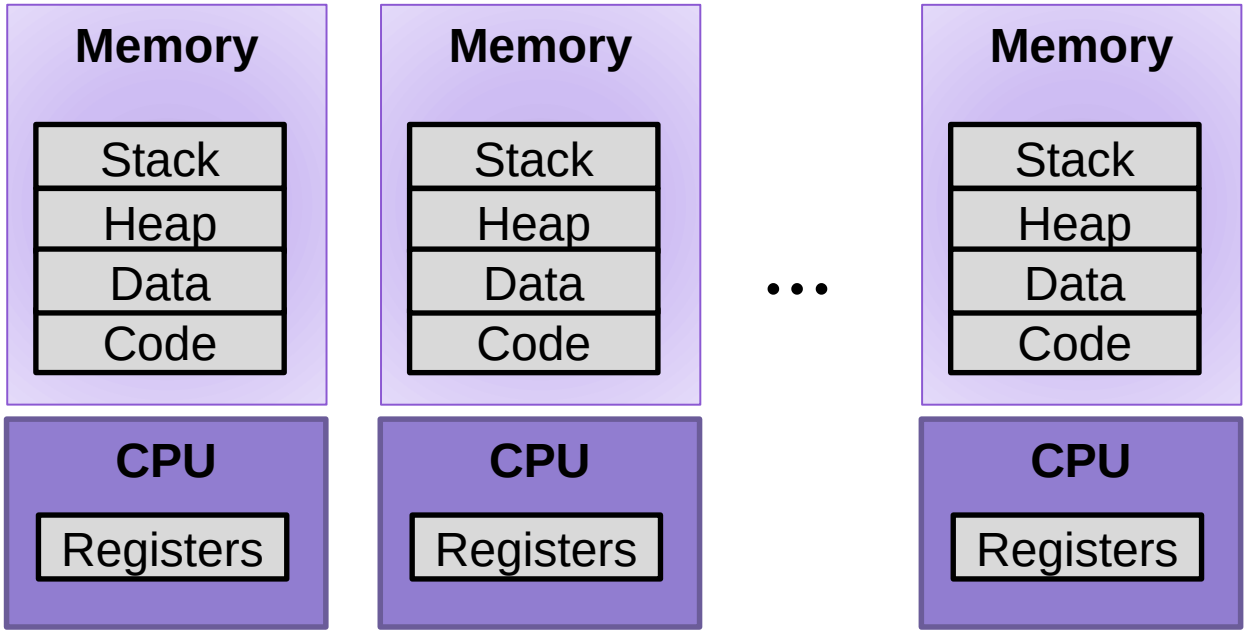
Lecture 16: Virtual Memory

CS 105

Spring 2025

Multiprocessing: The Illusion



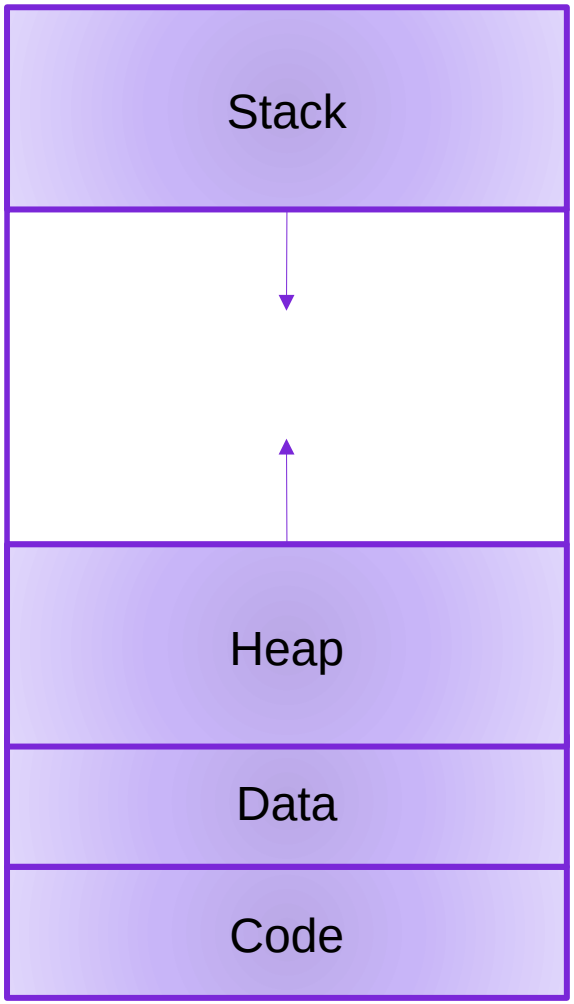
- Process provides each program with two key abstractions:
  - ***Logical control flow***
    - Each program seems to have exclusive use of the CPU
    - Provided by kernel mechanism called ***context switching***
  - ***Private address space***
    - Each program seems to have exclusive use of main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

Multiprocessing: The Reality

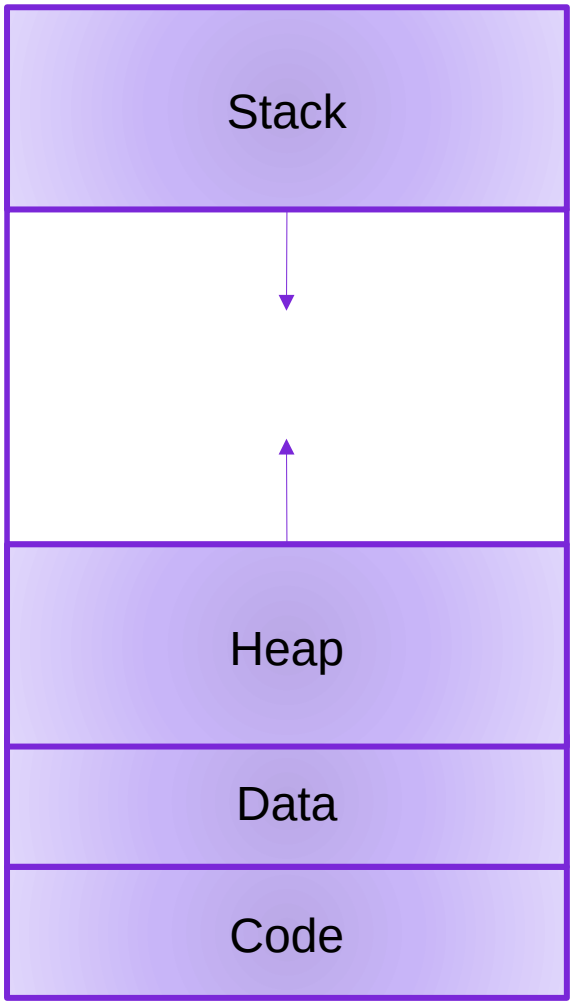
- Computer runs many processes simultaneously
- Running program “top” on Linux/Mac
  - E.g., system has 558 processes, 2 of which are active
  - Identified by Process ID (PID)

```
top - 11:35:22 up  3:38,  1 user,  load average: 1.11, 0.68, 1.10
Tasks: 560 total,   2 running, 558 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.4 us,  1.5 sy,  0.8 ni, 97.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Virtual Memory Goals

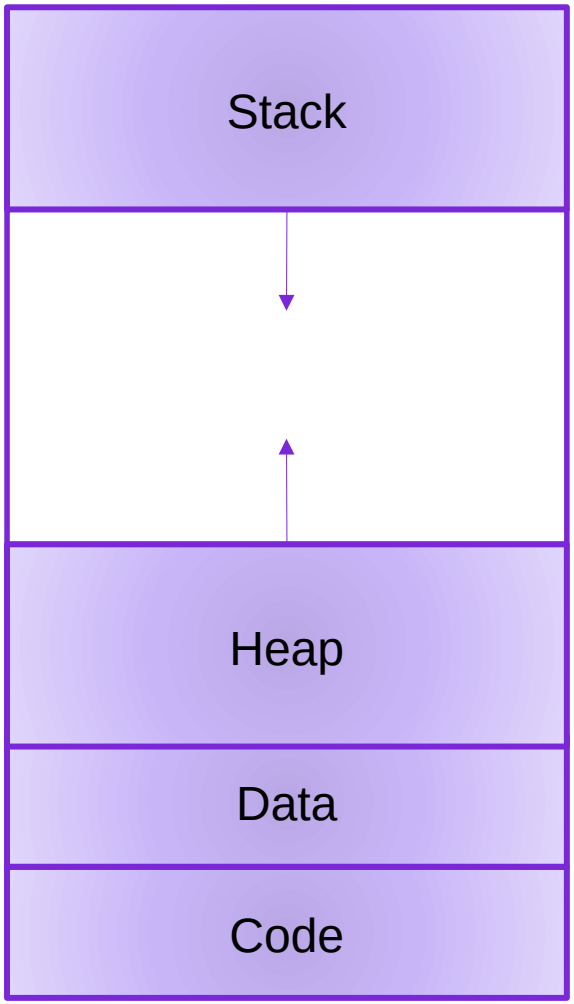


Virtual Memory Goals



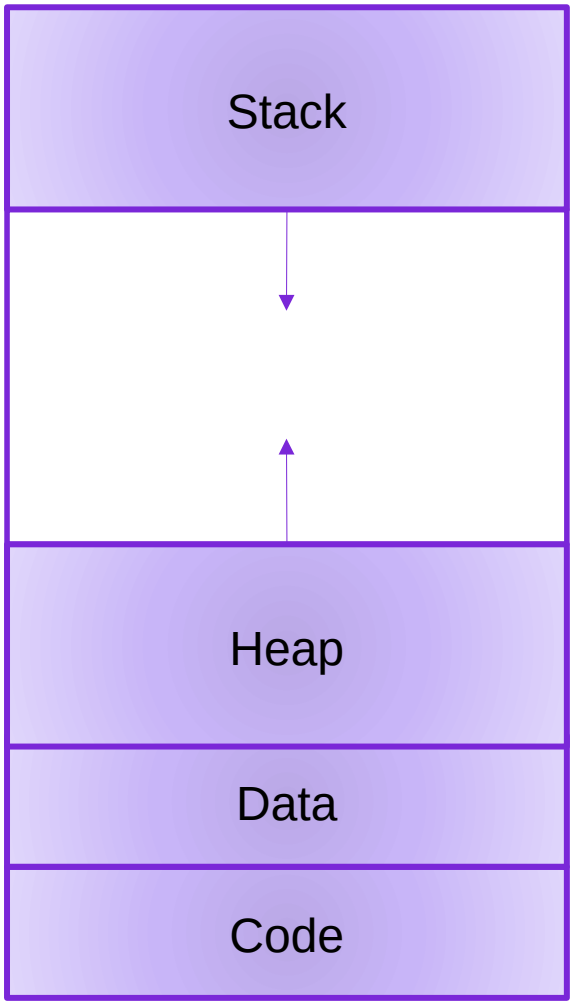
- **Isolation:** don’t want different process states collided in physical memory

Virtual Memory Goals



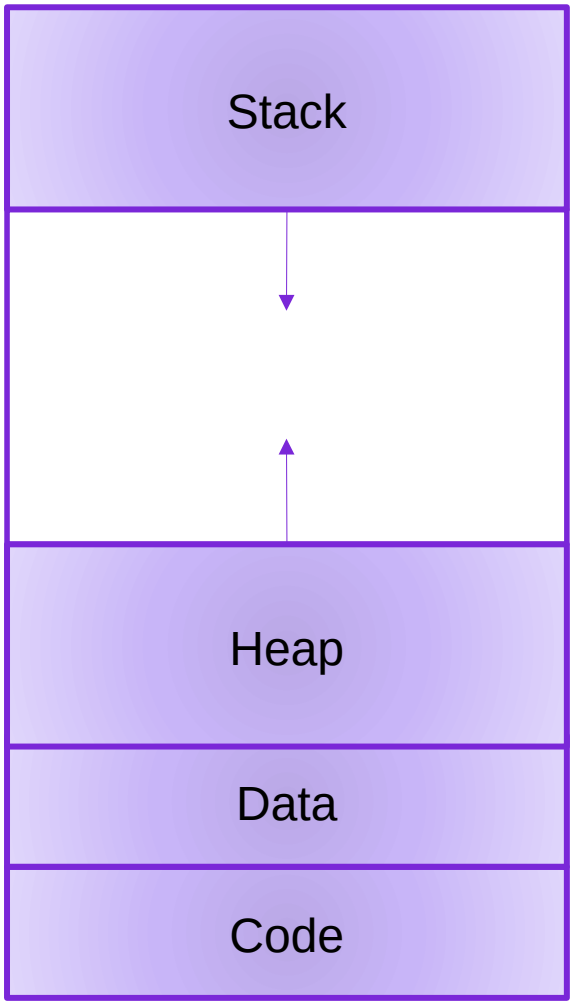
