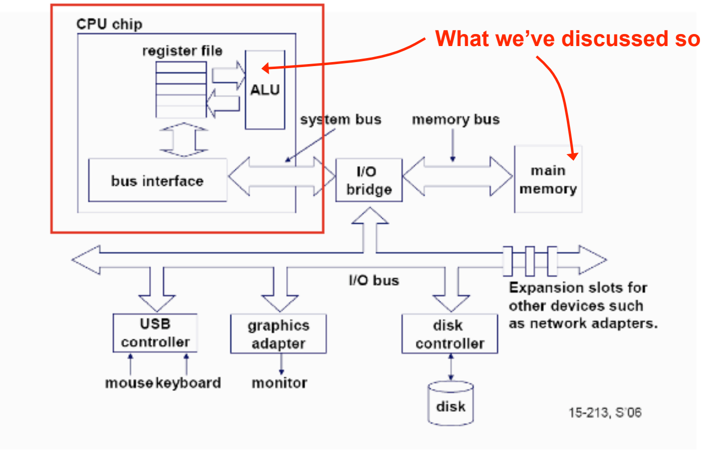
A short introduction to multi-core, threads

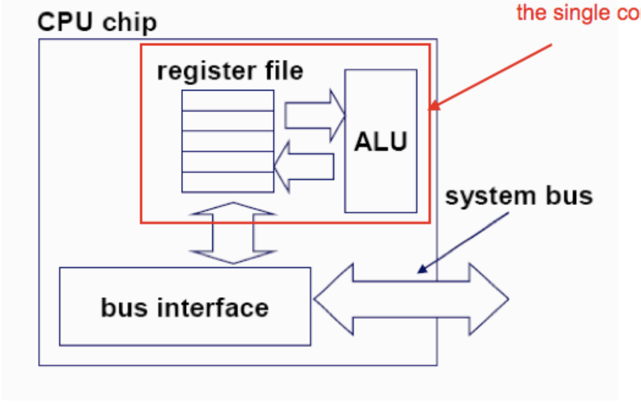
(Adapted from Jernej Barbic)

Single-core computer



What we've discussed so far.

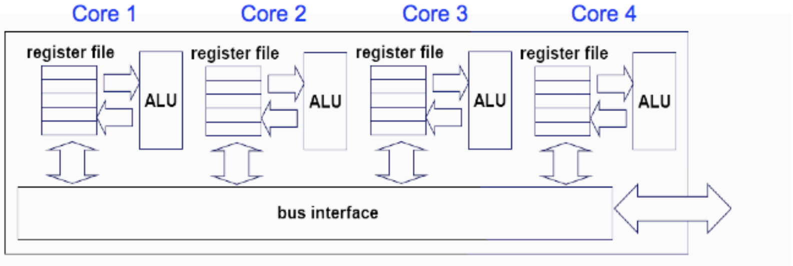
Single-core CPU chip



the single core

Multi-core architectures

- This lecture is about a new trend in computer architecture: Replicate multiple processor cores on a single die.

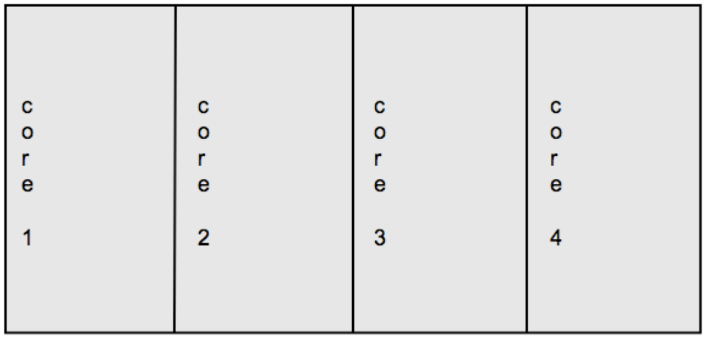


Multi-core CPU chip

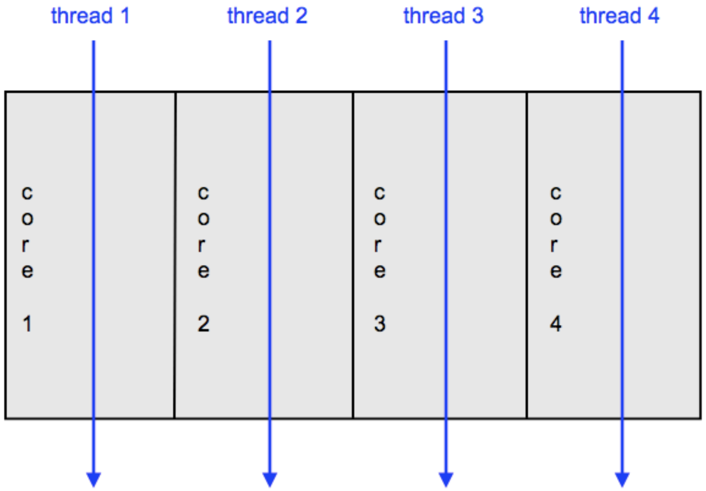
In lab, you'll see that we can do things in parallel, but you must pay cost of interconnection overhead.

