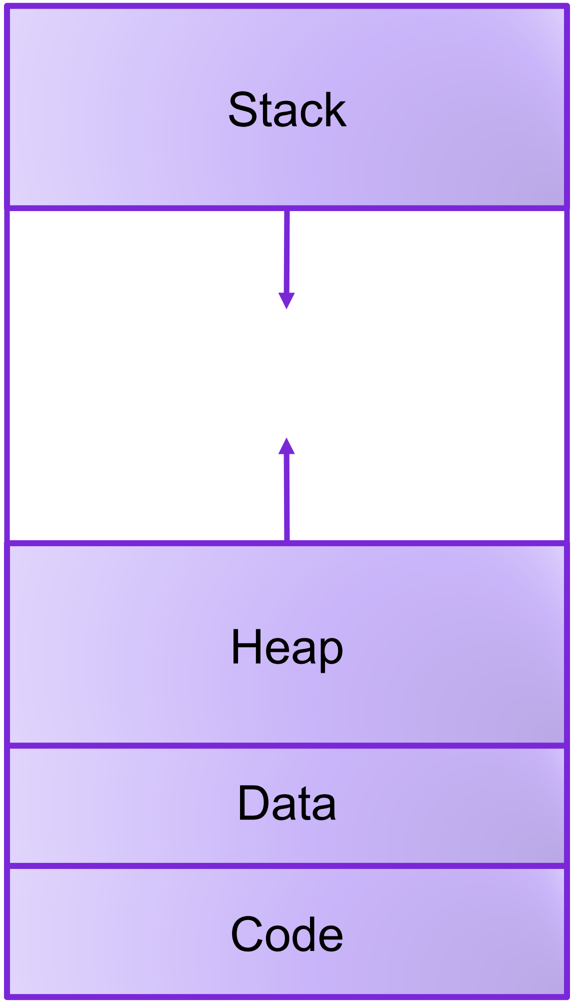
Lecture 19: Virtual Memory

CS 105

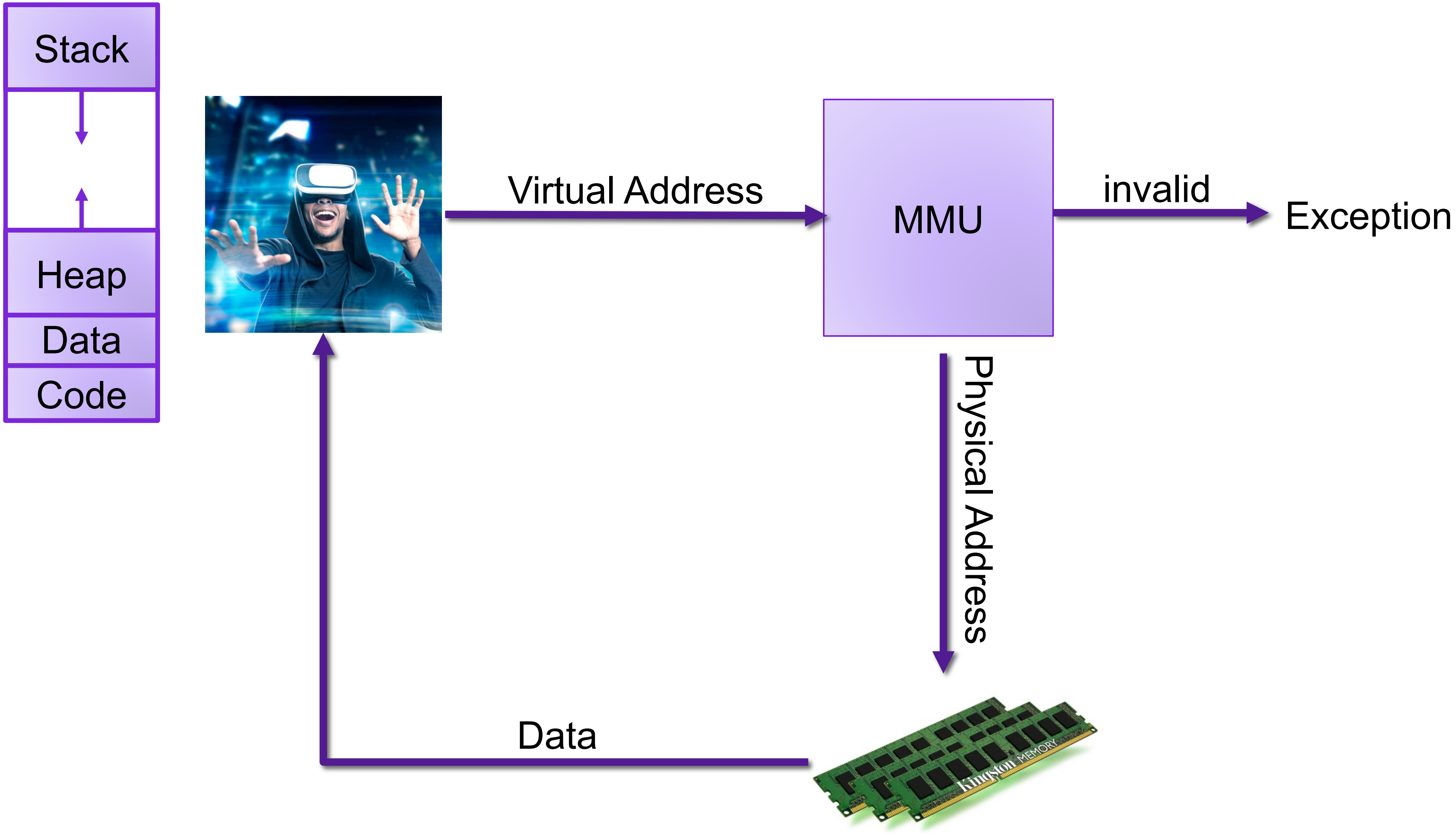
Virtual Memory Goals



- **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

Process View vs. OS View

Address Translation