- **Isolation:** don't want different process states collided in physical memory
- **Efficiency:** want fast reads/writes to memory

Virtual Memory Goals



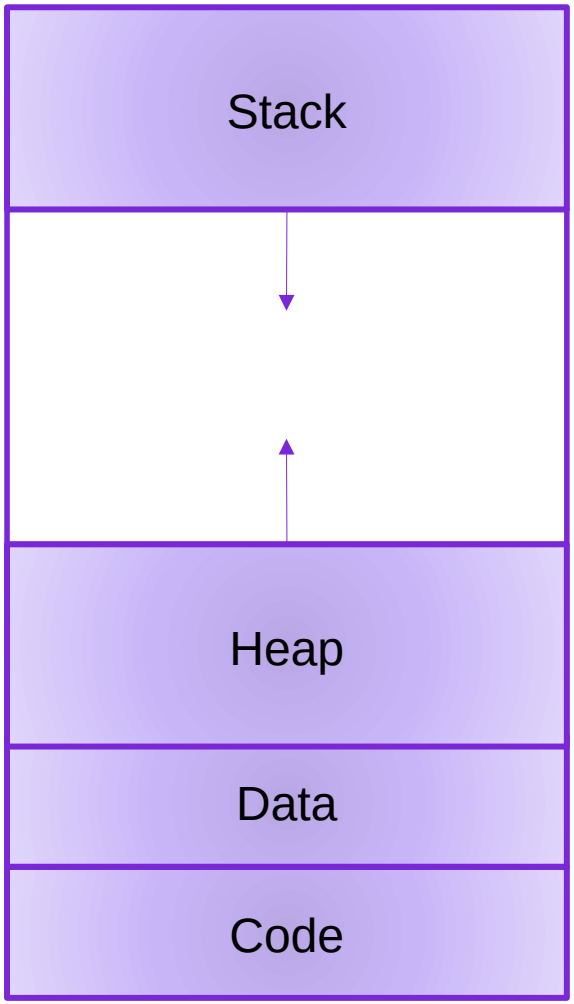
- **Isolation:** don't want different process states collided in physical memory
- **Efficiency:** want fast reads/writes to memory
- **Sharing:** want option to overlap for communication

Virtual Memory Goals



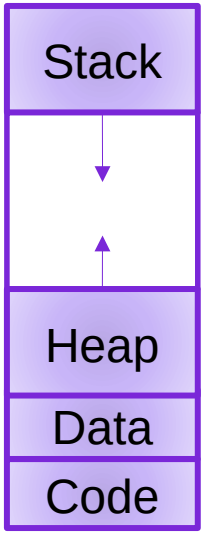
- **Isolation:** don't want different process states collided in physical memory
- **Efficiency:** want fast reads/writes to memory
- **Sharing:** want option to overlap for communication
- **Utilization:** want best use of limited resource

Virtual Memory Goals



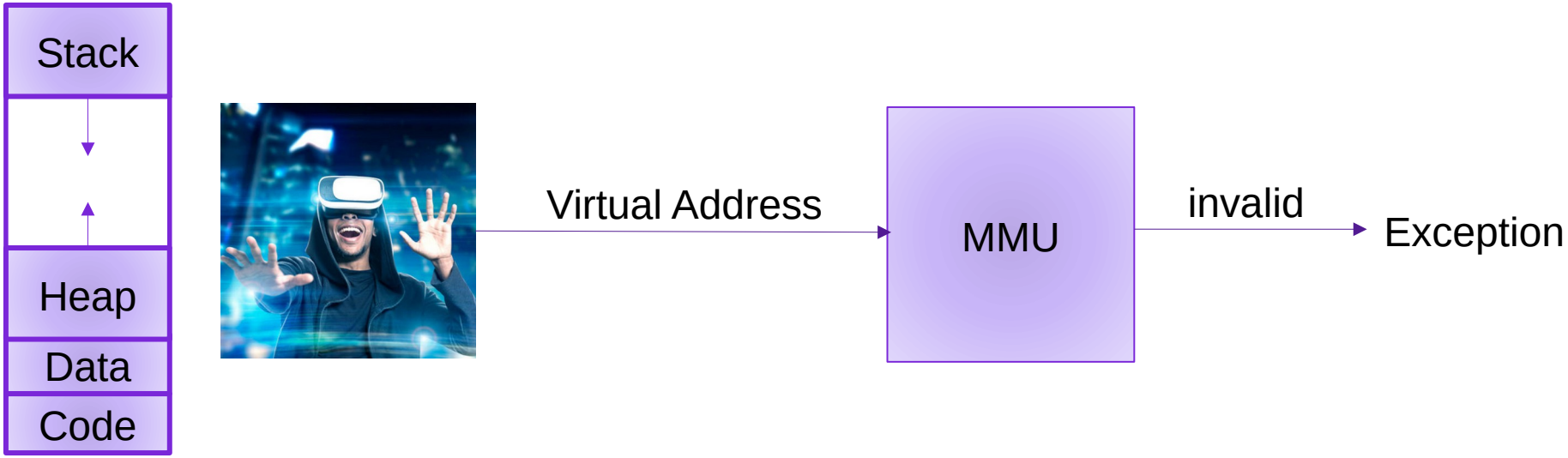
- **Isolation:** don't want different process states collided in physical memory
- **Efficiency:** want fast reads/writes to memory
- **Sharing:** want option to overlap for communication
- **Utilization:** want best use of limited resource
- **Virtualization:** want to create illusion of more resources

Address Translation

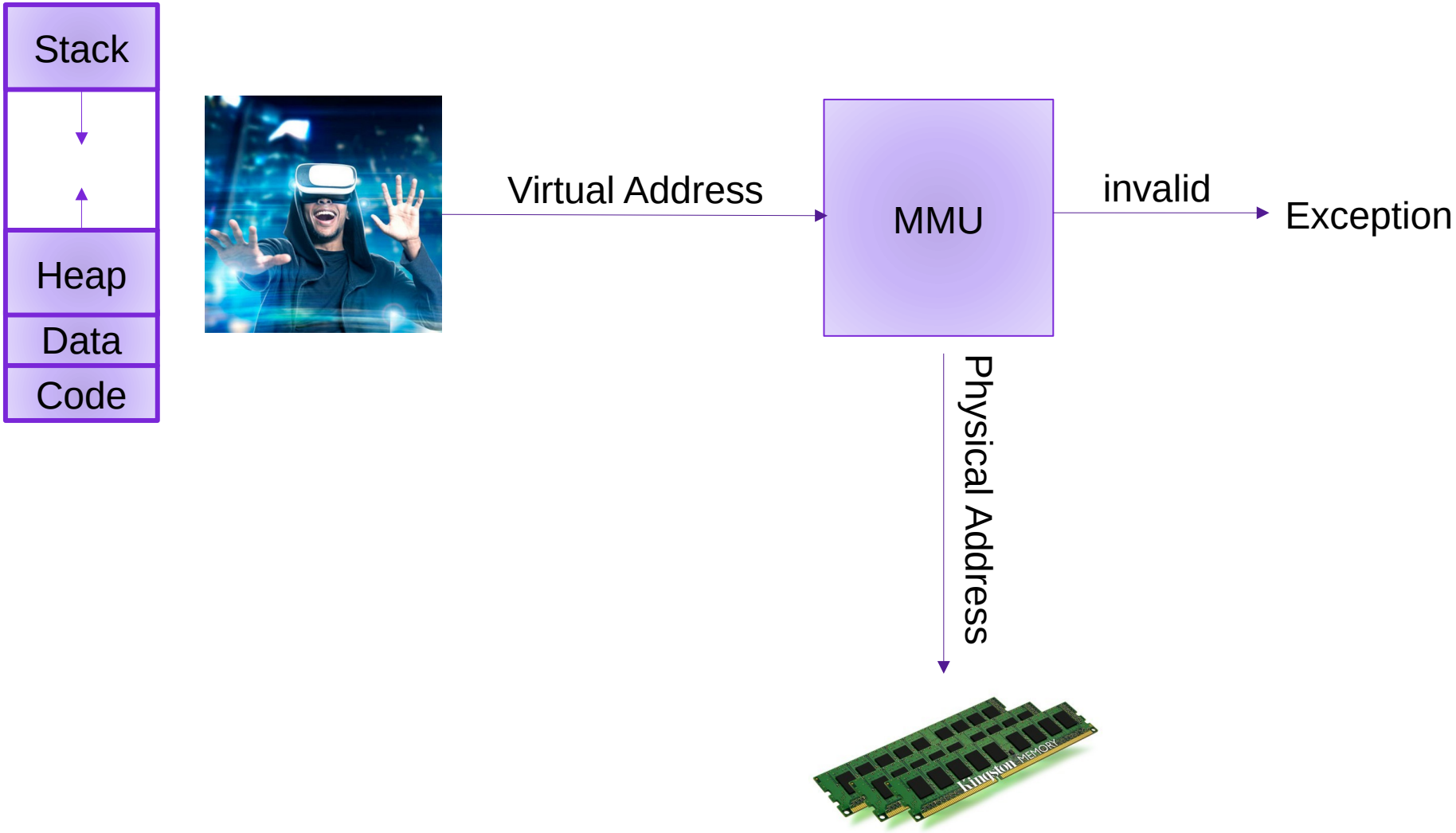


Address Translation

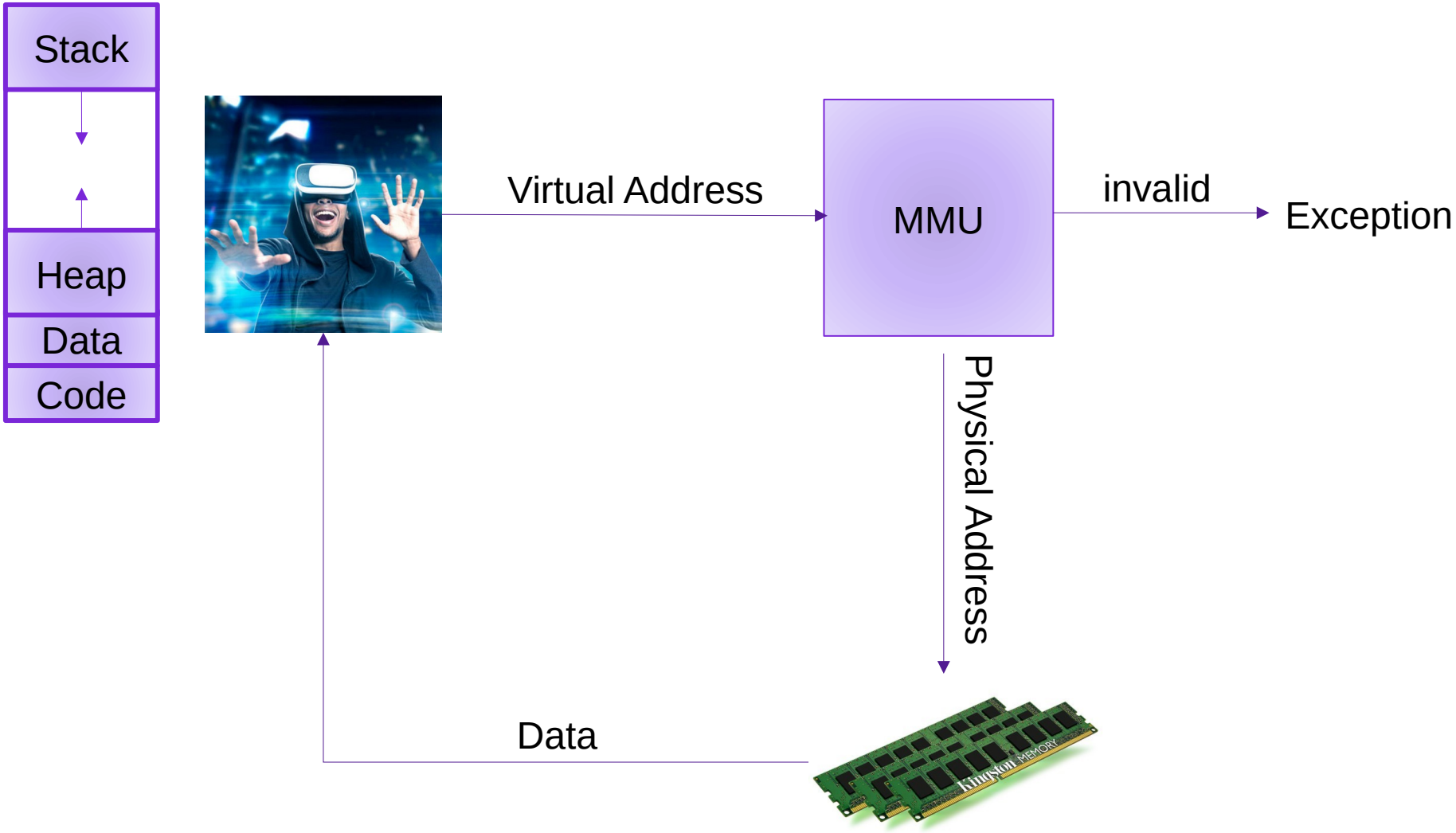
Address Translation



Address Translation

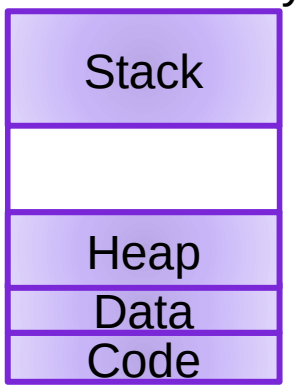


Address Translation

