Topic 1: Overview of Computer Organization and Systems Programming

CSE 30: Computer Organization and Systems Programming
Summer 2014

Diba Mirza
Dept. of Computer Science and Engineering
University of California, San Diego
Information about the Instructor/TAs

- Instructor: Diba Mirza
- Education: Ph.D. (ECE, UCSD)
- Office: 2124 EBU3B
- Email: dimirza@eng.ucsd.edu
- Office hours:
  - Tu, Fri 10:00am-10:50am
  - Or by appointment
- TAs: Riley Yeakle, Ning Liu
- Tutors: Alex Rosengarten, Martin Gao, Ben Martin, Junjie Luo
Goals of the course

1. Hone your C, learn the language of the machine: ARM Assembly

2. Become better programmers
   - Go beyond black box programming
   - Explore your bugs

3. Understand how a computer works
   - Look under the hood of high-level programs
   - Learn big ideas that have shaped computing: interesting!
   - Understand the limits of a computer
What we will learn

1. What the programmer writes?
   - A high level language: Specifically C

2. How the program is converted to the language of h/w?
   - Assembly Language: Specifically ARM

3. How the machine executes the program?
   - The main components of the computer
   - Peek into the processor
   - Interaction of the processor with other components of the computer (Memory, I/O)

4. What are the causes for errors in our high-level code?

5. Why do programs go slow?
## Logistics: Course Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW/PA Assignments</td>
<td>30%</td>
<td></td>
</tr>
<tr>
<td>Midterm</td>
<td>25%</td>
<td>1.5 hrs</td>
</tr>
<tr>
<td>Final</td>
<td>35%</td>
<td>3 hrs</td>
</tr>
<tr>
<td>Class participation (Clickers)</td>
<td>10%</td>
<td></td>
</tr>
</tbody>
</table>
Grading structure and policy

- What do I need to get an A- or better?
  - >90% overall

- What do I need to pass (C or better)?
  - > 50% overall AND
  - Must appear on Final exam
Logistics: Resources

- All information about the class is on the class website
  - Approx Syllabus
  - Schedule
  - Readings
  - Assignments
  - Grading policy
  - Forum (Piazza)

I will assume that you check these daily

- Grades will be posted on Gradesource
Logistics: References

- **Main references:**

*Other suggested reading on course website*
PAs: Raspberry Pi!

Why Pi?
Practice programming for an ARM based embedded platform!

Collect your Pi from Riley and Ning during discussion-TODAY!

Collateral returned at the end of the course if the board is in working condition.

Get your own and we’ll put an image for you!
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 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
Do you have your own Raspberry Pi or plan to buy one soon?
A. Yes
B. No
Familiarity with C

How familiar are you with C?

A. I know C pretty well and have a lot of programming experience
B. Reasonably well, lots of gaps
C. I don’t know what a pointer in C is
D. All I know is that it is a programming language that follows B.
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 ....
The Evolution of Computing

2400 BC

Pascaline

17th Century

Schickard’s Machine
Big Idea behind early ‘computers’

Fixed Program Model

- Specific (computation) Problem
- Circuit to solve it

- The ‘program’ was wired into the computing device
The Evolution of Computing

- 2400 BC: Abacus
- 17th Century: Schickard’s Machine
- 17th Century: Pascaline
- 1804: Jacquard’s Loom
- 1822: Analytical Engine
- Automated textile looms
Next big idea... The stored program model

- Key Ideas:
  - Computer divided into two components: Processor and Memory
  - Program and data stored in the same place: memory

- Consequences
  - Programs easily fed into the computer
  - Avoid clumsy methods of programming

Stored Program Model proposed by Jon Von Neumann
We can do a lot of complex computation by:
• Designing a minimal set of instructions that the machine can understand
• Writing programs in terms of these instructions

Have a new problem?
• Don’t change the machine
• Change the recipe
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
Revolution:
1\textsuperscript{st} Large Scale, General Purpose Electronic Computer

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

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

Vacuum tubes

(1.6 x 11.1 mm)

ENIAC Stored Program Model

1949

Integrated Circuit

Exponential Growth of Computation

Moore’s Law

1965
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

Number of transistors shipped per year

Source: Dataquest/Intel, 12/02
Side effects of Moore’s Law

Average transistor price per year

Source: Dataquest/Intel12/02
Side effects of Moore’s Law

World-wide semiconductor revenues

Source: Intel/WSTS, 12/02
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

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

“Flipper Net”

Thermistor

ADCP

Wave - Tide Pressure Sensor

CTD

ADP
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
- Under development by Curt Schurgers (ECE), Jules Jaffe, Peter Franks (SIO), Raymond de Callafon (MAE)
Sample
By: Michael Luong

Face Detection Activated:
Searching Target Database...
Target: Bear
Searching Species Database...
Species: Black Bear
Search Successful
Safari Park: Future Deployment

Targets:
National Geographic Engineers for Exploration

Happening at UCSD now:

http://ngs.ucsd.edu/
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

Diagram:
- Hardware: CSE 140
- Software: CSE 30
- Instruction Set Architecture
  - Processor
  - Memory
  - I/O system
  - Datapath & Control
  - Digital Design
  - Circuit Design
  - Transistors

Classes:
- CSE 120
- CSE 131
- CSE 140
- CSE 100, 101
- Algos: CSE 100, 101
Levels of Representation

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

Assembly Language Program (e.g., ARM)

Machine Language Program (ARM)

Compiler

Assembler

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

Register File

ALU

Galileo Galilei (1564-1642)
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
- Take a break and let’s talk bits and bytes: Number representation
Reading Assignment

- ARM 1.4, 1.5.1, 1.5.3