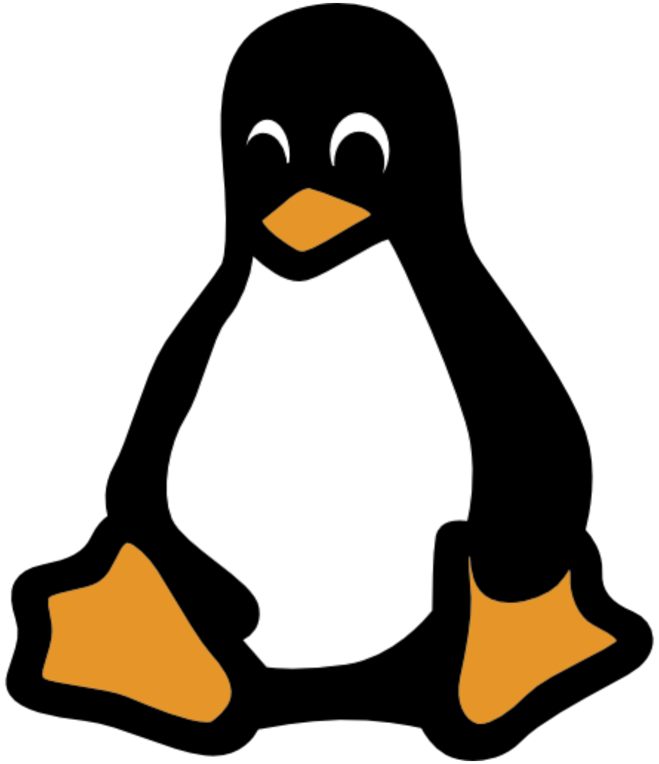
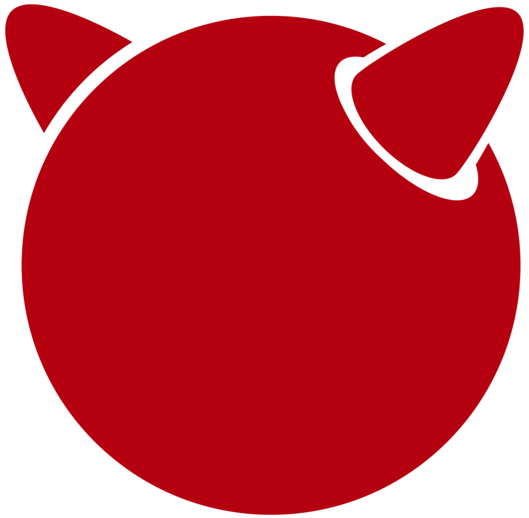
CS 1217 Operating Systems

Lecture 1 - Intro to Operating Systems, Course Logistics

Spring 2023

Operating Systems everywhere!





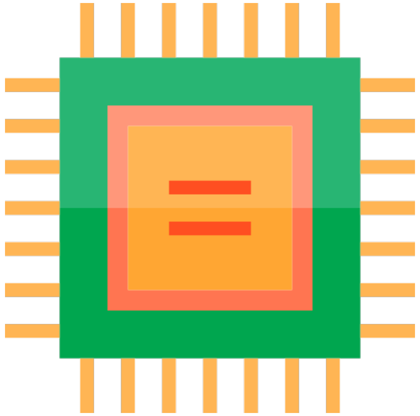
What is an OS?

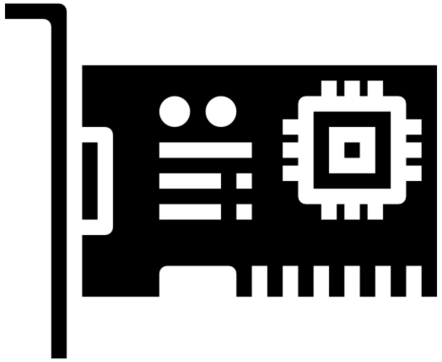
- A computer **program** that does a bunch of things
- Manages **hardware resources**, possibly among multiple programs
- Provides useful **abstractions**

- Overall, makes using the computing system **easier**

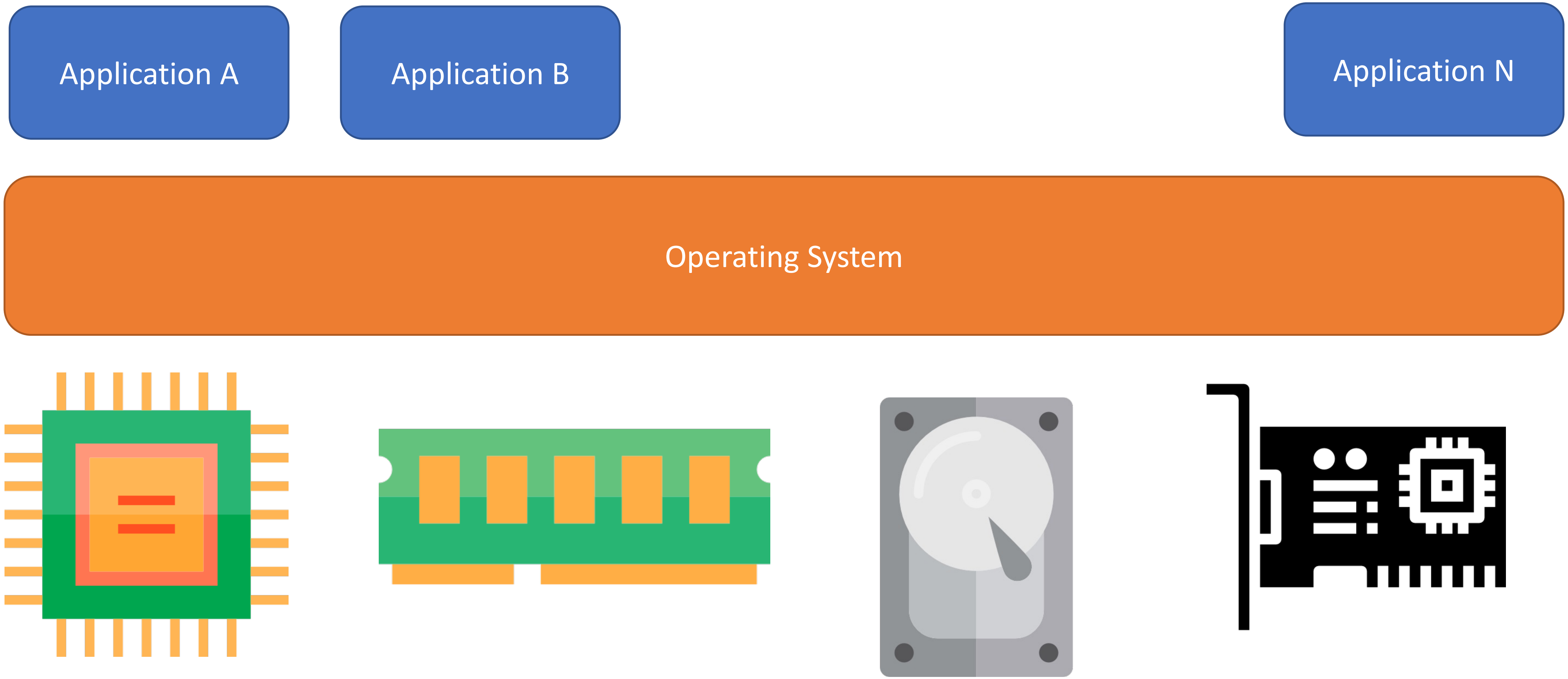
- Also provides multiplexing, isolation, safety etc.
- Ability to add more features, capabilities without affecting user programs