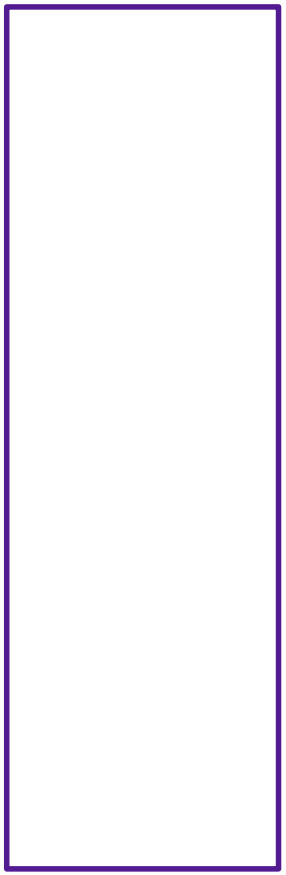


Base-and-Bound

Virtual Memory

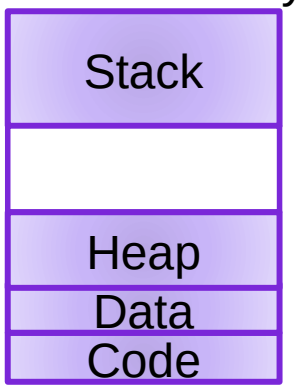


Physical Memory

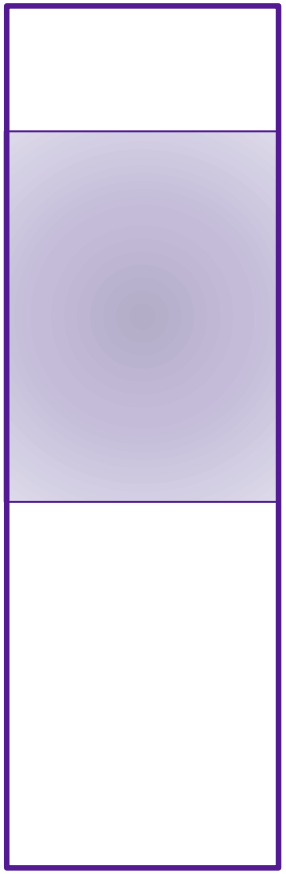


Base-and-Bound

Virtual Memory

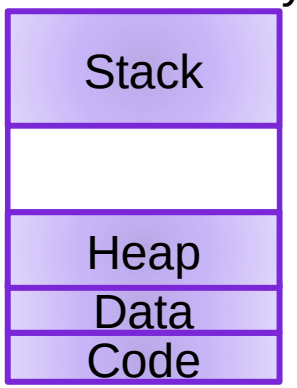


Physical Memory

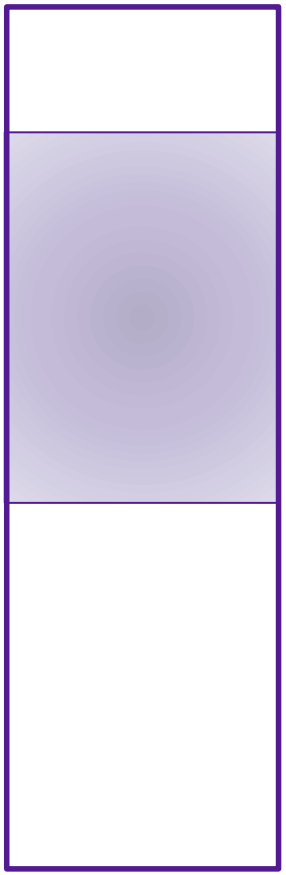


Base-and-Bound

Virtual Memory

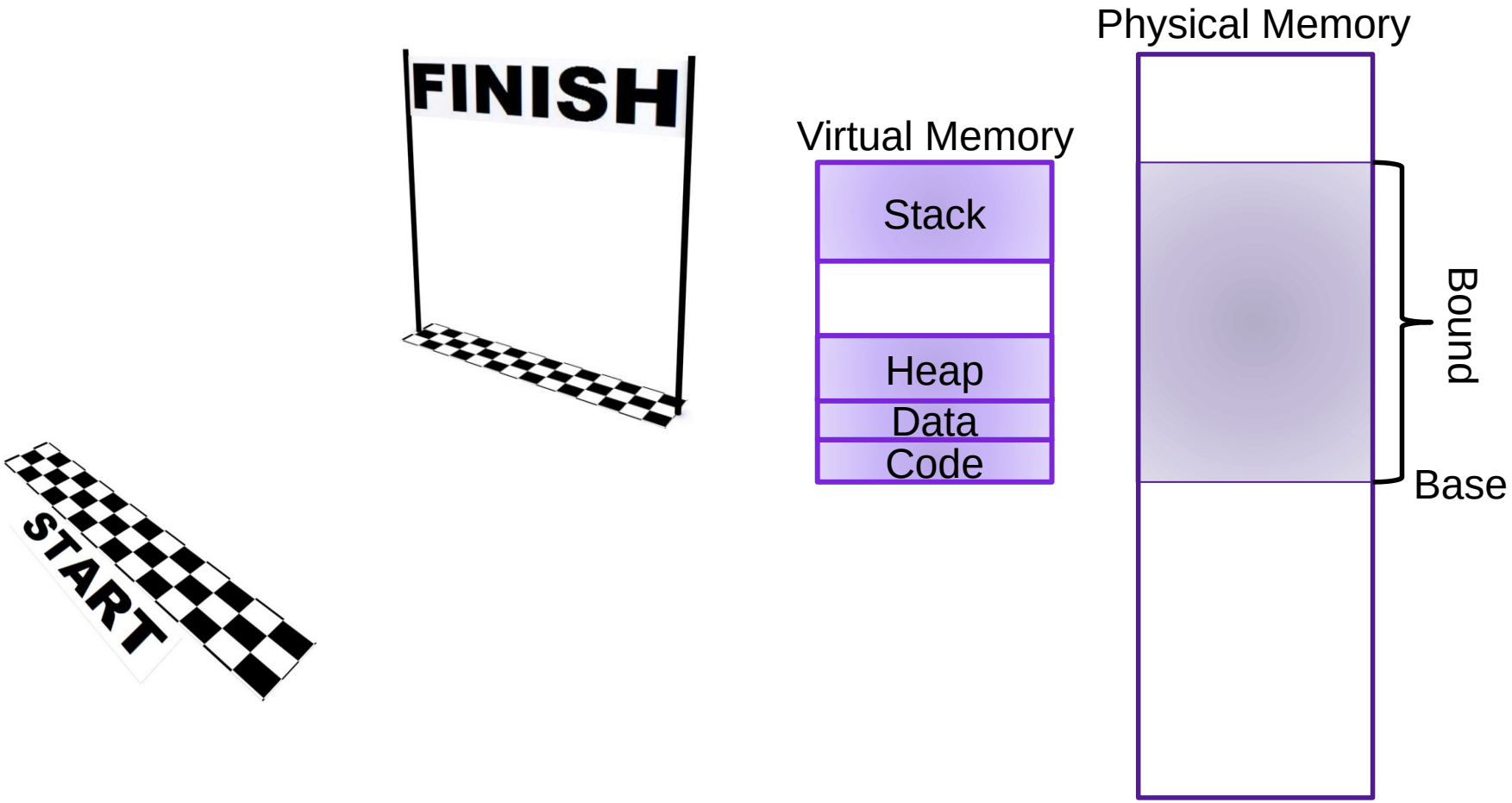


Physical Memory

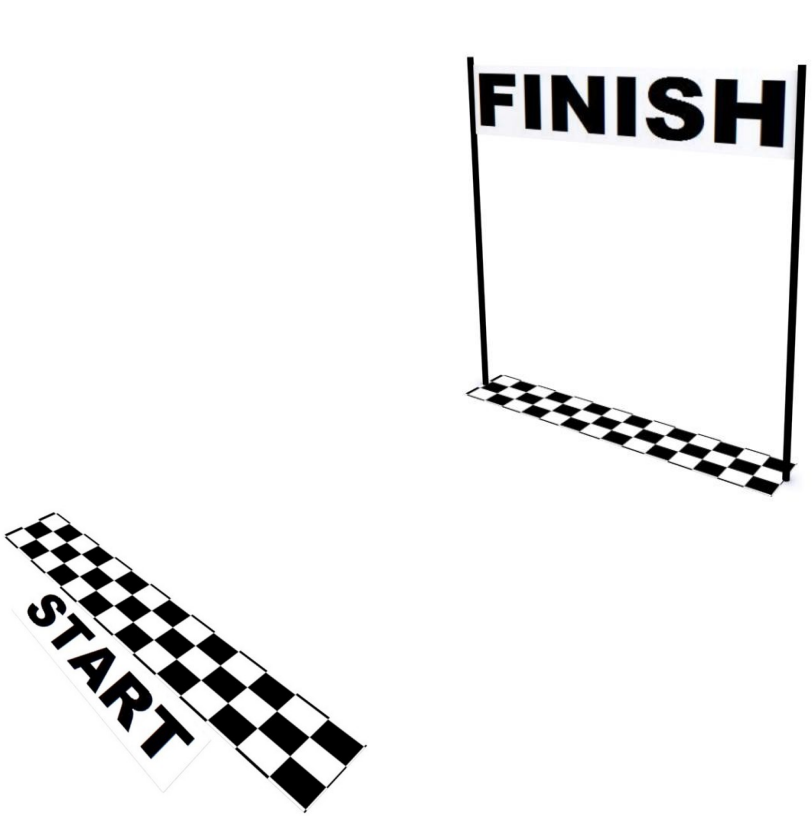


Base

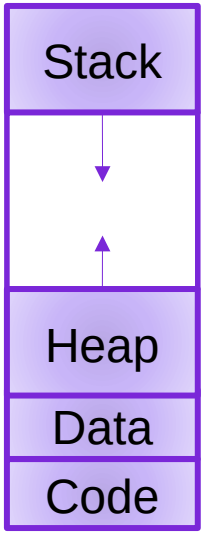
Base-and-Bound



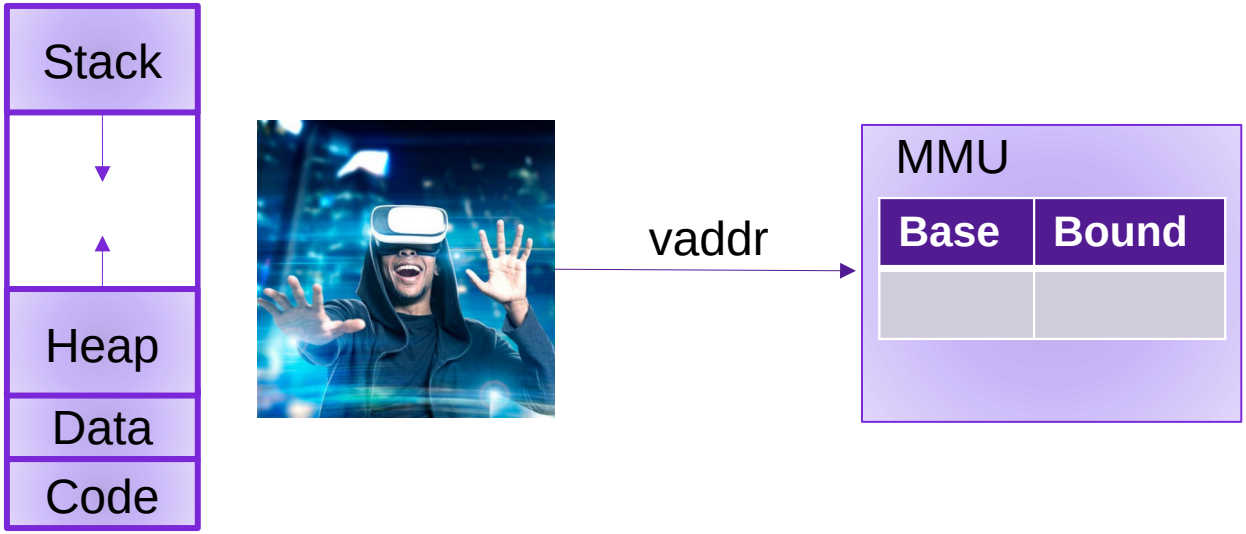
Base-and-Bound



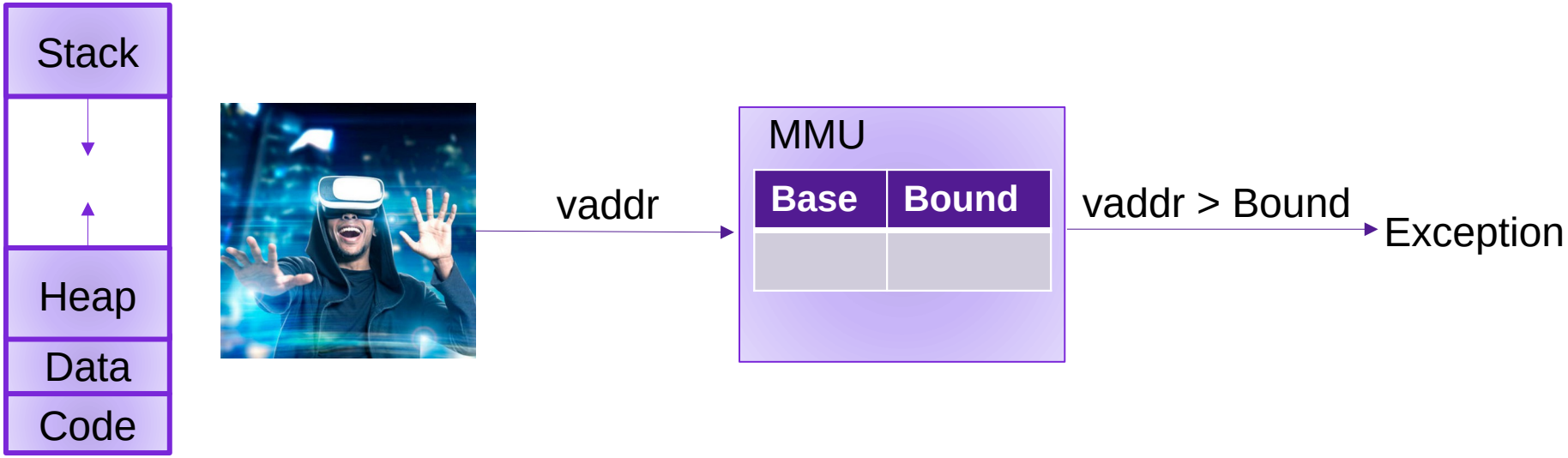
Base-and-Bound



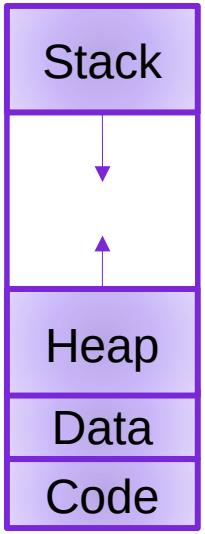
Base-and-Bound



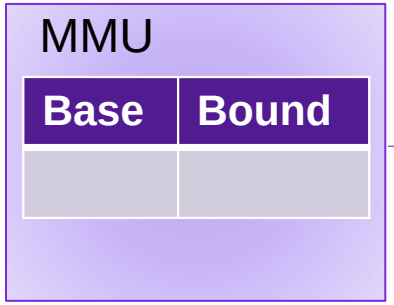
Base-and-Bound



Base-and-Bound



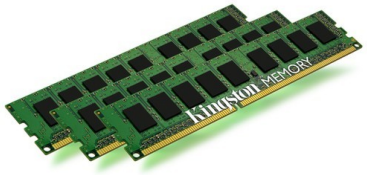
vaddr



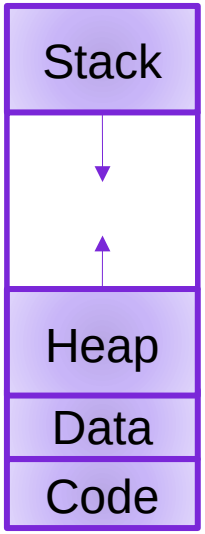
vaddr > Bound

Exception

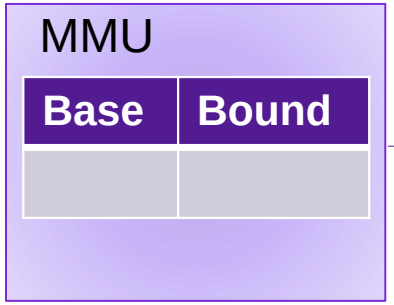
paddr = vaddr + Base



Base-and-Bound



vaddr

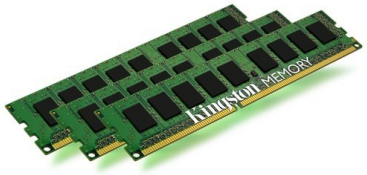


vaddr > Bound

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?

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

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

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

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