Multi-core CPU chip

- The cores fit on a single processor socket
- Also called CMP (Chip Multi-Processor)



The cores run in parallel



Threads vs. Processes

- Process:
  - Full blown virtual machine
  - Has:
    - PC, register state,
    - In Memory:
      - stack, code, data, page table, etc.
  - Process context switch = lots of work
    - (1000s of ns)
- Thread:
  - Multiplexed CPU only

What’s a thread?

- Basic unit of CPU utilization
- Can think of a light weight process
- Consists of:
  - Program counter, register set, stack space
  - But…
    - Shares code, data, OS resources (open files, etc.) with peer threads

Interest in threads…

- Interest in threads originally stems from evolution of *Symmetric Multiprocessors*
  - – (SMP = fancy name for > 1 CPU)
- SMP:
  - – Idea became popular 6-7 years ago
    - Dual processor SMP more cost effective than 2 uniprocessor boxes
      - – Dell Workstation 2.4 GHz Intel Xeon = $2584
      - – 2nd processor = just an additional $434
  - – Multiple CPUs in a single box sharing memory, I/O resources

CMP a lot like SMP

- CMP = "chip multiprocessor"
  - – Another fancy name for multi-core
  - – Now, technically possible, economical, and physically necessary to put multiple cores on 1 chip

Why are CMPs "physically necessary"?

- Idea:
  - – 2 cores, 2 GHz lead to heat of X
  - – 1 core, 4 GHz leads to heat of 4X
- From performance perspective
  - – If you can parallelize code, execution time on machine 1 is the same as machine 2
- From heat perspective
  - – Heat of machine 1 is less than machine 2

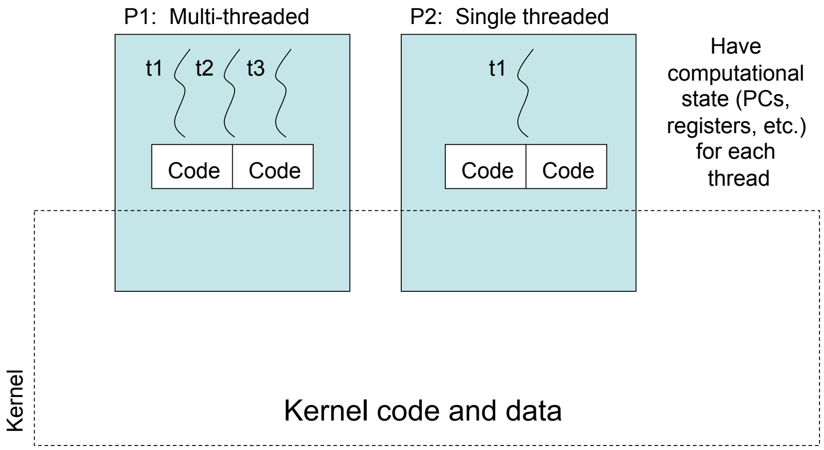
See Lecture 01 slides…

Context switches

- Threads more easily context switched
  - Just registers + PC, not memory management
  - Could run on different processors concurrently in SMP
    - Share CPU in uniprocessor
    - Note: Brings up synchronization issues
      - (Not covered until CSE 30322)