Stack

Heap

Data

Code

Virtual Address

MMU

invalid

Exception

Physical Address

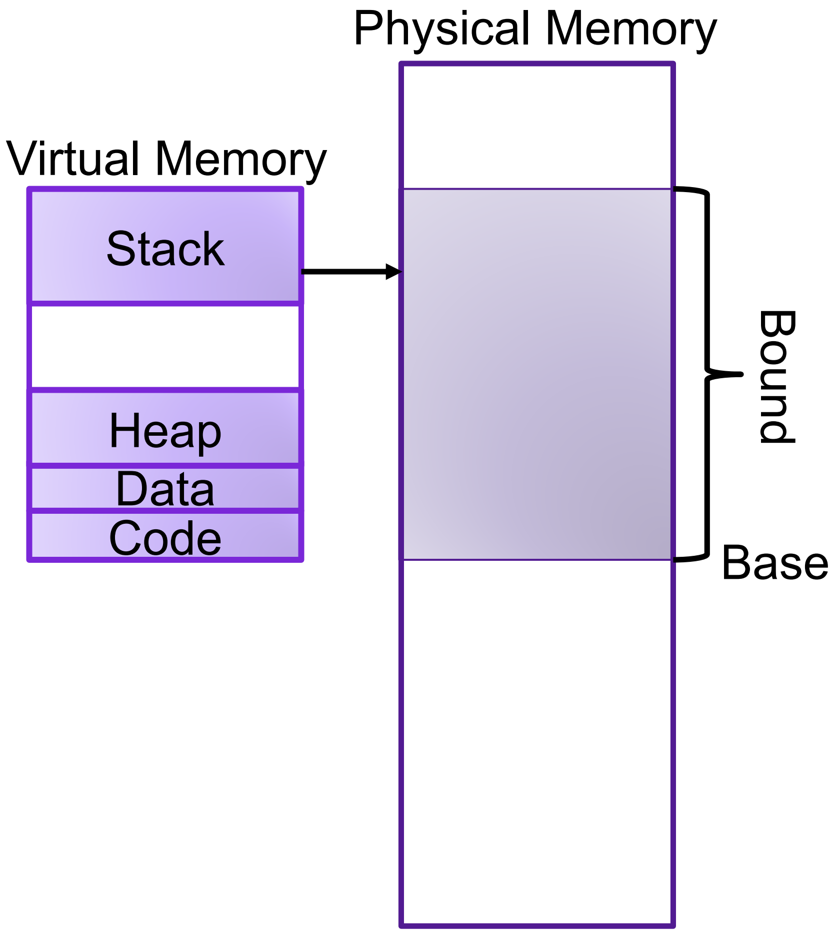
Data

Possibilities

Let’s look at some good and bad possibilities

- Base-and-bound
- Segmentation
- Paging

