CS 1217

Lecture 12

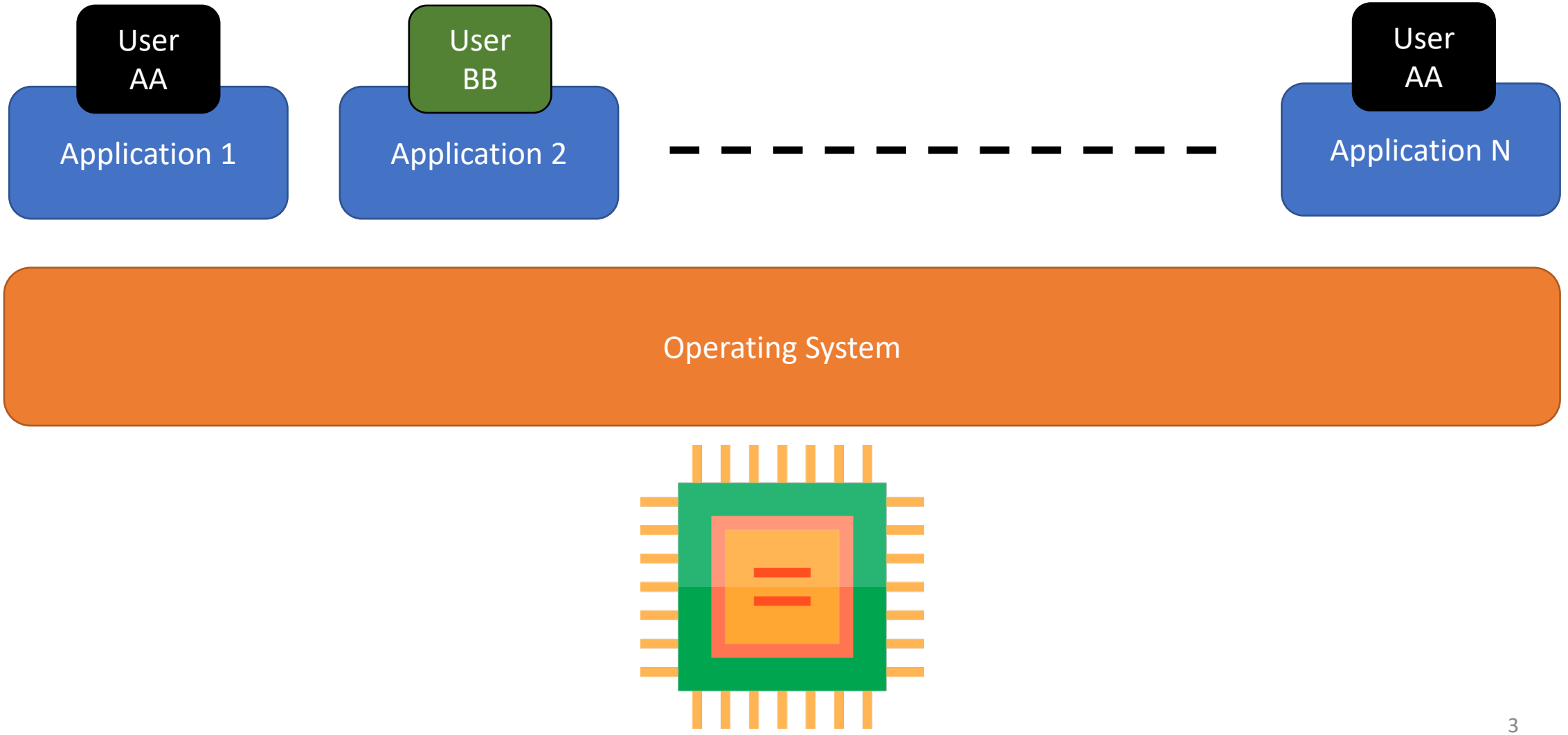
Intro to Memory Management, Virtual Memory

