Introduction to Computer Systems

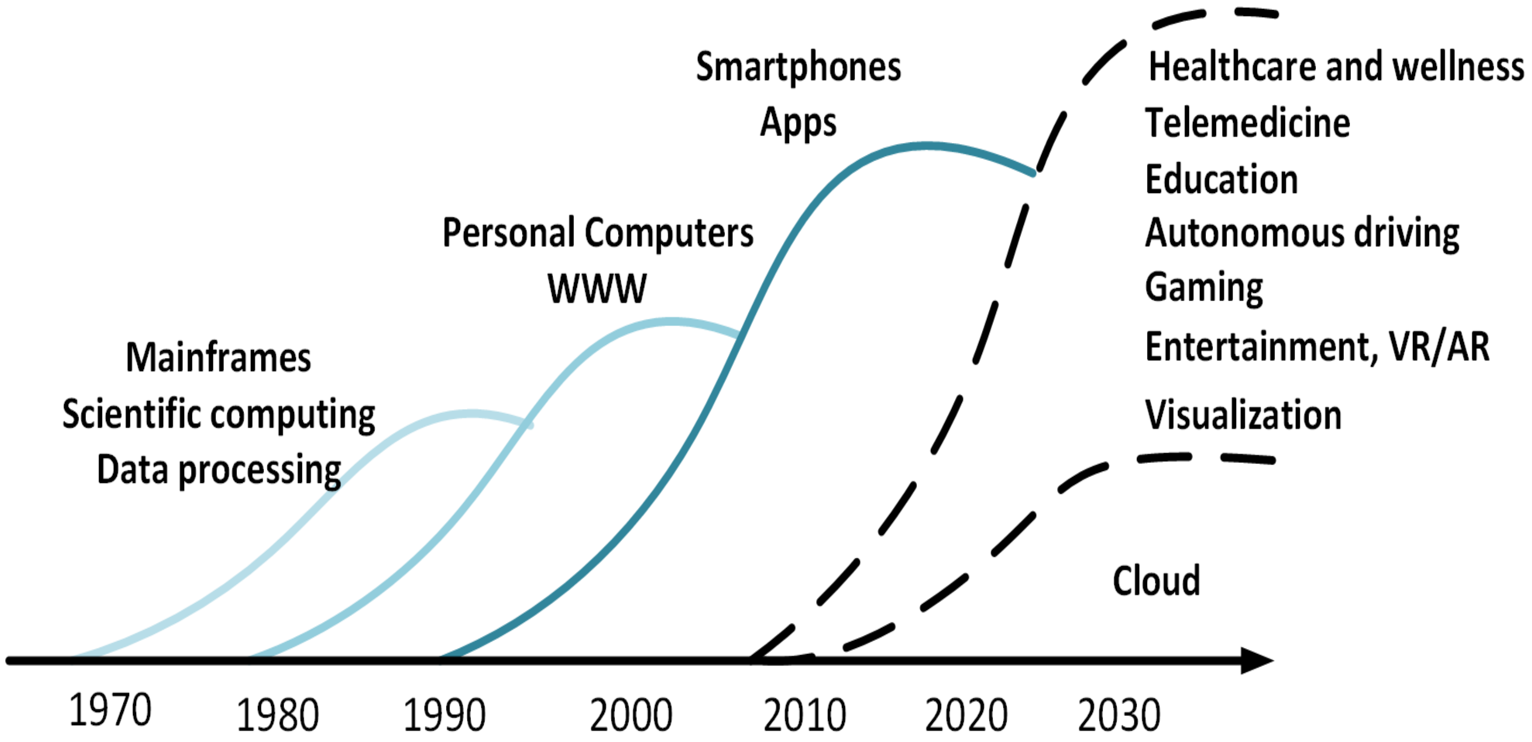
CMPT 295

AN x64 PROCESSOR IS SCREAMING ALONG AT BILLIONS OF CYCLES PER SECOND TO RUN THE XNU KERNEL, WHICH IS FRANTICALLY WORKING THROUGH ALL THE POSIX-SPECIFIED ABSTRACTION TO CREATE THE DARWIN SYSTEM UNDERLYING OS X, WHICH IN TURN IS STRAINING ITSELF TO RUN FIREFOX AND ITS GECKO RENDERER, WHICH CREATES A FLASH OBJECT WHICH RENDERS DOZENS OF VIDEO FRAMES EVERY SECOND
BECAUSE I WANTED TO SEE A CAT JUMP INTO A BOX AND FALL OVER.
I AM A GOD.