Base-and-Bound



Physical Memory

Virtual Memory

Stack

Heap

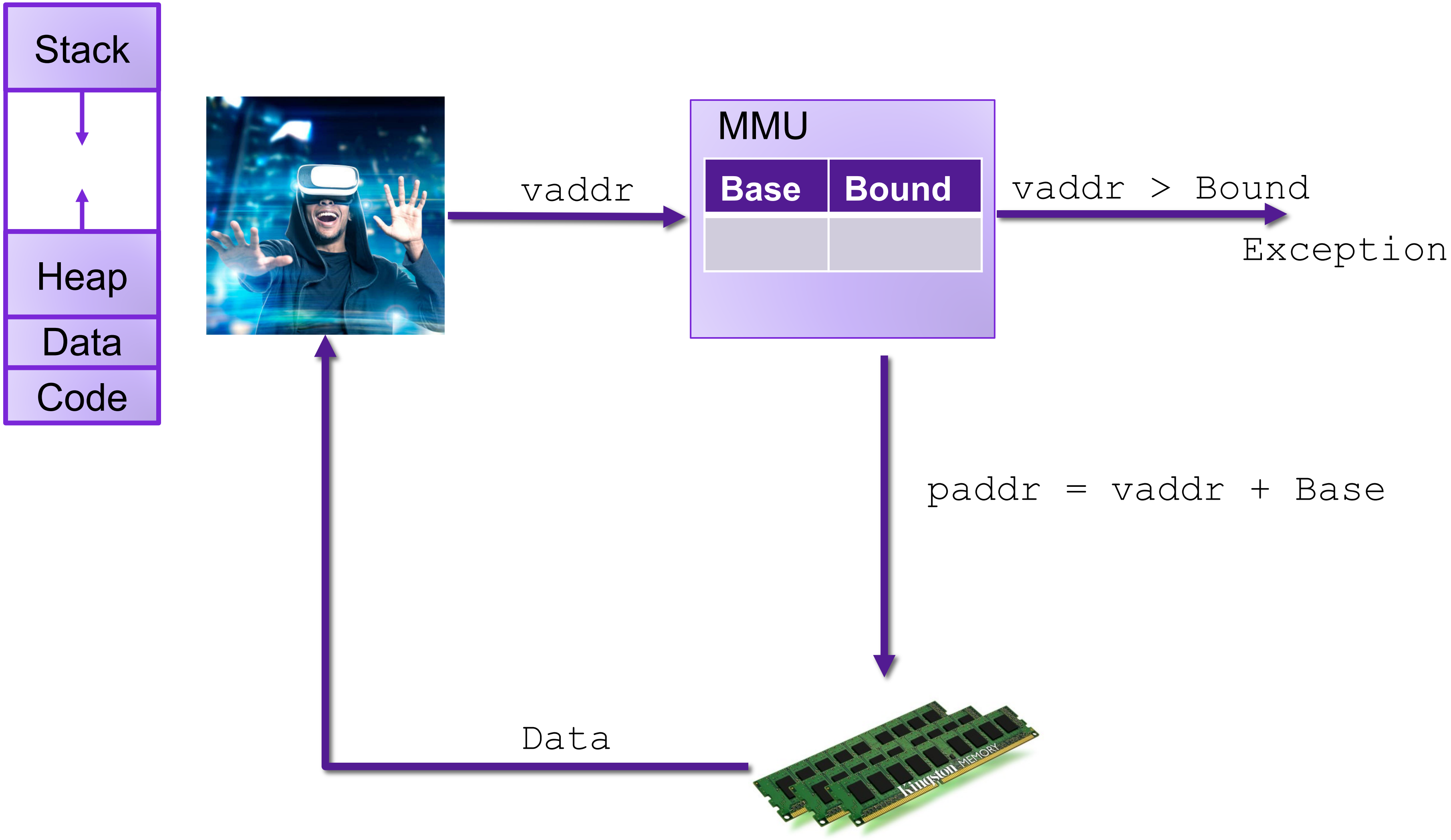
Data

Code

Bound

Base

Base-and-Bound



Stack

Heap

Data

Code

vaddr

MMU

Base	Bound

vaddr > Bound

Exception

paddr = vaddr + Base

