Topic 1: Overview of Computer Organization and Systems Programming

CSE 30: Computer Organization and Systems Programming
Fall 2016

Diba Mirza
Dept. of Computer Science and Engineering
University of California, San Diego

About me

- Instructor: Diba Mirza
- Ph.D. in Computer Engineering, UCSD
- Teaching faculty @ CSE (this is my 3rd year teaching ☺)
- Office: 2124 EBU3B
- Email: dimirza@eng.ucsd.edu
- Office hours:
  - Mondays and Fridays 2:30pm – 3:30pm
  - Or by appointment

Goals of the course

- Go under the hood of high-level programs
- Identify causes for errors in high-level code
- Improve performance of your programs
- Unix compilation and debugging tools
- Basic software engineering: git and TDD
- Learn big ideas that have shaped computing
- Programming in C and ARM

Understand how a computer works from a programmer's perspective

About you

- What is your familiarity with C?

A. Know nothing or almost nothing about it.

B. Used it a little, beginner level.

C. Some expertise, lots of gaps though.

D. Lots of expertise, a few gaps.

E. Know too much

About you…

What is your familiarity/confidence with using version control with Git or any other Version Control System?

A. Know nothing or almost nothing about it.

B. Used it a little, beginner level.

C. Some expertise, lots of gaps though.

D. Lots of expertise, a few gaps.

E. Know too much

About you…

What is your familiarity/confidence with coding in pairs (pair programming)?

A. Know nothing or almost nothing about it.

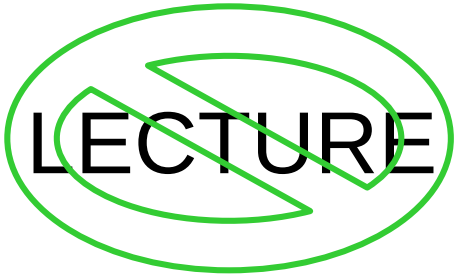
B. Used it a little, beginner level.

C. Some expertise, lots of gaps though.

D. Lots of expertise, a few gaps.

E. Know too much; I have no life.

About This Class



You must attend class
You must prepare for class
You must participate in class

In class we will use Clickers!

- Lets you vote on multiple choice questions in real time.

Lecture: Peer Instruction

- I will pose carefully designed questions. You will
  - Solo vote: Think for yourself and select answer
  - Discuss: Analyze problem in teams of two or three
    - Practice analyzing, talking about challenging concepts
    - Reach consensus
    - If you have questions, raise your hand and I will come over
  - Group vote: Everyone in group votes
    - You must all vote the same to get your point
  - Class wide discussion:
    - Led by YOU (students) – tell us what you talked about in discussion that everyone should know!

Why Peer Instruction?

- You get to make sure you are following the lecture.
- I get feedback as to what you understand.
- It's less boring!
- Research shows it promotes more learning than standard lecture.

Take a minute to introduce yourself to your group

Logistics: Resources

All information about the class is on the class website: https://ucsd-cse30-fall-2016.github.io

- *Approx* Syllabus
- Schedule
- Readings
- Assignments
- Forum (Piazza)
- Grades published on gradescope
- Grading policy

I will assume that you check these daily

Logistics: Course Components

Website: https://ucsd-cse30-fall-2016.github.io

4 PA Assignments	**28%**
Reading Quizzes (7)	**10%**
Midterm (2)	**30%**
Final (1)	**30%**
Class participation (Clickers)	**2%**

Grading structure and policy

- We will follow standard grading
- What do I need to get an A- or better?
  - >90% overall

- What do I need to get (C- or better)?
  - > 70% overall AND
  - Must appear on Final exam
  - >50% on the programming assignments

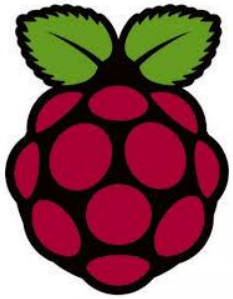
  Plusses and minuses at instructors discretion

Logistics: References

- *Required textbooks:*
  1. *Digital Design and Computer Architecture: ARM Edition*, by Sarah and David Harris. E-book available for free via the library
  2. *C programming: A modern approach, 2nd edition*, by K. N. King

Other suggested reading on course website

PAs: Raspberry Pi!



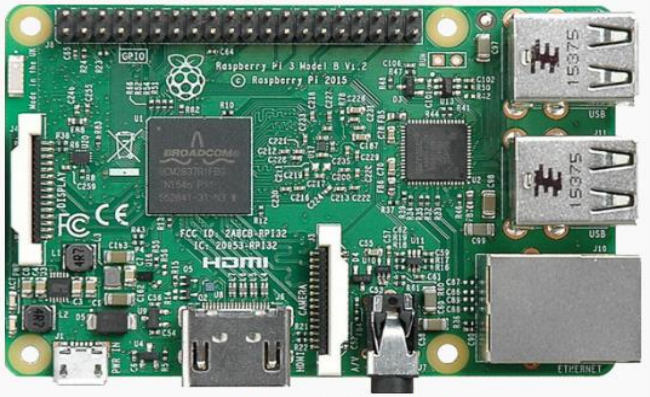
- We will program for an ARM based embedded platform!
- You may choose to go either of the following routes:
  - Purchase a Raspberry Pi ($75 investment) and work on the actual hardware
  - Emulate the Raspberry Pi in software using docker (it's a like a virtual machine but much faster)

Raspberry Pi 2

Raspberry Pi 2 model B

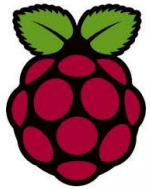
Release date	February 2015; 4 months ago
Introductory price	US$35
Operating system	Same as for Raspberry Pi 1 plus Windows 10 and additional variants of Linux such as Ubuntu
CPU	900 MHz quad-core ARM Cortex-A7
Memory	1 GB RAM
Storage	MicroSDHC slot
Graphics	Broadcom VideoCore IV
Power	4.0 W

Raspberry Pi 3



Raspberry Pi 3 model B

Release date	29 February 2016; 6 months ago
Introductory price	US$35
Operating system	Raspbian Ubuntu MATE Snappy Ubuntu Core Windows 10 IoT Core[1] RISC OS Debian Arch Linux ARM
System-on-chip used	Broadcom BCM2837
CPU	1.2 GHz 64-/32-bit quad-core ARM Cortex-A53
Memory	1 GB LPDDR2 RAM at 900 MHz[2]
Storage	MicroSDHC slot
Graphics	Broadcom VideoCore IV at higher clock frequencies (300 MHz & 400 MHz) than previous that run at 250 MHz
Power	800 mA (4.0 W)
Website	raspberrypi.org



Working out Logistics

- Have you bought a Raspberry Pi or own one?

A. Yes

B. No

Pair Programming Guidelines

You may work alone or with a partner from the SAME section of CSE 30

Basic rules for working with a partner

- All code written with two programmers at one machine
- You must plan *ahead of time* when you will get together
- You can change partner for each PA
- Don't be a jerk

Selecting partners: Factors to consider

- Schedule compatibility
- Roughly equal "eagerness"
- Roughly equal experience
- Partner from SAME section

Start looking for a partner today!

