Section 7: Intro to Lab 4

CSE 451 18WI

Memory vs Disk

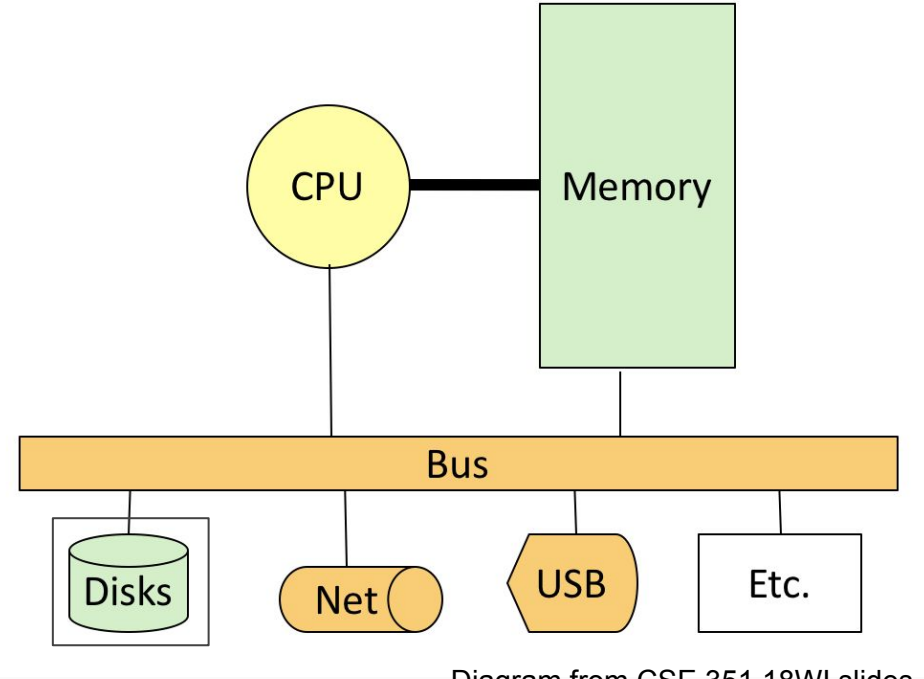
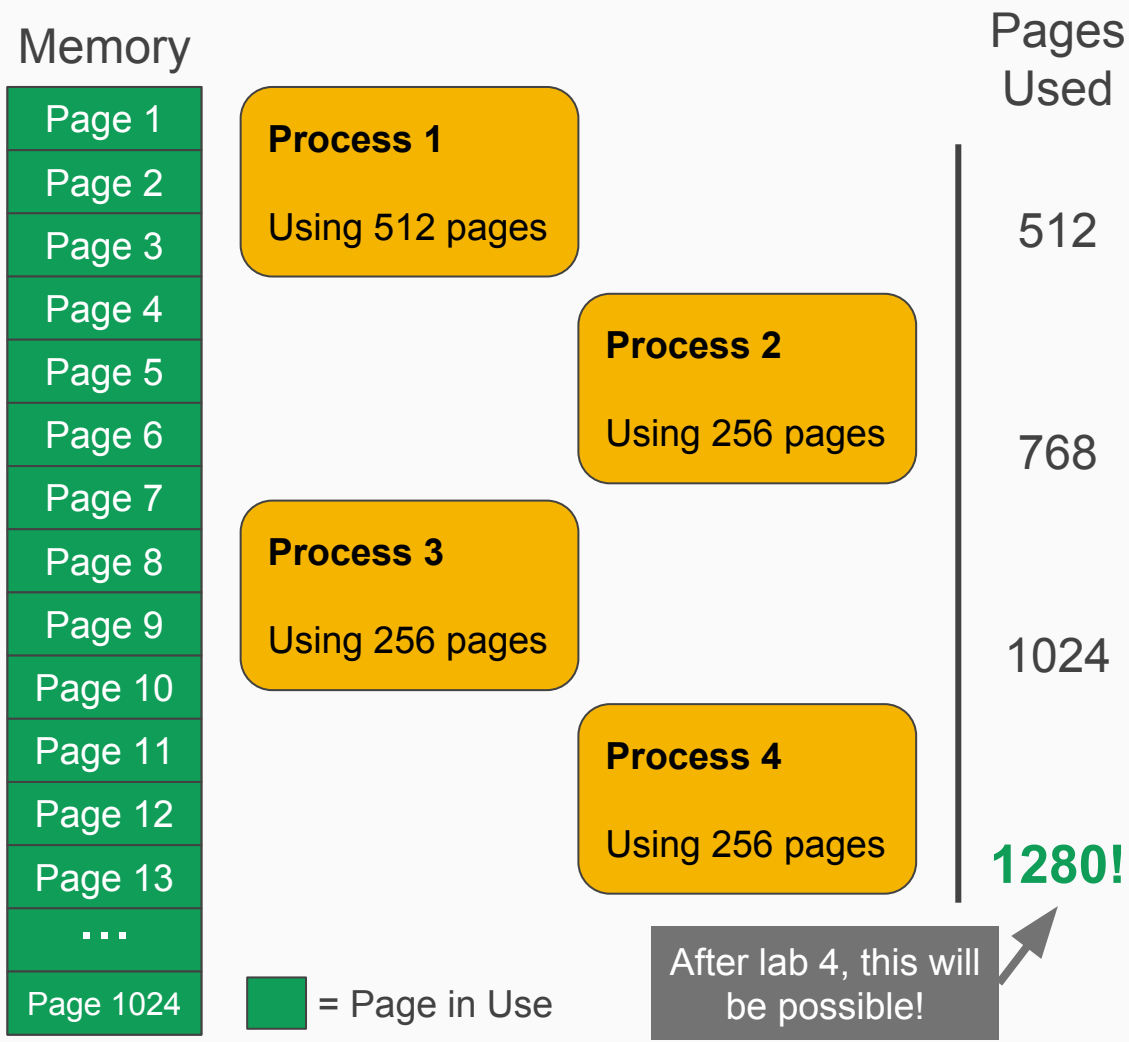


