Section 3: Virtual Memory

CSE 451 18SP

Announcements

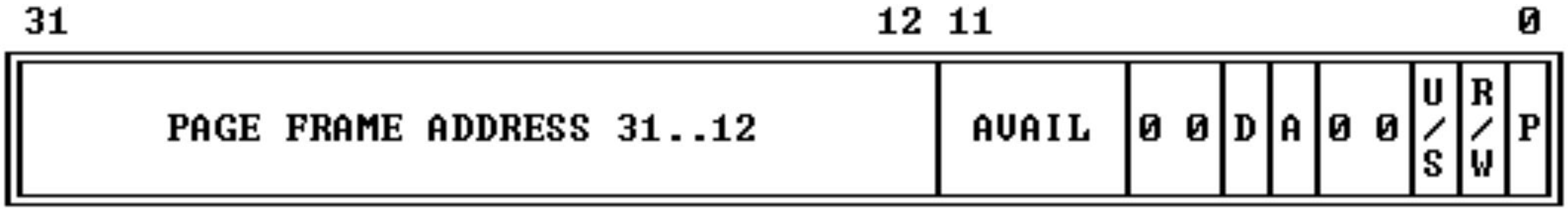
- Lab 2 is due Tuesday 4/17 @ 11pm
    - Don’t forget to answer questions!

Free List Review
(on board)

Address Translation

Page Table Entry

Figure 5-10. Format of a Page Table Entry




```
P       - PRESENT
R/W     - READ/WRITE
U/S     - USER/SUPERVISOR
D       - DIRTY
AVAIL   - AVAILABLE FOR SYSTEMS PROGRAMMER USE

NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.
```

From Intel Manual: https://courses.cs.washington.edu/courses/cse451/18sp/readings/i386/s05_02.htm

JOS Address Translation (Note PTE is simplified, see previous slide for full format)

VA: 0x5ec62b56

pgdir 0xff000000

Index		
0	0xff001000	1
	...	
16	0xff008000	1
	...	
379	0xffa00000	1
	...	
1023	0xff400000	1

page table page **0xff001000**

Index		
	...	
283	0x001a3000	1
	...	
374	0x001a9000	1
	...	

page table page **0xffa00000**

Index		
	...	
98	0x001e0000	1
	...	
948	0x001ed000	1
	...	

page table page **0xff008000**

Index		
	...	
386	0x001bc000	1
	...	
619	0x001bf000	1
	...	

page table page **0xff400000**

Index		
	...	
326	0x001d5000	1
	...	
725	0x001d9000	1
	...	

JOS Address Translation (Note PTE is simplified, see previous slide for full format)

VA: `0x5ec62b56`

Index	pgdir 0xff000000	
0	0xff001000	1
	...	
16	0xfff08000	1
	...	
379	0xfffa0000	1
	...	
1023	0xff400000	1

1) Get binary representation of the virtual address:
   01011110110001100010101101010110
2) Extract first 10 bits
   01011110110001100010101101010110
3) Convert this to an index into the page directory:
   0101111011 binary = **379** decimal
4) Locate the PDE stored in index 379
5) Look at what address the page table page is stored at

JOS Address Translation (Note PTE is simplified, see previous slide for full format)

VA: `0x5ec62b56`

page table page **`0xffa00000`**

Index		
	...	
98	`0x001e0000`	1
	...	
948	`0x001ed000`	1
	...	

1) Extract the next 10 bits:
   `01011110110001100010101101010110`

3) Convert this to an index into the page table:
   `0001100010` binary = **98** decimal

4) Locate the PTE stored in index 98

5) Look at what the physical address is

6) Use last 12 bits as offset (3 hex digits = 12 bits)
   Physical Address = **`0x001e0b56`**

