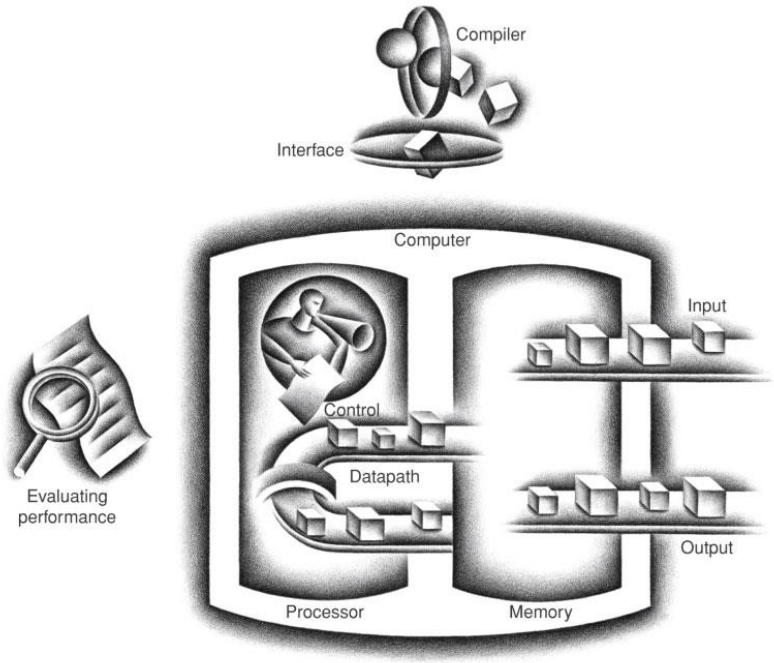
معماری کامپیوتر

فصل یک

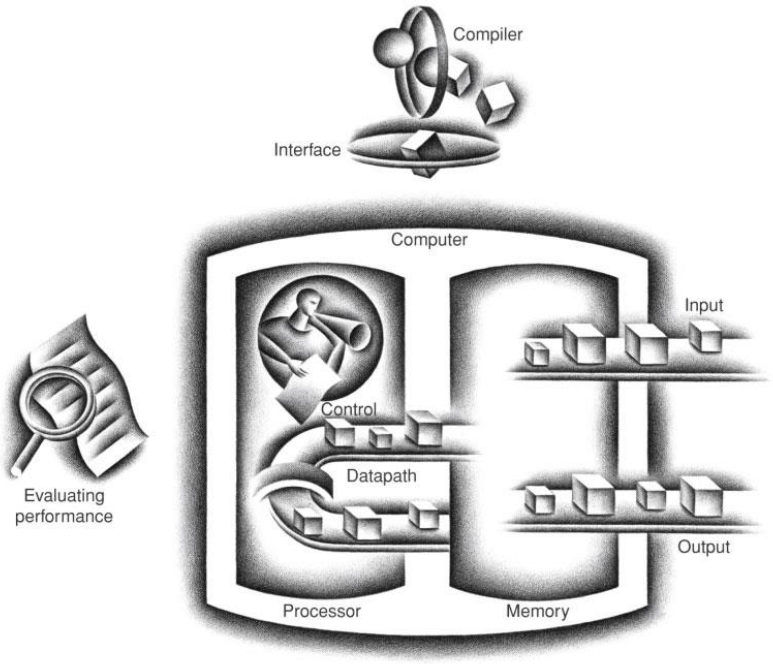
مقدمه



Computer Architecture

Chapter One

Introduction



Copyright Notice

Parts (text & figures of this lecture are adopted from:

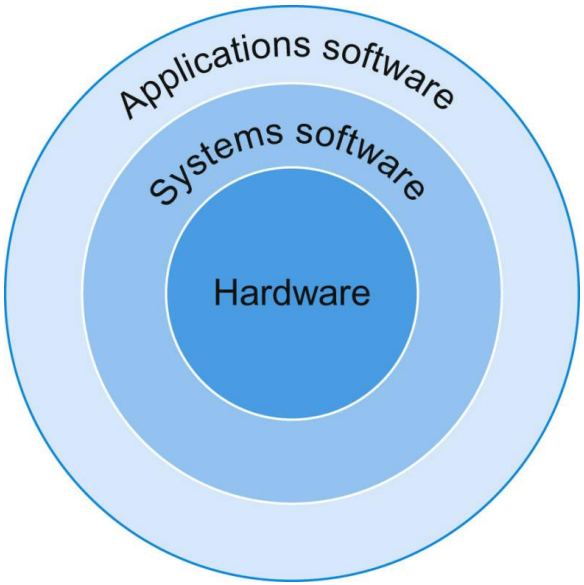
- M. M. Mano, C. R. Kime & T. Martin, “Logic & Computer Design Fundamentals”, 5th Ed., Pearson, 2015
- D. Patterson, J. Henessy, “Computer Organization & Design, The Hardware/Software Interface, MIPS Edition”, 6th Ed., MK Publishing, 2020
- A. Tanenbaum, “Structured Computer Organization”, 5th Ed., Pearson, 2006

Learned So Far

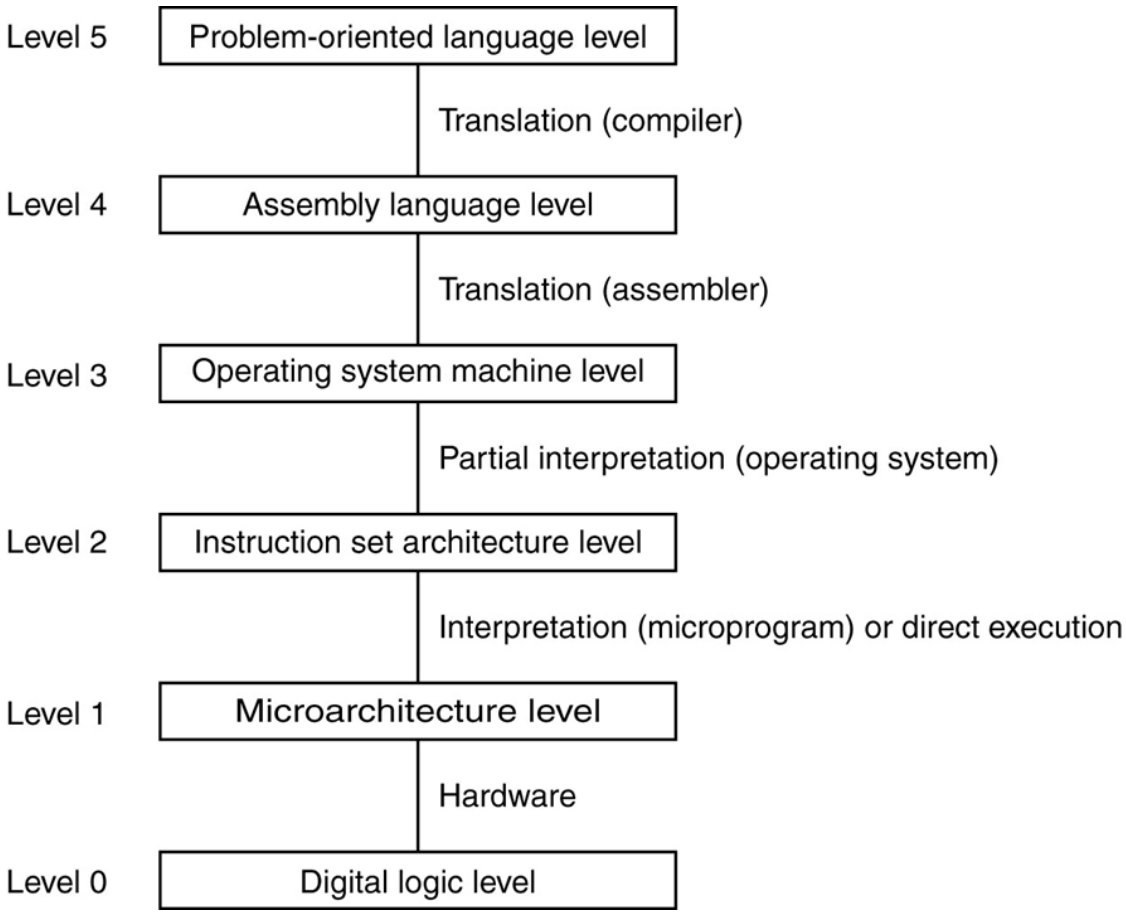
- Logic Design
  - Combinational & Sequential Circuit Design
- Computer Structure & Language
  - Computer Organization Overview
  - Instruction Set Architecture (ISA)
  - Assembly Language
  - Arithmetic Operations
  - Number Representation