Logistics

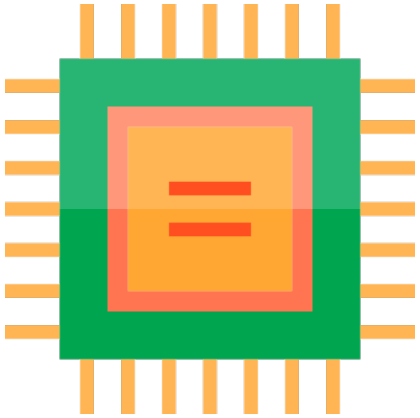
- Lab 2 will be coming out this week

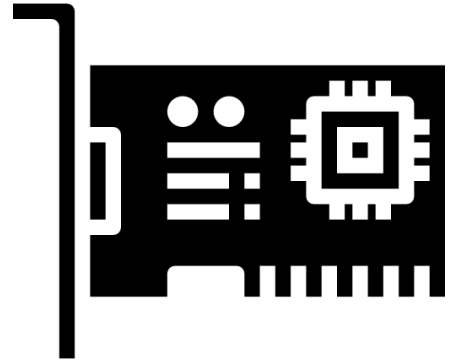
- Midterm Exam : Saturday, 18th March

Virtualizing the CPU



Major System Components





Let us talk about Memory Now

MANAGING DATA LOCALITY IN FUTURE MEMORY HIERARCHIES USING A HARDWARE SOFTWARE CODESIGN APPROACH

by

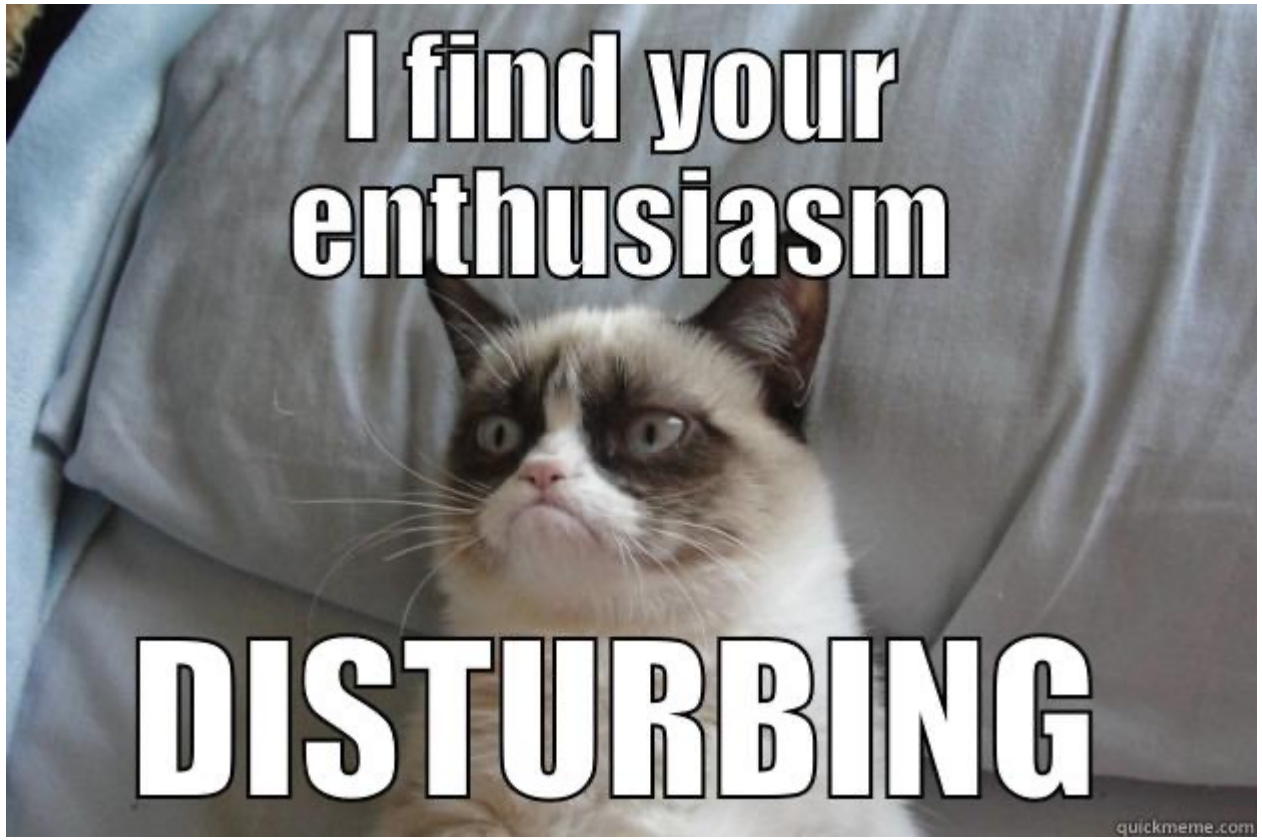
Manu Awasthi

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science



