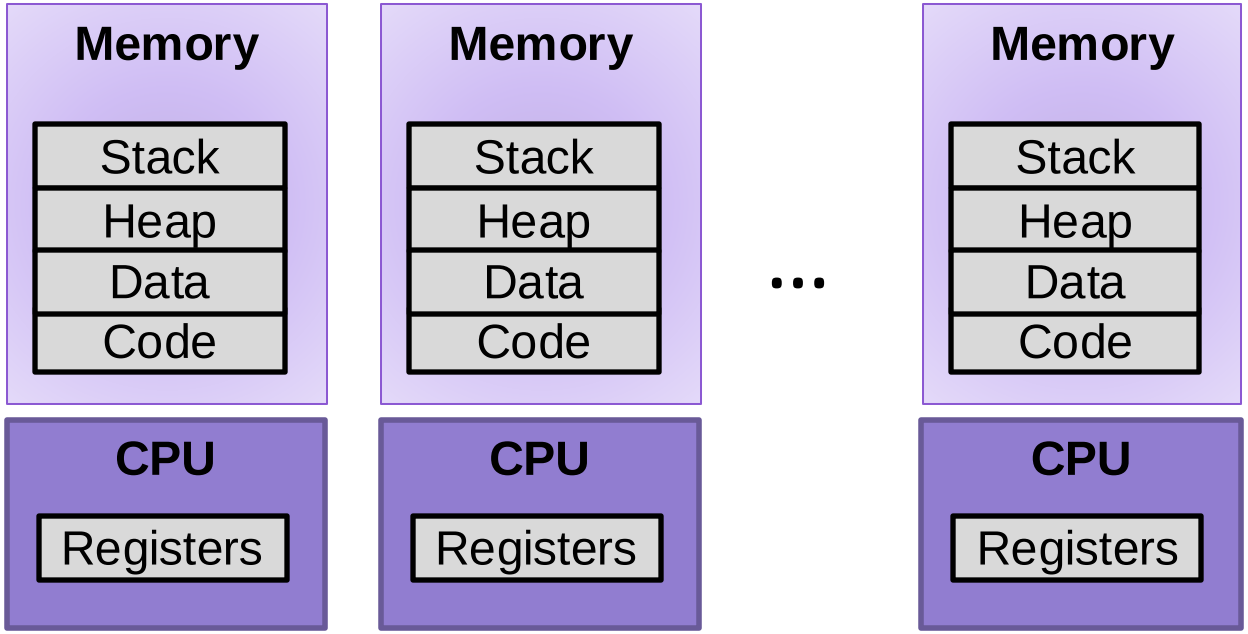
Lecture 16: Virtual Memory

CS 105 Fall 2024

Multiprocessing: The Illusion

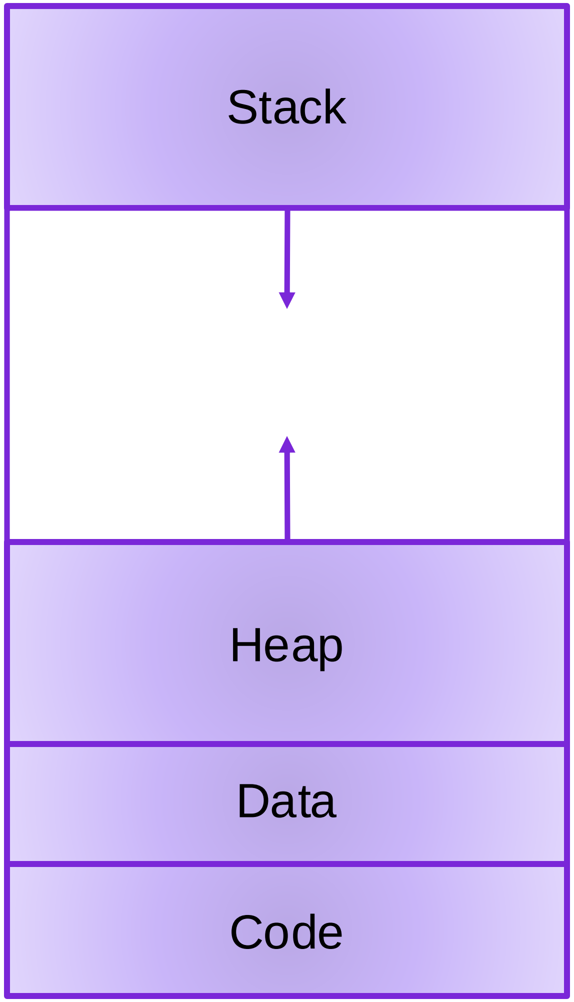


- Process provides each program with two key abstractions:
  - ***Logical control flow***
    - Each program seems to have exclusive use of the CPU
    - Provided by kernel mechanism called ***context switching***
  - ***Private address space***
    - Each program seems to have exclusive use of main memory.
    - Provided by kernel mechanism called ***virtual memory***