Computer System

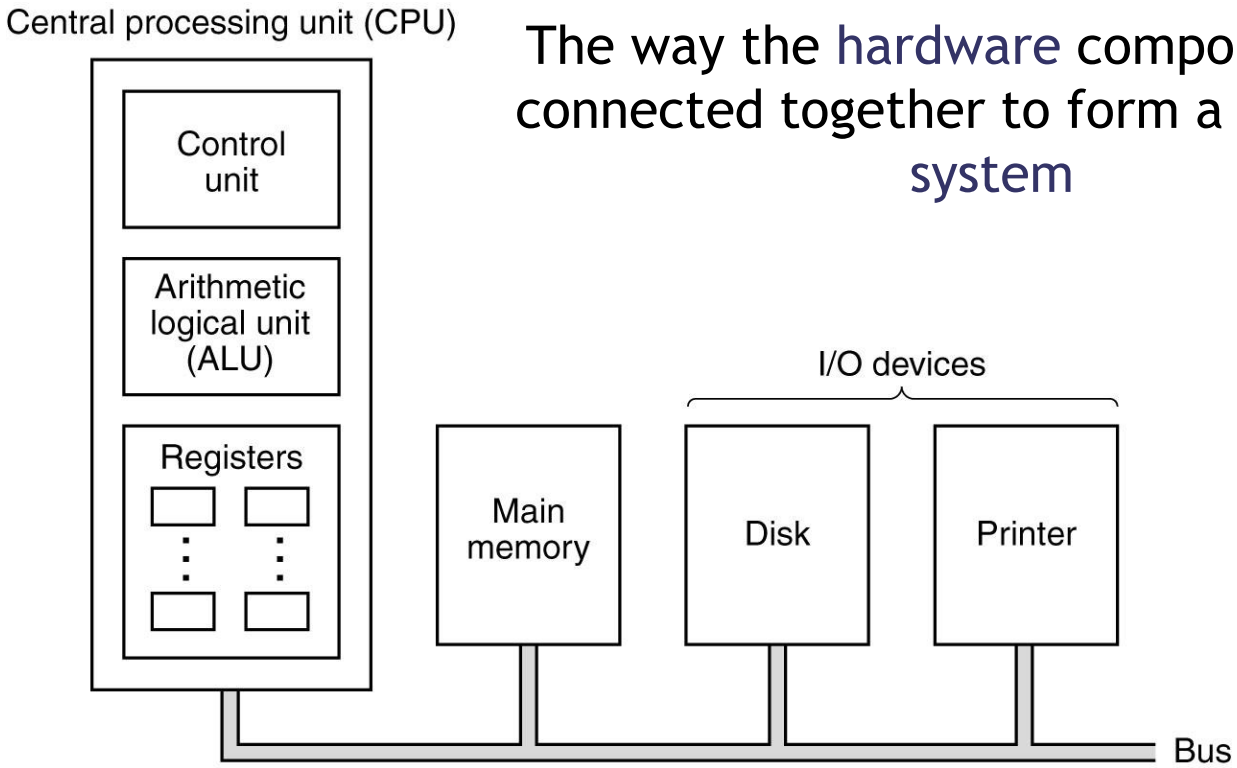
A computer system consists of hardware & software that are combined to provide a tool to solve problems (with best performance)