http://xkcd.com/676/

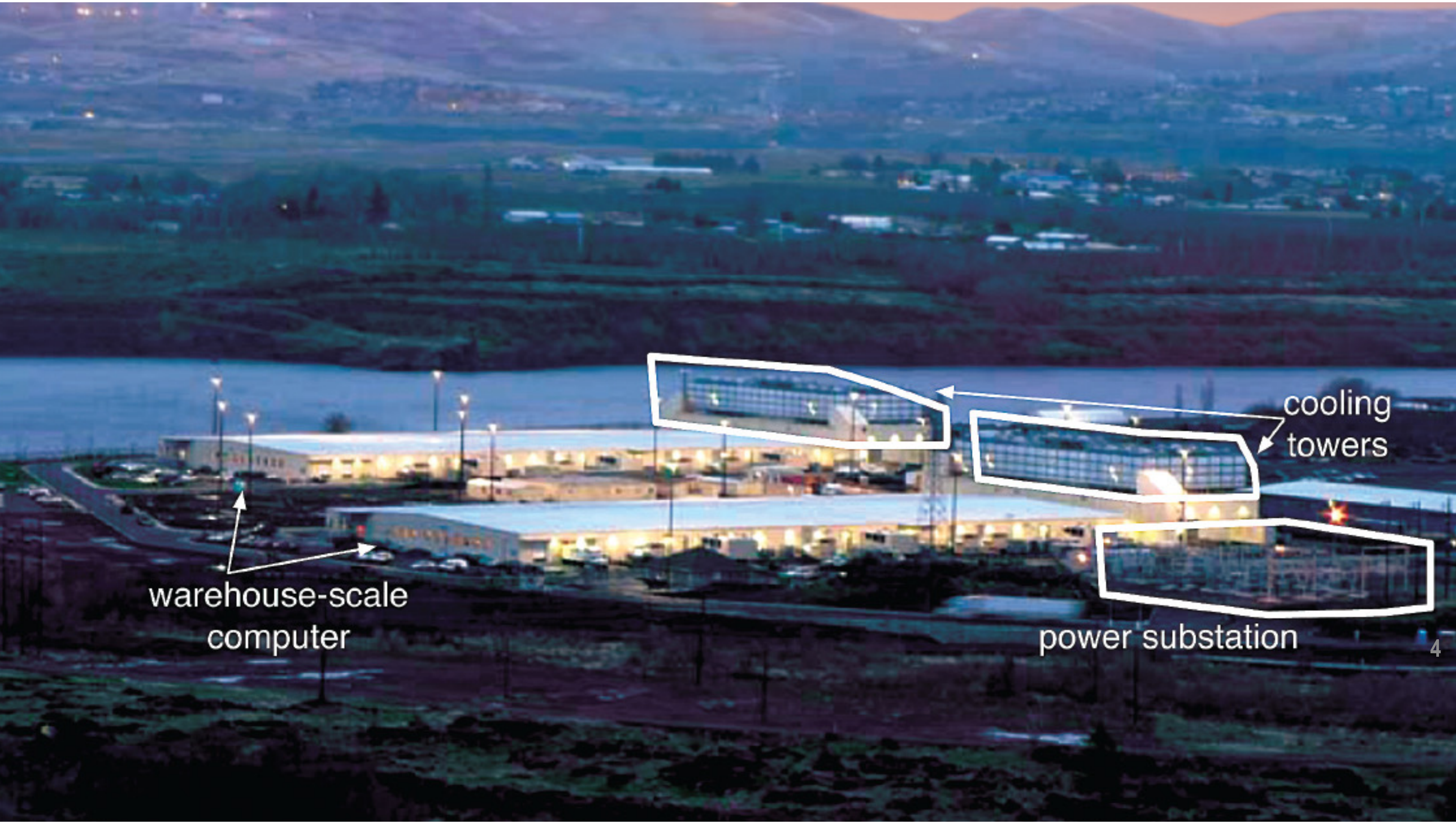
Why is computer systems exciting today?

