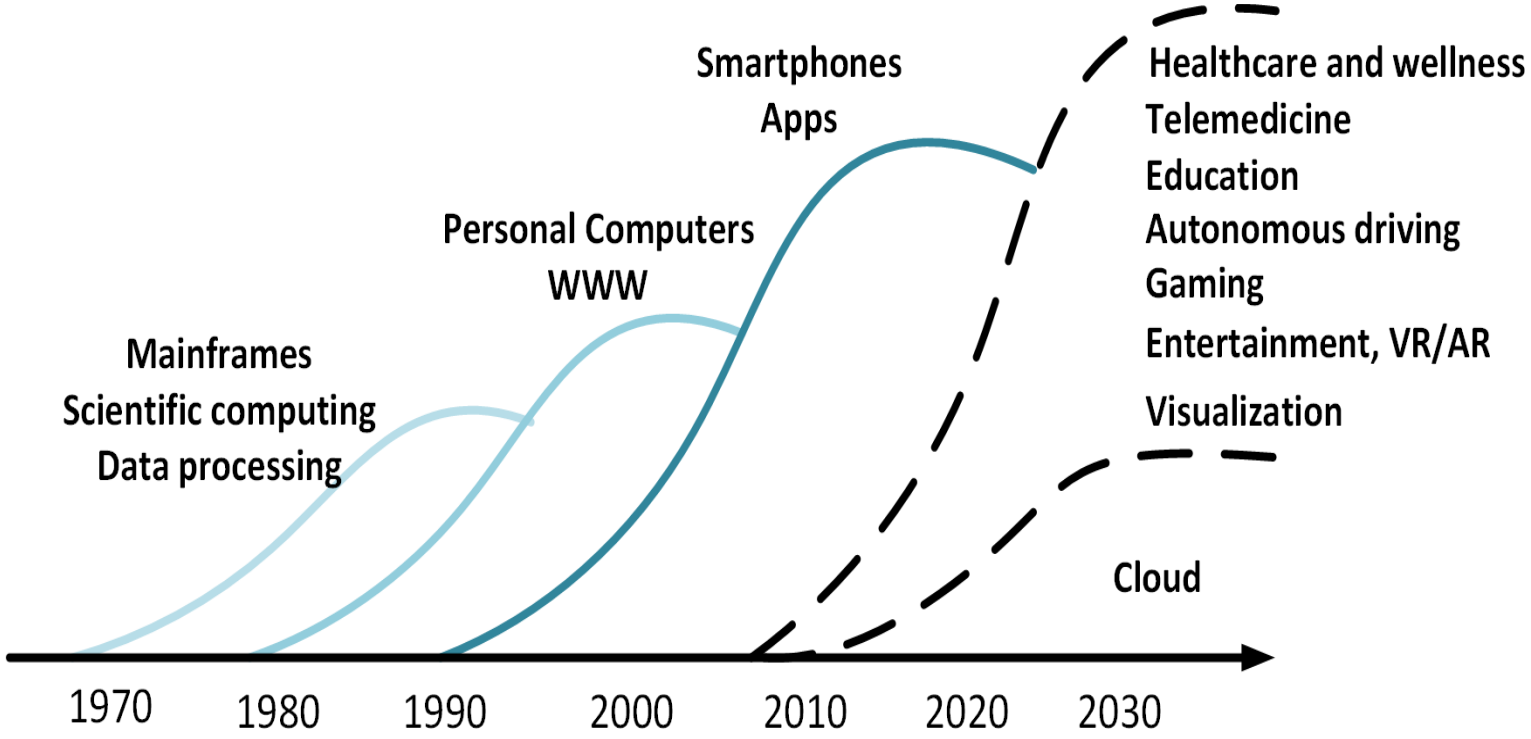
Introduction to Computer Systems

CMPT 295

http://xkcd.com/676/

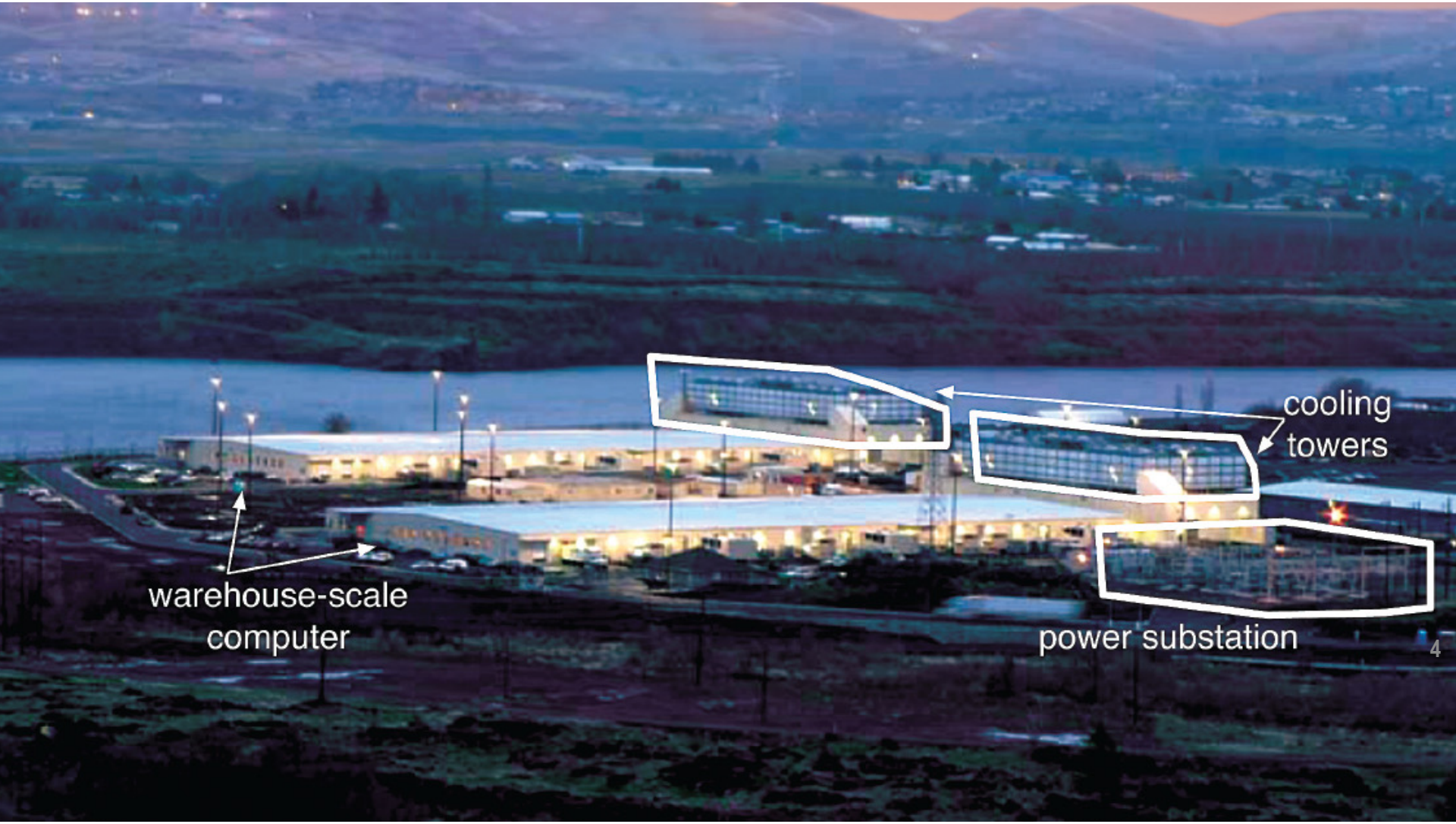
Why is computer systems exciting today?

- Number of deployed devices continues growing, but no single killer app
  - Diversification of needs, architectures



Personal Mobile Devices

Network Edge Devices


cooling
towers
warehouse-scale
computer
power substation


My other computer
is a data center


ALIENWARE
ALIENWARE
51

CMPT 295 is NOT about C Programming

- It is about the hardware-software interface
  - What does the programmer need to know to achieve the highest possible performance

- Languages like C are closer to the underlying hardware, unlike languages like Snap*!*, Python, Java
  - We can talk about hardware features in higher-level terms
  - Allows programmer to explicitly harness underlying hardware parallelism for high performance

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:

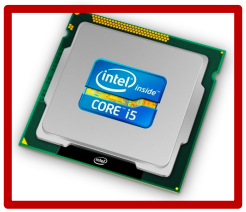
```
get_mpg(car*):
        lw      a5,0(a0)
        lw      a4,4(a0)
        divw    a5,a5,a4
        fcvt.s.w        fa0,a5
        ret
```

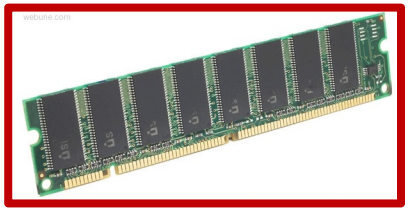
Machine code:

```
0111010000011000
10001101000010000000010
1000100111000010
11000001111110100011111
```

