ECE 30
Introduction to Computer Engineering

Professor Gert Lanckriet
University of California, San Diego
Goal of the course

- Introduction to computer engineering: *building modern computer systems*
- Two components
  - Hardware: computer architecture and organization
  - Software: assembly language programming
- Both together determine what we ultimately want: PERFORMANCE
Prerequisites

- Two components
  - Hardware:
    - ECE 25 basics of logic design
  - Software:
    - ECE 15 high-level programming language (C, C++, Java)
      - waiver exam...
In 10 weeks, you will know...

- How computers look “under the hood”
- How C programs are translated into a low-level hardware language
- How the resulting program interfaces with the hardware
- How the hardware executes it
- How hardware and software aspects influence performance
In 10 weeks...

“... I understand a lot more about computers and how they work at the bit level and what type of hardware logic is used ...”
SW & HW

High-level Programming Language

- Fortran
- C

Compiler

- Assembly Instructions (MIPS R2000)

Functional Blocks

- CPU
- MEM
- I/O

Digital Circuits
Components

- Hardware
  - CPU
    - Datapath: arithmetic and logic operations
    - Control: direct operations in datapath
  - Memory
    - Stores instructions and data
  - I/O
    - Mouse, keyboard, display, disk drive, network interface, etc.

- Software
  - Compiler
Components

Datapath

Memory

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

... program & data

I/O

Fetch – Decode – Execute Cycle
Why would you take this class?

- To understand how a MAC or PC is built
- To impress a boy/girlfriend on a date
- EE majors need it for graduation
- To justify to your parents why they need to spend $40,000 for college education
- It’s the only class in the EE curriculum which doesn’t require Calculus as a prereq
- Burning desire to build a hot rod microprocessor one day
Why I am teaching it...

- Research:
  - Machine Learning / Artificial Intelligence
  - Applied Statistics
  - Music Information Retrieval

- No need to impress my girl-friend, to graduate, to get A’s, to justify tuition or to avoid calculus...
Why I am teaching it...

- It is cool stuff...
- Computers: third revolution of civilization
- Had transport industry kept pace with computer industry
  We could travel coast to coast in a second & for 99 cents
Imagine...

We could live in Tahiti...

... work in New York ...

... go to Moscow for an evening at Bolshoi Ballet!
Also affecting AI

KERNELTRON

(Genov, Cauwenberghs)

- **AI on a chip**
- Support vector machines (SVM)
- Outperforms existing SVM software by far
- SW/HW co-design
History of computing systems

- Sit back,
- Relax,
- Be impressed…
Generation -1
Early days (…-1642)

- Calculation needed for transactions and maintaining inventories
- Counting by means of matching one set of objects with another set: stones and sheep.
- Adding or subtracting groups of objects to or from the sack of counting stones
Generation -1
Early days (…-1642)

- Early counting tables: *abaci*
  - formalizing this counting method
  - introducing the concept of positional notation that we use today
Generation -1
Early days (...-1642)

- Only much later, counting became an abstract process and groups of objects were represented by a symbolic name.

- This allowed for computing on ... papyrus
Generation -1
Early days (…-1642)

- Twelfth century:
  Muhammad ibn Musa Al'Khowarizmi developed the concept of a written process to be followed to achieve some goal
- Published a book
- Hence the word algorithm
Generation -1
Early days (...-1642)

- Leonardo Da Vinci (1500): drawing of a mechanical calculator
Blaise Pascal, son of a tax collector, created in 1642 an adding machine with automatic carries from one position to the next.
Generation 0
Mechanical (1642-1945)

- Addition achieved by underlying gears, turning as each digit was dialed
- Sum displayed in a window above the "keyboard"
- Hardwired algorithm
Charles Babbage recognized in 1822 that most navigation tables contained lots of errors leading to the loss of ships.