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

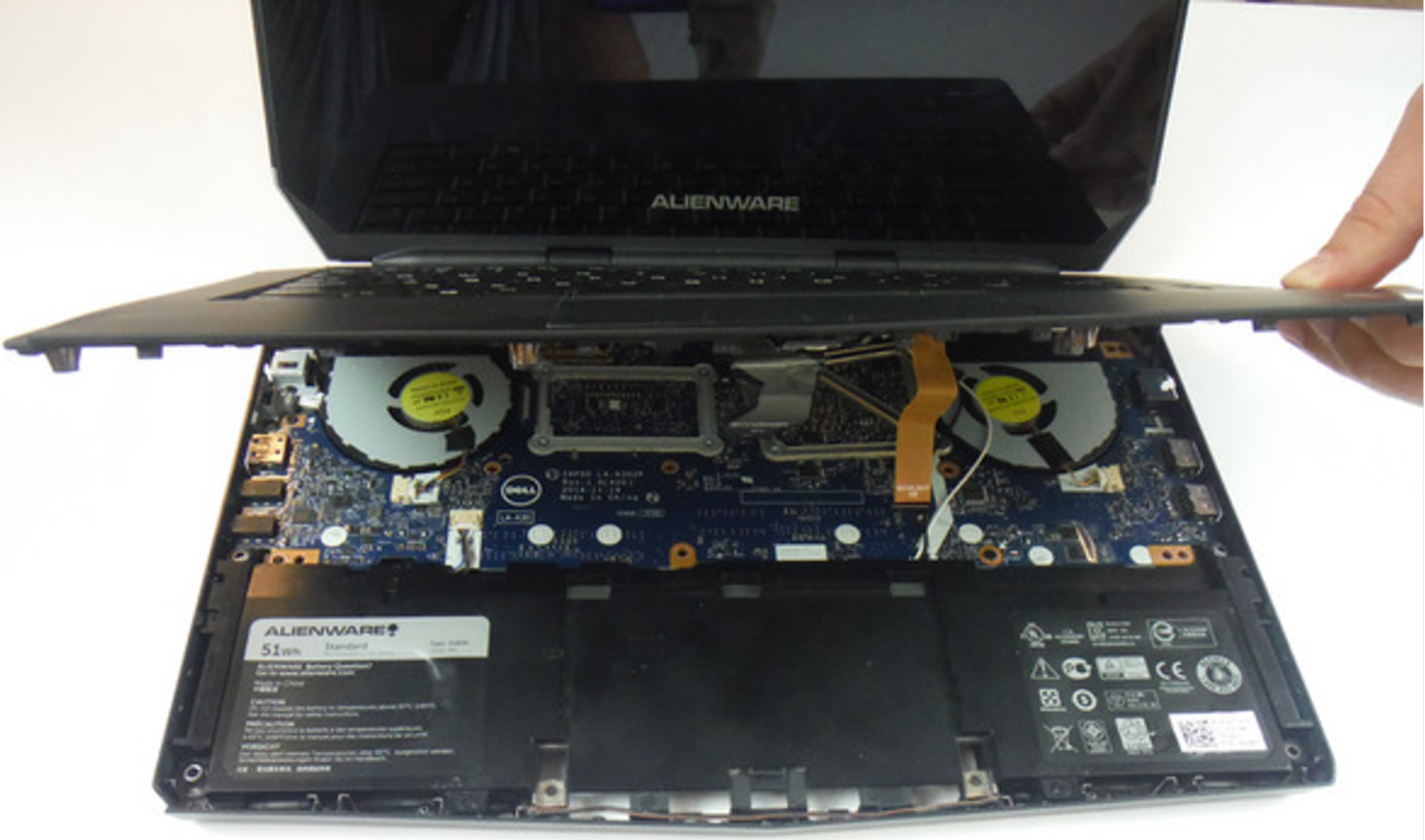


Personal
Mobile
Devices

Network
Edge
Devices


cooling towers
warehouse-scale computer
power substation

My other computer is a data center


ALIENWARE
ALIENWARE
51Wh
Standard

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:

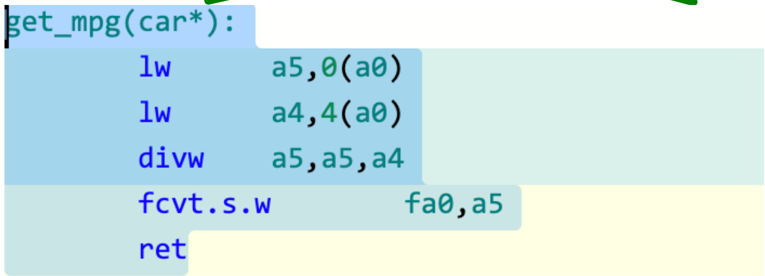
```
car *c = malloc(sizeof(car));
c->miles = 100;
c->gals = 17;
float mpg = get_mpg(c);
free(c);
```

Java:

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

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

Assembly language:



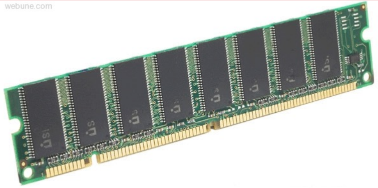
```
get_mpg(car*):
        lw      a5,0(a0)
        lw      a4,4(a0)
        divw    a5,a5,a4
        fcvt.s.w        fa0,a5
        ret
```

Machine code:

```
0111010000011000
100011010000010000000010
1000100111000010
110000011111101000011111
```

OS:

Computer system:



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)

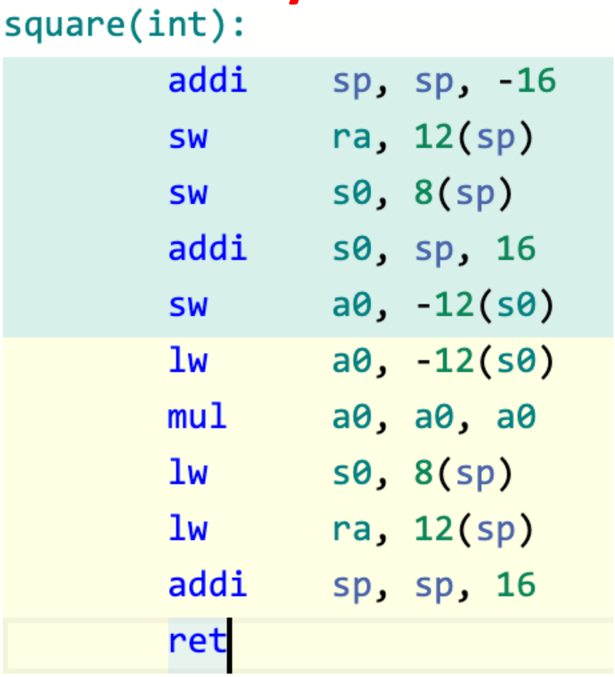
C Program

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

Binary

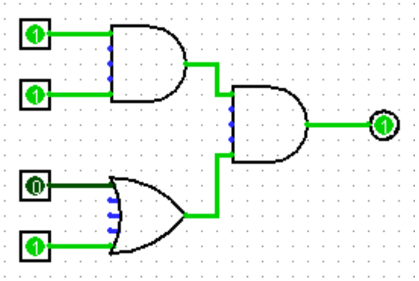
```
0x00000317
0x00830067
0xff010113
0x00112623
0x00812423
0x01010413
0xfea42a23
.......
```