OS:

Computer system:





- Memory & data
- Arrays & structs
- Integers & floats
- RISC V assembly
- Procedures & stacks
- Executables
- Memory & caches
- Processor Pipeline
- Performance
- Parallelism

Course Perspective

- CMPT 295 will make you a better programmer
  - Purpose is to show how software really works
    - Understanding of some of the abstractions that exist between programs and the hardware they run on, why they exist, and how they build upon each other
  - Understanding the underlying system makes you more effective
    - Better debugging
    - **Better basis for evaluating performance**
    - How multiple activities work in concert (e.g. OS and user programs)
  - "Stuff everybody learns and uses and forgets not knowing"

- CMPT 295 presents a world-view that will empower you
  - The intellectual and software tools to understand the trillions+ of 1s and 0s that are "flying around" when your program runs

What is this class really about ?

4 Great Ideas in Computer Science

1. Layers of Abstraction
2. Locality/Memory Hierarchy
3. Parallelism
4. Performance Measurement

Great Idea #1: Abstraction (Levels of Representation/Interpretation)

C Program

```
int square(int num) {
return num * num;
}
```

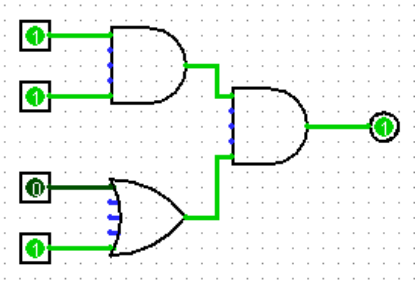
Binary

0x00000317
0x00830067
0xff010113
0x00112623
0x00812423
0x01010413
0xfea42a23
.....

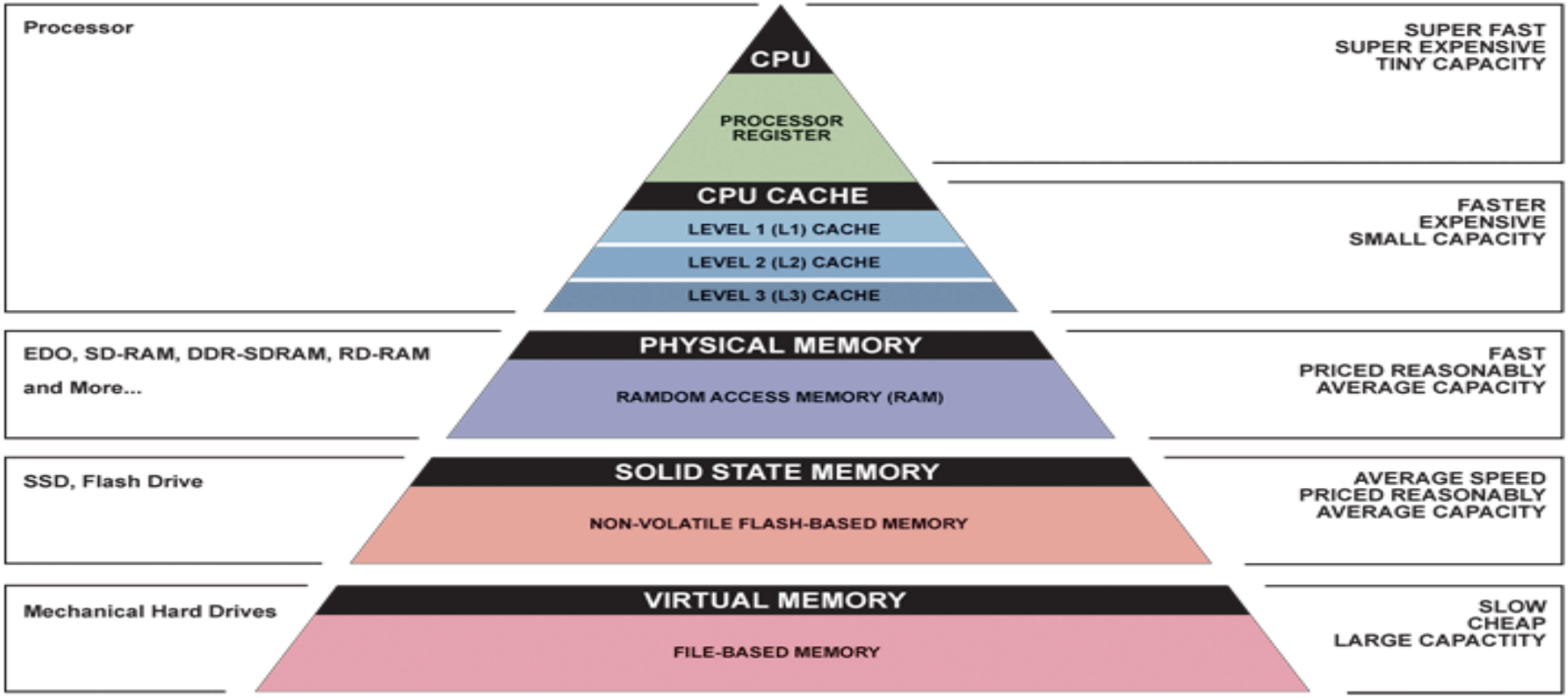
Assembly

```
square(int):
        addi    sp, sp, -16
        sw      ra, 12(sp)
        sw      s0, 8(sp)
        addi    s0, sp, 16
        sw      a0, -12(s0)
        lw      a0, -12(s0)
        mul     a0, a0, a0
        lw      s0, 8(sp)
        lw      ra, 12(sp)
        addi    sp, sp, 16
        ret
```

