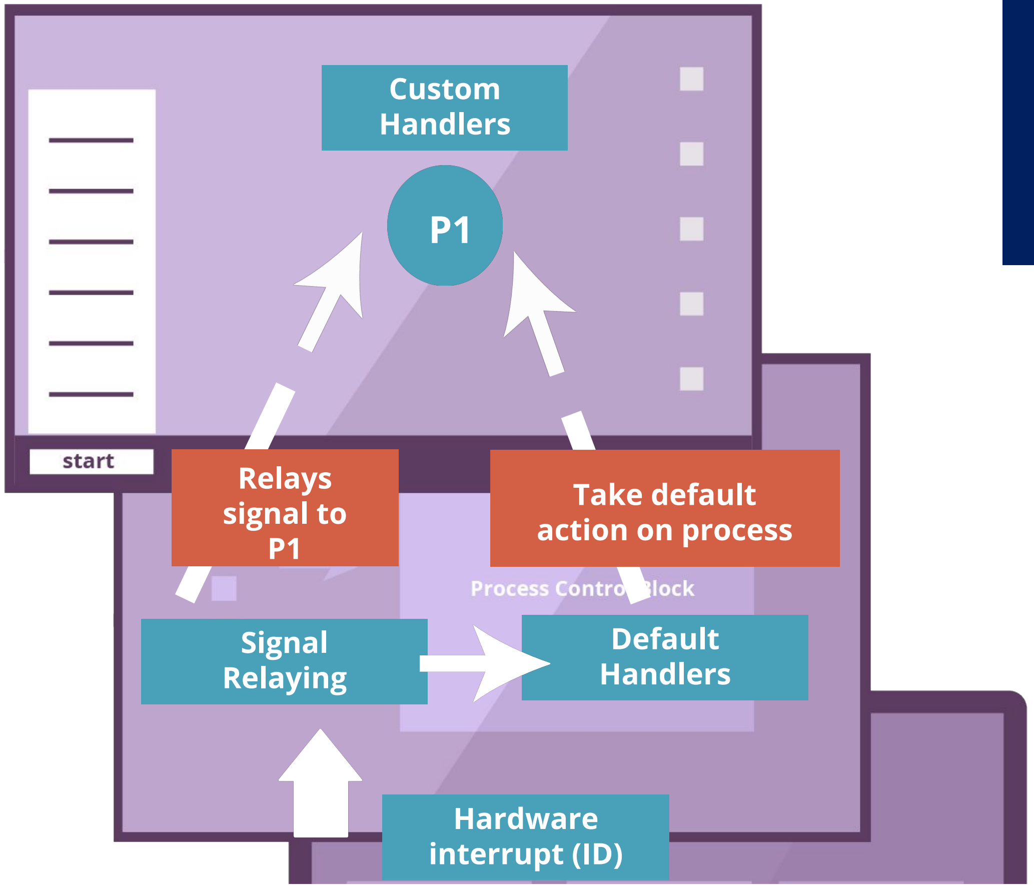
Inter-Process Signaling

User Space

What does the process need to receive these signals?

Terminal Control!

