Virtual Memory

Yasmine Alonso & Poojan Pandya

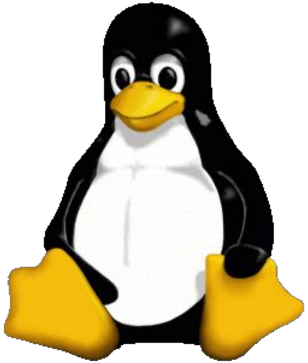
August 14, 2023

Inspired by slides from Winter ‘23 offering of CS111

Announcements

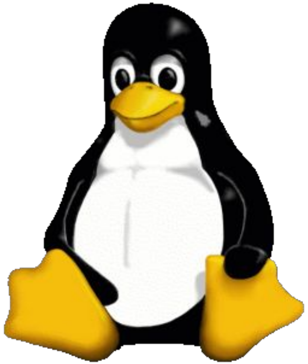
- Final Exam on 8/18 - see all logistics and practice materials here
- **No late days on Assignment 6** beyond the grace period
    - Hard deadline Thursday 8/17 at 11:59pm
- Retroactive citations due 8/18 at 11:59pm
    - See our honor code policy and the citation handout
    - Reach out to Amrita and Elyse with any questions, or ask your SL
- Fill out End of Quarter Survey by 5pm for a participation bonus!
    - Let us know what to focus on in the review session tomorrow
- Amrita’s OH: Wednesday hosted by SLs, Friday canceled

Relevant prior information – OS



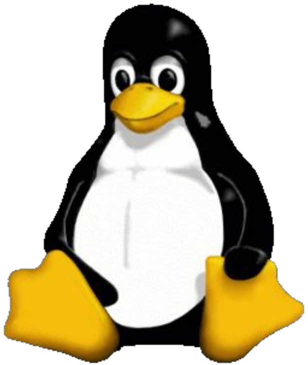
Relevant prior information – OS

- What is an *Operating System* (OS)?



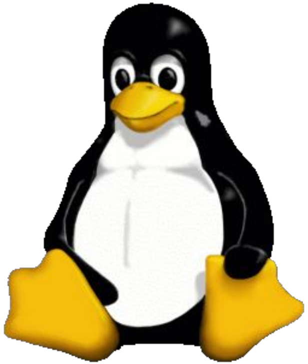
Relevant prior information – OS

- What is an *Operating System* (OS)?
    - Super great software that allows everything we do on the computer to communicate with the actual hardware!



Relevant prior information – OS

- What is an *Operating System* (OS)?
  - Super great software that allows everything we do on the computer to communicate with the actual hardware!
  - Supports the basic functions of a computer
    - Scheduling which processes get to run when
    - Manages memory – which program gets memory where?



Relevant prior information – OS

- What is an *Operating System* (OS)?
  - Super great software that allows everything we do on the computer to communicate with the actual hardware!
  - Supports the basic functions of a computer
    - Scheduling which processes get to run when
    - Manages memory – which program gets memory where?
  - Apple macOS, Linux OS, Microsoft Windows, Apple iOS, Android OS… and more!

Relevant prior information – OS

- What is an *Operating System* (OS)?
  - Super great software that allows everything we do on the computer to communicate with the actual hardware!
  - Supports the basic functions of a computer
    - Scheduling which processes get to run when
    - Manages memory – which program gets memory where?
  - Apple macOS, Linux OS, Microsoft Windows, Apple iOS, Android OS… and more!
- Want to learn more about operating systems? *Take CS111!*

Relevant prior information – Process

Relevant prior information – Process

- What is a *process*?

Relevant prior information – Process

- What is a *process*?
  - An instance of a program running on your computer!

Relevant prior information – Process

- What is a *process*?
  - An instance of a program running on your computer!
    - Running your CS106B Recursion Adventures assignment
    - Application processes: processes related to stuff you currently have open
      - Chrome browser
      - Spotify
      - Microsoft PowerPoint
    - Stuff going on in the background
      - Anti-virus software
      - OS processes – stuff the OS has to run to manage everything properly

Relevant prior information – Process

- What is a *process*?
    - An instance of a program running on your computer!
        - Running your CS106B Recursion Adventures assignment
        - Application processes: processes related to stuff you currently have open
            - Chrome browser
            - Spotify
            - Microsoft PowerPoint
        - Stuff going on in the background
            - Anti-virus software
            - OS processes – stuff the OS has to run to manage everything properly
- Activity Monitor on Mac
- ‘Ctrl’ + ‘Shift’ + ‘Esc’ and select Task Manager on Windows

Relevant prior information – Disk

- What is the *disk* on your computer?
  - Another place where we can store data

Relevant prior information – Disk

- What is the *disk* on your computer?
  - Another place where we can store data

Memory (RAM)	Disk

Relevant prior information – Disk

- What is the *disk* on your computer?
    - Another place where we can store data

Memory (RAM)	Disk
Fast to access, but less space (my laptop has 16 GB RAM)	Slower to access, but more space (my laptop has 995 GB of disk space)

Relevant prior information – Disk

- What is the *disk* on your computer?
  - Another place where we can store data

Memory (RAM)	Disk
Fast to access, but less space (my laptop has 16 GB RAM)	Slower to access, but more space (my laptop has 995 GB of disk space)
“Volatile memory” – lose access to whatever’s stored here upon power off	“Non-volatile memory” – data persists even upon power off

Relevant prior information – Disk

- What is the *disk* on your computer?
  - Another place where we can store data

Memory (RAM)	Disk
Fast to access, but less space (my laptop has 16 GB RAM)	Slower to access, but more space (my laptop has 995 GB of disk space)
“Volatile memory” – lose access to whatever’s stored here upon power off	“Non-volatile memory” – data persists even upon power off
More expensive	Less expensive

How can multiple processes share RAM?

- When you run your program, it’s not the only process running on your computer
    - Web browser, Slack, QT Creator, Etc.
- What would happen if all processes had access to the same chunk of memory?

How can multiple processes share RAM?

- When you run your program, it’s not the only process running on your computer
  - Web browser, Slack, QT Creator, Etc.
- What would happen if all processes had access to the same chunk of memory?
  - Can overwrite any memory
  - Can’t isolate processes

Goals of OS Memory Management

- **Multitasking**
  - Allow multiple processes to be memory-resident at once

Goals of OS Memory Management

- **Multitasking**
  - Allow multiple processes to be memory-resident at once
- **Transparency**
  - No process should need to know memory is shared. Each must run regardless of the number and/or locations of processes in memory

Goals of OS Memory Management

- **Multitasking**
  - Allow multiple processes to be memory-resident at once
- **Transparency**
  - No process should need to know memory is shared. Each must run regardless of the number and/or locations of processes in memory
- **Isolation**
  - Processes must not be able to corrupt each other

Goals of OS Memory Management

- **Multitasking**
  - Allow multiple processes to be memory-resident at once
- **Transparency**
  - No process should need to know memory is shared. Each must run regardless of the number and/or locations of processes in memory
- **Isolation**
  - Processes must not be able to corrupt each other
- **Efficiency**
  - Shouldn’t be degraded badly by sharing

Load-Time Relocation

Load-Time Relocation

- Idea: When it's time for a process to run, give it a set chunk of space in memory
    - ALL MEMORY belonging to that process goes there – stack, heap, etc.

