CMSC 132: Computer Architecture
First Examination Reviewer

Computer Architecture in a Nutshell

Some concerns:

- How are commands/instructions represented?
- What commands/instructions are included?
- How much memory is needed?
- What is the priority: speed, efficiency or cost?

Computer History

Electronic computers as we know them today began in around 1940s to 1950s.

Figure 1: Electronic Numerical Integrator And Computer (ENIAC)

Computers were done for some specific purpose contrary to a general purpose computer we understand today.

- Decrypting/Encrypting messages
- Numeric computing
- Business computation

Electric Current Controllers

Vacuum Tube

- Diodes, triodes, etc.
- Vacuum tubes are:
  1. unreliable
  2. has high electric consumption
  3. gives off too much heat

Transistor

- less power consumption
- smaller compared with vacuum tubes
- greater reliability compared with vacuum tubes

Integrated Circuit

- smaller and lighter compared with transistors
- more reliable compared with transistors

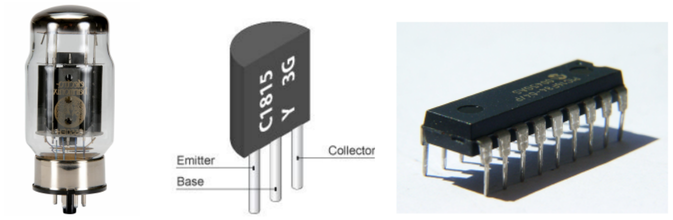

Figure 2: Images of Vacuum Tube, Transistor, and Integrated Circuit

Computer Architecture

According to Wikipedia (Accessed 2013), Computer Architecture "is a **set of disciplines that describes a computer system by specifying its parts and their relations**."

- Coined by IBM in 1964 for the use with the IBM 360.
- Used by **Amdahl**, **Blaauw** and **Brooks** used the term to refer to the programmer-visible portion of the instruction set.
- Machines with same architecture must be able to run the same software.
- Defined architecture as the *structure of a computer that a machine language programmer must understand to write a correct (timing independent) program for that machine.*
- Those who design computers must be proficient in computer architecture.
  1. Determine what attributes of a new computer are important
  2. Design the computer to maximize efficiency and performance considering power, availability and cost
  3. Define what instructions are supported, how much memory used, etc. (ISA)

Instruction Set Architecture (ISA)

- The "actual programmer-visible instruction set"
- Boundary between the hardware and the software
  1. More like an interface between the two
  2. Definition of memory mapping
- Defines what instructions can be understood by the processor

Seven Dimensions of ISA

Class of ISA

- General-purpose vs special-purpose register architectures
- Mostly general-purpose
- **Example:** 80x86 has 16 general-purpose registers and 16 that can hold floating point data

Memory Addressing

- Virtually all desktop and server computers, including the 80x86 and MIPS, use byte addressing to access memory operands
- Some (e.g., ARM, MIPS) require to be aligned.
- An access to an object of size s bytes at byte address A is aligned if A mod s = 0.
- Aligned operands are generally faster.

Addressing Modes

- Specify the address of a memory object, registers and constant operands
- Examples:

  Immediate (`add R1, 5`)

  Register (`add R1, R2`)

  Displacement (`add R1, 100(R2)`)

Types and Sizes of Operands

- 8-bit (ASCII character)
- 16-bit (Unicode character or half word)
- 32-bit (integer or word)
- 64-bit (double word or long integer)
- IEEE 754 floating point in 32-bit (single precision) and 64-bit (double precision).

Operands

The general categories of operations are data transfer, arithmetic, logical, control, and floating point.

Control Flow Instructions

The support for conditional branches, unconditional jumps, procedure calls, and returns. **Example:** `BE`, `BNE`

Encoding an ISA

- There are two basic choices on encoding: fixed length and variable length.
- Variable length instructions can take less space compared with fixed-length.

The Von Neumann Architecture

Proposed by **John von Neumann**, American-Hungarian which is a design architecture for an electronic digital computer; a single memory, stored program architecture based on von Neumanns paper.