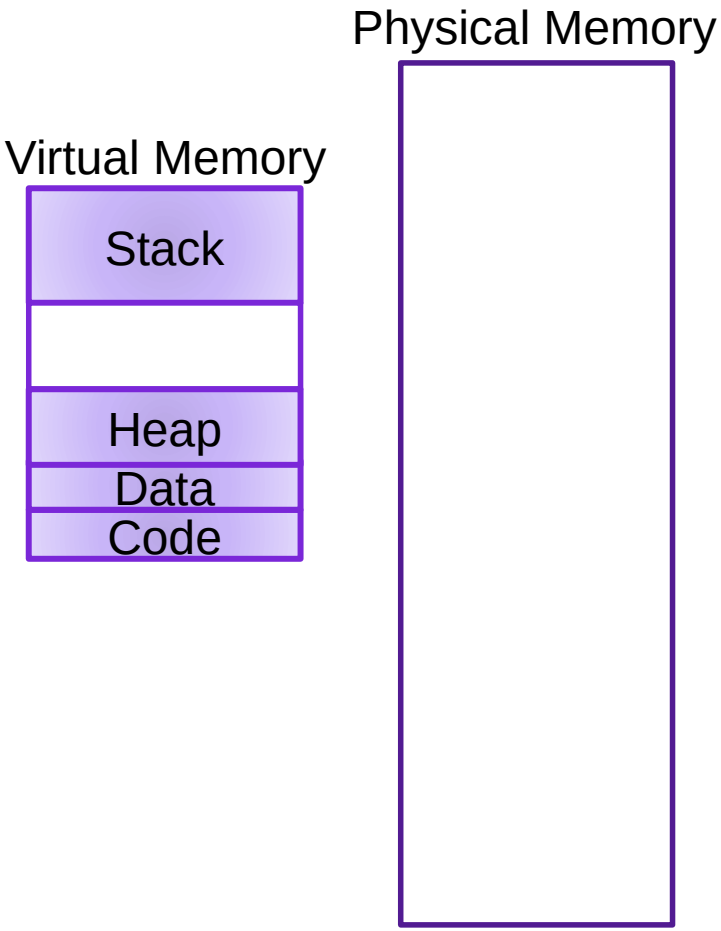
Evaluating Base-and-Bound

- **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

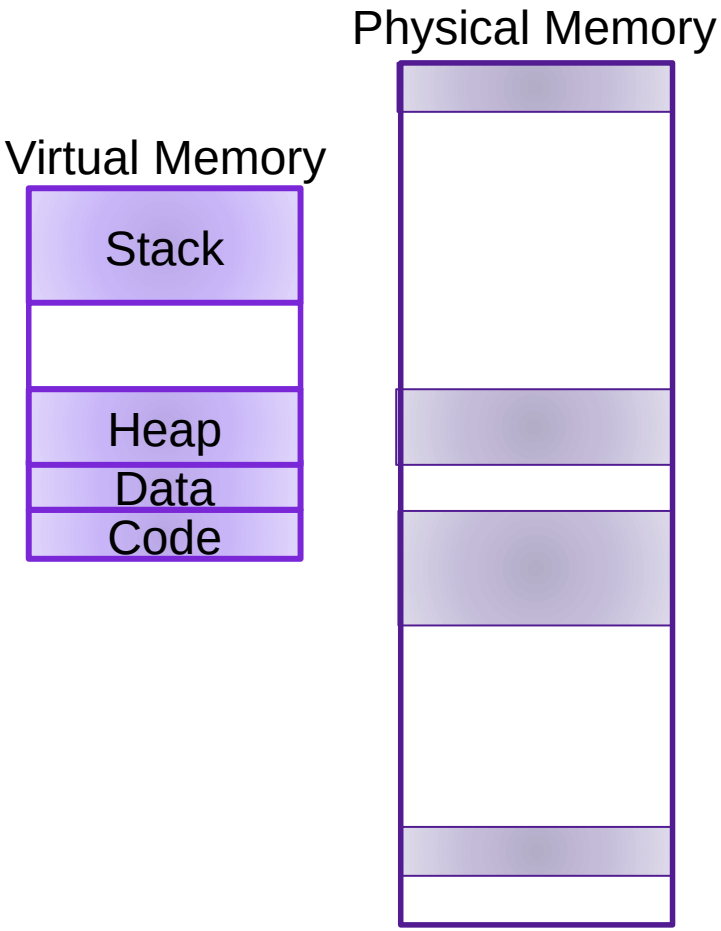
Evaluating Base-and-Bound

- **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

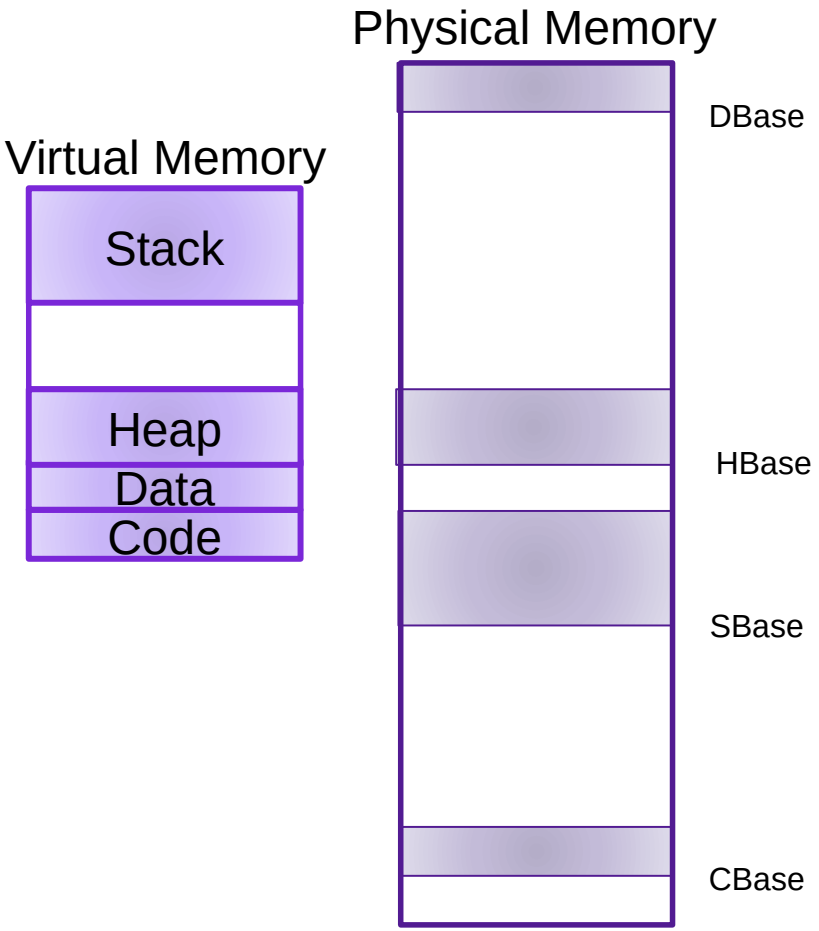
Segmentation



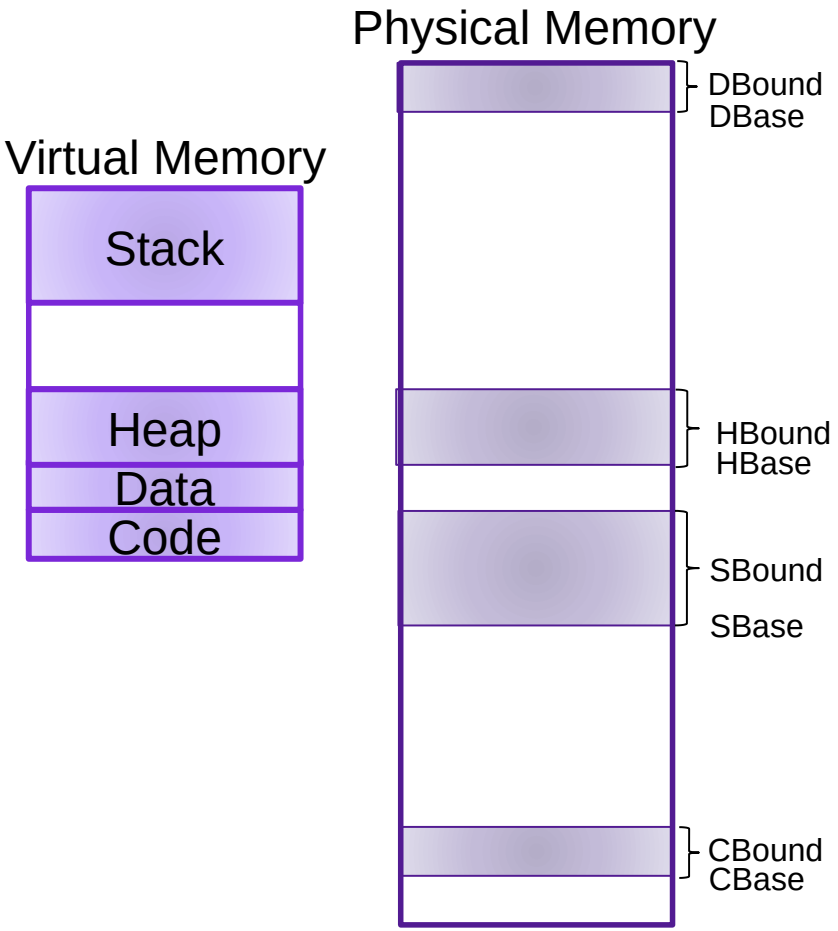
Segmentation



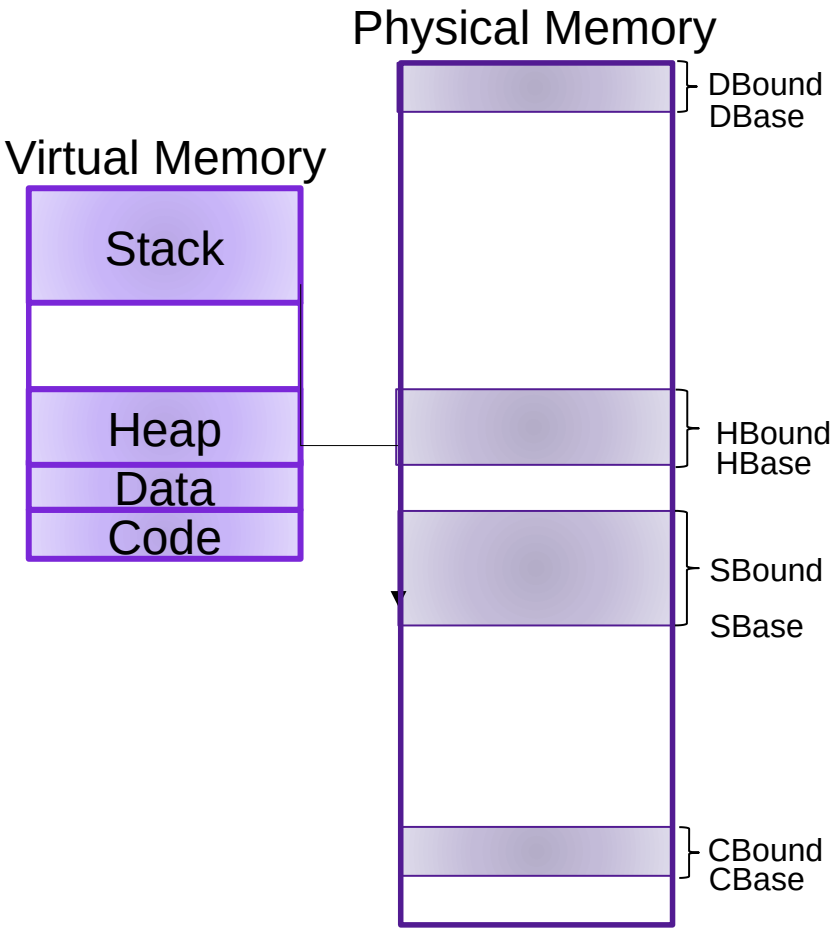
Segmentation



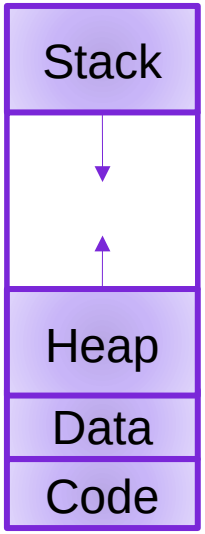
Segmentation



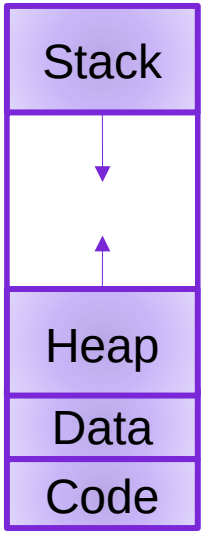
Segmentation



Segmentation



Segmentation

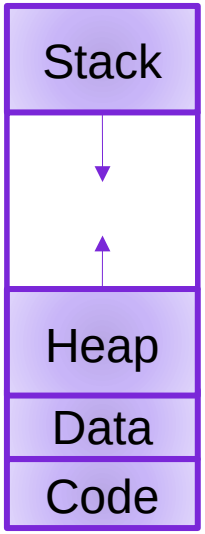


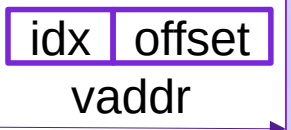
vaddr

MMU

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

Segmentation

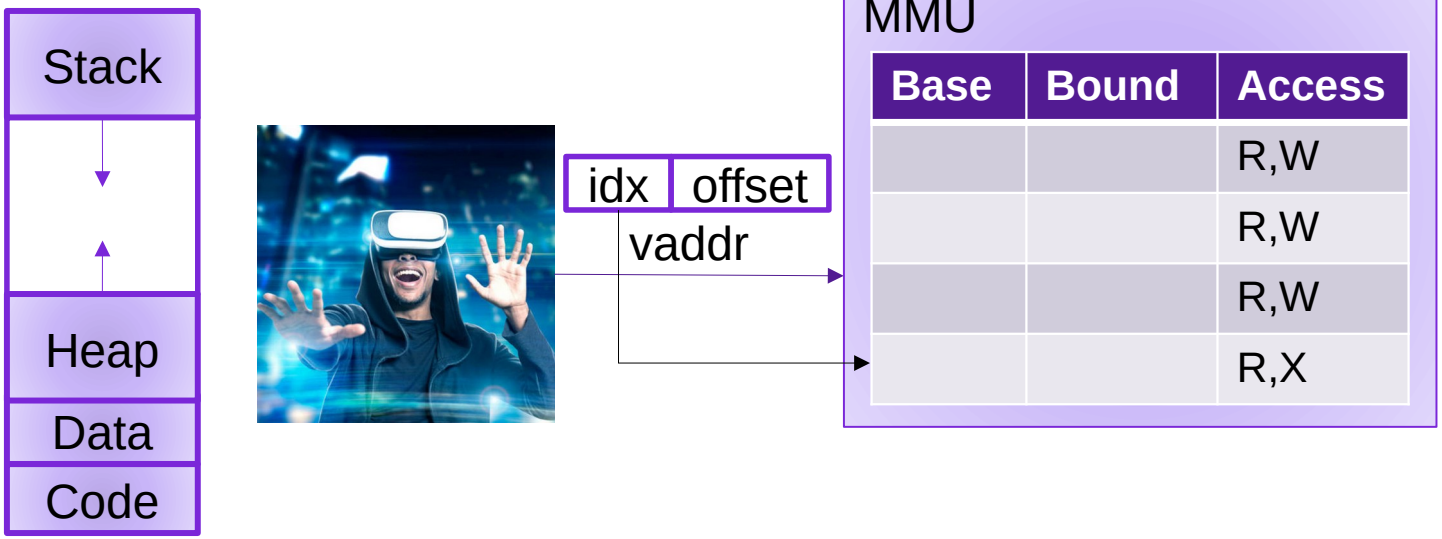




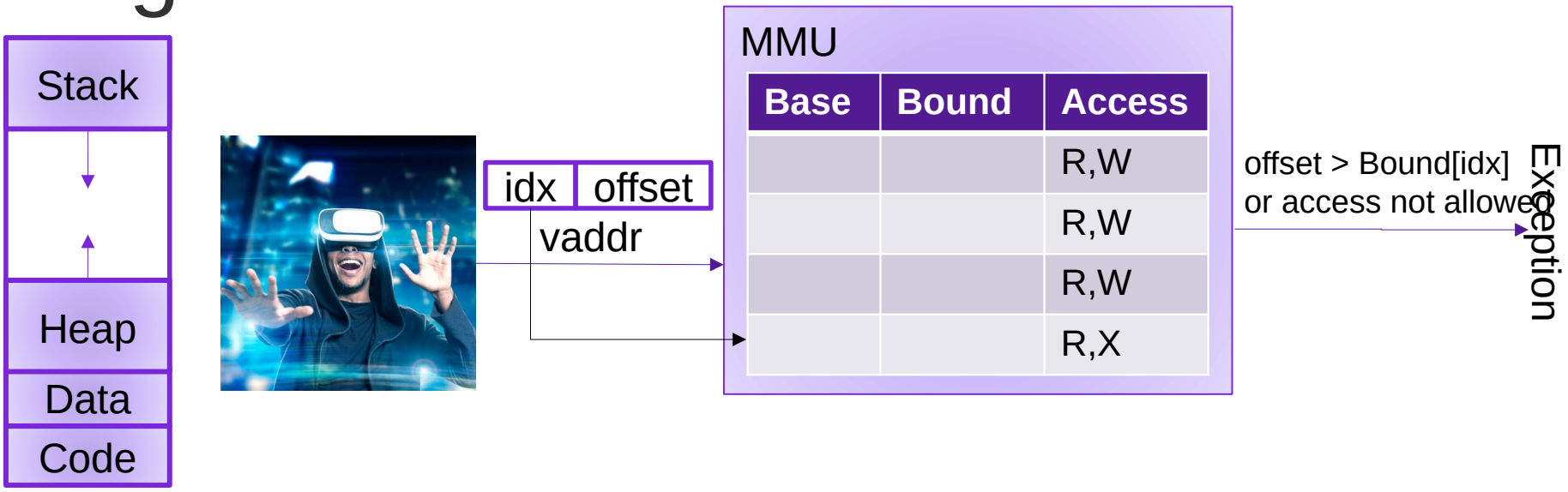
MMU

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

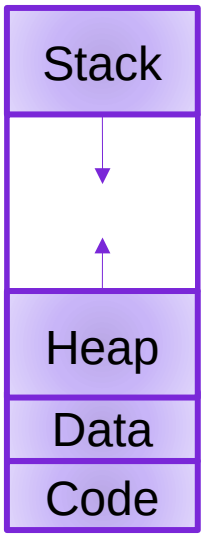
Segmentation



Segmentation



Segmentation



idx | offset

vaddr

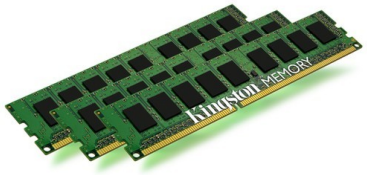
MMU

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

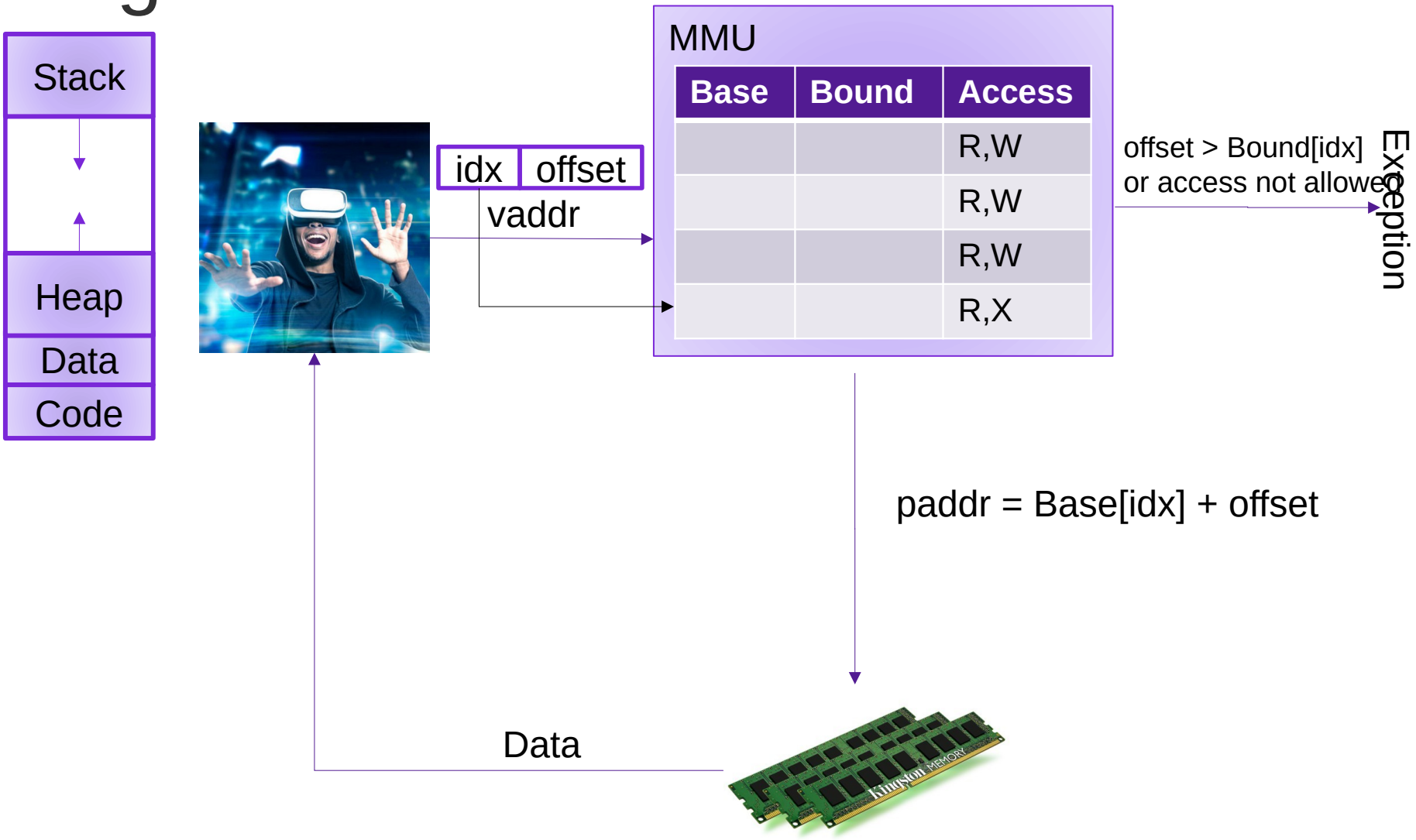
offset > Bound[idx]
or access not allowed

Exception

paddr = Base[idx] + offset



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?

