Computer Architecture and Operating System

Prof. Indranil Sengupta

Department of Computer Science and Engineering

IIT Kharagpur

Memory Addressing and Architecture Types

Overview of Memory Organization

- Memory is one of the most important sub-systems of a computer that determines the overall performance.
- Conceptual view of memory:
  - Array of storage locations, with each storage location having a unique address.
  - Each storage location can hold a fixed amount of information (multiple of bits, which is the basic unit of data storage).
- A memory system with *M* locations and *N* bits per location, is referred to as an *M x N memory*.
  - Both *M* and *N* are typically some powers of 2.
  - Example: 1024 x 8, 65536 x 32, etc.

Some Terminologies

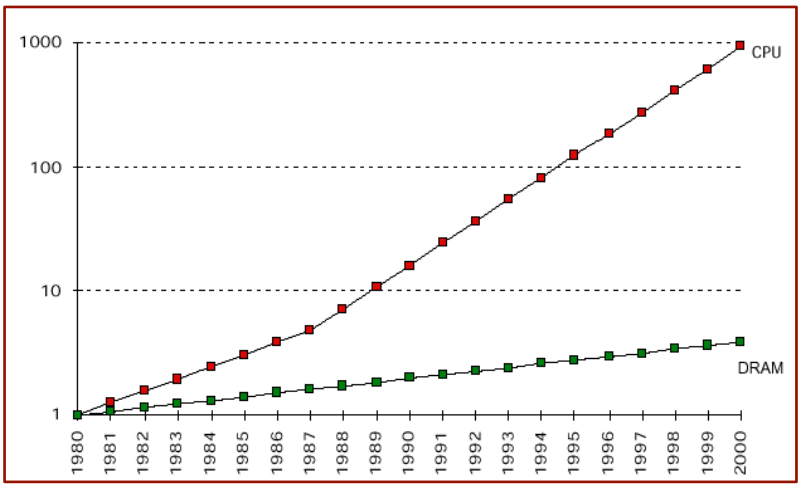
- *Bit*: A single binary digit (0 or 1).
- *Nibble*: Collection of 4 bits.
- *Byte*: Collection of 8 bits.
- *Word*: Does not have a unique definition.
  (Varies from one computer to another; typically 32 or 64 bits)

How is Memory Organized?

- Memory is often *byte organized*.
  - Every byte of the memory has a unique address.
- Multiple bytes of data can be accessed by an instruction.
  - Example: *Half-word* (2 bytes), *Word* (4 bytes), *Long Word* (8 bytes).
- For higher data transfer rate, memory is often organized such that multiple bytes can be read or written simultaneously.
  - Necessary to bridge the processor-memory speed gap.
  - Shall be discussed later.

Processor-Memory Performance Gap

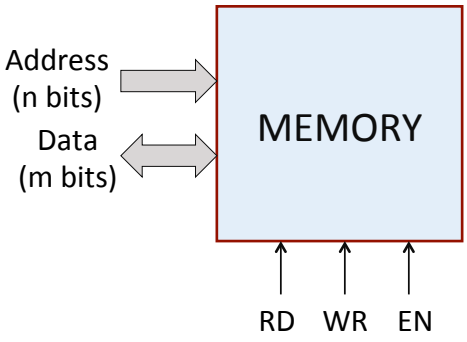
- With technological advancements, both processor and memory are becoming faster.
- However, the speed gap is steadily increasing.
- Special techniques are needed to bridge this gap.
  - Cache memory
  - Memory interleaving



How do we Specify Memory Sizes?

Unit	Bytes	In Decimal
8 bits (B)	1 or 2^0	10^0
Kilobyte (KB)	1024 or 2^{10}	10^3
Megabyte (MB)	1,048,576 or 2^{20}	10^6
Gigabyte (GB)	1,073,741,824 or 2^{30}	10^9
Terabyte (TB)	1,099,511,627,776 or 2^{40}	10^{12}
Petabyte (PB)	2^{50}	10^{15}
Exabyte (EB)	2^{60}	10^{18}
Zettabyte (ZB)	2^{70}	10^{21}

- If there are *n* bits in the address, the maximum number of storage locations can be 2^n.
  - For n=8, 256 locations.
  - For n=16, 64K locations.
  - For n=20, 1M locations.
  - For n=32, 4G locations.
- Modern-day memory chips can store several Gigabits of data.
  - Dynamic RAM (DRAM).