Multiprocessing: The Reality

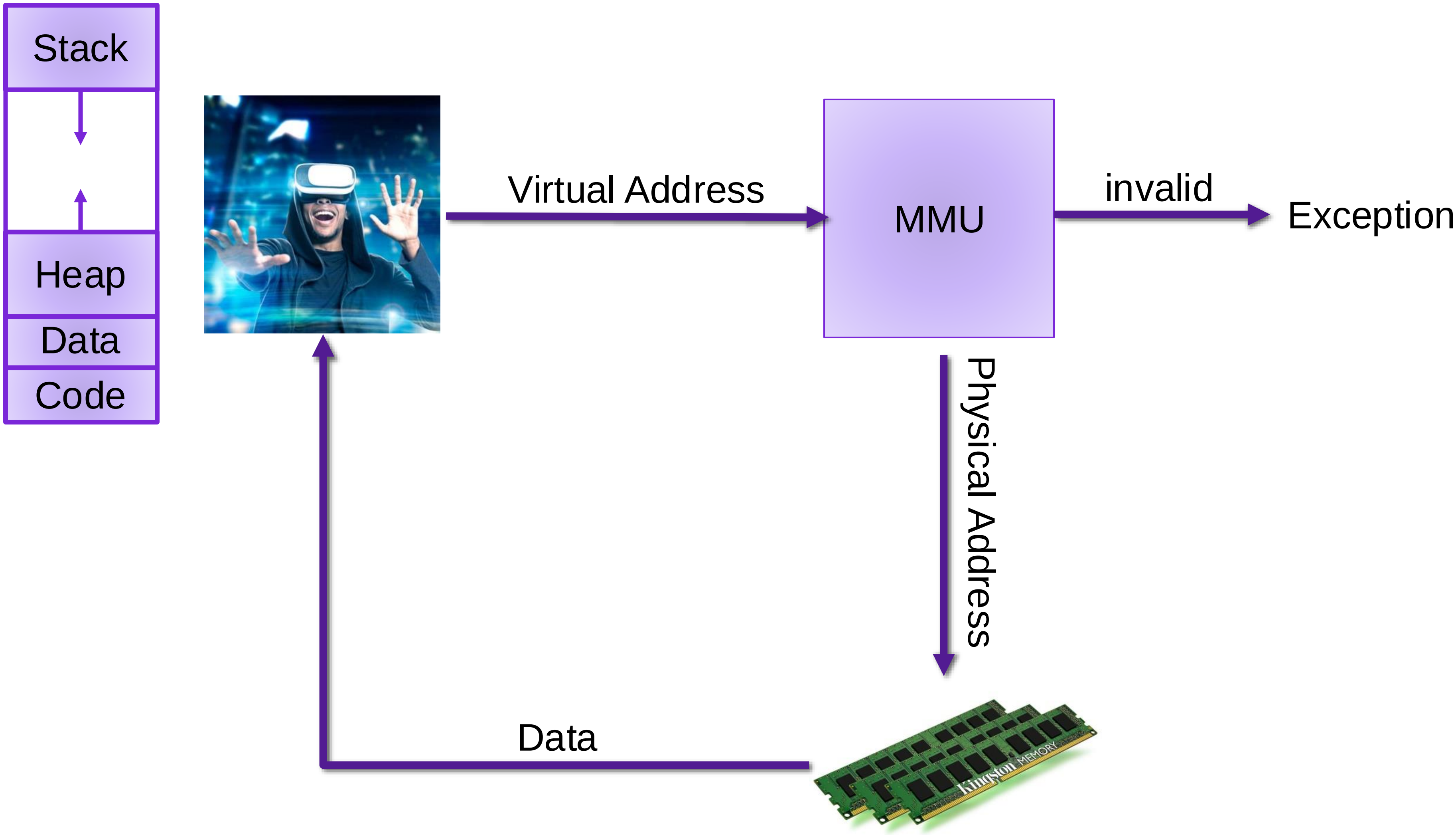
- Computer runs many processes simultaneously
- Running program “top” on Mac
  - System has 123 processes, 5 of which are active
  - Identified by Process ID (PID)

