Abstractions and Computer Systems

Gabriel Hugh Elkaim
Magic Number =

End of lecture: MODERN
Computing Machines are Everywhere!

- General Purpose
- Special Purpose (Embedded)
- Embedded
Computing Machines: Distinguishing Features

- Cost: $1-2\text{ millions} \to \infty, < $1k
- Speed: clock speed (20 MHz)
- Size: 48 bits - 64 bits
- Price/Performance
- Ease of Use (software & support)
- Intenace
- Power
Computing Systems
The Turing Machine (1.4)

While Ada Lovelace was the original programmer, the first person to prove the full power of software was Alan Turing (1912-1954).
The Turing Machine (2.4)

Universal Computational Device:

- Given enough time & memory, all computers are capable of calculating the exact same thing.

Turing's Thesis:

Every computation can be performed by some Turing machine, a universal Turing machine can imitate any other.
The Turing Machine (3.4)
The Turing Machine (4.4)
Problem Transformation and Abstraction

- Ultimate objective: transform a problem from human speech to electrons in a circuit.
- This is computer science & computer engineering
Computer Architecture (1.2)

Problems

Accents

Higher-Level Languages

Assembly Language

Microcode
Computer Architecture (2.2)

- Problems
- Algorithms
- Language
  - Instruction Set Architecture
  - Microarchitecture
  - Circuits
  - Devices
Computer Science

Definition - The study of algorithms and data structures to solve problems.

Abstraction - Software design that allows the programmer to focus on critical and of problem.

$\text{pow}(\text{num}, 2)$
Procedure or Function

```c
int average(a, b)
{
    int avg;
    avg = (a+b)/2;
    return (avg);
}

main()
{
    x = 4;
    y = 2;
    k = average(x, y);
}
```
Computer Engineering

**Definition:** Creluln application of engineering principles & methods to the design & development of hardware & software systems.

**Abstract:** Use a level of hardware abstraction that allows the designer to focus on critical set of details.
Instruction Set Architecture (ISA)

Interface between a computer's hardware and its software. Defines exactly what the computer's instructions do, now they are specified.
Central Processing Unit (CPU)
Motherboard: System
CPU: Package
SoC—System on a Chip

800 PROCESSOR

- Krait 400 CPU features 28nm process technology superior 2GHz+ performance
- Adreno 330 for advanced graphics
- Hexagon QDSP6 for ultra low-power applications and custom programmability
- Integrated LTE, 802.11ac, USB 3.0 and BT 4.0 offers broad array of high-speed connectivity

MULTIMEDIA
- Audio, video and capture

CAMERA

DISPLAY/LED

NAVIGATION

CONNECTIVITY
- 4G LTE, Wi-Fi, USB, BT and FM

Ultra HD Capture and Playback
- DTS-HD and Dolby Digital Plus audio
- Expanded gestures
- 55MP with dual ISP
- Support for up to 2560x2048 display
- Miracast 1080p HD support
- EZat GNSS with support for three GPS constellations
CPU: Microarchitecture (1.3)
CPU: Microarchitecture (2.3)
CPU: Microarchitecture (3.3)
Advanced Microarchitecture (1.3)

Intel Core i7
Advanced Microarchitecture (2.3)
Advanced Microarchitecture (3.3)

Mobile Chip

- Processor
- Graphics
- Core
- Core
- Shared L3 Cache
- System Agent, Display Engine & Memory Controller
- Memory Controller I/O

Gabriel Hugh Elkaim
Abstraction - The notion that we can concentrate on one "level" of the big picture at a time, with confidence that we can connect to levels above & below.

Framing - figuring at which level of abstraction is currently appropriate.
Hardware vs. Software

Abstraction does NOT mean being clueless about your neighboring levels.

In particular - hardware & software are inherently connected, especially done in this class.
Hardware vs. Software Engineers
Computer Organization

Electronic Devices ➔ Desired Behavior
General Purpose Computers

Devices -> General Purpose Computer -> Desired Behavior

Components: 
- Hardware
- Software
Simplified Computer Model

- Fetch instructions
- Load a value of a variable
- Store a new value to a variable
- Read (load or fetch)
- Write
MIPS Instruction Set Architecture (ISA)

Gabriel Hugh Elkaim
CISC vs. RISC

CISC: Complex Instruction Set Computing
- lots of instructions of variable size
- very memory optimized
- very few registers

RISC: Reduced Instruction Set Computing
- less instructions, all of a fixed width
- optimized for speed
- load/store architecture
What is “Modern”

ciscy risc

riscy cisc
“Hello World” (K&R)

- The only way to learn a new programming language is by writing programs in it. The first program to write is the same for all languages:
  
  \[\text{Print the words}\]
  
  \[\text{hello, world}\]

- This is a big hurdle; to leap over it you have to be able to create the program text somewhere, compile it successfully, load it, run it, and find out where your output went.