He applies to the British Government for assistance, and receives the first government grant for computer research.
Generation 0
Mechanical (1642-1945)

- Babbage designs “Difference Engine”
- Computes entries in navigation tables
- Application specific hard-coded machine
Generation 0
Mechanical (1642-1945)

- Ada Augusta King, Countess of Lovelace, becomes the first programmer in 1842.
- Writes a program for the “Analytical Engine”, Babbage’s second and this time programmable device.
Generation 0
Mechanical (1642-1945)

- Increasing population in the US
- Demands of Congress to ask more questions in each census
- Need to improve processing methodology (otherwise 1890 census data only processed by 1900)
- **Herman Hollerith** won competition to deliver data processing equipment for 1890 US Census
- Company founded: **Hollerith Tabulating Company**
  - became one of the three composing Calculating-Tabulating-Recording (C-T-R) company in 1914
  - eventually renamed **IBM** in 1924
Generation 0
Mechanical (1642-1945)
Generation 0
Mechanical (1642-1945)

- **Konrad Zuse** (Berlin, 1935): developed his Z-1 computer in his parent's living room
- Relay computer, using binary arithmetic
- Instruction cycle time: 6 seconds (0.17 Hz)
Generation 0
Mechanical (1642-1945)

- Howard Aiken (late 1930’s) conceived first large-scale, automatic, general purpose, electromechanical calculator: IBM Automatic Sequence Control Calculator [ASCC]
- Program not stored in machine: driven by a paper tape containing the instructions
Generation 0
Mechanical (1642-1945)

- **Grace Hopper** found the first computer bug
- Bug beaten to death in the jaws of a relay!
- Glued it into the logbook of the computer
- When the machine stops (frequently) she told Howard Aiken that they are "debugging" the computer.
Generation 1
Vacuum Tubes (1945-1955)

- From relays (electromechanical switches) to vacuum tubes (electronic switches)
- Work on ENIAC (Electronic Numerical Integrator and Computer) started in 1943 by John Mauchly and J. Presper Eckert
- First large-scale, electronic, digital computer capable of being reprogrammed to solve range of computing problems
  → First general purpose computer
Generation 1
Vacuum Tubes (1945-1955)

- 18,000 vacuum tubes, 1,500 relays, 30 ton, 140 kW, 20 registers of 10 decimal digits, $500,000
- 1,900 additions per second
- “In the future computers will weigh at most 1.5 ton” (Popular Mechanics, 1949)
Generation 1
Vacuum Tubes (1945-1955)

- Programmed via 6000 multi-choice switches and tons of wires
Generation 1
Vacuum Tubes (1945-1955)

- John von Neumann (1946): invented the stored program machine
- Program was not anymore stored in switches and wires or on punched paper, but in program memory
- Computer architecture consisting of
  - Controller
  - ALU with accumulator
  - Program/data memory
- First to use binary arithmetic instead of decimal arithmetic
- Today’s computers still have this von Neumann architecture
Generation 1
Vacuum Tubes (1945-1955)

- **1948**: first stored program machine operational at the University of Manchester: the Manchester Mark I
Generation 1
Vacuum Tubes (1945-1955)

- 1952: John von Neumann with his new machine
Generation 1
Vacuum Tubes (1945-1955)

- **1952**: Grace Hopper introduces concept of reusable software
- She described how a computer could be used to compile codes written in a high-level language into pre-written low-level code segments, that then get assembled into programs in low-level code
- This described the concept of a compiler, i.e., language translation
1954: John Backus (IBM) develops programming language allowing users to express their problems in commonly understood mathematical formulae: FORTRAN

First FORTRAN compiler:
2000 punched cards
(2000 lines of - undocumented - code)
Generation 2
Discrete transistors (1955-1965)

- William Shockley, John Bardeen, and Walter Brattain invent in 1947 the "transfer resistance" device, later to be known as the transistor
- Allowed electronic devices to be smaller, lighter and cheaper
Generation 2
Discrete transistors (1955-1965)

- **1955**: IBM unveils the IBM704
- Mainframe computer using discrete transistors
- Connected to several dumb terminals
- Idea of central computer with distributed data input and output was born
- First machine with floating point logic
- Clock: **300 kHz**
Generation 3
IC: Integrated circuits (1965-1980)

