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/
- 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

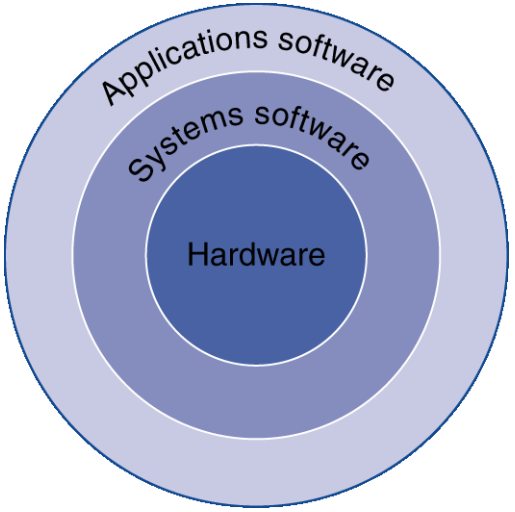
- some material and figures taken from textbook and accompanying slides:

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

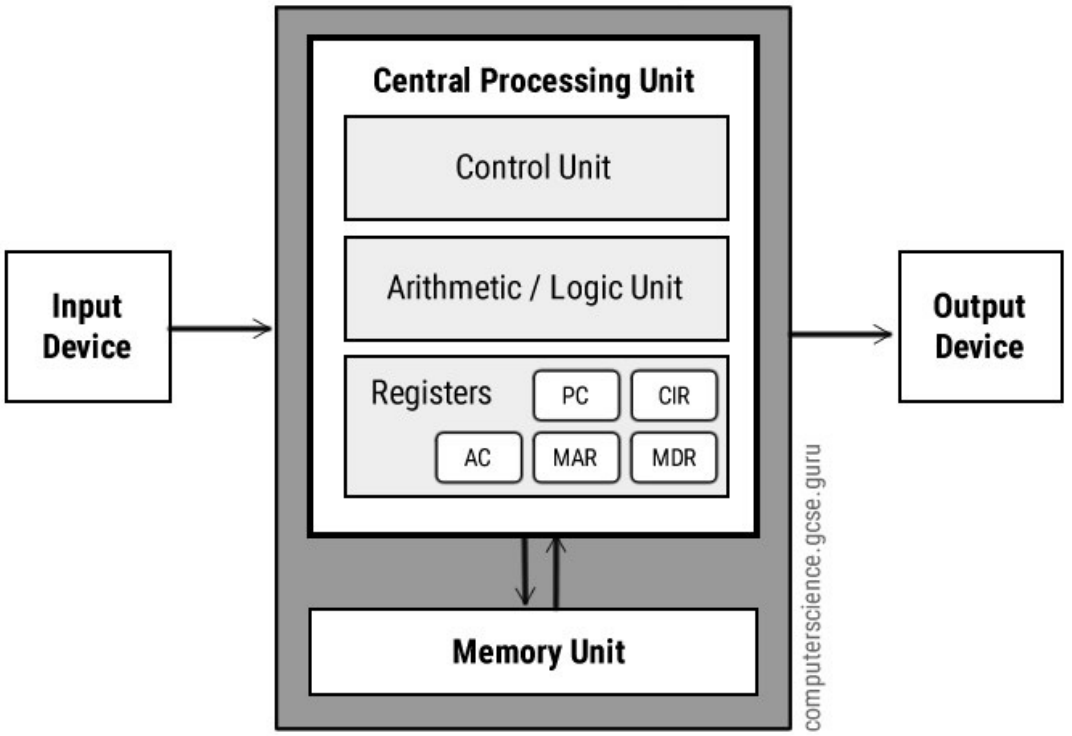
Motivation and Background

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

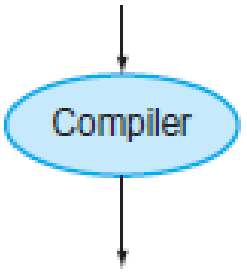
Program

- 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
```

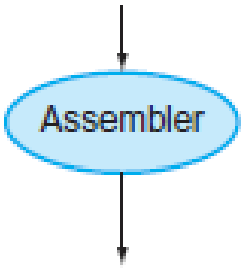
High-level language program (in C)

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



Assembly language program (for MIPS)