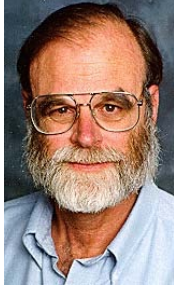
Logic



Great Idea #2: Principle of Locality/ Memory Hierarchy



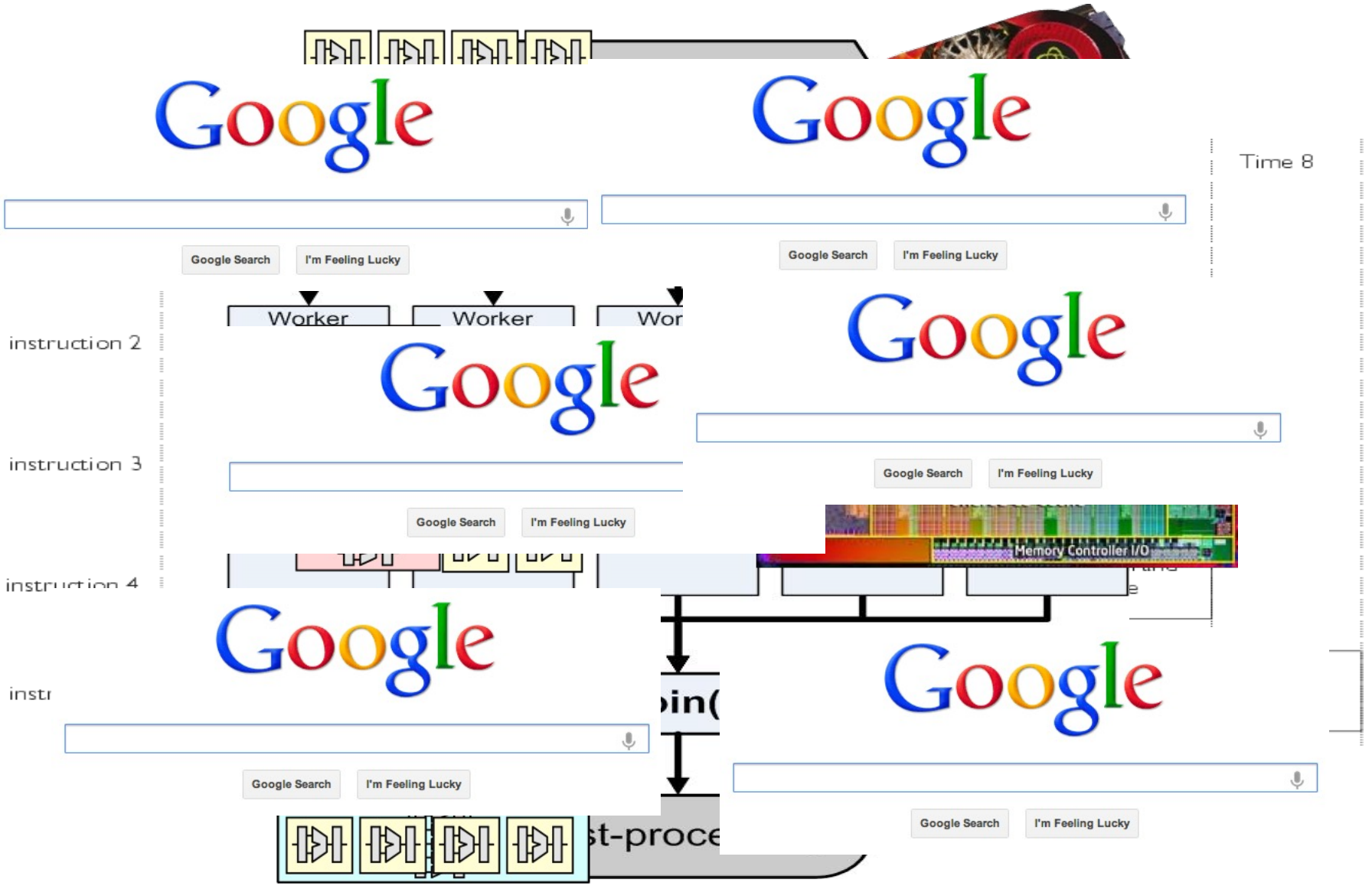
Jim Gray's Storage Latency Analogy: How Far Away is the Data?