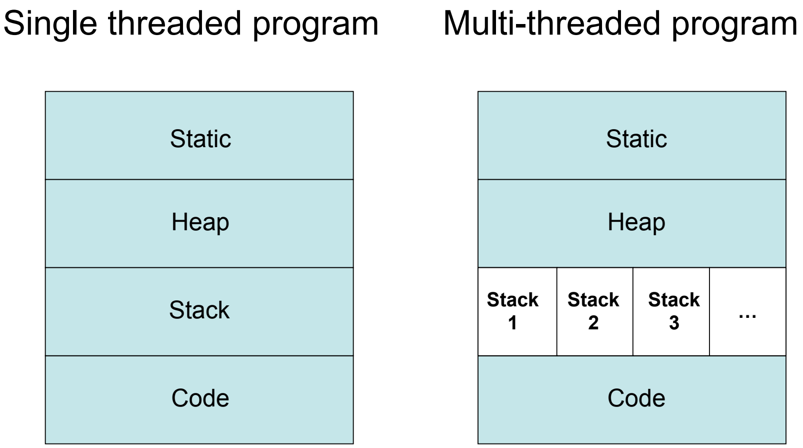
Example

- Two single threaded applications on 1 machine



Have computational state (PCs, registers, etc.) for each thread

Another view



MT program has per thread stack:
Heap, static, and code are common to all threads

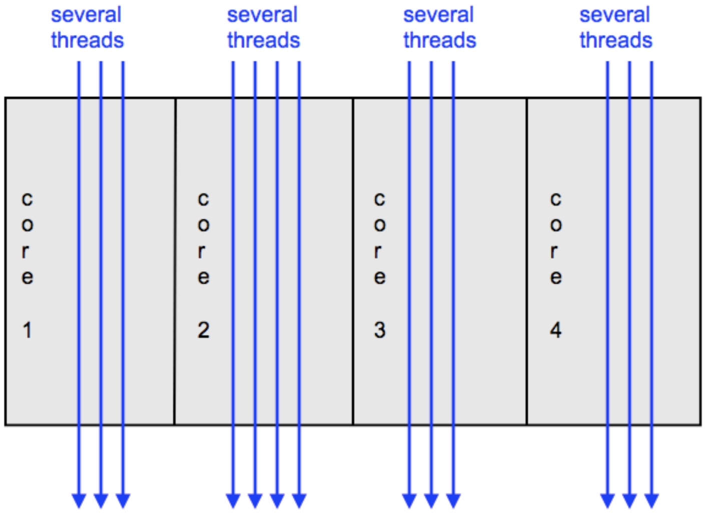
Threads and the OS

- Programs in traditional OS are single threaded
  - 1 PC per program (process), 1 stack, 1 set of CPU registers
  - If process blocks (i.e. disk I/O, NW communication, etc.) then no progress for program as a whole
    - E.g. from a lw instruction…

Multi-threaded OS

- Examples:
  - Digital unix, Free BSD, Sun Solaris, etc.
- State:
  - In ST state contained in a process
  - In MT, state contained in multiple threads
- Idea:
  - w/MT, if one thread of execution blocks, can switch to another one without a context switch

Within each core, threads are time-sliced (just like on a uniprocessor)



Interaction with OS

- OS perceives each core as a separate processor
- OS scheduler maps threads/processes to different cores
- Most major OS support multi-core today

Why multi-core ?

- Difficult to make single-core clock frequencies even higher
- Deeply pipelined circuits:
  - heat problems
  - speed of light problems
  - difficult design and verification
  - large design teams necessary
  - server farms need expensive air-conditioning
- Many new applications are multithreaded
- General trend in computer architecture (shift towards more parallelism)

Instruction-level parallelism

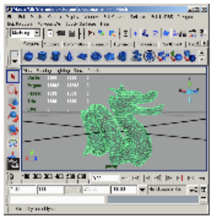
- Parallelism at the machine-instruction level
- The processor can re-order, pipeline instructions, split them into microinstructions, do aggressive branch prediction, etc.
- Instruction-level parallelism enabled rapid increases in processor speeds over the last 15 years

Thread-level parallelism (TLP)

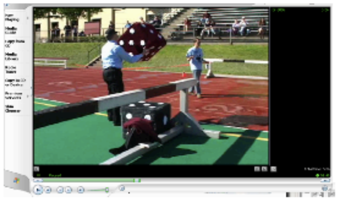
- This is parallelism on a more coarser scale
- Server can serve each client in a separate thread (Web server, database server)
- A computer game can do AI, graphics, and physics in three separate threads
- Single-core superscalar processors cannot fully exploit TLP
- Multi-core architectures are the next step in processor evolution: explicitly exploiting TLP

What applications benefit from multi-core?

- Database servers
- Web servers (Web commerce)
- Compilers
- Multimedia applications
- Scientific applications, CAD/CAM
- In general, applications with *Thread-level parallelism* (as opposed to instruction-level parallelism)



Each can run on its own core



More examples

- Editing a photo while recording a TV show through a digital video recorder
- Downloading software while running an anti-virus program
- “Anything that can be threaded today will map efficiently to multi-core”
- BUT: some applications difficult to parallelize