Exceptional Control Flow: Exceptions and Processes

Processes

These slides adapted from materials provided by the textbook authors.

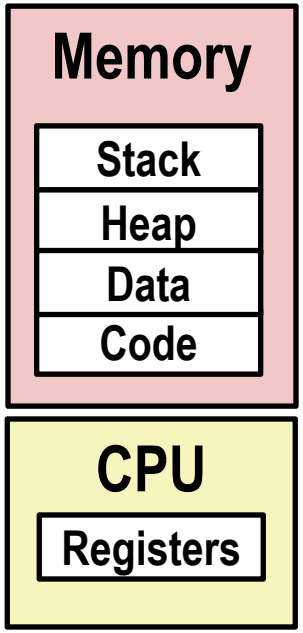
Exceptional Control Flow

- Exceptional Control Flow
- Exceptions
- **Processes**
- Process Control

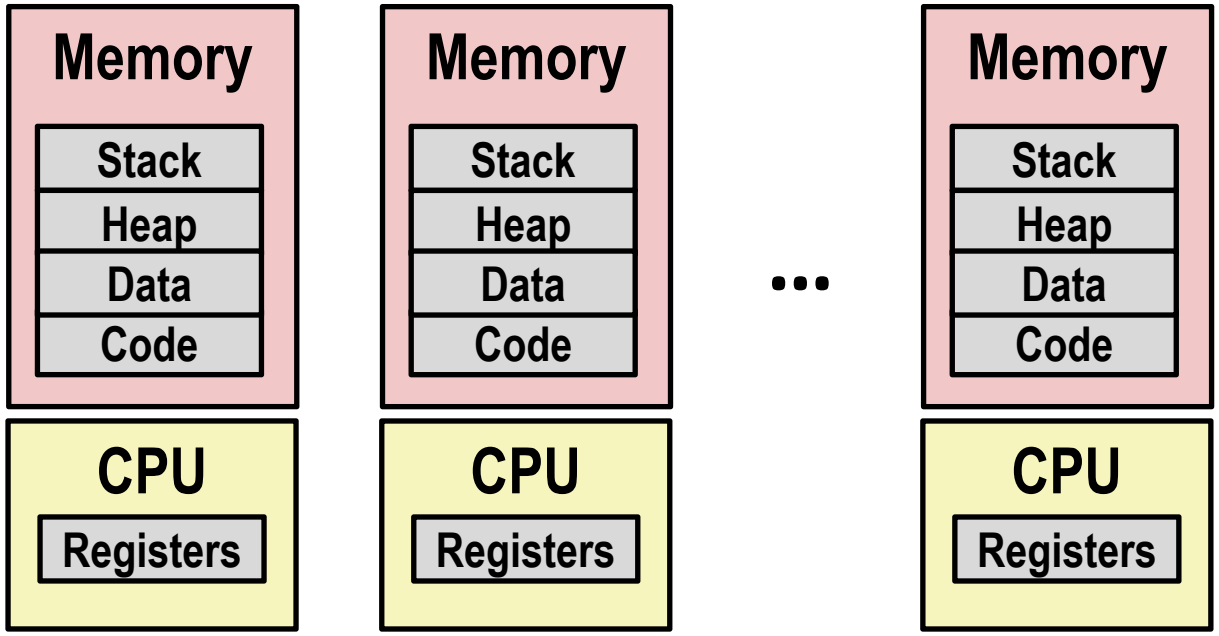
Processes

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

- **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 Illusion



- **Computer runs many processes simultaneously**
  - Applications for one or more users
    - Web browsers, email clients, editors, …
  - Background tasks
    - Monitoring network & I/O devices