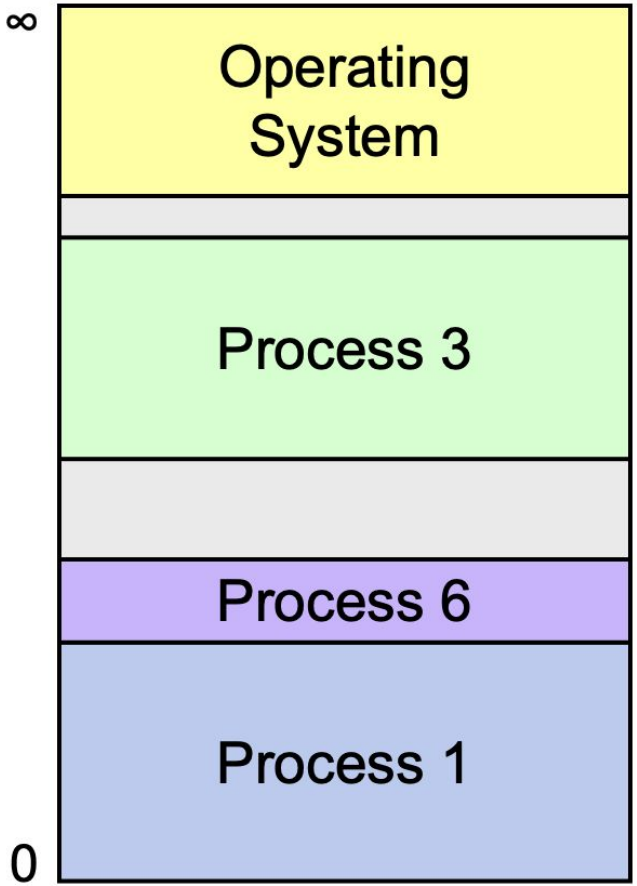
Load-Time Relocation

- Idea: When it's time for a process to run, give it a set chunk of space in memory
  - ALL MEMORY belonging to that process goes there – stack, heap, etc.
- Interesting fact – when a program is compiled, it is compiled assuming its memory starts at address 0
  - Must update the process' addresses when we load it to match its real starting address (i.e. shift addresses by some constant factor)

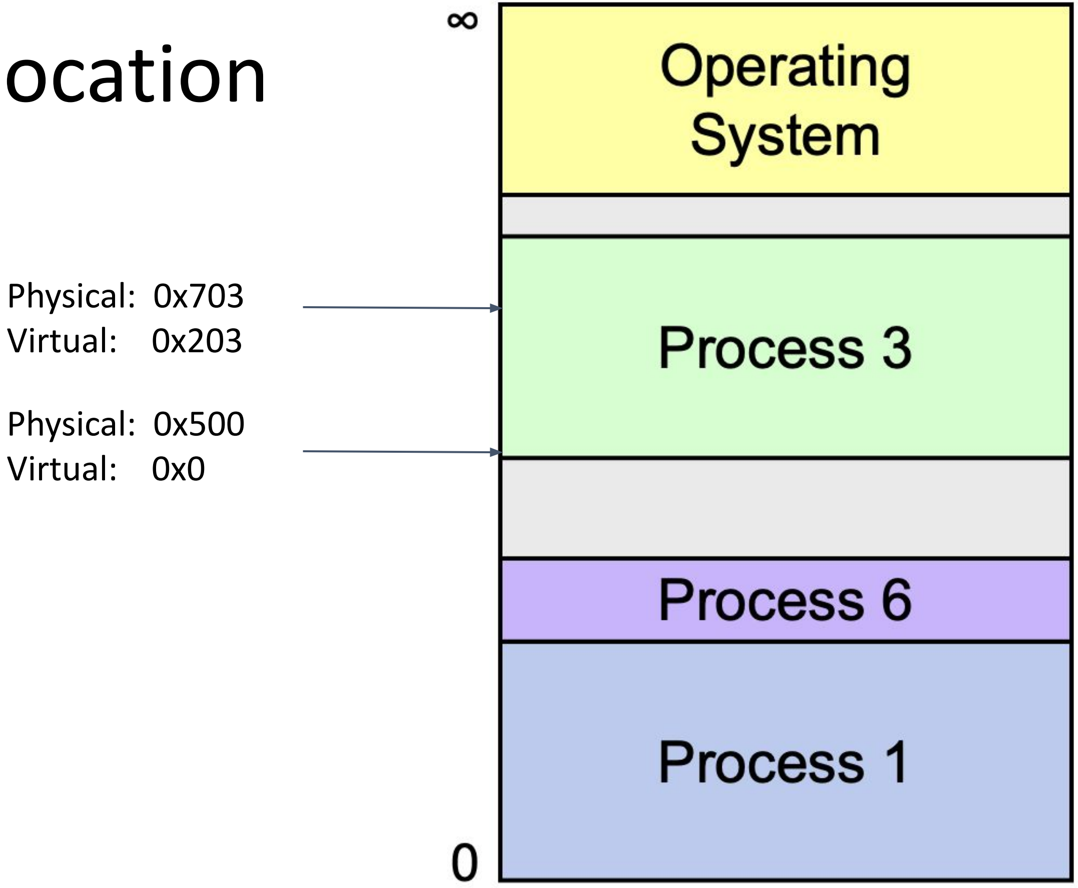
Load-Time Relocation

- Idea: When it's time for a process to run, give it a set chunk of space in memory
    - ALL MEMORY belonging to that process goes there – stack, heap, etc.
- Interesting fact – when a program is compiled, it is compiled assuming its memory starts at address 0
    - Must update the process' addresses when we load it to match its real starting address (i.e. shift addresses by some constant factor)
- Use first-fit or best-fit allocation to manage available memory