Major System Components





Manage Resources



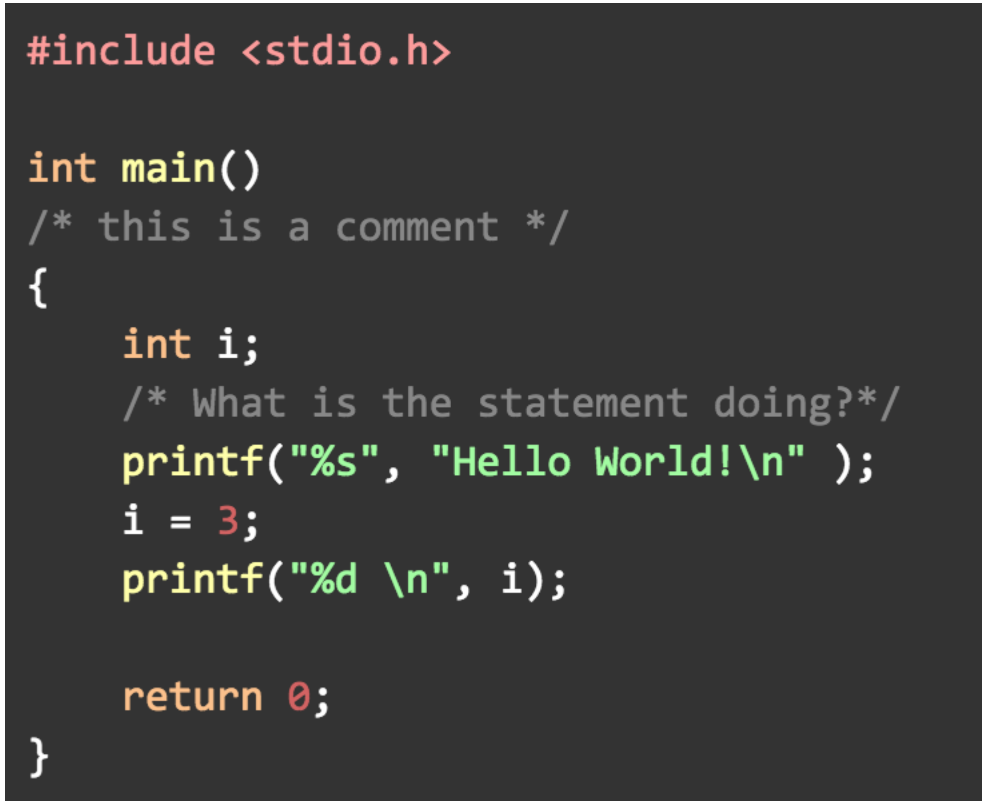
Application A

Application B

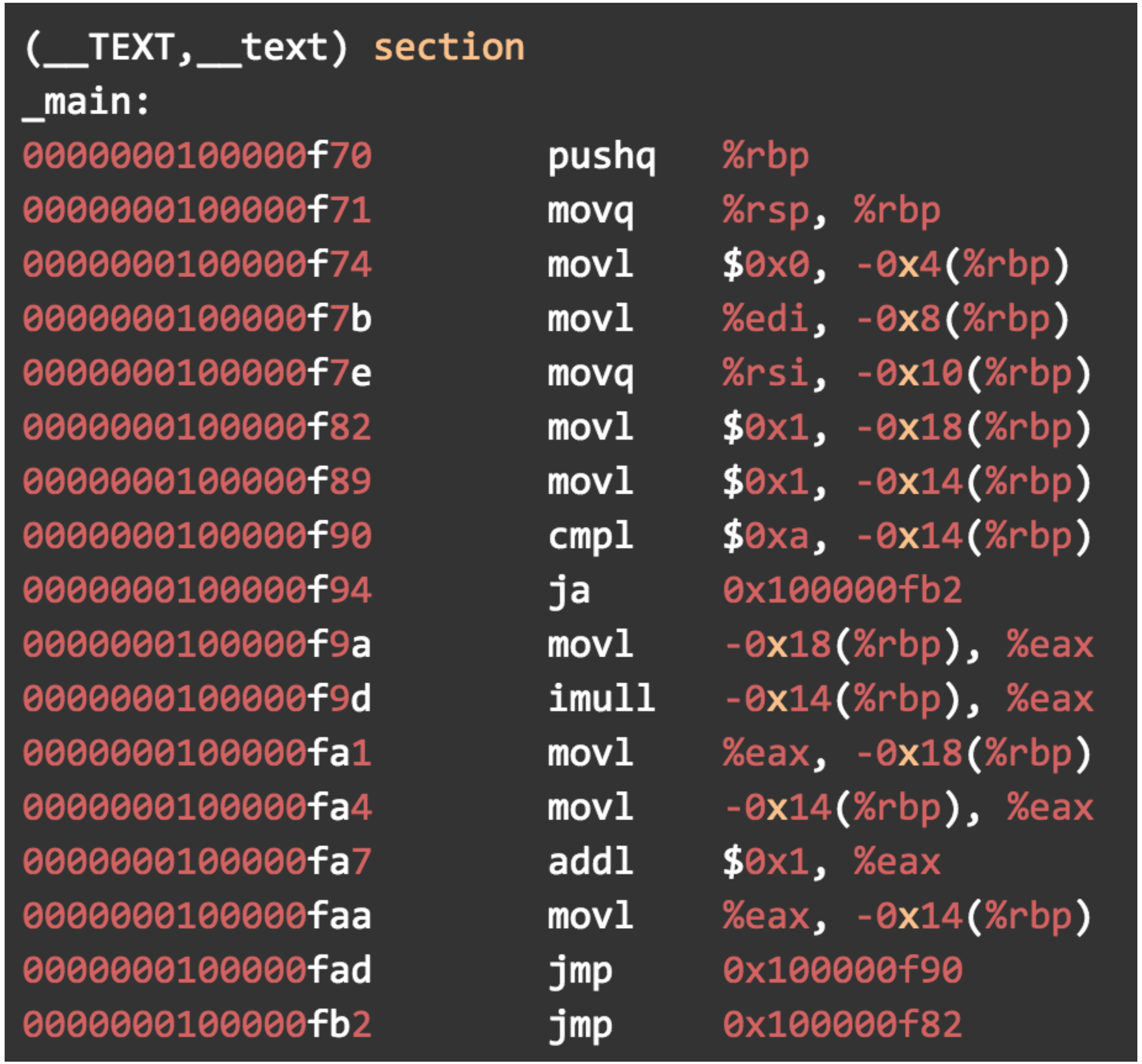
Application N

Operating System

Code -> Execution

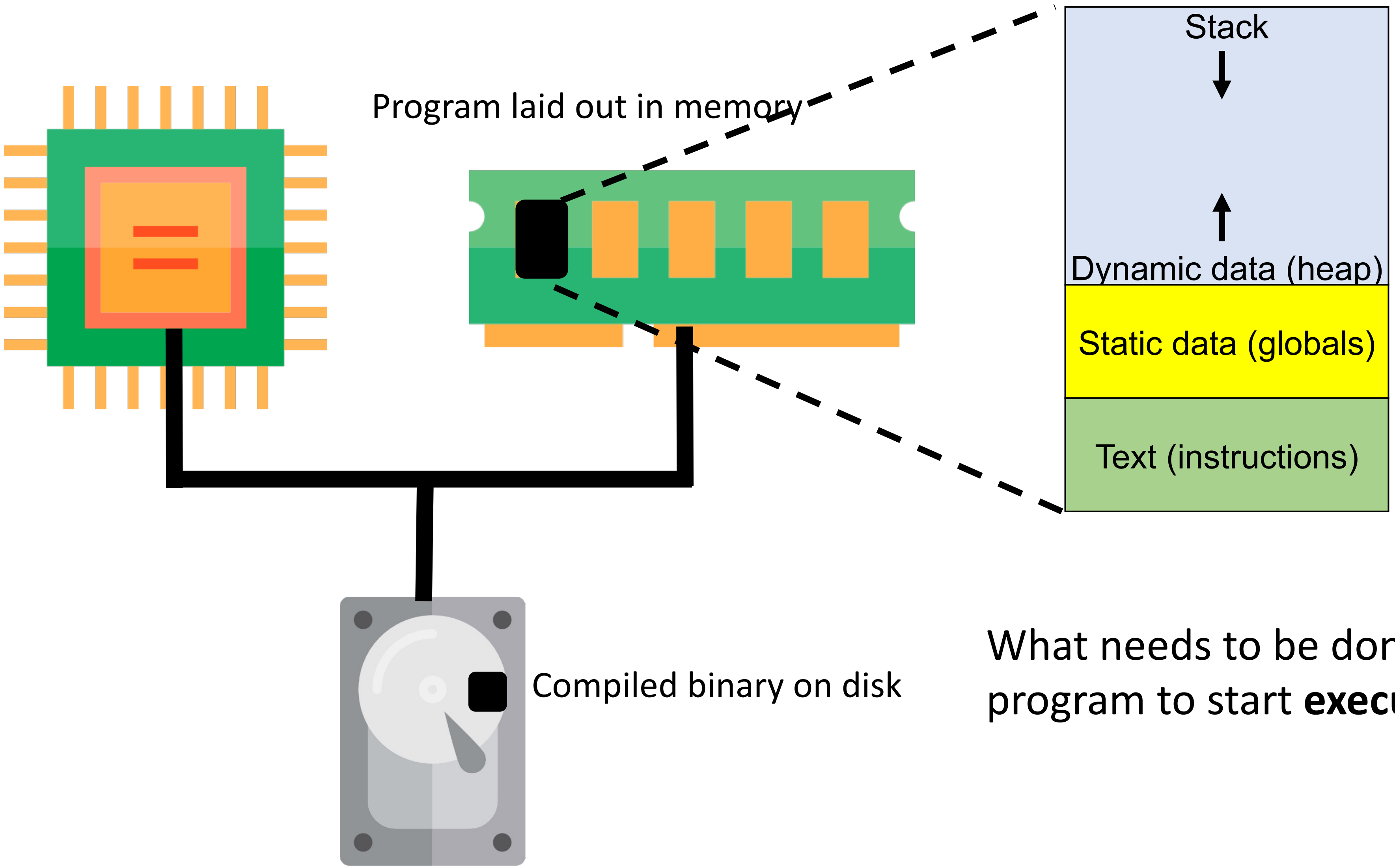


```
#include <stdio.h>

int main()
/* this is a comment */
{
    int i;
    /* What is the statement doing?*/
    printf("%s", "Hello World!\n" );
    i = 3;
    printf("%d \n", i);

    return 0;
}
```



```
(__TEXT,__text) section
_main:
0000000100000f70        pushq   %rbp
0000000100000f71        movq    %rsp, %rbp
0000000100000f74        movl    $0x0, -0x4(%rbp)
0000000100000f7b        movl    %edi, -0x8(%rbp)
0000000100000f7e        movq    %rsi, -0x10(%rbp)
0000000100000f82        movl    $0x1, -0x18(%rbp)
0000000100000f89        movl    $0x1, -0x14(%rbp)
0000000100000f90        cmpl    $0xa, -0x14(%rbp)
0000000100000f94        ja      0x100000fb2
0000000100000f9a        movl    -0x18(%rbp), %eax
0000000100000f9d        imull   -0x14(%rbp), %eax
0000000100000fa1        movl    %eax, -0x18(%rbp)
0000000100000fa4        movl    -0x14(%rbp), %eax
0000000100000fa7        addl    $0x1, %eax
0000000100000faa        movl    %eax, -0x14(%rbp)
0000000100000fad        jmp     0x100000f90
0000000100000fb2        jmp     0x100000f82
```