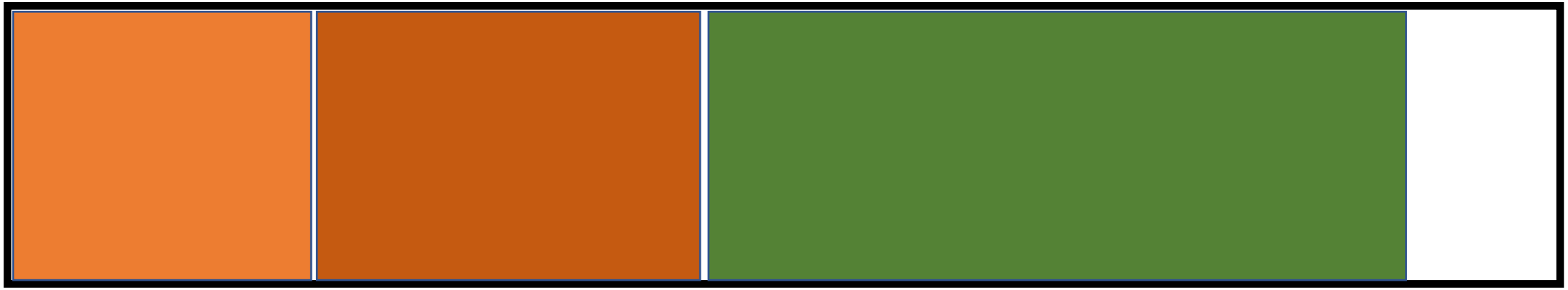
Spatio - Temporal Resource Multiplexing

- Resources can be multiplexed in two ways: **space** and **time**

- **Time Multiplexing:** sharing a resource by dividing up access to it over time.
  - Ride “sharing”
  - CPU scheduling on a single-core system

- **Space Multiplexing:** sharing a resource by dividing it into smaller pieces.
  - Shared apartments
  - Memory management