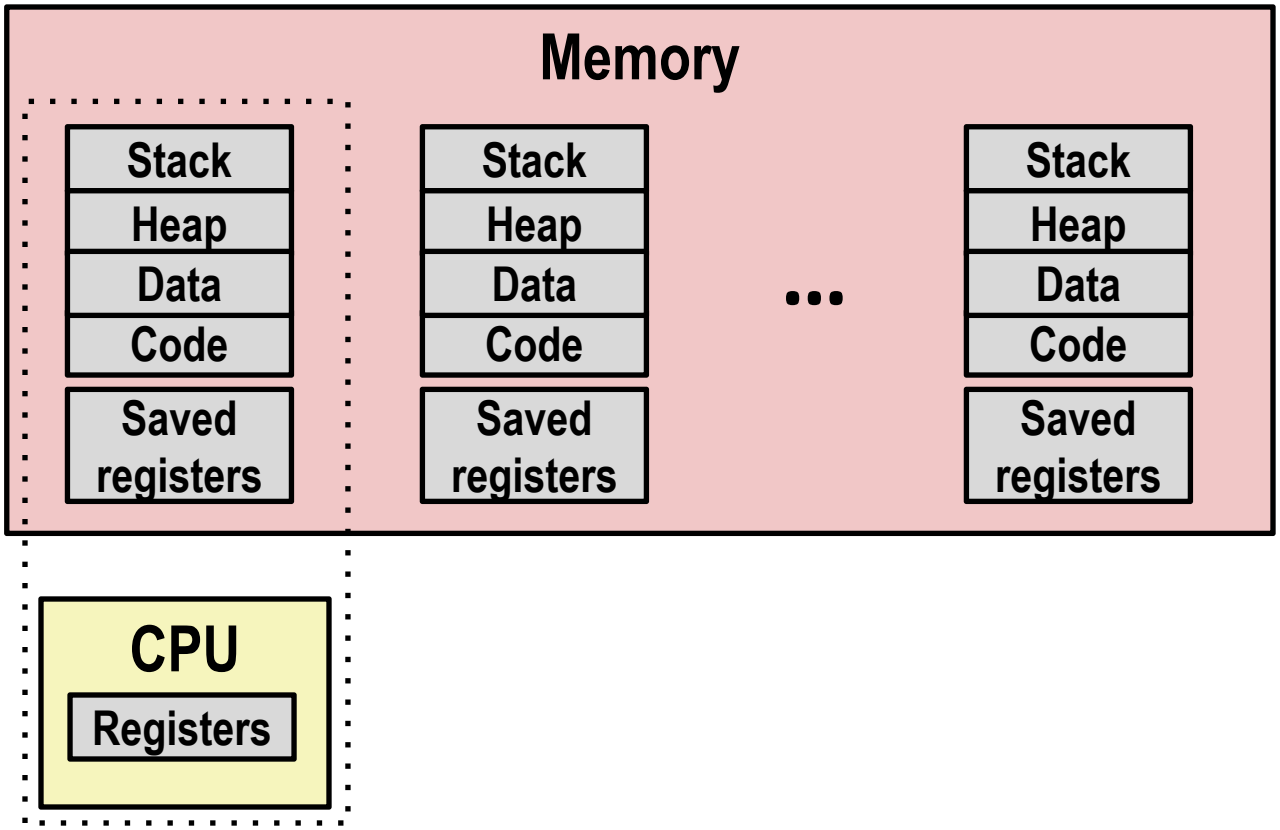
Multiprocessing Example

```
xterm
Processes: 123 total, 5 running, 9 stuck, 109 sleeping, 611 threads                11:47:07
Load Avg: 1.03, 1.13, 1.14  CPU usage: 3.27% user, 5.15% sys, 91.56% idle
SharedLibs: 576K resident, 0B data, 0B linkedit.
MemRegions: 27958 total, 1127M resident, 35M private, 494M shared.
PhysMem: 1039M wired, 1974M active, 1062M inactive, 4076M used, 18M free.
VM: 280G vsize, 1091M framework vsize, 23075213(1) pageins, 5843367(0) pageouts.
Networks: packets: 41046228/11G in, 66083096/77G out.
Disks: 17874391/349G read, 12847373/594G written.

PID    COMMAND      %CPU TIME     #TH   #WQ  #PORT #MREG RPRVT  RSHRD  RSIZE  VPRVT  VSIZE
99217- Microsoft Of 0.0  02:28.34 4     1    202   418   21M    24M    21M    66M    763M
99051  usbmuxd      0.0  00:04.10 3     1    47    66    436K   216K   480K   60M    2422M
99006  iTunesHelper 0.0  00:01.23 2     1    55    78    728K   3124K  1124K  43M    2429M
84286  bash         0.0  00:00.11 1     0    20    24    224K   732K   484K   17M    2378M
84285  xterm        0.0  00:00.83 1     0    32    73    656K   872K   692K   9728K  2382M
55939- Microsoft Ex 0.3  21:58.97 10    3    360   954   16M    65M    46M    114M   1057M
54751  sleep        0.0  00:00.00 1     0    17    20    92K    212K   360K   9632K  2370M
54739  launchdadd   0.0  00:00.00 2     1    33    50    488K   220K   1736K  48M    2409M
54737  top          6.5  00:02.53 1/1   0    30    29    1416K  216K   2124K  17M    2378M
54719  automountd   0.0  00:00.02 7     1    53    64    860K   216K   2184K  53M    2413M
54701  ocspd        0.0  00:00.05 4     1    61    54    1268K  2644K  3132K  50M    2426M
54661  Grab         0.6  00:02.75 6     3    222+  389+  15M+   26M+   40M+   75M+   2556M+
54659  cookied      0.0  00:00.15 2     1    40    61    3316K  224K   4088K  42M    2411M
53818  mdworker     0.0  00:01.67 4     1    52    91    7628K  7412K  16M    48M    2438M
50878  mdworker     0.0  00:11.17 3     1    53    91    2464K  6148K  9976K  44M    2434M
50410  xterm        0.0  00:00.13 1     0    32    73    280K   872K   532K   9700K  2382M
50078  emacs        0.0  00:06.70 1     0    20    35    52K    216K   88K    18M    2392M
```