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?

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?

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?

0x4748

- 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	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?

00	0000000001

0x4748

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

11	0101000111

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?

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

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

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

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

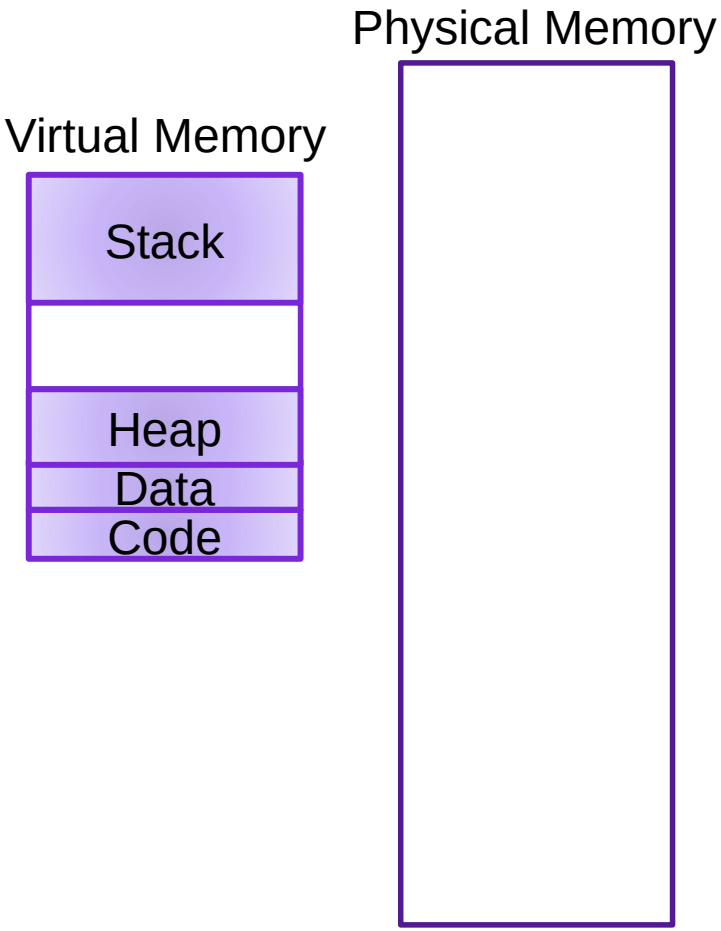
Evaluating Segmentation

- **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

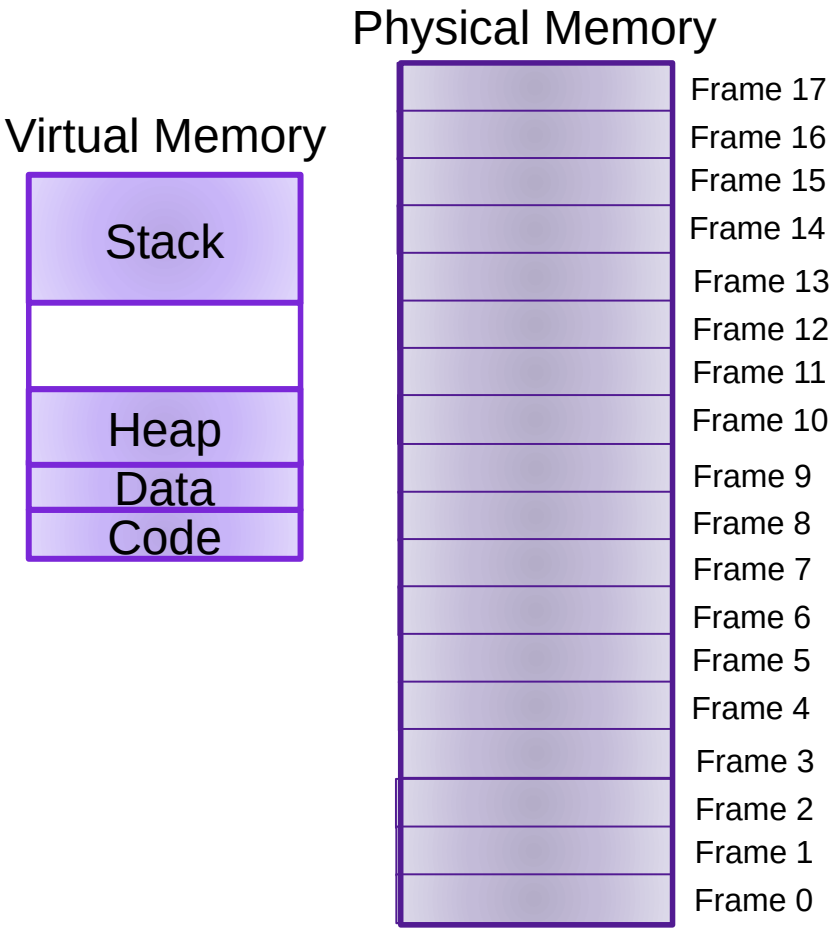
Evaluating Segmentation

- **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

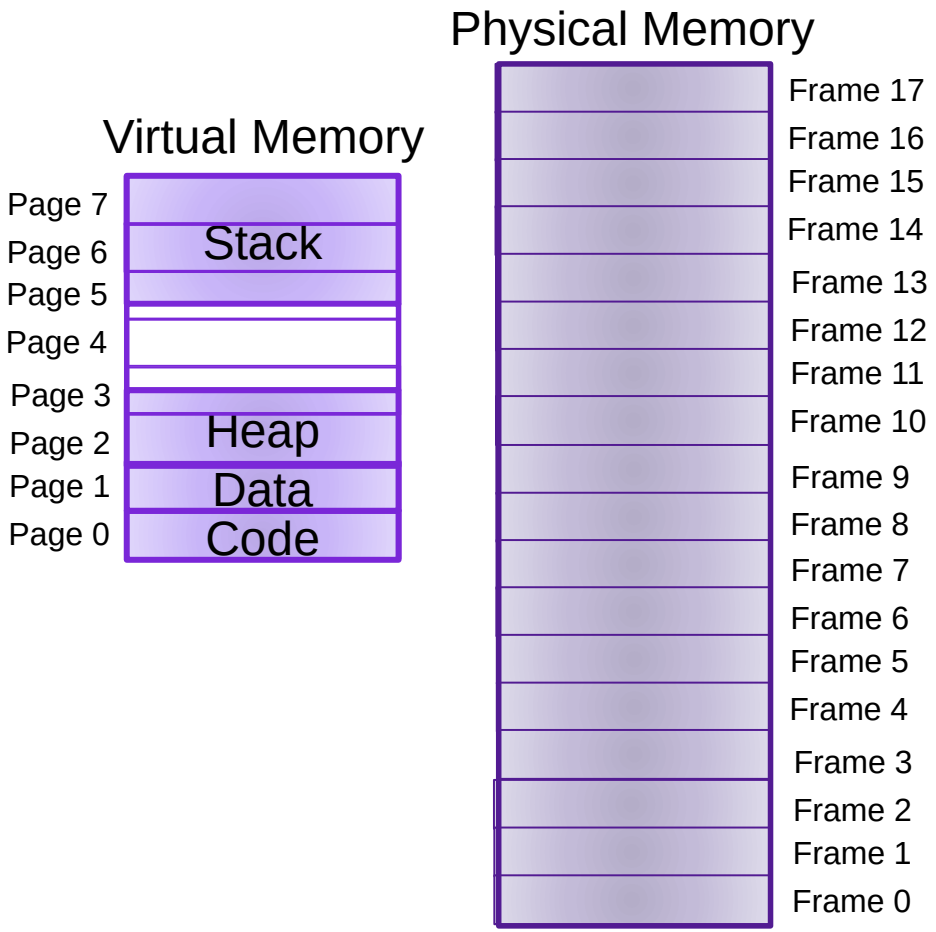
Paging



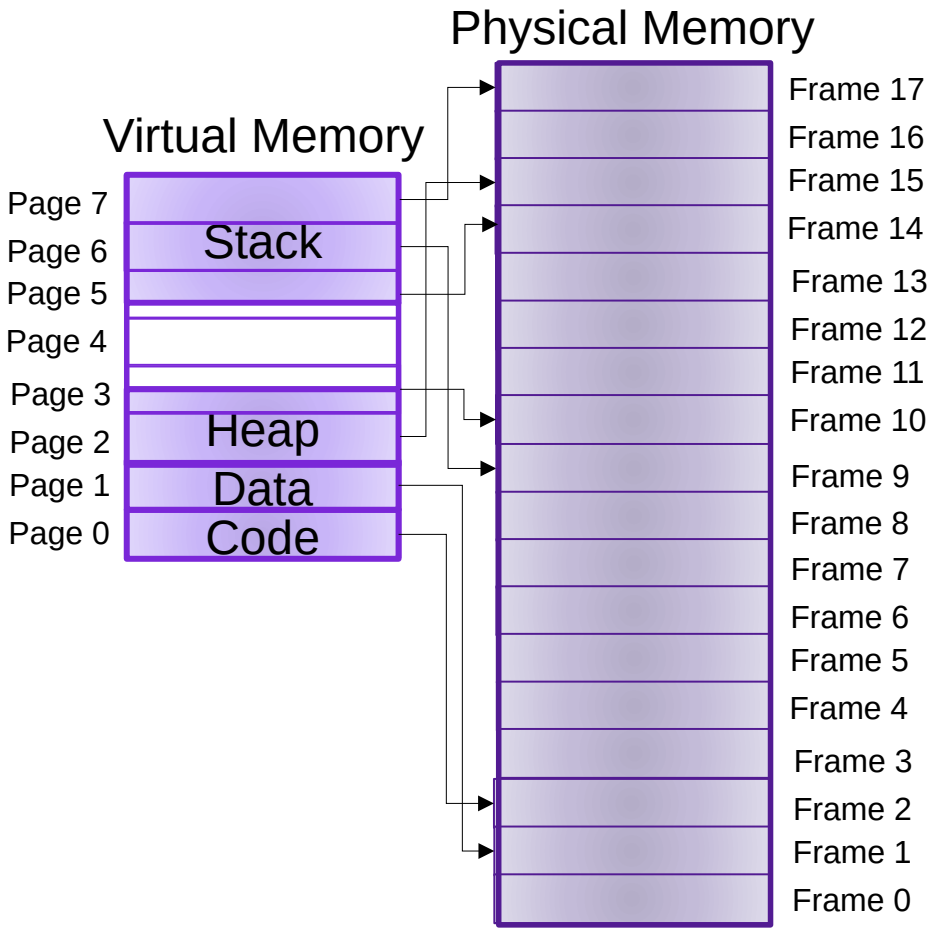
Paging


Physical Memory
Virtual Memory
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

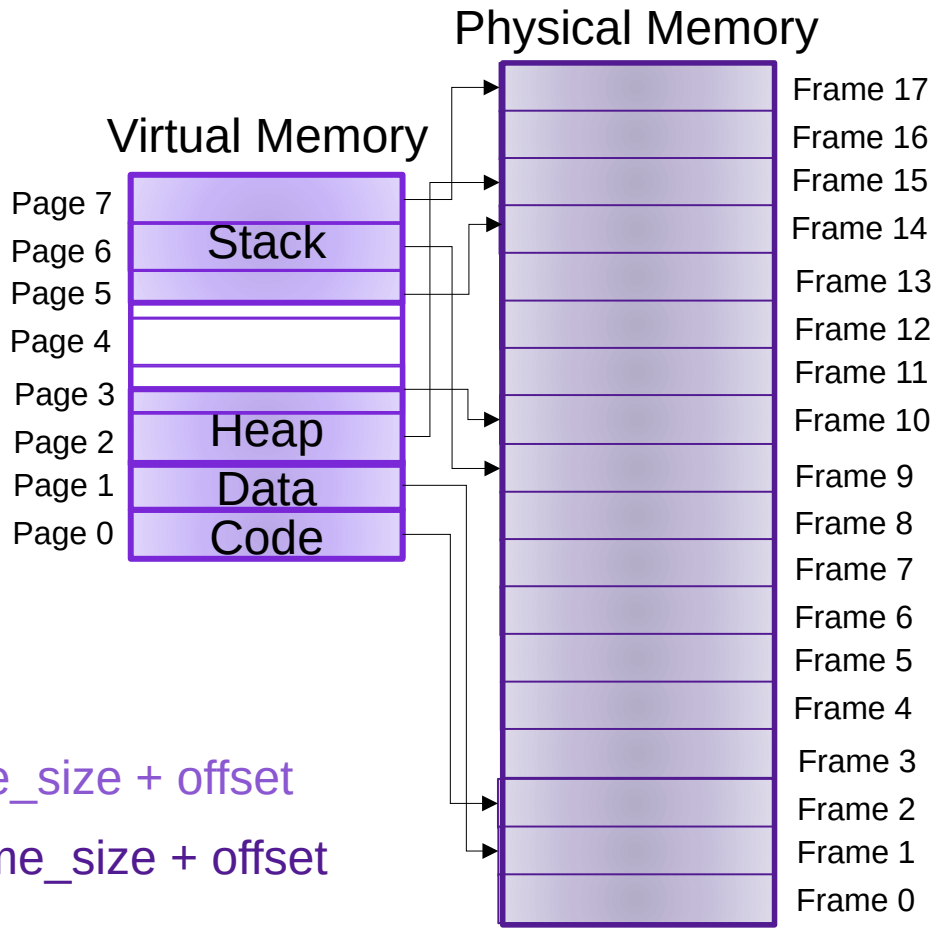
Paging


Virtual Memory