- 1958: Jack St. Clair Kilby (TI) conceived and proved idea of integrating one transistor with resistors and capacitors on single semiconductor chip (size: half paper clip)
Generation 3
IC: Integrated circuits (1965-1980)

- **1965**: PDP-8 (DEC)
- First TRUE minicomputer
- Widely sold (300,000)
- 6 KB main memory
- Minuscule (8) instruction set
- Excellent interfacing
- $16,000
Generation 4
VLSI (1980-2005)

- VLSI = Very Large-Scale Integration

- Thousands, then hundreds of thousands of transistors, and currently far beyond one million, on one chip
Generation 4
VLSI (1980-2005)

- **1971**: Ted Hoff produces the Intel 4004, creating ONE chip for a calculator. It is the first microprocessor, i.e., entire-processor-on-a-chip (2400 transistors)
Generation 4
VLSI (1980-2005)

- Edward Roberts, William Yates and Jim Bybee (1973-1974) to develop the MITS Altair 8800, the first personal computer
- 256 bytes of memory, no keyboard, no display, and no auxiliary storage device
- $375
- Bill Gates and Paul Allen wrote their first product for the Altair: BASIC compiler
Generation 4
VLSI (1980-2005)

- **1977**: Apple II
- **1 MHz**
- **4 KB of memory**
- Audio cassette interface for loading programs and storing data
- **BASIC programming language built in**
- **$1298**
**Generation 4**  
**VLSI (1980-2005)**

- **1981**: popular IBM “PC”
- Standard storage device: cassette tape
- Floppy disk drive optional
- No hard disk
- **64 kB** on main board
- Three 64 kB expansion cards
- **4.77 MHz**
- **$1,565**
Generation 4
VLSI (1980-2005)

- **1986**: Cray-XMP supercomputer
- 4 processors
- 100 MHz
- Water-cooled
- 128MB RAM
- $15 MILLION
Generation 4
VLSI (1980-2005)

- **2000**: same performance reached in a PC by a single chip, the *Pentium III*
Generation 5
ULSI (2005-…)

- **ULSI** = ultra-large scale integration
  - More than 1 million transistors on a chip
  - No qualitative leap with VLSI

- **GBs** memory
  - 6KB (‘65), 64KB (‘81)

- **GHz** clock
  - .17Hz (‘35), 5 MHz (‘81)

- **TBs** hard drive
  - 10MB in ‘80s

- All under $1,000

- **Driver**: silicon IC technology!
Indeed...

- Had transport industry kept pace with computer industry
Practical Information

- Website: gertlanckriet.com
- Course load: 4 units
- Lectures: theory
  - Tu / Th - 12:30pm-1:50pm - CENTR 212
  - Tu / Th - 2:00pm-3:20pm - CENTR 212
- Discussion sections: review, examples
  - Mo - 3:00-3:50pm - CENTR 109
  - Mo - 4:00-4:50pm - CENTR 109
Practical Information

- **Instructor**
  - Gert Lanckriet
  - Office: 5604 Jacobs Hall
  - Office hours: Tuesday 3:30-3:55pm; Wednesday 3:45-4:50pm

- **Head TA**
  - Yonatan Vaizman
  - Head TA is **your principal contact** for ECE30: email **ANY** ECE30-related questions or comments to **ecethirty@gmail.com**. The head TA will route your question appropriately.
  - Gert will forward every ECE30-related email he receives to **ecethirty@gmail.com**. Except for email marked with “[FOR THE PROFESSOR ONLY]” in the subject line (which he may still forward to the head TA, depending on the question).
Practical Information

- **Project TAs**
  - Daniel Heineck and Sibo Wang
  - Office: WLH, room 2213B
  - Office hours: see project assignment

- **TAs / Tutors**
  - David Hall, Rahgav Abboy, Shengyao Guo, Cuong Luong, Shreyash Bhattarai, Junjie Lou
  - Office: WLH, room 2213B
  - Office hours: see website
Grading