Address	Contents
0000 0000	0000 0000 0000 0001
0000 0001	0000 0100 0101 0000
0000 0010	1010 1000 0000 0000
⋮	⋮
1111 1111	1011 0000 0000 1010

An example: 2^8 x 16 memory

Some Examples

1. A computer has 64 MB (megabytes) of byte-addressable memory. How many bits are needed in the memory address?
   - Address Space = 64 MB = 2^6 x 2^{20} B = 2^{26} B
   - If the memory is byte addressable, we need 26 bits of address.
2. A computer has 1 GB of memory. Each word in this computer is 32 bits. How many bits are needed to address any single word in memory?
   - Address Space = 1 GB = 2^{30} B
   - 1 word = 32 bits = 4 B
   - We have 2^{30} / 4 = 2^{28} words
   - Thus, we require 28 bits to address each word.
   - If the memory is “byte addressable”, we would need 30 bits of address.

Byte Ordering Conventions

- Many data items require multiple bytes for storage.
- Different computers use different data ordering conventions.
  - Low-order byte first
  - High-order byte first
- Thus a 16-bit number 11001100 00101010 can be stored as either:

11001100	00101010

or

00101010	11001100

Data Type	Size (in Bytes)
Character	1
Integer	4
Long integer	8
Floating-point	4
Double-precision	8

Typical data sizes

- The two conventions have been named as:

a) Little Endian

- The least significant byte is stored at lower address followed by the most significant byte. Examples: Intel processors, DEC alpha, etc.
- Same concept followed for arbitrary multi-byte data.

b) Big Endian

- The most significant byte is stored at lower address followed by the least significant byte. Examples: IBM's 370 mainframes, Motorola microprocessors, TCP/IP, etc.
- Same concept followed for arbitrary multi-byte data.

An Example

- Represent the following 32-bit number in both Little-Endian and Big-Endian in memory from address 2000 onwards:

  01010101 00110011 00001111 11000011

Little Endian	
Address	Data
2000	11000011
2001	00001111
2002	00110011
2003	01010101

Big Endian	
Address	Data
2000	01010101
2001	00110011
2002	00001111
2003	11000011

Memory Access by Instructions

- The program instructions and data are stored in memory.
  - In *von-Neumann architecture*, they are stored in the same memory.
  - In *Harvard architecture*, they are stored in different memories.
- For executing the program, two basic operations are required.
  a) **Load**: The contents of a specified memory location is read into a processor register. *LOAD R1, 2000*
  b) **Store**: The contents of a processor register is written into a specified memory location. *STORE 2020, R3*

An Example

- Compute S = (A + B) – (C – D)

```
LOAD   R1,A
LOAD   R2,B
ADD    R3,R1,R2        // R3 = A + B
LOAD   R1,C
LOAD   R2,D
SUB    R4,R1,R2        // R4 = C – D
SUB    R3,R3,R4        // R3 = R3 – R4
STORE  S,R3
```

A typical program in **assembly language**.

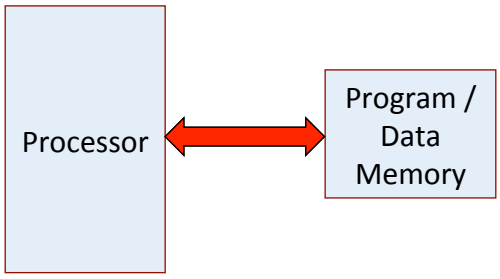
Assembly language instructions are machine dependent.

Classification of Computer Architecture

- Broadly can be classified into two types:
  a) Von-Neumann architecture
  b) Harvard architecture

Von-Neumann Architecture

- Instructions and data are stored in the *same* memory module.
- Suitable for most of the general purpose processors.
- Disadvantage:
  - The processor-memory bus acts as the bottleneck.
  - All instructions and data are moved back and forth through the pipe.



Harvard Architecture

- Separate memory for program and data.
  - Instructions are stored in program memory and data are stored in data memory.
- Instruction and data accesses can be done in parallel.
- However, the processor-memory bottleneck remains.