Data

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?

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?

0x127b

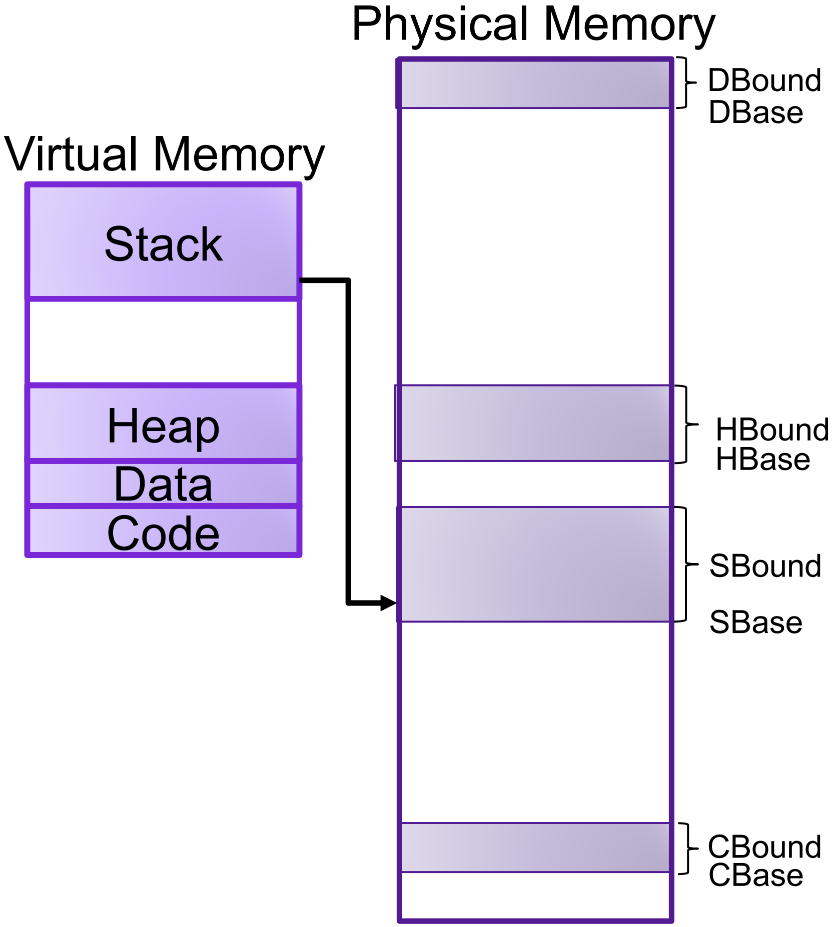
- What is the physical address that corresponds to the virtual address 0x123?

