

# Operating Systems

Limited Direct Execution + Memory

Virtualisation

Nipun Batra

# Administrative

---

1. Next Wednesday answer sheets in lab session
2. Projects - list would be available on Monday
  1. Project 5 -> 8% (3% reduced from homework)
  2. More details on Tuesday...

# Space v/s Time Multiplexing

---

Time multiplexing : Share resource by dividing over time

# Space v/s Time Multiplexing

---

Time multiplexing : Share resource by dividing over time

1. CPU scheduling on single core

# Space v/s Time Multiplexing

---

Time multiplexing : Share resource by dividing over time

1. CPU scheduling on single core
2. Think more?!

# Space v/s Time Multiplexing

---

Time multiplexing : Share resource by dividing over time

1. CPU scheduling on single core
2. Think more?!
3. Class room scheduling - single class runs in at any given point of time

# Space v/s Time Multiplexing

---

Time multiplexing : Share resource by dividing over time

1. CPU scheduling on single core
2. Think more?!
3. Class room scheduling - single class runs in at any given point of time
4. TDMA??

# Space v/s Time Multiplexing

---

Space multiplexing : Share resource by dividing into smaller pieces



# Space v/s Time Multiplexing

---

Space multiplexing : Share resource by dividing into smaller pieces

1. CPU scheduling on multiple cores?

# Space v/s Time Multiplexing

---

Space multiplexing : Share resource by dividing into smaller pieces

1. CPU scheduling on multiple cores?
2. Cake sharing

# Space v/s Time Multiplexing

---

Space multiplexing : Share resource by dividing into smaller pieces

1. CPU scheduling on multiple cores?
2. Cake sharing
3. Think more?

# Space v/s Time Multiplexing

---

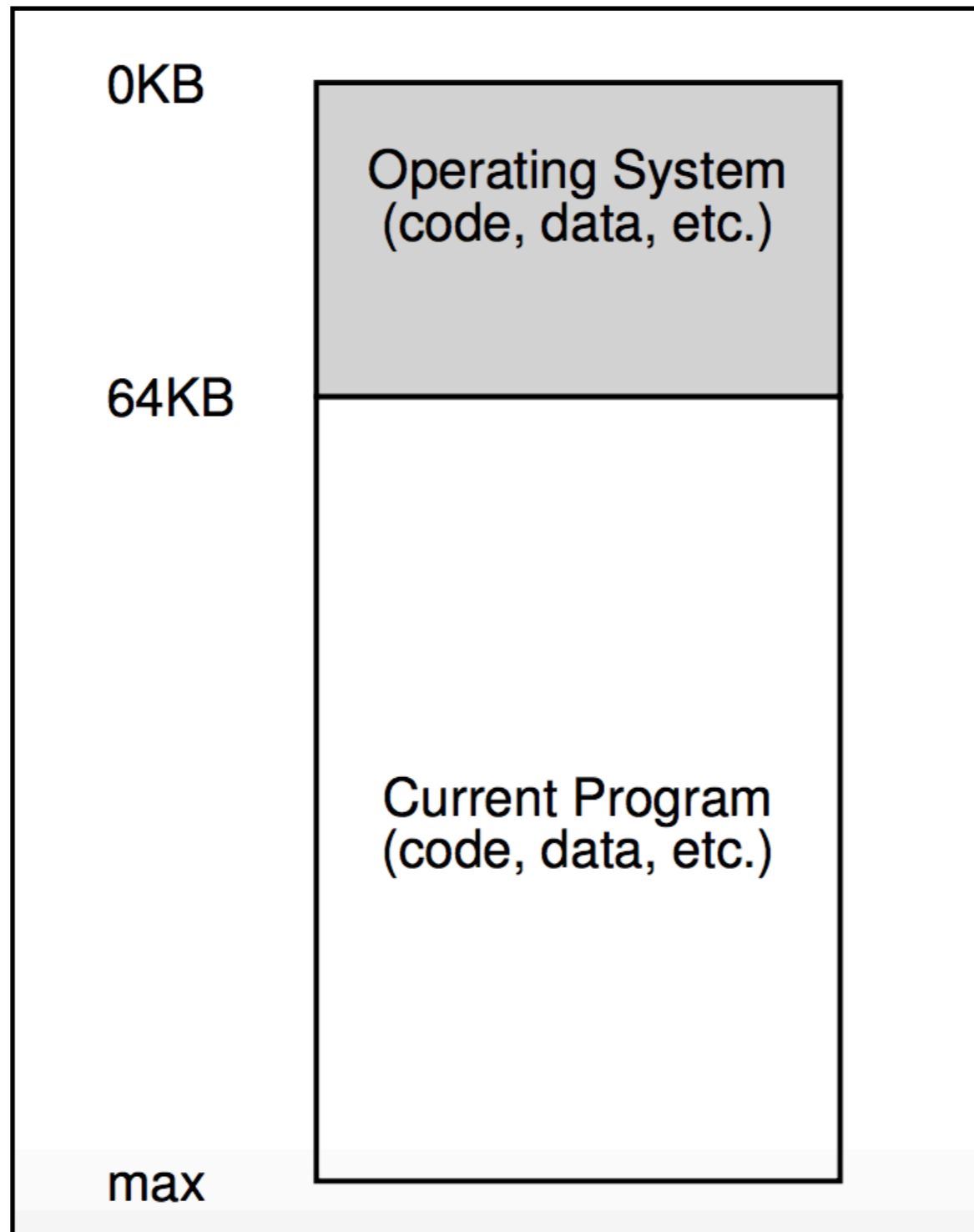
Space multiplexing : Share resource by dividing into smaller pieces

1. CPU scheduling on multiple cores?
2. Cake sharing
3. Think more?
4. Memory management

# Memory Virtualisation

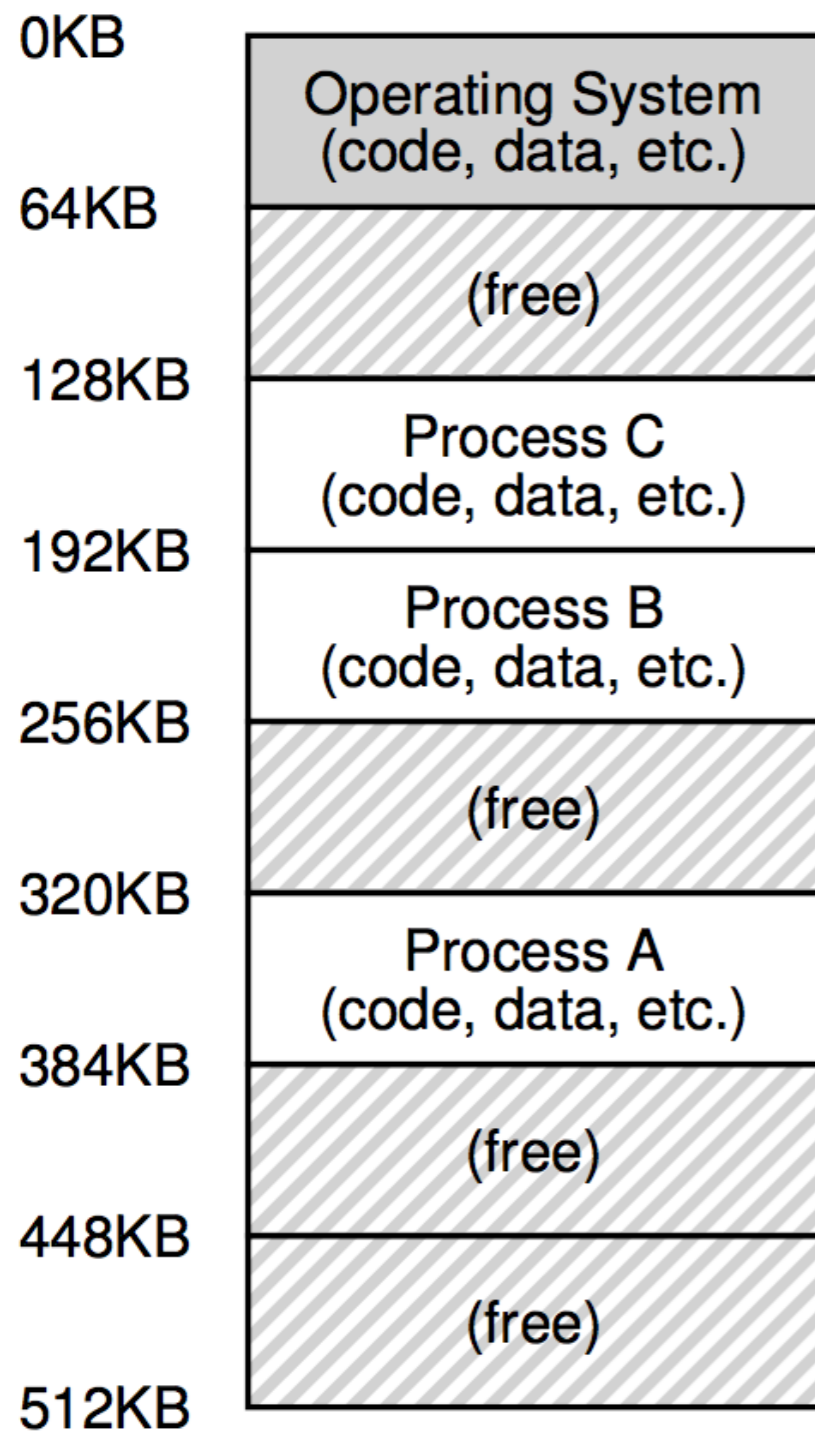
---

Early days  
Single program



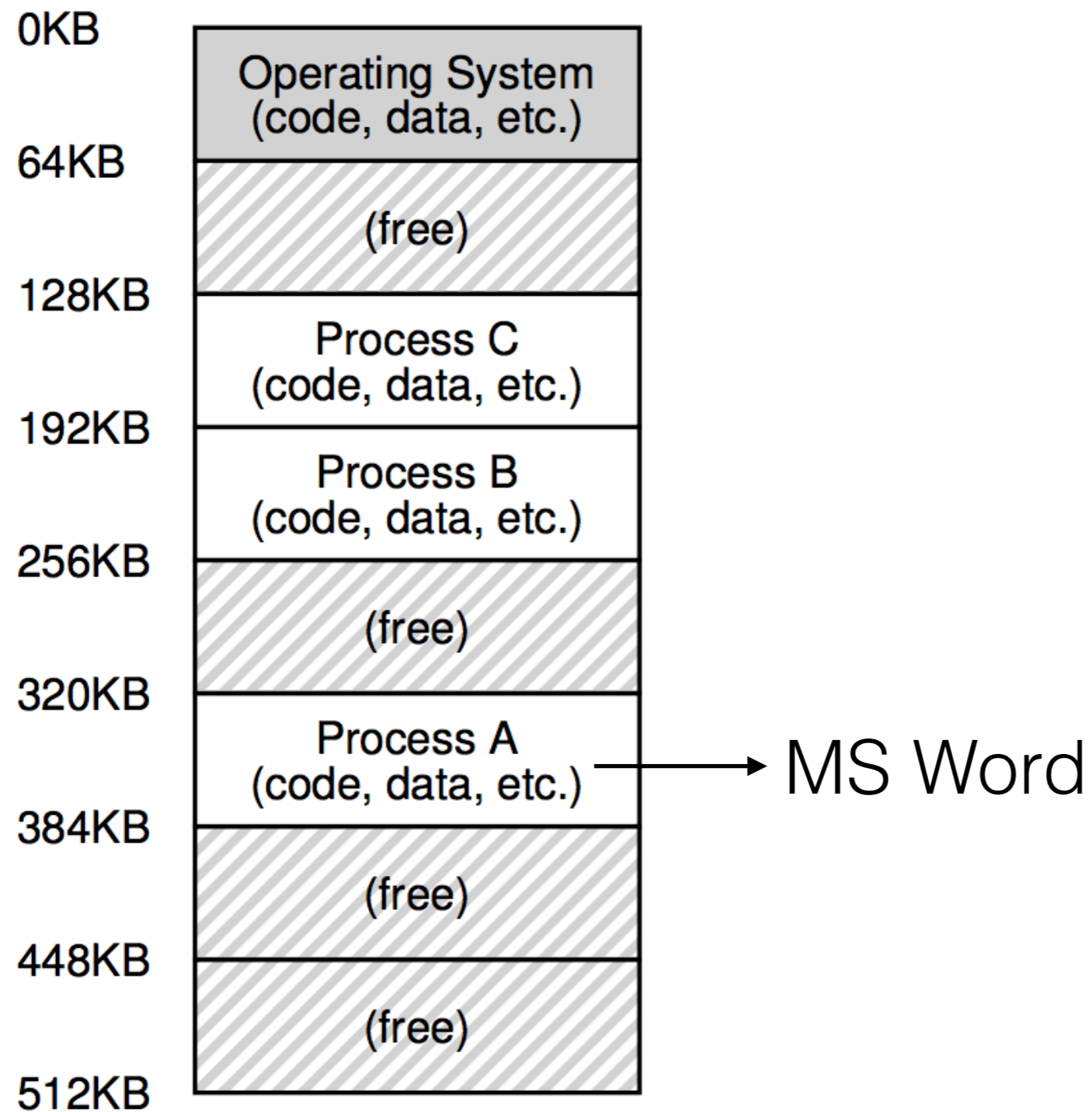
# Shared Memory

---



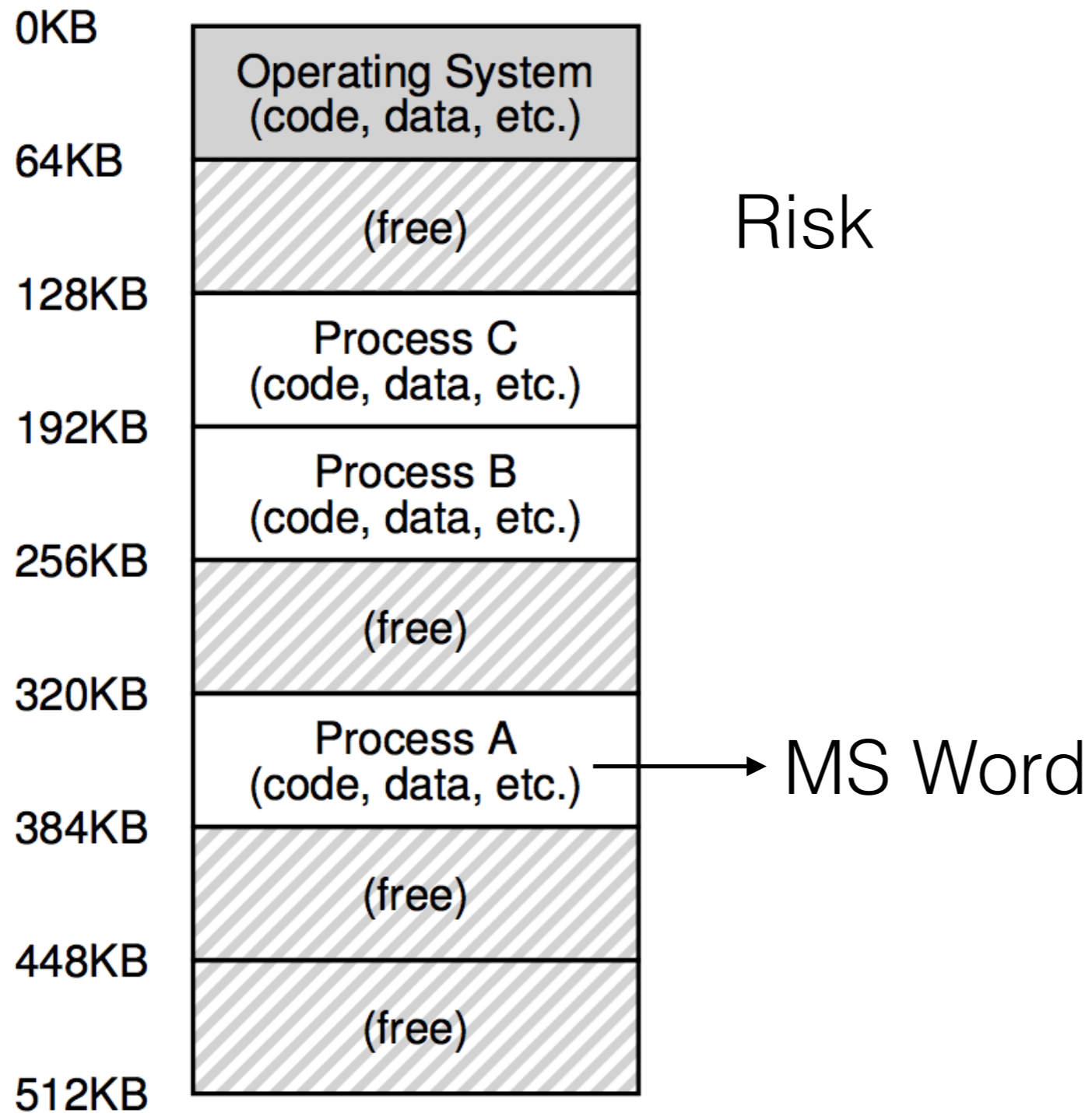
# Shared Memory

---



# Shared Memory

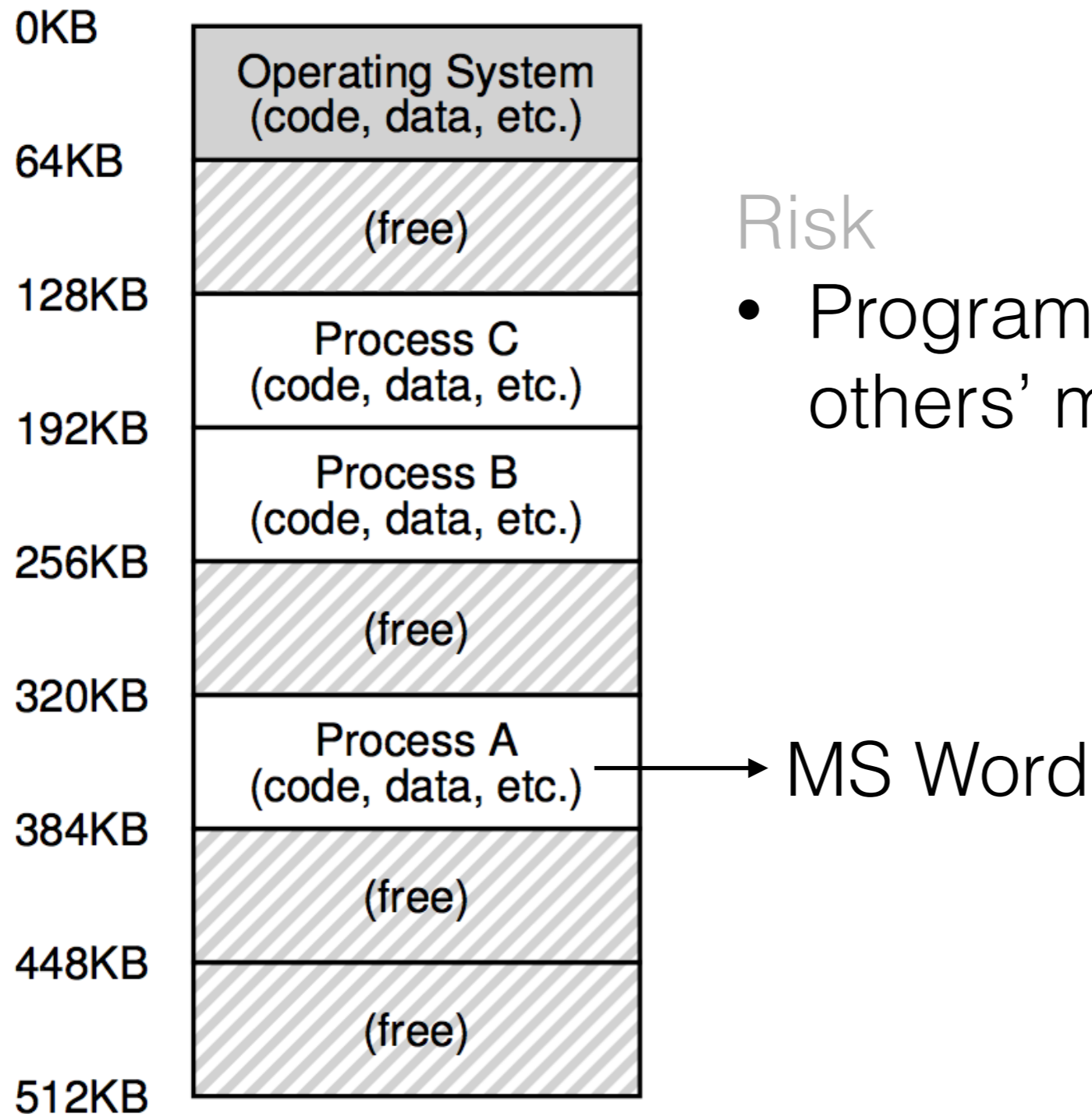
---





# Shared Memory

---



## Risk

- Programs accessing others' memory

# Direct Physical Memory Multiplexing

---



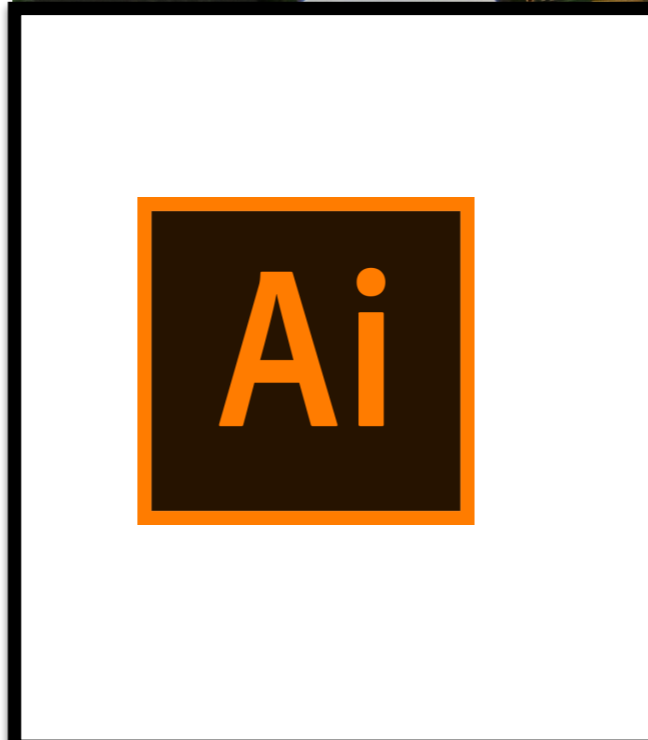
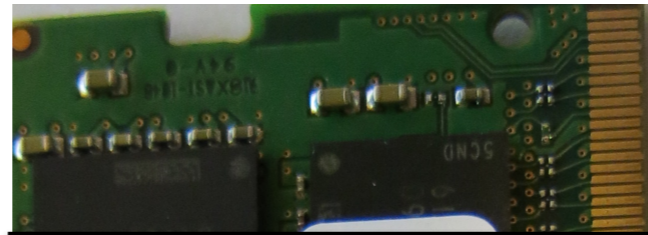
# Direct Physical Memory Multiplexing