Paging

Memory vs Disk

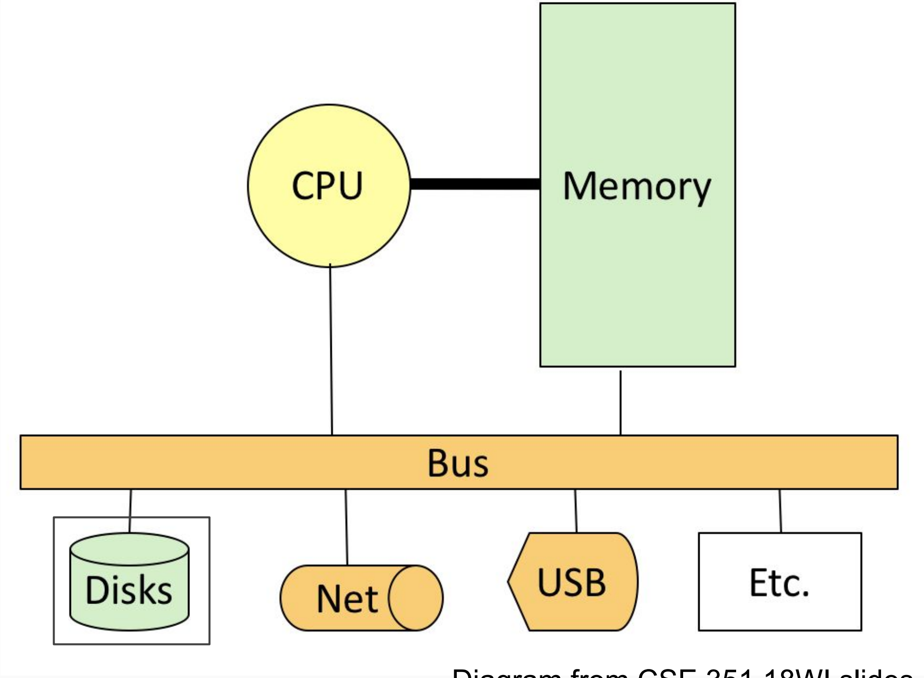
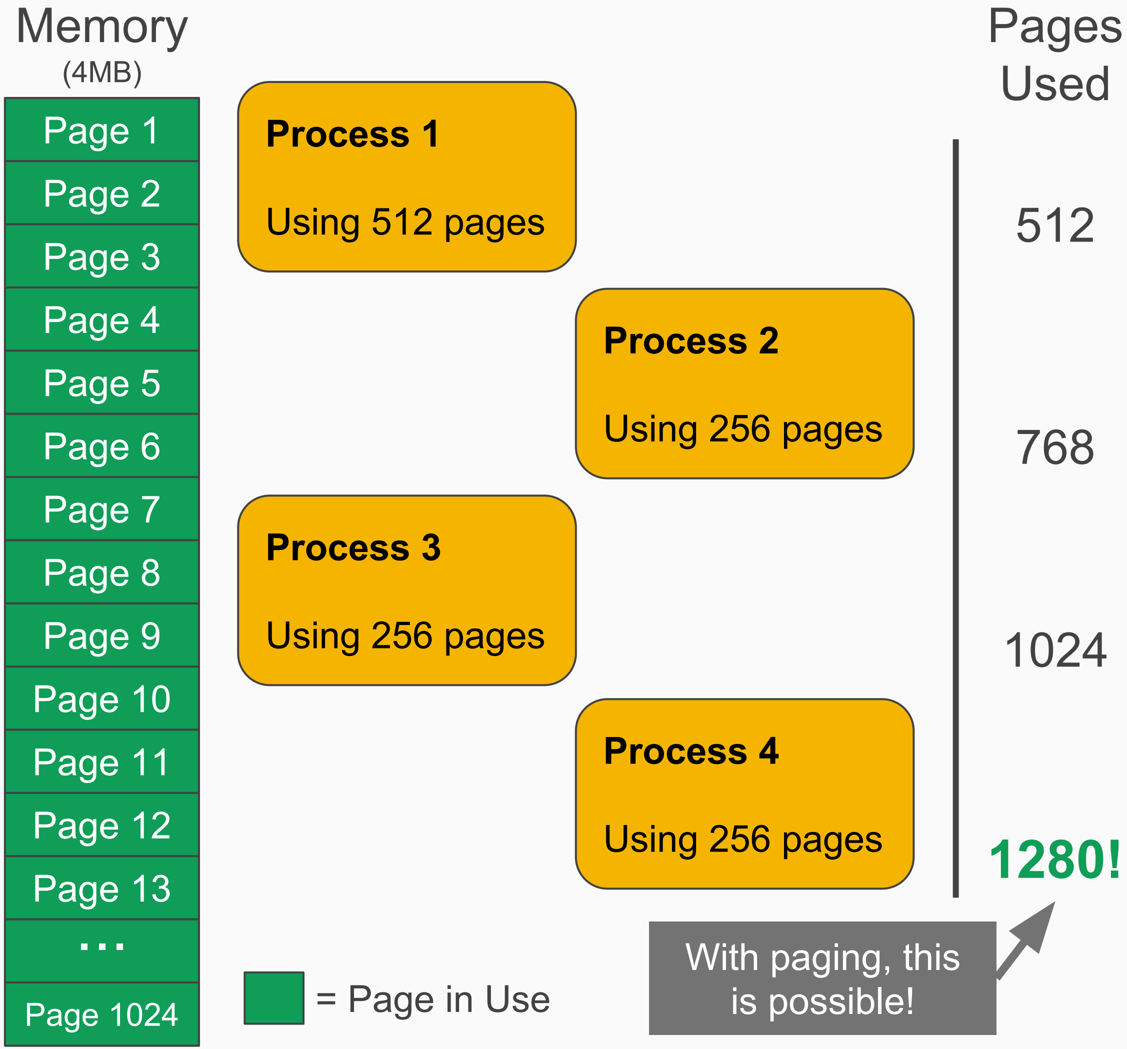