invalid

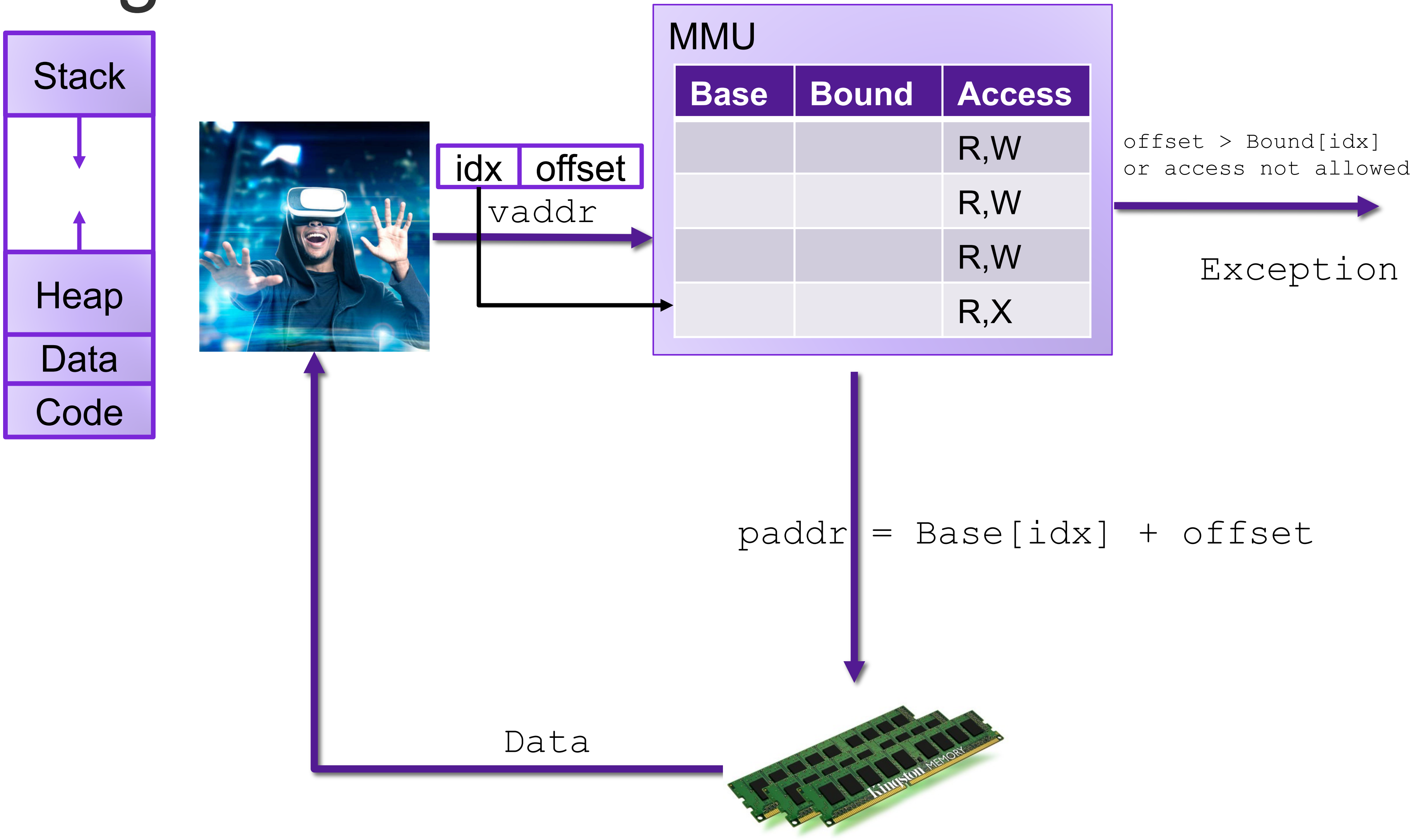
Evaluating Base-and-Bound

- **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



Stack

Heap

Data

Code

idx	offset

vaddr

MMU

Base	Bound	Access
		R,W
		R,W
		R,W
		R,X

offset > Bound[idx]
or access not allowed

Exception

paddr = Base[idx] + offset

Data

Exercise 2: Segmentation

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

Base	Bound	Access
`0x4747`	`0x080`	R,W
`0x2424`	`0x040`	R,W
`0x0023`	`0x080`	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`?

