Roadmap

C:

```
car *c = malloc(sizeof(car));
c->miles = 100;
c->gals = 17;
float mpg = get_mpg(c);
free(c);
```

Java:

```
Car c = new Car();
c.setMiles(100);
c.setGals(17);
float mpg =
    c.getMPG();
```

Assembly language:

```
get_mpg:
    pushq   %rbp
    movq    %rsp, %rbp
    ...
    popq    %rbp
    ret
```

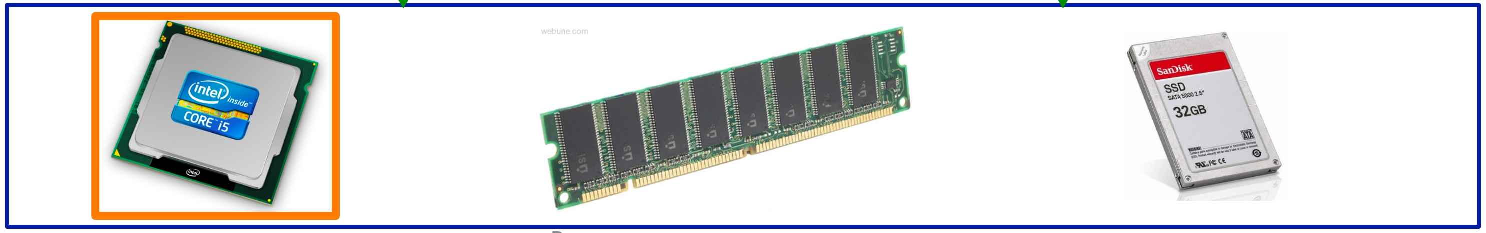
Machine code:

```
0111010000011000
100011010000010000000010
1000100111000010
110000011111101000011111
```

OS:

Windows 8 Mac

Computer system:



Section 8: Processes

- What is a process
- Creating processes
- Fork-Exec

What is a process?

- Why are we learning about processes?
  - Processes are another *abstraction* in our computer system – the process abstraction provides an *interface* between the program and the underlying CPU + memory.
- What do processes have to do with exceptional control flow?
  - Exceptional control flow is the mechanism that the OS uses to enable multiple processes to run on the same system.

- What is a program? A processor? A process?

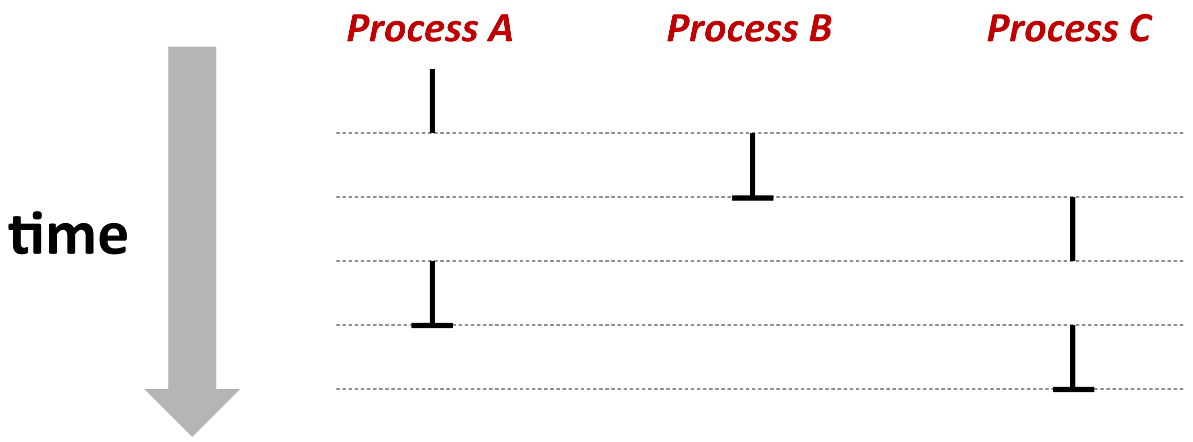
Processes

- **Definition: A *process* is an instance of a running program**
  - One of the most important ideas in computer science
  - Not the same as “program” or “processor”

- **Process provides each program with two key abstractions:**
  - Logical control flow
    - Each process seems to have exclusive use of the CPU
  - Private virtual address space
    - Each process seems to have exclusive use of main memory
- **Why are these illusions important?**
- **How are these illusions maintained?**
  - Process executions interleaved (multi-tasking)
  - Address spaces managed by virtual memory system – next course topic