Jim Gray
Turing Award

10^9	Tape /Optical Robot	Andromeda	2,000 Years
10^6	Disk	Pluto	2 Years
		Whistler	1.5 hr
100	Memory		
10	On Board Cache	This Campus	10 min
2	On Chip Cache	This Room	
(ns) 1	Registers	My Head	1 min

Great Idea #3: Parallelism



Great Idea #4: Performance Measurement and Improvement

- Matching application to underlying hardware to exploit:
  - Locality
  - Parallelism
  - Special hardware features, like specialized instructions (e.g., matrix manipulation)

- Latency
  - How long to set the problem up
  - How much faster does it execute once it gets going
  - It is all about *time to finish*

Bookmarks

- Course Website:
  http://cs.sfu.ca/~ashriram/Courses/CS295/
  - Schedule, policies, materials, videos, assignments, etc.

- Discussion in groups (link in course Website):
  - Announcements made here
  - Ask and answer questions – staff will monitor and contribute

- Github: Homework/Assignments and Labs

Textbooks

Essential books

A good C book

- All about Programming (Hilton and Bracy)
  - Lecture readings

- *Computer Architecture (RISC V edition)*
  - David Patterson John Hennessy
  - Lecture readings

Extra (some modules, which we will provide)

- *Computer Systems: A Programmer's Perspective*
  - Randal E. Bryant and David R. O'Hallaron
  - Some labs

My goal as an instructor

- To make your experience in CMPT 295 as enjoyable & informative as possible
  - Humor, enthusiasm & technology-in-the-news in lecture
  - Fun, challenging projects & HW
  - Pro-student policies (exam clobbering)

- I know I speak fast when I get excited about material. I'm told every semester. Help me slow when I go toooo fast.
  - Please give feedback so we can improve! We will listen!!

Grading (Online version specific)

- **Assignments/Homework:** 50% total
  - Autograded; *Individual work only*
- **Labs:** 2.5%
  - Graded by TAs
- **Exams:** Midterm (15%) and Final (30%)
- Weekly quiz participation: 2.5%
- EPA – 2% Extra credit. Read course page.

Tips for Success in 295

- Attend all lectures and sections
  - Avoid devices during lecture please
- Do the textbook readings ahead of time
- Learn by doing
  - Can answer many questions by writing small programs
- Visit piazza often
  - Ask questions and try to answer fellow students' questions
- Go to labs (required and count for grade)
- Find a study and homework group
- Start assignments early
- Don't be afraid to ask questions

Introductions: You!

- ~100-400 students registered, single lecture
- CS majors, and more
  - Most of you will find almost everything in the course new
- Get to know each other and help each other out!
  - Learning is much more fun with friends
  - Working well with others is a valuable life skill
  - Diversity of perspectives expands your horizons

Collaboration and Academic Integrity

- All submissions are expected to be yours and yours alone
- You are encouraged to discuss your assignments with other students (*ideas*), but we expect that what you turn in is yours
- It is NOT acceptable to copy solutions from other students or to copy (or start your) solutions from the Web (including Github)
- Our goal is that *YOU* learn the material so you will be prepared for exams, interviews, and the future

Some fun topics that we will touch on

- Which of the following seems the most interesting to you?

a) What is a GFLOP and why is it used in computer benchmarks?
b) How and why does running many programs for a long time eat into your memory (RAM)?
c) What is stack overflow and how does it happen?
d) Why does your computer slow down when you run out of *disk* space?
e) What is the meaning behind the different CPU specifications? (e.g. # of cores, # and size of cache, supported memory types)