Virtual Memory Goals

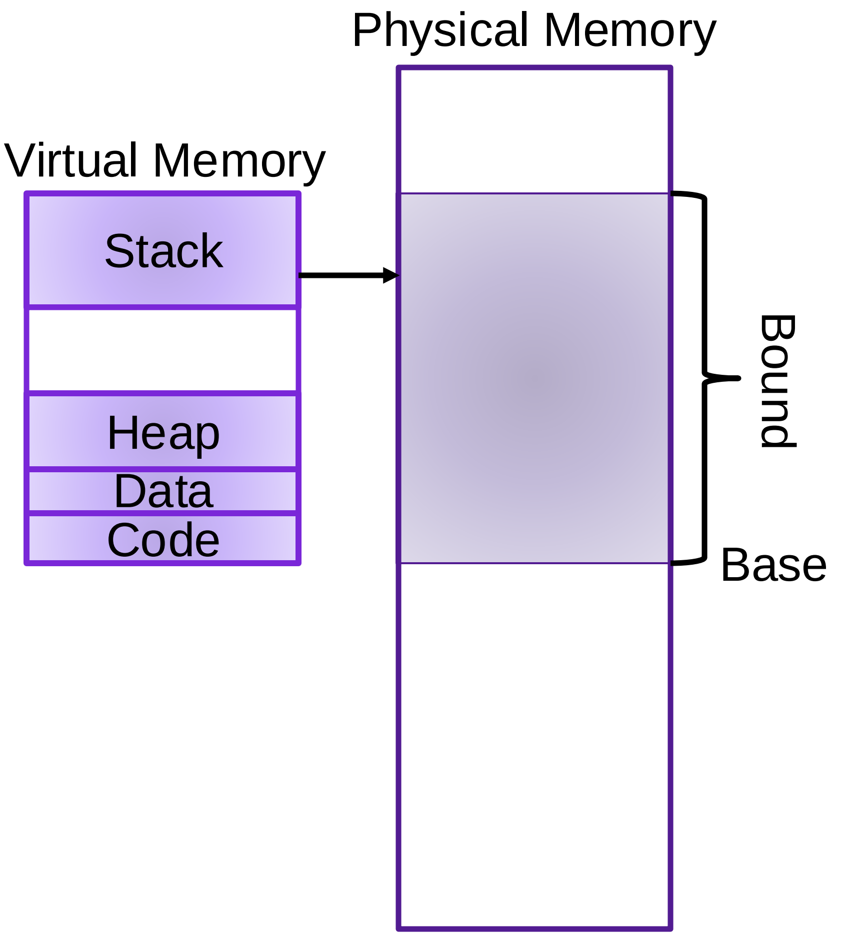


- **Isolation:** don't want different process states collided in physical memory
- **Efficiency:** want fast reads/writes to memory
- **Sharing:** want option to overlap for communication
- **Utilization:** want best use of limited resource
- **Virtualization:** want to create illusion of more resources