Assembly

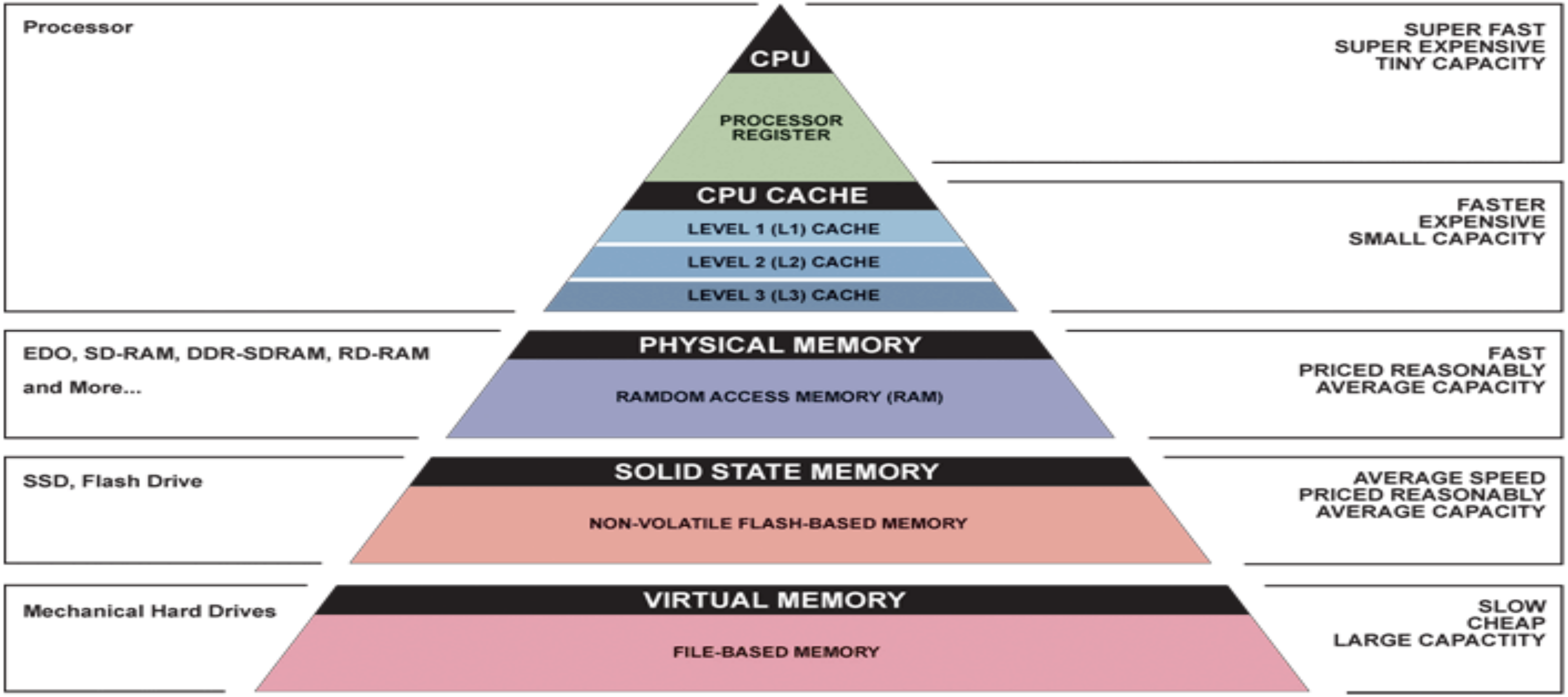


```
square(int):
        addi    sp, sp, -16
        sw      ra, 12(sp)
        sw      s0, 8(sp)
        addi    s0, sp, 16
        sw      a0, -12(s0)
        lw      a0, -12(s0)
        mul     a0, a0, a0
        lw      s0, 8(sp)
        lw      ra, 12(sp)
        addi    sp, sp, 16
        ret
```