Operating System



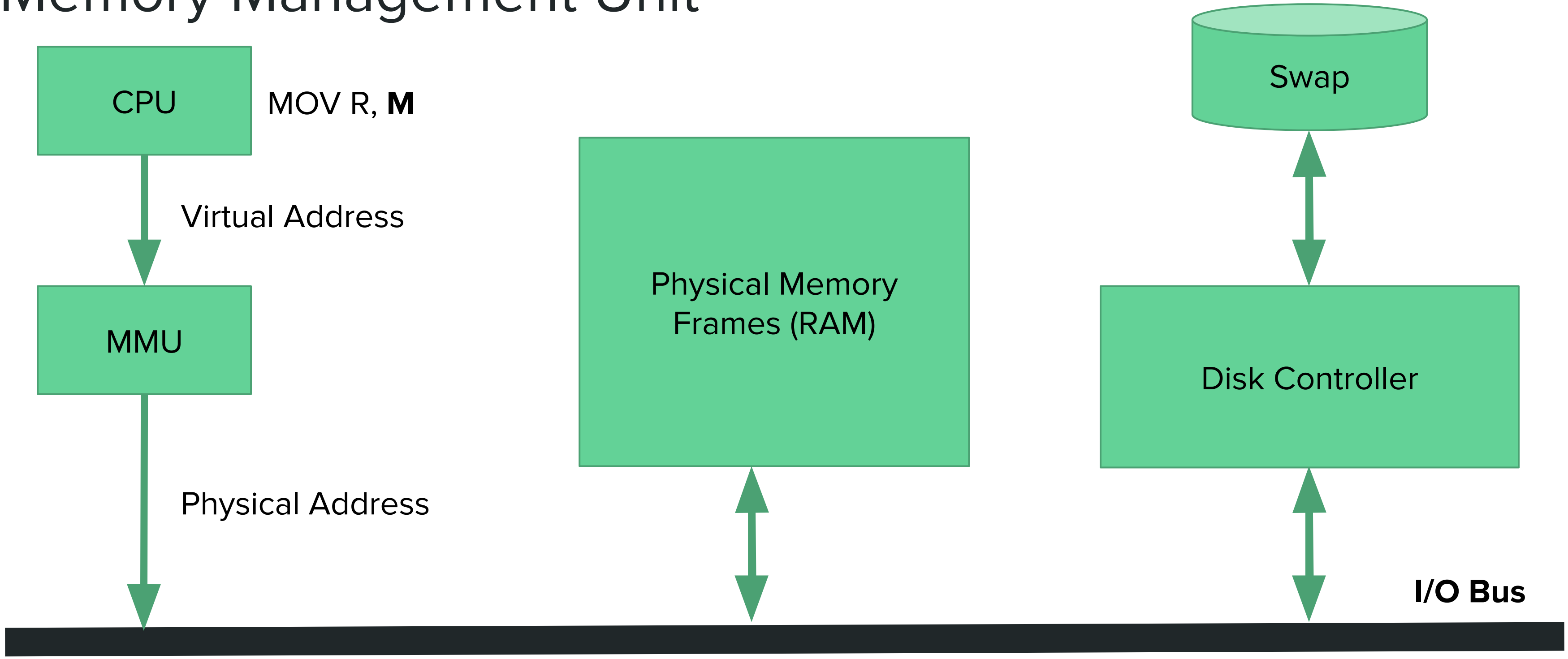
Other Process Related Topics…

1. Redirection
2. Pipes
3. Terminal Control

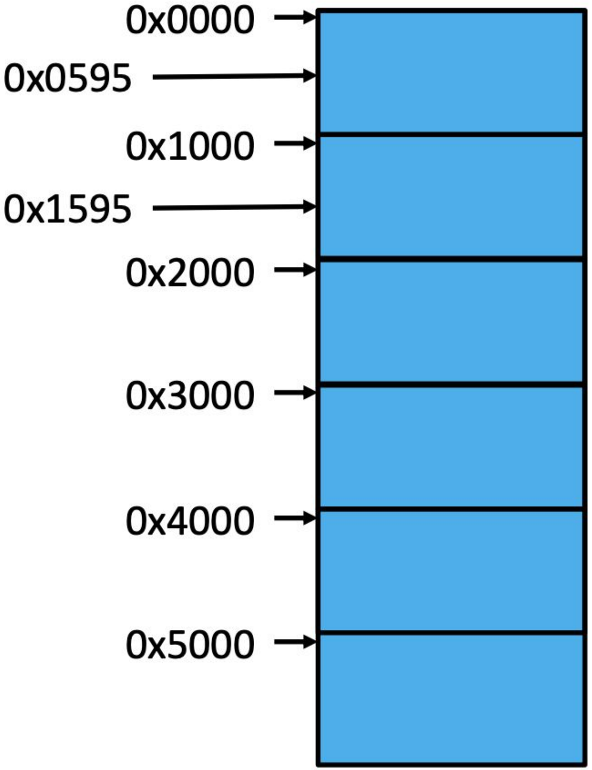
Virtual Memory

- **The x86-64 architecture (as of 2016) allows 48 bits for virtual memory and, for any given processor, up to 52 bits for physical memory. These limits allow memory sizes of 256 TiB (256 × 10244 bytes) and 4 PiB (4 × 10245 bytes), respectively.**
- Each program “thinks” it has that much memory
- But in real life, memory is bounded physically by RAM and SSD

Memory Management Unit



Memory Translation



What is the page size?