Page 7
Page 6
Page 5
Page 4
Page 3
Page 2
Page 1
Page 0
Stack
Heap
Data
Code
Physical Memory
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


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

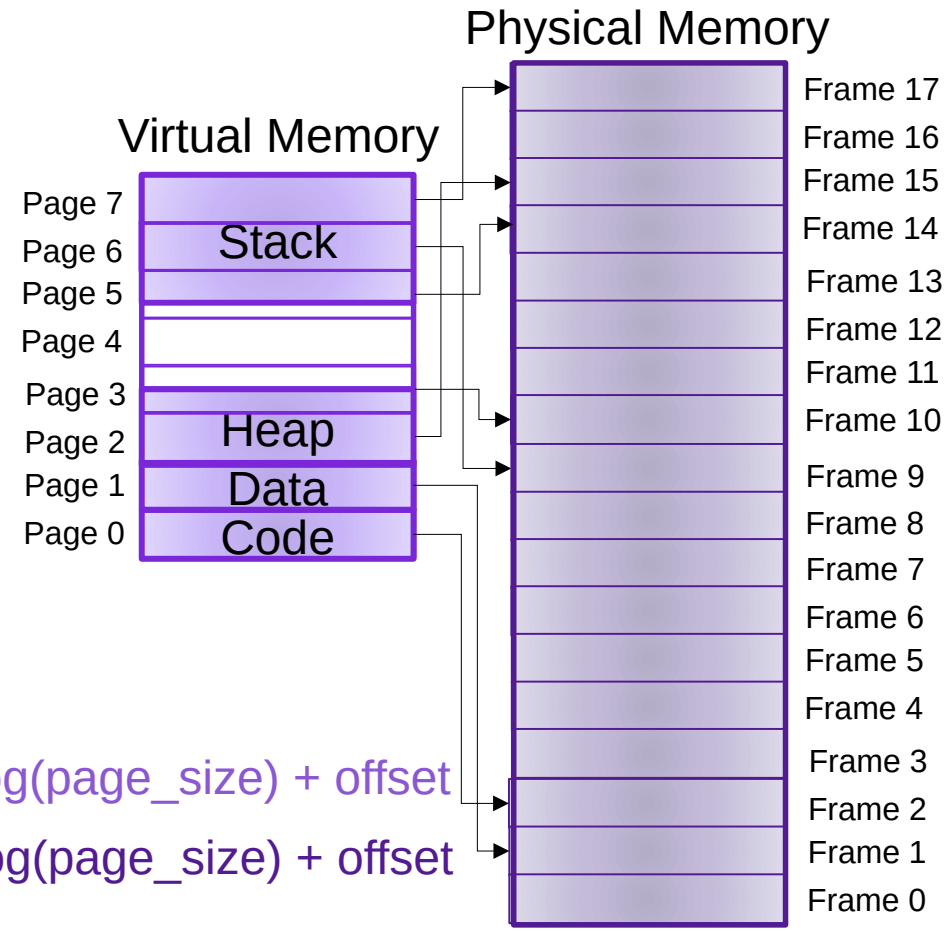
Paging



vaddr = page_num*page_size + offset

paddr = frame_num*frame_size + offset

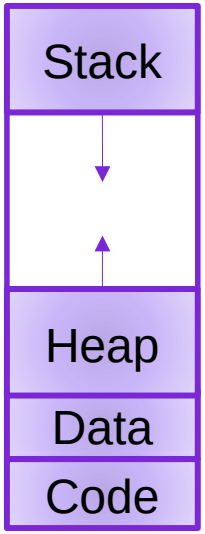
Paging



vaddr = page_num<< log(page_size) + offset

paddr = frame_num<<log(page_size) + offset

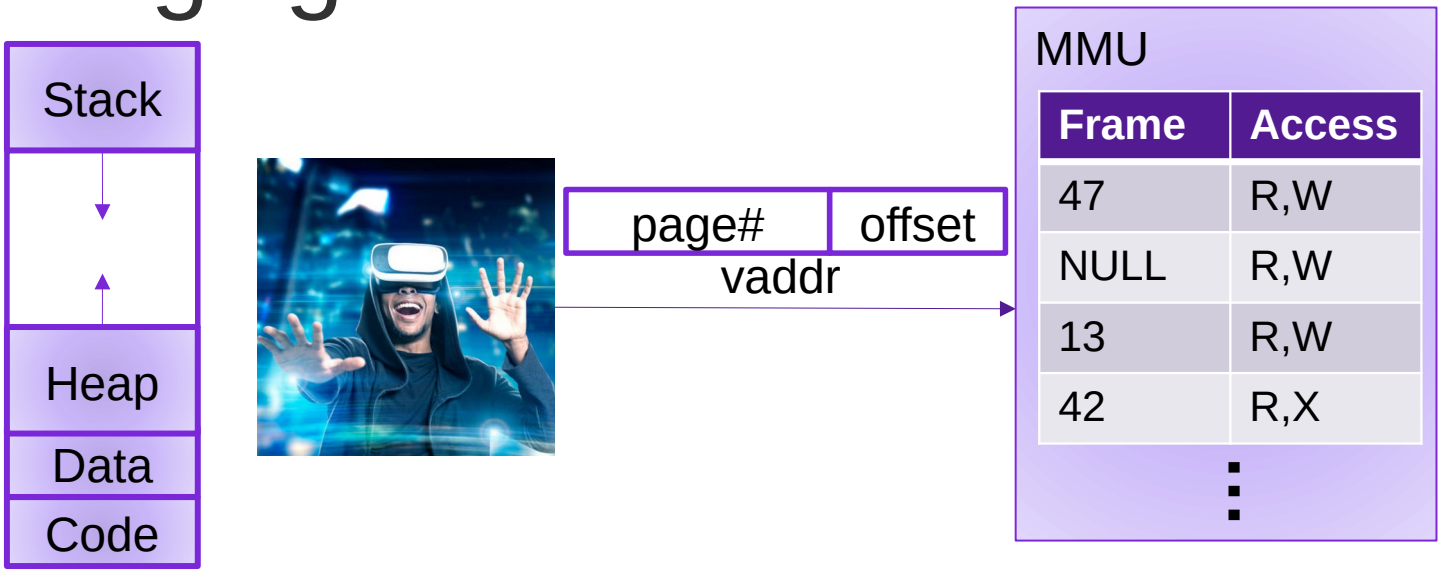
Paging



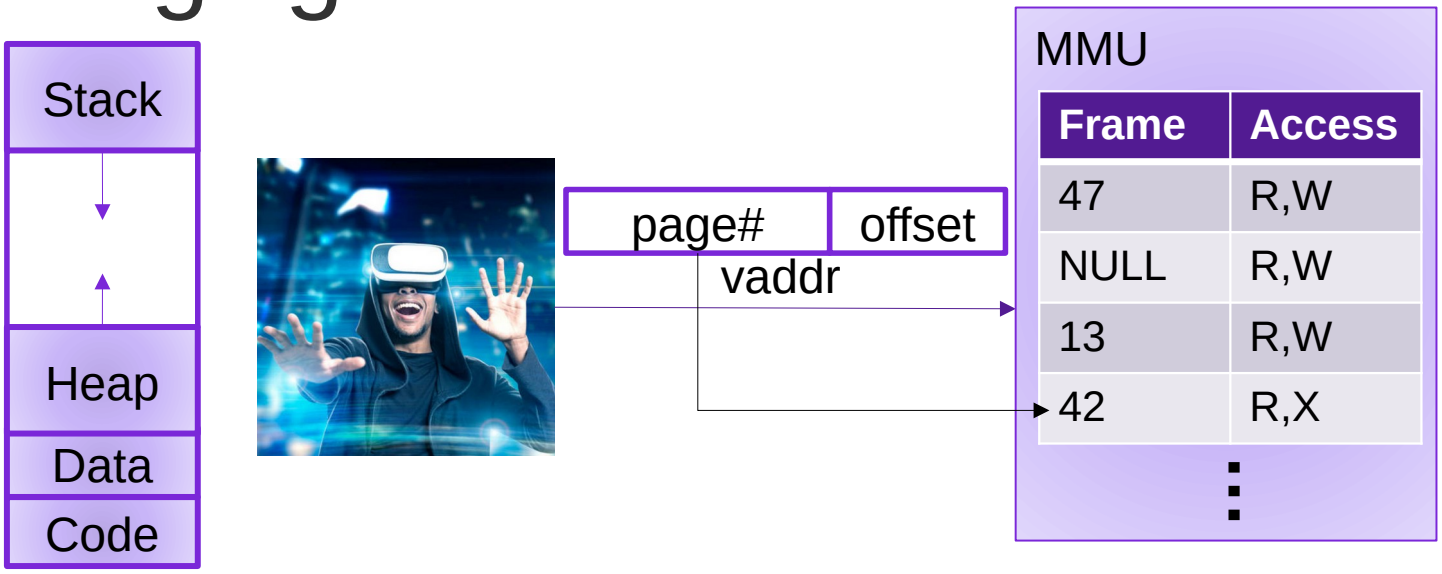
Paging


Stack
Heap
Data
Code
vaddr
MMU
Frame
Access
47
R,W
NULL
R,W
13
R,W
42
R,X

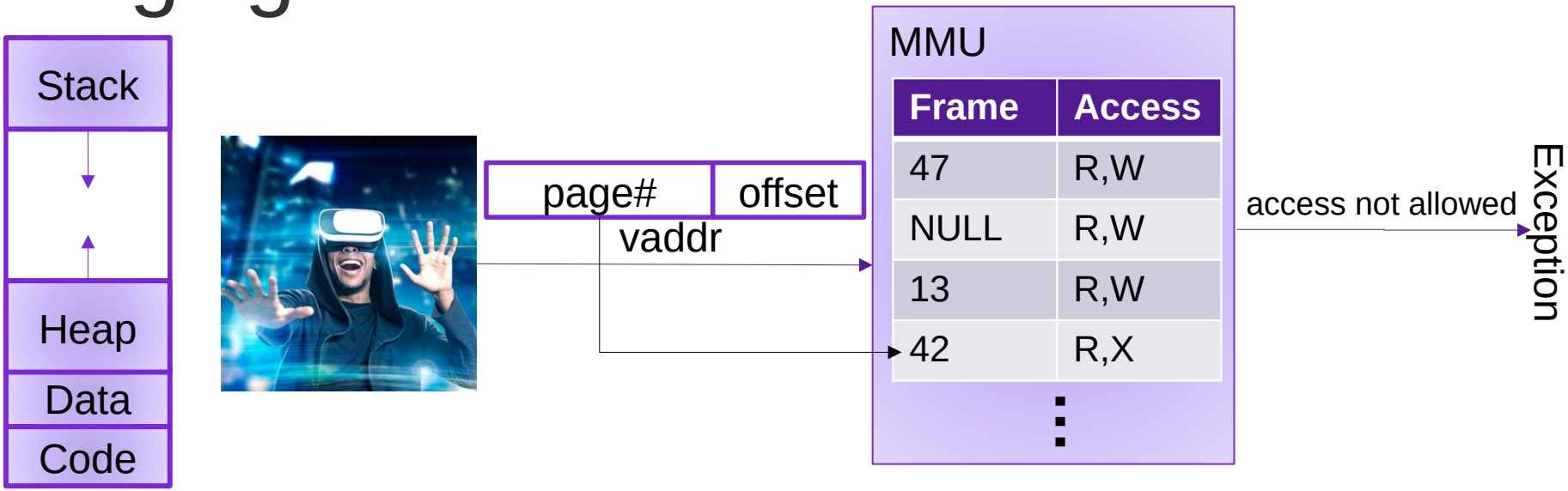
Paging


Stack
Heap
Data
Code
page#
offset
vaddr
MMU
Frame
Access
47
R,W
NULL
R,W
13
R,W
42
R,X

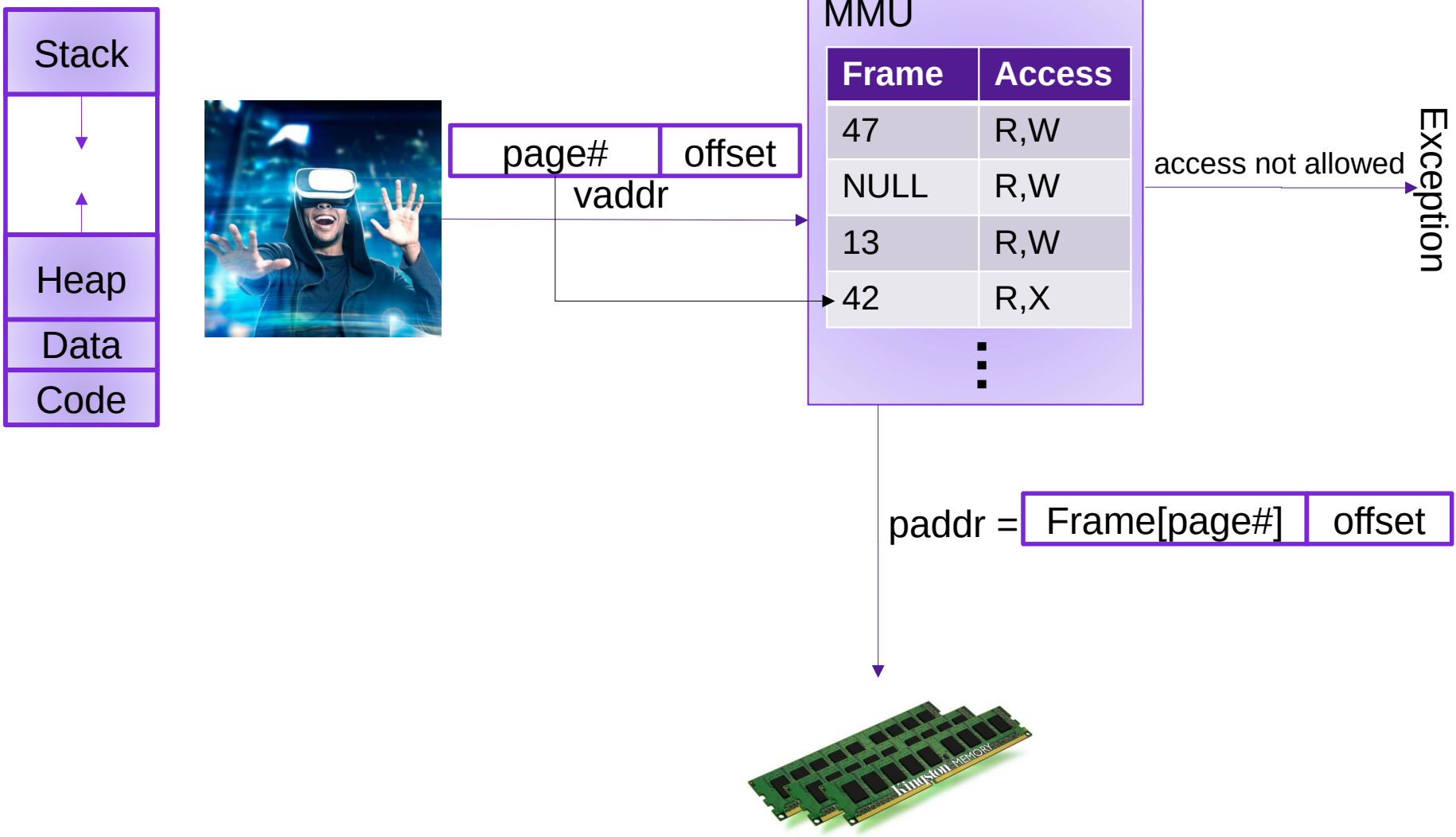
Paging


Stack
Heap
Data
Code
page#
offset
vaddr
MMU
Frame
Access
47
R,W
NULL
R,W
13
R,W
42
R,X

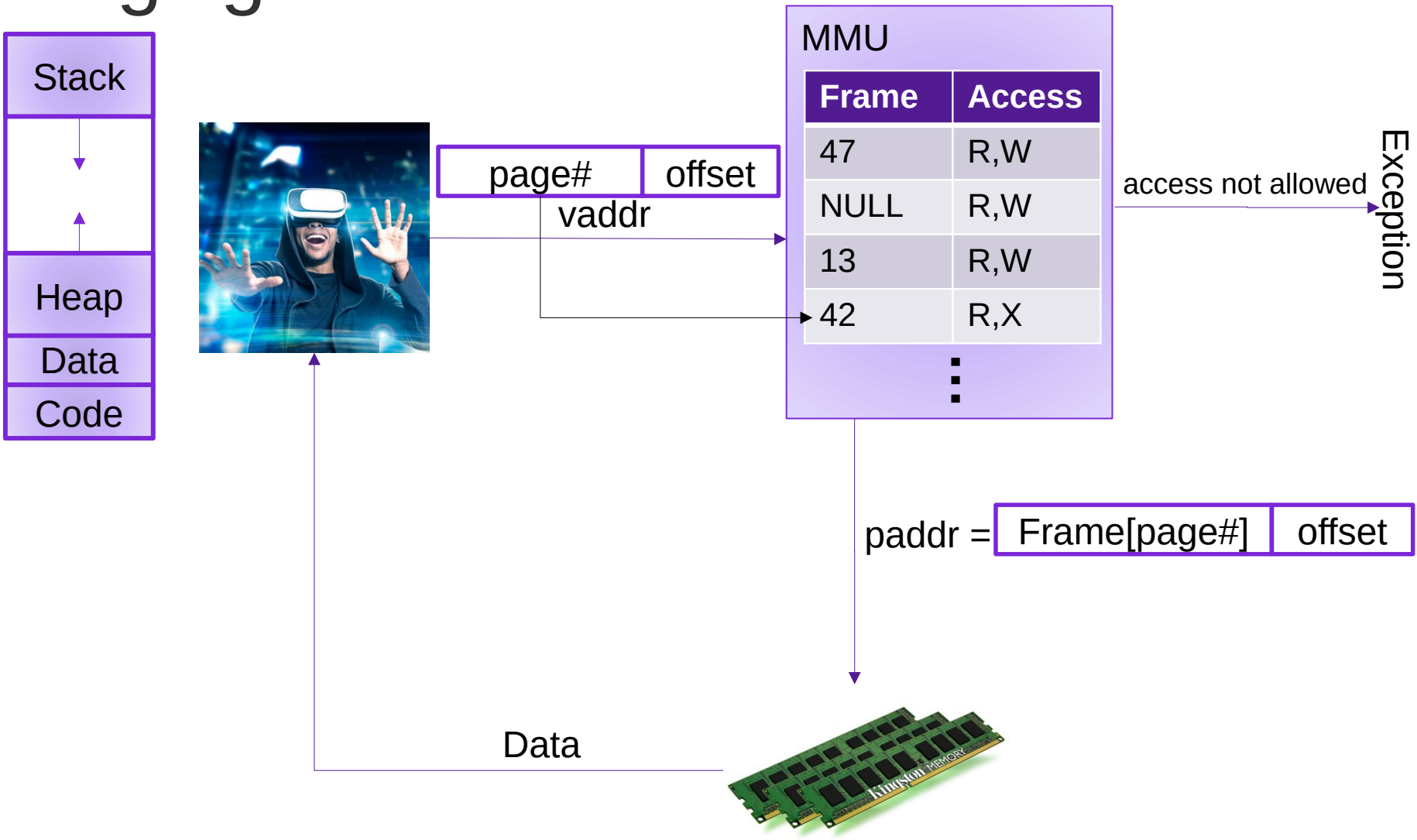
Paging


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

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
Kingston MEMORY

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?