Diagram from CSE 351 18WI slides

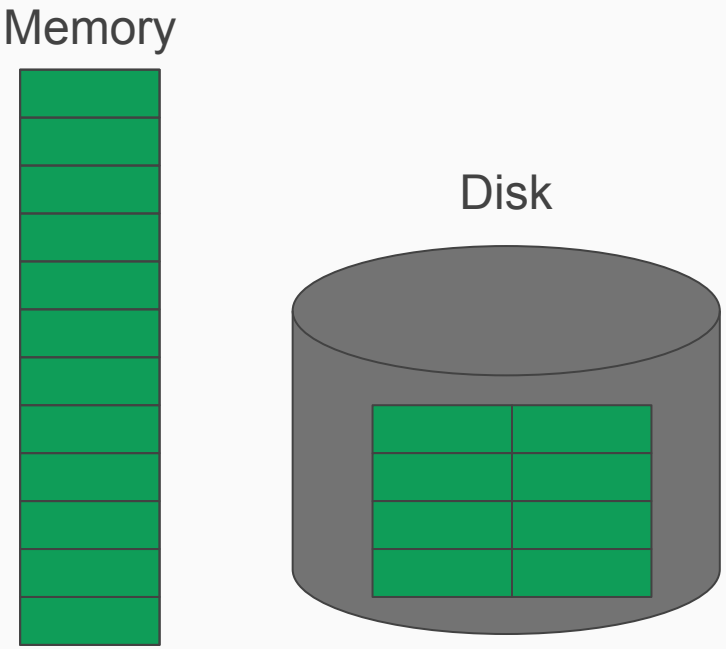
- Memory is in close proximity to the CPU
  - Fast!
  - Volatile (loss of power == loss of all data in memory)
  - More expensive
- Disk is farther away from the CPU
  - Much slower than main memory
  - Non-volatile (loss of power != loss of data), persistent
  - Less expensive

Virtual Memory

- Illusion that each process has all of memory to itself
- Would be nice if this illusion held even when processes together use more space than available in memory