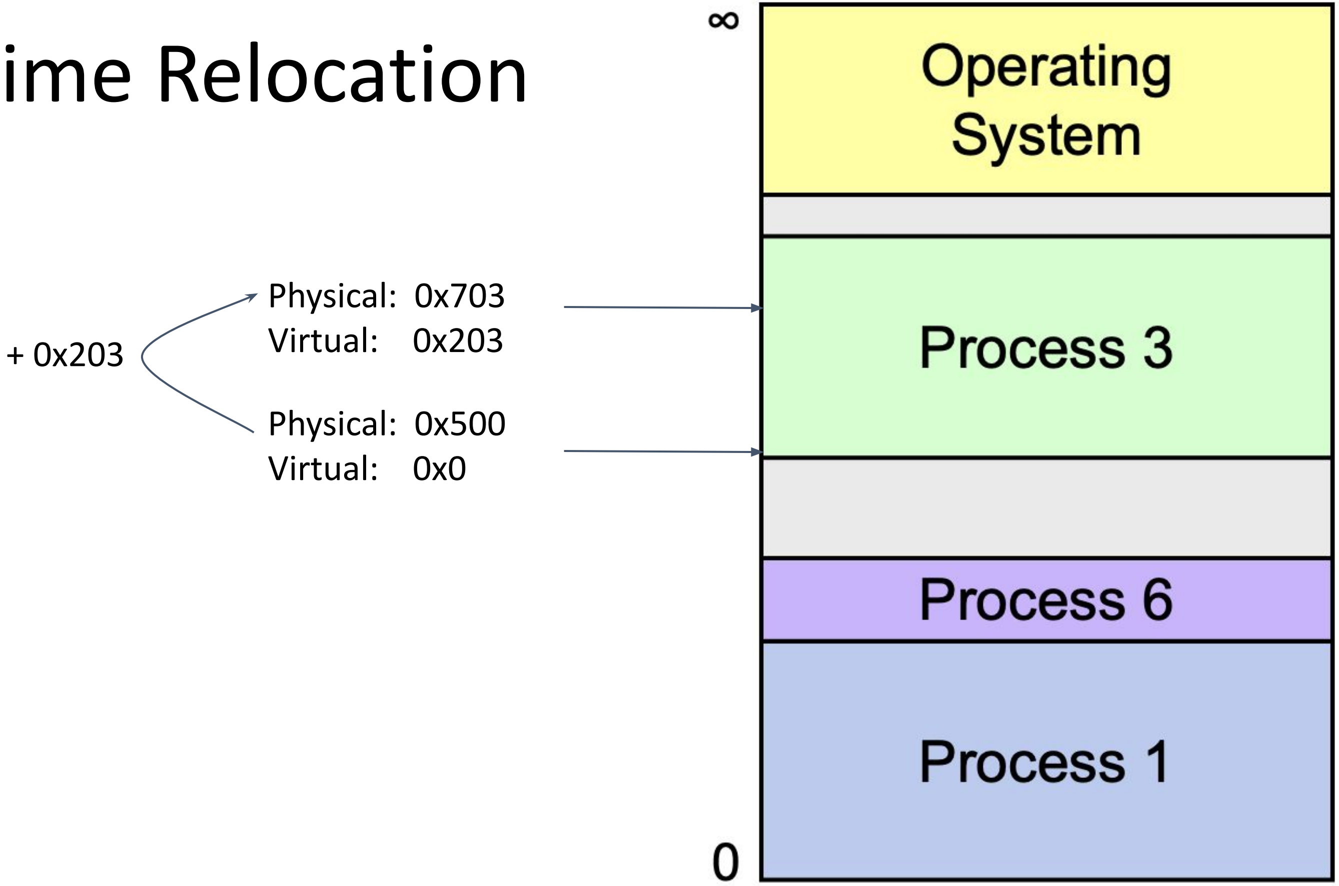
Load-Time Relocation



Load-Time Relocation



Load-Time Relocation

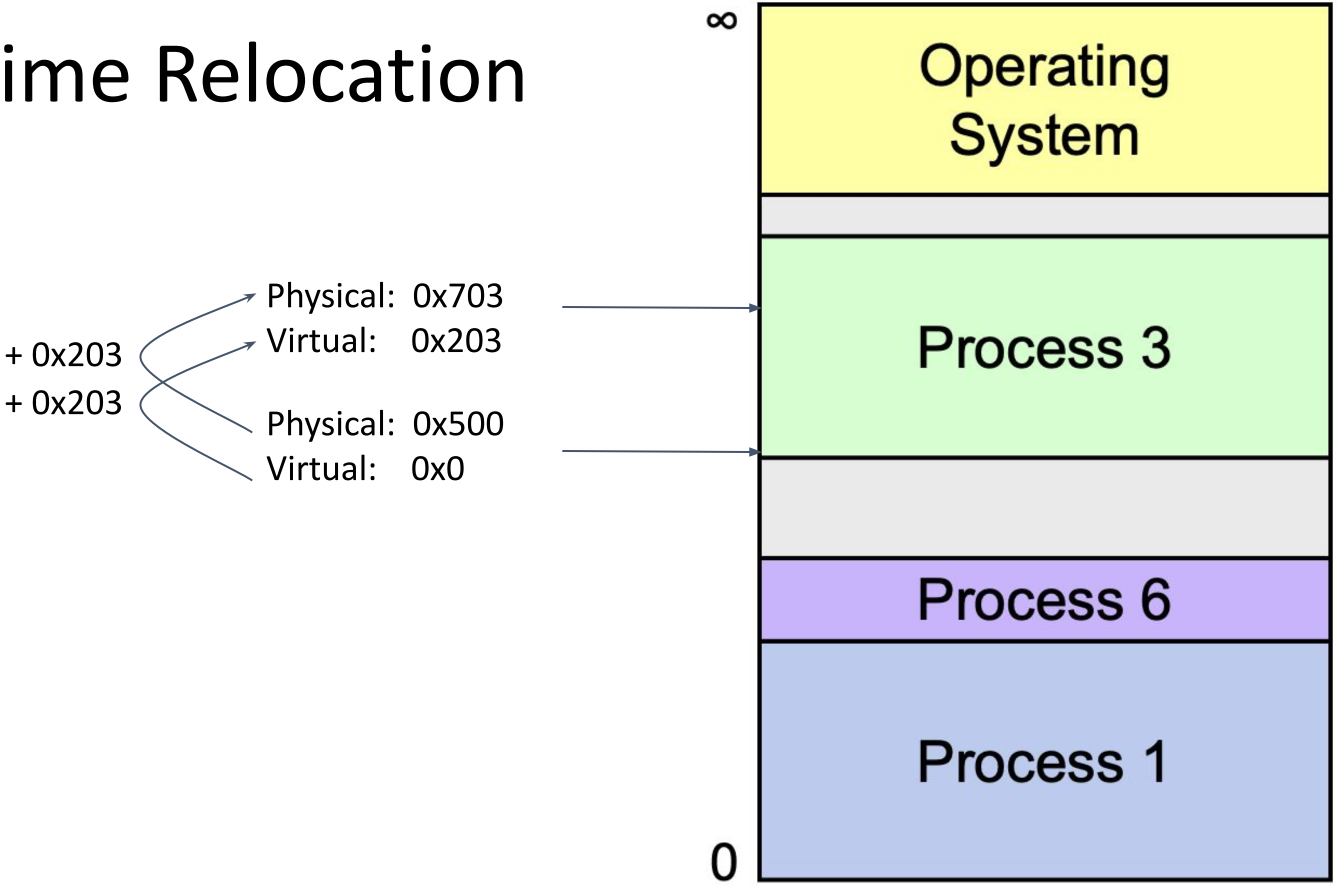


+ 0x203

Physical: 0x703
Virtual: 0x203

Physical: 0x500
Virtual: 0x0

Load-Time Relocation



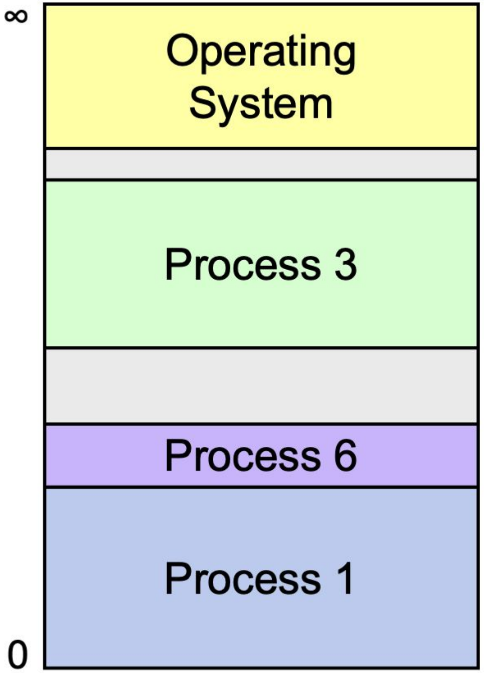
+ 0x203

+ 0x203

Physical: 0x703
Virtual: 0x203

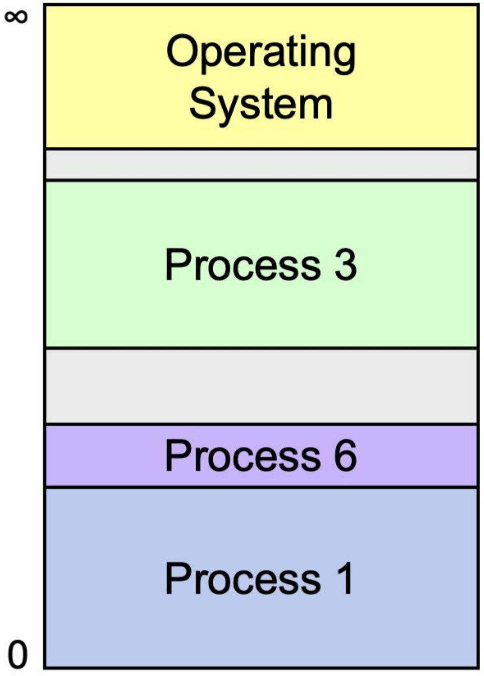
Physical: 0x500
Virtual: 0x0

Issues with Load-Time Relocation



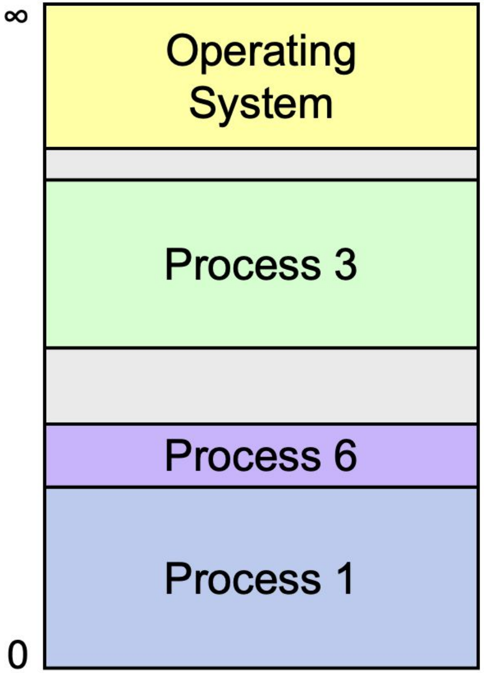
Issues with Load-Time Relocation

- No isolation: one process could invade another process' space (or the OS's – this is very bad!)



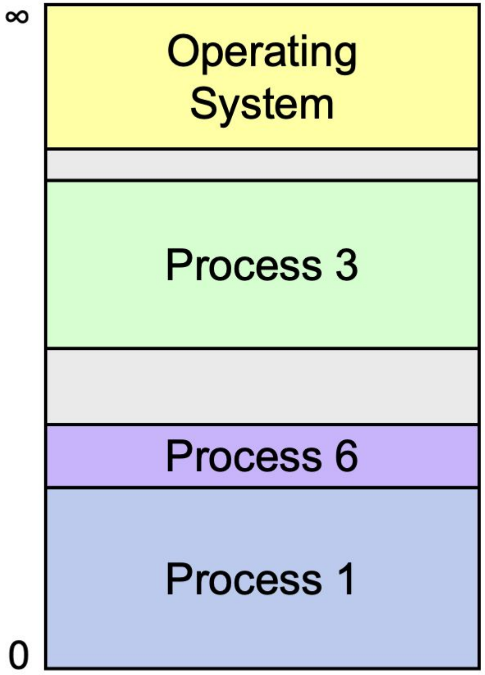
Issues with Load-Time Relocation

- No isolation: one process could invade another process’ space (or the OS’s – this is very bad!)
- Need to decide how much memory space a process deserves ahead of time (predict the future!)