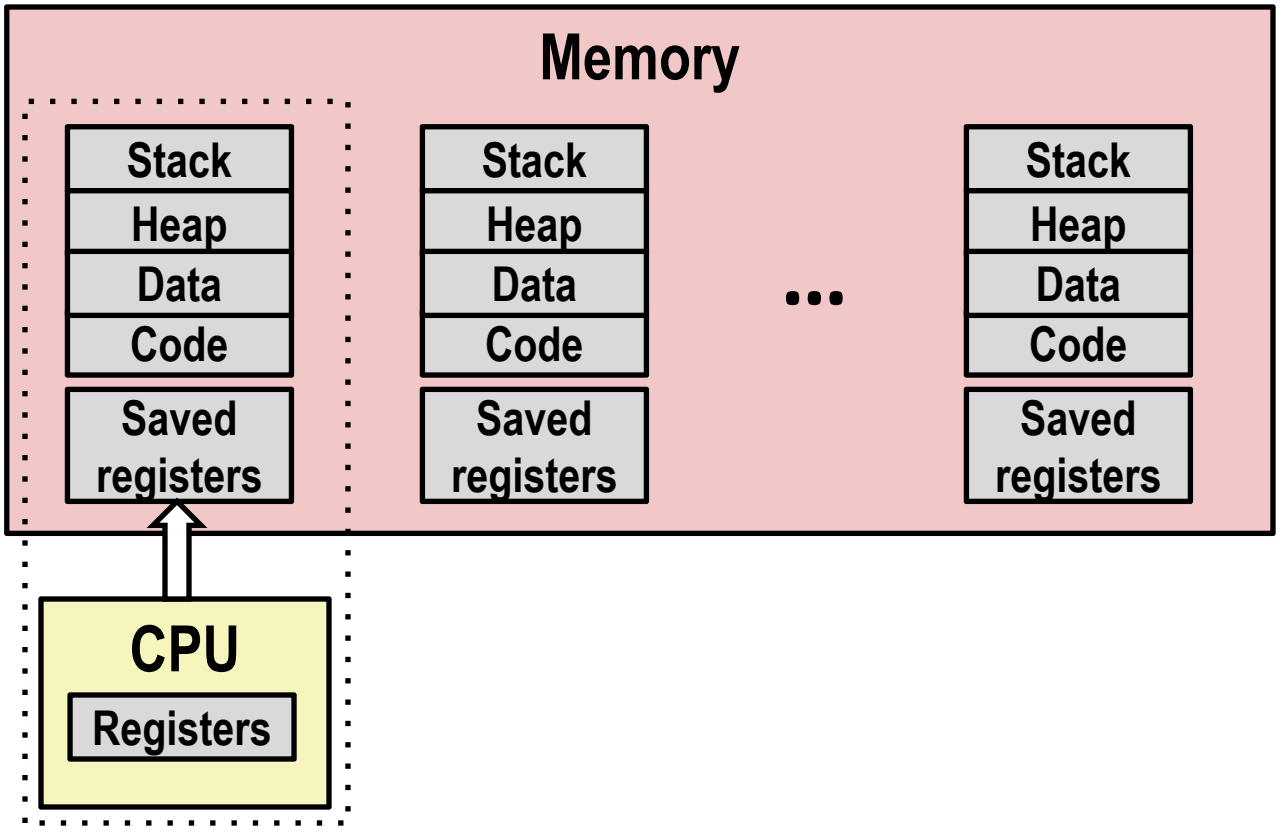
- **Running program "top" on Mac**
  - System has 123 processes, 5 of which are active
  - Identified by Process ID (PID)