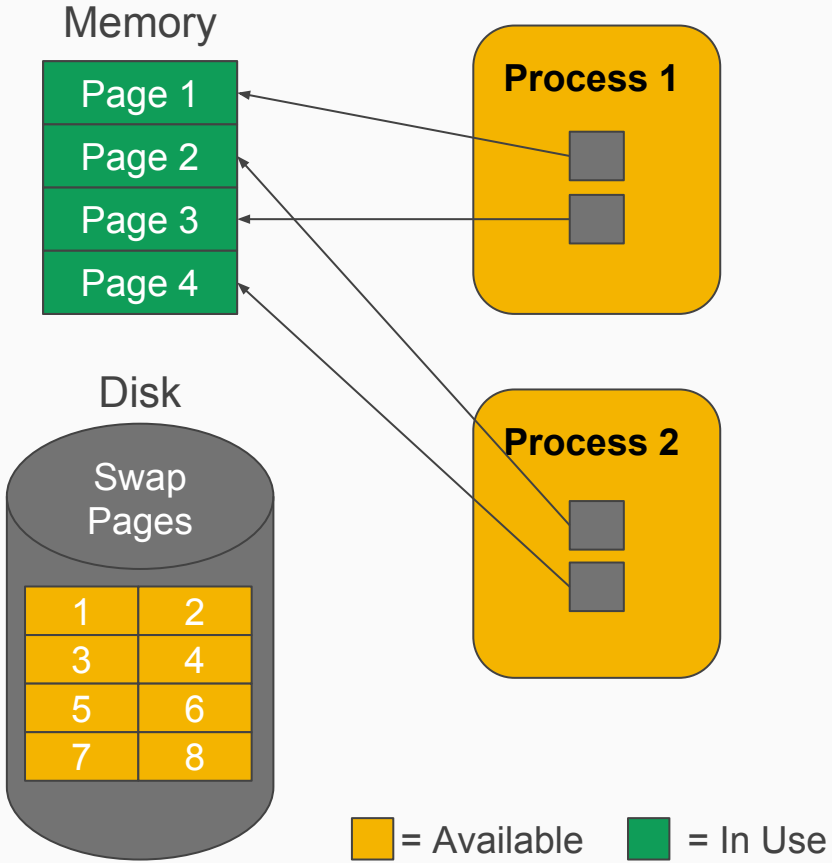


Creating the illusion of more memory



- Since we need to make it seem like there is more than 4MB of memory, we will need somewhere else to store data
- Can use the disk to store extra data, and page it in to memory on demand (called “**paging**”)

Paging Example - Assumes OS has only 4 pages memory for simplicity



This mapping could be obtained as a result of the following requests:

Proc 1: Requests a page of memory

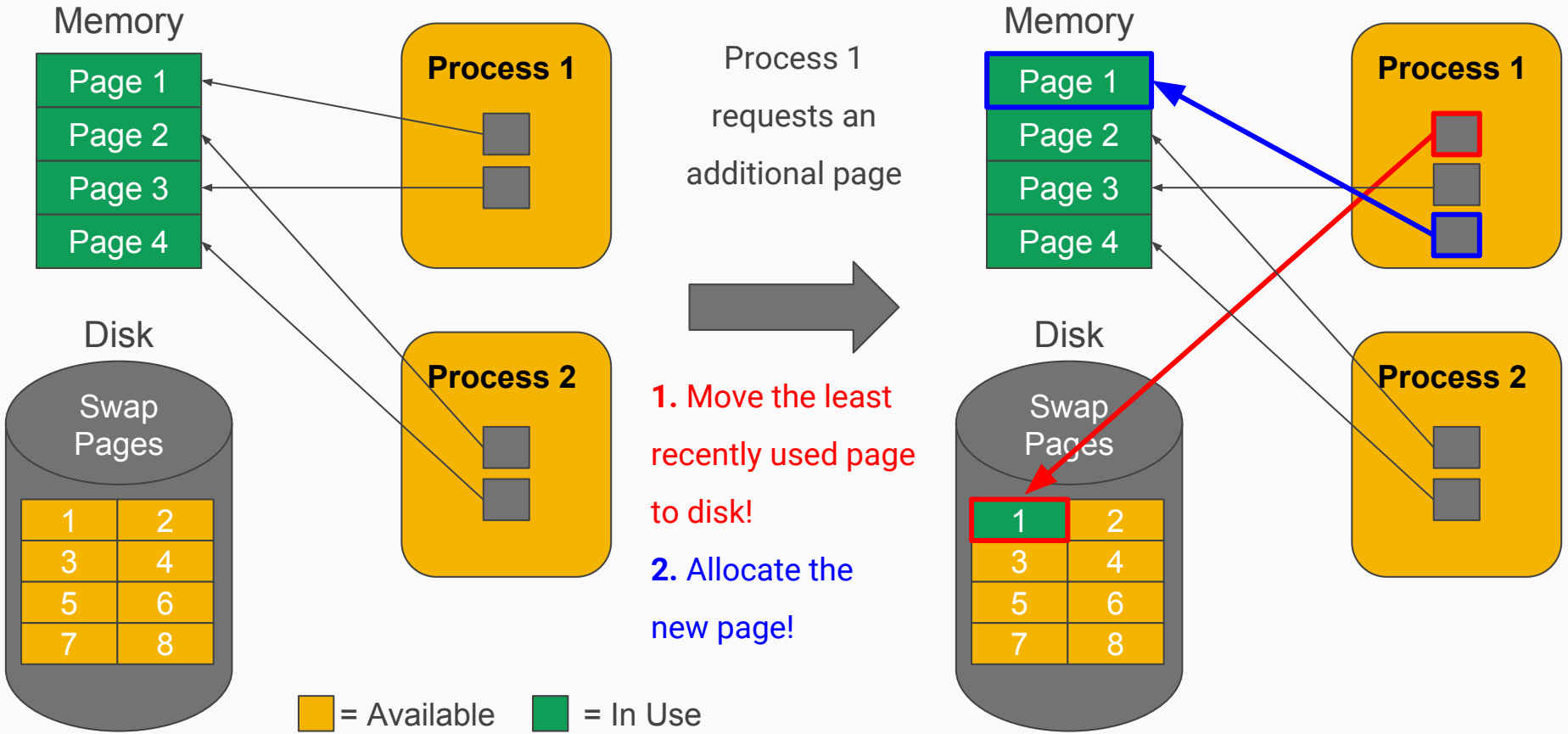
Proc 2: Requests a page of memory

Proc 1: Requests a page of memory

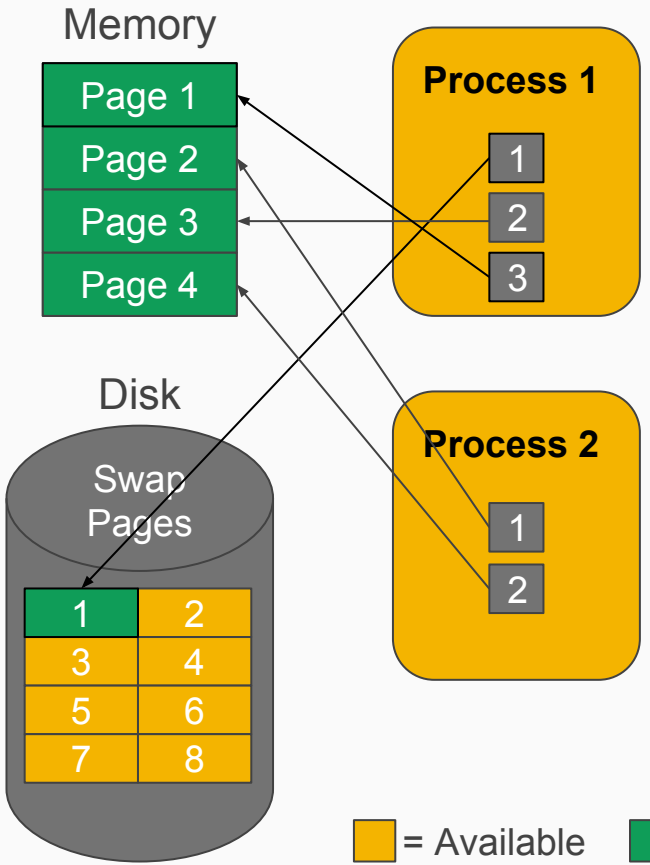
Proc 2: Requests a page of memory

Note: This example is highly simplified

Paging Example - Swap page to disk



