EE 109 Unit 7 – Computer Organization
You Can Do That...

Cloud & Distributed Computing
(CyberPhysical, Databases, Data Mining, etc.)

Applications
(AI, Robotics, Graphics, Mobile)

Systems & Networking
(Embedded Systems, Networks)

Architecture
(Processor & Embedded HW)

Devices & Integrated Circuits
(Semiconductors & Fabrication)

Where we will head now...

- C / C++ / Java
- Assembly / Machine Code
- Scripting & Interfaces
- Networked Applications
- OS
- Libraries
- Processor / Memory / I/O
- Functional Units
  (Registers, Adders, Muxes)
- Logic Gates
- Transistors
- Voltage / Currents

SW

HW
Computer Engineering as Abstraction Levels

Software Code

Chips (Processors)

Functional Units

Logic

Transistors

if (x > 0) then
  x = x + y - z;
  a = h*x;

CMPR X,0
JLE SKIP
ADD X,X,Y
SUB X,X,Z
SKIP MUL A,B,X

Applications
C / C++ / Java
Assembly / Machine Code
OS
Libraries
Processor / Memory / I/O
Functional Units (Registers, Adders, Muxes)
Logic Gates
Transistors
Voltage / Currents

Controlling Input (Gate)
Output (Drain)
Source

x
y
z

A
B

S

AND gate

1110010101…
Motivation

• We will start to learn assembly language so that...
  – ...we understand why high level code has some of the constructs it has (if, while, etc)
  – ...we understand the basic hardware inside a computer and why certain structures are there
  – ...we can start to understand why HW companies create the structures they do (multicore processors)
  – ...we can start to understand why SW companies deal with some of the issues they do (efficiencies, etc.)
Computer Organization

- Three primary sets of components
  - Processor
  - Memory
  - I/O (everything else)
- Tell us where things live?
  - Running code
  - Compiled program (not running)
  - Circuitry to execute code
  - Source code file
  - Data variables
  - Data for the pixels being displayed on your screen
Input / Output

• Processor performs reads and writes to communicate with I/O devices just as it does with memory
  – I/O devices have locations (i.e. registers) that contain data that the processor can access
  – These registers are assigned unique addresses just like memory

FE may signify a white dot at a particular location

'a' = 61 hex in ASCII
Processor

- 3 Primary Components inside a processor
  - ALU
  - Registers
  - Control Circuitry

- Connects to memory and I/O via address, data, and control buses (bus = group of wires)
Arithmetic and Logic Unit (ALU)

- Executes arithmetic operations like addition and subtraction along with logical operations (AND, OR, etc.)
Registers

• Some are for general use by software
  – Registers provide fast, temporary storage locations within the processor (to avoid having to read/write slow memory)

• Others are required for specific purposes to ensure proper operation of the hardware

Processor

Memory

0
1
2
3
4
5
6
General Purpose Registers

- Registers available to software instructions for use by the programmer/compiler
- Instructions use these registers as inputs (source locations) and outputs (destination locations)
What if we didn’t have registers?

- Example w/o registers: F = (X+Y) – (X*Y)
  - Requires an ADD instruction, MULtiply instruction, and SUBtract Instruction
  - w/o registers
    - ADD: Load X and Y from memory, store result to memory
    - MUL: Load X and Y again from mem., store result to memory
    - SUB: Load results from ADD and MUL and store result to memory
    - 9 memory accesses
What if we have registers?

- Example w/ registers: \( F = (X+Y) - (X*Y) \)
  - Load \( X \) and \( Y \) into registers
  - ADD: \( R0 + R1 \) and store result in \( R2 \)
  - MUL: \( R0 * R1 \) and store result in \( R3 \)
  - SUB: \( R2 - R3 \) and store result in \( R4 \)
  - Store \( R4 \) back to memory
  - 3 total memory access

<table>
<thead>
<tr>
<th>Processor</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ALU} ) ( \text{ADD, SUB, AND, OR} )</td>
<td>( X ) ( Y ) ( F )</td>
</tr>
<tr>
<td>PC</td>
<td>0</td>
</tr>
<tr>
<td>Addr</td>
<td>1</td>
</tr>
<tr>
<td>Data</td>
<td>2</td>
</tr>
<tr>
<td>Control</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

![Diagram showing processor and memory connections](image)
Other Registers

- Some bookkeeping information is needed to make the processor operate correctly

- Example: Program Counter (PC)
  - Recall that the processor must fetch instructions from memory before decoding and executing them
  - PC register holds the address of the currently executing instruction
Fetching an Instruction

- To fetch an instruction
  - PC contains the address of the instruction
  - The value in the PC is placed on the address bus and the memory is told to read
  - The PC is incremented, and the process is repeated for the next instruction
Fetching an Instruction

- To fetch an instruction
  - PC contains the address of the instruction
  - The value in the PC is placed on the address bus and the memory is told to read
  - The PC is incremented, and the process is repeated for the next instruction
Control Circuitry

- Control circuitry is used to decode the instruction and then generate the necessary signals to complete its execution.
- Controls the ALU.
- Selects Registers to be used as source and destination locations.
Control Circuitry

• Assume 0201 hex is machine code for an ADD instruction of $R2 = R0 + R1$
• Control Logic will...
  – select the registers (R0 and R1)
  – tell the ALU to add
  – select the destination register (R2)
BACKUP
Dive Into A Smartphone

• What's inside?
  – Apple A8 APL1011 SoC + Elpida 1 GB LPDDR3 RAM
  – NXP LPC18B1UK ARM Cortex-M3 Microcontrollers (which is the proper name for the M8 motion coprocessor)
  – Qualcomm MDM9625M LTE Modem
  – InvenSense MP67B 6-axis gyroscope and accelerometer combo
  – SK Hynix H2JTDG8UD1BMS 128 Gb (16 GB) NAND Flash
  – Murata 339S0228 Wi-Fi Module
  – Broadcom BCM5976 Touchscreen Controller
  – Qualcomm PM8019 power management IC
  – Cirrus Logic 338S1201 audio codec

http://www.zdnet.com/whats-inside-the-iphone-6-plus-7000033873/