Course Problems…Cheating

- What is cheating?
  - Studying together in groups is encouraged
  - Turned-in work must be *completely* your own.
  - Common examples of cheating: running out of time on a assignment and then pick up output, take homework from box and copy, person asks to borrow solution "just to take a look", copying an exam question, …
  - Both "giver" and "receiver" are equally culpable

- Cheating on PA and HW/ exams; In most cases, F in the course.
- Any instance of cheating will be referred to Academic Integrity Office

Email Policy

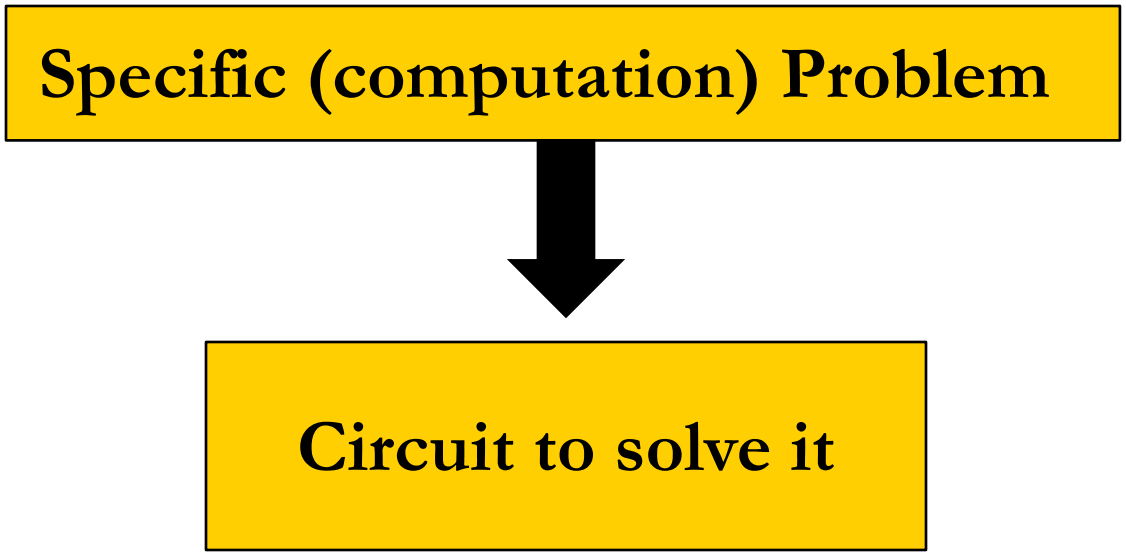
- Please use the forum as much as possible!
  - Your classmates benefit from your questions
  - Your classmates can answer your questions
  - I will check the forum daily

- I will attempt to respond to emails within 24 hours

Let’s look at the evolution of the modern digital computer ….

Big Idea behind early 'computers'

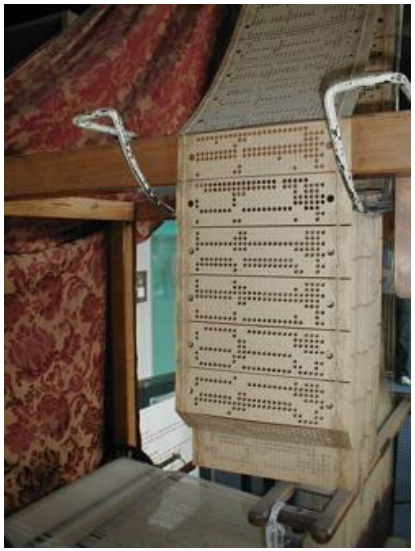
Fixed Program Model

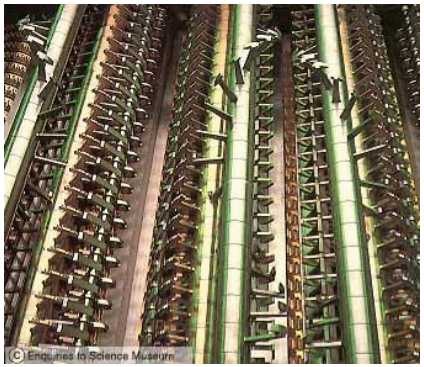


- The 'program' was wired into the computing device

The Evolution of Computing

Automated textile looms





Schickard's Machine

Jacquard's Loom

Analytical Engine

2400 BC

17th Century

1804

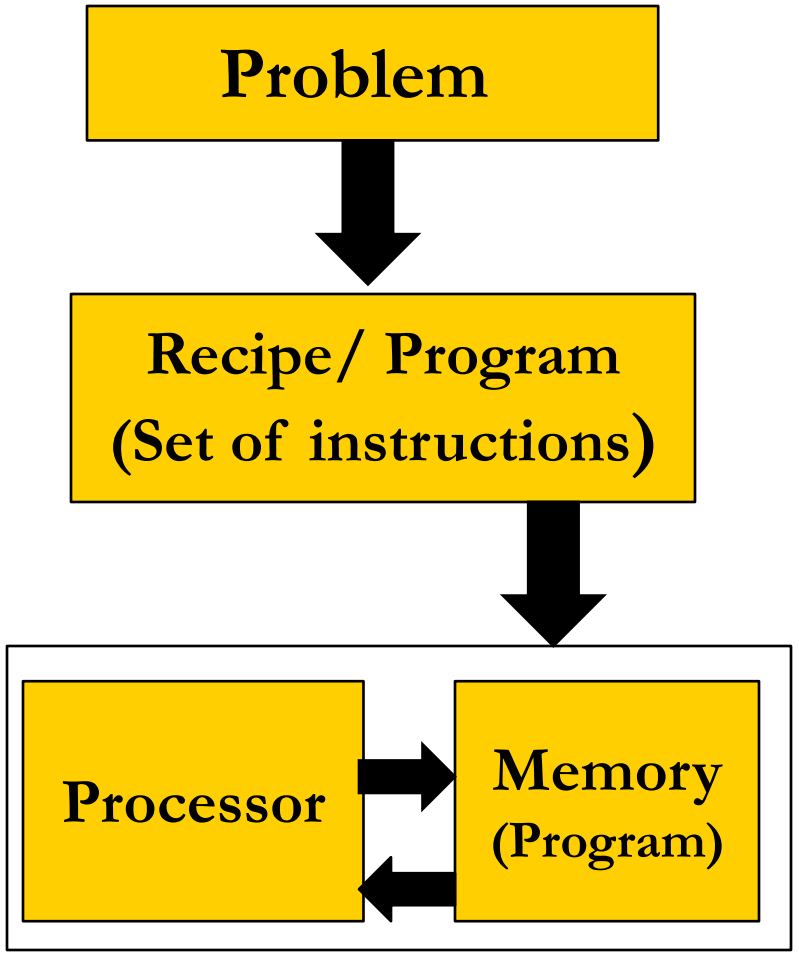
1822

Next big idea…The stored program model

- Key Idea(s):
  - Computer divided into two components: Processor and Memory
  - Program and data stored in the same place: memory