Diagram from CSE 351 18WI slides

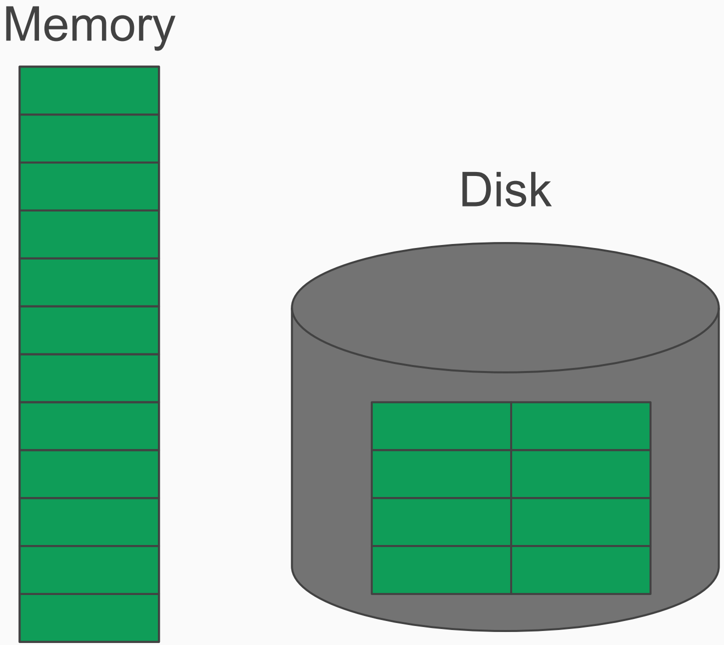
- Memory is in close proximity to the CPU
  - Fast!
  - Volatile (loss of power == loss of all data in memory)
  - More expensive
- Disk is farther away from the CPU
  - Much slower than main memory
  - Non-volatile (loss of power != loss of data), persistent
  - Less expensive