Life Cycle of Program



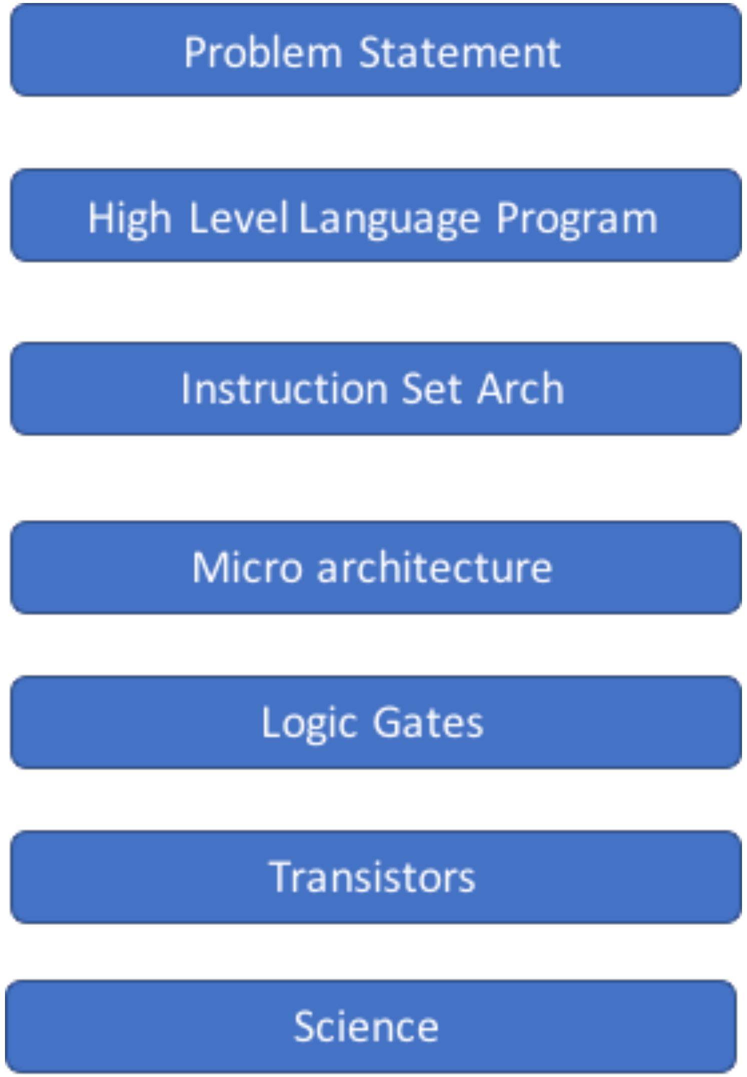
What needs to be done for the program to start **executing**?

Abstractions

- Abstractions *simplify application design* by:

- **hiding undesirable properties**
  - Helps us reason easily about system components and behavior
- **Adding hierarchies for easier understanding and implementation**

- OS abstractions provide an **interface** to application programmers



Problem Statement

High Level Language Program

Instruction Set Arch

Micro architecture

Logic Gates

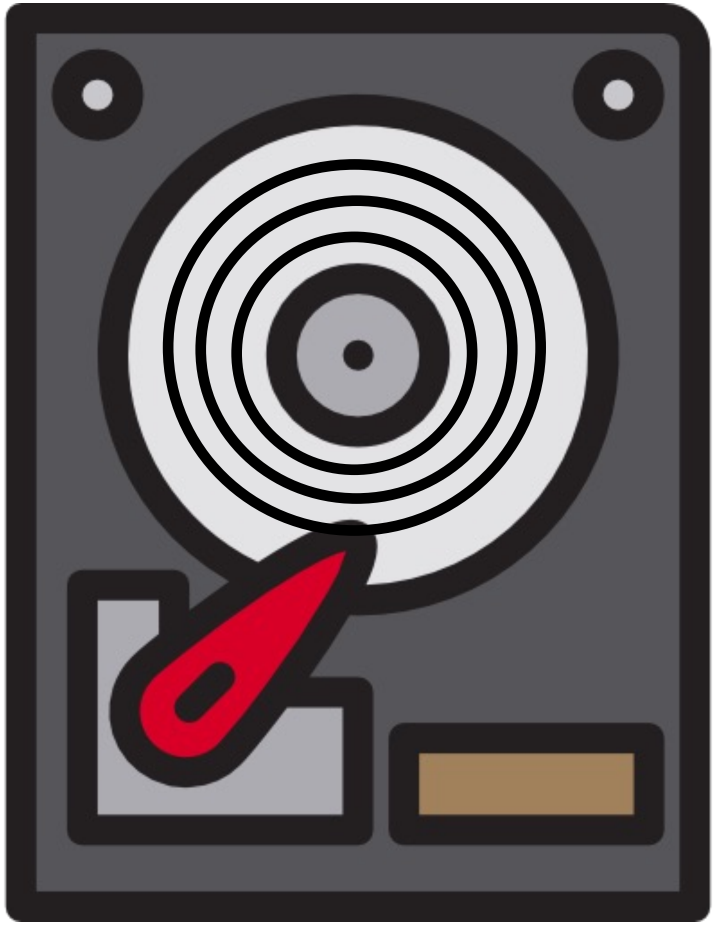
Transistors

Science

Example Abstraction: Files (Disk)

- You want to read a file from disk
- Mechanism 1
  - Platter 4, Track 84, Sector 32 ***then***
  - Platter 6, Track 2048, Sector 212, ***then***
  - …
- Mechanism 2

```
fd = fopen("MyData.txt",”w");
fprintf(fd,"%d",num);
fclose(fd);
```



Policy vs Mechanism

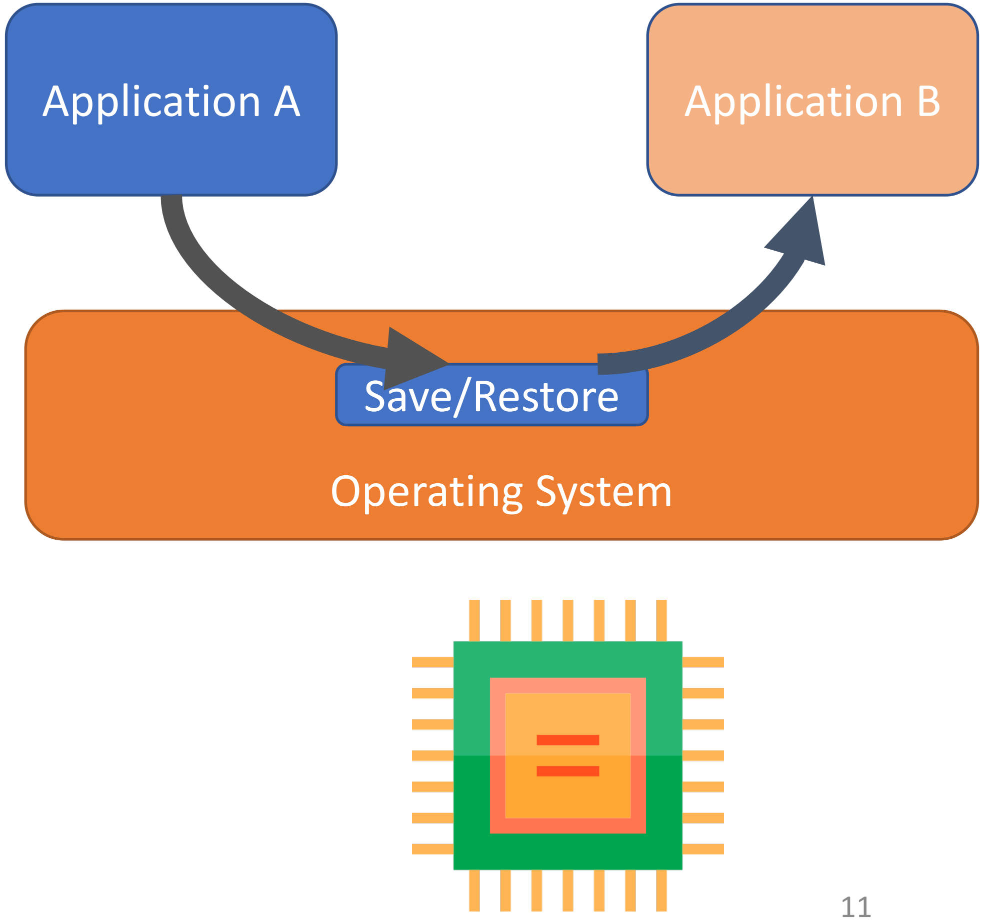
- **Policy**: What do you want to do?