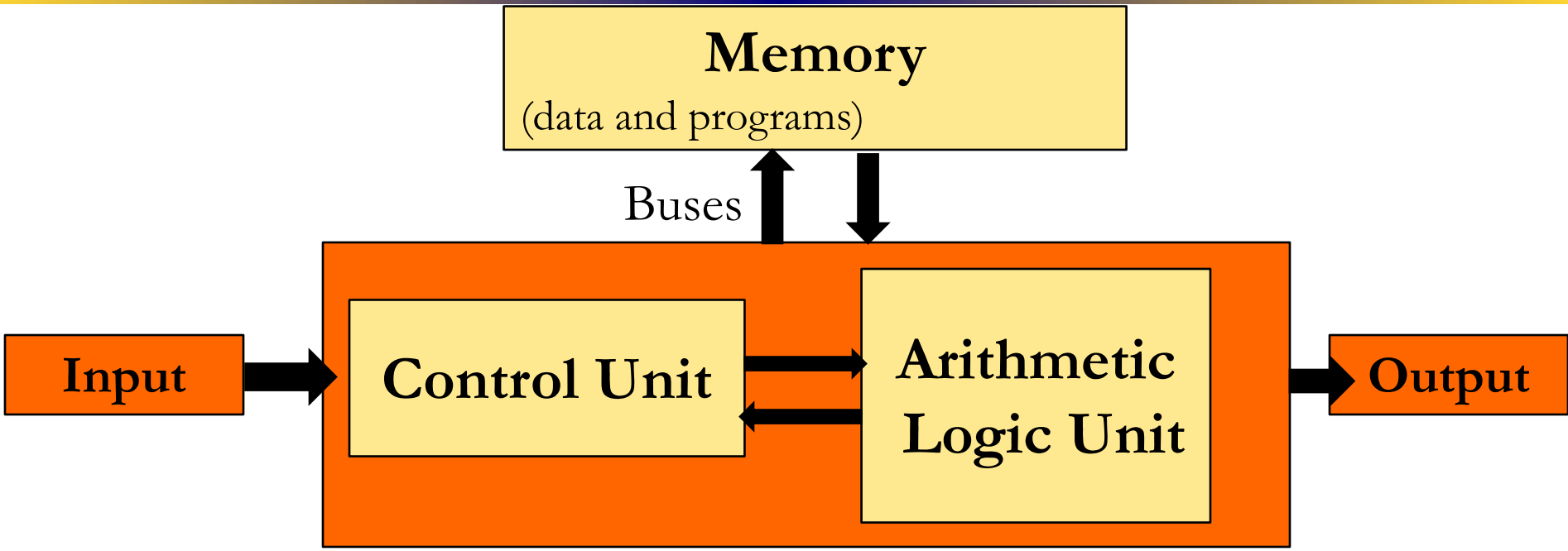
Have a new problem?

- Don't change the machine
- Change the recipe



Stored Program Model
proposed by Jon Von Neumann

The Von Neumann Architecture



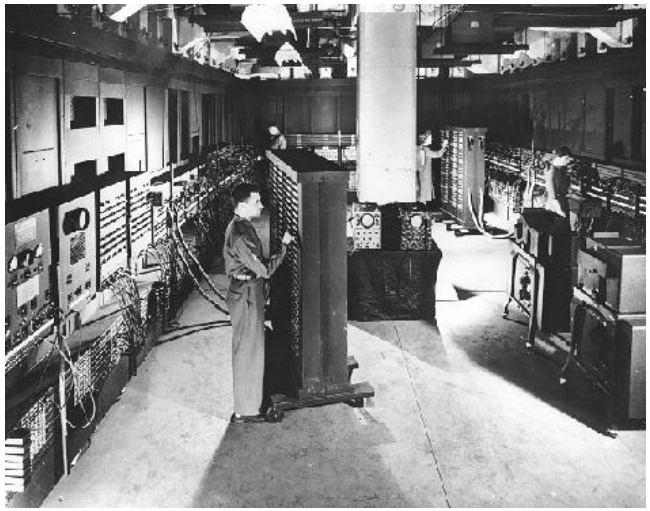
4 Basic Components of a Computer:

1. **Memory**: a long but finite sequence of cells (1D)
   - Each cell has a distinct address
   - Data in each cell: instruction, data or the address of another cell
2. **Control Unit**: Fetches instructions from memory and decodes them
3. **Arithmetic Logic Unit**: Does simple math operations on data
4. **Input/Output:** The connections with the outside world

The Evolution of Computing

Revolution:
1st Large Scale, General Purpose Electronic Computer

ENIAC



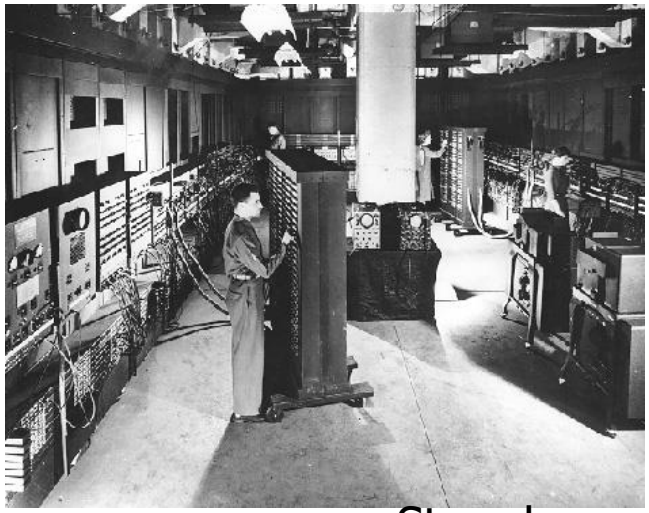
- More complex electronic circuits
- Solved more general problems
- Programming involved configuring external switches or feeding instructions through punched cards

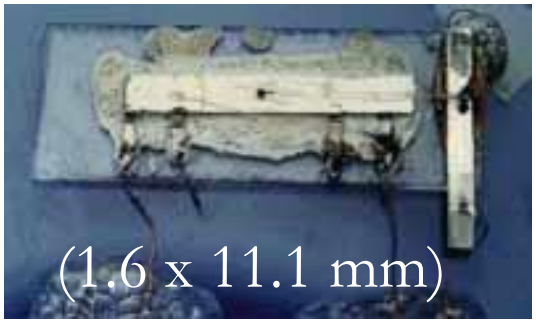
WWII | The stored program model

The Evolution of Computing

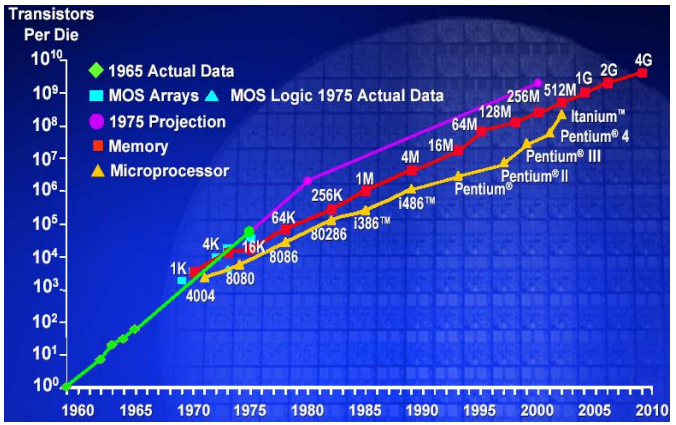
Revolution: Integrated Circuit:
Many digital operations on the same material