Multiprocessing: The (Traditional) Reality



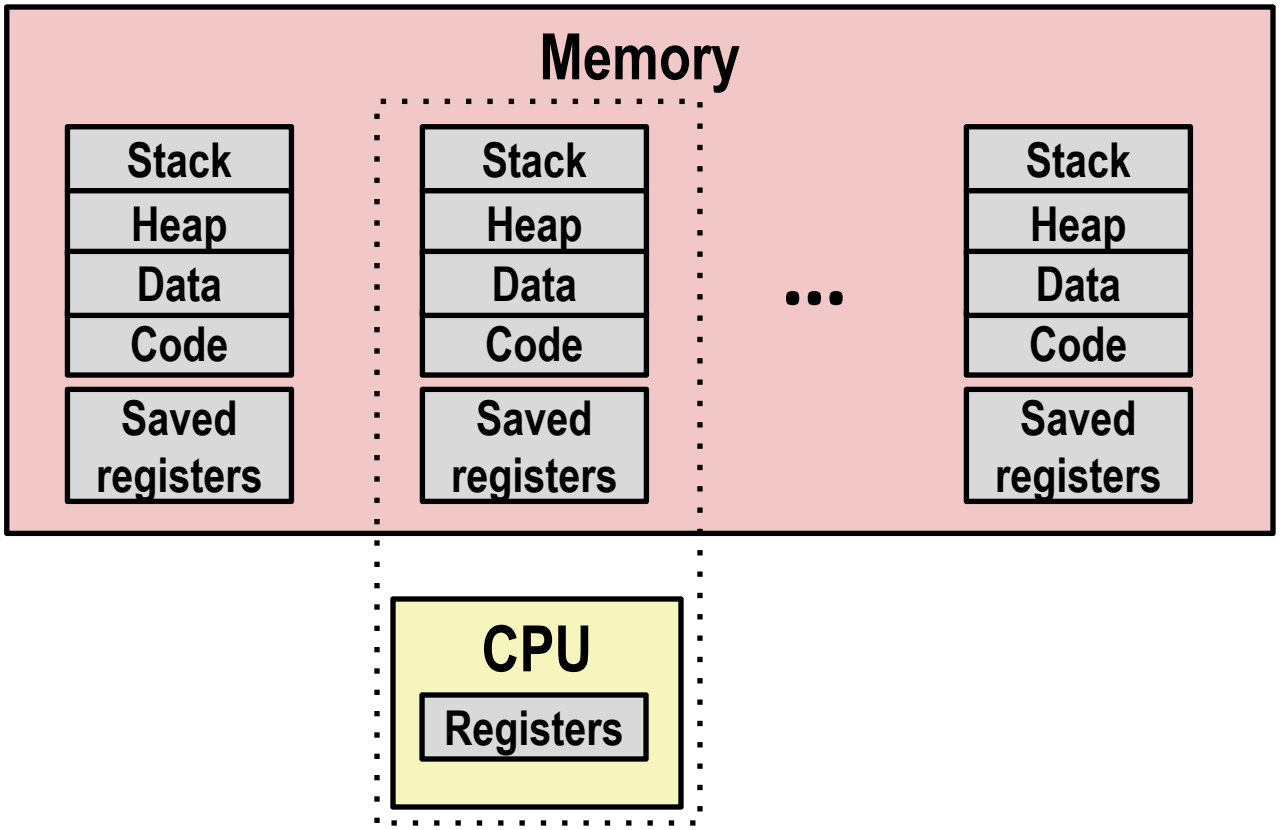
- **Single processor executes multiple processes concurrently**
  - Process executions interleaved (multitasking)
  - Address spaces managed by virtual memory system (later in course)
  - Register values for nonexecuting processes saved in memory