- **Mechanism**: means to an end, how do you want to actually implement the policy
  - One policy can have multiple implementations
- Example:
- **Policy** : an application should be able to read a file from disk using “fopen()”
- **Mechanism**: actual implementation of fopen, can vary between systems

Resource Management: CPU

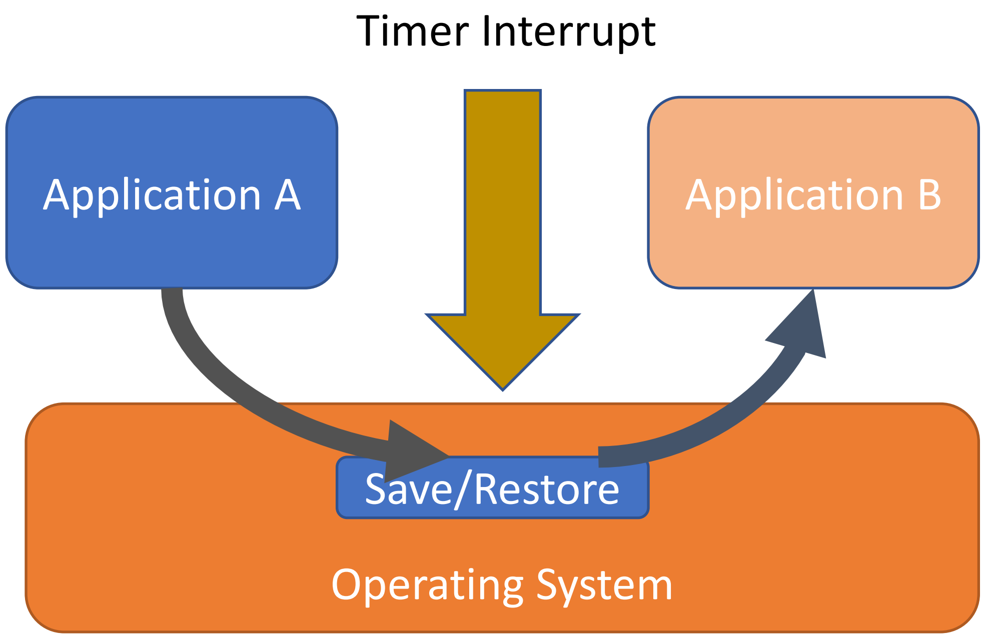
- Many applications run “concurrently” on a machine

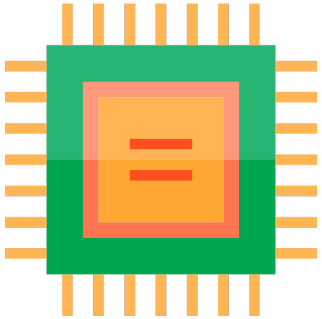
- “Virtualizing” the CPU: Every process thinks that it is the only one that is running on the system



Resource Management: CPU : Scheduling

- Multiple applications are running
- Who gets to "hog" the CPU?
- How long does it get to do that for?
- Which program should be run next?