---



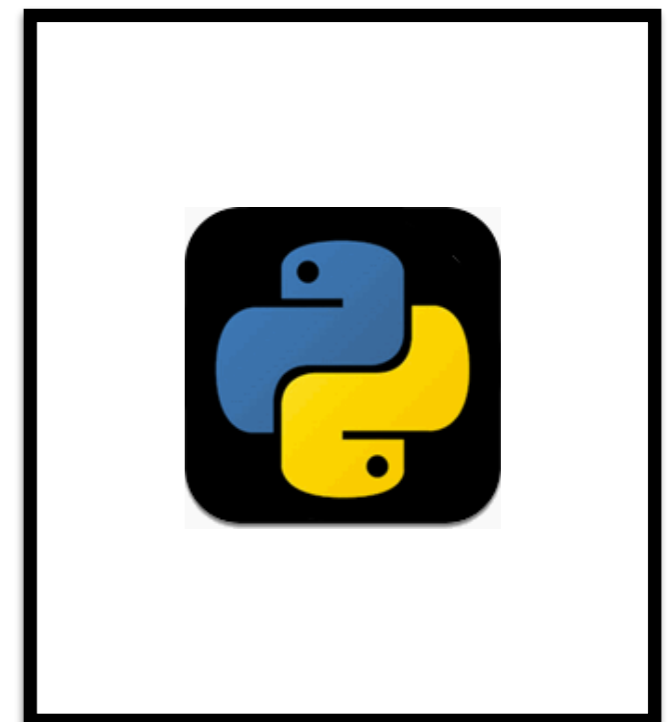
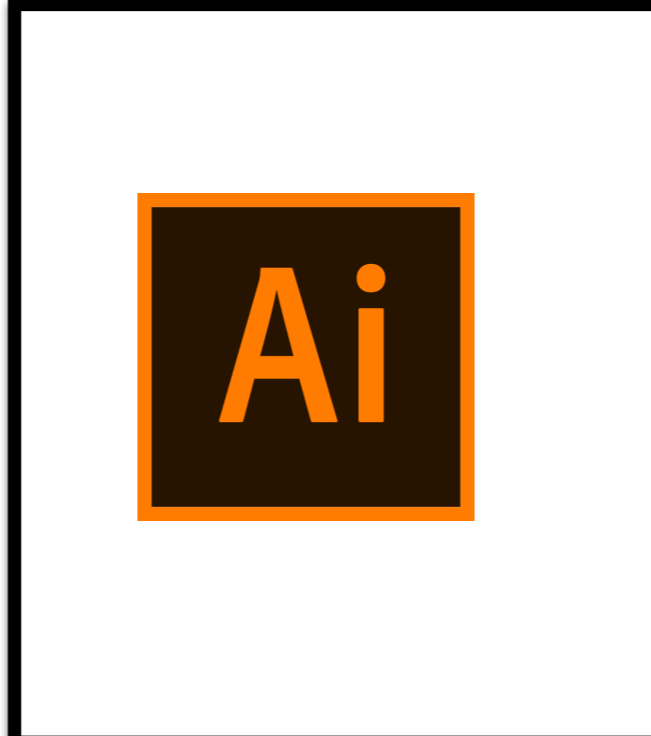
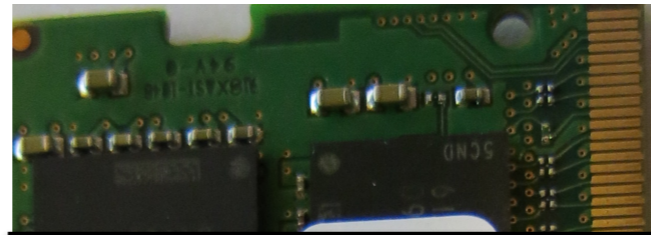
# Direct Physical Memory Multiplexing

---



# Direct Physical Memory Multiplexing

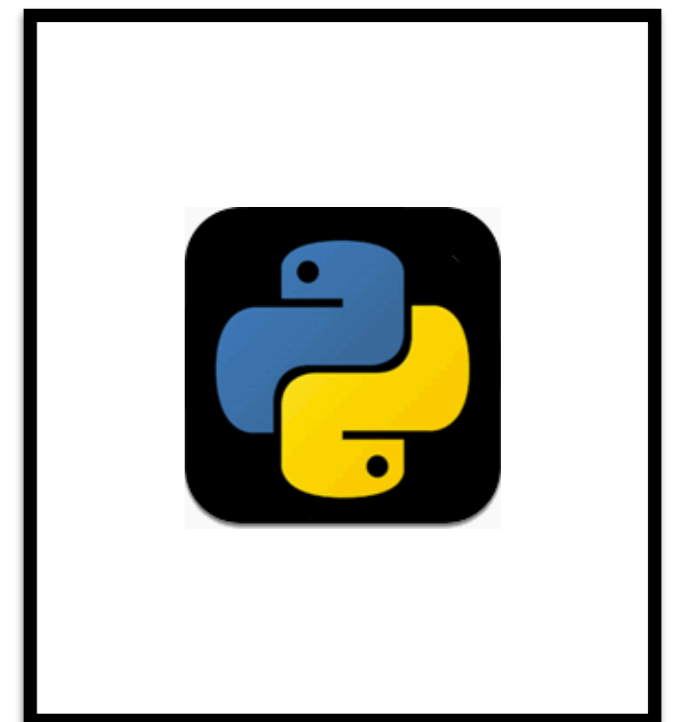
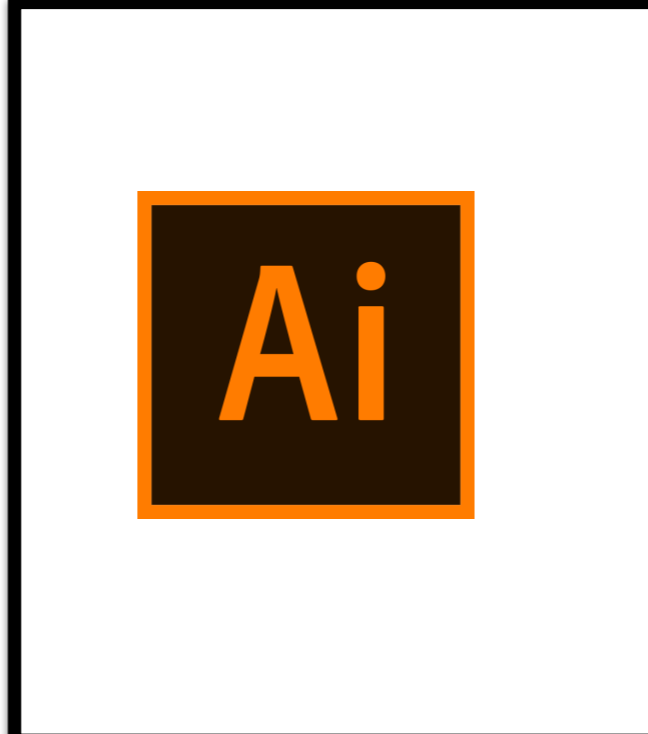
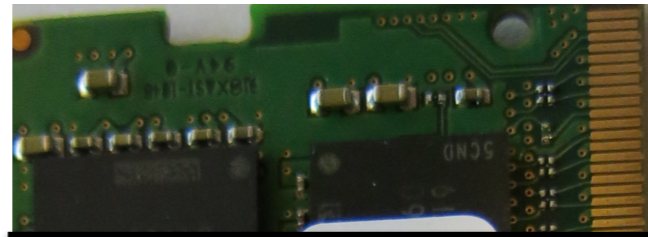
---



# Direct Physical Memory Multiplexing

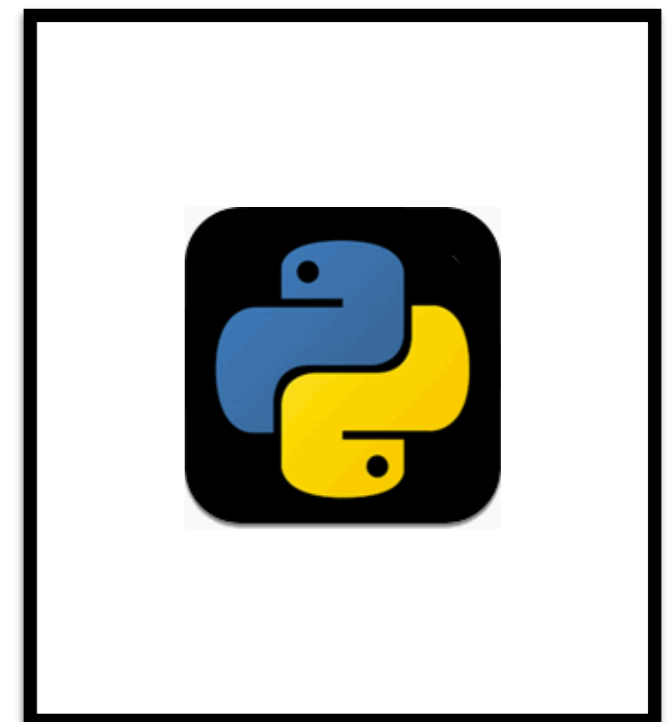
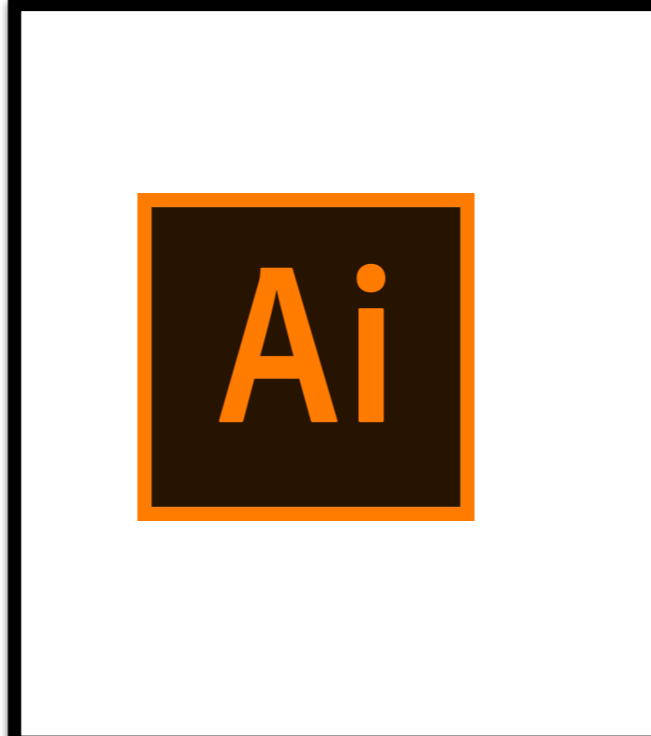
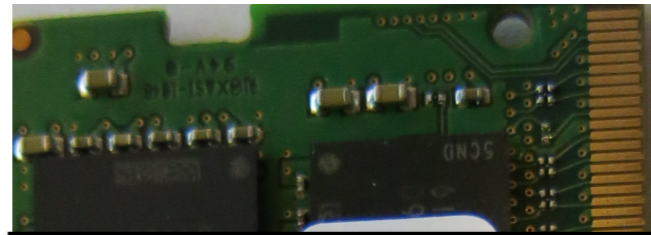
---

Limited to  
physical memory  
on the system



# Direct Physical Memory Multiplexing

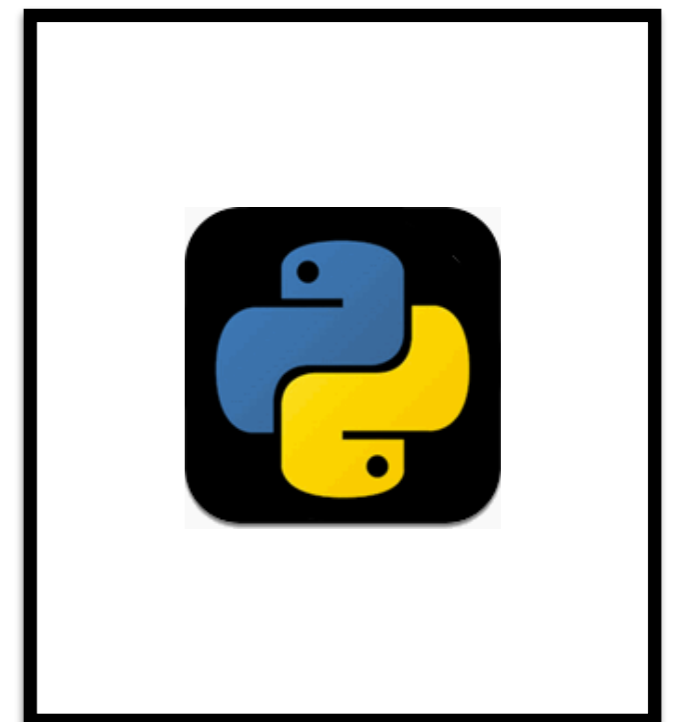
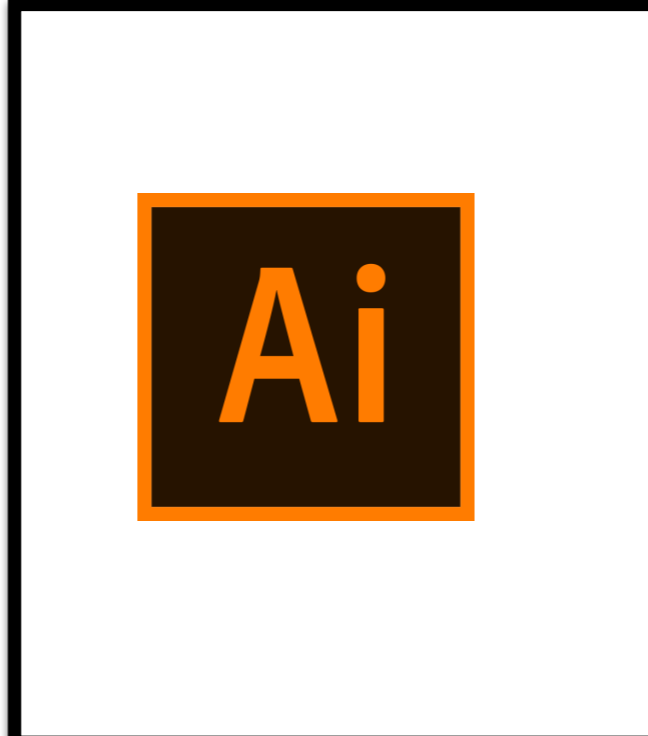
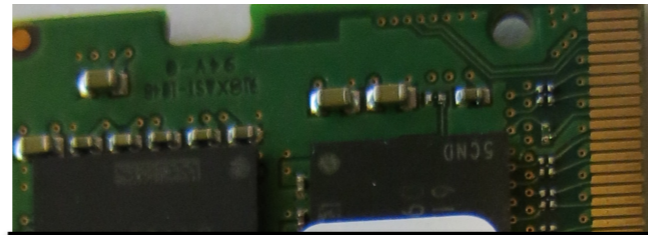
---



# Direct Physical Memory Multiplexing

---

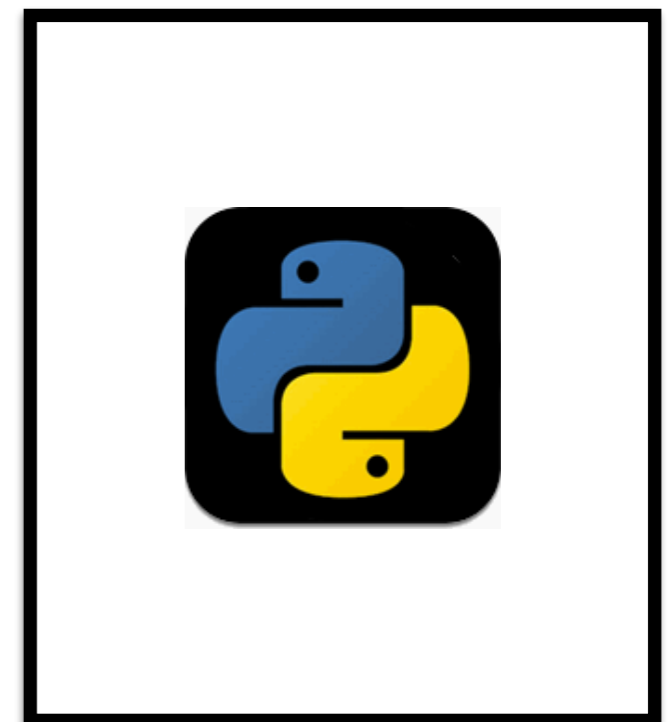
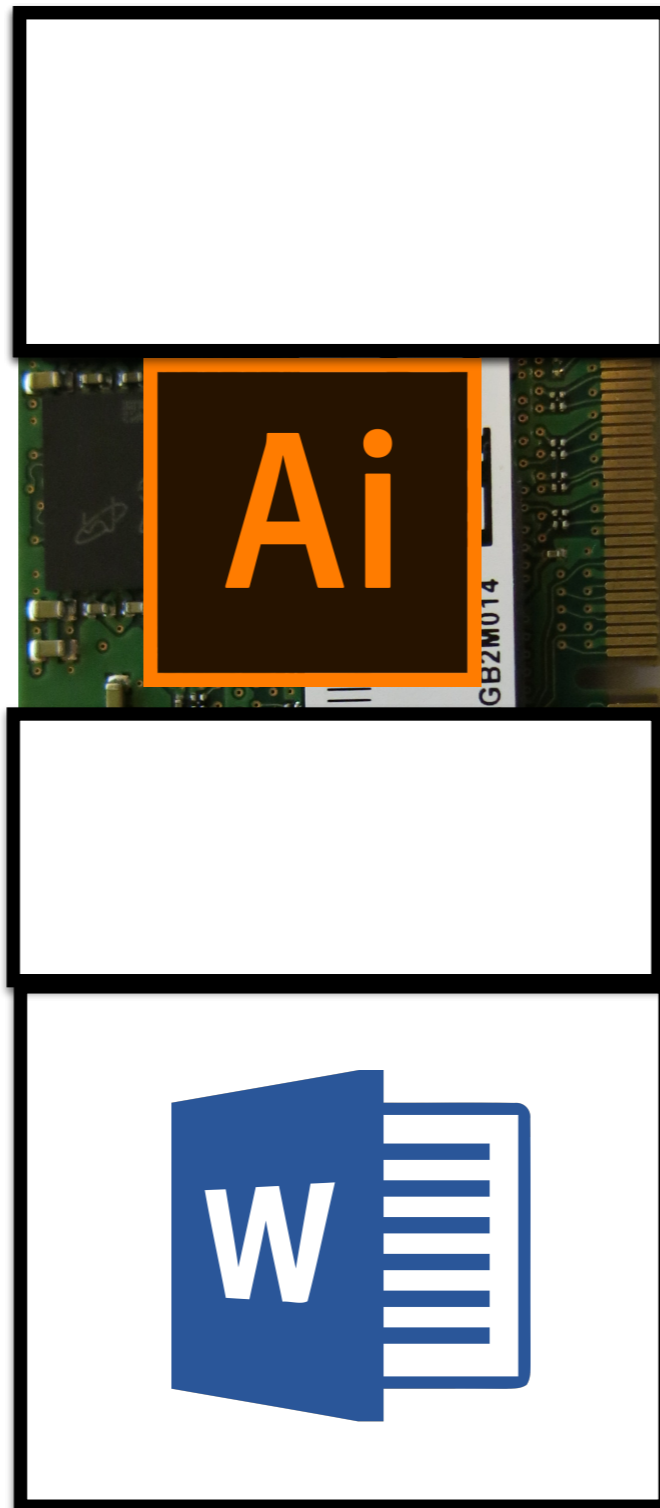
What if process says it wants full memory?