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

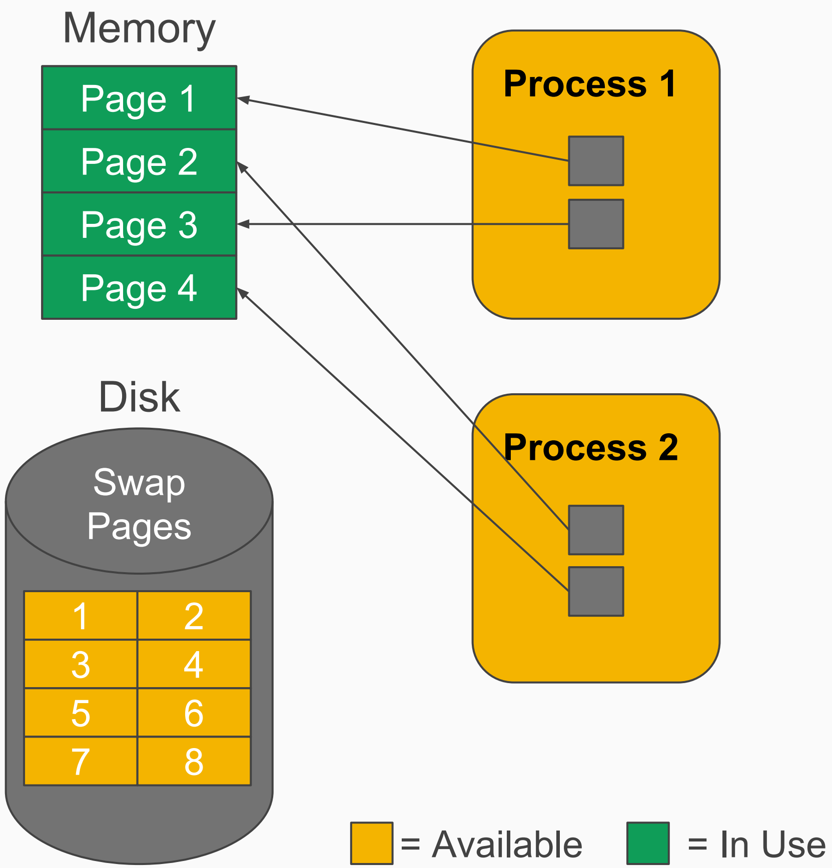


Creating the illusion of more memory



- 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

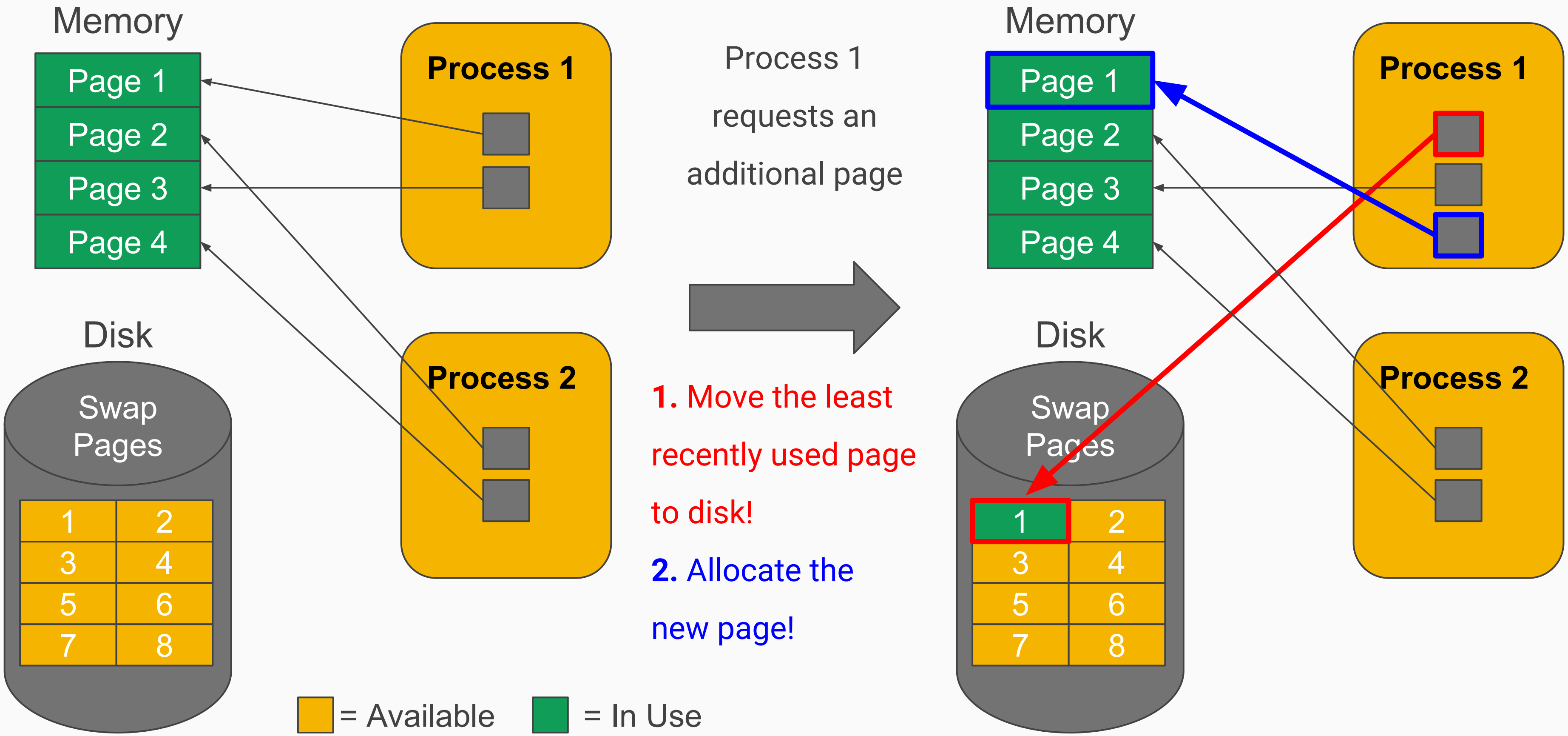
Proc 2: Requests a page of memory

Proc 1: Requests a page of memory