Vacuum tubes





Exponential Growth of Computation



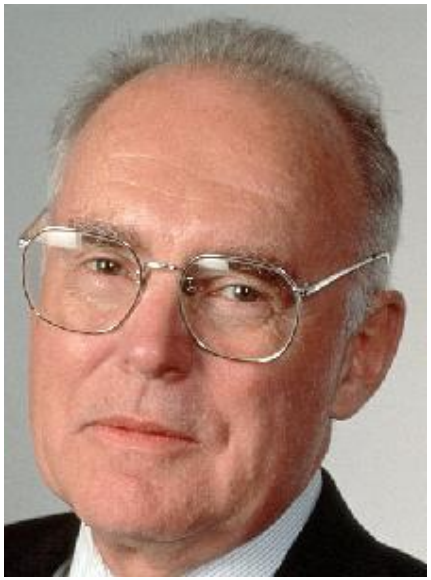
ENIAC Stored Program Model — WWII

Integrated Circuit — 1949

Moore's Law — 1965

UCSD

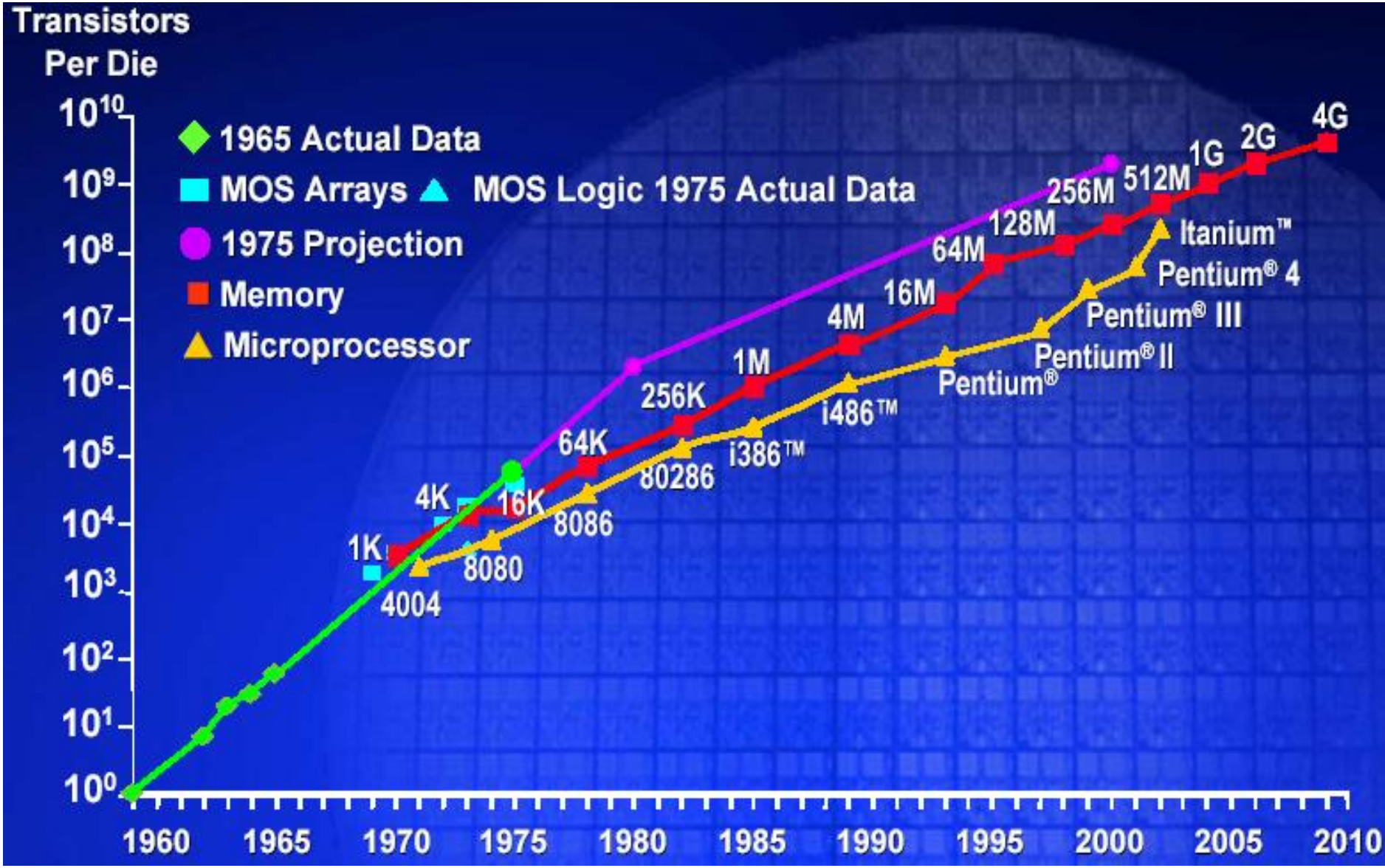
Technology Trends: Microprocessor Complexity