# Direct Physical Memory Multiplexing

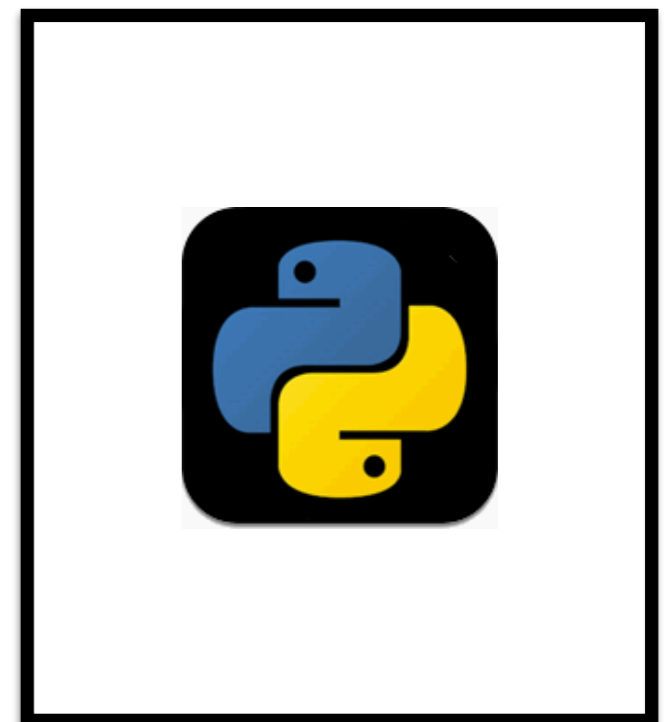
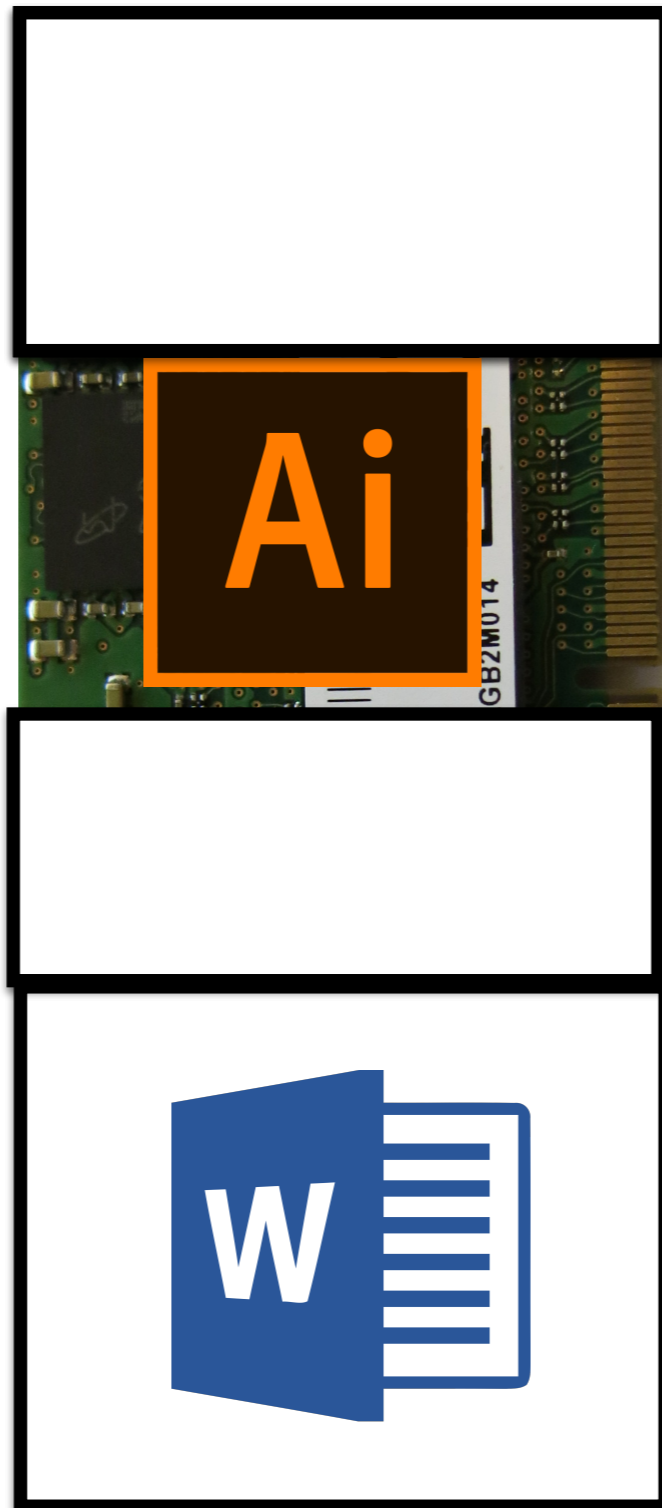
---



# Direct Physical Memory Multiplexing

---

What if process says it wants full memory?

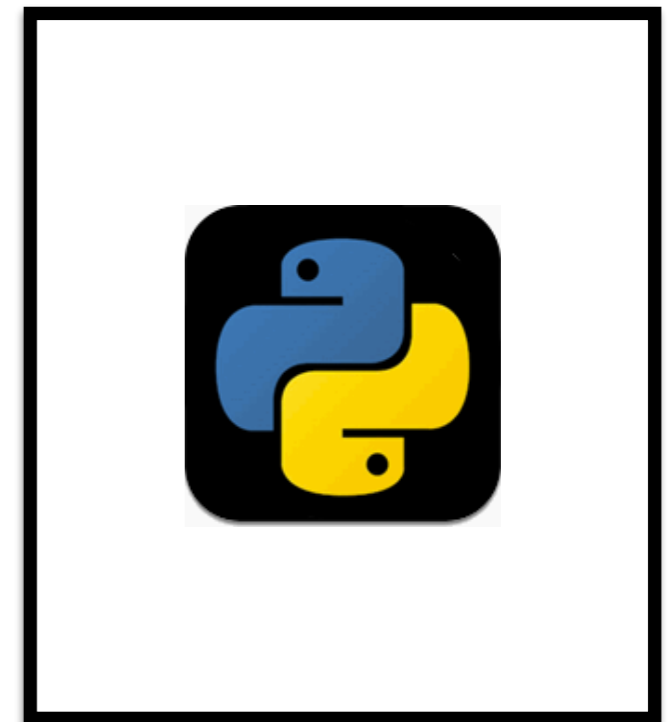
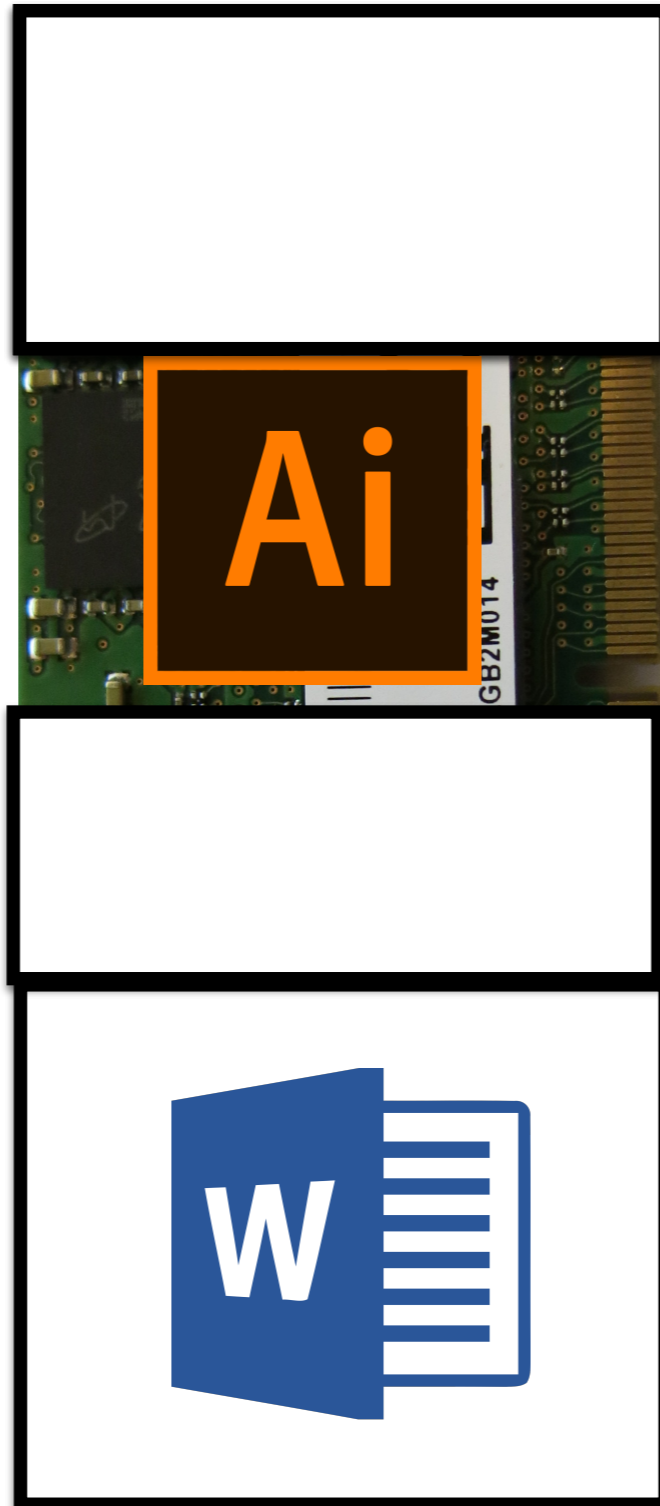


# Direct Physical Memory Multiplexing

---

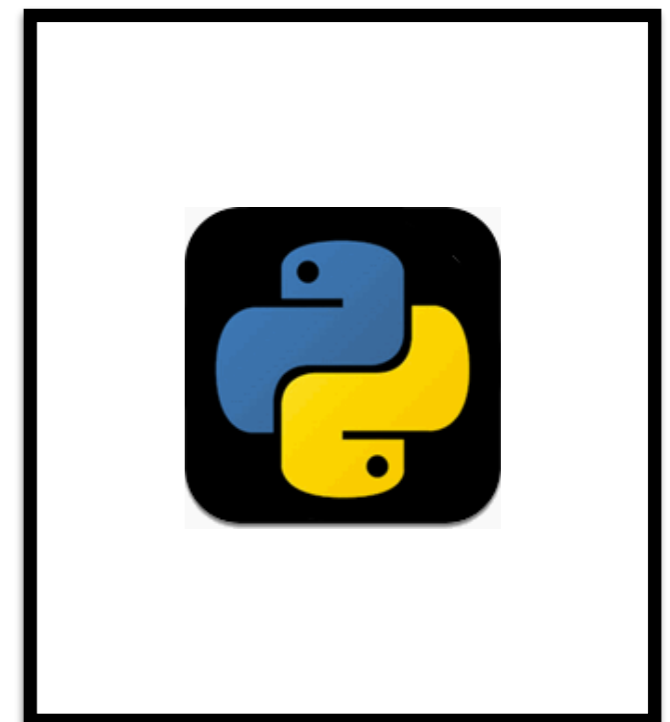
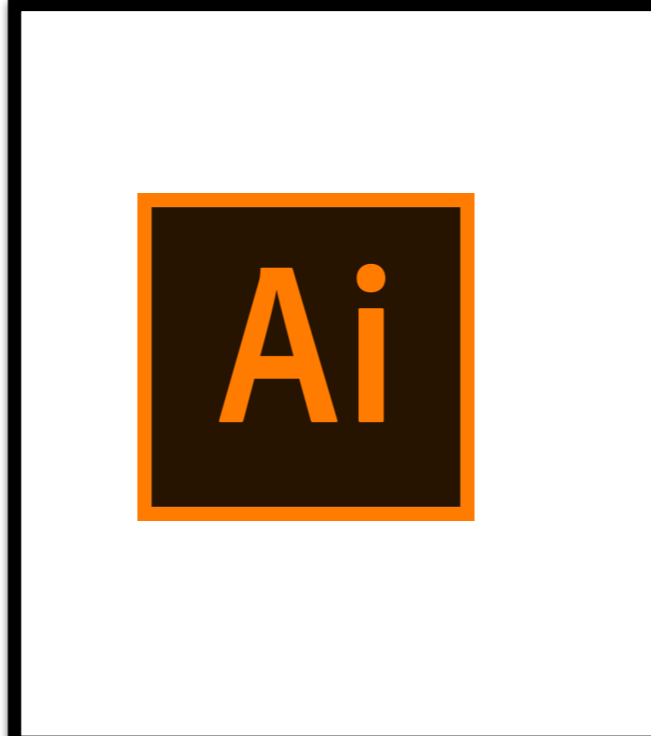
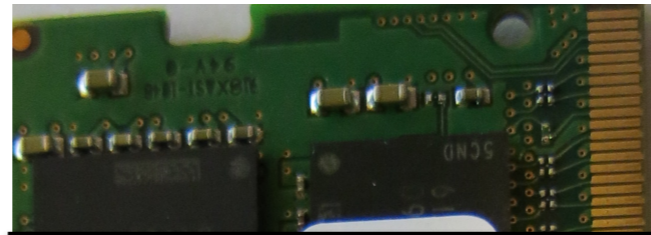
## Internal Fragmentation

What if process says it wants full memory?



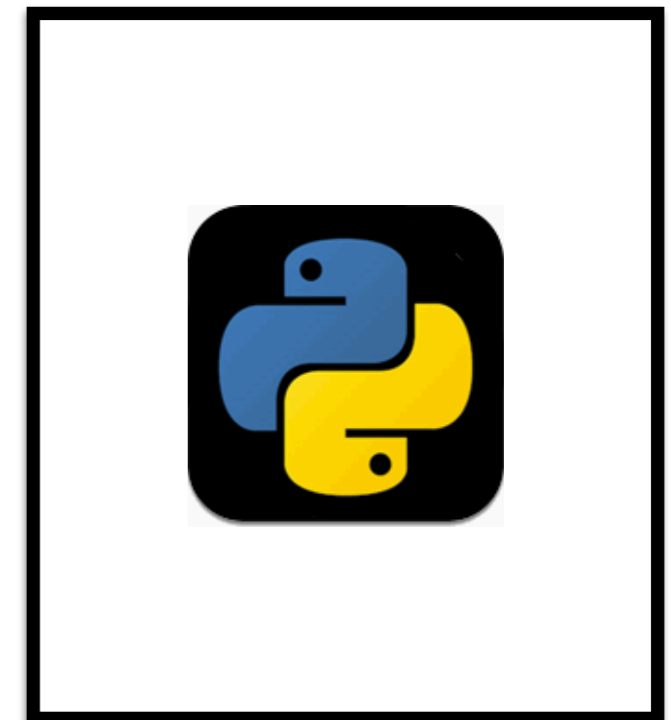
# Direct Physical Memory Multiplexing

---



# Direct Physical Memory Multiplexing

---

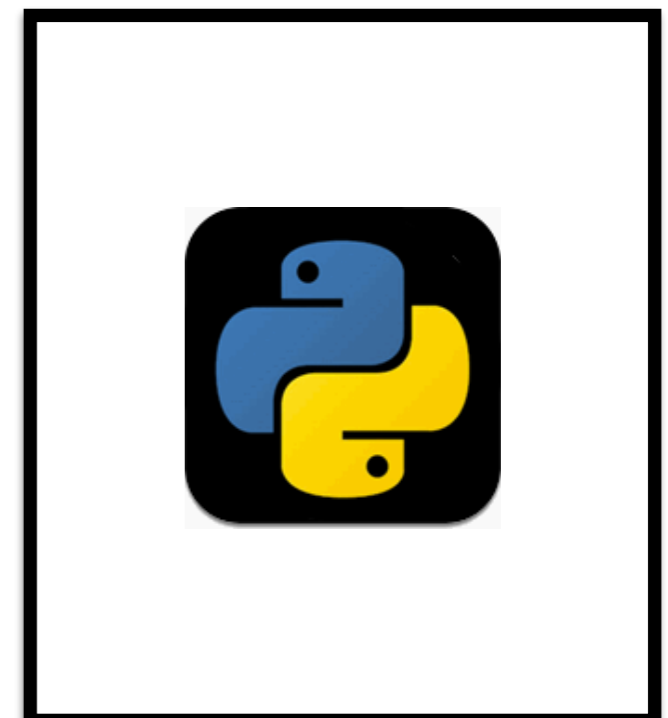
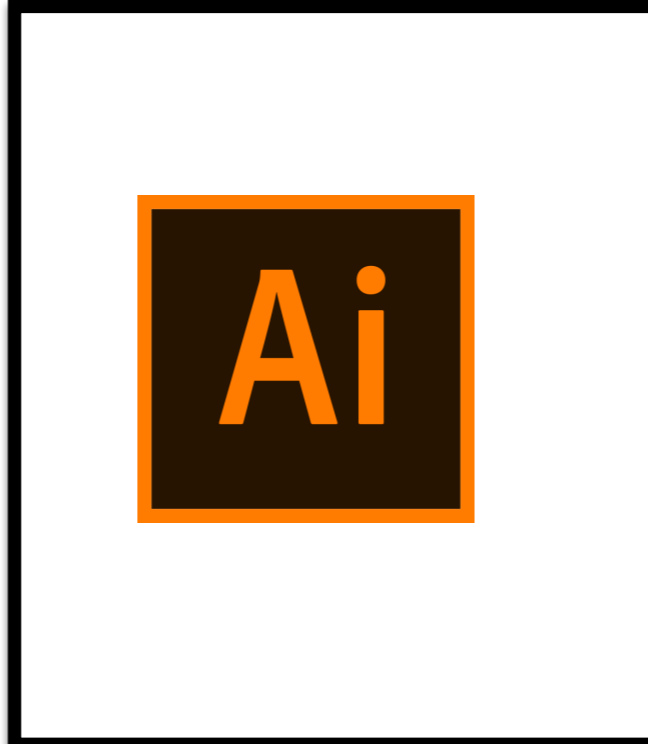


# Direct Physical Memory Multiplexing

---

Can Python run now?