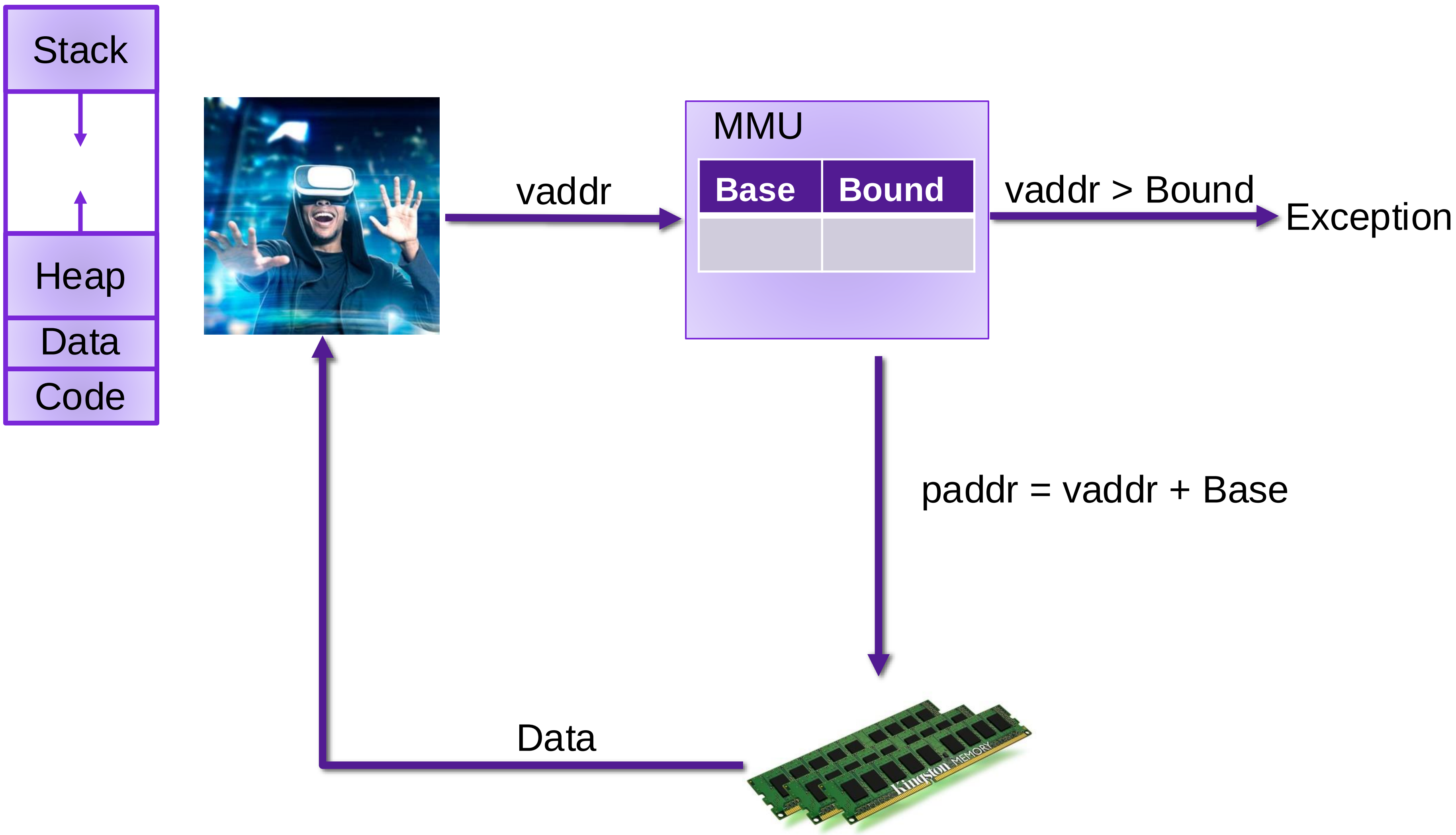
Address Translation



Base-and-Bound



Base-and-Bound



Exercise 1: Base-and-Bound

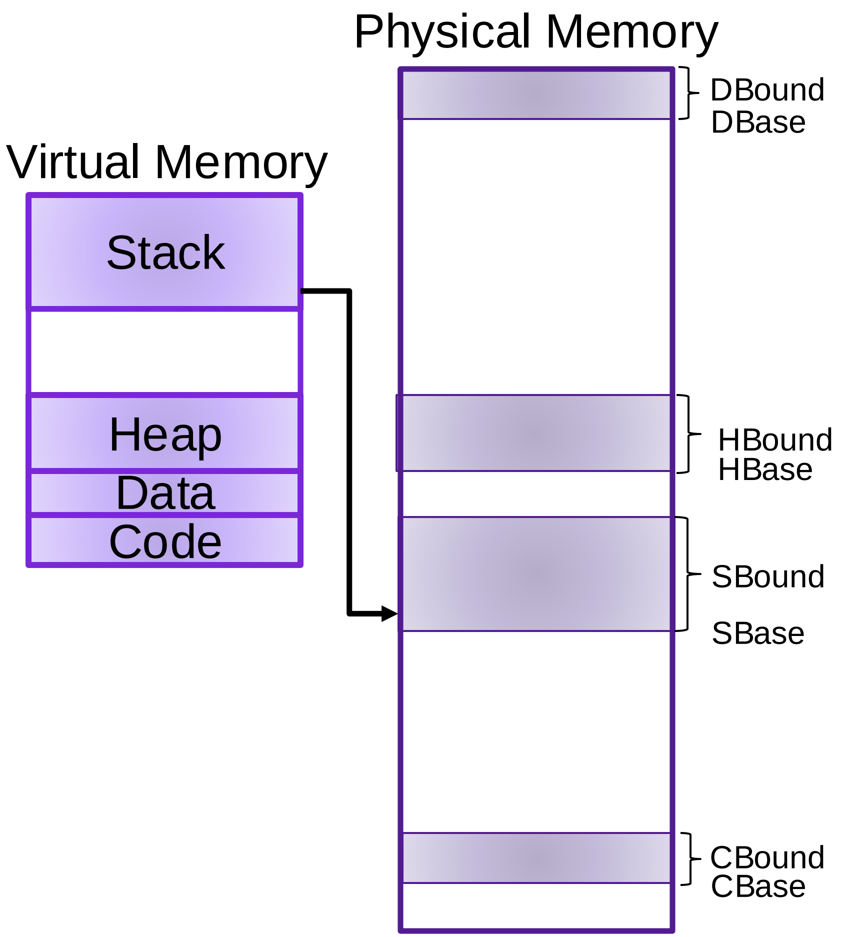
Assume that you are currently executing a process P with Base 0x1234 and Bound 0x100.