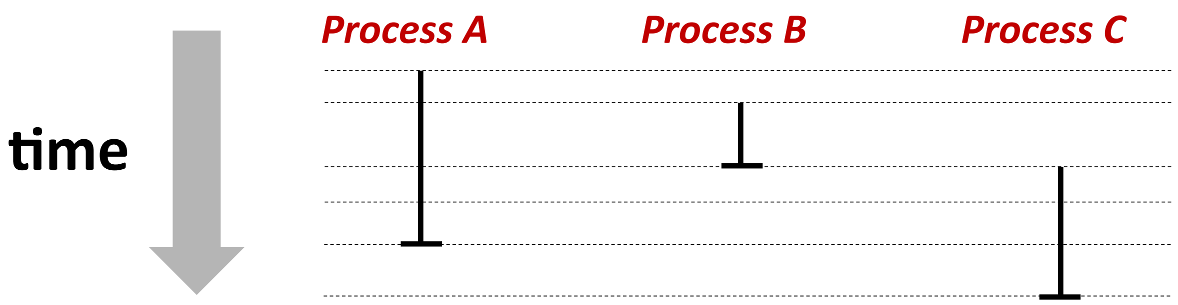
Concurrent Processes

- **Two processes *run* *concurrently* (are concurrent) if their instruction executions (flows) overlap in time**
- **Otherwise, they are *sequential***
- **Examples:**
  - Concurrent: A & B, A & C
  - Sequential: B & C



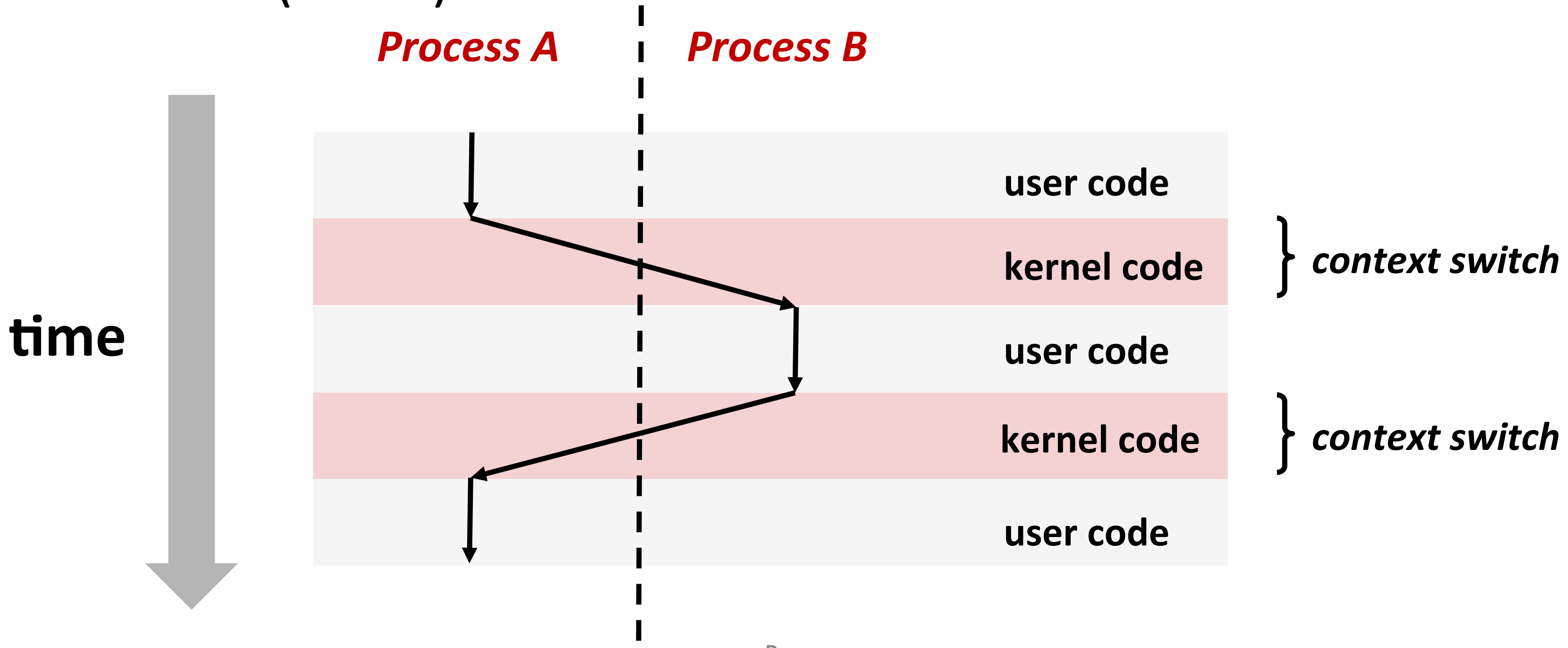
User View of Concurrent Processes

- **Control flows for concurrent processes are physically disjoint in time**
  - CPU only executes instructions for one process at a time
- **However, we can think of concurrent processes as executing in parallel**



Context Switching

- **Processes are managed by a shared chunk of OS code called the *kernel***
  - Important: the kernel is not a separate process, but rather runs as part of a user process
- **Control flow passes from one process to another via a *context switch…*** (how?)


Process A
Process B
time
user code
kernel code
user code
kernel code
user code
context switch
context switch