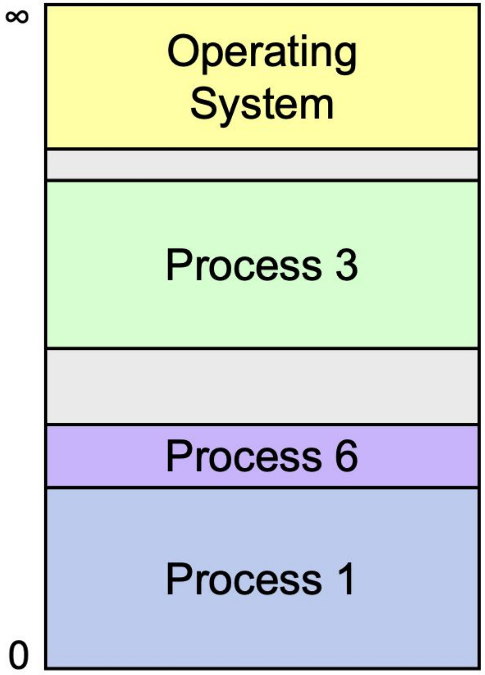
Issues with Load-Time Relocation

- No isolation: one process could invade another process’ space (or the OS’s – this is very bad!)
- Need to decide how much memory space a process deserves ahead of time (predict the future!)
- Potential fragmentation



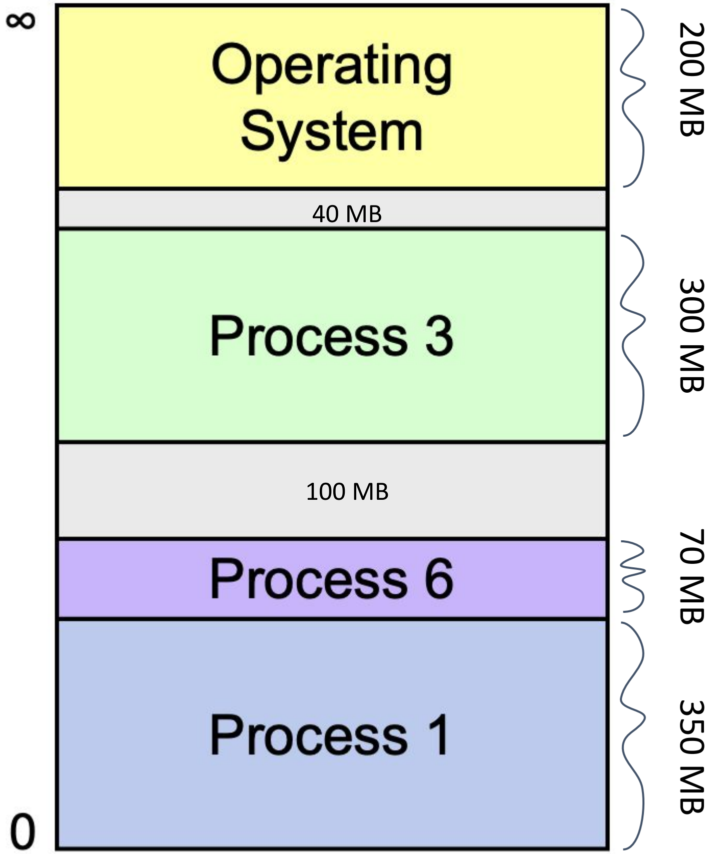
Issues with Load-Time Relocation

- No isolation: one process could invade another process' space (or the OS's – this is very bad!)
- Need to decide how much memory space a process deserves ahead of time (predict the future!)
- Potential fragmentation
- Can't grow regions if adjacent chunk of space is in use
- And many more…



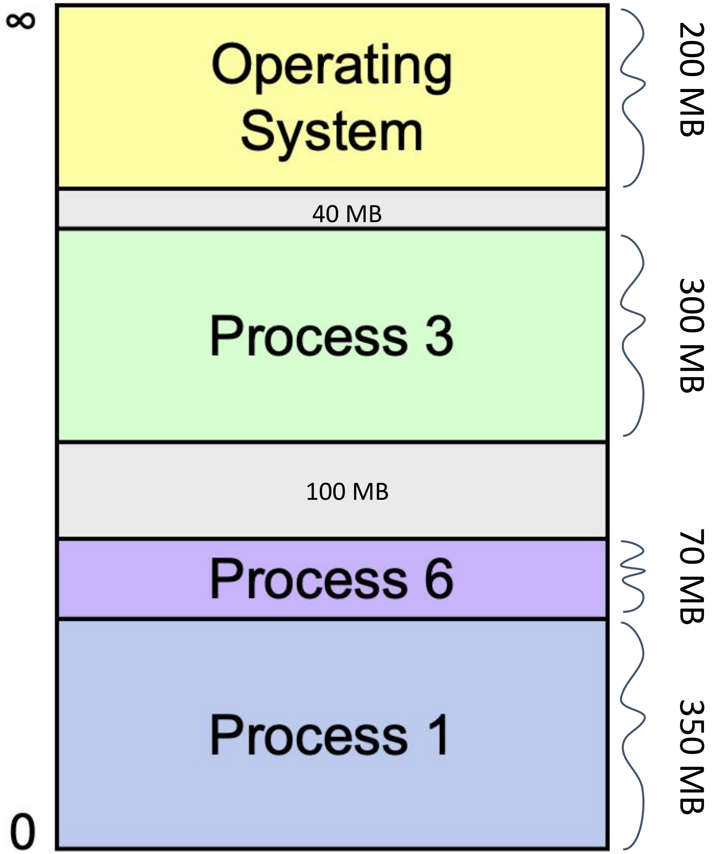
Aside: Fragmentation

- A problem that can occur with chunks of memory



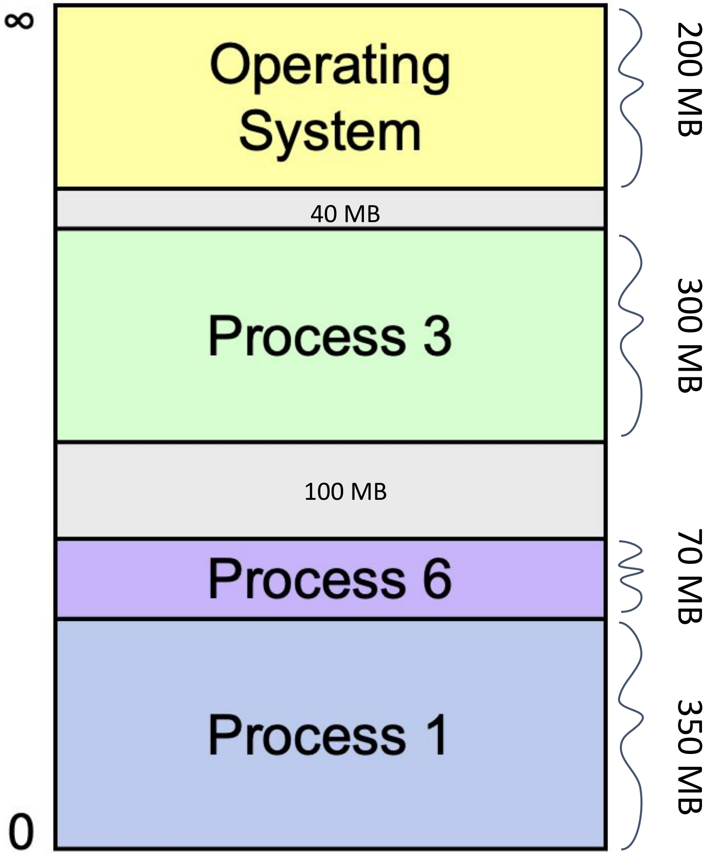
Aside: Fragmentation

- A problem that can occur with chunks of memory
- Example: what if we wanted to introduce process 7, and give it 130 MB of space?



Aside: Fragmentation

- A problem that can occur with chunks of memory
- Example: what if we wanted to introduce process 7, and give it 130 MB of space?
  - There isn’t one contiguous chunk of space with enough room! :(



Virtual Memory: a crazy idea!

- What if the operating system intercepted **every** memory reference and mapped it to a different place?
  - Spoiler: this is what actually happens

Virtual Memory: a crazy idea!