```
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
```



Binary machine language program (for MIPS)

```
00000000101000100000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
10001110000100100000000000000100
10101110000100100000000000000000
10101101111000100000000000000100
00000011111000000000000000001000
```

Process

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

Processors (CPUs)

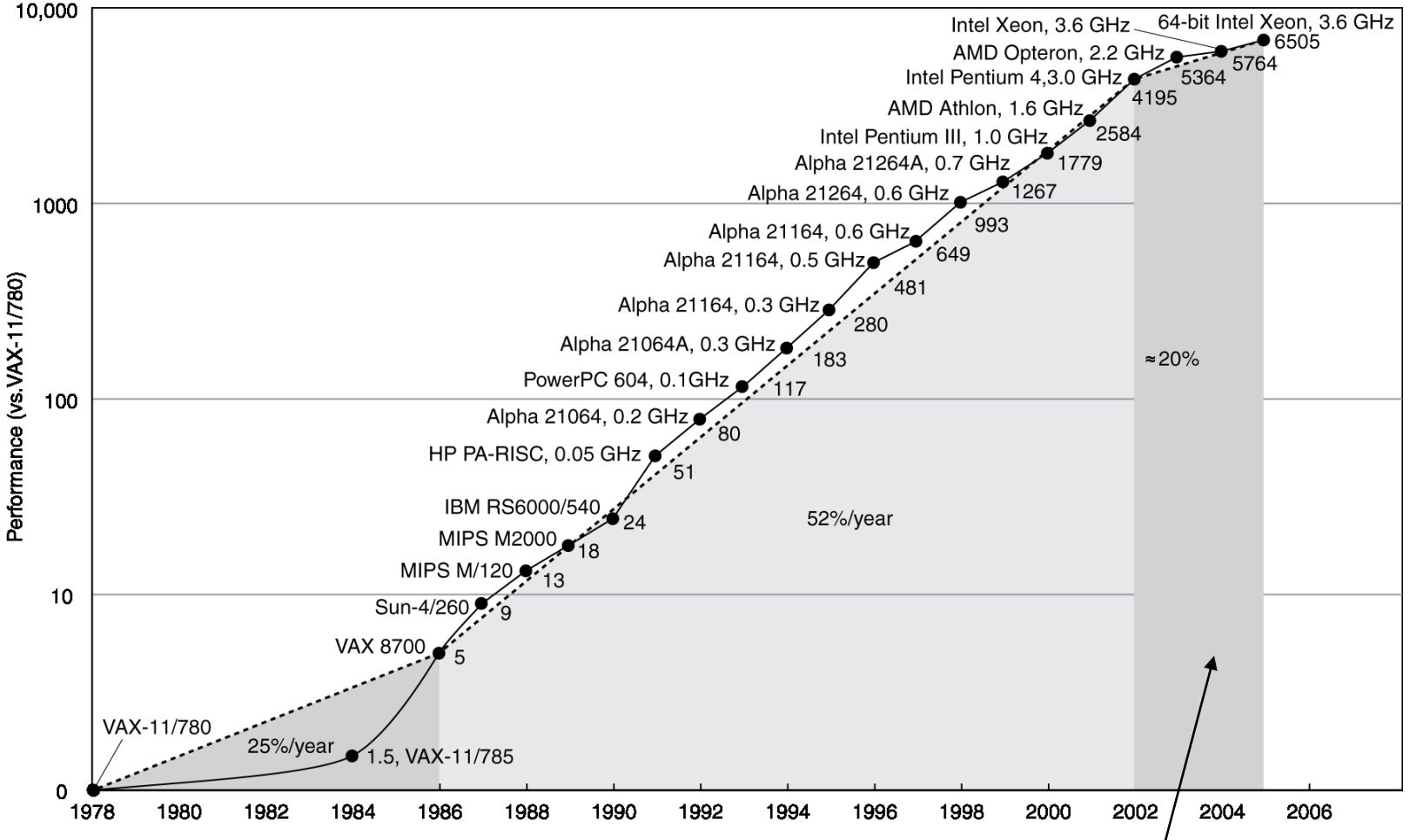
- The hardware that executes the program instructions

Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated Circuit	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000

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

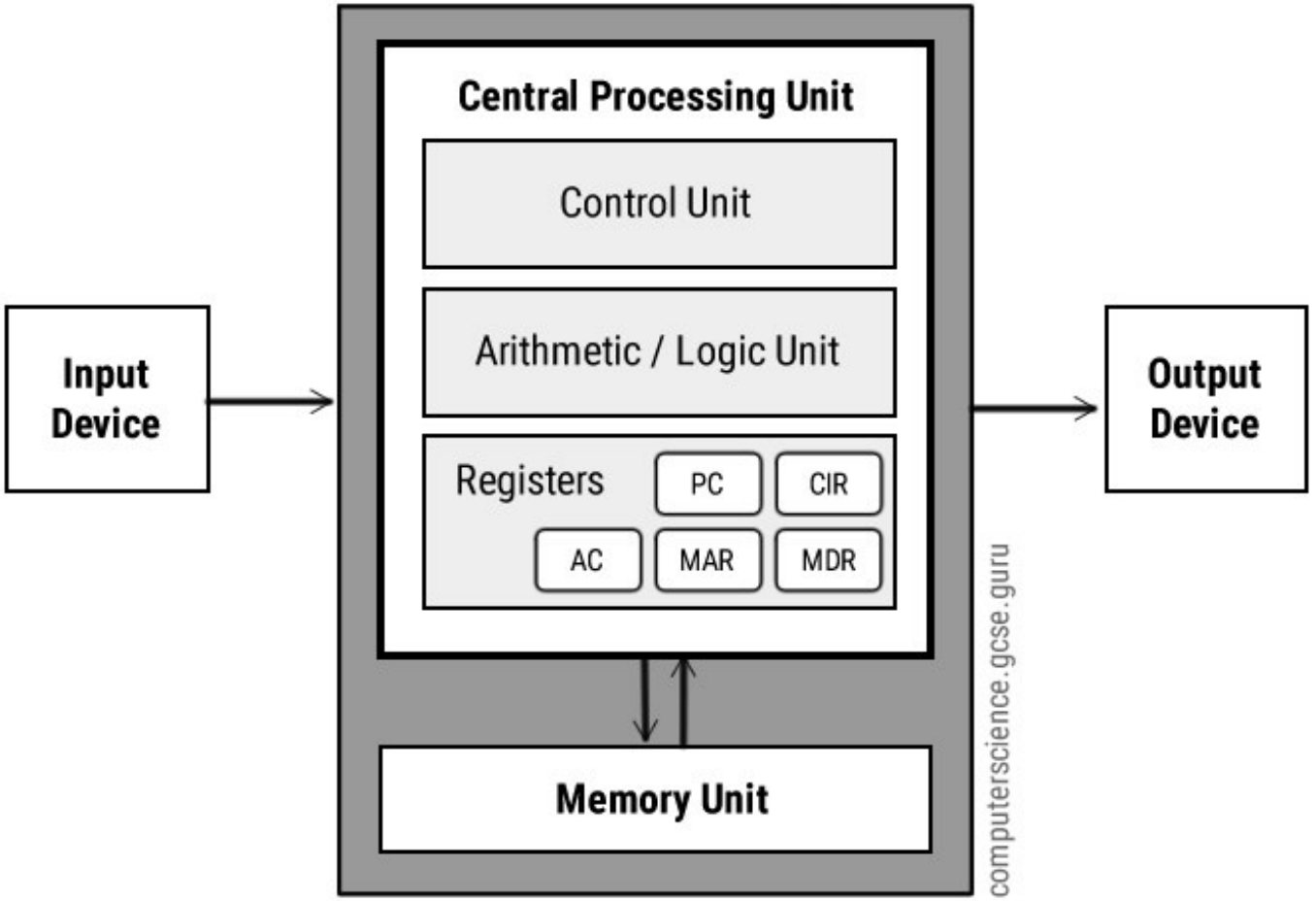
- faster processor
- more processors (needs parallelization)
- better software (algorithms)

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”

CS 230

Course Topics / Modules

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