Total memory -  
Memory req for  
Illustrator  $\geq$   
Memory req for  
Python



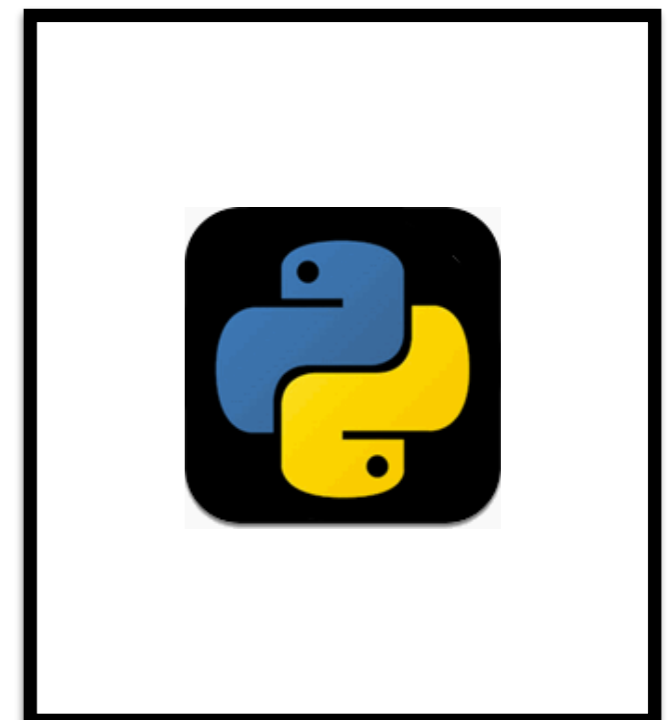
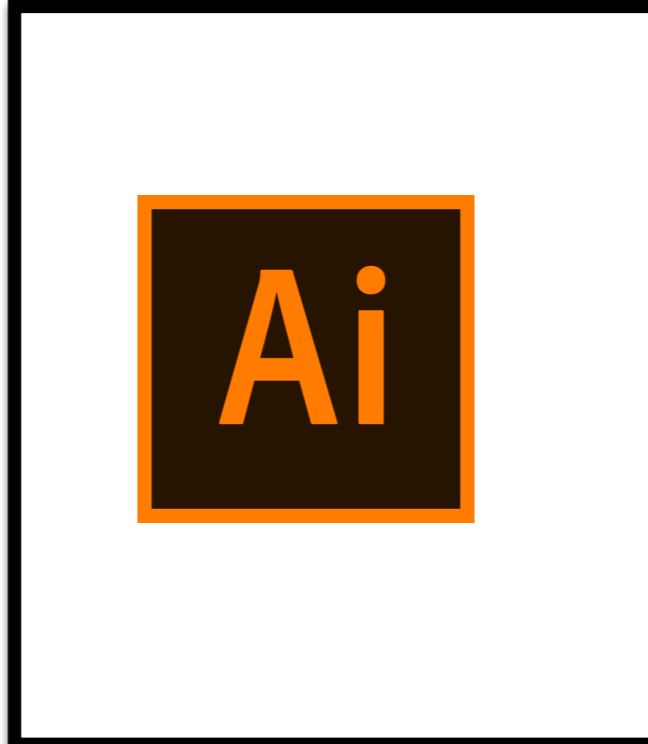
# Direct Physical Memory Multiplexing

---

## External Fragmentation

Can Python run now?

Total memory -  
Memory req for  
Illustrator  $\geq$  =  
Memory req for  
Python



# Defragmentation Memories ...



r/funny

Search Reddit



r/funny

## Posts



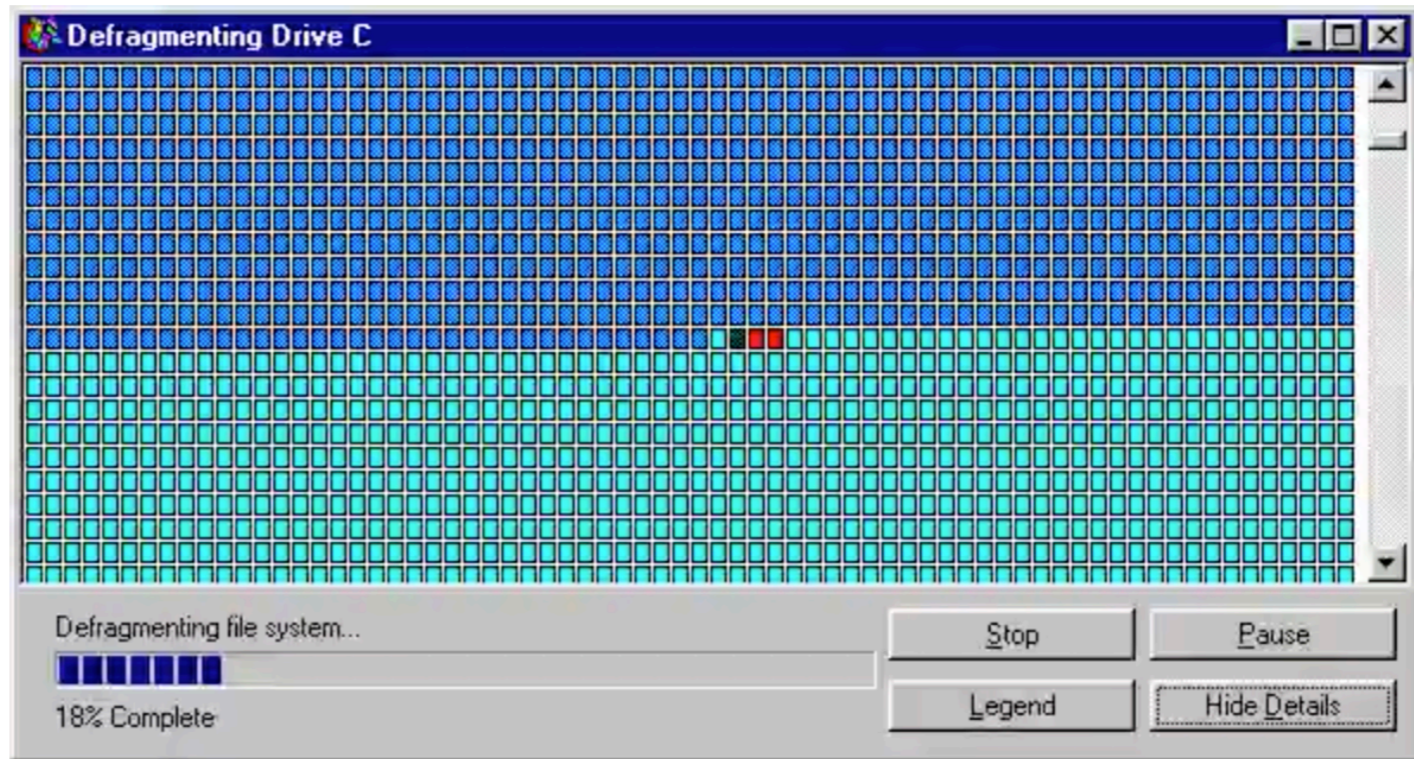
Posted by u/Riconot-so-suave 5 years ago

2.5k



The younger generation will never know relaxing meditation like this.

[i.imgur.com/f73ksX...](https://i.imgur.com/f73ksX...)



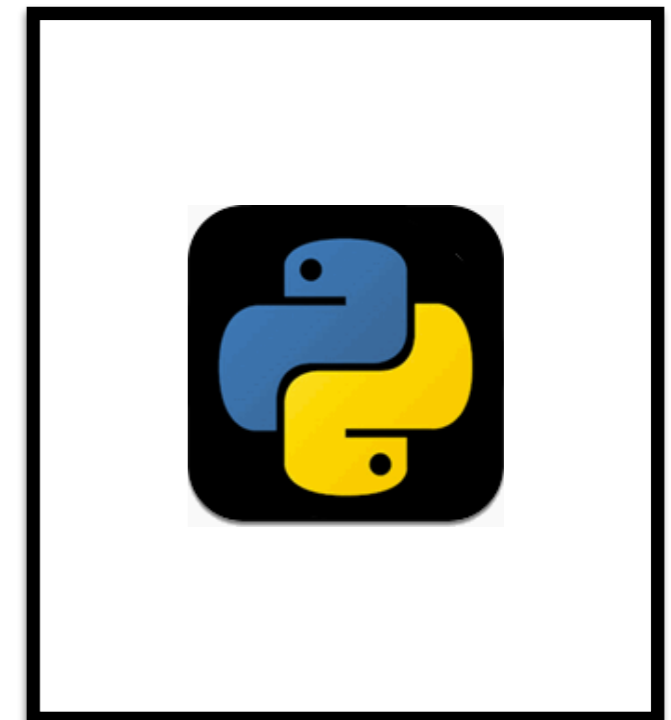
408 Comments Share Save ...



# Direct Physical Memory Multiplexing

---

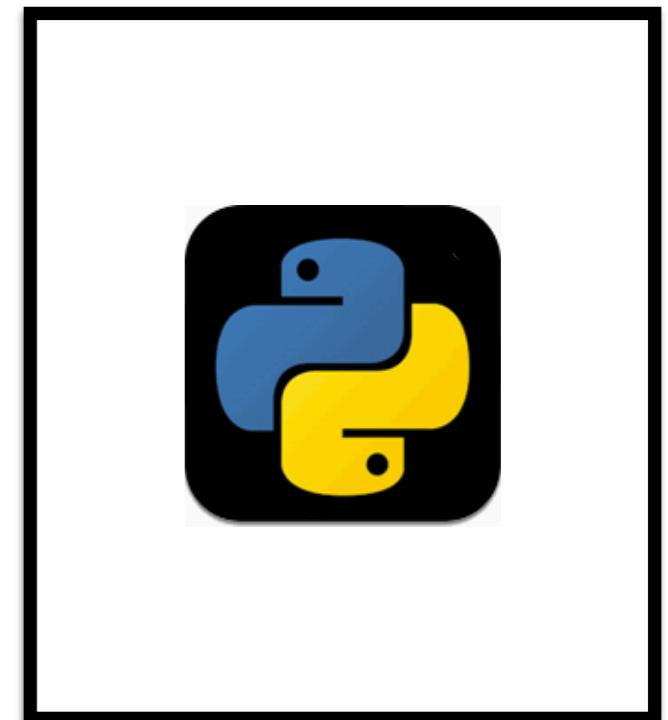
## Defragmentation



# Direct Physical Memory Multiplexing

---

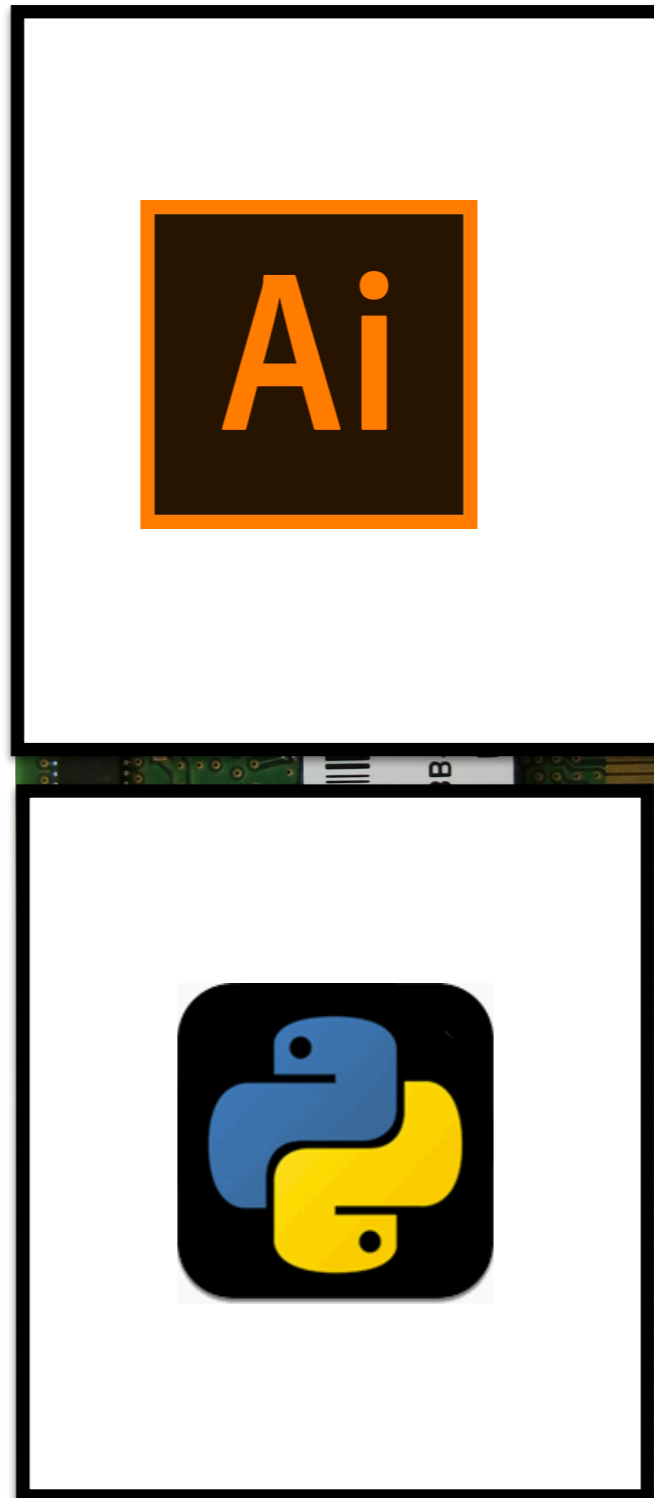
## Defragmentation



# Direct Physical Memory Multiplexing

---

Defragmentation



# Goals of OS for Memory Virtualisation/ Management

---

1. Transparency
  1. Physical memory is invisible to user program
  2. Program thinks it has own private large (contiguous + plentiful) memory
2. Efficiency
  1. Not taking very long
  2. Not taking too much space
3. Protection/Isolation
  1. Protect processes from each other

# Memory Interface

---

1. Load (address)
2. Store (address, value)

# Physical v/s Virtual Memory

---

# Physical v/s Virtual Memory

---

1. Abstraction : Break the connection between physical memory and an address

# Physical v/s Virtual Memory

---

1. Abstraction : Break the connection between physical memory and an address
2. Data accessed using memory interface is virtual address



# Physical v/s Virtual Memory

---

1. Abstraction : Break the connection between physical memory and an address
2. Data accessed using memory interface is virtual address
  1. Physical address points to memory

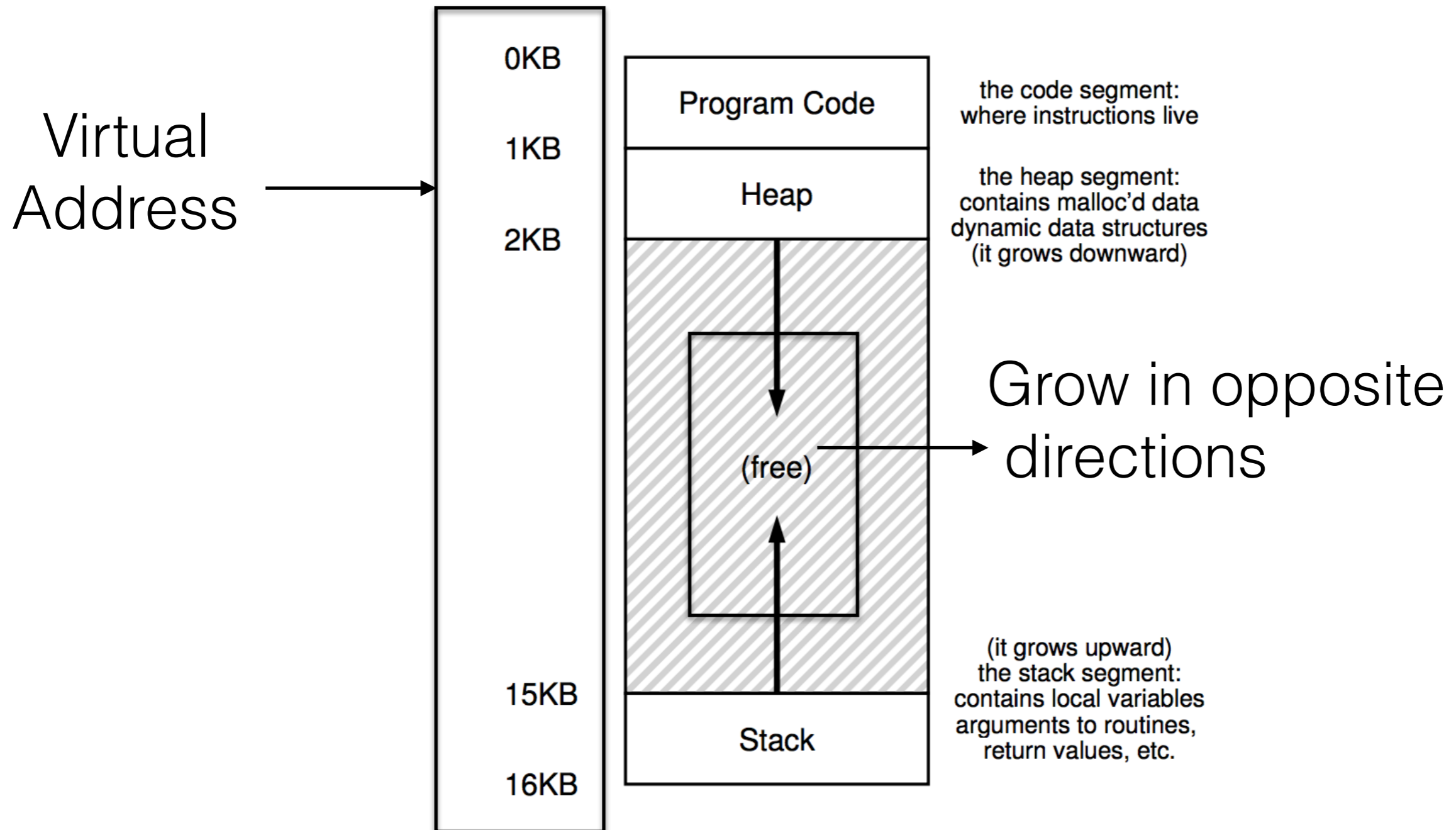
# Physical v/s Virtual Memory

---

1. Abstraction : Break the connection between physical memory and an address
2. Data accessed using memory interface is virtual address
  1. Physical address points to memory
  2. Virtual address points to something *acting like memory*

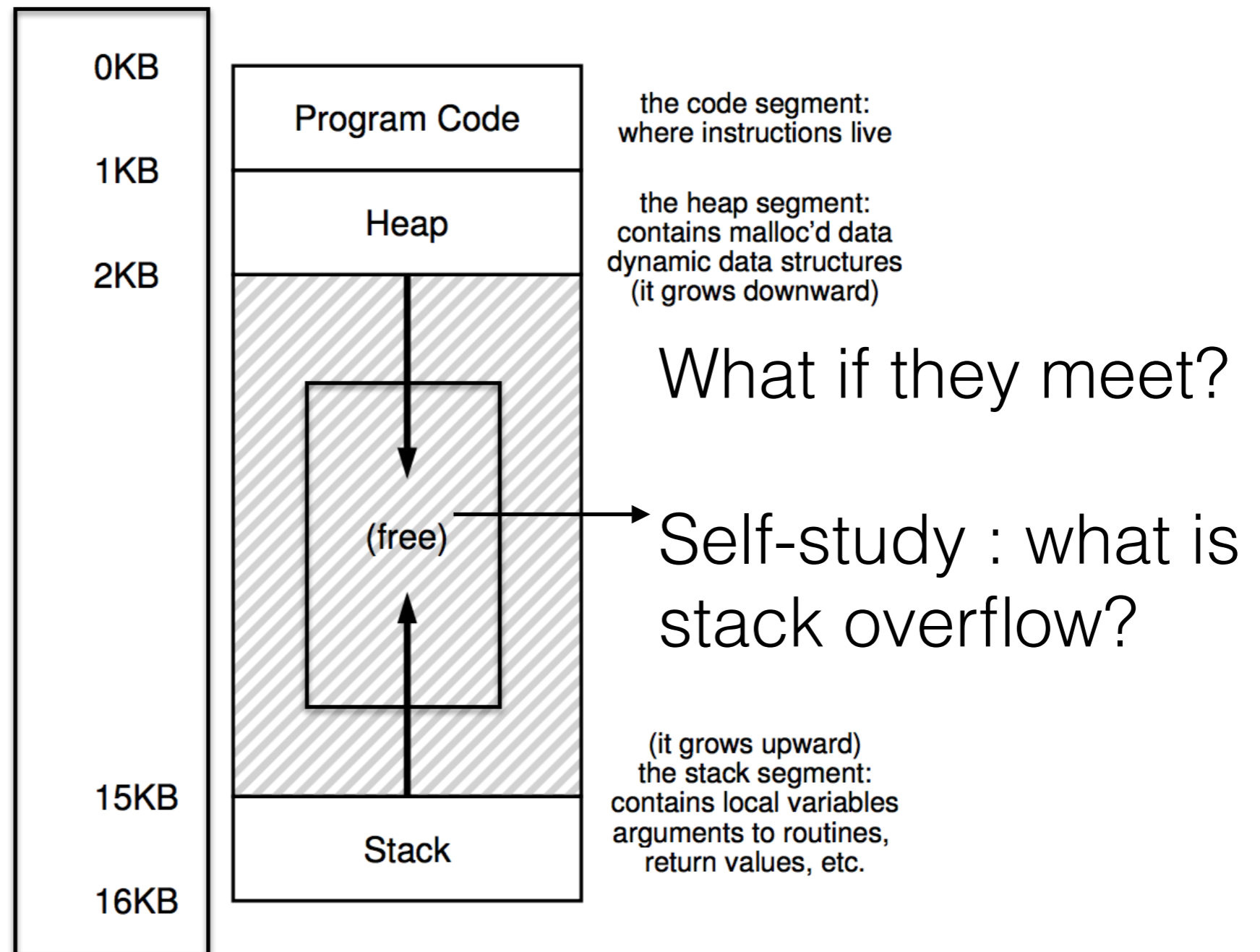
# Address Space

---



# Stack Overflow?!

---



# Stack Overflow?!

---

```
def fac(n):  
    if n==1 or n==0:  
        return 1  
    else:  
        return n*fac(n-1)
```