Chunk up Physical Memory

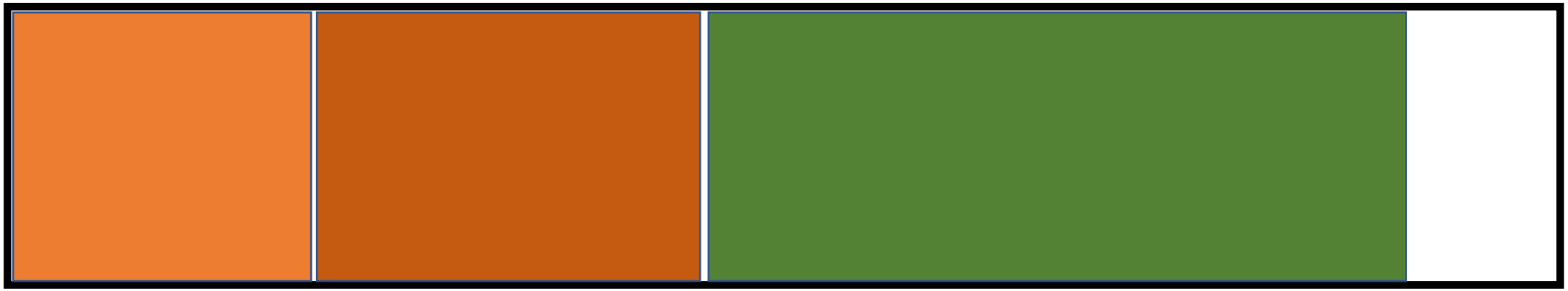


Issues

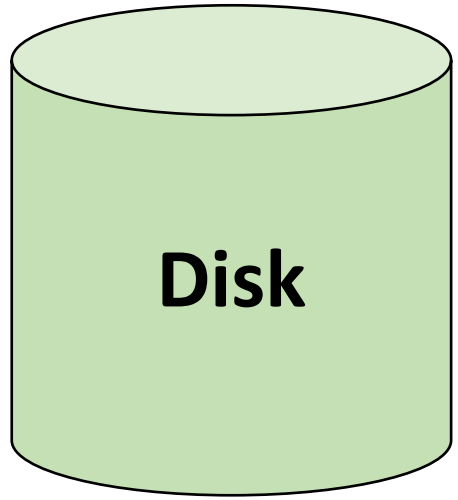
- Allocating Process some more memory at run time?