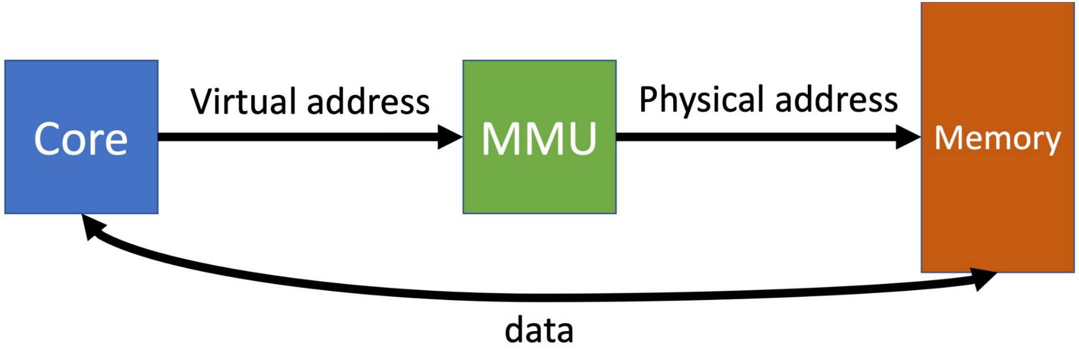
- What if the operating system intercepted **every** memory reference and mapped it to a different place?
    - Spoiler: this is what actually happens
- Wouldn’t that be really expensive?

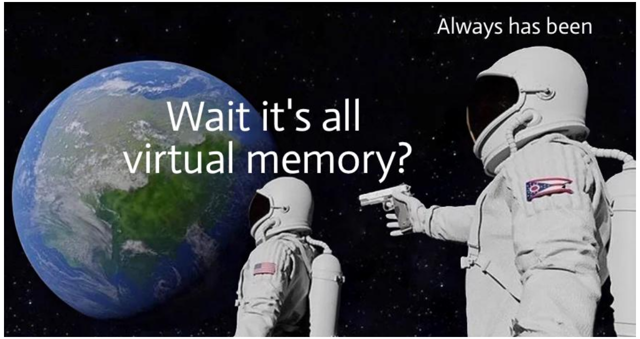
Virtual Memory: a crazy idea!

- What if the operating system intercepted **every** memory reference and mapped it to a different place?
  - Spoiler: this is what actually happens
- Wouldn’t that be really expensive?
  - Yes, and that’s why computers have a dedicated piece of hardware called the **Memory Management Unit (MMU)** to do memory address translation

So all of our memory address are fake?

Yeah…





How can we make every process *think* it has access to all of memory?

Idea 1: Base & Bound

- Each process has a base memory address and a maximum memory address

Base: 0x100
Bound: 0x200

Base: 0x300
Bound: 0x600

Base: 0x700
Bound: 0x900

Qt		
Base: 0x100	Base: 0x300	Base: 0x700
Bound: 0x200	Bound: 0x600	Bound: 0x900

What happens when QT creator accesses memory address 0x0?

Base: 0x100
Bound: 0x200

Base: 0x300
Bound: 0x600

Base: 0x700
Bound: 0x900

What happens when QT creator accesses memory address 0x0?

- Translate to physical address 0x100

Base: 0x100
Bound: 0x200

Base: 0x300
Bound: 0x600

Base: 0x700
Bound: 0x900

What happens when QT creator accesses memory address 0x0?

- Translate to physical address 0x100

What happens when Chrome accesses memory address 0x400?

Qt		
Base: 0x100	Base: 0x300	Base: 0x700
Bound: 0x200	Bound: 0x600	Bound: 0x900

What happens when QT creator accesses memory address 0x0?

- Translate to physical address 0x100

What happens when Chrome accesses memory address 0x400?

- Translate to physical address 0x700
- Error because it’s out of bounds!

Qt		
Base: 0x100	Base: 0x300	Base: 0x700
Bound: 0x200	Bound: 0x600	Bound: 0x900

What happens when QT creator accesses memory address 0x0?

- Translate to physical address 0x100

What happens when Chrome accesses memory address 0x400?

- Translate to physical address 0x700
- Error because it’s out of bounds!

What happens when Spotify wants more space?

Base: 0x100
Bound: 0x200

Base: 0x300
Bound: 0x600

Base: 0x700
Bound: 0x900

What happens when QT creator accesses memory address 0x0?

- Translate to physical address 0x100

What happens when Chrome accesses memory address 0x400?

- Translate to physical address 0x700
- Error because it’s out of bounds!

What happens when Spotify wants more space?

- Increase the bound!

What are the **benefits** of this approach?

Qt

Base: 0x100
Bound: 0x200

Base: 0x300
Bound: 0x600

Base: 0x700
Bound: 0x900

What are the **benefits** of this approach?

Base: 0x100 Bound: 0x200	Base: 0x300 Bound: 0x600	Base: 0x700 Bound: 0x900

- Inexpensive translation – just doing addition
- Doesn’t require much additional space – just base + bound
- The separation between virtual and physical addresses means we can move the physical memory location and simply update the base, or we could even swap memory to disk and copy it back later when it’s actually needed

What are the **drawbacks** of this approach?

Base: 0x100
Bound: 0x200

Base: 0x300
Bound: 0x600

Base: 0x700
Bound: 0x900

What are the **drawbacks** of this approach?

Qt		
Base: 0x100	Base: 0x300	Base: 0x700
Bound: 0x200	Bound: 0x600	Bound: 0x900

- One contiguous region per program
- Fragmentation
- Growing can only happen upwards with the bound

Can we do better?

Yes!! Although you probably guessed that…

Paging

- Key idea: Each process's virtual (and physical) memory is divided into fixed-size chunks called pages (common size is 4KB pages)

Paging

- Key idea: Each process's virtual (and physical) memory is divided into fixed-size chunks called pages (common size is 4KB pages)
- A "page" of virtual memory maps to a "page" of physical memory

Paging

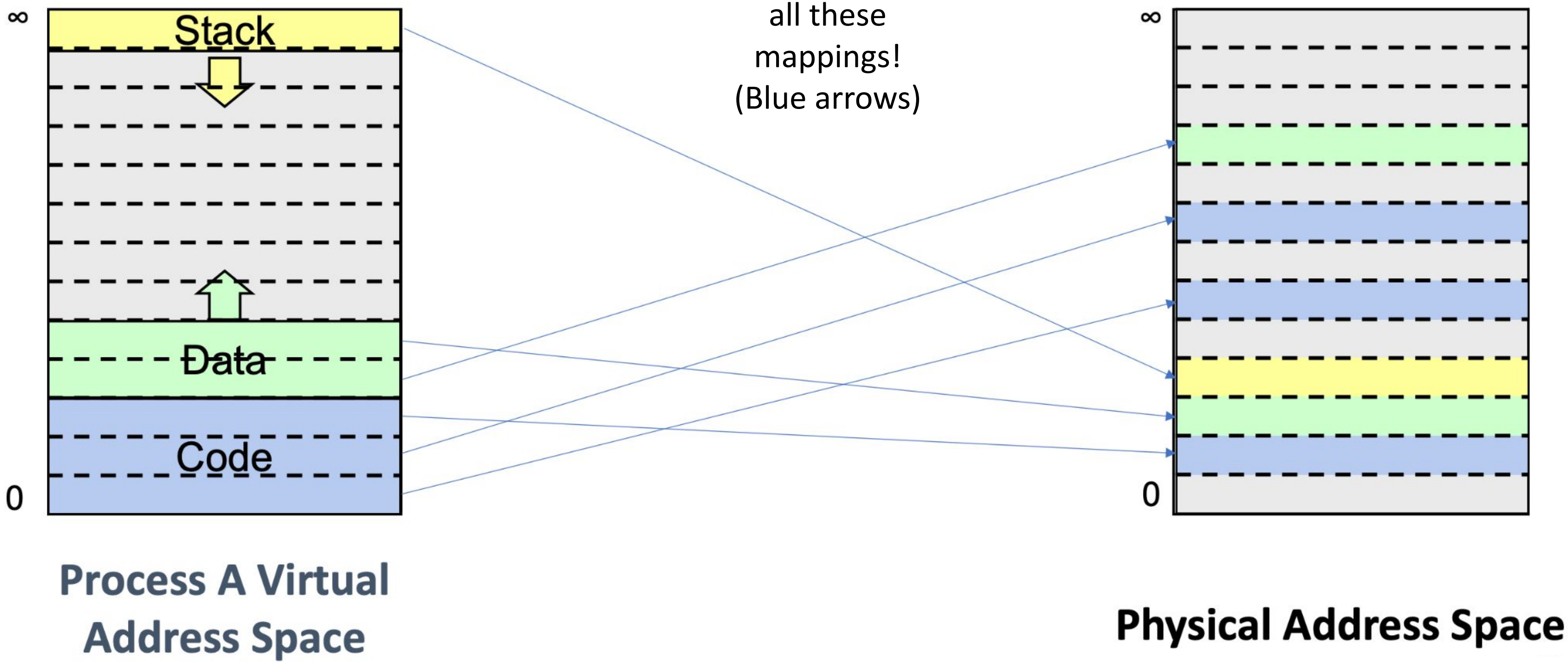
- Key idea: Each process’s virtual (and physical) memory is divided into fixed-size chunks called pages (common size is 4KB pages)
- A “page” of virtual memory maps to a “page” of physical memory
- The page number is an ID # for a page. We have virtual page numbers and physical page numbers

Paging

- Key idea: Each process's virtual (and physical) memory is divided into fixed-size chunks called pages (common size is 4KB pages)
- A "page" of virtual memory maps to a "page" of physical memory
- The page number is an ID # for a page. We have virtual page numbers and physical page numbers
    - Virtual page number 45 does not correspond necessarily to physical page 45 → page map to keep track of v-page to p-page mappings!!