4KB / 4096 bytes

How many bits represent page offset?

12 bits

How many bits represent each page?

4 bits

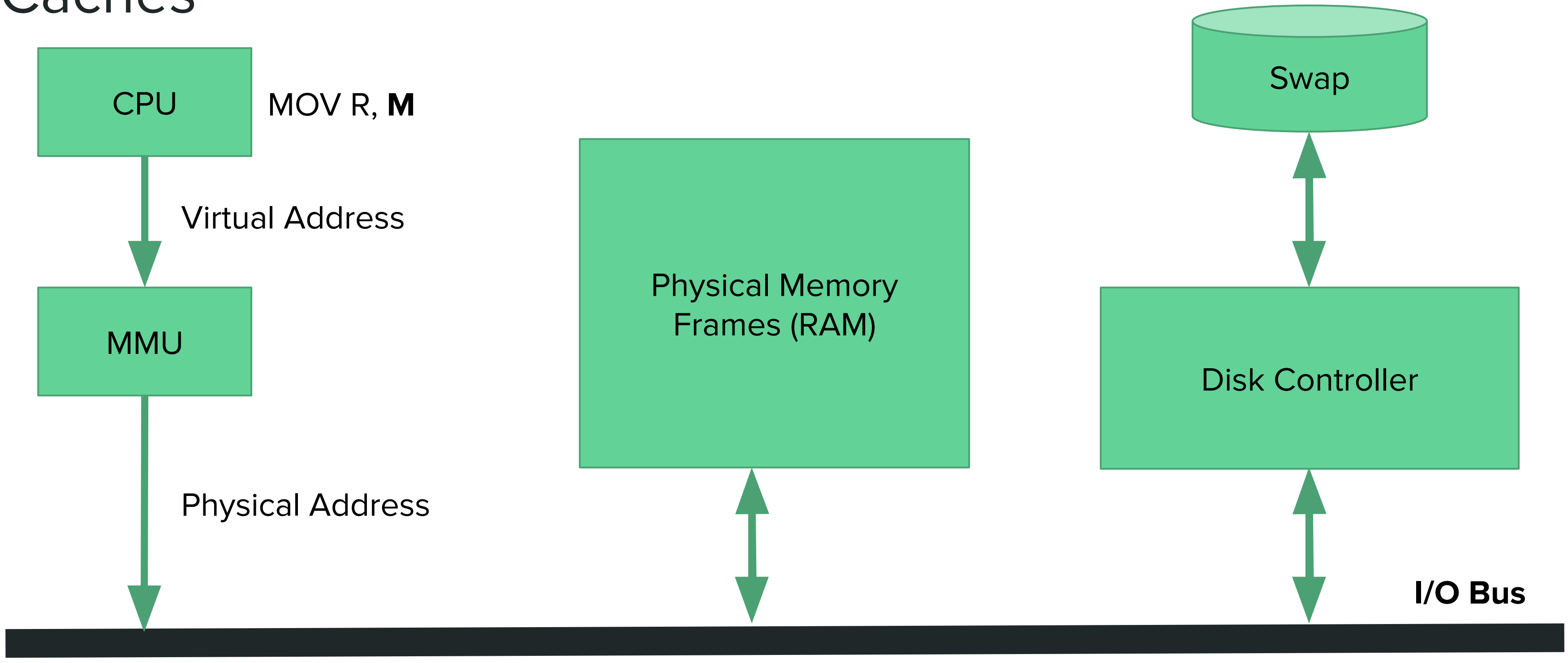
How many pages?

