CS 230 – Introduction to Computers and Computer Systems

Module 0 – Introduction

Aakar Gupta
aakar.gupta@uwaterloo.ca

(Slides based on materials prepared by Sandy Graham)
Goals / Overview
Goals

- overview of computer systems
  - from bottom to top
- understand basic challenges & techniques
- understand performance implications
Course Information

- Course materials found on [https://www.student.cs.uwaterloo.ca/~cs230/](https://www.student.cs.uwaterloo.ca/~cs230/)

- Communication
  - Piazza
  - Office Hours
    - IA – Murray Dunne – Thursdays – 3:30-4:30pm (E5-5022)
    - Instructor – Aakar Gupta – Tuesdays – 10-11am (DC2129)
  - Email
Course/Assignment Tools

- student.cs environment
- UNIX tools
- Python
- MIPS assembler – Java based
- MIPS emulator – Java based

- material will be made available as needed
- also: attend the tutorials!
About the Slides

- some material and figures taken from textbook and accompanying slides:

David Patterson and John Hennessy. *Computer Organization and Design – The Hardware/Software Interface*

- figures taken from other sources are shown with reference

- other material newly developed for this course
Motivation and Background
Below Your Program

- application software
  - high-level language
- system software
  - operating system – windows, linux
  - compilers
- hardware
  - processor, memory, I/O
Model of a Computer

- von Neumann model
- CPU
  - Control unit
  - Arithmetic/logic unit
- I/O
  - user, storage, network
- Memory
  - program & data stored in memory
Program

- Sequence of instructions stored in memory
- Instructions are just collections of bits that the computer understands and obeys – *on/off*
- *Bit* – 0 or 1
- Example:

  1000110010100000  Machine Language
  add A,B  Assemble Language
  A+B  High-Level Language
High-level language program (in C)

```c
swap(int v[], int k)
{
    int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

Compiler

Assembly language program (for MIPS)

```assembly
swap:
    multi $2, $5, 4
    add $2, $4, $2
    lw $15, 0($2)
    lw $16, 4($2)
    sw $16, 0($2)
    sw $15, 4($2)
    jr $31
```

Assembler

Binary machine language program (for MIPS)

```
0000000001010001000000000100011000
0000000001000001000010000100000001
1000110111100100000000000000000000
1000111000100100000000000000000010
1010110000100100000000000000000000
1010110111100100000000000000000100
0000001111100000000000000000000100
```
Process

- Program and data “in action”
- Actual execution of the program instructions
- Usually managed by the operating system
Processors (CPUs)

- The hardware that executes the program instructions

<table>
<thead>
<tr>
<th>Year</th>
<th>Technology</th>
<th>Relative performance/cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1951</td>
<td>Vacuum tube</td>
<td>1</td>
</tr>
<tr>
<td>1965</td>
<td>Transistor</td>
<td>35</td>
</tr>
<tr>
<td>1975</td>
<td>Integrated Circuit</td>
<td>900</td>
</tr>
<tr>
<td>1995</td>
<td>Very large scale IC (VLSI)</td>
<td>2,400,000</td>
</tr>
<tr>
<td>2005</td>
<td>Ultra large scale IC</td>
<td>6,200,000,000</td>
</tr>
</tbody>
</table>
Moore's Law

- transistor density doubles every two years
  - every year 1959-1975
- in the past
  - transistor density translated into processing power
  - almost double speed every 2 years...
  - Reduces response time, increases performance
- not applicable anymore
Uniprocessor Performance

Constrained by power, instruction-level parallelism, memory latency
Multiprocessors

- multicore microprocessors
  - more than one processor per chip
- requires explicitly parallel programming
  - compare with instruction level parallelism (hidden)
- hard to do
Performance

- faster processor
- more processors (needs parallelization)
- better software (algorithms)
Efficiency Matters

- More processors require more power
- Network-centric computing, Internet
  -> large data centers
- hardware cheap, but
  - power consumption -> heat
  - heat -> cooling -> more power consumption
  - money and environment costs
- software performance
  - Minimize instructions
  - Minimize memory
• Program
• Data
Data & Registers

- Data needed by the running programs
  - Usually refers to RAM – Random Access Memory

- Registers are high speed storage areas in the CPU. All data must be stored in a register before it can be processed.
Trade-Offs

- almost everything in CS is a trade-off
  - very few absolute truths
- “fast, good, or cheap – pick two”
Course Topics / Modules

- arithmetic, hardware, data
- assembly language
- system internals
- build and runtime
- multiprocessing
- operating systems (if time allows)