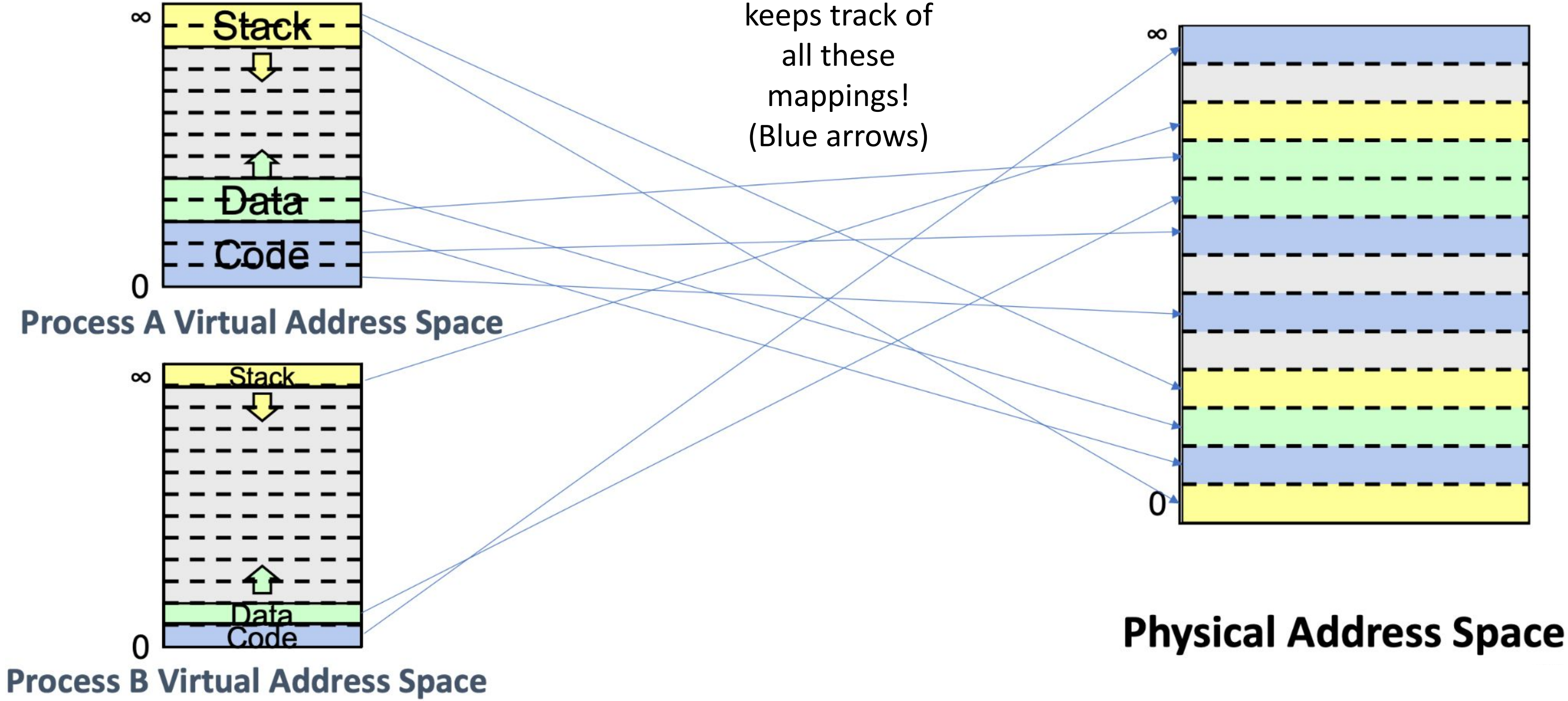
Paging

- Key idea: Each process's virtual (and physical) memory is divided into fixed-size chunks called pages (common size is 4KB pages)
- A "page" of virtual memory maps to a "page" of physical memory
- The page number is an ID # for a page. We have virtual page numbers and physical page numbers
    - Virtual page number 45 does not correspond necessarily to physical page 45 → page map to keep track of v-page to p-page mappings!!
- OS keeps track of which pages are in use by a process, and which are available to give out



The page map keeps track of all these mappings! (Blue arrows)

The page map keeps track of all these mappings! (Blue arrows)


∞
Stack
Data
Code
0
Process A Virtual Address Space
∞
Stack
Data
Code
0
Process B Virtual Address Space
∞
0
Physical Address Space

Page Map

- The page map maps virtual pages to physical pages

Page Map

- The page map maps virtual pages to physical pages
    - Can use this to translate virtual addresses to physical ones – you'll see in a sec!

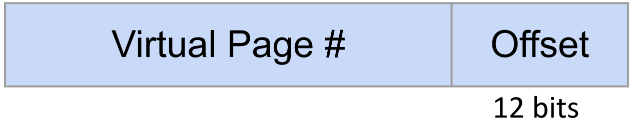
Page Map

- The page map maps virtual pages to physical pages
  - Can use this to translate virtual addresses to physical ones – you'll see in a sec!
- You can think of this map the same way you think of the Stanford Library Map you've used all quarter long!

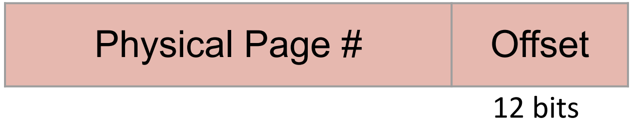
Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703

Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703

Virtual Address

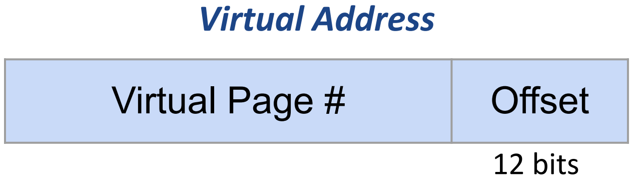


Physical Address



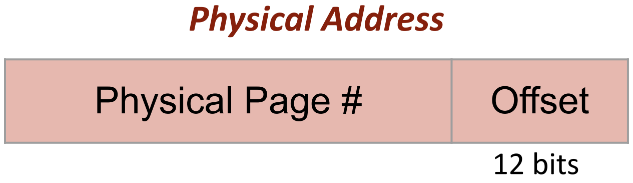
Virtual Page Number	Physical Page Number
…	…

Our pages are 4KB (4096 bytes). Offsets will therefore be 0 to 4095 byte offsets. We don't need to get into the nitty-gritty of hexadecimal, but just know that the offset only needs to be 12 bits to store numbers in that range.


Virtual Address

Virtual Page # | Offset

12 bits

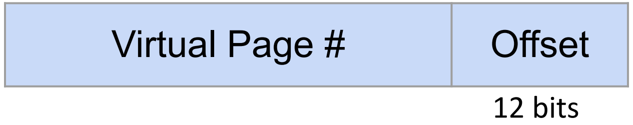

Physical Address

Physical Page # | Offset

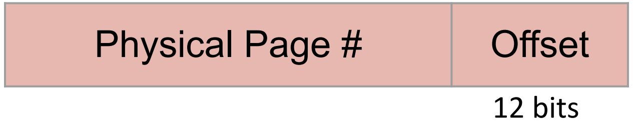
12 bits

Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703

Virtual Address



Physical Address



Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703

Virtual Address

0x2	0x238
Virtual Page #	12 bits

0x2238

Physical Address

???	???
Physical Page #	12 bits

Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703

Virtual Address

0x2	0x238
Virtual Page #	12 bits

Physical Address

???	???
Physical Page #	12 bits

0x2238

Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703

Virtual Address

0x2	0x238
Virtual Page #	12 bits

Physical Address

0x905	???
Physical Page #	12 bits

0x2238

Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703

Virtual Address

0x2	0x238
Virtual Page #	12 bits

0x2238

Physical Address

0x905	0x238
Physical Page #	12 bits

Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703

Virtual Address

0x2	0x238
Virtual Page #	12 bits

0x2238

Physical Address

0x905	0x238
Physical Page #	12 bits

0x905238

You try it!

Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703

Virtual Address

0x3	0x123
Virtual Page #	12 bits

0x3123

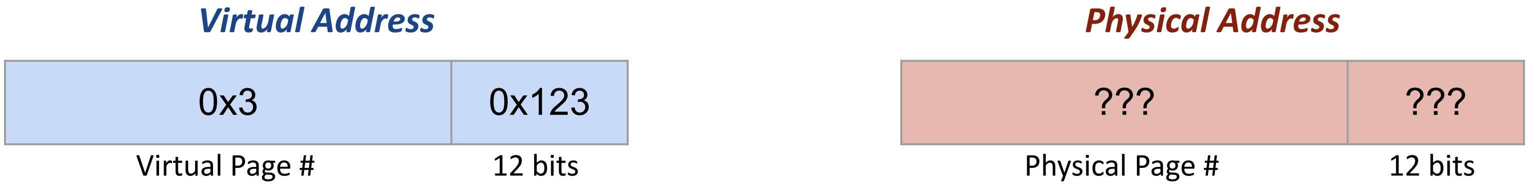
Physical Address

???	???
Physical Page #	12 bits

0x???????

You try it!

Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703



Virtual Address

0x3	0x123
Virtual Page #	12 bits

Physical Address

???	???
Physical Page #	12 bits

0x3123

0x???????

You try it!

Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703

Virtual Address

0x3	0x123
Virtual Page #	12 bits

0x3123

Physical Address

1231	???
Physical Page #	12 bits

0x???????

You try it!

Virtual Page Number	Physical Page Number
…	…
0x3	0x1231
0x2	0x905
0x1	0x1212
0x0	0x703

Virtual Address

0x3	0x123
Virtual Page #	12 bits

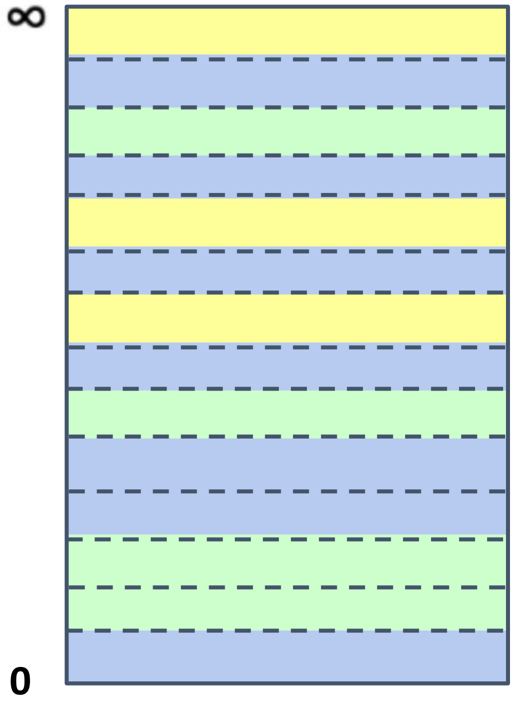
0x3123

Physical Address

1231	123
Physical Page #	12 bits

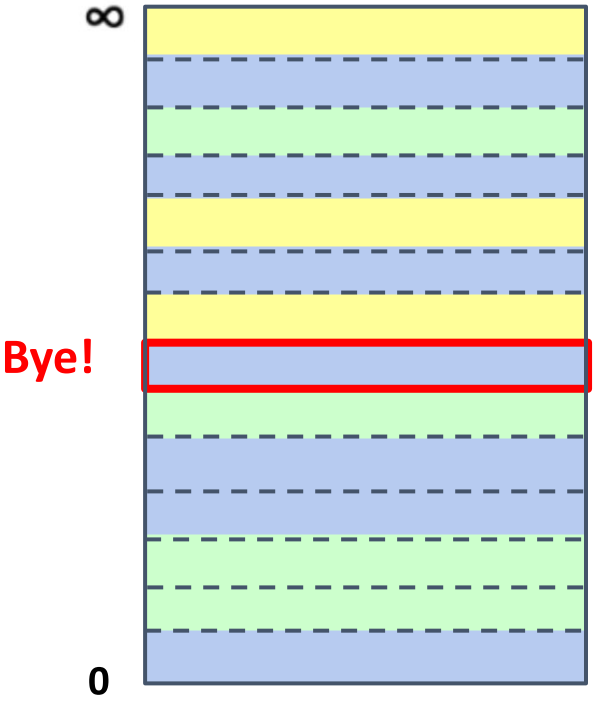
0x1231123

What if there's no physical memory left?

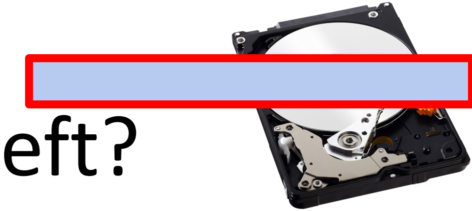


What if there's no physical memory left?

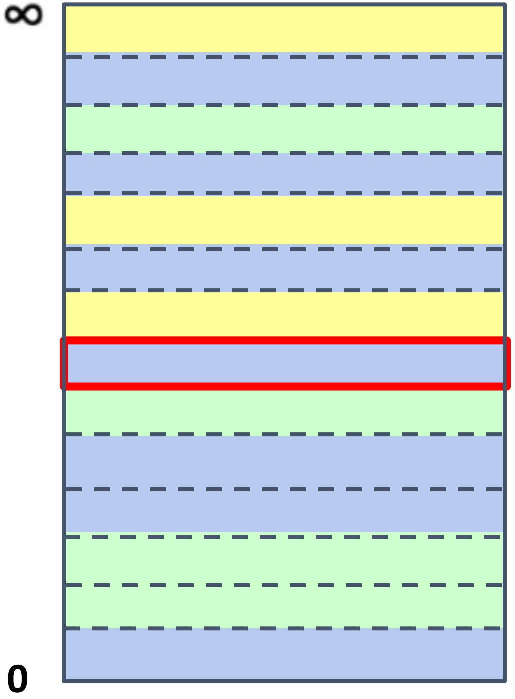
- Choose a page to swap out
  - How do we choose this page? Ask us after class/take CS111!



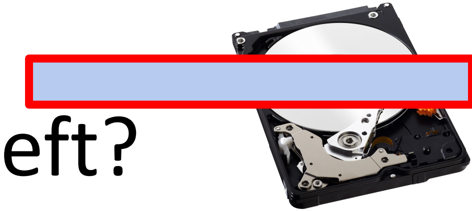
What if there's no physical memory left?



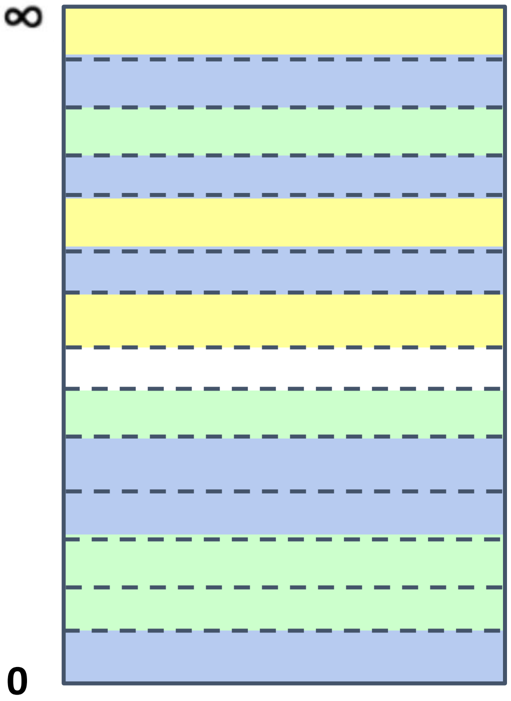
- Choose a page to swap out
  - How do we choose this page? Ask us after class/take CS111!
- Write this page to disk (to store it for the time being)



What if there’s no physical memory left?