Multiprocessing: The (Traditional) Reality



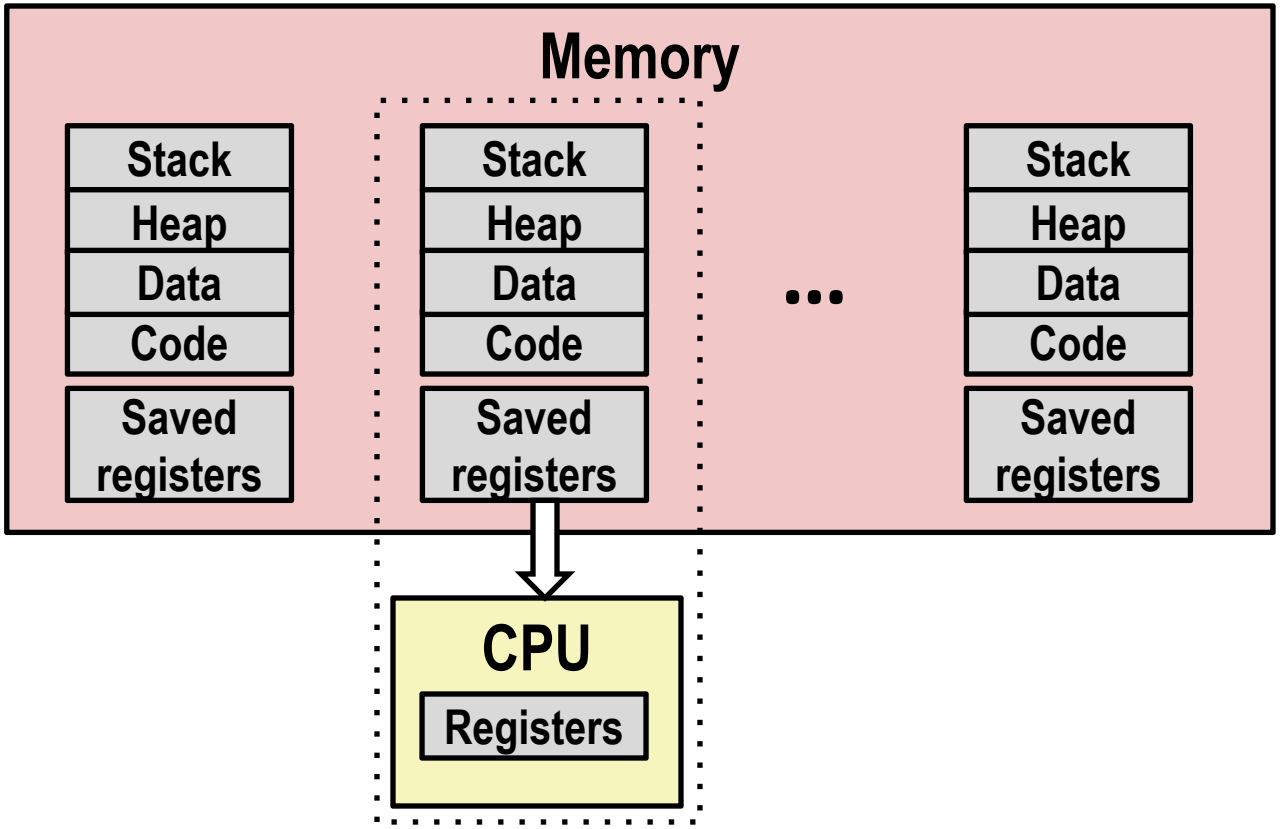
- Save current registers in memory

Multiprocessing: The (Traditional) Reality



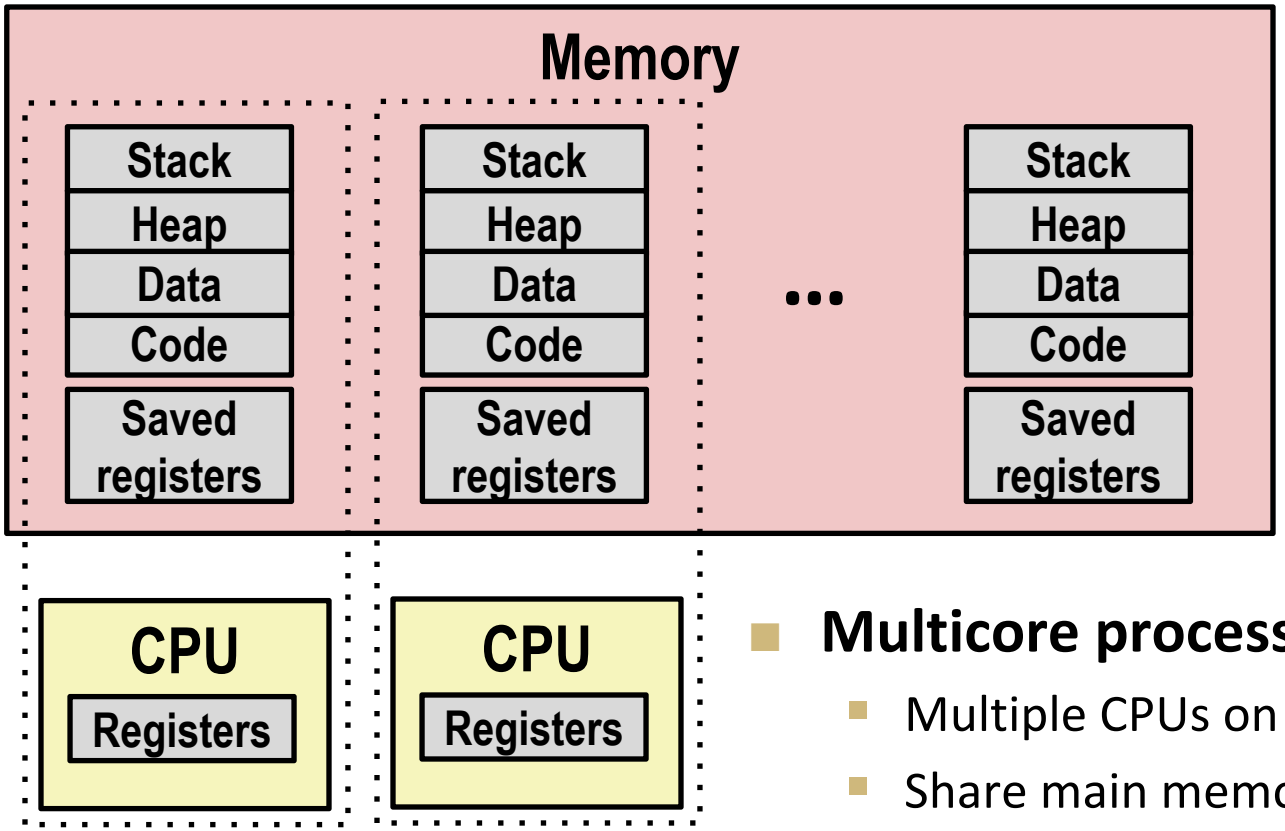
- Schedule next process for execution

Multiprocessing: The (Traditional) Reality



- **Load saved registers and switch address space (context switch)**

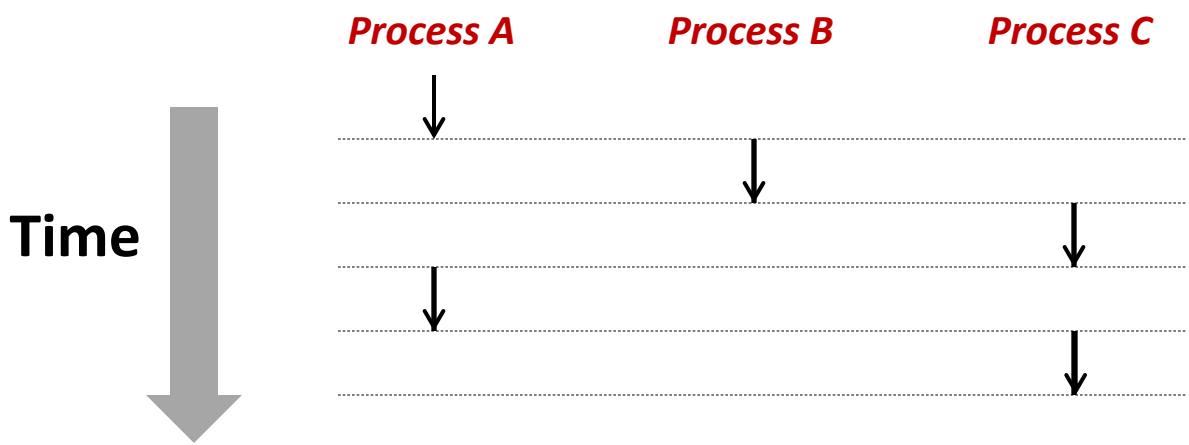
Multiprocessing: The (Modern) Reality



- **Multicore processors**
  - Multiple CPUs on single chip