- 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?

- 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?

0x237

- 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

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

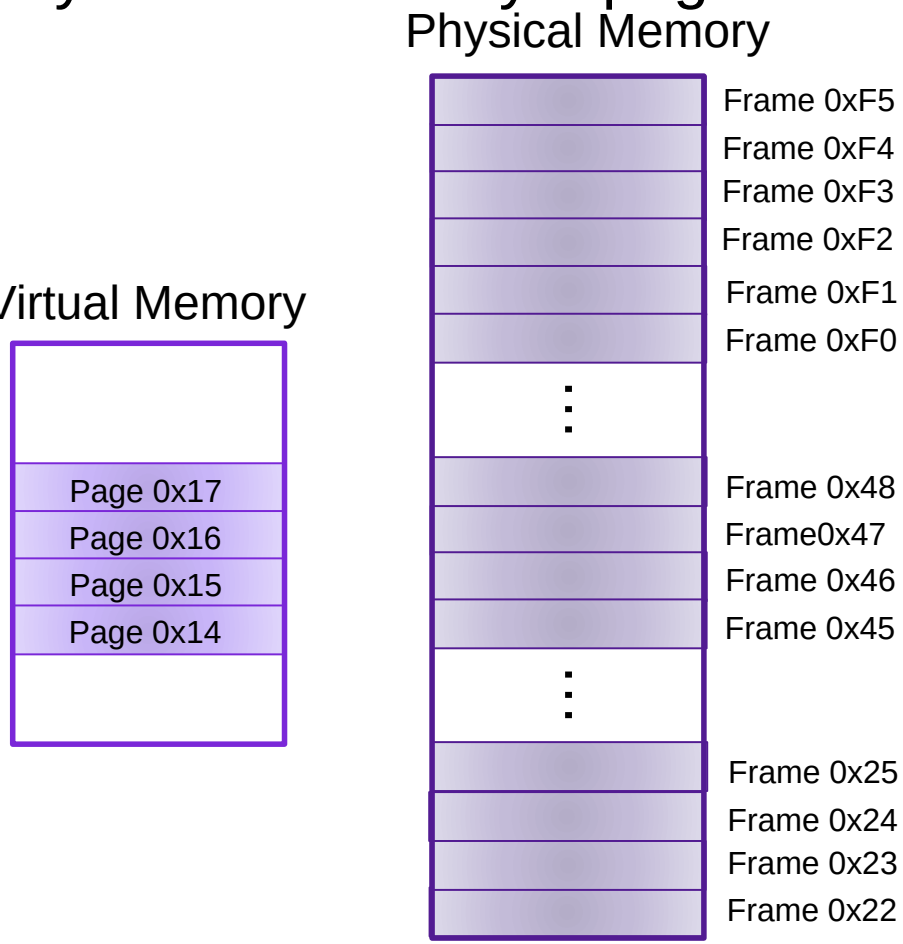
Assume that you are currently executing a process P with the following page table on a system with 16 byte pages:

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

Exercise 3: Paging

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

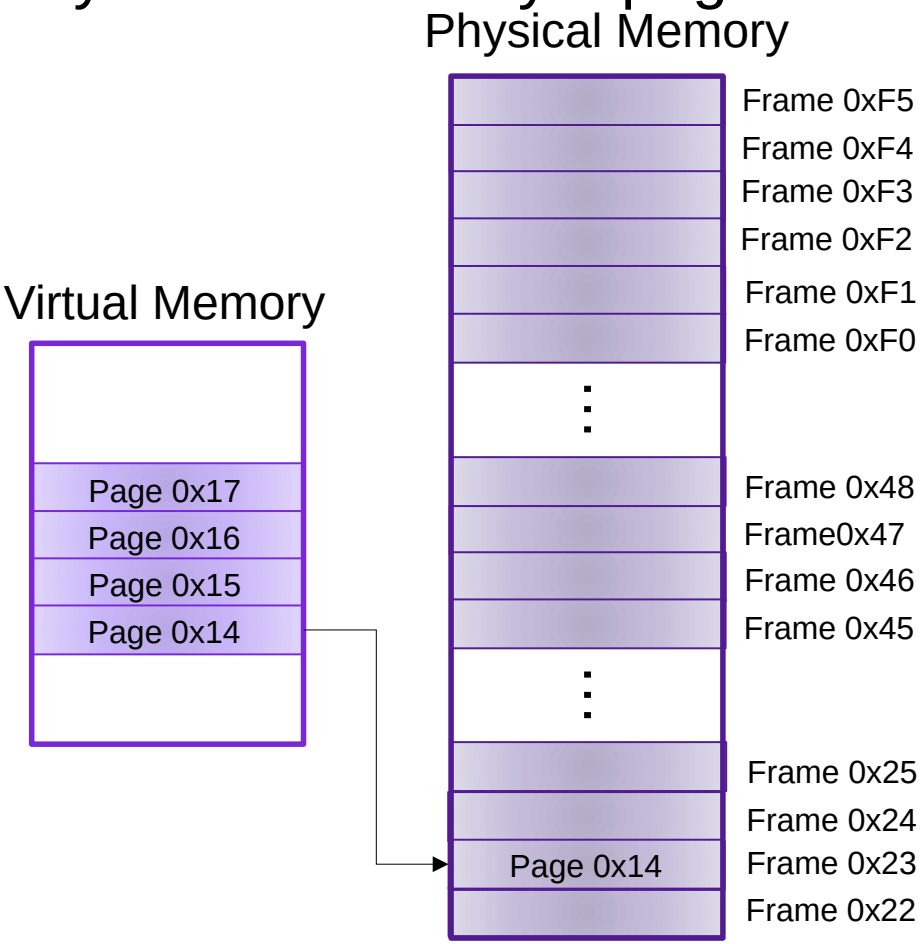
	Frame	Access
⋮		
0x17	0x47	R,W
0x16	0xF4	R,W
0x15	NULL	R,W
0x14	0x23	R,X
⋮		



Exercise 3: Paging

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

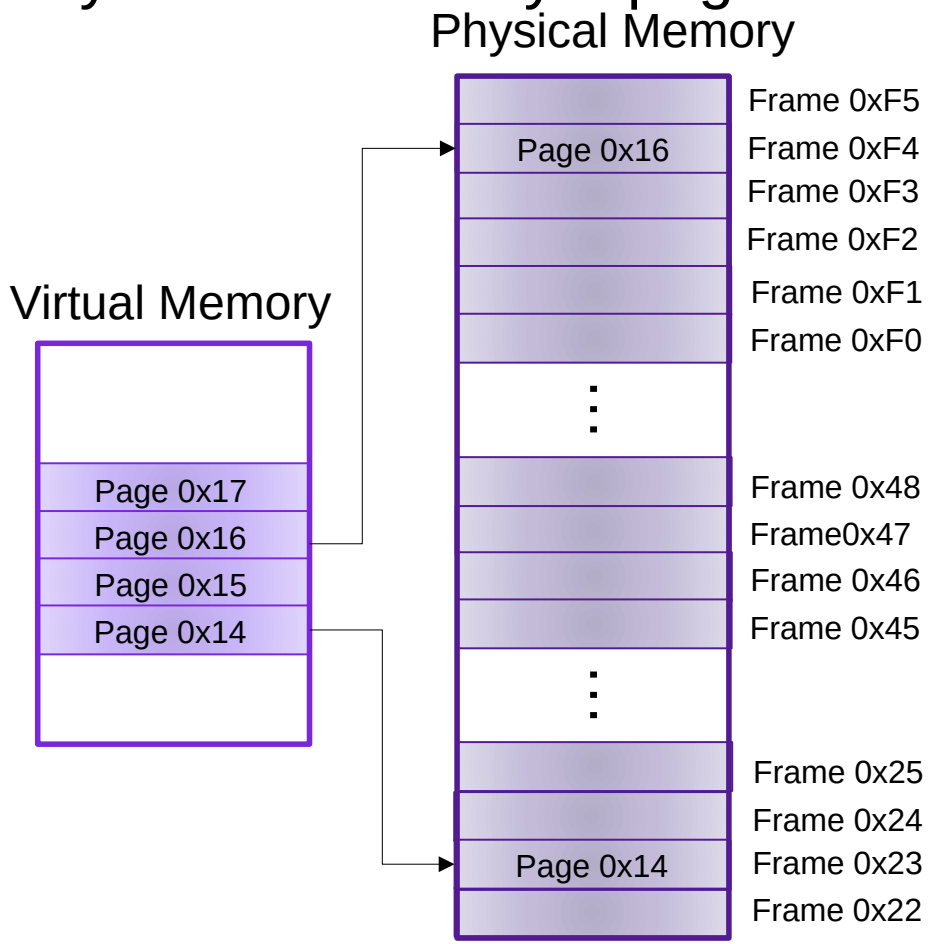
	Frame	Access
⋮		
0x17	0x47	R,W
0x16	0xF4	R,W
0x15	NULL	R,W
0x14	0x23	R,X
⋮		



Exercise 3: Paging

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

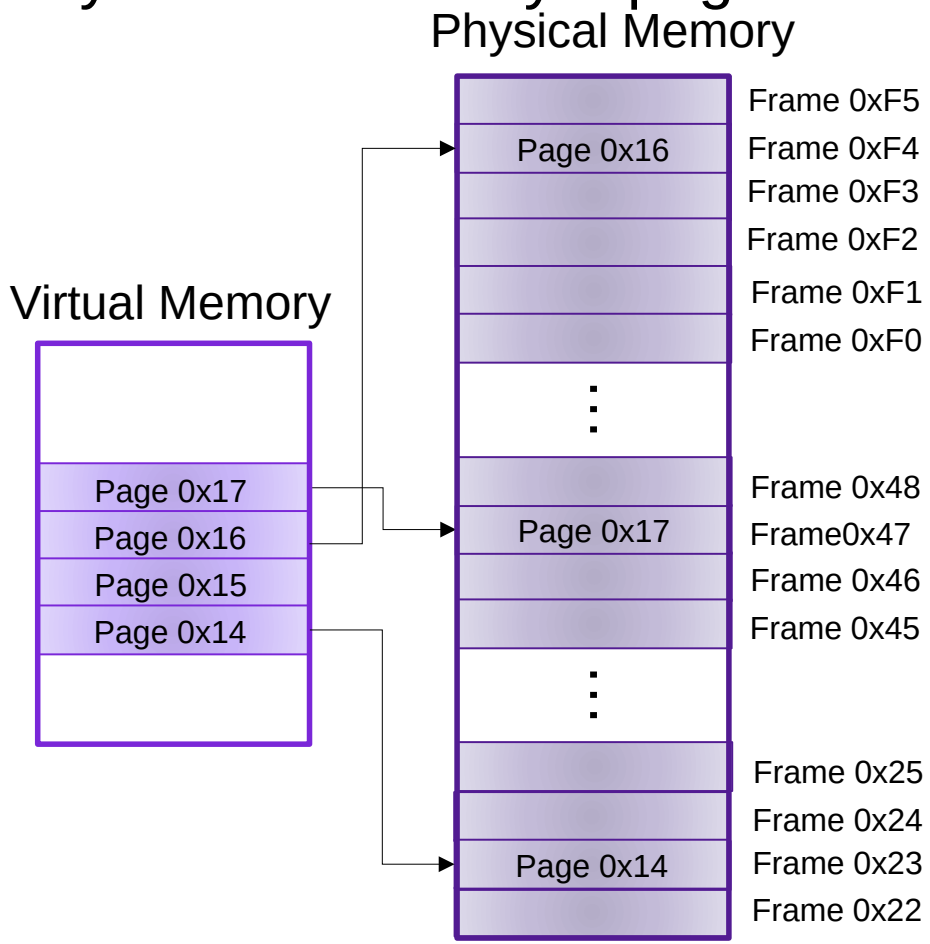
	Frame	Access
⋮		
0x17	0x47	R,W
0x16	0xF4	R,W
0x15	NULL	R,W
0x14	0x23	R,X
⋮		



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

MMU

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

MMU

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

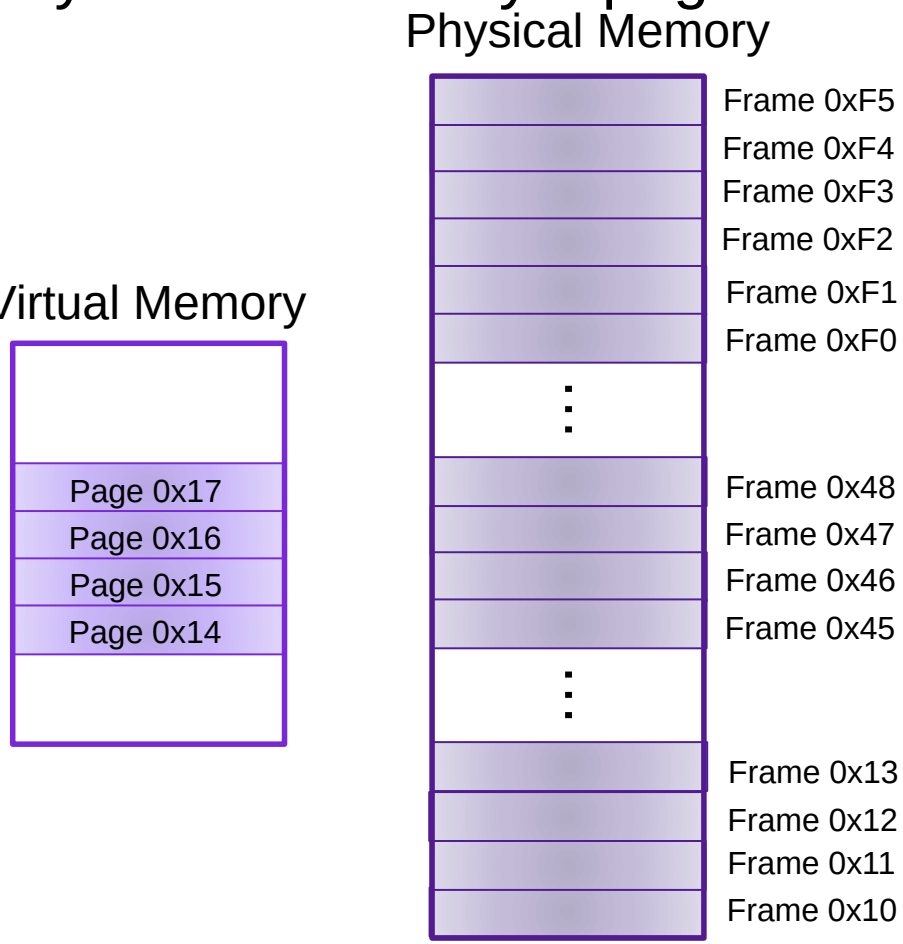
MMU

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:

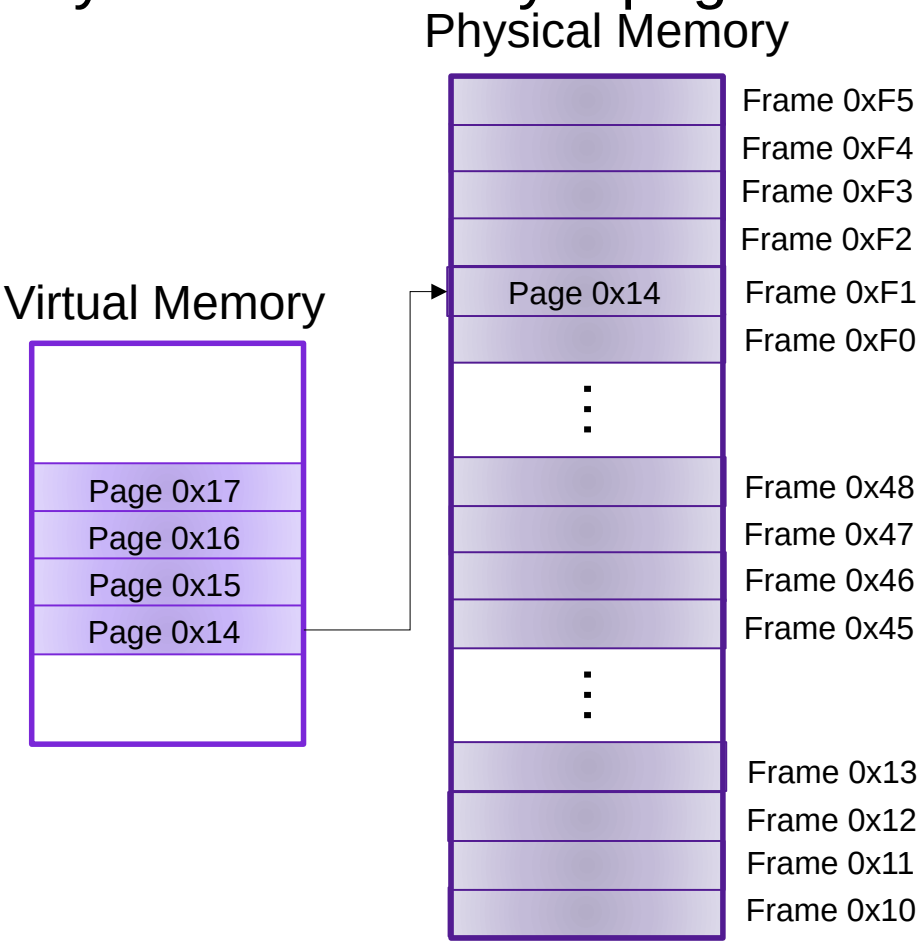
	v	Frame	Access
⋮			
0x17	1	0x47	R,W
0x16	0	NULL	R,W
0x15	0	0x13	R,W
0x14	1	0xF1	R,X
⋮			



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:

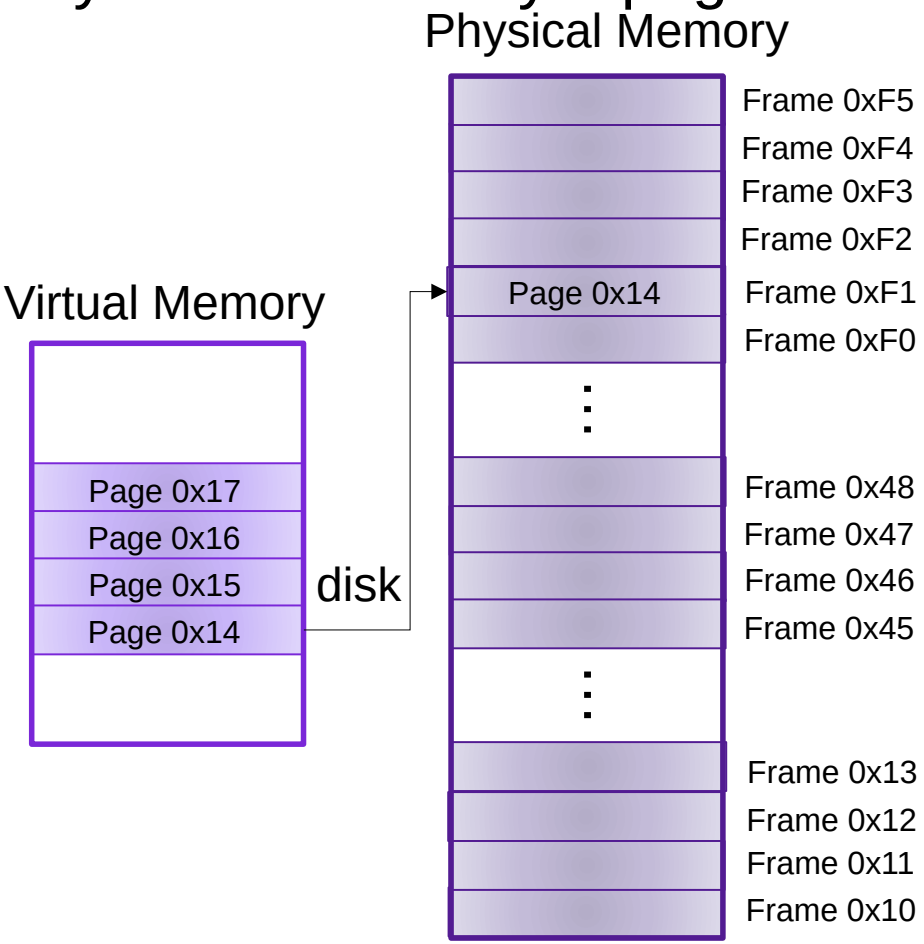
	v	Frame	Access
⋮			
0x17	1	0x47	R,W
0x16	0	NULL	R,W
0x15	0	0x13	R,W
0x14	1	0xF1	R,X
⋮			



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:

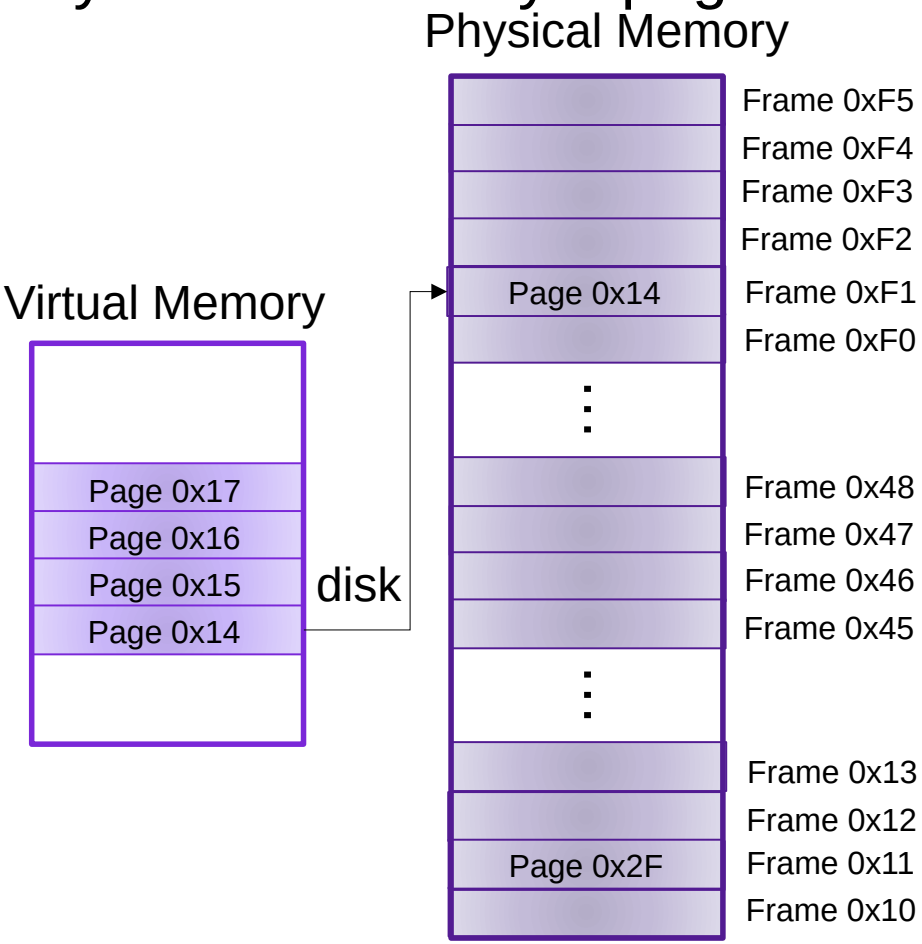