- Choose a page to swap out
  - How do we choose this page? Ask us after class/take CS111!
- Write this page to disk (to store it for the time being)
- Mark that old page map entry as not present



Aside: the “other info” in the page map

Virtual Page Number	Physical Page Number	Present in RAM?
…	…	…
0x3	0x1231	True
0x2	0x905	False
0x1	0x1212	True
0x0	0x703	True

This column holds the information as to whether or not that page information is currently in memory–true if it is, false if it’s currently on disk.

Aside: the “other info” in the page map

Virtual Page Number	Physical Page Number	Present in RAM?
…	…	…
0x3	0x1231	True
0x2	0x905	False
0x1	0x1212	True
0x0	0x703	True

This column holds the information as to whether or not that page information is currently in memory–true if it is, false if it’s currently on disk.

When we swap a page to disk, say that blue one we just kicked off, we’d change the Present in RAM? bool for that entry to be false.

Aside: the “other info” in the page map

Virtual Page Number	Physical Page Number	Present in RAM?
…	…	…
0x3	0x1231	True
0x2	0x905	False
0x1	0x1212	True
0x0	0x703	True

This column holds the information as to whether or not that page information is currently in memory–true if it is, false if it’s currently on disk.

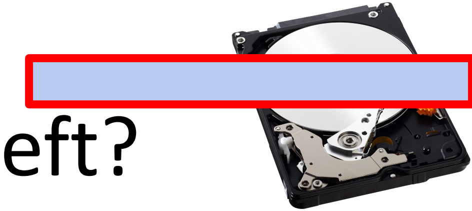
When we swap a page to disk, say that blue one we just kicked off, we’d change the Present in RAM? bool for that entry to be false.

Aside: the “other info” in the page map

Virtual Page Number	Physical Page Number	Present in RAM?
…	…	…
0x3	0x1231	**False**
0x2	0x905	False
0x1	0x1212	True
0x0	0x703	True

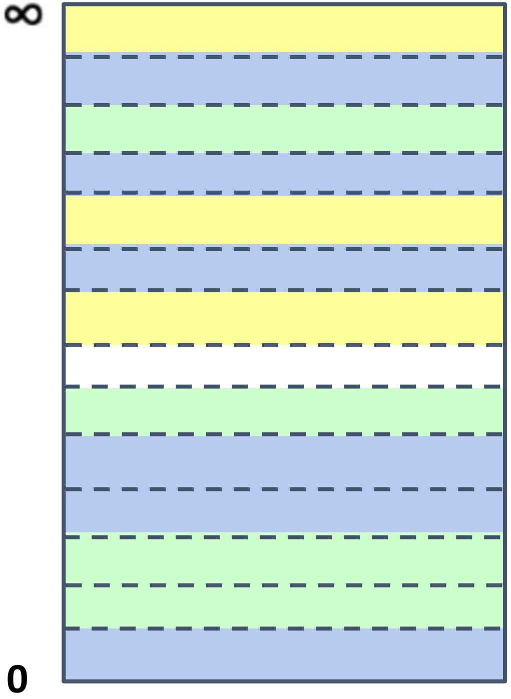
This column holds the information as to whether or not that page information is currently in memory–true if it is, false if it’s currently on disk.

When we swap a page to disk, say that blue one we just kicked off, we’d change the Present in RAM? bool for that entry to be false.

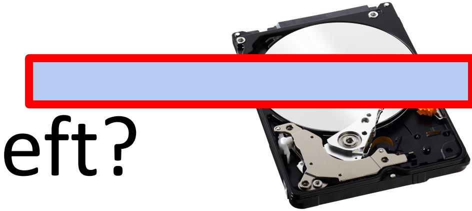


What if there's no physical memory left?

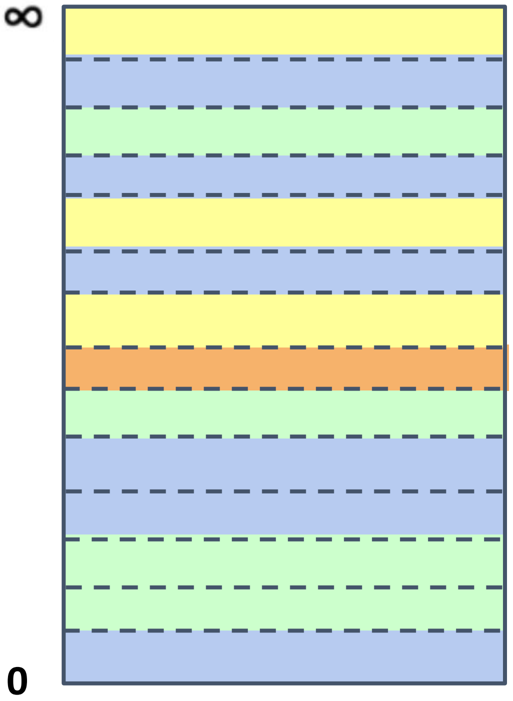
- Choose a page to swap out
  - How do we choose this page? Ask us after class/take CS111!
- Write this page to disk (to store it for the time being)
- Mark that old page map entry as not present



What if there’s no physical memory left?



- Choose a page to swap out
  - How do we choose this page? Ask us after class/take CS111!
- Write this page to disk (to store it for the time being)
- Mark that old page map entry as not present
- Update the new page map entry to map to this physical page, and to be present!



Update page map for new page!

Virtual Page Number	Physical Page Number	Present in RAM?
0x4	0x1231	True
0x3	0x1231	False
0x2	0x905	False
0x1	0x1212	True
0x0	0x703	True

This is the new orange page!

This is the page we just kicked off.

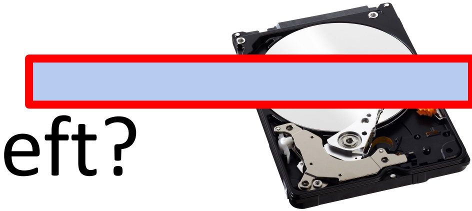
Update page map for new page!

Virtual Page Number	Physical Page Number	Present in RAM?
0x4	**0x1231**	True
0x3	**0x1231**	False
0x2	0x905	False
0x1	0x1212	True
0x0	0x703	True

This is the new orange page!

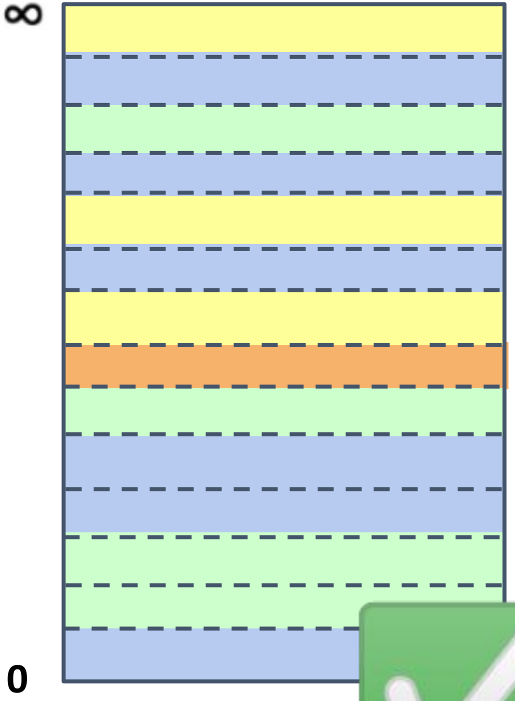
This is the page we just kicked off.

Note: the physical page number is the same! This makes sense–only one of them is present at a time, so it's okay that that they share the same physical address.



What if there's no physical memory left?

- Choose a page to swap out
  - How do we choose this page? Ask us after class/take CS111!
- Write this page to disk (to store it for the time being)
- Mark that old page map entry as not present
- Update the new page map entry to map to this physical page, and to be present!



What happens when we access a page that has been kicked out?

Virtual Page Number	Physical Page Number	Present in RAM?
…	…	…
0x3	0x1231	True
0x2	0x905	False
0x1	0x1212	True
0x0	0x703	True

Let’s saw we want to access virtual page **0x2**

- See that the page map entry is not present
- Check “disk swap” region for the page
  - If it’s not there, throw an error
- Once we find it, swap this page back into memory and kick out something else

Swap Demo

Recap

- Introduction to the Operating System
- Why do we need virtual memory?
  - Need to isolate processes!
- How can we implement virtual memory?
  - Base & Bound
  - Paging
- What happens when we run out of space in memory?
  - Swap to disk!

Final Review Session tomorrow!

Remember to fill out this form to let us know which topics to focus on