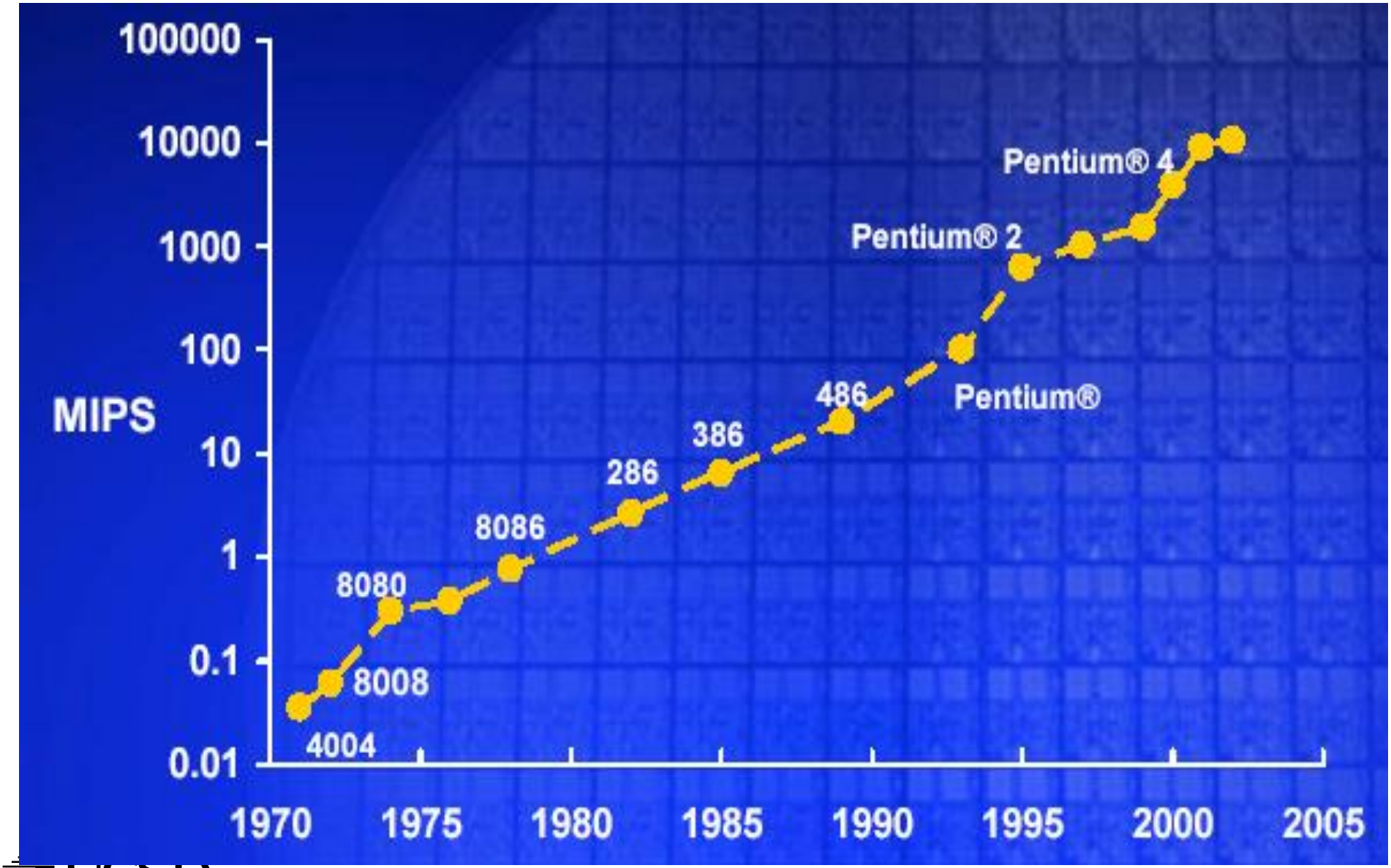
Gordon Moore
Intel Cofounder

In 1965, Gordon Moore predicted that the number of transistors per chip would double every 18 months (1.5 years)

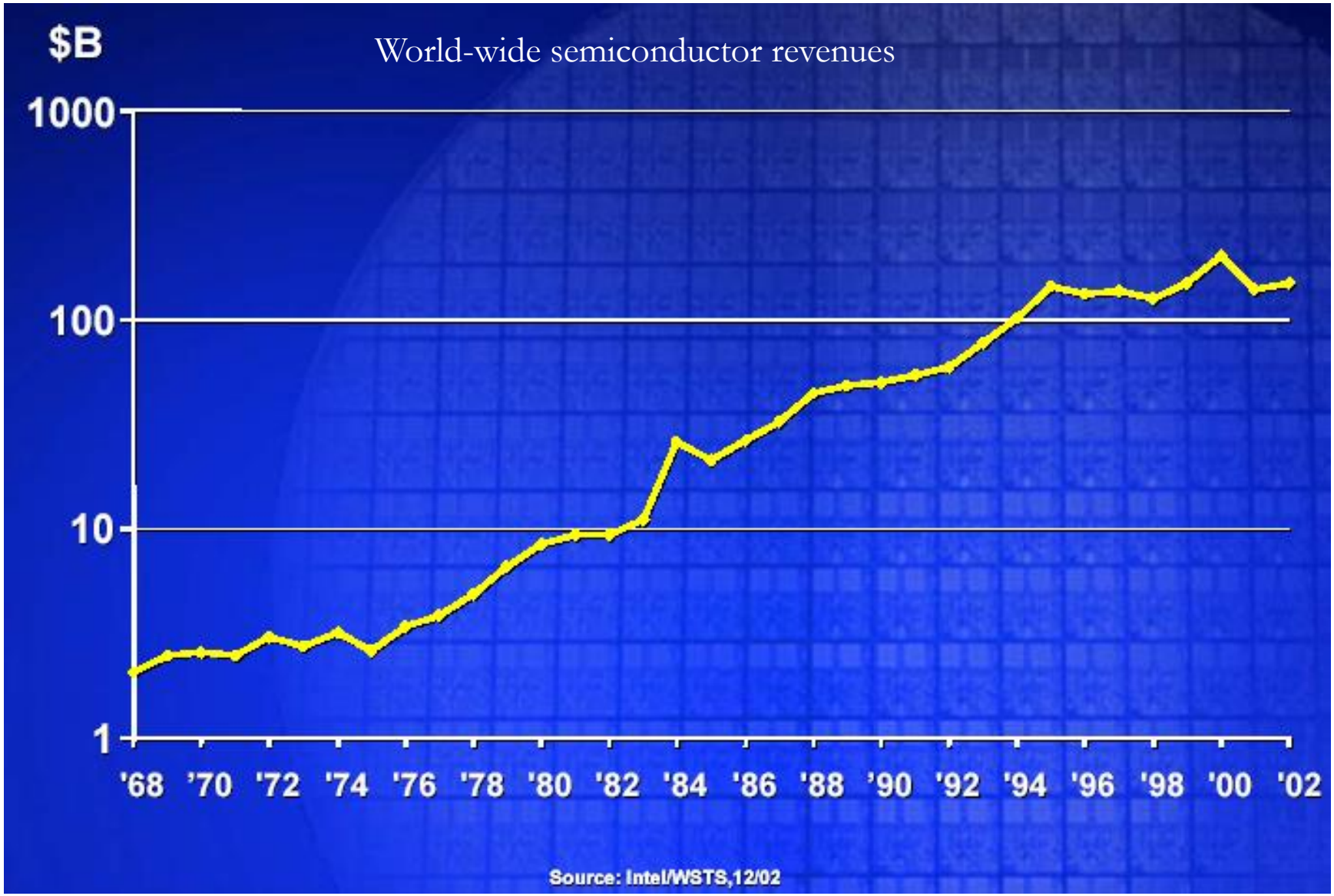
Exponential growth in computing



Side effects of Moore's Law



Side effects of Moore's Law

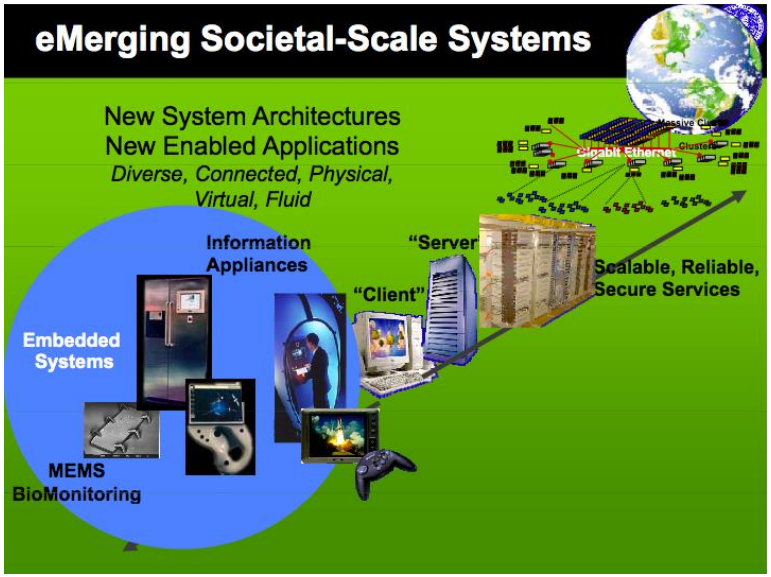


Computer Technology – Dramatic Change!