16 pages

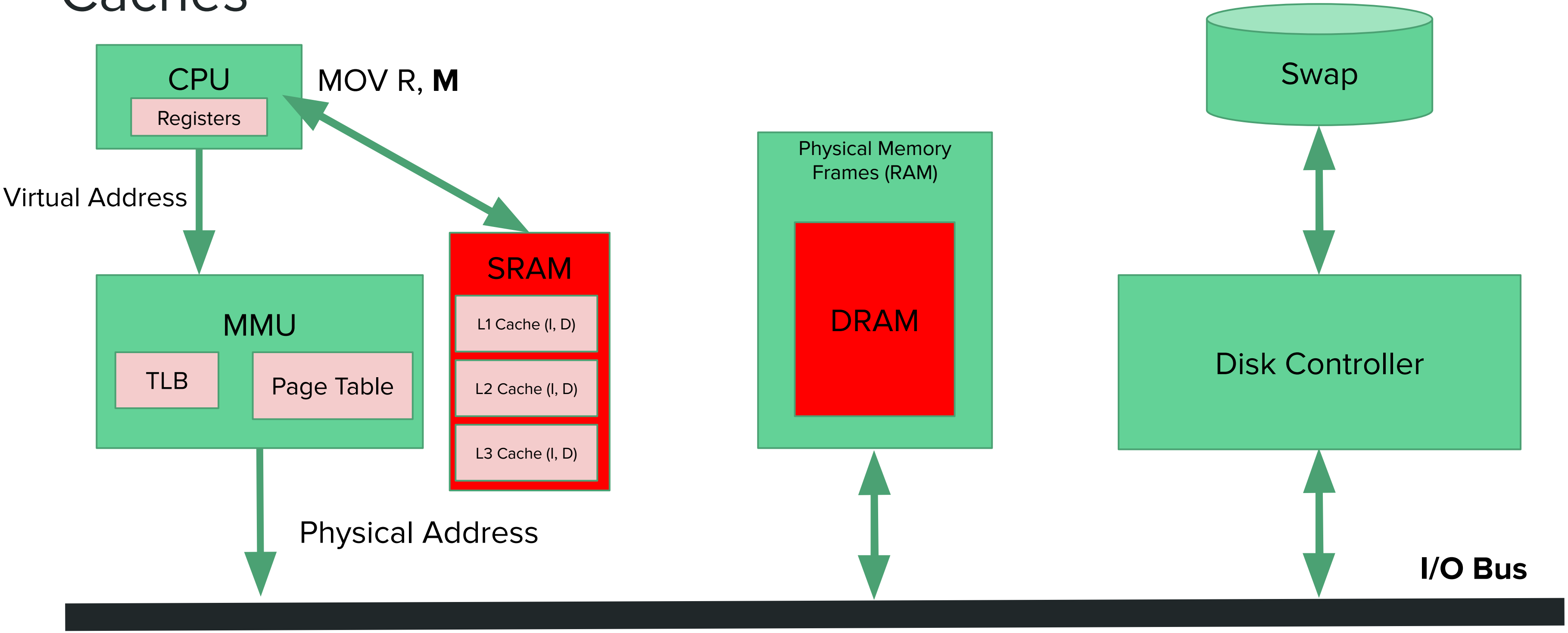
How many addresses per page?

4096

Caches


CPU
MOV R, **M**
Virtual Address
MMU
Physical Address
Physical Memory Frames (RAM)
Swap
Disk Controller
I/O Bus

Caches


CPU
Registers
MOV R, **M**
Virtual Address
MMU
TLB
Page Table
SRAM
L1 Cache (I, D)
L2 Cache (I, D)
L3 Cache (I, D)
Physical Address
Physical Memory Frames (RAM)
DRAM
Swap
Disk Controller
I/O Bus

Least Recently Used (LRU)

- Memory is limited
- Which line of memory to evict/replace when we run out of memory?
  - LRU
- Advantages of LRU
  - Generally good performance, we are evicting a page that is “least” frequently used
  - Reduces number of page faults
- Disadvantages of LRU
  - Quite costly to find the LRU page
- Think of a scenario where LRU may actually hurt performance
  - Sequential Access: If some sequential access pattern forces LRU to evict and re-allocate parts of memory, we will have poor performance

Threads!

- Processes vs Threads?
    - Processes are more “heavy weight” than threads
        - Unique memory address space,
    - THREADS SHARE MEMORY!
        - Unique stack, PC, and registers, all in one address space
    - Processes are Isolated
    - BOTH can run concurrently but,
    - Context Switching is more expensive in Processes