  - Share main memory (and some of the caches)
  - Each can execute a separate process
    - Scheduling of processors onto cores done by kernel

Concurrent Processes

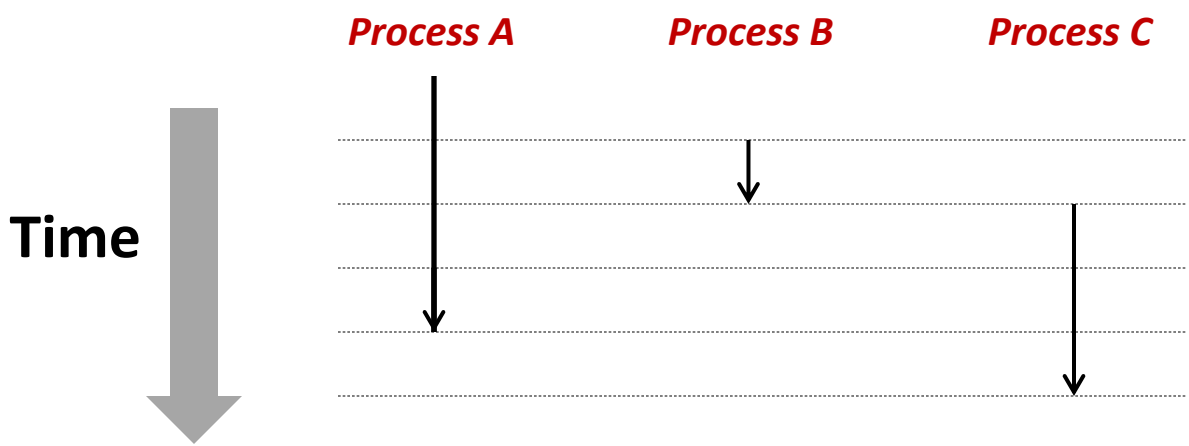
- **Each process is a logical control flow.**
- **Two processes *run concurrently* (*are concurrent)* if their flows overlap in time**
- **Otherwise, they are *sequential***
- **Examples (running on single core):**
  - Concurrent: A & B, A & C
  - Sequential: B & C



User View of Concurrent Processes

- Control flows for concurrent processes are physically disjoint in time

- However, we can think of concurrent processes as running in parallel with each other



Context Switching

- **Processes are managed by a shared chunk of memory-resident OS code called the *kernel***
  - Important: the kernel is not a separate process, but rather runs as part of some existing process.
- **Control flow passes from one process to another via a *context switch***