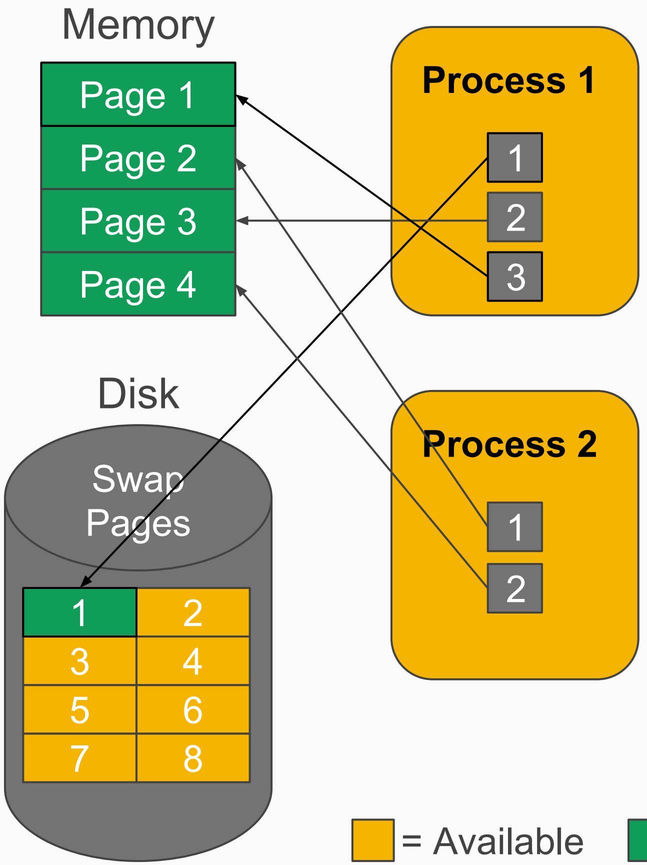
Proc 2: Requests a page of memory

Note: This example is highly simplified

Paging Example - Swap page to disk



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

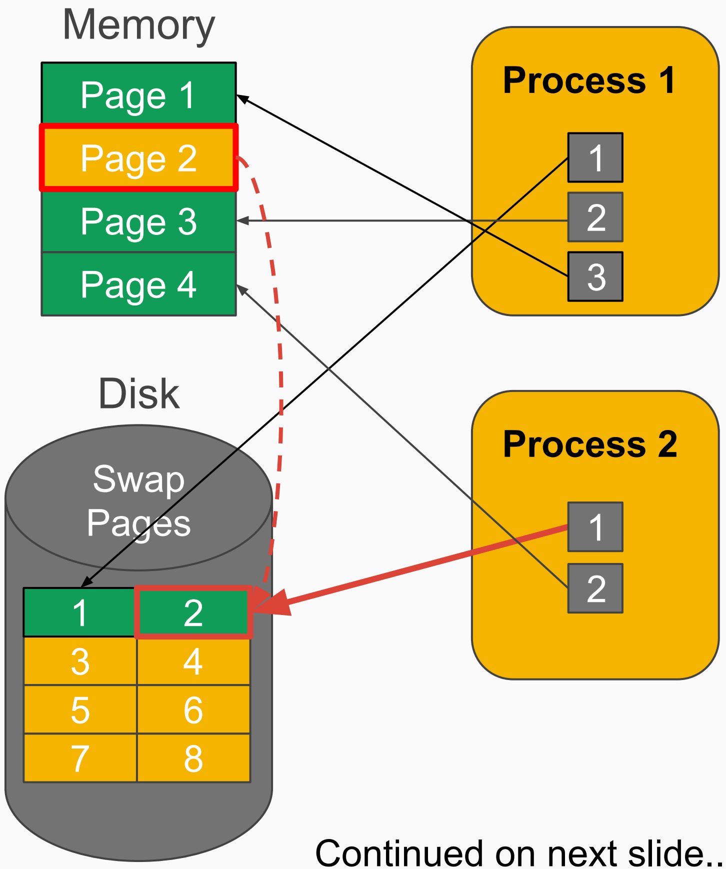


Process 1 tries to read from its 1st page

Page Fault!