- Fragmentation
  - Internal
  - External

Capacity Issues

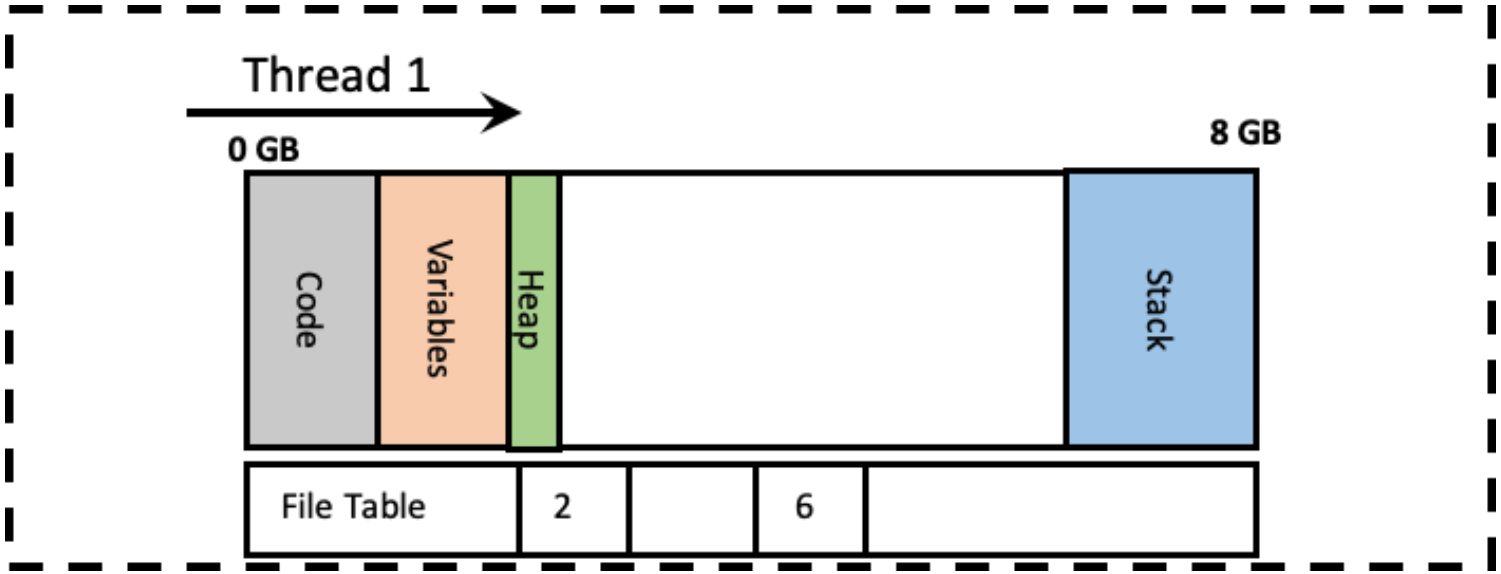


0 GB **Main Memory** **16 GB**



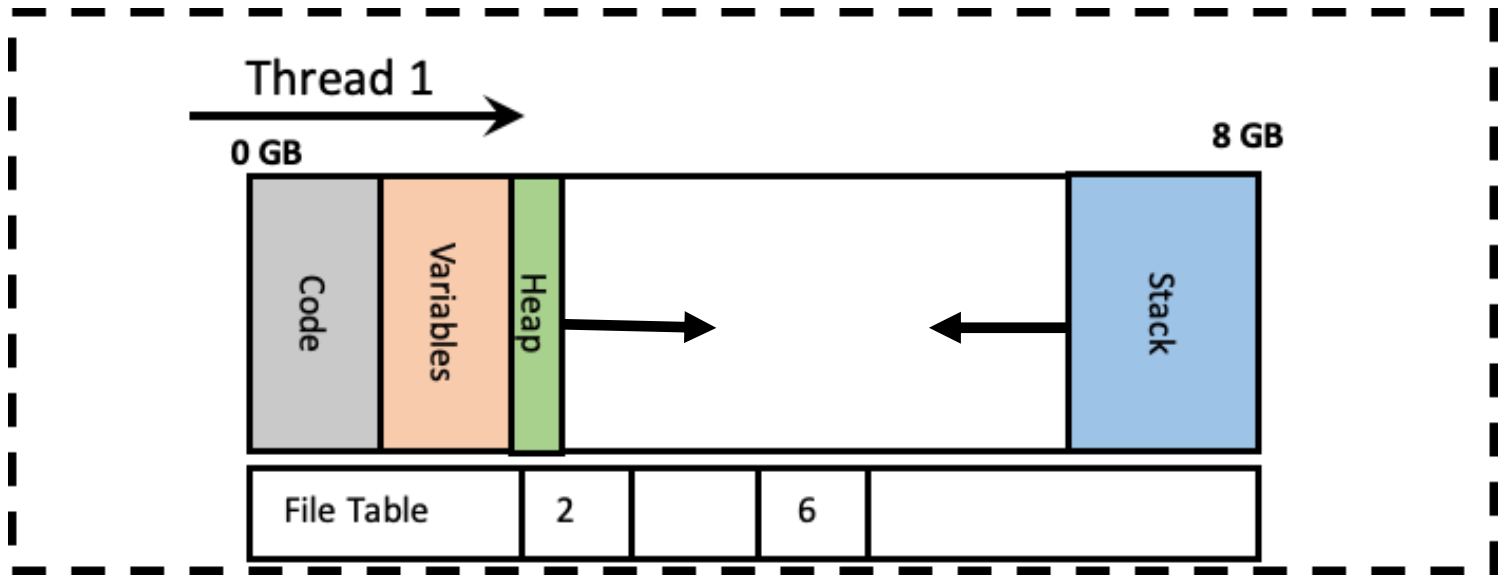
Process Address Spaces

- Processes are under illusion that they have access to **entire**, **contiguous** address space



How does this help?

- The uniformity of address spaces simplifies process **layout**:
  - I always put my code and static variables at 0x10000."
  - "My heap always starts at 0x20000000 and grows **up**."
  - "My stack always starts at 0xFFFFFFFF and grows **down**."