# Exec Revisited

---

`fork_same_address.c`

# Example


---

```
void func() {  
    int x = 3000; //  
    x = x + 3;    //  
    ...  
}
```

# Example

---

```
void func() {  
    int x = 3000; // Compiler  
    x = x + 3;    //  
    ...
```





# Example

---

```
void func() {  
    int x = 3000; //  
    x = x + 3;   //  
    ...  
                Compiler  
                ───────────▶  
128: movl 0x0(%ebx), %eax  
132: addl $0x03, %eax  
135: movl %eax, 0x0(%ebx)
```

# Example

---

Compiler →  
128: movl 0x0(%ebx), %eax  
132: addl \$0x03, %eax  
135: movl %eax, 0x0(%ebx)

# Example

---

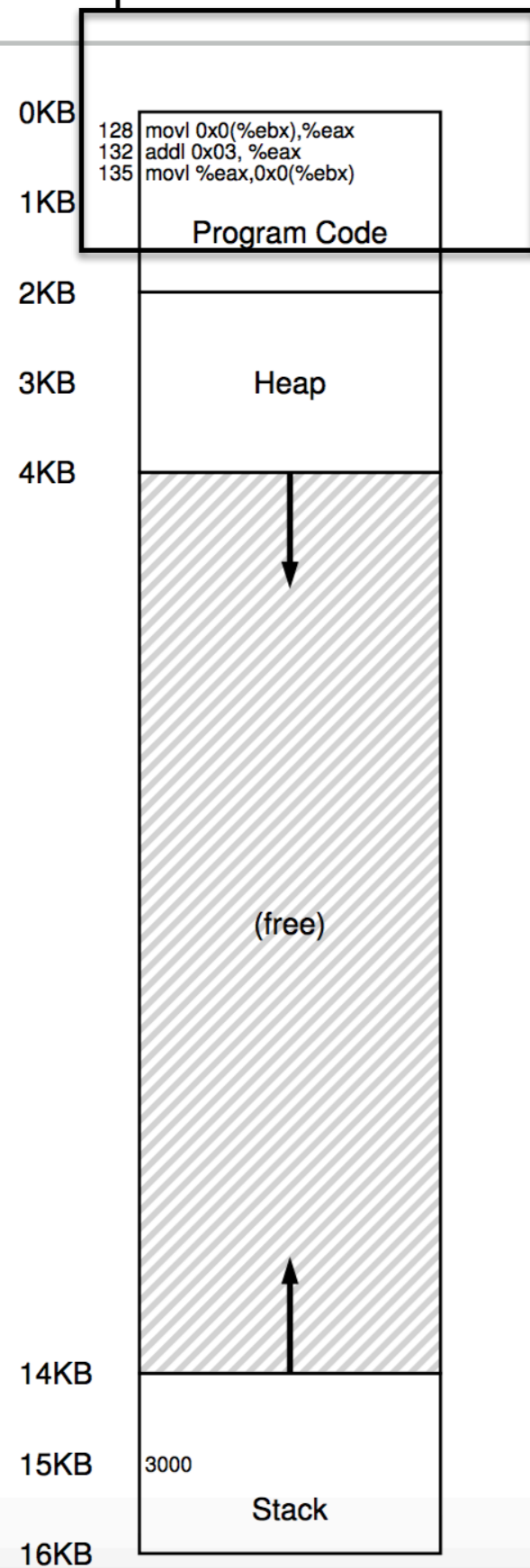
```
128: movl 0x0(%ebx), %eax
132: addl $0x03, %eax
135: movl %eax, 0x0(%ebx)
```

# Example



```
128: movl 0x0(%ebx), %eax
132: addl $0x03, %eax
135: movl %eax, 0x0(%ebx)
```

# Example



```
128: movl 0x0(%ebx), %eax
132: addl $0x03, %eax
135: movl %eax, 0x0(%ebx)
```

# Example



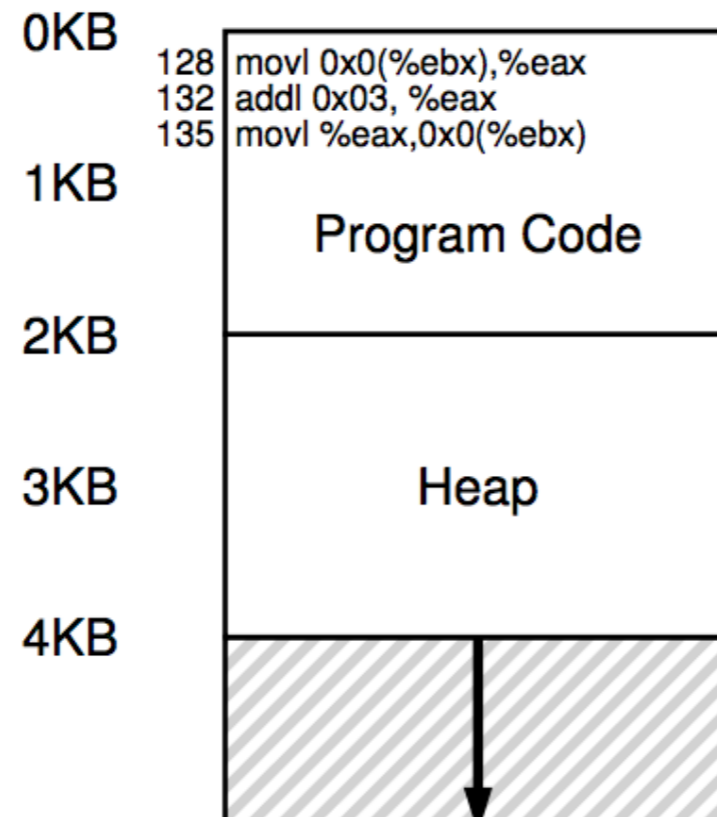
```
128: movl 0x0(%ebx), %eax
132: addl $0x03, %eax
135: movl %eax, 0x0(%ebx)
```