- What is the physical address that corresponds to the virtual address 0x47?
- What is the physical address that corresponds to the virtual address 0x123?

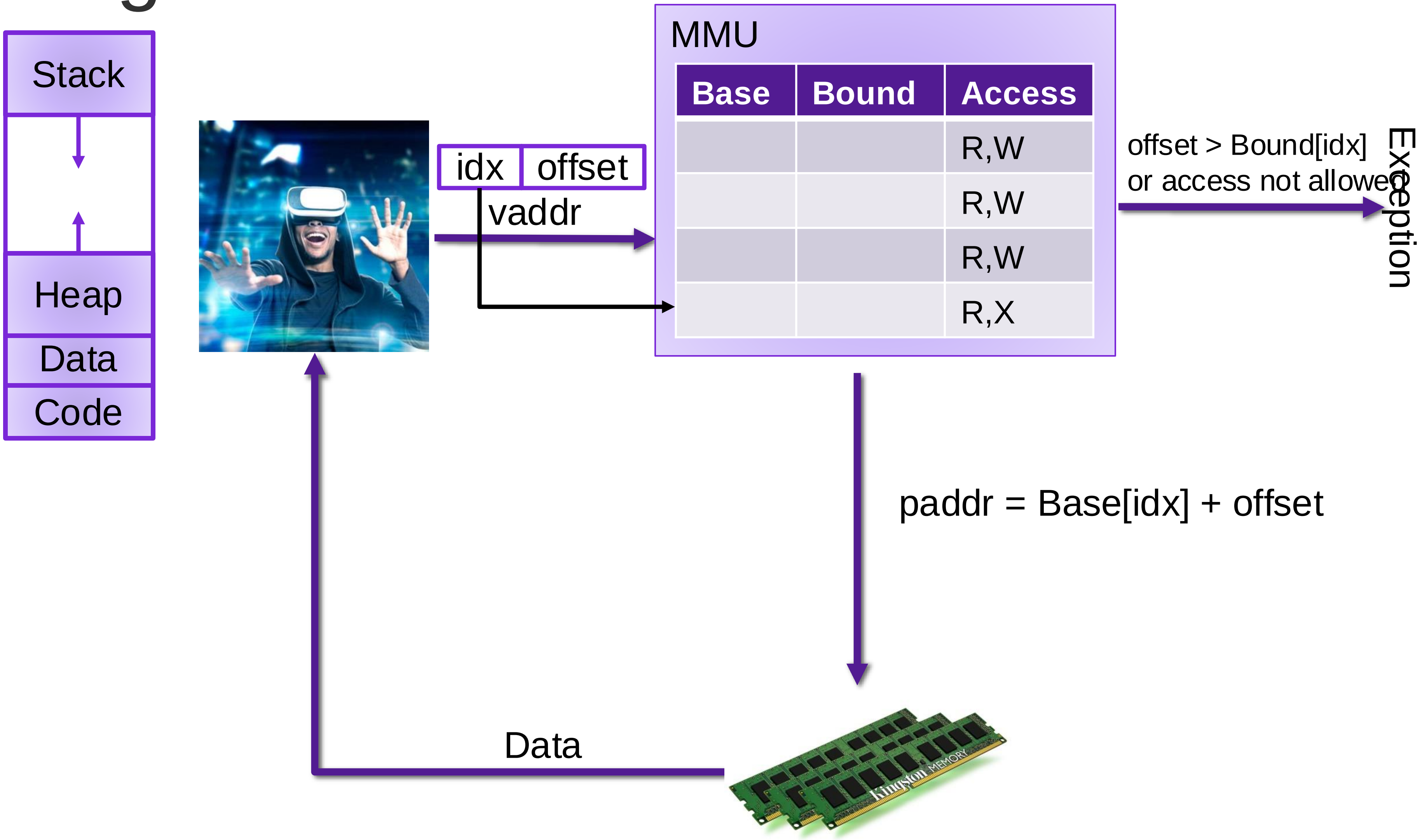
Evaluating Base-and-Bound

- **Isolation:** don't want different process states collided in physical memory 
- **Efficiency:** want fast reads/writes to memory 
- **Sharing:** want option to overlap for communication 
- **Utilization:** want best use of limited resource 
- **Virtualization:** want to create illusion of more resources 