- **Quizzes** (4 quizzes, 10% each) : 40%
- **Programming project** : 15%
- **Final exam** : 45%

**TOTAL** : 100%

- Curve, absolute grading and common sense
Grading

- If total below 50% or final below 50%: FAIL
- First question final << programming project = NO GOOD
- Historically
  - 60% - 68% ~ C
  - 70% - 80% ~ B
  - 82% - 100% ~ A
- “Grey zone factors”: lab participation and attendance, final (e.g., first question << programming project?)
- Do NOT underestimate the class
Do not underestimate the class

20 HOURS PER WEEK
Study problems

- Problem sets with solutions

- To help you to study and understand the class material

- Posted weekly, after covering theory

- Strongly recommended to work them out, study and try to understand them yourself, before asking questions
Quizzes

- 4 quizzes, 10% each
- During Monday discussion sections (04/20, 05/11, 05/25, 06/01)
- 05/25 rescheduled for 05/26: 7-8am and 8-9am
- "Quiz office hour":
  - Sunday 04/19, 05/10, 05/24, 05/31: day before quiz
  - 2:00-3:30pm, WLH 2213B
- Graded quizzes can be viewed during Junjie’s office hours:
  - Regular office hours, Sa post-quiz 9am-noon
  - Can be viewed until Sat before following quiz/final
Quizzes

- **Regrading of quizzes**
  - If you write anything on the quiz while viewing it, it will not be accepted for regrading.
  - All regrade requests must be emailed to ecethirty@gmail.com, with a clear and detailed explanation of WHY a regrade is being requested and for which problems.
  - A regrade request may result in the grader rechecking and regrading ALL problems. This may lead to an increase or decrease of the quiz grade.
  - **Deadline for regrade requests:** Friday before next quiz/final
    - May 9th for quiz 1
    - May 23th for quiz 2
    - May 31st for quiz 3
    - June 7th for quiz 4
Quizzes

- Based on understanding of class material, lab problems and study problems
- To give you an idea of possible quiz problems: example problems
  - Posted weekly
  - Solutions posted later in the quarter
- Guidance and discussion:
  - Office hours and “Quiz office hour”
  - No solutions are “given away” (take time to think)
  - If you cannot show us that you tried hard to work them out and understand them: we will not answer questions
Labs

- Problems and examples (while class is about theory)
- Led by the TAs
- Personal guidance: ask any questions you have
- Stimulate discussion about class material
Labs

- Tu / We / Th / Fr
  - WLH, 2213B

- Bring your laptop (no laptop: teaming up ok)

- Attendance:
  - Week 2, 3, 5, 7: mandatory
  - Week 4, 6, 8, 9, 10: mandatory unless satisfactory quiz grade
  - Check website “Updates and announcements”
Labs

- Progress monitored and logged (solutions are checked off by the professor or TA)

- You are free (and may need to) work on the lab problems at home

- You cannot leave the lab unless you finish it

- If out of time: finish at home and get solutions checked off at TA office hours no later than Friday the same week (no solving during the TA office hours)
Programming project

- A mini programming project in R2000 assembly language, verified using a simulator

- In groups of two (select your own partner - **ON TIME**):
  - partner from same lab section as yours
  - email partner info to ecethirty@gmail.com by April 23rd, 2015
  - if you don’t find a partner: email ecethirty@gmail.com by April 20th, 2015
  - missing deadlines or no partner = lose points

- Assignment online by April 27th

- Introduced during discussion section week 5
Programming project

- Project labs: week 5 and 7
  - 30 minutes extension
  - Come prepared!