	v	Frame	Access
⋮			
0x17	1	0x47	R,W
0x16	0	NULL	R,W
0x15	0	0x13	R,W
0x14	1	0xF1	R,X
⋮			



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:

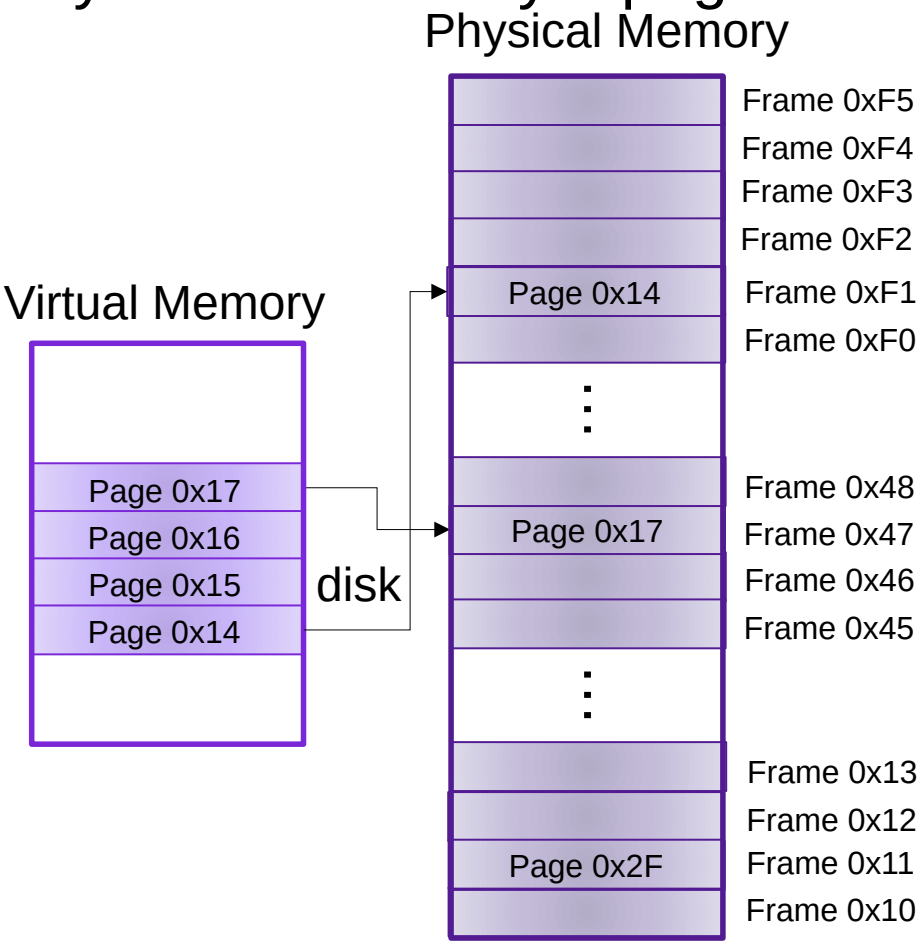
	v	Frame	Access
⋮			
0x17	1	0x47	R,W
0x16	0	NULL	R,W
0x15	0	0x13	R,W
0x14	1	0xF1	R,X
⋮			



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:

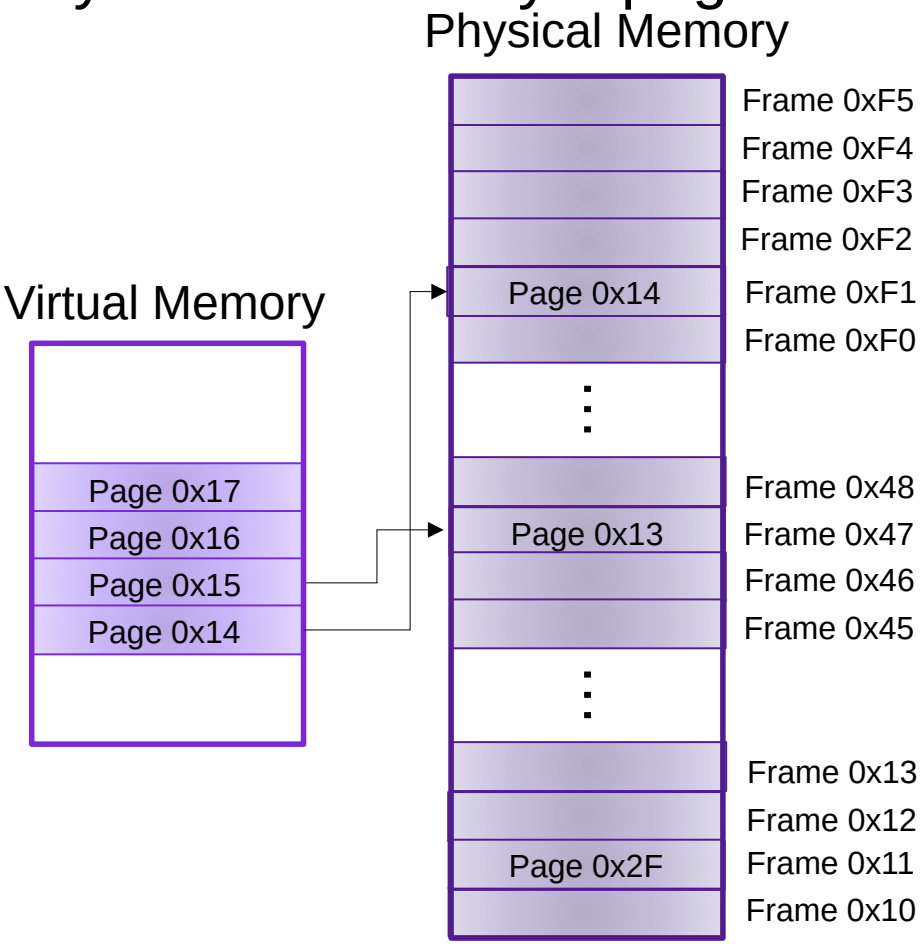
	v	Frame	Access
⋮			
0x17	1	0x47	R,W
0x16	0	NULL	R,W
0x15	0	0x13	R,W
0x14	1	0xF1	R,X
⋮			



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	0	0x47	R,W
0x16	0	NULL	R,W
0x15	1	0x47	R,W
0x14	1	0xF1	R,X
⋮			



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?

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?

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?

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?

0x2447

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

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

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

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

page fault

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

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

page fault

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

0xF8	0x13

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

page fault

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

0xF8	0x13

segfault

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

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

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

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

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

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