Segmentation



Segmentation



Exercise 2: Segmentation

Assume that you are currently executing a process P with the following segment table:

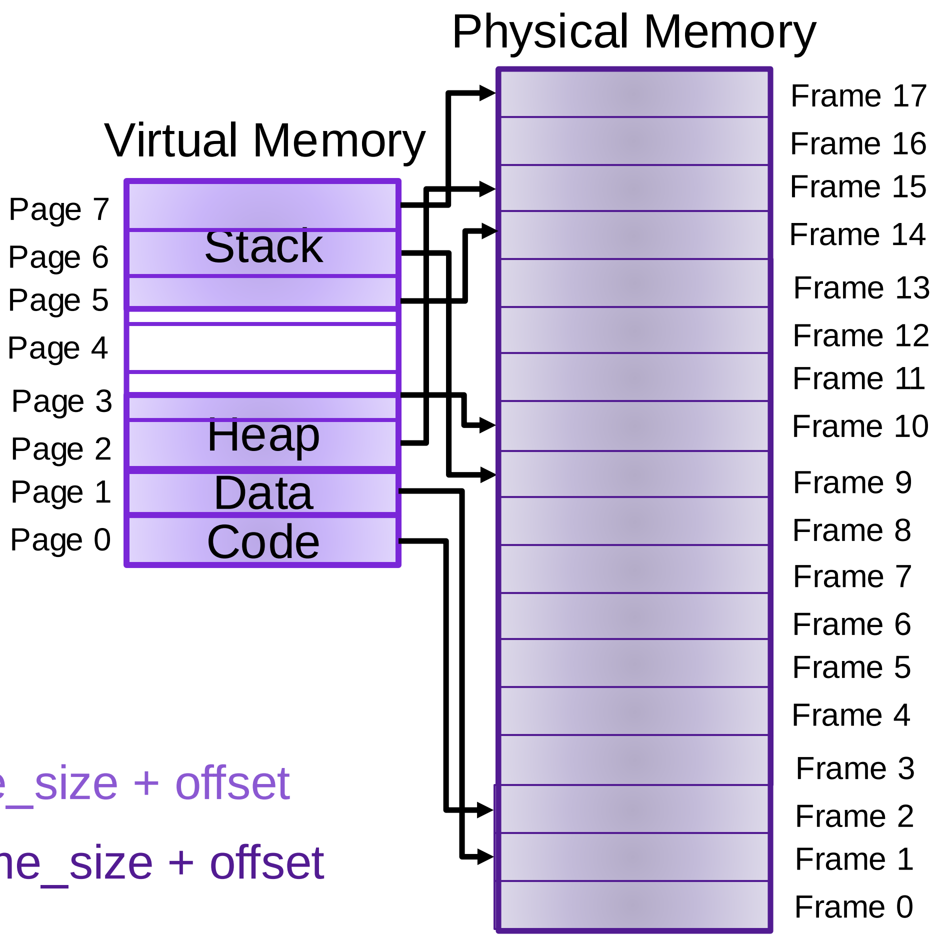
Base	Bound	Access
0x4747	0x80	R,W
0x2424	0x40	R,W
0x0023	0x80	R,W
0x1000	0x200	R,X

- What is the physical address that corresponds to the virtual address 0x001?
- What is the physical address that corresponds to the virtual address 0xD47?