Paging Example - Page fault (Page not present), Part 1

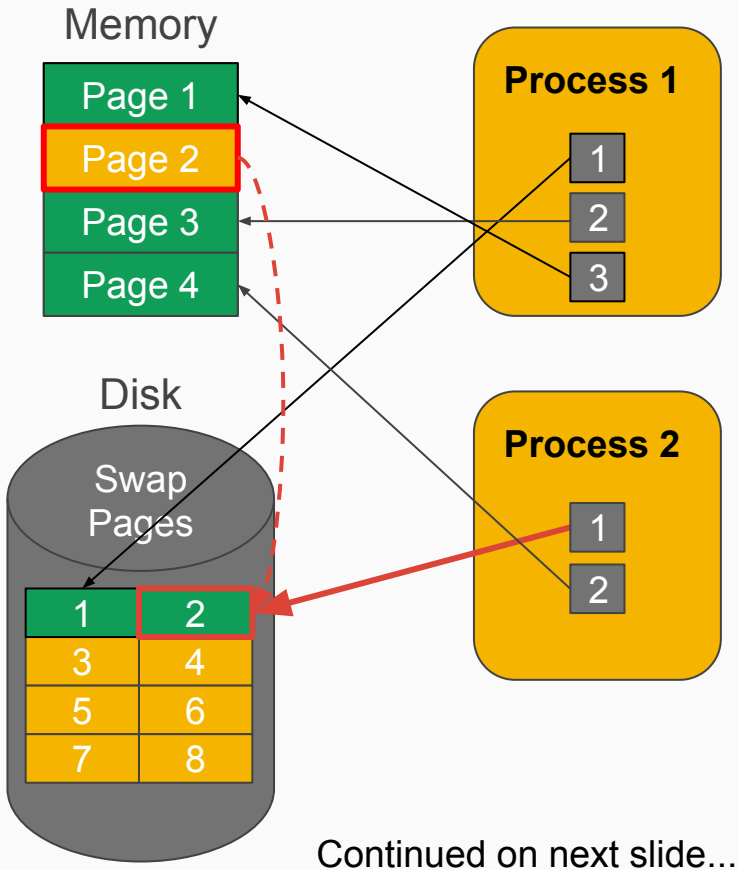


Process 1 tries to read from its 1st page

Page Fault!

Need to make room for the page stored on disk.

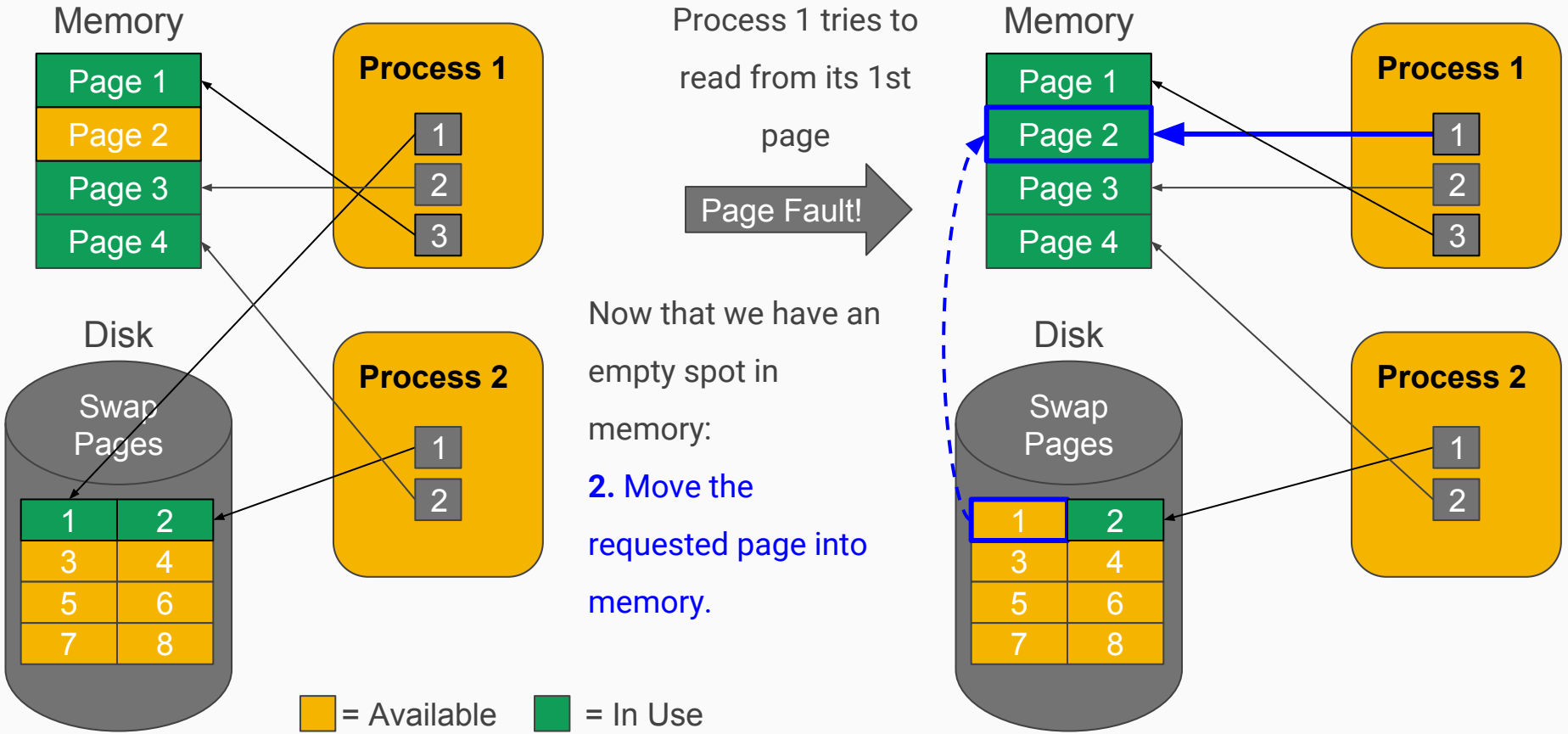
1. Move the least recently used page to disk to make room!