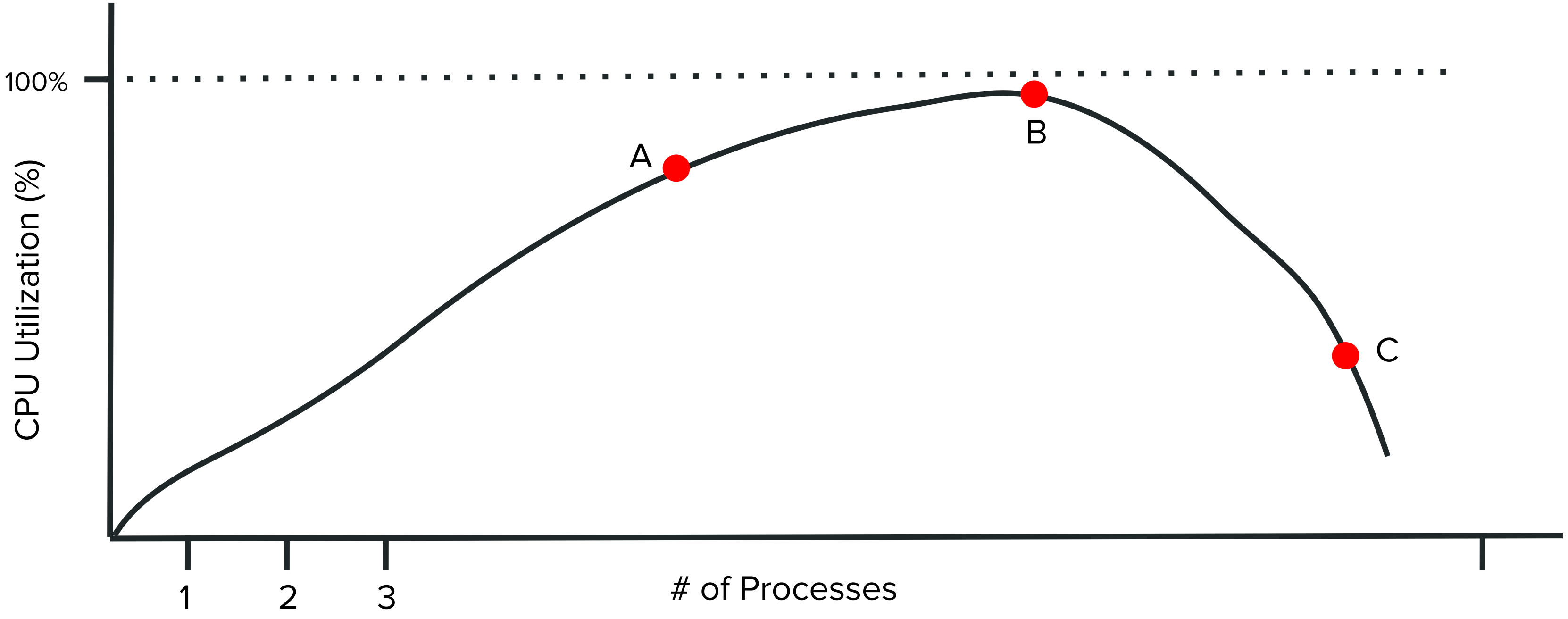
…

- Scheduling is **NOT COVERED** in the midterm!

Some Thinking Questions 1

Consider this graph of CPU Utilization vs # of Processes Running

What is happening at each point?



A: Context Switching becomes a problem

B: SUM(memory utilization) > RAM, so we start using more SWAP file

C: Thrashing: Most of the memory access causes a page fault, and we use SWAP a lot

Some Thinking Questions 2

Consider the following system:

32 bit address space
2GiB physical memory size
byte addressable
32KiB page size

Total virtual address space in bytes? 2^32 bytes Number of bits per virtual address? 32 bits

Number of page offset bits? 15 bits

How many page table entries per page table? 2^17 page table entries

How many frames in physical memory? 2GiB/32KiB = 2^16 frames

What if the architecture was 2 byte addressable?

2^33 bytes, 32bits, 14bits, 2^18 page table entries, 2^16 frames, but ½ # of addresses in physical memory

Some Thinking Questions 3

For the following state whether it would be better to use multiple threads or processes:

1. You want to process a big image by calculating the average of all pixels **Threads**
2. You want to compile a huge C++ library with over 2000 source files **Threads**
3. You have a system of receiving and logging lots of transactions, each action needs data integrity **Process**
4. You have a word processor that constantly checks for spelling mistakes, grammar issues, and syntax errors. **Threads**

More Practice Problems

https://www.seas.upenn.edu/~cis3800/23fa/exams/midterm0