CPU Virtualization Demo

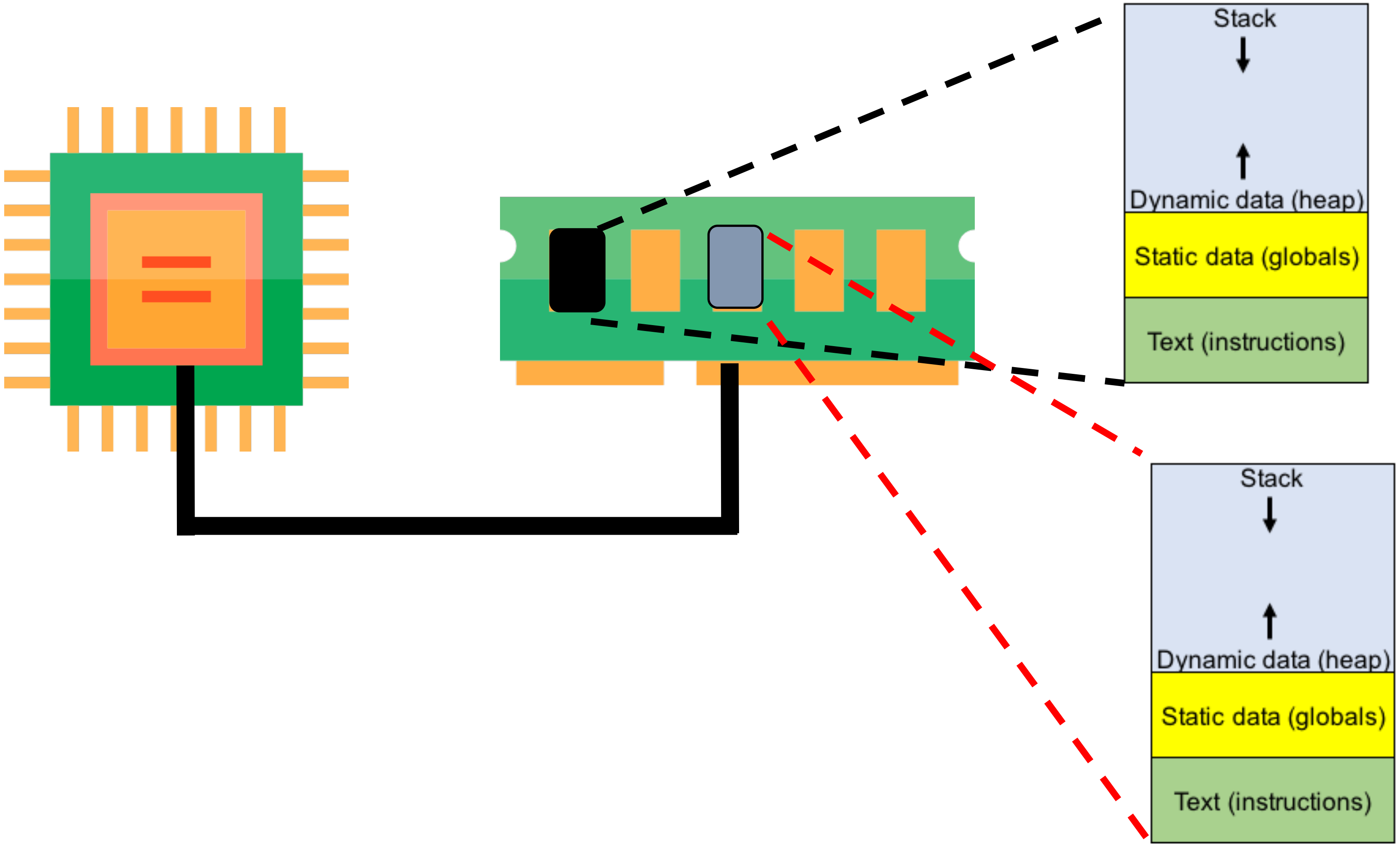
Resource Management: Memory

- Memory is shared by multiple processes; physically a single resource

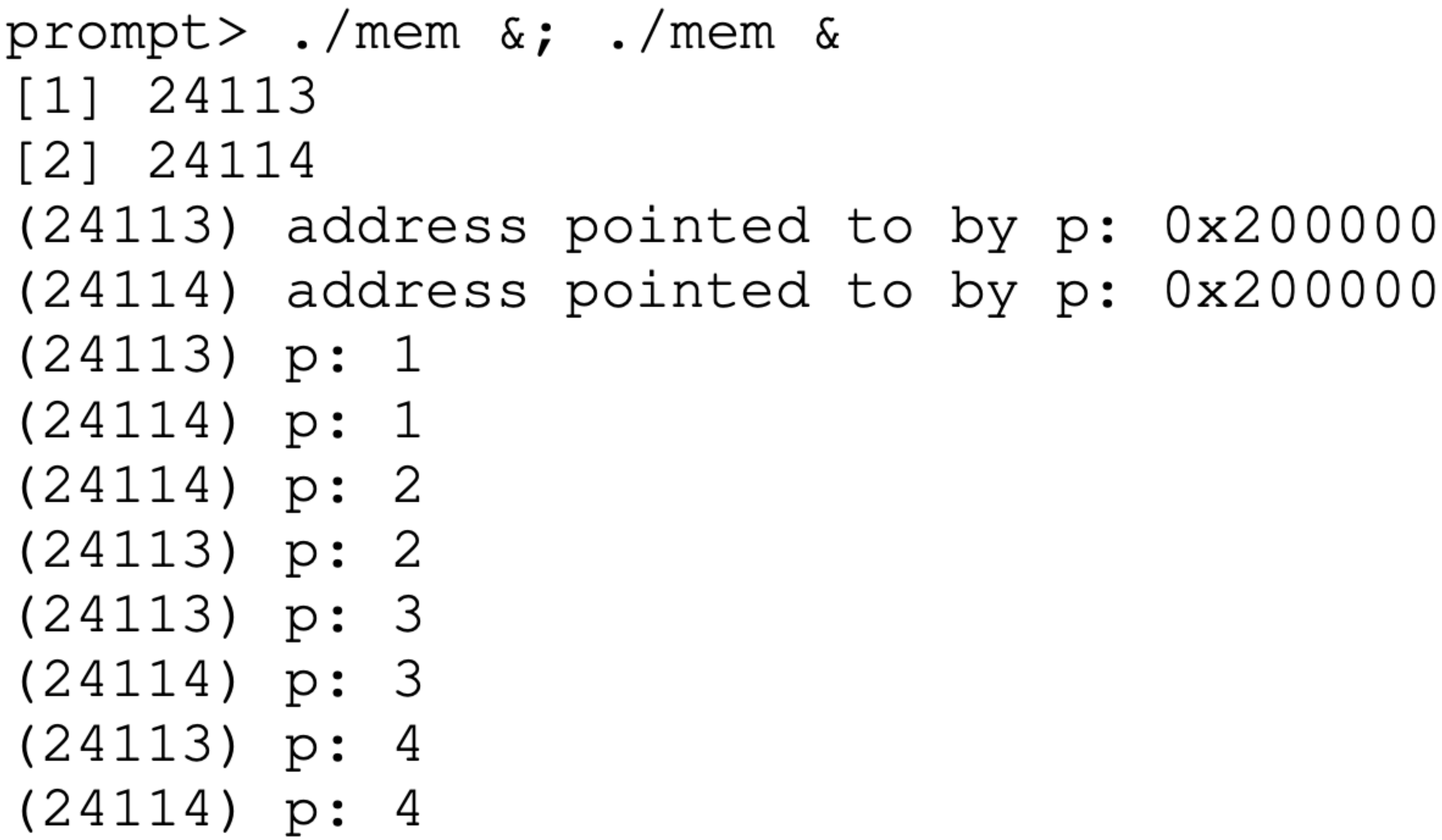
- Each program, while running “thinks” that it is the only program, with access to “all” the memory on the system

- OS virtualizes the memory to provide each program that illusion
  - Literally called virtual memory

