CS 50: Introduction to Computer Science I
Harvard College
Fall 2007

Scribe Notes
Week 4 Monday
Anjuli Kannan

Announcements (0:00-5:00)

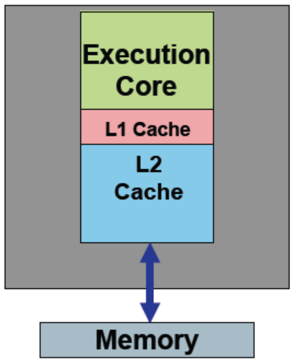
- More HCS seminars coming up. Check website.
- If you're looking for something to do for your final project, you can get involved in restructuring the house facebook and website system. Email zstone@fas.harvard.edu for details.
- Guest lecture today from Professor Margo Seltzer!

Let's Talk about Hardware (5:00-13:00)

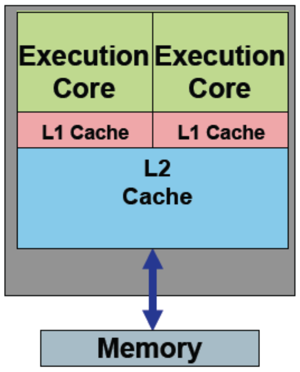
- Moore's Law: The number of transistors that can be placed on a chip doubles approximately every two years
- This was stated in 1965 by Gordon Moore, CEO of Intel
- Unlike a lot of technology-related predictions made that long ago, Moore's Law has been pretty accurate
- So what? What does it mean if we can fit more transistors on a chip?
  - More transistors means smaller transistors
  - Smaller transistors means less space between them
  - Less space between them means greater speed
- For a long time, the clock frequency of processors was getting faster, and then it leveled off
- Chip performance is still improving
- How are we going to keep up the performance of chips? Maybe we can use all these transistors somehow…

Multicore Processors (13:00-18:00)

- Here is what a typical processor looks like:



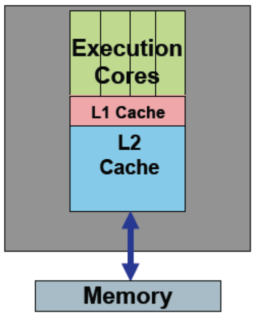
- Caches are pieces of memory arranged in a hierarchy
- Memory on the chip (in caches) runs close to same speed of processor
- Memory on the hard drive much slower to access
- So what do we do with all our extra transistors? Put two cores on the chip.
- Here is a dual core processor:



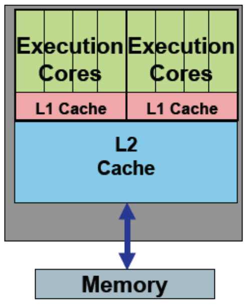
- We have two execution engines, each with an L1 cache and a shared L2 cache.
- Intuitively, more work can get done because we can do two things at once
- But what if they both need memory that's not in their L1? Two processors fight for access to L2.
- So the improvement in performance depends on what we're doing
  - If we need a lot of memory out of the hard drive, we certainly won't be getting twice the performance
  - But if we everything we need is in L1, we can get pretty close to twice the performance
- What else can we do with our transistor budget?

Multithreaded Processors (18:00-23:00)

- Instead of duplicating the execution core (and L1), put bunch of different execution cores that share an L1 cache – not quite as much logic has to be duplicated



- Implications? Probably not all running as fast as they would individually since they're sharing L1.
- But if there's a delay getting to memory, it's hidden. Memory can be used more efficiently as requests of cores are juggled
- What else can we do?
- Put 'em together and you get multicore multithreaded processors!



- Today you can get a chip with 8 cores, each with 4 threads—that's 32 threads!
- Tomorrow we're going to have not 10 threads, not 32 threads, but 100 threads!!
- In five or seven years, there may be 1000 threads!!!!!
- This is fantastic!
- But wait…what are we going to use all these threads for?

Let's Talk about Software (23:00-29:00)

- Problem: single-thread performance isn't going to get any faster. In fact, it might slow down!
- Maybe we can do more things at once, and that should go faster because we won't have to trade between them. We call this parallelism.
- But how many things do you actually do? If you're really a multitasker (playing music, word processing, compiling code, Gchatting, etc.) *maybe* you do like 10 things at once. But what are you going to do with the other 990 cores?
- We need to take advantage of the fact that we can do a lot of things at the same time.
- Actually, people in the scientific community have already thought about this.
- Why? Because they do computations on huge datasets, like the human genome or the universe.
- Can we make use of their work thus far and apply it?
- Well, our datasets aren't large enough to take advantage of their work on parallelism.

What Does this Mean for You? (29:00-33:00)

- "Everything: the world is changing!"
- Computers are a growth industry.
- In order to keep this up, we need to solve this problem.
- Otherwise, computers are doomed to turn into a replacement industry, like frigerators!
- Nobody's writing parallel programs now, but in five years they're going to have to
- We need to think about this problem now:
    - How do you take large tasks and break them into small tasks that can be parallelized?
    - What language should we use?
- As a budding computer scientists, you will undoubtedly be affected by this problem.
- Indeed you may even play a role in solving it…