- Memory
  - DRAM capacity: 2x / 2 years (since ‘96);
    64x size improvement in last decade.
- Processor
  - Speed 2x / 1.5 years (since ‘85);
    100X performance in last decade.
- Disk
  - Capacity: 2x / 1 year (since ‘97)
    250X size in last decade.

Current State of Computing

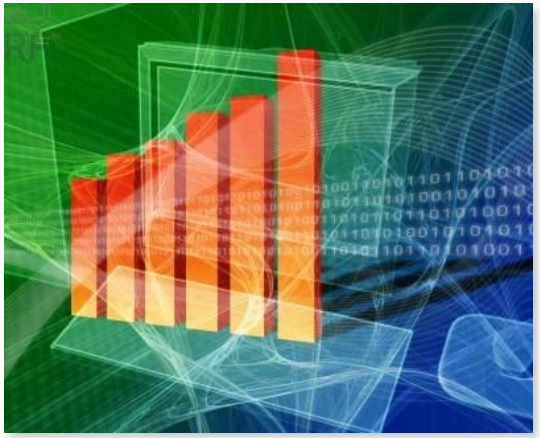
- Computers are cheap, embedded everywhere
- Transition from how to we build computers to how to we use computers



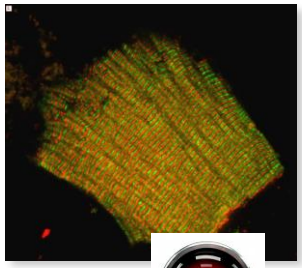
The Next REvolution

Ecological Monitoring

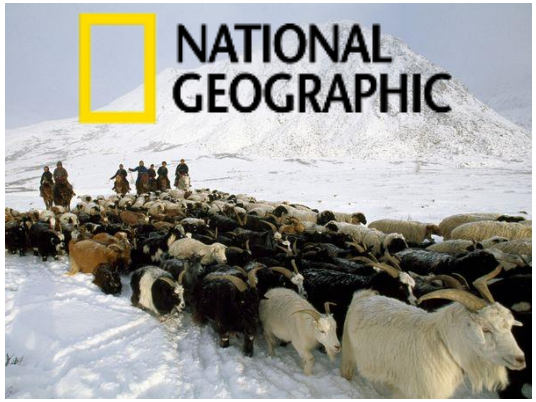
Financial Computation



Biotechnology



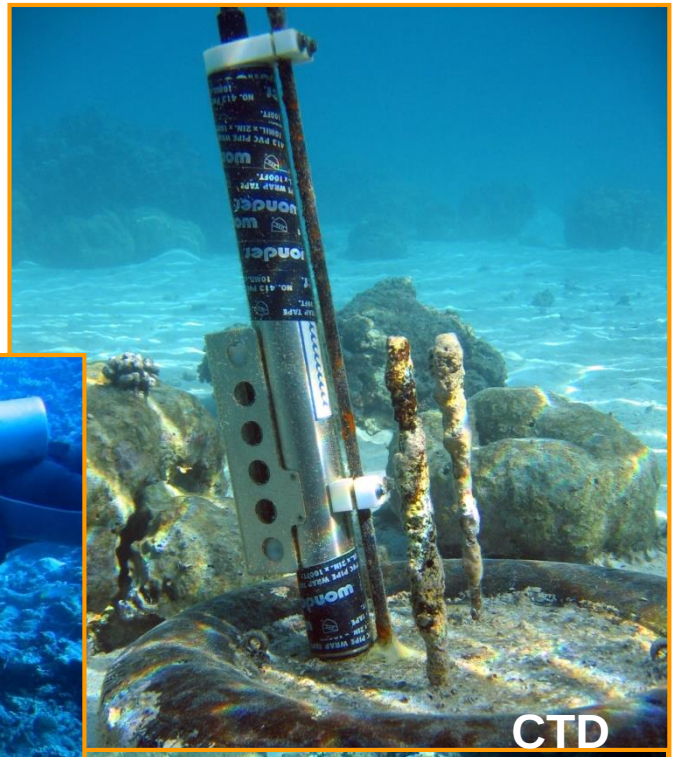
Robotics

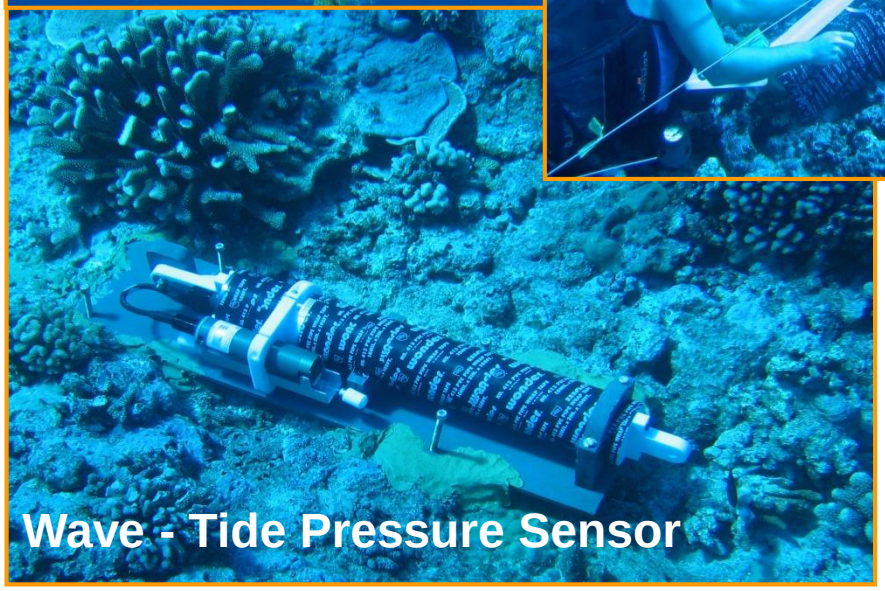


"The use of [these embedded computers] throughout society could well dwarf previous milestones in the information revolution."