Memory Virtualization

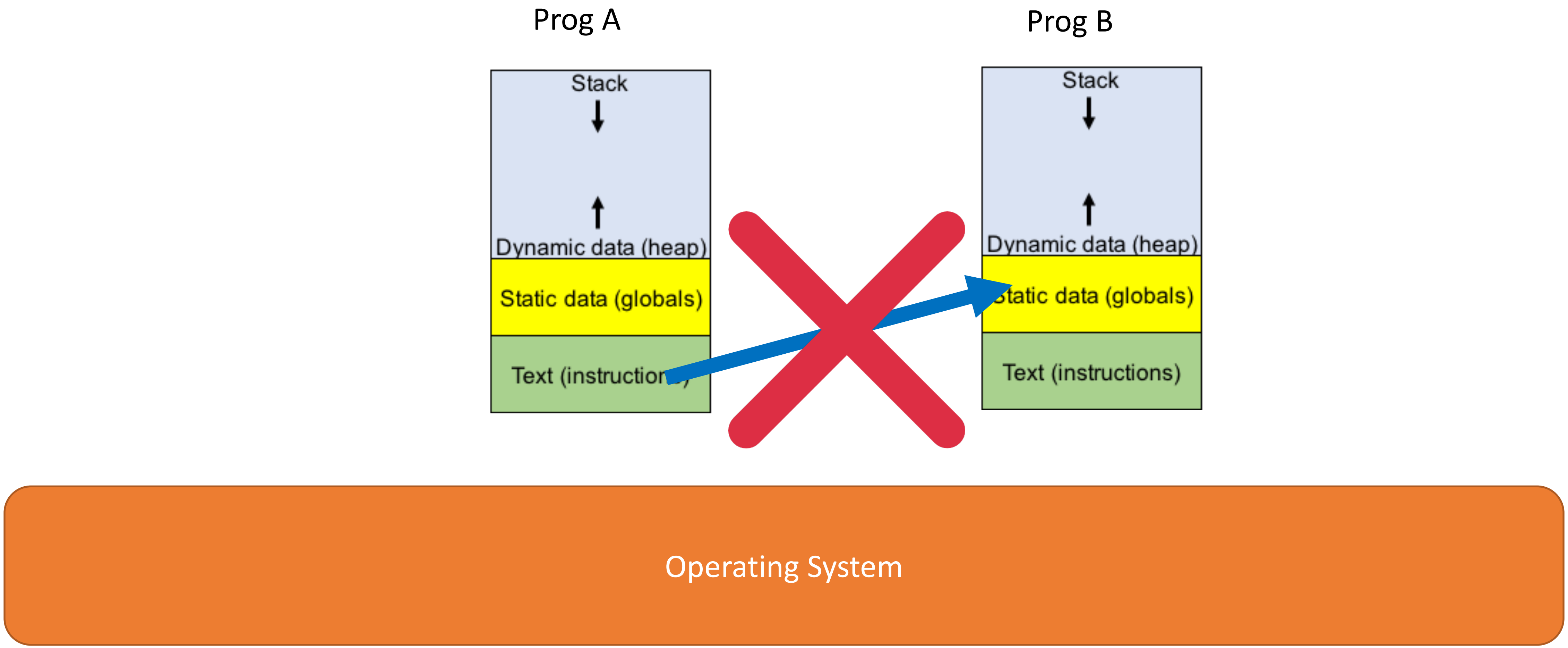


Memory Virtualization Demo



```
prompt> ./mem &; ./mem &
[1] 24113
[2] 24114
(24113) address pointed to by p: 0x200000
(24114) address pointed to by p: 0x200000
(24113) p: 1
(24114) p: 1
(24114) p: 2
(24113) p: 2
(24113) p: 3
(24114) p: 3
(24113) p: 4
(24114) p: 4
```

Process Isolation



Design Goals for OS

- Provide **abstractions** for easier use of a computer system
- Provide **high performance** and keep the **overheads low**
- Provide **protection** between applications
- Do all of the above **reliably**

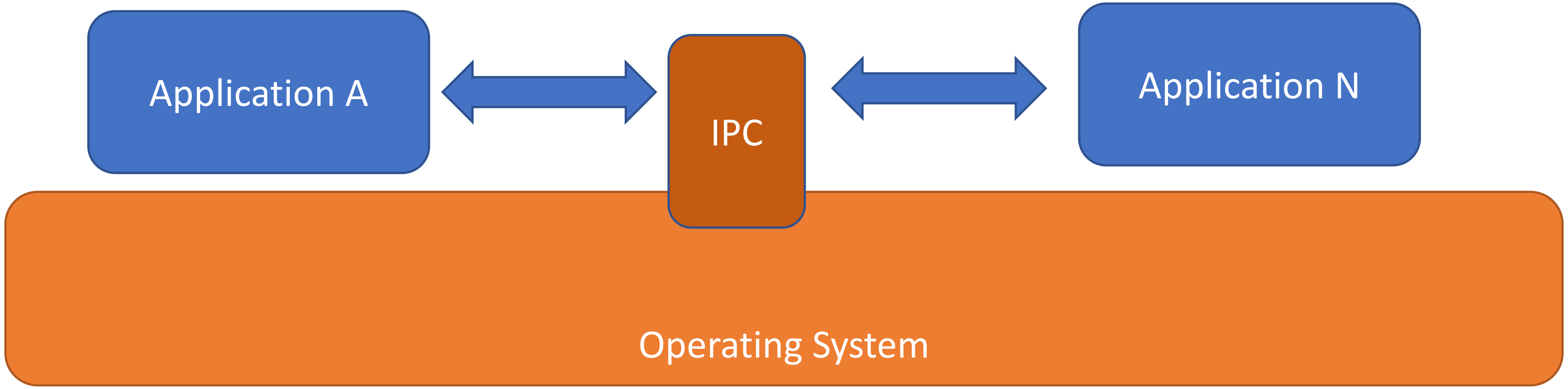
- Secondary goals
  - energy-efficiency
  - Security – provide defense against malicious applications
  - Mobility – ability to run on varied types of devices

What all is the OS supposed to do?

- Process management
- Process scheduling
- Inter-process communication and synchronization
- Memory management
- I/O management

Communication Between Processes

- Is it required?



Name some examples of where this might be required?

Concurrency

- Ability to do work simultaneously, possibly out of order, without affecting the final result
  - Work = code (instructions really)
- Allows for parallelism => higher performance
  - Reduced latency, increased throughput
- Correctness guarantees are paramount
  - Program order should be guaranteed
  - Some can be provided by the hardware
  - Rest need to be taken care of by the OS

Why study Operating Systems

- Understand how computer systems **actually** work
- Neat software systems, optimized over decades
  - Address a number of software design challenges relevant today
  - Emphasizes design before writing code
- Programming for performance, low overheads
  - “closer” to the machine
- Dispel the notion: C is a dead language
- Somebody write me an OS in Python

OS : S/W Engineering, System Design

- Decades worth of person-hours have gone into designing, coding and debugging
- Complex systems, with multiple components, each one is also pretty complex
- Ideas are cheap; implementation develops skills
- developing these skills => overall better employability for you.

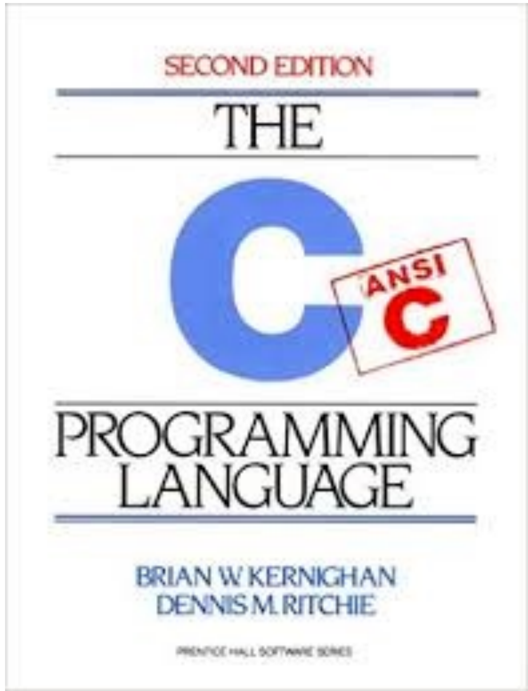
Texts

- Textbook(s)
  - **Operating Systems: Three Easy Pieces** by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau
  - Freely (and legally) available online – www.ostep.org

  - **xv6, a simple, Unix-like teaching operating system** by Russ Cox, Frans Kaashoek and Robert Morris
  - Freely available online - https://pdos.csail.mit.edu/6.828/2018/xv6/book-rev11.pdf
    - We will be referring to rev-11.pdf

Prerequisites

- C coding skills
  - xv6 is written in C
- Coding skills implies
  - read, code and debug
- All assignments/labs are in C
- Many questions about explaining xv6 code
- Be able to code in a Linux environment
- Some x86 assembly language skills; will develop over time



xv6

- xv6 is a “teaching” operating system developed at MIT
- Has around 9K lines of C code
- Boots a basic kernel, uses QEMU to emulate the hardware

- We will learn more about xv6 as class progresses
- Will use if for assignments and labs

Coding Environment

- Linux based VM has been created
- Class page has the link to download, instructions on how to get started.