# Example



# Example

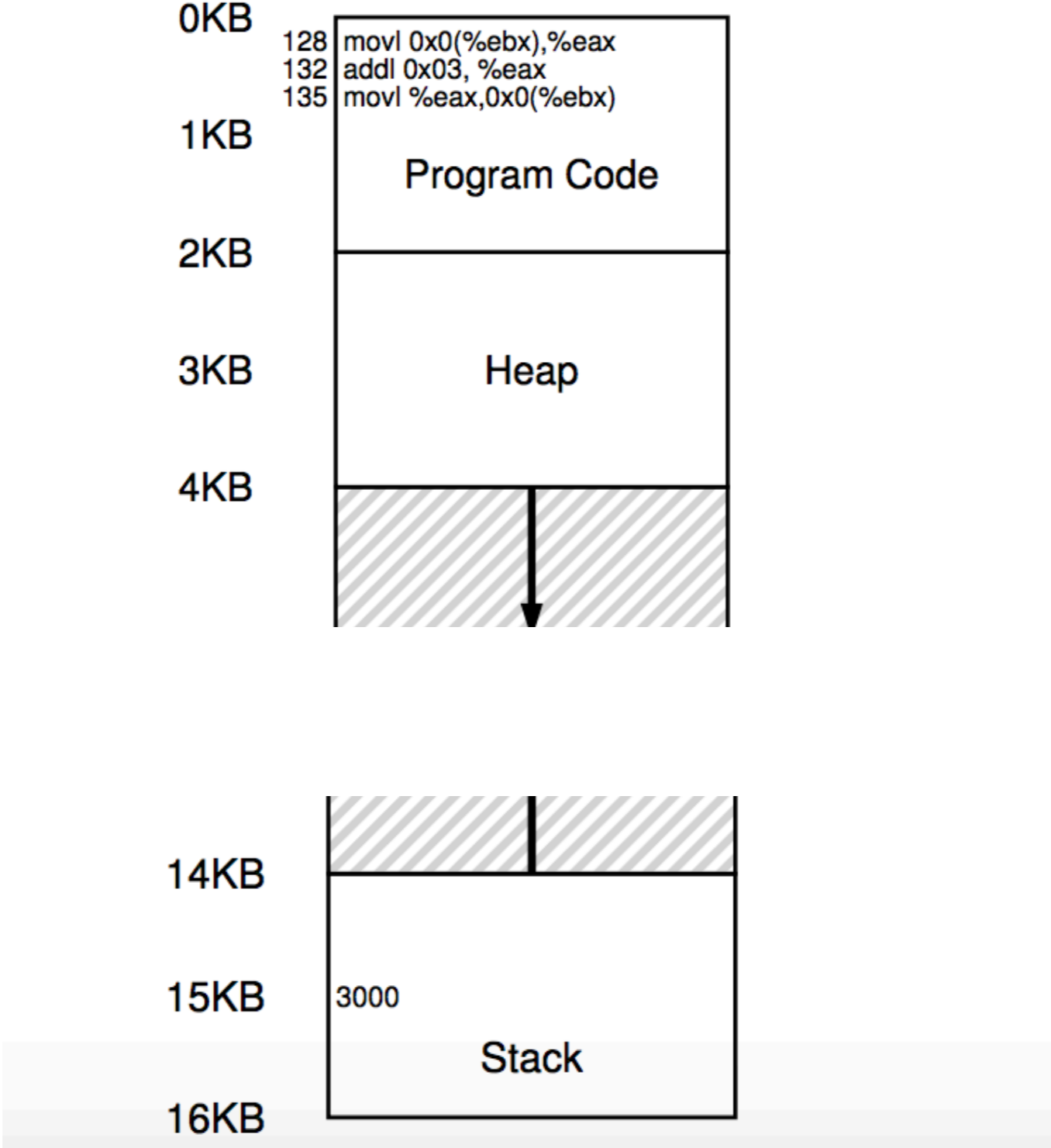
---



Fetch

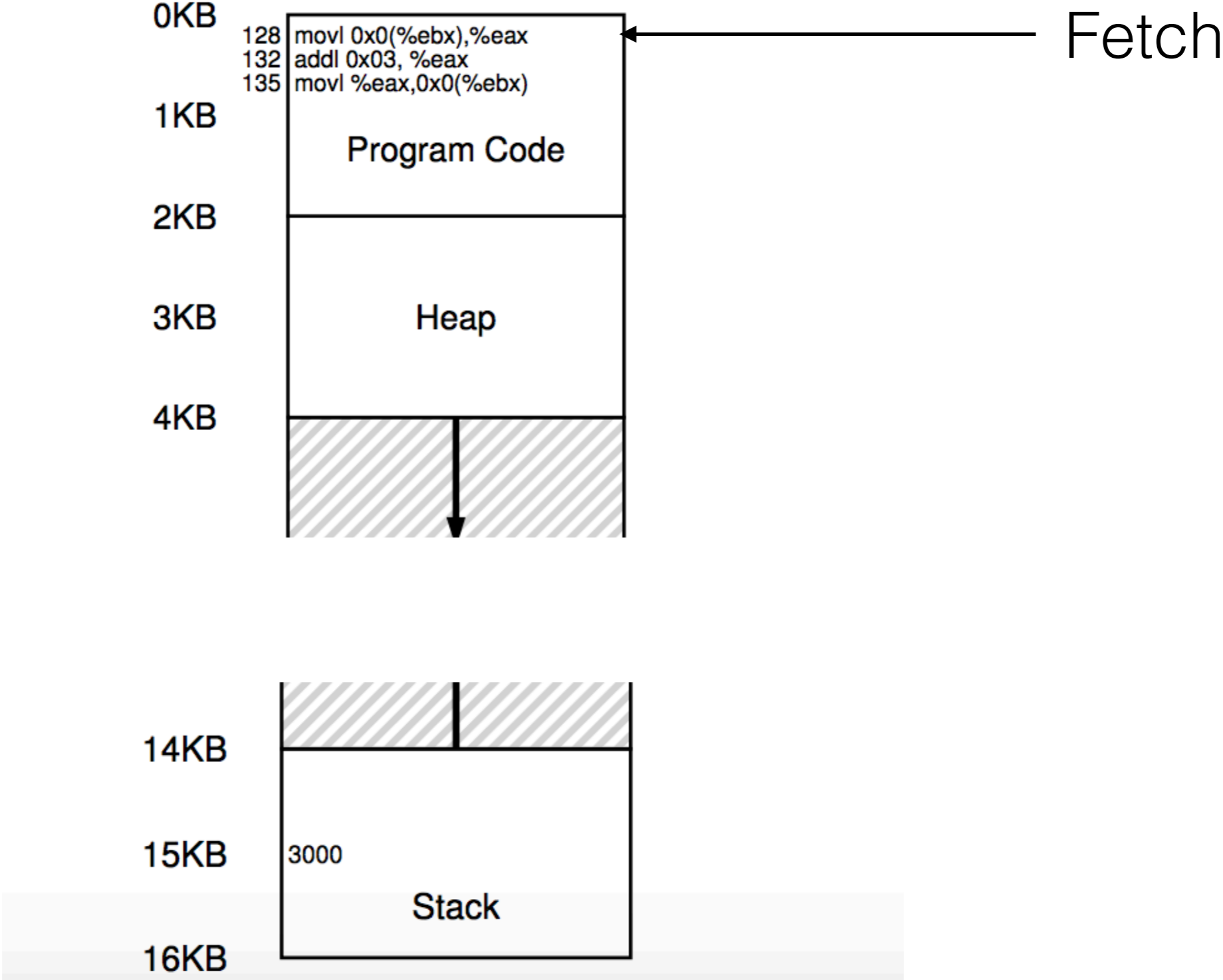


# Example

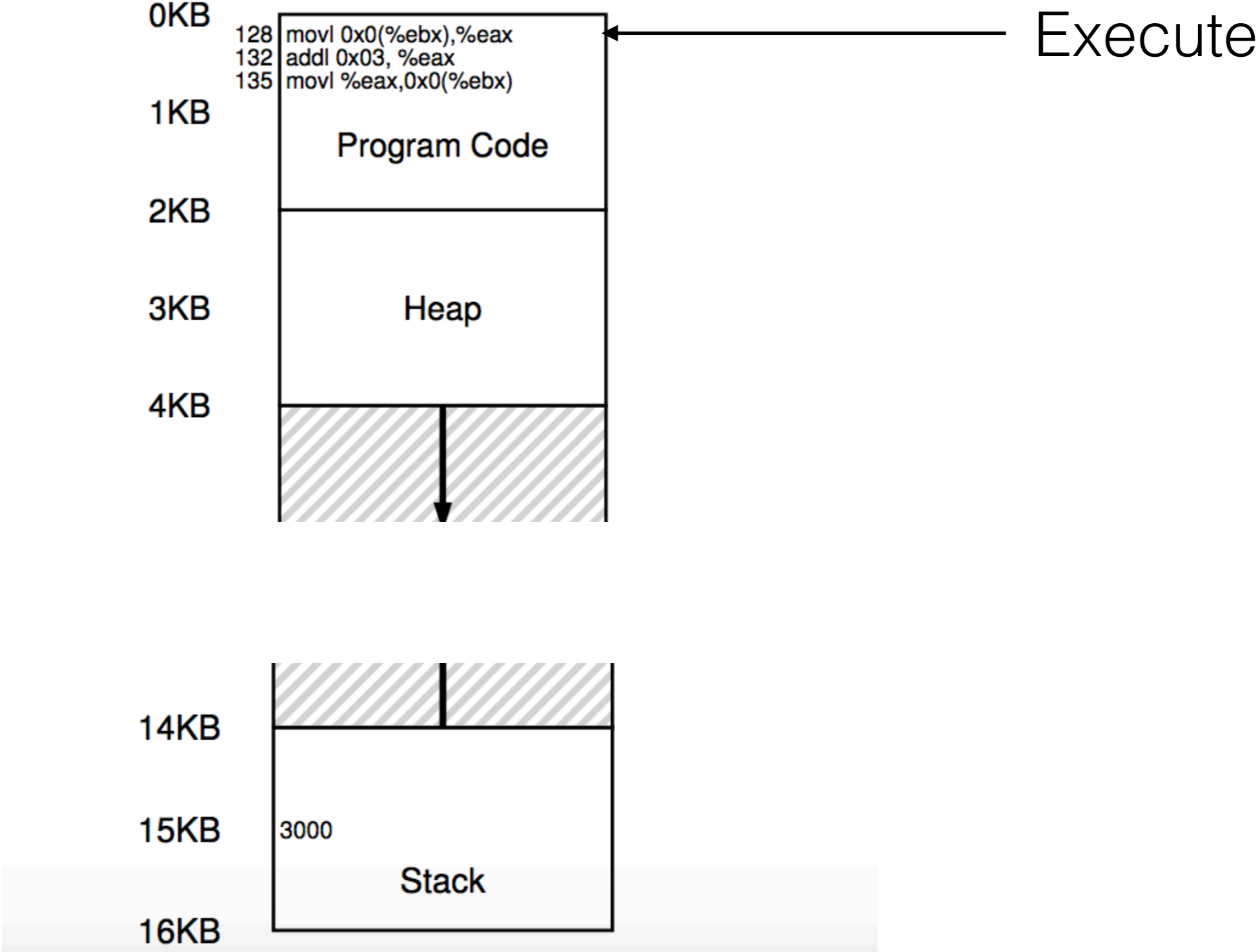


Fetch

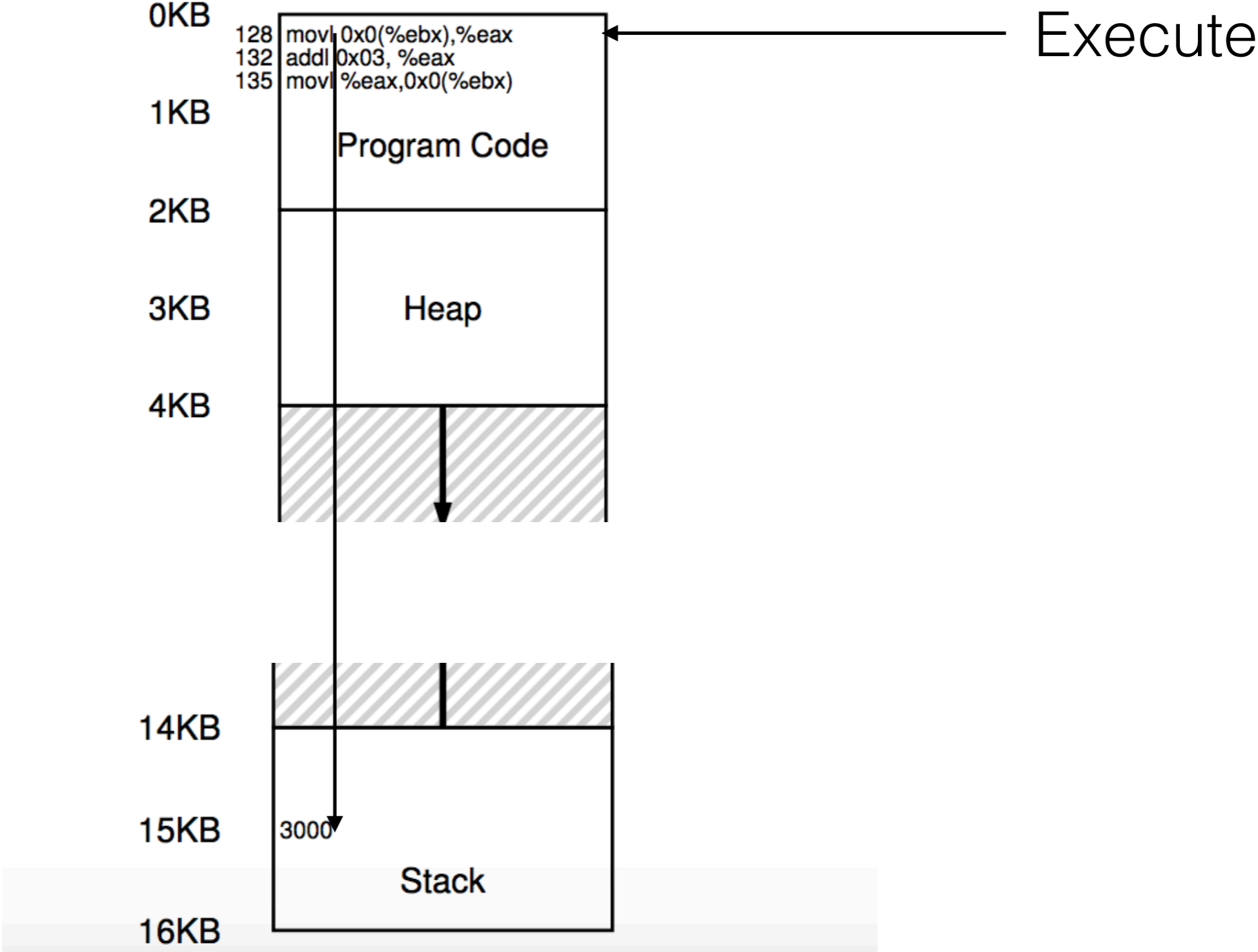
# Example



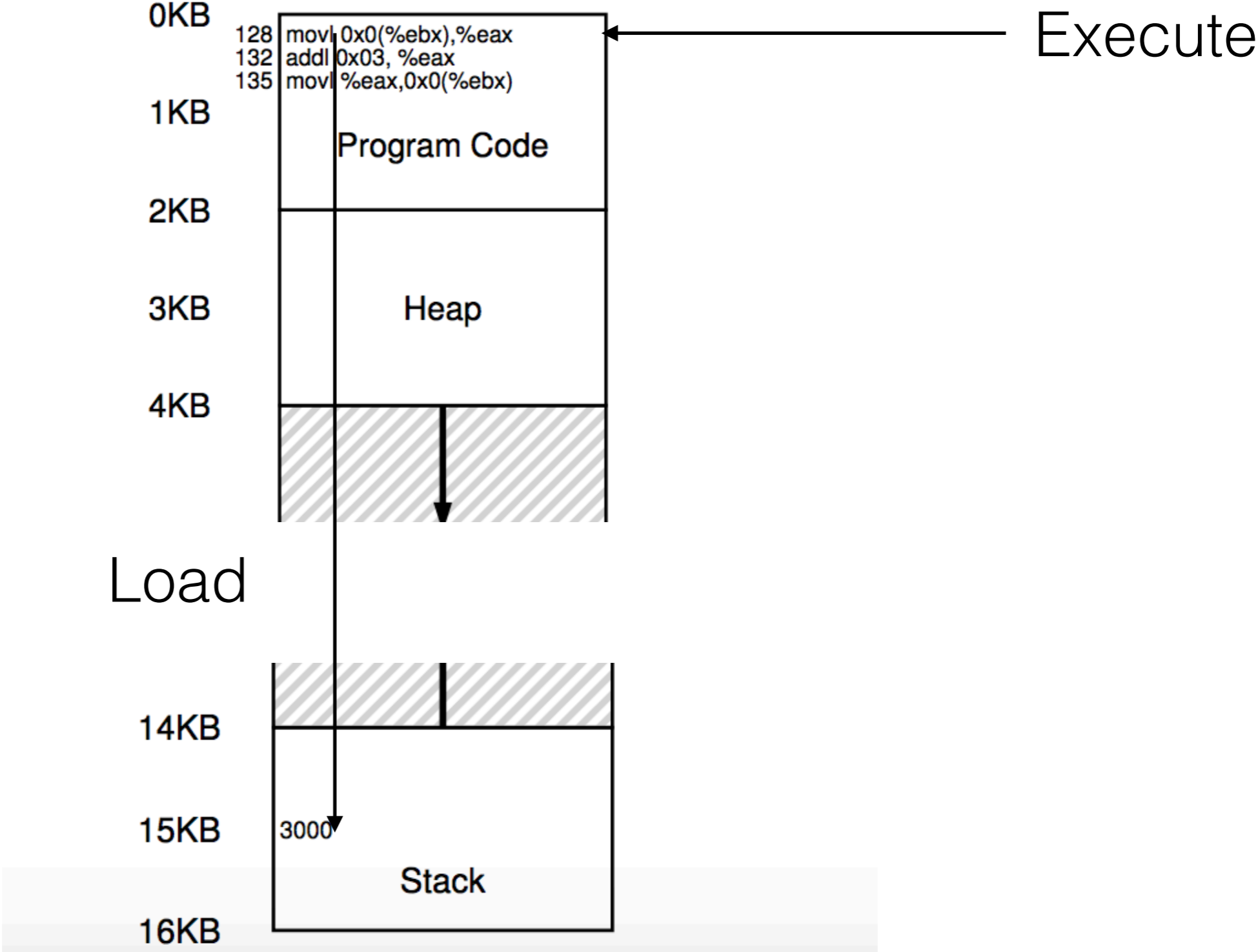
# Example



# Example

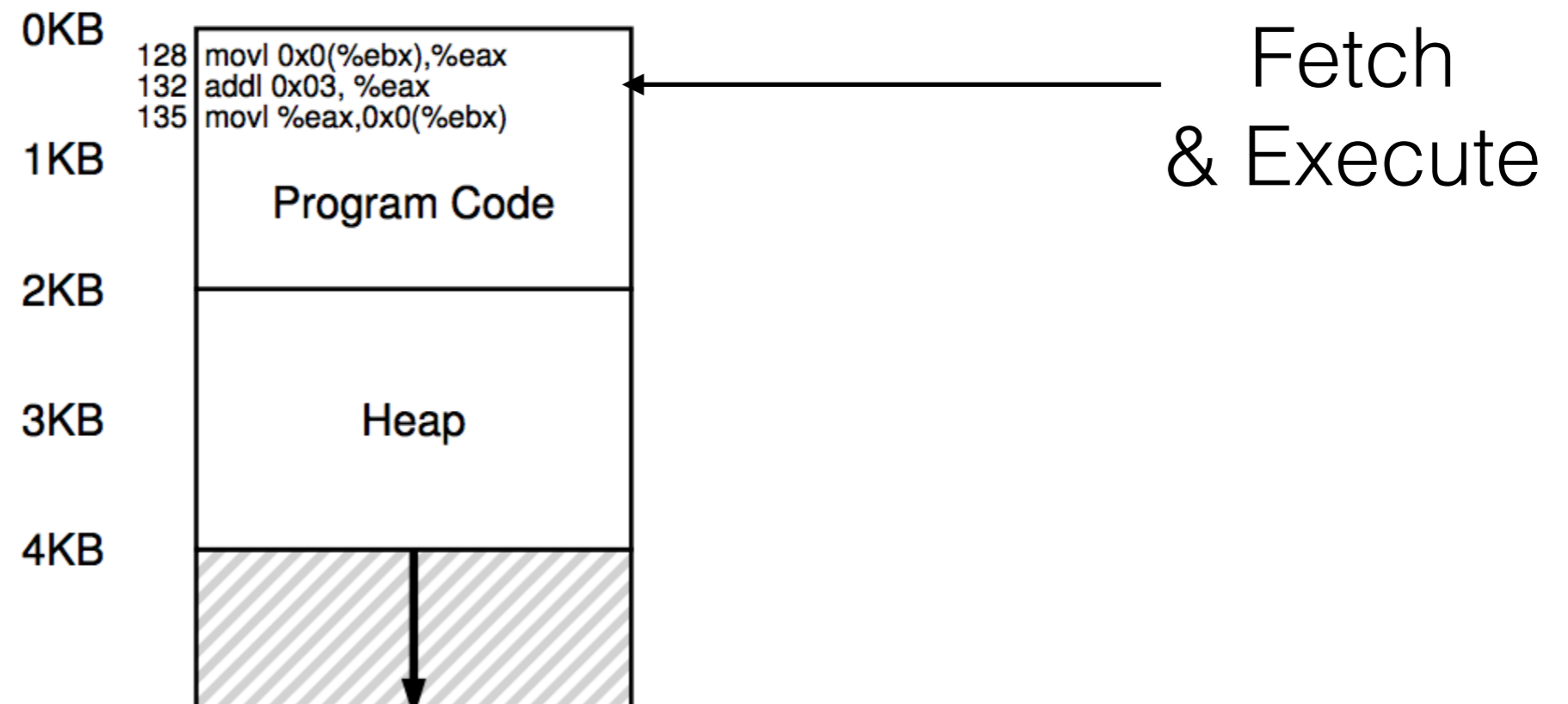


# Example

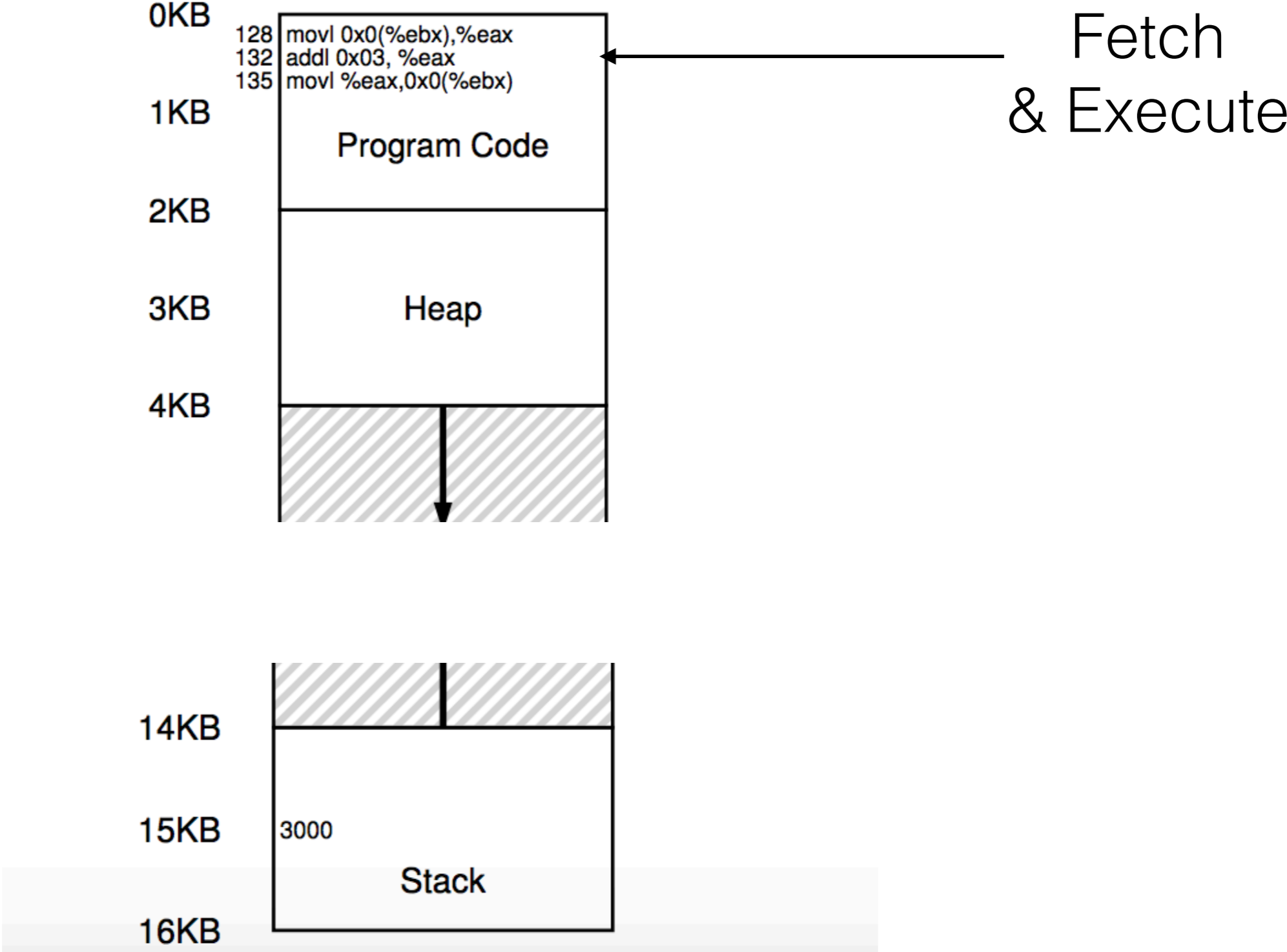


# Example

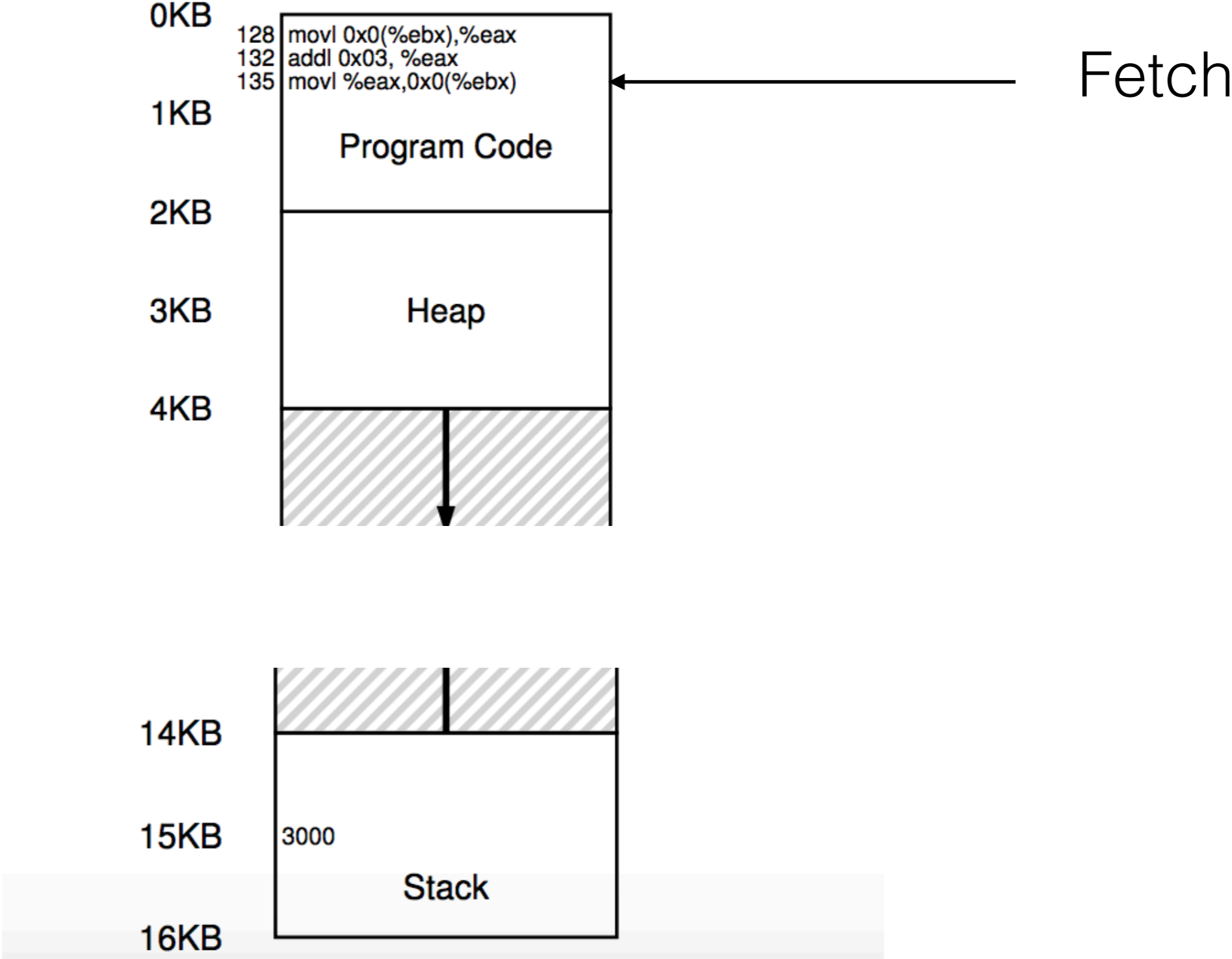