Exercise 2: Segmentation

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

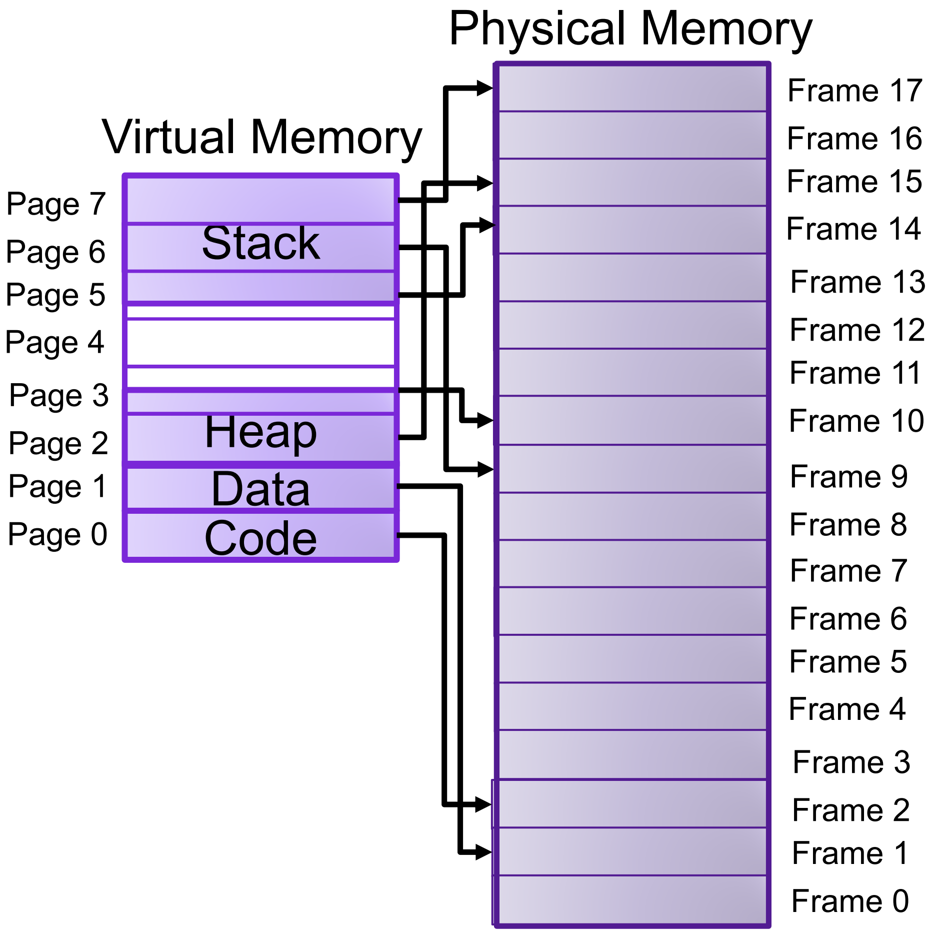
Base	Bound	Access
0x4747	0x080	R,W
0x2424	0x040	R,W
0x0023	0x080	R,W
0x1000	0x200	R,X

- What is the physical address that corresponds to the virtual address 0x001? | 00 | 0000000001 | **0x4748**
- What is the physical address that corresponds to the virtual address 0xD47? | 11 | 0101000111 | **0x1147**