Computer System Abstraction



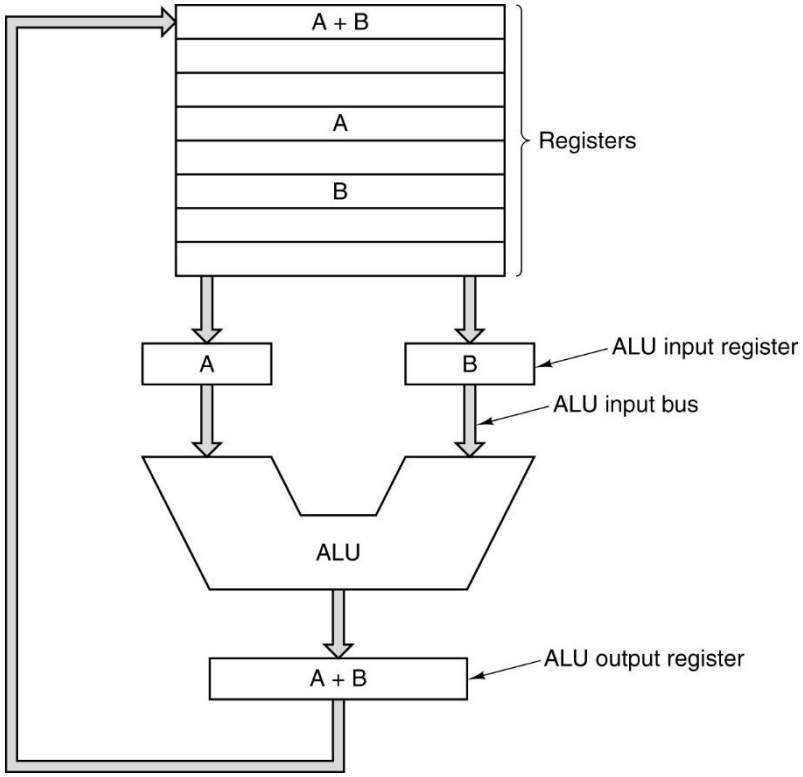
Computer Organization



The way the hardware components are connected together to form a computer system

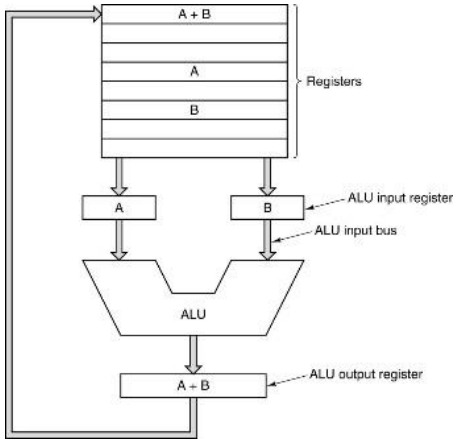
CPU Organization

The data path of a typical von Neumann machine



Data Path Cycle

- The process of running two operands through the ALU and storing the result
  - the heart of most CPUs
  - defines what the machine can do
- The faster the data path cycle is, the machine runs faster
- Modern computers have
  - multiple ALUs operating in parallel and/ or
  - specialized ALUs for different functions



Stored Program Concept

Both instructions and data are represented as a series of 0's and 1's (numbers) & stored in a linear memory array