Need to make room for the page stored on disk.

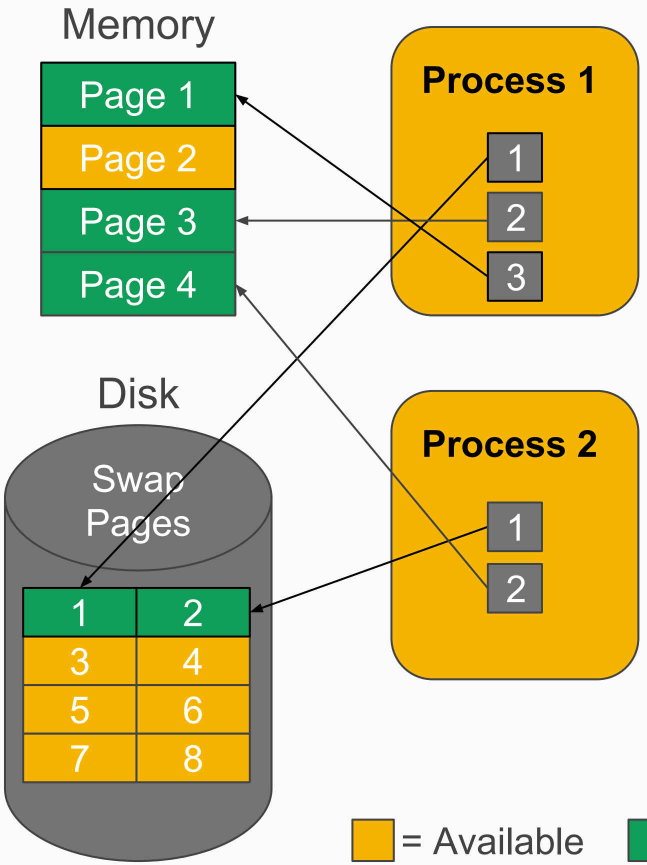
1. Move the least recently used page to disk to make room!



= Available = In Use

Continued on next slide...

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

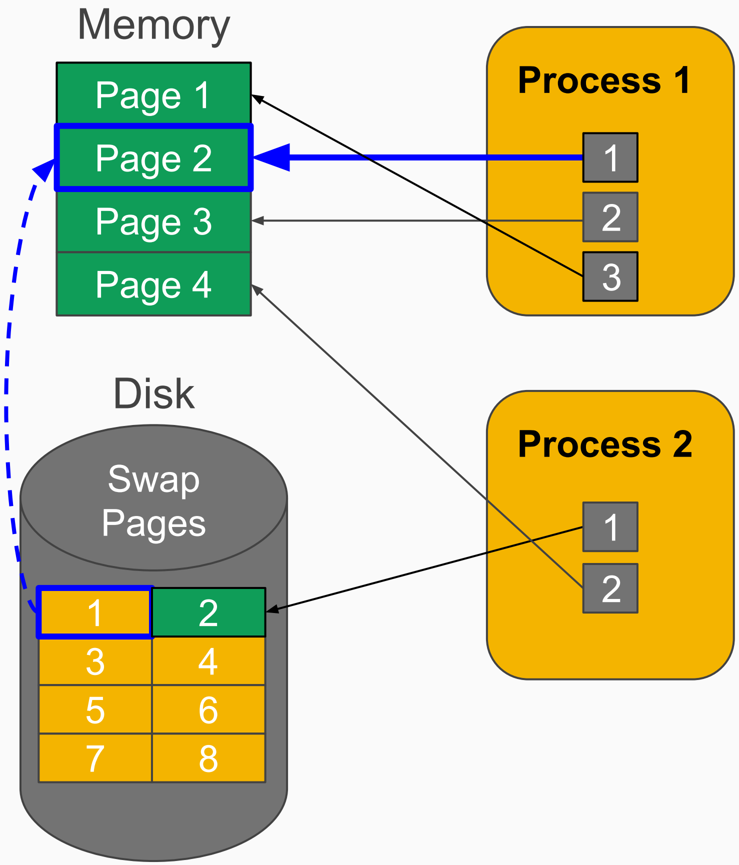


Process 1 tries to read from its 1st page

Page Fault!

Now that we have an empty spot in memory:

2. Move the requested page into memory.



= Available = In Use