Benefits

- Process layout is specified by Executable and Linker Format (ELF) file.
- Some layout is the function of **convention**.

```
cs304@cs304-devel:~/test$ readelf -l cpu_test | more

Elf file type is EXEC (Executable file)
Entry point 0x400a30
There are 6 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  LOAD           0x0000000000000000 0x0000000000400000 0x0000000000400000
                 0x00000000000b56b6 0x00000000000b56b6  R E    0x200000
  LOAD           0x00000000000b6120 0x00000000006b6120 0x00000000006b6120
                 0x00000000000051b8 0x00000000000068e0  RW     0x200000
  NOTE           0x0000000000000190 0x0000000000400190 0x0000000000400190
                 0x0000000000000044 0x0000000000000044  R      0x4
  TLS            0x00000000000b6120 0x00000000006b6120 0x00000000006b6120
                 0x0000000000000020 0x0000000000000060  R      0x8
  GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     0x10
  GNU_RELRO      0x00000000000b6120 0x00000000006b6120 0x00000000006b6120
                 0x0000000000002ee0 0x0000000000002ee0  R      0x1
```

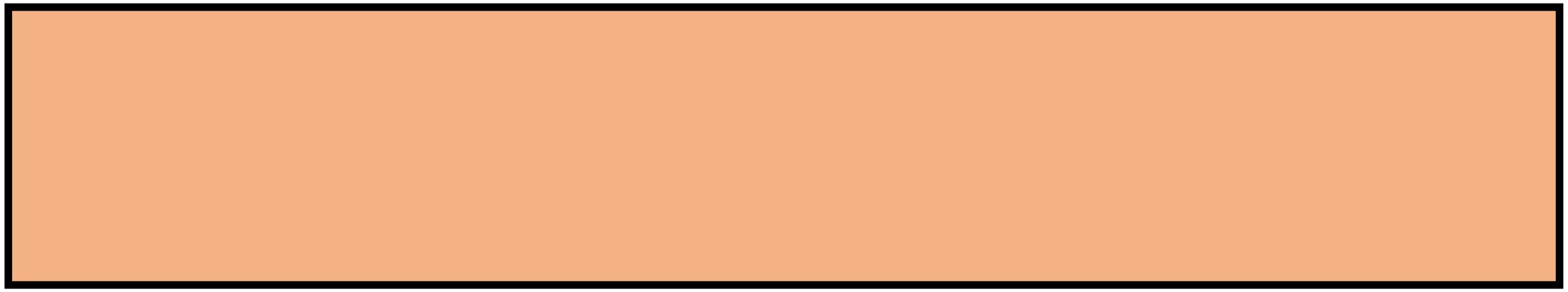
Other Benefits : Relocatibility

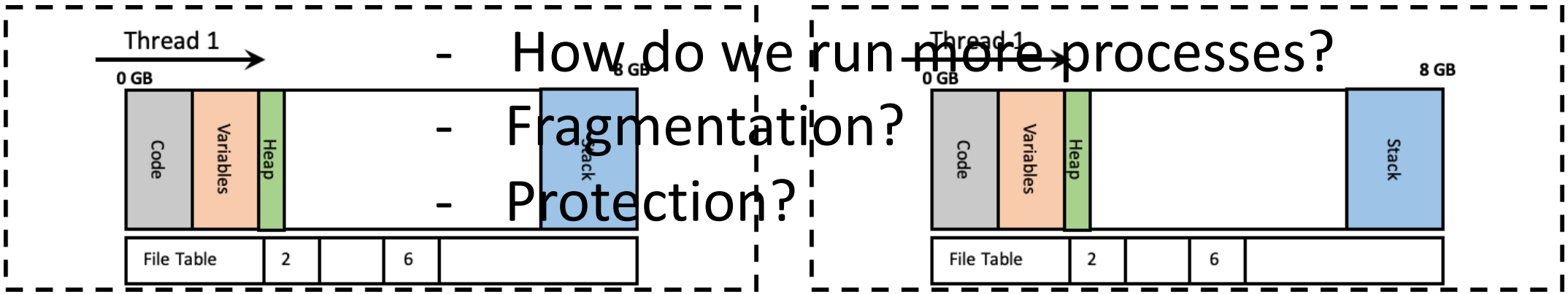
```
int data[128];

data[5] = 8; // Where is data[5]?

result = bar(data[28]); // Where is bar?
```