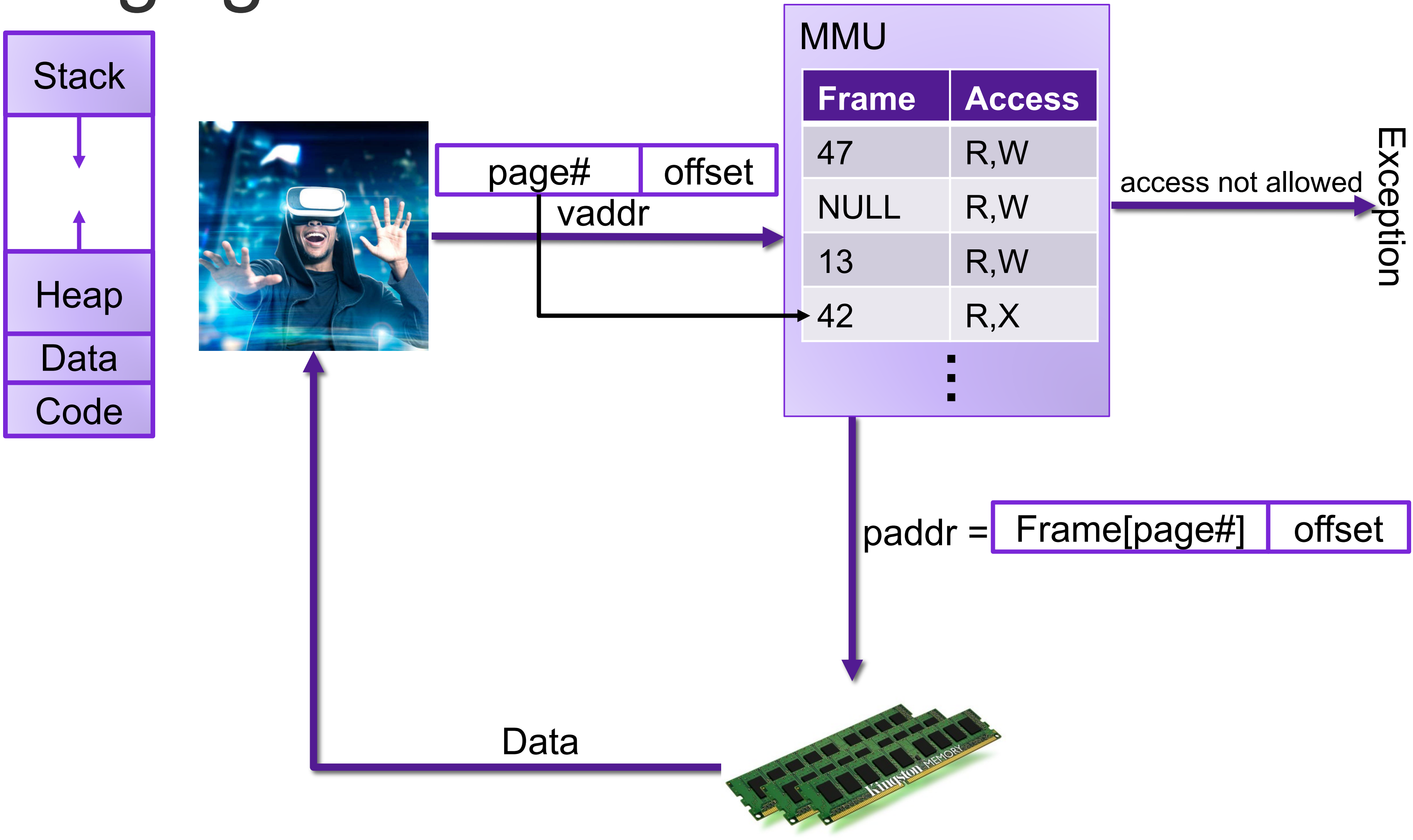
Evaluating Segmentation

- **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


Physical Memory
Virtual Memory
Page 7
Page 6
Page 5
Page 4
Page 3
Page 2
Page 1
Page 0
Stack
Heap
Data
Code
Frame 17
Frame 16
Frame 15
Frame 14
Frame 13
Frame 12
Frame 11
Frame 10
Frame 9
Frame 8
Frame 7
Frame 6
Frame 5
Frame 4
Frame 3
Frame 2
Frame 1
Frame 0