Von Neumann Model

Instruction Execution Steps

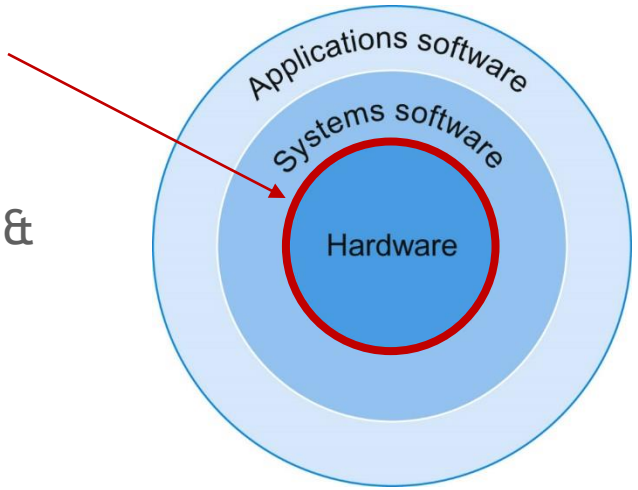
- Fetch – Decode – Execute Cycle:
  1. Next instruction is fetched from memory
  2. Program counter is incremented
  3. Instruction is decoded
  4. Operands are located and obtained
  5. Instruction is executed
  6. Results are stored in memory
  7. Control returns to step 1

Eight Design Ideas

- Design for Moore's Law
- Use abstraction to simplify design
- Make the common case fast
- Performance via parallelism
- Performance via pipelining
- Performance via prediction
- Hierarchy of memories
- Dependability via redundancy

Instruction Set Architecture (ISA)

- How the machine appears to a machine language programmer
- What a compiler outputs
  - ignoring operating-system calls & symbolic assembly language
- Specifies:
  - Memory Model
  - Registers
  - Available data types
  - Available instructions



Key ISA Decisions

- Instruction length?
- How many registers?
- Where operands reside?
- Which instructions can access memory?
- Instruction format?
- Operand format?
  - How many? How big?

ISA Classes

Register-Memory

```
ADD R1,A,B
ADD R2,C,D
MOV X,R1,R2
```

```
MOV R1,A
ADD R1,B
MOV R2,C
ADD R2,D
MUL R1,R2
MOV X,R1
```

X = (A + B) * (C + D)

Reg-Reg

```
LOAD   R1,A
LOAD   R2,B
LOAD   R3,C
LOAD   R4,D
ADD    R1,R1,R2
ADD    R3,R3,R4
MUL    R1,R1,R3
STORE  R1
```

Stack

```
PUSH A
PUSH B
ADD
PUSH C
PUSH D
ADD
MUL
POP X
```

Accumulator

```
LOAD   A
ADD    B
STORE  T
LOAD   C
ADD    D
MUL    D
STORE  X
```

Addressing Modes

- No address field in the instruction:
  - Implied Addressing: Operand is an implied register
  - Immediate Addressing: Operand is a constant value named in the instruction
- Register Addressing:
  - Direct: Operand is in a register named in the instruction
  - Indirect, autodec/inc: Operand address is in a register named in the instruction
- Memory Addressing:
  - Direct: Operand is in the memory, its address is in the instruction
  - Indirect: Operand is in the memory, the address of its address is in the instruction
- Register & Memory Addressing:
  - Relative Addressing: Effective address = (PC) + constant
  - Base Register Addressing: Effective address = (a base reg) + constant
  - Indexed Addressing: Effective address = (an index reg) + constant

Typical Instruction Set

- Arithmetic
- Logical
- Data Transfer
  - CPU ↔ Memory
  - CPU ↔ I/O
- Control
  - Conditional branch
  - Unconditional branch

RISC vs. CISC

- Complex Instruction Set Architecture
  - A large variety of instructions
- Reduced Instruction Set Architecture
  - Limited number of common instructions
- Hybrid Solution
  - RISC core & CISC interface
  - Taking advantage of both architectures

Modern (RISC) Design Principles

- Instructions should directly be executed by hardware
  - no or very rare interpretation by microinstructions
- Maximize the rate at which instructions are issued
  - by means of instruction level parallelism
- Instructions should be easy to decode
  - regular, fixed length, small number of fields
- Only loads and stores should reference memory
- Plenty of registers

Outlines

- Computer System
  - Computer System Abstraction
  - Computer System Organization
    - CPU Organization
      - Data Path Cycle
      - Stored Program Concept
      - Instruction Execution Steps
  - Computer System Organization (Eight Design Ideas)
- Instruction Set Architecture (ISA)
  - Key ISA Decisions
  - ISA Classes
  - Addressing Modes
  - Typical ISA
  - RISC vs CISC