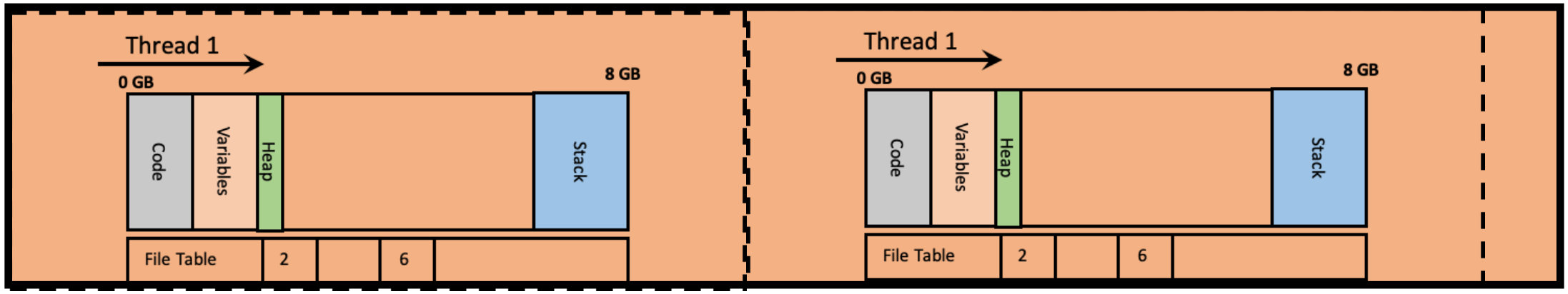
Implementation Issues



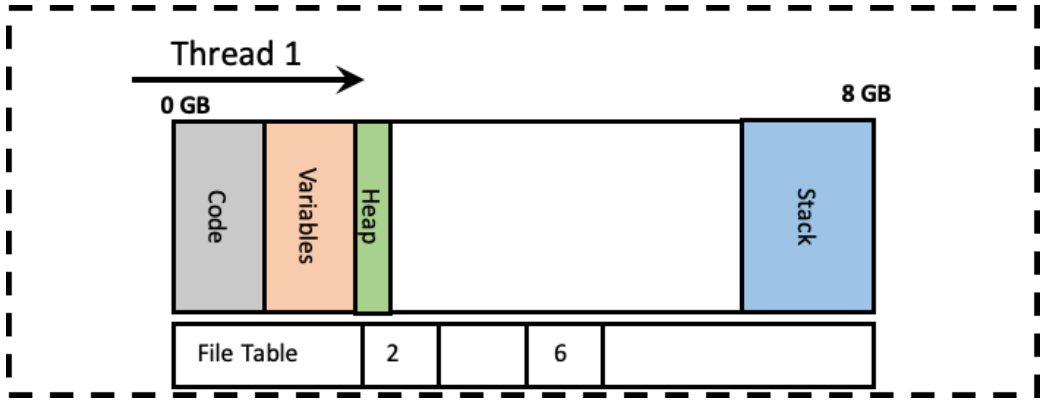


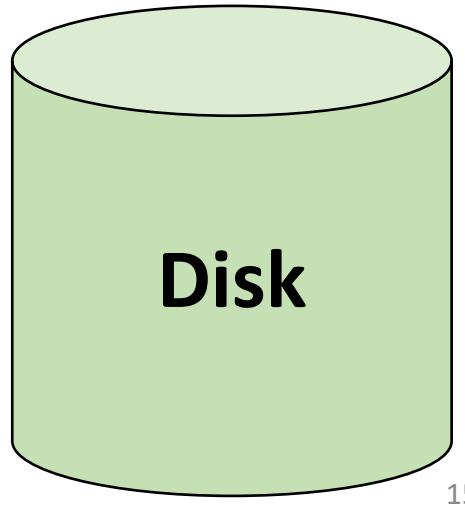
- How do we run more processes?
- Fragmentation?
- Protection?