- With these mechanical details mastered, everything else is comparatively easy.
Hello World (1.3)

# Daniel J. Ellard -- 02/21/94 Modified by Max Dunne 09/27/17
# hello.asm -- A "Hello World" program.
# Registers used:
#  $v0 - syscall parameter and return value.
#  $a0 - syscall parameter-- the string to print.

.text

main:
    la $a0, hello_msg  # load the addr of hello_msg into $a0.
    li $v0, 4          # 4 is the print_string syscall.
    syscall            # do the syscall.
    li $v0, 10         # 10 is the exit syscall.
    syscall            # do the syscall.

# Data for the program:

.data

hello_msg: .asciiz "Hello World CMPE012L\n"

# end hello.asm
Hello World (2.3)

# Daniel J. Ellard -- 02/21/94 Modified by Max Dunne 09/27/17
# hello.asm-- A "Hello World" program.
# Registers used:
#      $v0 - syscall parameter and return value.
#      $a0 - syscall parameter-- the string to print.
.text

main:
    la $a0, hello_msg  # load the addr of hello_msg into $a0.
    li $v0, 4          # 4 is the print_string syscall.
    syscall            # do the syscall.
    li $v0, 10         # 10 is the exit syscall.
    syscall            # do the syscall.

# Data for the program:
.data
hello_msg: .asciiz "Hello World CMPE012L\n"

# end hello.asm

Gabriel Hugh Elkaim
Hello World (3.3)

# Daniel J. Ellard -- 02/21/94 Modified by Max Dunne 09/27/17
# hello.asm-- A "Hello World" program.
# Registers used:
#  $v0 - syscall parameter and return value.
#  $a0 - syscall parameter-- the string to print.
.text
main:
   la $a0, hello_msg   # load the addr of hello_msg into $a0.
   li $v0, 4            # 4 is the print_string syscall.
   syscall              # do the syscall.
   li $v0, 10           # 10 is the exit syscall.
   syscall              # do the syscall.

# Data for the program:
.data
hello_msg: .asciiz "Hello World CMPE012L\n"

# end hello.asm
Instruction Set Architecture (ISA)

- Programmer visible part of computer and operating system
  - Memory organization (address space)
  - Register set: how many, what size, how are they used
  - Instruction set
    - Operands (what commands I can actually give to computer)
    - Data types
    - Addressing modes
MIPS Architecture

- 32 bit words
- 32 32-bit wide registers
- Byte addressable memory
- Many opcodes - small register
  ~ 117

\[ 2^{10} = \text{Kilb} \]
\[ 2^{20} = \text{Mega} \]
\[ 2^{30} = \text{Giga} \]
\[ 2^{40} = 4 \text{TB} \]
Memory vs. Registers

Memory - Address Space \(2^{32}\) 32 bit addresses
Addressability - byte (8 bits)

Register - temporary storage, accessed in a single clock cycle
32 - 0...31
specific uses, specific purposes
others not directly accessible (PC, SP, AP 30 to multiply)

Gabriel Hugh Elkaim
Access Time of Storage

- RAID
- Cache
- SSD
- HDD
- ND
More on Memory

- Memory is NOT directly manipulated

  - First access it (copy it to register)
  - Manipulate it

  - Write back to register

  - Data: either read or write only

  - Text: read only memory (Flash)
### Register File (1.2)

#### Table 1: Register Conventions

<table>
<thead>
<tr>
<th>CPU Register</th>
<th>Symbolic Register</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
<td>zero</td>
<td>Always 0 (note 1)</td>
</tr>
<tr>
<td>r1</td>
<td>at</td>
<td>Assembler Temporary</td>
</tr>
<tr>
<td>r2 - r3</td>
<td>v0-v1</td>
<td>Function Return Values</td>
</tr>
<tr>
<td>r4 - r7</td>
<td>a0-a3</td>
<td>Function Arguments</td>
</tr>
<tr>
<td>r8 - r15</td>
<td>t0-t7</td>
<td>Temporary – Caller does not need to preserve contents</td>
</tr>
<tr>
<td>r16 - r23</td>
<td>s0-s7</td>
<td>Saved Temporary – Caller must preserve contents</td>
</tr>
<tr>
<td>r24 - r25</td>
<td>t8 - t9</td>
<td>Temporary – Caller does not need to preserve contents</td>
</tr>
<tr>
<td>r26 - r27</td>
<td>k0 - k1</td>
<td>Kernel temporary – Used for interrupt and exception handling</td>
</tr>
<tr>
<td>r28</td>
<td>gp</td>
<td>Global Pointer – Used for fast-access common data</td>
</tr>
<tr>
<td>r29</td>
<td>sp</td>
<td>Stack Pointer – Software stack</td>
</tr>
<tr>
<td>r30</td>
<td>s8 or fp</td>
<td>Saved Temporary – Caller must preserve contents OR Frame Pointer – Pointer to procedure frame on stack</td>
</tr>
<tr>
<td>r31</td>
<td>ra</td>
<td>Return Address (note 1)</td>
</tr>
</tbody>
</table>