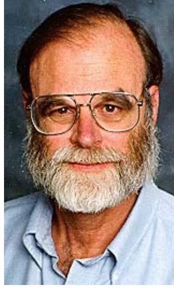
Logic



Great Idea #2: Principle of Locality/ Memory Hierarchy

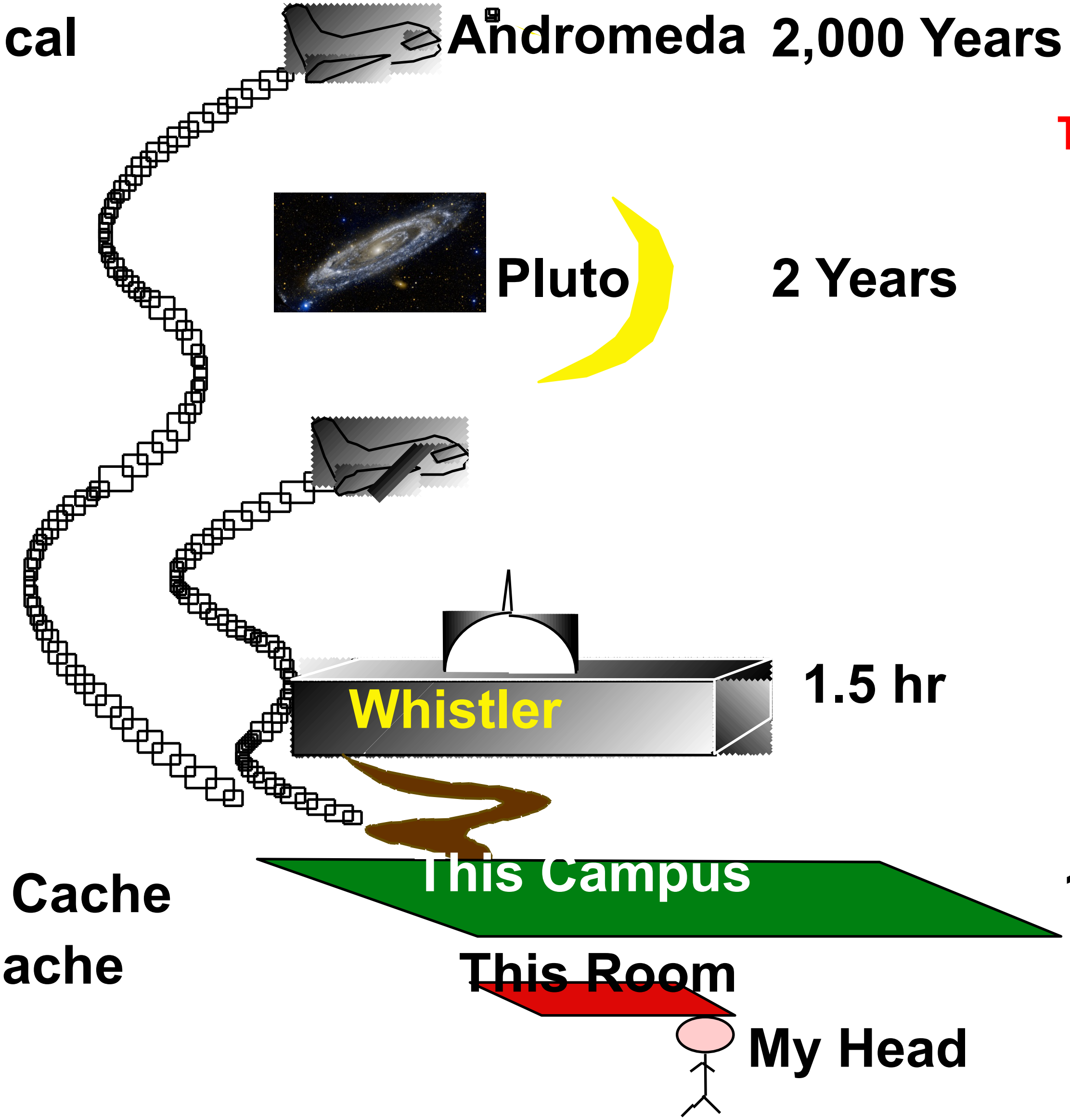


Jim Gray's Storage Latency Analogy: How Far Away is the Data?



Jim Gray
Turing Award

(ns)	
10^9	Tape /Optical Robot
10^6	Disk
100	Memory
10	On Board Cache
2	On Chip Cache
1	Registers



Andromeda 2,000 Years

Pluto 2 Years

Whistler 1.5 hr

This Campus 10 min

This Room

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!!

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

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)