Evaluating Segmentation

- **Isolation:** don’t want different process states collided in physical memory 
- **Efficiency:** want fast reads/writes to memory 
- **Sharing:** want option to overlap for communication 
- **Utilization:** want best use of limited resource 
- **Virtualization:** want to create illusion of more resources 

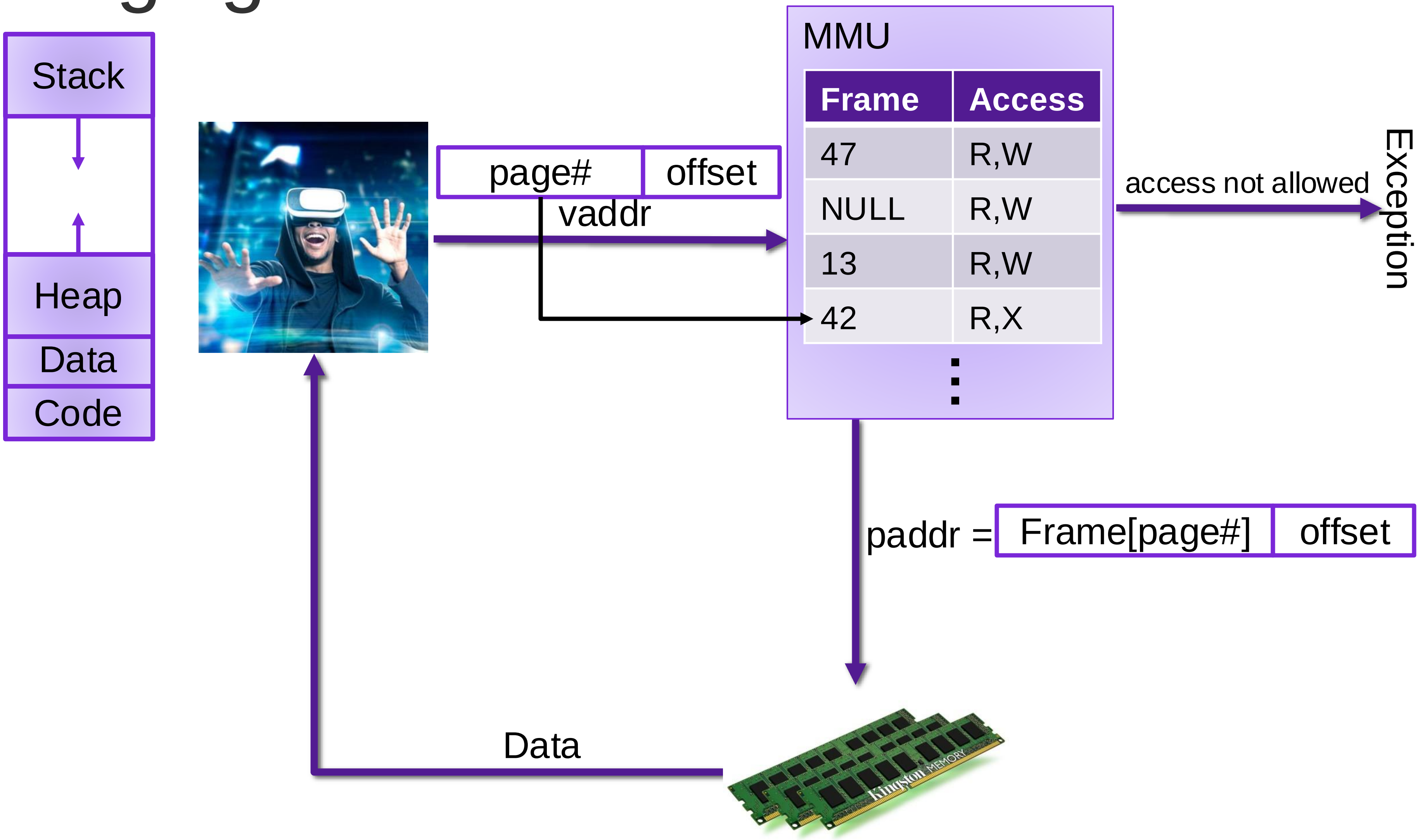
Paging



vaddr = page_num*page_size + offset

paddr = frame_num*frame_size + offset

Paging


Stack
Heap
Data
Code
page#
offset
vaddr
MMU
Frame
Access
47
R,W
NULL
R,W
13
R,W
42
R,X
access not allowed
Exception
paddr =
Frame[page#]
offset
Data
Kingston MEMORY

Exercise 3: Paging

Assume that you are currently executing a process P with the following page table on a system with 16 byte pages:

⋮	Frame	Access
0x17	0x47	R,W
0x16	0xF4	R,W
0x15	NULL	R,W
0x14	0x23	R,X
⋮		

- What is the physical address that corresponds to the virtual address 0x147?
- What is the physical address that corresponds to the virtual address 0x16E?