= Available = In Use

Continued on next slide...

Paging Example - Page fault (Page not present), Part 2



XK's Memory

XK's hardware is emulated by QEMU. In `kernel/Makefrag` we set up the options we will pass to QEMU.

Before (Labs 1 - 3):

16MB (4096 pages)

```
QEMUOPTS += -m 16M
```

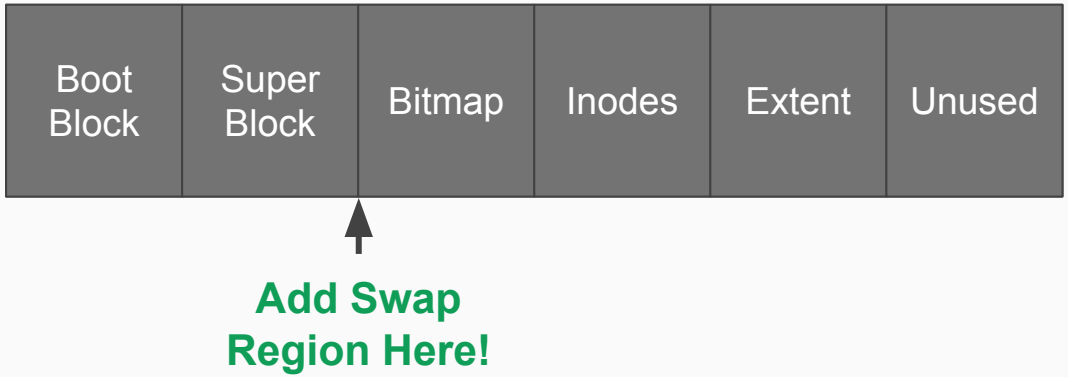
After (Lab 4):

4MB (1024 pages)

```
QEMUOPTS += -m 4M
```

XK's Disk

- Set up in **`mkfs.c`** (this file is used by QEMU, run by the host OS before XK boots and sets up the disk)
- Need to add a swap region to use for pages swapped out to disk



- 512 bytes in a disk block
- 4096 bytes in a page
- Therefore, need 8 disk blocks per swap page

Representing the Swap

- How do you add the swap region to disk?
  - Hint: **lab4.md** diagram and **mkfs.c**
- How should we keep track of a memory page that is in swap region?
  - Hint: See how **kalloc.c** tracks physical pages for a design example
- How do you track in a vspace whether a page is in physical memory or swap memory?
  - Hint: look at **vpage_info** and how that was used in Lab 3 COW fork
- What should happen when a swapped memory page is shared via copy-on-write fork?

Swap In

- When should we load pages from the swap region?
  - Hint: similar to lab 3's "when should we make a physical copy of a COW page?"
- When a page is swapped in, what needs to be updated?
  - Hint: who/what keeps track of whether a virtual page is in the swap?

Swap Out

- When should we flush pages to the swap?
  - Hint: Look at **kalloc.c** and at the algorithm in **lab4.md**
- Is there a set of memory pages you don't want to flush to swap?
  - Hint: What happens if the trap code page is in the swap?
- When a page is swapped out, what needs to be updated?
  - Hint: who/what keeps track of whether a virtual page is present in physical memory?

LRU Approximation (Second Chance)

- Like FIFO, with a small change
  - Has it been accessed since the last time I checked this page?
    - If so, skip for now and clear the access bit
    - Otherwise, evict!
      - Are there any exceptions to this? (Hint: previous slide)
- You'll want to use **vawasaccessed()** in **kernel/vspace.c**