---



# Example

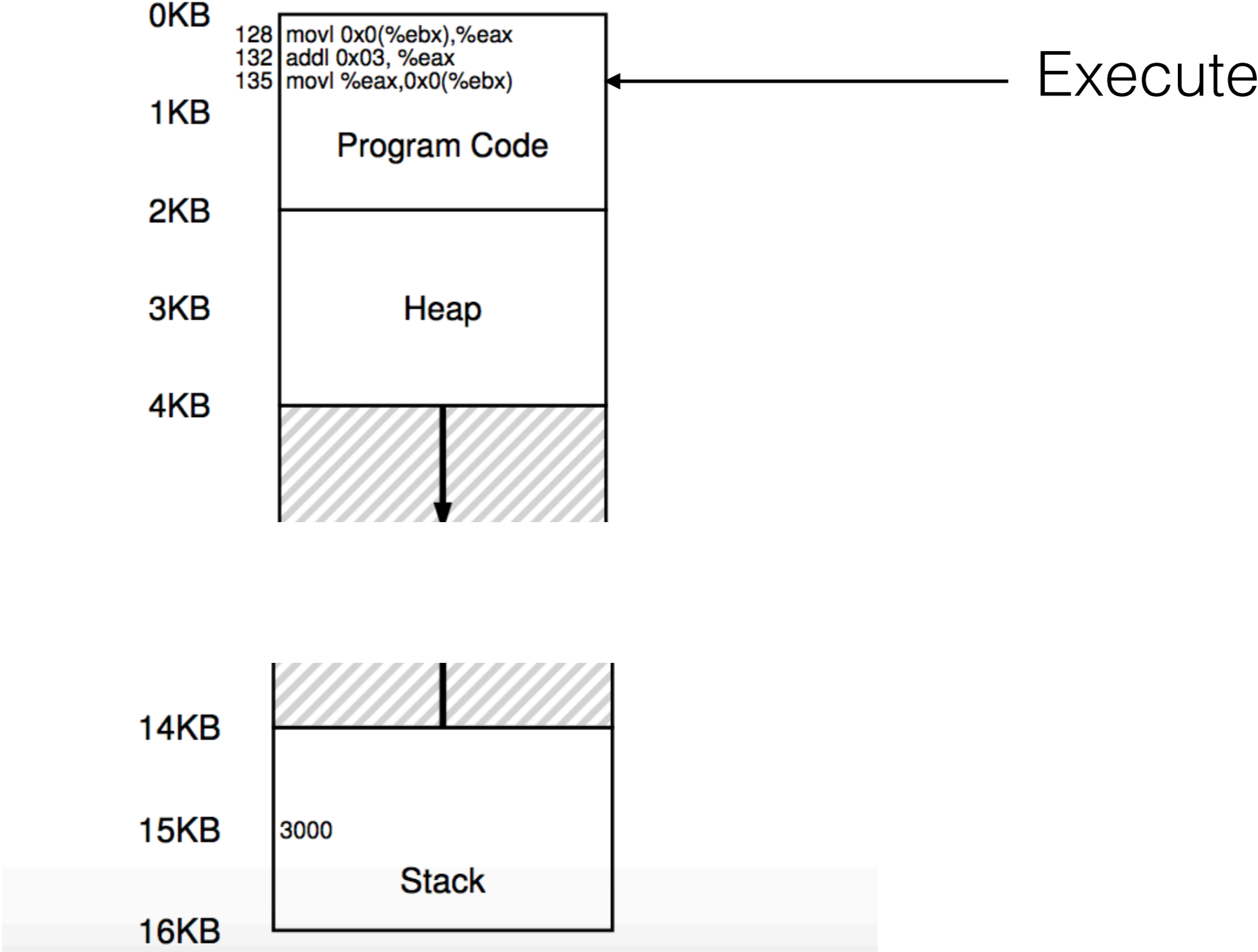


# Example

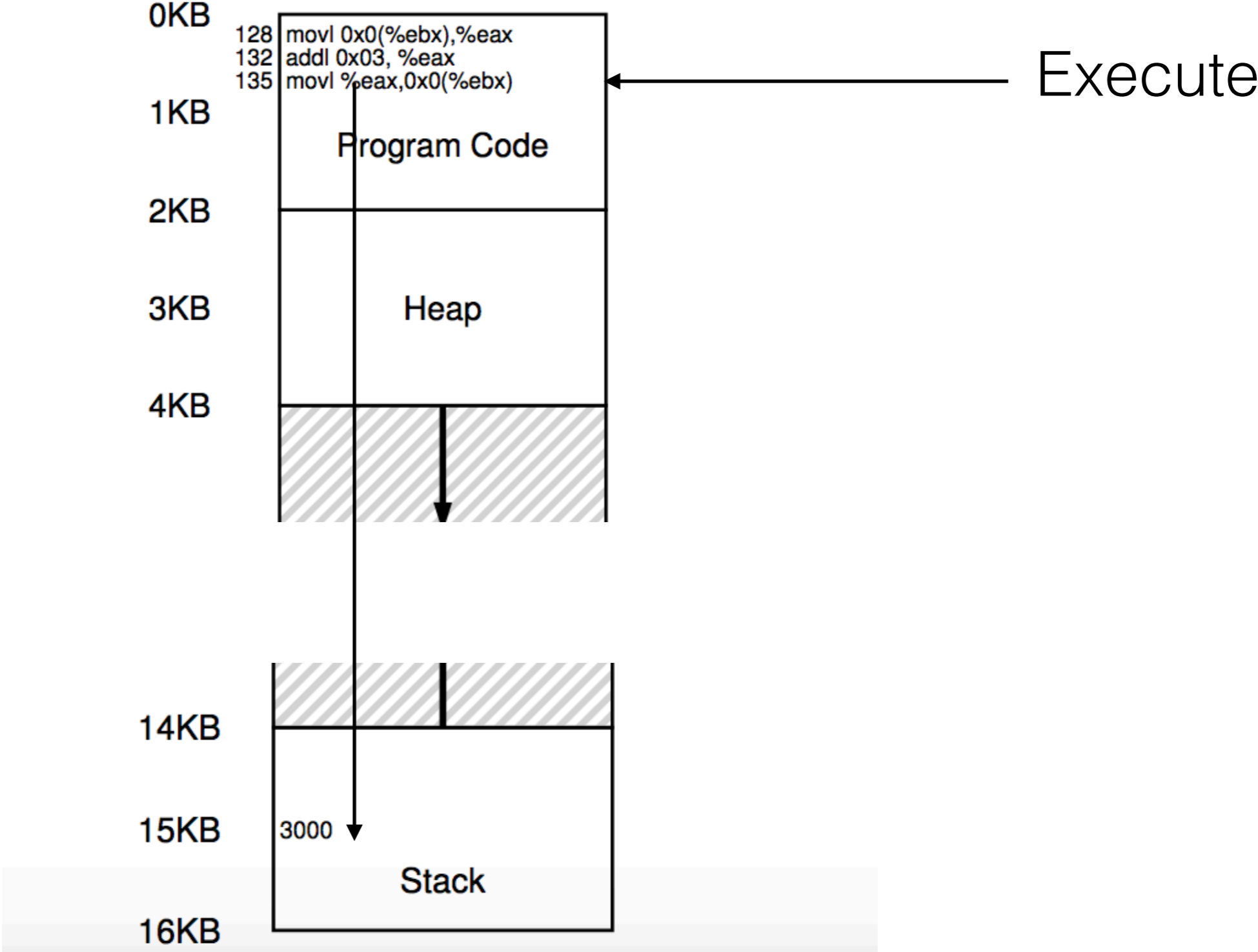




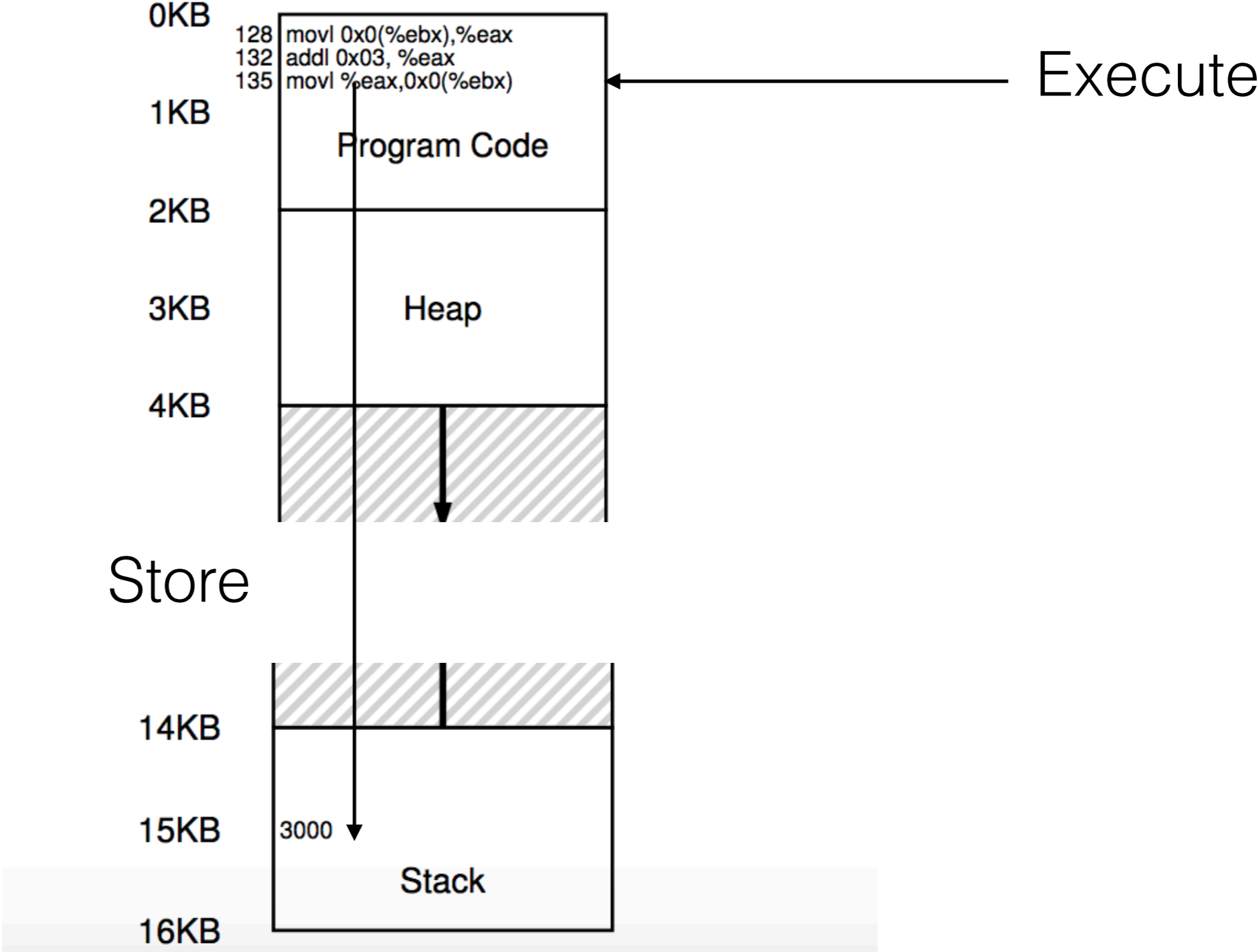
# Example



# Example

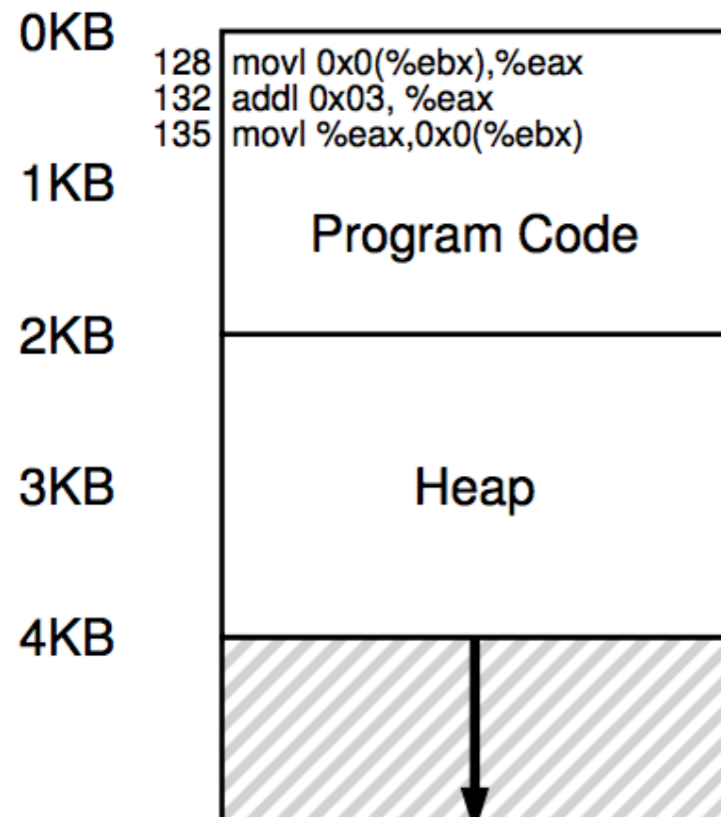


# Example

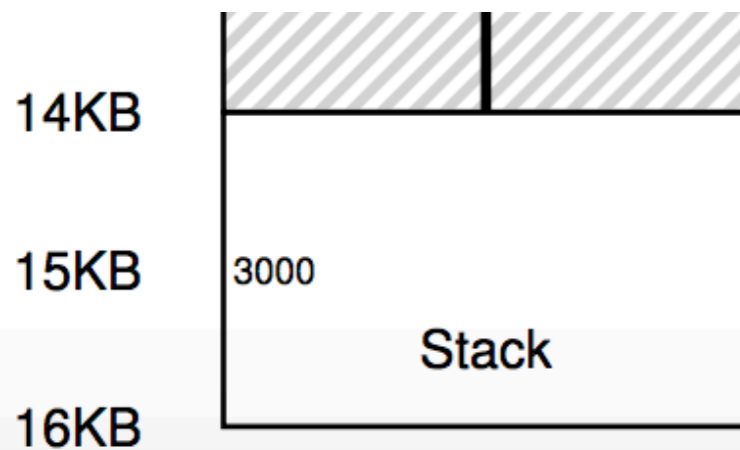


# Pop Quiz

---

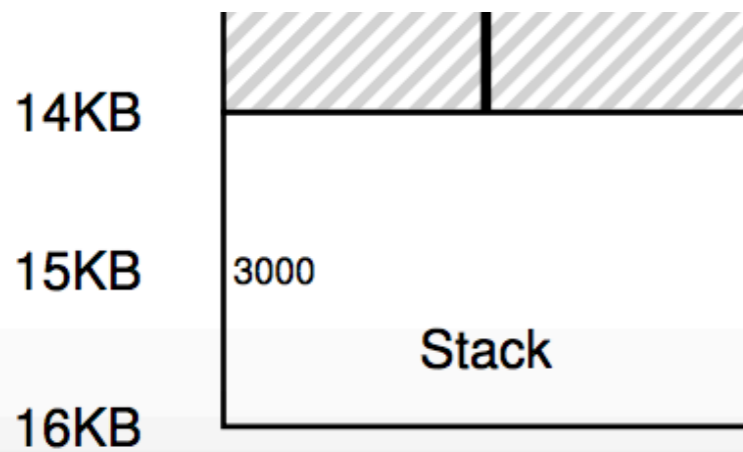
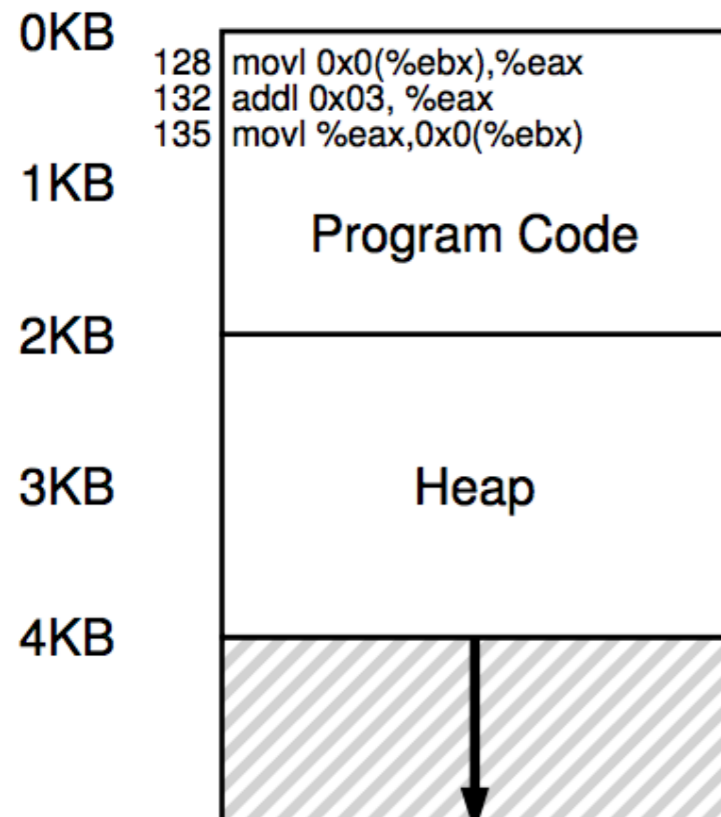


Do all process start and end from 0 KB and 16 KB?