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

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
⋮		

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

00010100	0111

  0x237

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

00010110	1110

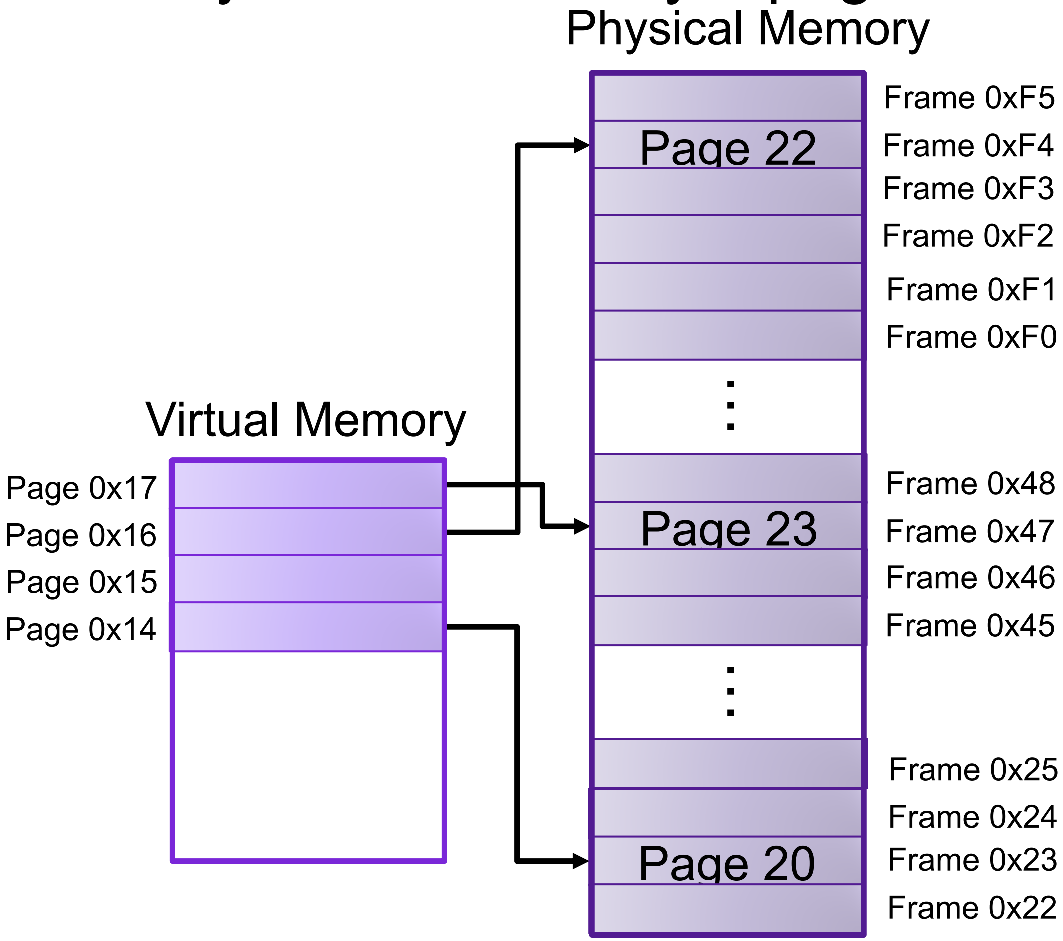
  0xF4E

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
⋮		

0x147 → **0x237**



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
  - takes time to handle, so context switch
  - evict another page in memory to make space (which one?)

MMU

v	Frame	Access
1	47	R,W
0	NULL	R,W
0	13	R,W
1	42	R,X

⋮

Thrashing

- Working set is the collection of a pages a process requires in a given time interval

- If the working set doesn't fit in memory, then the program will thrash

- There are no OS solutions to this problem

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`?

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?

0xF9	0x47	**0x2447**

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

0xF7	0x00	**0x2300**

Page fault

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 ✓