Exercise 3: Paging

Assume that you are currently executing a process P with the following page table on a system with 16 byte pages:

	Frame	Access
⋮		
0x17	0x47	R,W
0x16	0xF4	R,W
0x15	NULL	R,W
0x14	0x23	R,X
⋮		

Memory as a Cache

- each page table entry has a valid bit
- for valid entries, frame indicates physical address of page in memory
- a **page fault** occurs when a program requests a page that is not currently in memory
  - handled much like a cache miss
  - evict another page in memory to make space (which one?)
  - takes time to handle, so context switch

MMU

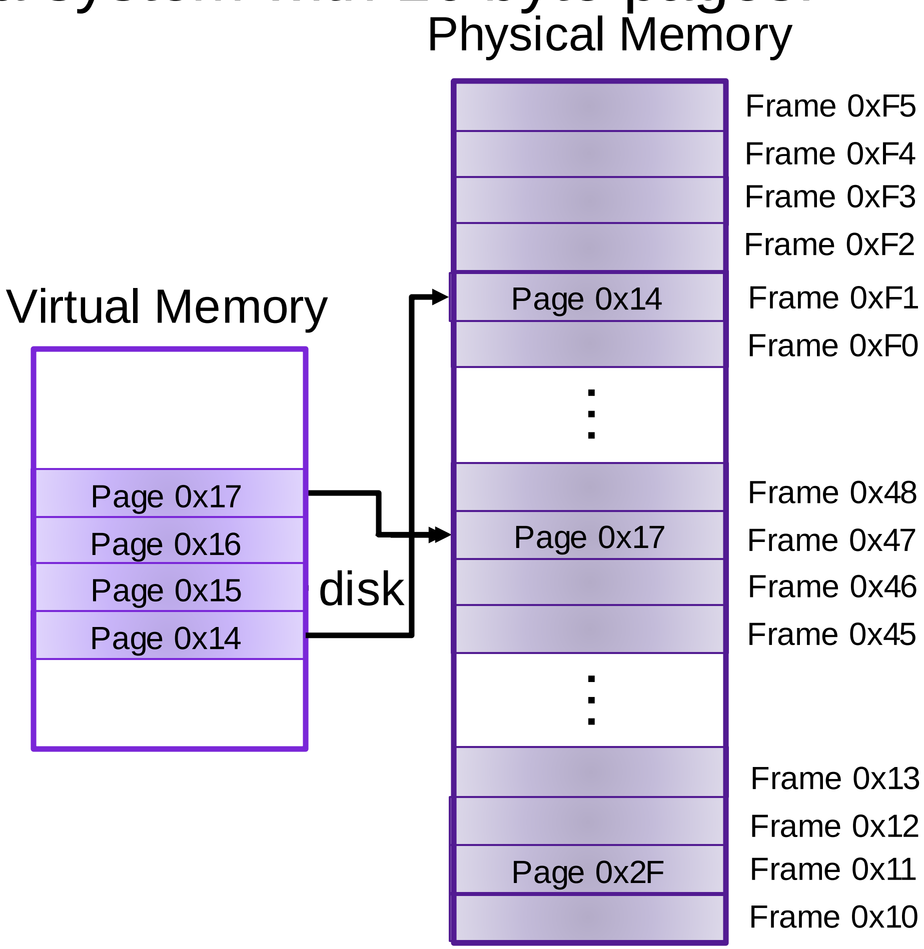
v	Frame	Access
1	0x47	R,W
0	NULL	R,W
0	0x13	R,W
1	0xF1	R,X

⋮

Memory as a Cache

Assume that you are currently executing a process P with the following page table on a system with 16 byte pages:

	v	Frame	Access
⋮			
0x17	1	0x47	R,W
0x16	0	NULL	R,W
0x15	0	0x13	R,W
0x14	1	0xF1	R,X
⋮			



Thrashing

- working set is the collection of a pages a process requires in a given time interval
- if it doesn't fit in memory, program will thrash

Exercise 4: Paging

Assume that you are currently executing a process P with the following page table on a system with 256 byte pages:

	v	Frame	Access
⋮			
0xFA	1	0x47	R,W
0xF9	1	0x24	R,W
0xF8	0	NULL	R,W
0xF7	0	0x23	R,X
⋮			

- What is the physical address that corresponds to the virtual address 0xF947?
- What is the physical address that corresponds to the virtual address 0xF700?
- What is the physical address that corresponds to the virtual address 0xF813?

Evaluating Paging

- **Isolation:** don't want different process states collided in physical memory ✓
- **Efficiency:** want fast reads/writes to memory ✗
- **Sharing:** want option to overlap for communication ✓
- **Utilization:** want best use of limited resource ✓
- **Virtualization:** want to create illusion of more resources ✓