**Note 1:** Hardware enforced, not just convention
Register File (2.2)

- Temporary  $t6 ... $t9
  - Procedures can destroy these. (syscalls)
- Saved  $s6 ... $s7
  - Must be saved by the caller, remain unmodified
- Arguments  $a0 ... $a3 passed to the callee
- Returns  $r0 ... $r1

Gabriel Hugh Elkaim
Syntax of MIPS
Classes of MIPS Instructions
ALU: ADD (1.3)
ALU: ADD (2.3)
ALU: ADD (3.3)
Pseudo-Ops: ADDI (1.2)
Pseudo-Ops: ADDI (2.2)
Branch: BLT (1.2)
Branch: BLT (2.2)
ALU: SUB (1.2)
ALU: SUB (2.2)
ALU: AND
ALU: OR
ALU: XOR
ALU: NOR
ALU: MULT (1.2)
ALU: MULT (2.2)
ALU: DIV
ALU: Shifts (1.3)
ALU: Shifts (2.3)
ALU: Shifts (3.3)
Memory Access (1.3)
Memory Access (2.3)
Memory Access (3.3)
Midterm Rules

Gabriel Hugh Elkaim
Midterm Rules

- There are four versions of the midterm
  - A, B, C, D
- Midterm is a Scantron multiple choice test
- You will turn in both your Scantron **AND** the exam
- Midterm is closed book, closed notes, no calculators
  - No cell phones, smart watches, etc.
- Bring a RED Scantron and a #2 Pencil
Front Page

Name: ___________________ Account: ______________@ucsc.edu

CMPE12 W18 Midterm Version A
February 20, 2018

- NO CELL PHONES ALLOWED. Once the exam has started if we see a cell phone you are done with your exam at that point (no further marks allowed). Leave your cell phone in your bag.

- NO BAGS AT DESKS. All backpacks must go on the sides of the classrooms. If we see a bag after the exam has started we will require you to move it.

- SIT IN YOUR ASSIGNED SEAT. You will get your seating assignment once you arrive to the exam or before.

- BEFORE YOU START, please fill out your name, CruzID, and the number of your exam on your scantron. Fill out the version of your exam (it is at the top of this page).

- This exam is closed-book, closed-note, and no calculators.

- If you wish to leave the room during the exam, bring your exam, scantron and your ID to the front of the room. We will hold your exam and scantron for you and will check your ID upon returning the exam room.

- You are not required to show your work. However, if for some reason you are suspected of academic misconduct, your handwritten work is your best defense. There is plenty of room for work on the exam.

- Put your name and cruzID on every page.

- When you are done, turn in both your scantron and your exam to the staff at the front of the room. Have your UCSC ID card ready, as we will check them upon collecting your exam.

- Your grade will be out of 116 points.

- DO NOT OPEN THE EXAM UNTIL YOU ARE TOLD TO DO SO
Scantron Filling In

1. Your ID number is the most important piece of information on this form. Without it we cannot scan your exam.
2. Please write AND bubble in your 7-digit Student ID number.
3. If your instructor is giving multiple versions of your exam, it is imperative that you mark which version you are taking.
4. Write and bubble in your last name, followed by a space, followed by your first name, followed by another space and your middle initial.

- If you neglect to mark this field correctly, we cannot scan your exam, and your instructor may not give you credit!
- If there is only one version of the exam, bubble in Test Form A.

Name

- REMEMBER: Machines read bubbles. People read letters. By writing in this information, you can ensure that the person grading your exam can manually enter any information the machine cannot read.
Test Coverage

• Everything since the first lecture including Labs
  – All number systems, binary, gates, transistors, memory, latches, ALU, light MIPS coverage
• Multiple choice exam
• 58 questions, out of 116 points.
• Score will be normalized to 100%
Questions?