- Office hours project TA: weeks 5, 6, 7, 8, 9

- On and after June 2nd, questions will no longer be answered

- Due: Thursday, June 4, 2015 (email commented source code by 5pm); late submissions NOT accepted

- Similar to programming question on final
Programming project

COPYING CODE
or ACQUIRING CODE:

minimum penalty = “F” for class

(all partners, both groups)
Final Exam

- **Review session:** Sun June 7th, 2015 - 2:00-4:00pm
  - WLH, room 2213B

- **Dates:** TBD

- No calculator, no cheat sheet, no nothing

- **Based on:**
  - Labs
  - Lectures
  - Programming project
  - Quizzes
Practical Information

- **Textbook**
  

- **When lecture slides are used, they will be available on the website.**

- **Class notes will be posted before starting new chapter**
  
  - Recommended: read notes in advance
  - Contain lecture material: don’t copy diagrams
Practical Information

- **Podcast:** [http://podcast.ucsd.edu/](http://podcast.ucsd.edu/)

- **Announcements:**
  - Can be made via website, in class, or email
  - Your responsibility to keep up to date with all announcements
Course Syllabus & Overview

Study classical Von Neumann computer organization using a simple RISC machine (MIPS 2000)

- Introduction
- Performance metrics, Amdahl’s law
- Instruction set
- Assembly language programming
- Number representation, computer arithmetic
- Central processing Unit: data path and control
- Memory
# Class, Lab, Discussion Schedule & Suggested Reading

<table>
<thead>
<tr>
<th>WEEK 1</th>
<th>1-Apr</th>
<th>LECTURE</th>
<th>LECTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Introduction</td>
<td>Perf. Measures</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ch. 2</td>
<td>Ch. 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ch. 1</td>
<td>Ch. 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WEEK 2</th>
<th>8-Apr</th>
<th>DISCUSSION</th>
<th>LECTURE</th>
<th>LECTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Assembler Basics</td>
<td>Assembler Basics</td>
<td>Assembler Basics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LAB</td>
<td>LAB</td>
<td>LAB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Perf. Measures, Simulator Setup, Assembler Basics</td>
<td>Assembler Basics</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ch. 3</td>
<td>Ch. 2, App. A</td>
<td>Ch. 2, App. B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.1, 2.2, 2.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WEEK 3</th>
<th>15-Apr</th>
<th>DISCUSSION</th>
<th>LECTURE</th>
<th>LECTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LAB</td>
<td>LAB</td>
<td>LAB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assembler Basics</td>
<td>Assembler Basics</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ch. 3</td>
<td>Ch. 2, App. A</td>
<td>Ch. 2, App. B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WEEK 4</th>
<th>22-Apr</th>
<th>DISCUSSION</th>
<th>LECTURE</th>
<th>DISCUSSION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>QUIZ</td>
<td>Assembler: Proc.</td>
<td>Project Intro</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LAB</td>
<td>LAB</td>
<td>LAB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Performance Measures, Assembler</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ch. 3</td>
<td>Ch. 2, App. A</td>
<td>Ch. 2, App. B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.5, 2.6, 2.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WEEK 5</th>
<th>29-Apr</th>
<th>LECTURE</th>
<th>LECTURE</th>
<th>LECTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LECTURE</td>
<td>LECTURE</td>
<td>LECTURE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assembler: ex.</td>
<td>Comp. Arithmetic</td>
<td>Comp. Arithmetic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LAB</td>
<td>LAB</td>
<td>LAB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ch. 4</td>
<td>Ch. 3</td>
<td>Ch. 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.1, 3.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Heads up...

- “Lab was very useful! It would be cool if we were allowed to attend multiple sections.”

- “Quiz office hours helped a lot as well as first hour of the lab.”

- “Podcast ... allows students ... to really listen to and try to understand the lecture”

- “Good to read his lecture notes before class”

- “Goes to each lab session, talks to individuals.”
Heads up…

- “Expects a lot from students…”
  “… cares deeply about the success of the students.”

- “Tough… but fair”
  “… makes the class bearable”
  “… fun yet productive environment created by the instructor”

- “Quizzes and project require a firm understanding of the material…”
  “… plenty of study material provided to make sure you do well”
  “… asks if students don't understand so he can explain it again”
Heads up...

“This is a difficult course!! Be prepared.”
Heads up…

“This is a difficult course!!
Be prepared.”

20 HOURS PER WEEK