- Login: cs304
- Password: cs304

- Already has a working repo of xv6 cloned inside it
  - Feel free to play around with it. Instructions will follow

Course Logistics

- Teaching Assistants
  - Soham Bagchi (ASP 23)
  - Neil Chowdhary (ASP 23)
  - Aaryann Mavani (UG 23)
  - More TBD

- Course website : https://sites.google.com/ashoka.edu.in/cs1217
- Has course policies, links to slides, course schedule

Course Logistics

- Information push through Google Classroom

Programming Assignments

- (Mostly) will be done in teams of two
- **Significant** amount of effort; not all that hard
- If you put in the effort, you will get a good grade
- Grading might have a viva + demo component
- Currently deciding on code submission platform; most likely will use Git; otherwise default to Google Classroom

Course Logistics - Assignments

- You are expected to submit the assignments before the assigned deadline. We will enforce the following policies.

- Each team has 3 days of total slack time over the course of the semester.
  - The decrement in slack time will happen at the granularity of 1 day.
  - Once the 3 days are up, any submitted assignments will not be graded.

- There will be **no extensions** for assignments, unless for medical emergencies.

- Email requesting extensions for non-emergency reasons will automatically result in 5% reduction in max possible points

Grading

- Programming Assignments - 25%

- Exams
  - Midterm Exam - 25%
  - Final exam - 30% (cumulative)
  - Both exams will check for conceptual understanding of topics covered in class

- Surprise quizzes – 15%
  - Will be held during class
  - Number/weightage of each is instructor’s discretion

- Zines/educational videos (5%)

Lab Hours

- 2 hour slot : new introduction this year
- Attendance by everyone is **mandatory**
  - **No** weightage towards grade

- Idea is to work on programming assignments during the time
- TAs will be around to help

- Will start next week; need to schedule a slot by end of this week.

Attendance and Participation

- There is no weightage for attendance towards your grade
- However, you are expected to attend ***all*** lectures
  - Gives you a chance to ask questions
  - Everything is **not** in the textbooks
  - Surprise quizzes will happen during class
- Class participation is the best way to learn
  - I might cold call on you
  - Grades, traditionally have been positively correlated with attendance
- Class starts at **3:00 pm**. Please be on time.

Screen usage policy

- Research suggests
  - Laptops in class generally have adverse impact on learning outcomes
  - learning is *also* negatively impacted by *someone else's* laptop use
- References : https://cs.brown.edu/courses/cs019/2018/laptop-policy.html
- No screens, unless explicitly permitted

So, how do I take notes?

Academic Integrity

- A team (of 2 students) will form an **academic integrity unit**
- The team members can cheat among themselves as much as they like
  - Actually, are encouraged to to it!
- Outside of that, teams can discuss, but have to write code on their own.
- Details:
  - If any part is plagiarized, **both** partners fail.
  - If you have concerns about work your partner has submitted, immediately approach instructor.
  - If you do not we will assume later that your consent was given.
  - It is entirely your responsibility to ensure that your team's submission is fair and reflects your contributions

Academic Integrity

- If you cheat, I’ll make sure you fail
    - And will try my best to make sure that everybody knows about it

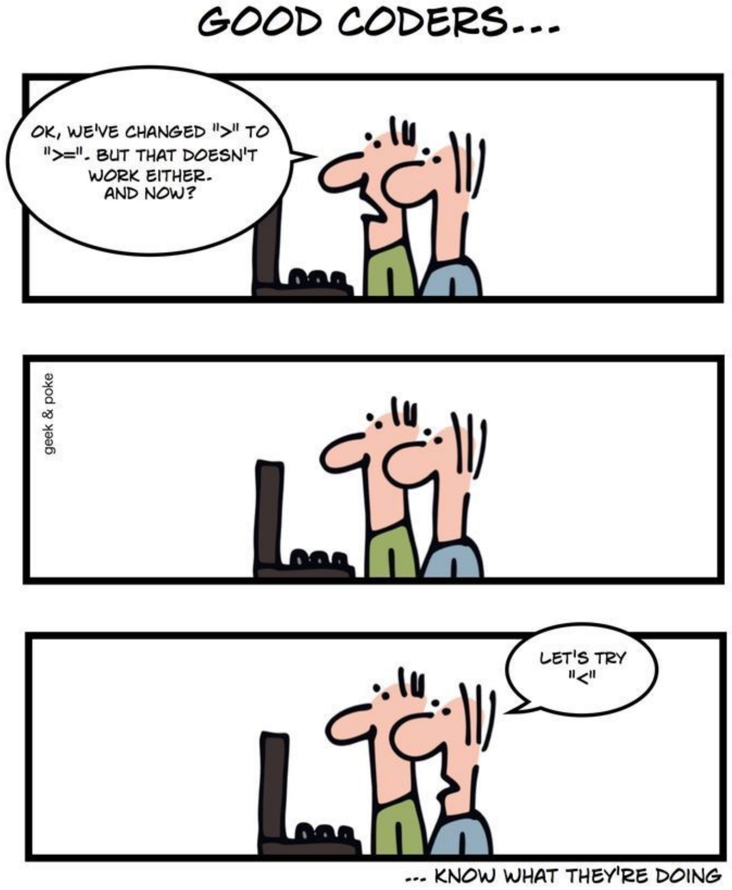
- No recommendation letters

- No project with me for your stay here

- It brings out my bad side, which I am not sure even I like.
    - You definitely will NOT like it.

General Remarks

- This will be a hard class, advised to not take it with that i mind
- Help me help you “Make an A”.
  - Not my words, but I like them
- Ask for help, come to office hours
  - Read the material
- I am perfectly fine not covering everything
  - Rather cover a small portion and make sure things are understood
- Pair programming is highly encouraged


GOOD CODERS...
OK, WE'VE CHANGED ">" TO ">=". BUT THAT DOESN'T WORK EITHER. AND NOW?
geek & poke
LET'S TRY "<"
... KNOW WHAT THEY'RE DOING

Next Steps

- Sign up for a partner, asap

- Import the VM image, and start playing around with it
    - Get comfortable with the shell
    - Get comfortable with basic Linux commands