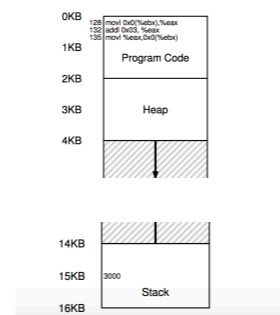
# Relocation

---

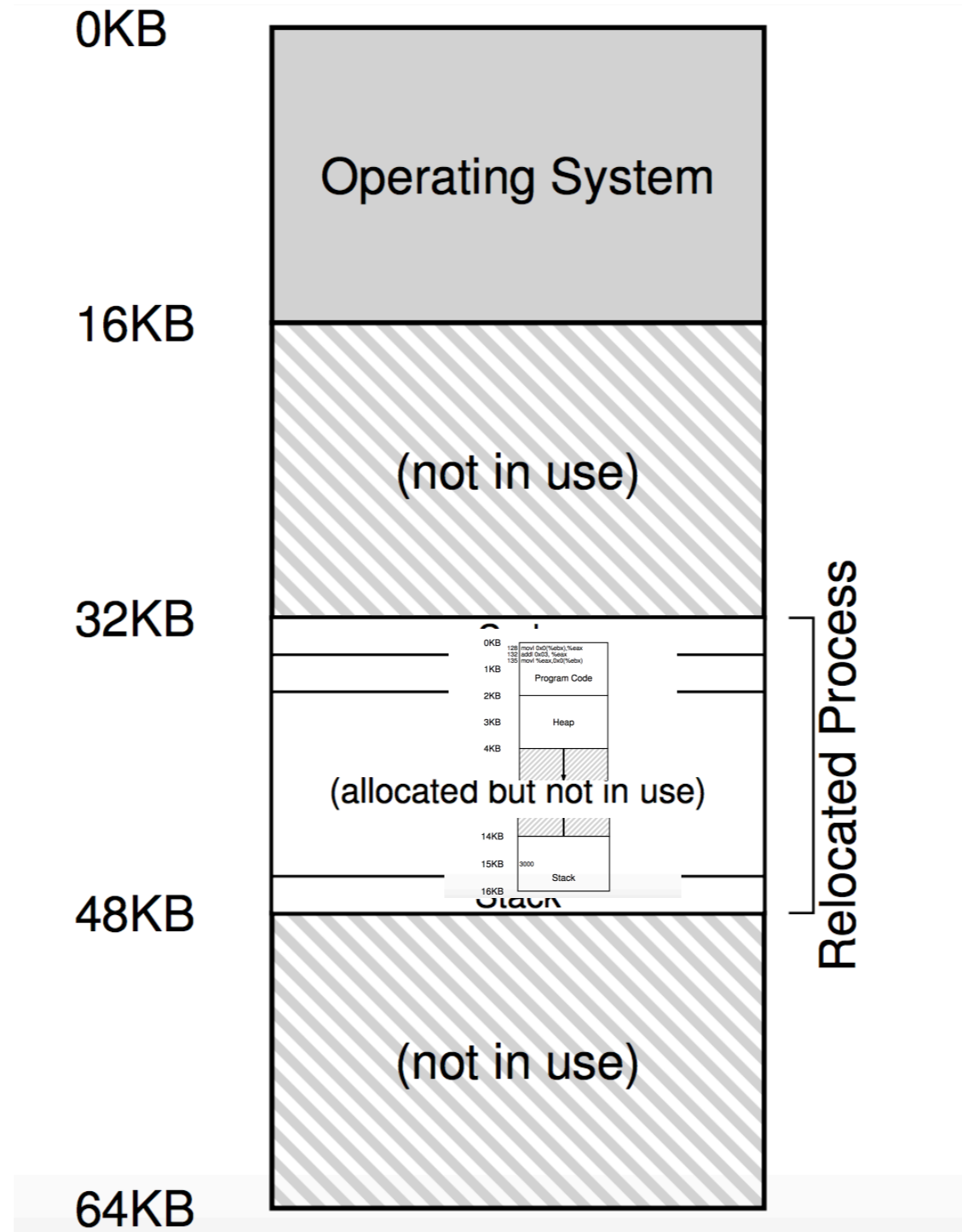


# Relocation

---



# Relocation



# General Address Translation

---

Kernel

CPU

MMU

Physical Memory





# General Address Translation

---

Kernel

CPU

MMU

Physical Memory



# General Address Translation

---

Kernel

CPU

MMU

Physical Memory

Virtual Address



# General Address Translation

---

Kernel

CPU

MMU

Physical Memory

Virtual Address  
0x10102030



# General Address Translation

---

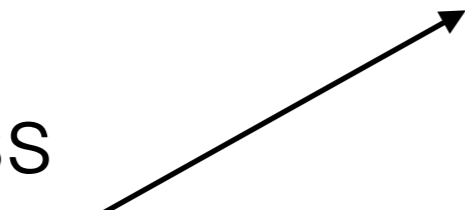
Kernel

CPU

MMU

Physical Memory

Virtual Address  
0x10102030



# General Address Translation

---



Physical Memory



# General Address Translation

---

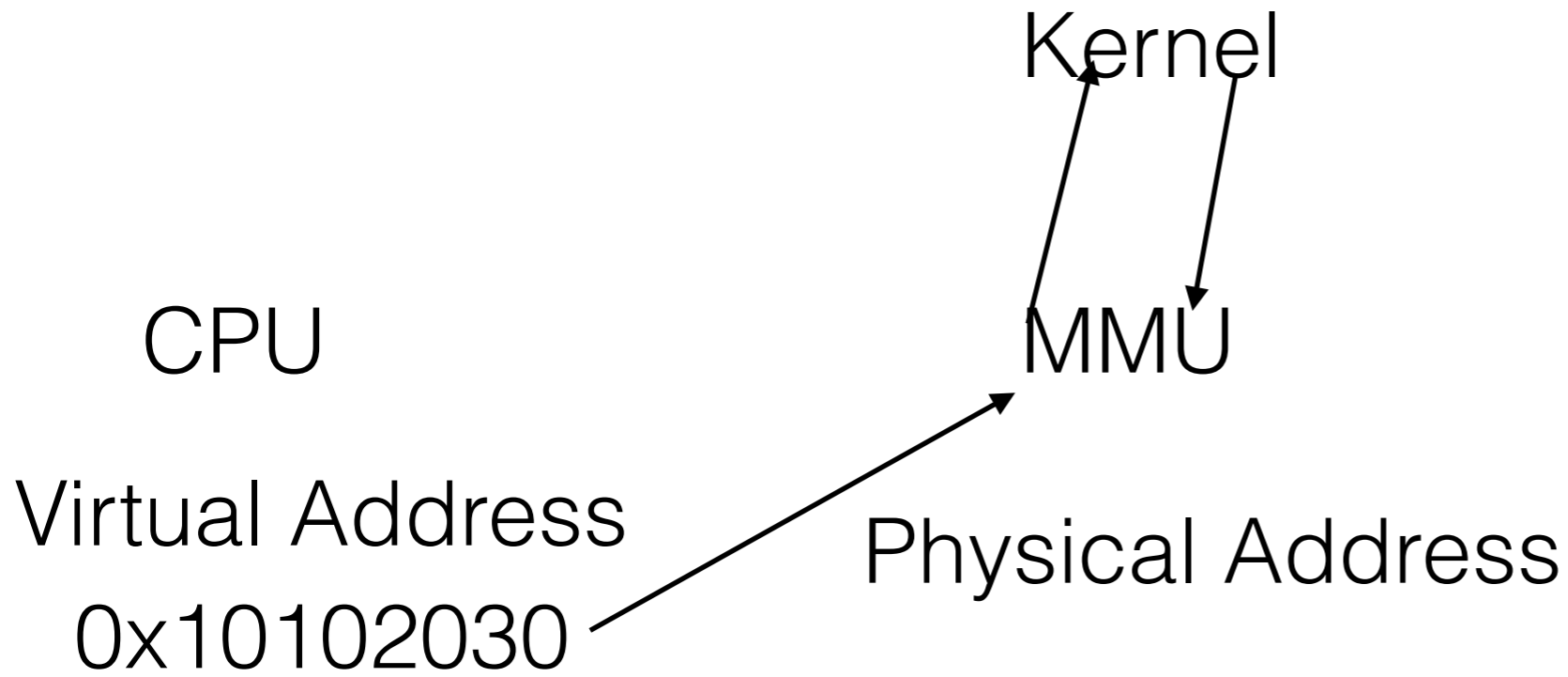


Physical Memory



# General Address Translation

---

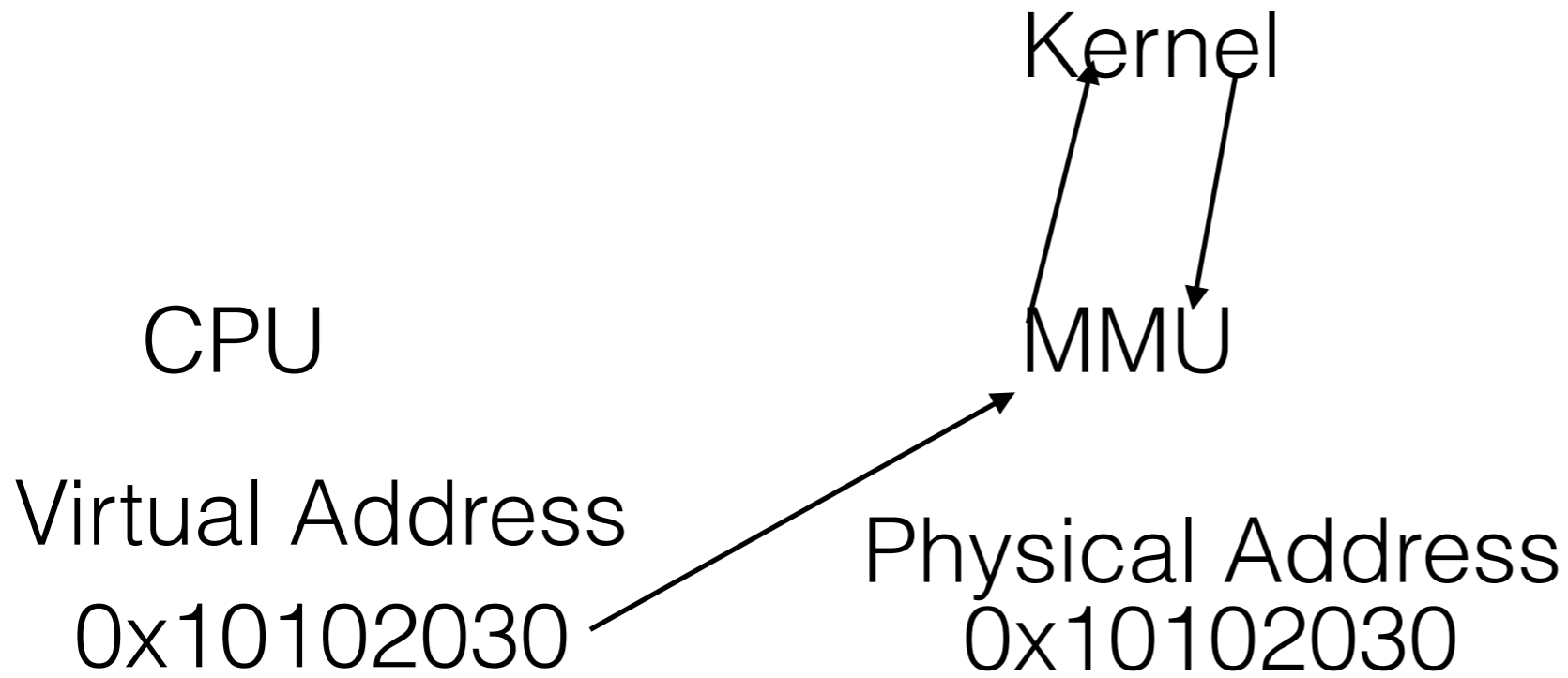


Physical Memory



# General Address Translation

---



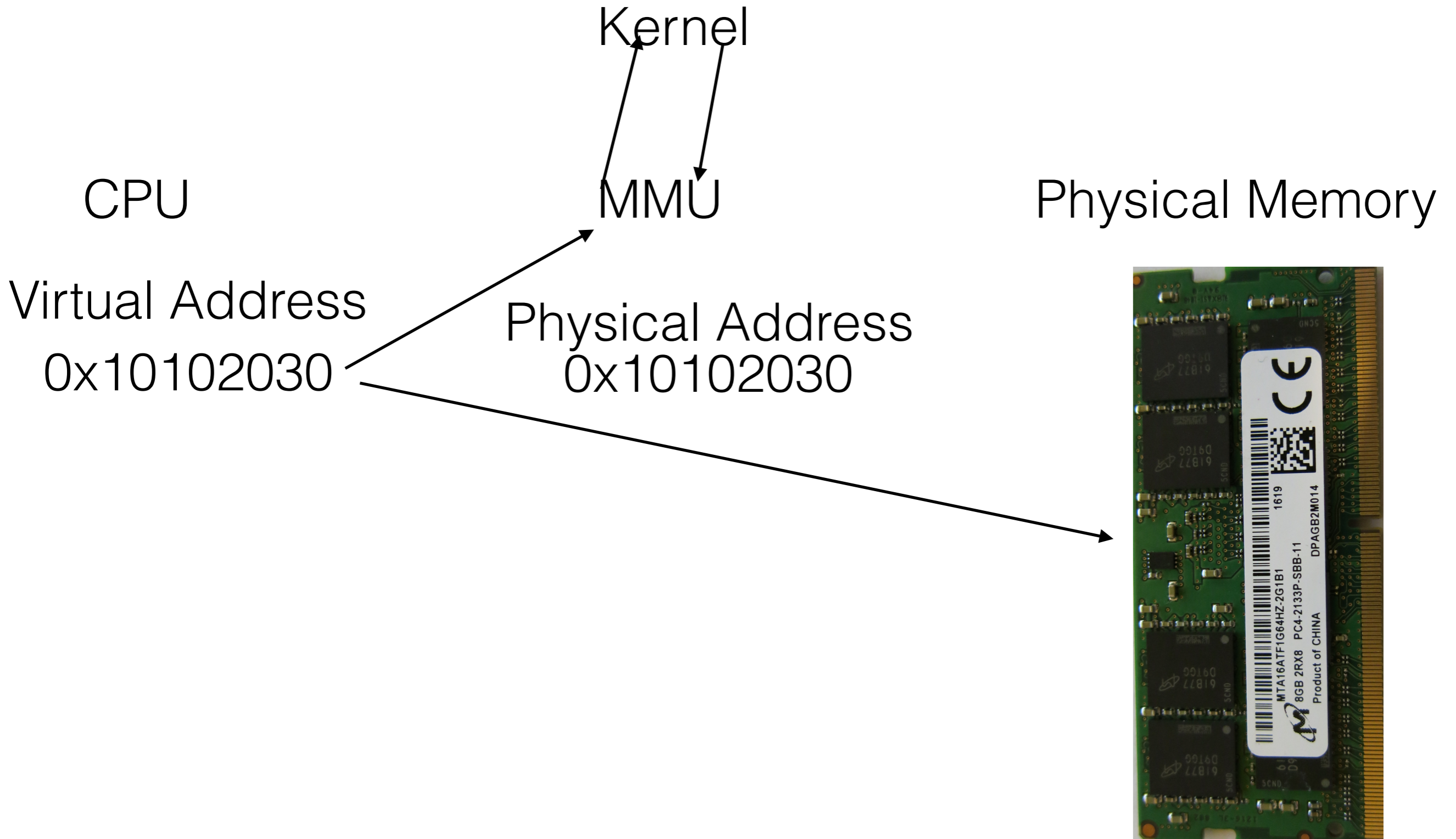
Physical Memory





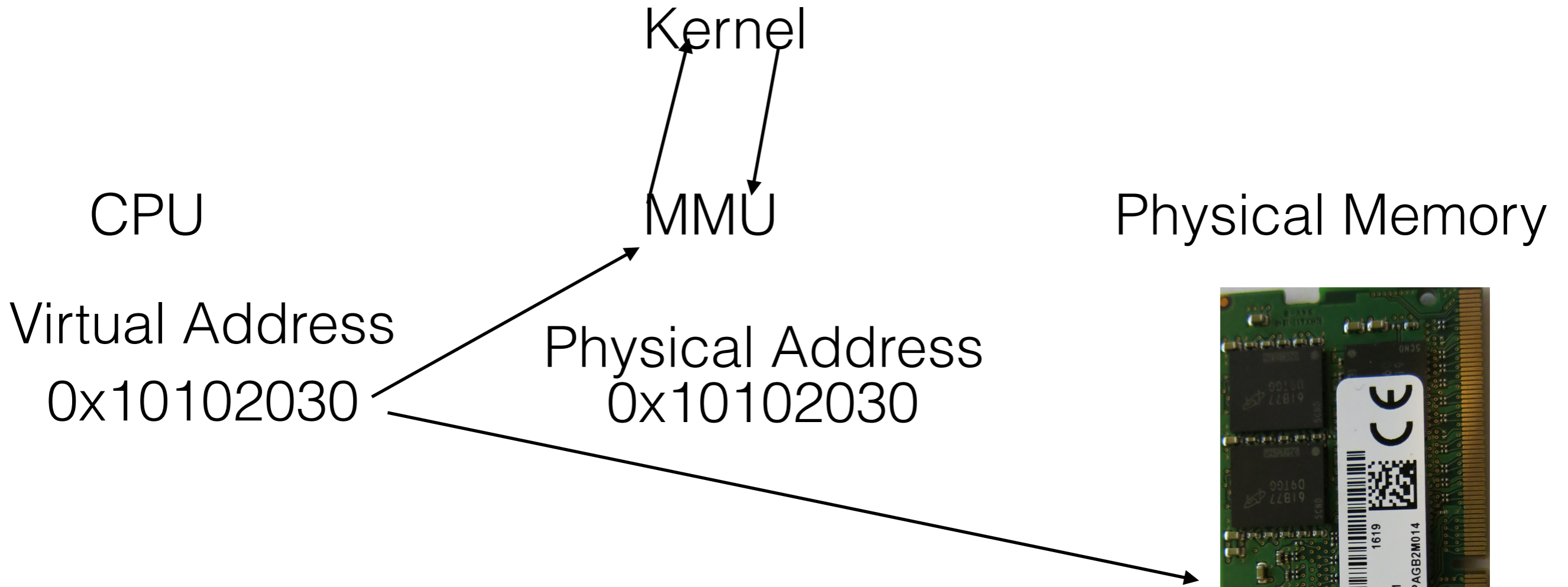
# General Address Translation

---



# General Address Translation

---

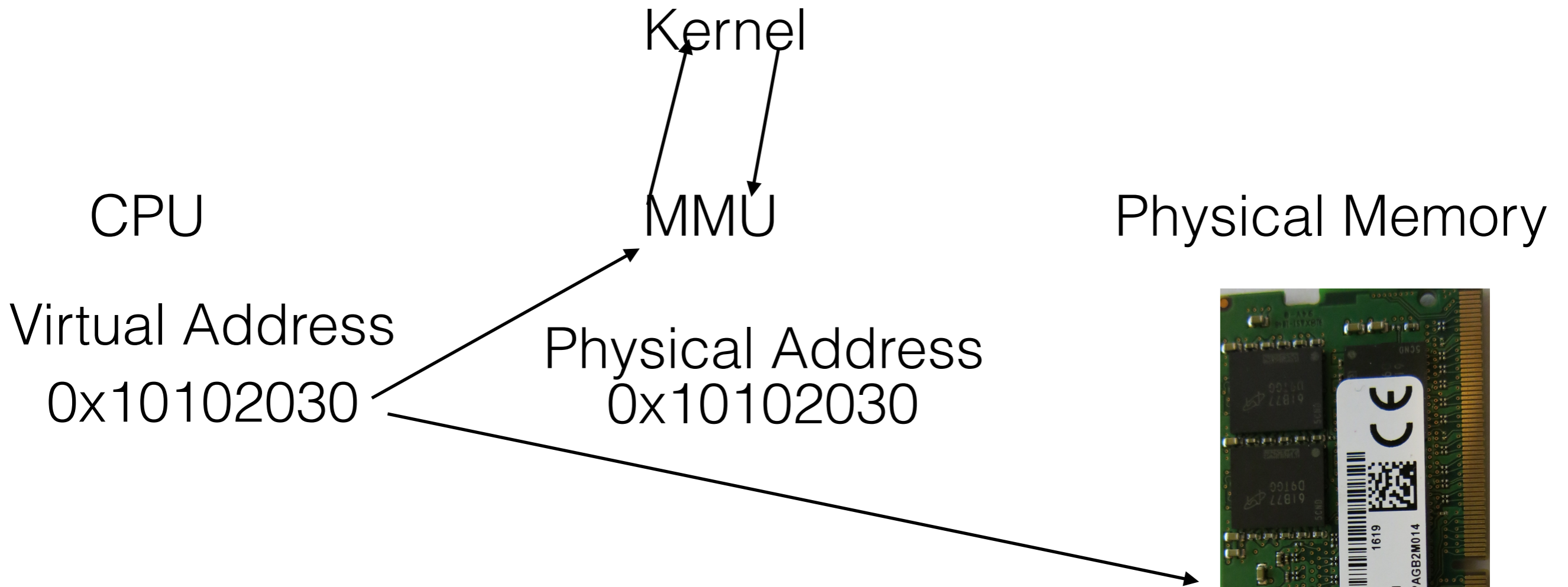


What if you want to translate same virtual address again?



# General Address Translation

---

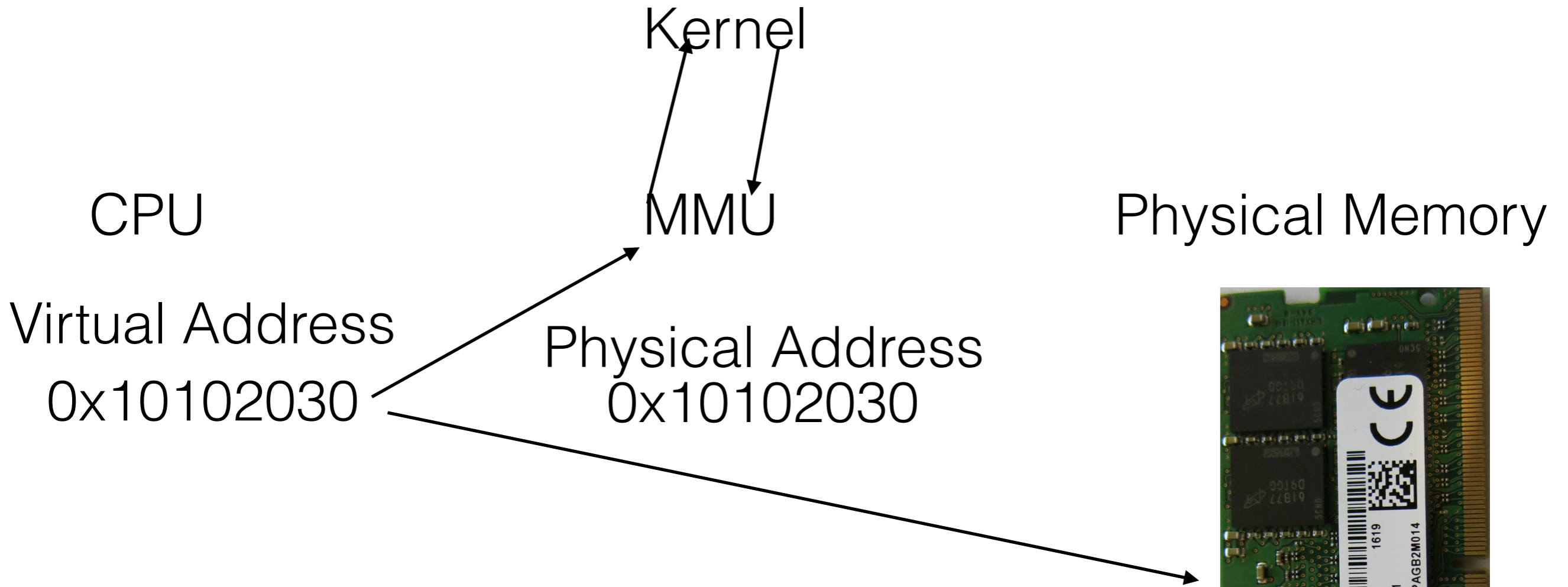


What if you want to translate same virtual address again?



# General Address Translation

---

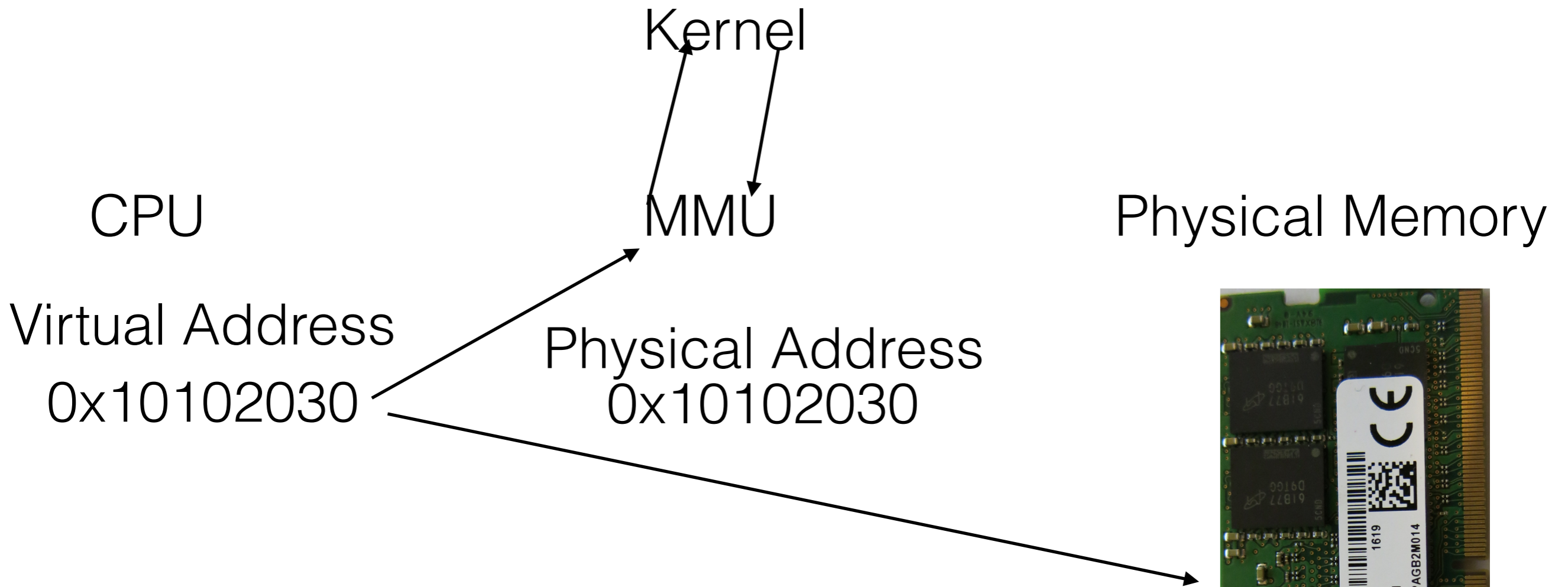


What if you want to translate same virtual address again?

Cache!!

# General Address Translation

---



What do you do with cache if there is a context switch?