Solutions



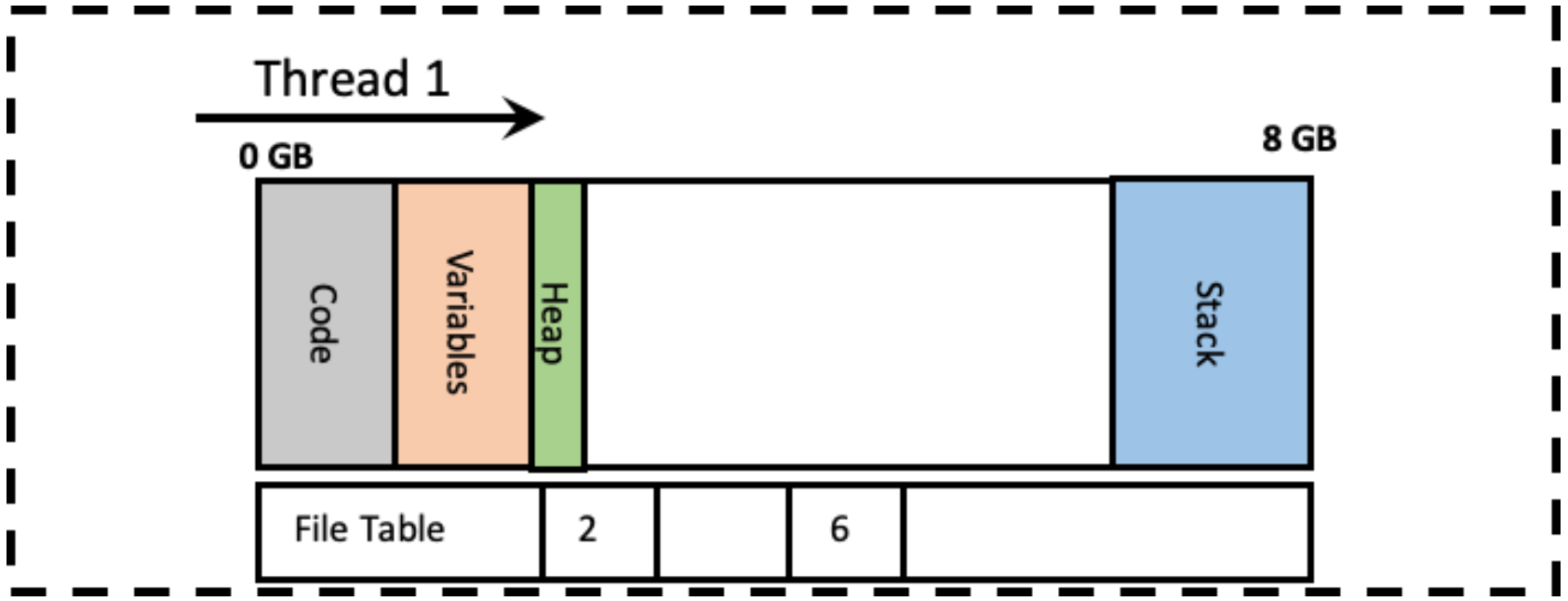
0 GB **Main Memory** **16 GB**





Other Issues

- Process **might not need all** the space that it thinks it has.
- Managing the memory at a finer granularity might help



Solution : Add another level of indirection

- “All problems in computer science can be solved by another level of indirection” – David Wheeler

“Fundamental theorem of software engineering"

https://en.wikipedia.org/wiki/David_Wheeler_(computer_scientist)

“Virtual” Address Spaces

- **Address translation**: 0x10000 to Process 1 is not the same as 0x10000 to Process 2 is not the same as…

- **Protection**: address spaces are intended to provide a *private* view of memory to each process.

- **Memory management**: together one or several processes may have **more address space** allocated than physical memory on the machine.