Existing Sensors

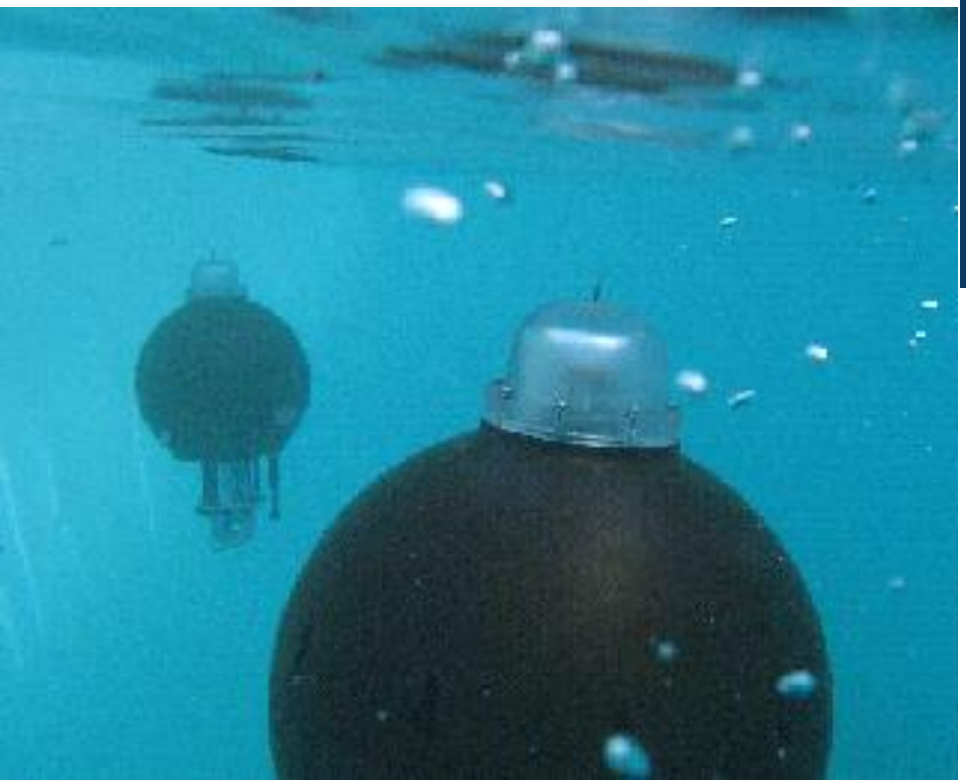
“Flipper Net”

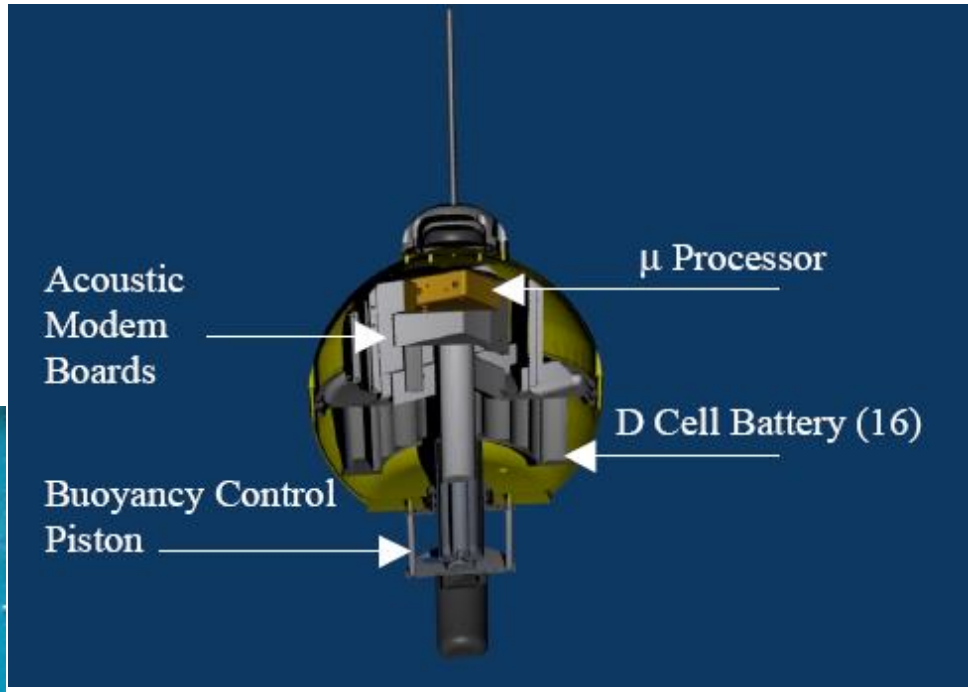




Drifters

- Autonomous Underwater Explorers: Self organizing drifters
- Dynamic, spatiotemporal 3D sampling
- Track water motions or mimic migration behavior of organisms