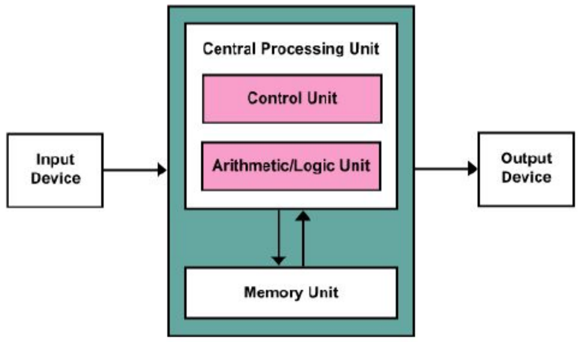

Figure 3: The Von Neumann Architecture

Arithmetic Logic Unit (ALU)

It is the digital circuit responsible for different arithmetic and logical operations.

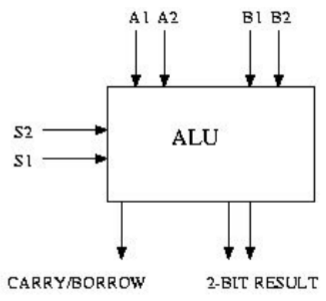

Figure 4: Arithmetic Logic Unit (ALU)

Control Unit (CU)

It is responsible for "controlling" other components (memory, ALU, etc.) behavior so that instructions are executed properly. Things that needed to be remembered:

- Instruction Register (IR)
- Program Counter (PC)
- Memory Address Register (MAR)

Memory

It is responsible for storing data.

Input/Output

They are devices that serve as an interface to communicate with the user

Architecture and Organization

Architecture

- Those attributes visible to the programmer
- Covers Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.

Organization

- How features are implemented
- Covers control signals, interfaces, memory technology.

Structure

Structure is the way in which components relate to each other.

Function

Function is the operation of individual components as part of the structure. All computer functions are:

- Data Processing
- Data Storage
- Data Movement
- Control

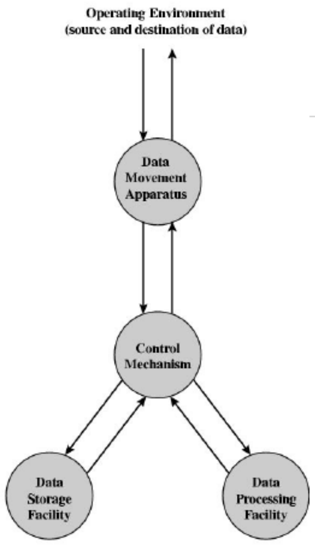

Figure 5: Function View

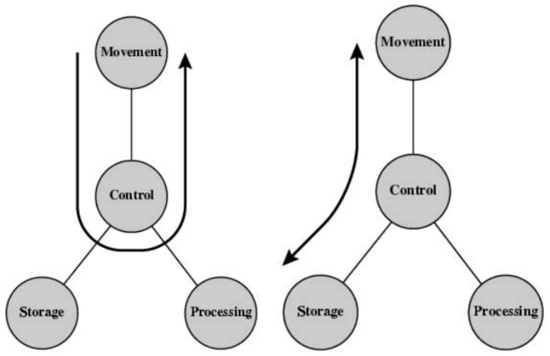

Figure 6: Data Movement and Storage

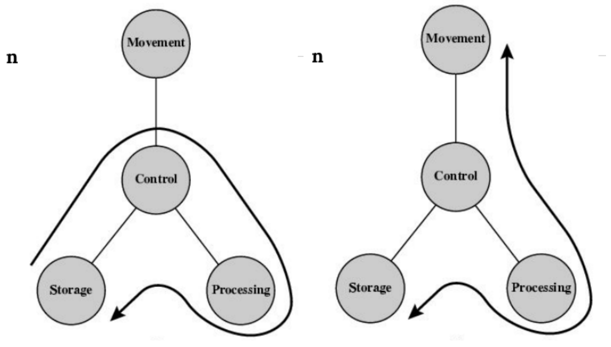

Figure 7: Processing to/from Storage and Processing from Storage to I/O

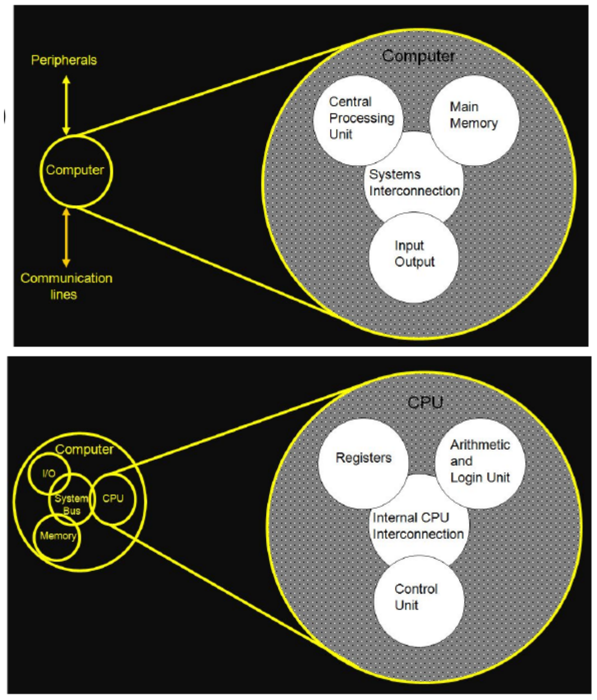

Figure 8: Structure: Top Level and CPU

The Fetch-Decode-Execute (FDE) Cycle

It is the process of getting program instruction, decoding the instruction through appropriate paths and executing the instruction.

Fetch

1. The address of the first instruction is pointed to by **PC**.
2. The **MAR** holds the address pointed to by **PC**.
3. **PC** (or the address held inside it) is incremented by one.
4. The instruction is fetched from memory; the data is copied to the **MDR**.

Decode

1. The instruction is determined and prepares appropriate registers for the instruction.
2. **Example:** Are there two operands or only one operand? What register(s) was/were used?

Execute

1. The instruction is coursed through the appropriate circuits.
2. The instruction is executed using the appropriate registers as input (if there are needed operands).

Instruction Sets

Reduced Instruction Set Computer (RISC)

It is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions, rather than a more specialized set of instructions often found in other types of architectures.

Complex Instruction Set Computer (CISC)

CISC are chips that are easy to program and which make efficient use of memory.

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory to memory: `LOAD` and `STORE` incorporated in instructions	Register to register: `LOAD` and `STORE` are independent instructions
Small code sizes, high cycles per second	Low cycles per second, large code sizes
Transistors used for storing complex instrutions	Spends more transistors on memory triggers

Machine Models

Stack-Based

- Number of Explicit Operands: 0
- Examples

  HP 3000

  ICL 2900

  Java Virtual Machine (modern)

  Intel x87 Floating Point Unit (modern)

Accumulator-Based

- Number of Explicit Operands: 1
- Common before 1980s

Register-Memory

- Number of Explicit Operands: 2 or 3
- Examples

  VAX

  80x86

Register-Register

- Number of Explicit Operands: 2 or 3
- Examples

  MIPS

  SPARC

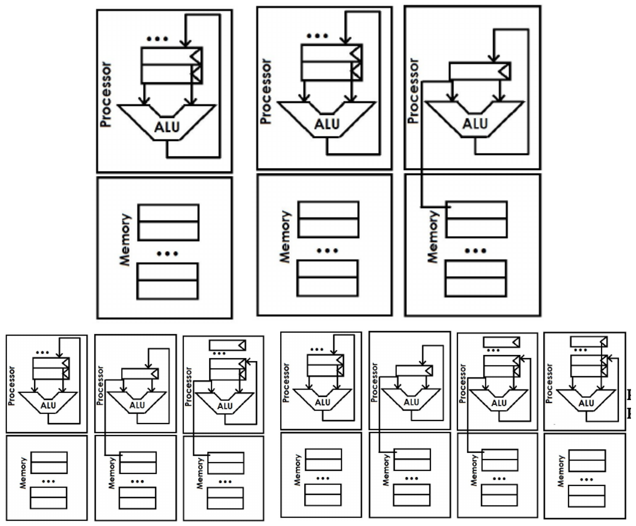

Figure 9: Stack-based, Accumulator-based, Register-Memory, and Register-Register ISA

Stack	Accumulator

`push A`
`push B`
`add`
`pop C`
	`load A`
	`add B`
	`store C`

Register-Memory	**Register-Register**

`load R1, A`
`add R3, R1, B`
`store R3, C`
	`load R1, A`
	`load R2, B`
	`add R3, R1, R2`
	`store R3, C`

Table 1: Instruction Representation of `C = A + B` in Different Machine Models