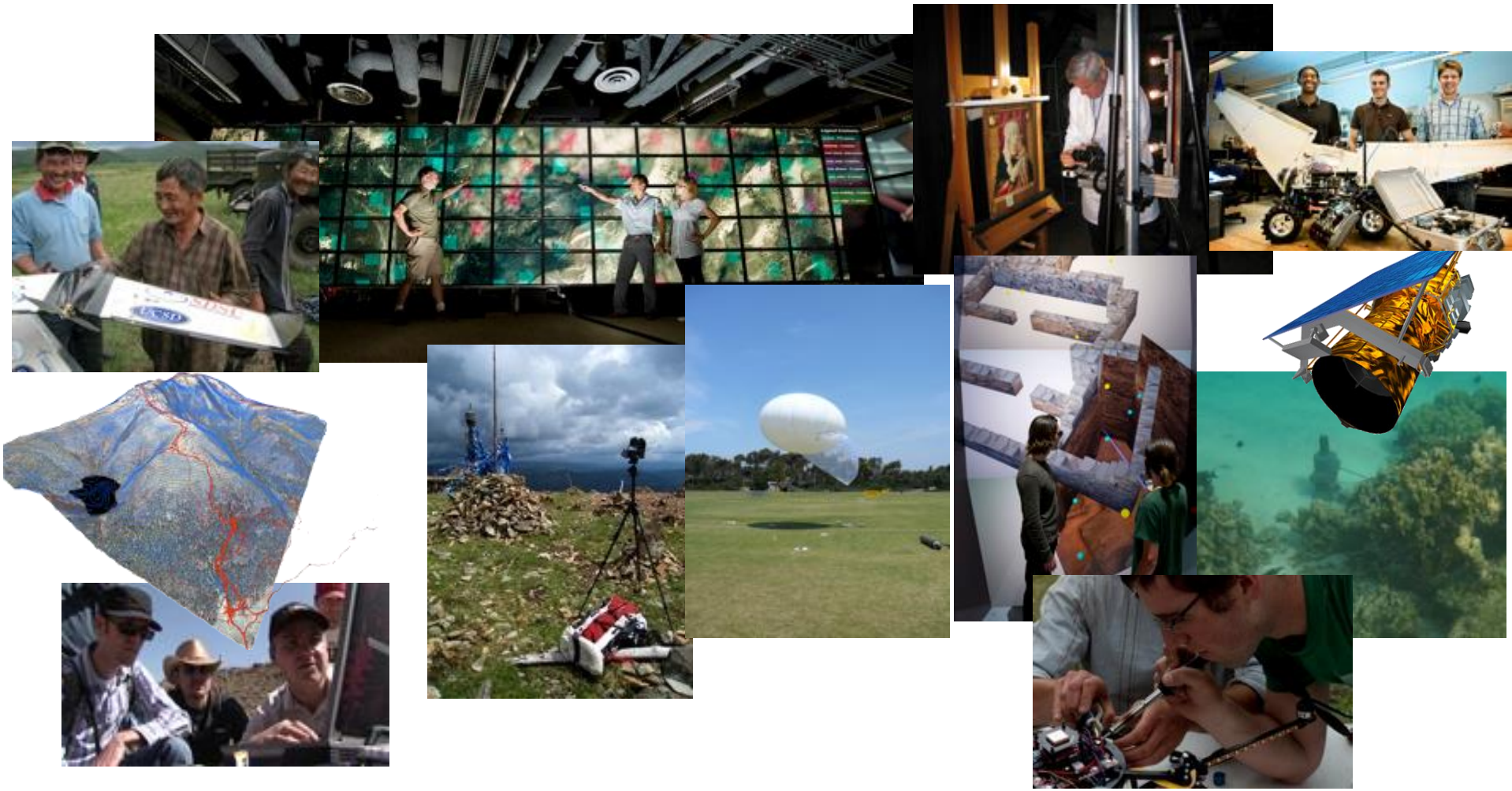




- Buoyancy control can follow ocean surface
- Acoustic modem for 3D localization amongst drifters
- 25 cm diameter
- Project in collaboration with Scripps Institution for Oceanography

National Geographic Engineers for Exploration

Happening at UCSD now:



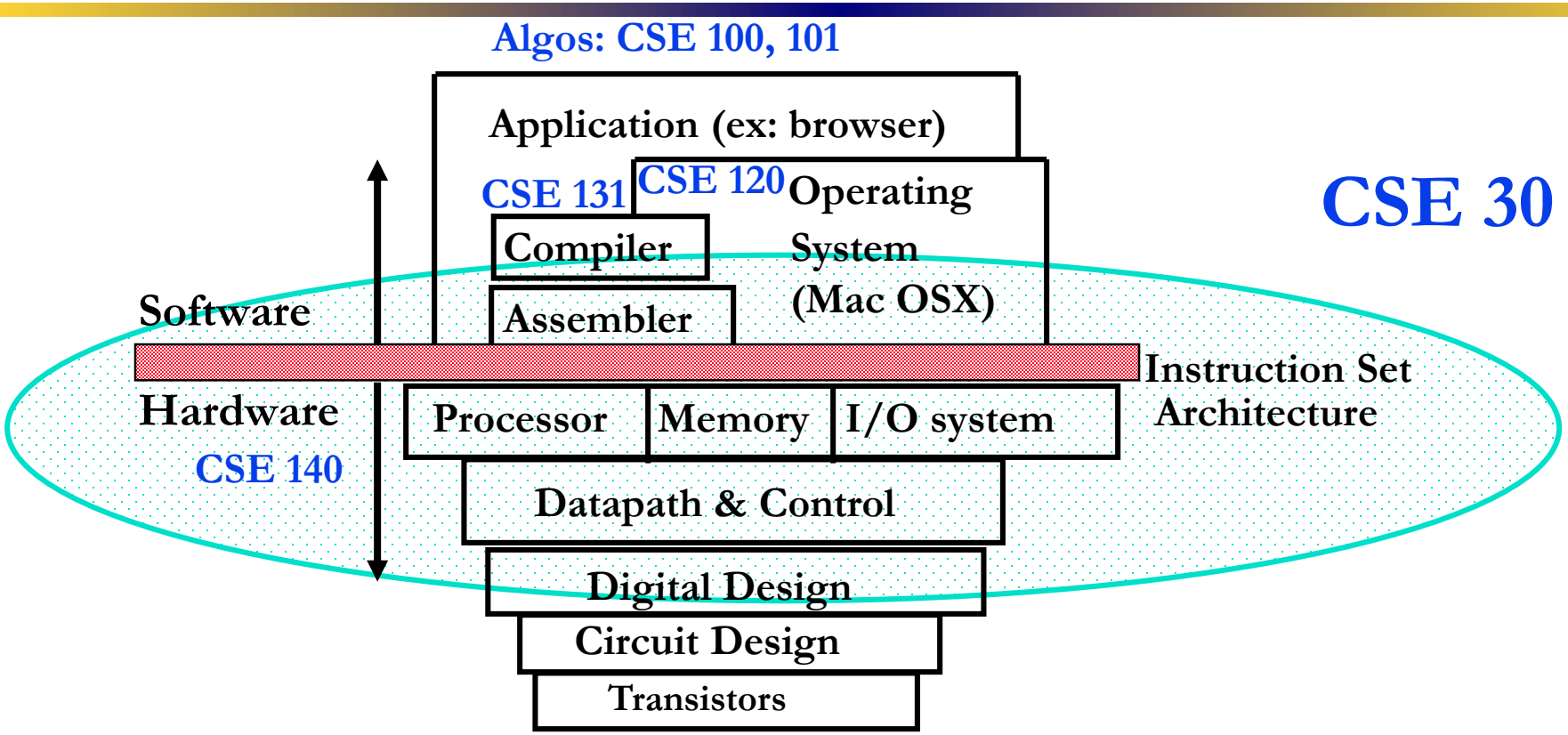
http://ngs.ucsd.edu/

UCSD

Computing Systems

- Increasingly smaller
- Higher performance
- More memory
- Lower power
- Embedded
- Everywhere
- …but extremely complex

How do we handle complexity?



- Big idea: Coordination of many *levels of abstraction*

Levels of Representation

High Level Language Program (e.g., C)

Compiler

Assembly Language Program (e.g.,ARM)

Assembler

Machine Language Program (ARM)

Machine Interpretation

Hardware Architecture Description (e.g., block diagrams)

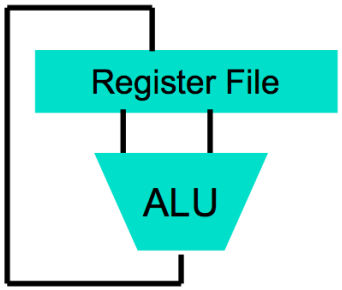
Architecture Implementation

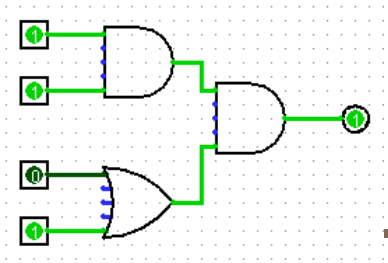
Logic Circuit Description (Circuit Schematic Diagrams)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
ldr   r0, [r2]
ldr   r1, [r2, #4]
str   r1, [r2]
str   r0, [r2, #4]
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```





Abstraction is good – but …

- We still need to understand the system!
- As a programmer you will be manipulating data.
- Data can be anything: numbers (integers, floating points), text, pictures, video !
- Writing efficient code involves understanding how "data" and programs are actually represented in memory
- Next class we'll talk about the bits and bytes of data: Number representation