Universiti Malaysia PAHANG
Engineering • Technology • Creativity

BCN 1043

COMPUTER ARCHITECTURE & ORGANIZATION

By
Dr. Mritha Ramalingam

Faculty of Computer Systems & Software Engineering
mritha@ump.edu.my

AUTHORS

- **Dr. Mohd Nizam Mohmad Kahar** (mnizam@ump.edu.my)
- **Jamaludin Sallim** (jamal@ump.edu.my)
- **Dr. Syafiq Fauzi Kamarulzaman** (syafiq29@ump.edu.my)
- **Dr. Mritha Ramalingam** (mritha@ump.edu.my)

Faculty of Computer Systems & Software Engineering

BCN 1043

COMPUTER ARCHITECTURE & ORGANIZATION

Chapter 5 continues…

COMPUTER MEMORY

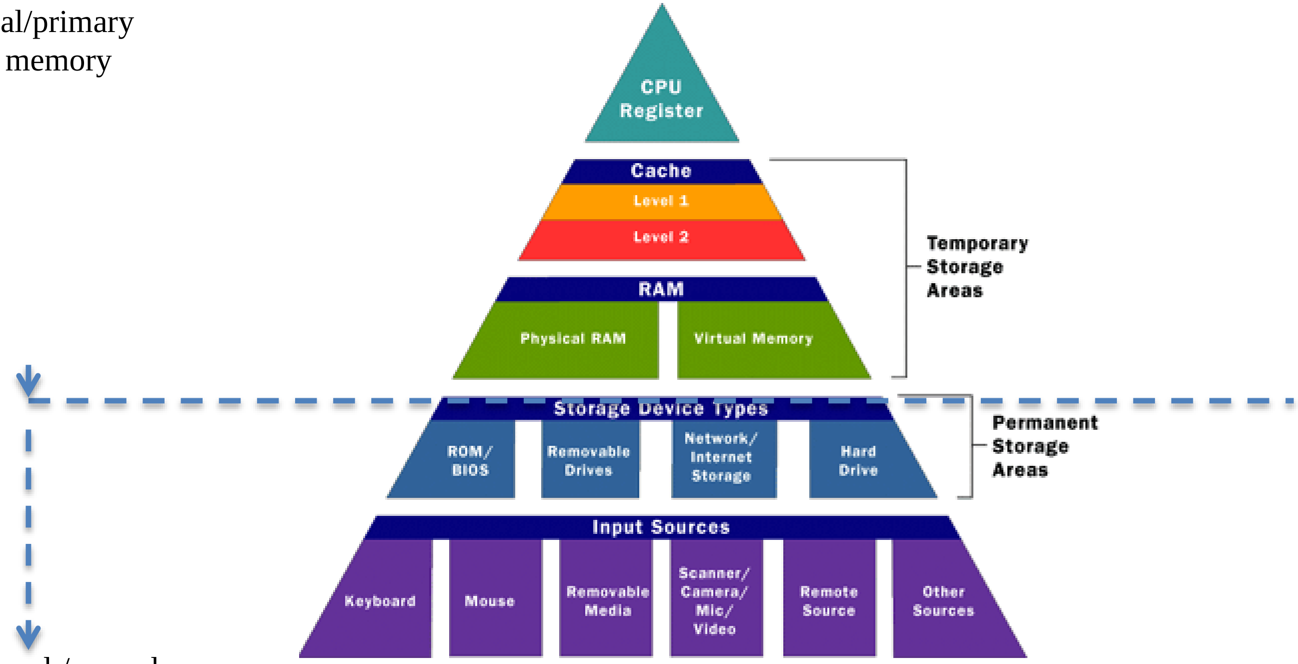
1. Storage System & Technology
2. Memory Hierarchy
3. Memory Organization and Operations
4. Cache Memories

Memory Hierarchy

- Major design objective of any memory system is
  - to provide adequate storage **capacity (how much?)**
  - at an acceptable level of **performance-access time(how fast?)**
  - at a reasonable **cost (how expensive?)**
- Trade-off among three key characteristics of memory: cost,capacity and access time.
- Interrelated way to meet this goal
  - Use a hierarchy of storage devices
- Characteristics of the memory hierarchy
  - Consists of distinct levels of memory components
  - Each level characterized by its size, access time, and cost per bit
  - Each increasing level in the hierarchy (go down memory hierarchy) consists of modules of larger capacity, slower access time, and lower cost per bit

Memory hierarchy

Internal/primary memory



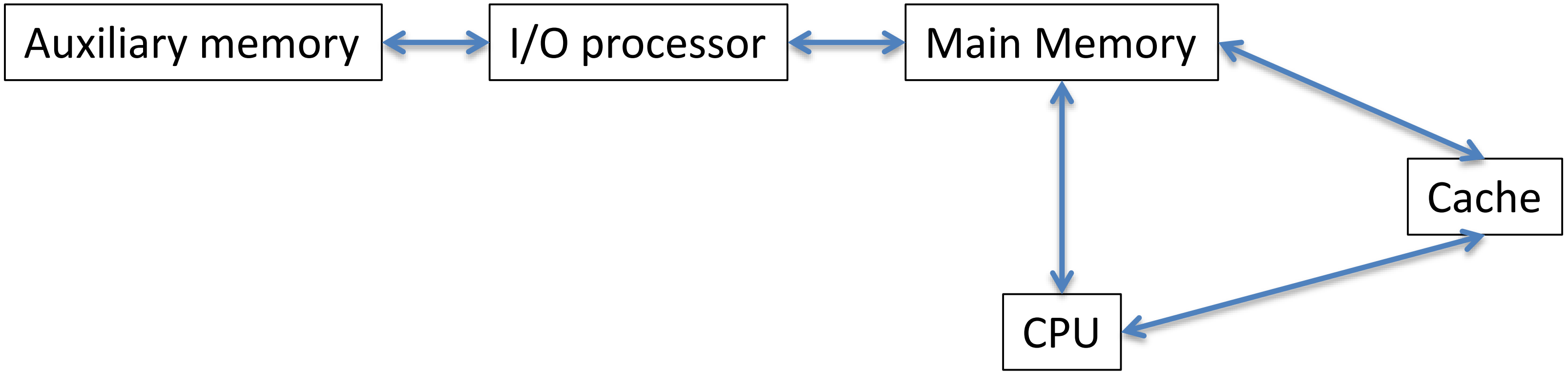
External /secondary memory

Source:http://s1.livrozilla.com

Memory Hierarchy

- The memory hierarchy system consists of all storage devices employed in a computer system from the slow by high-capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory

Memory Hierarchy



Volatile vs non-Volatile

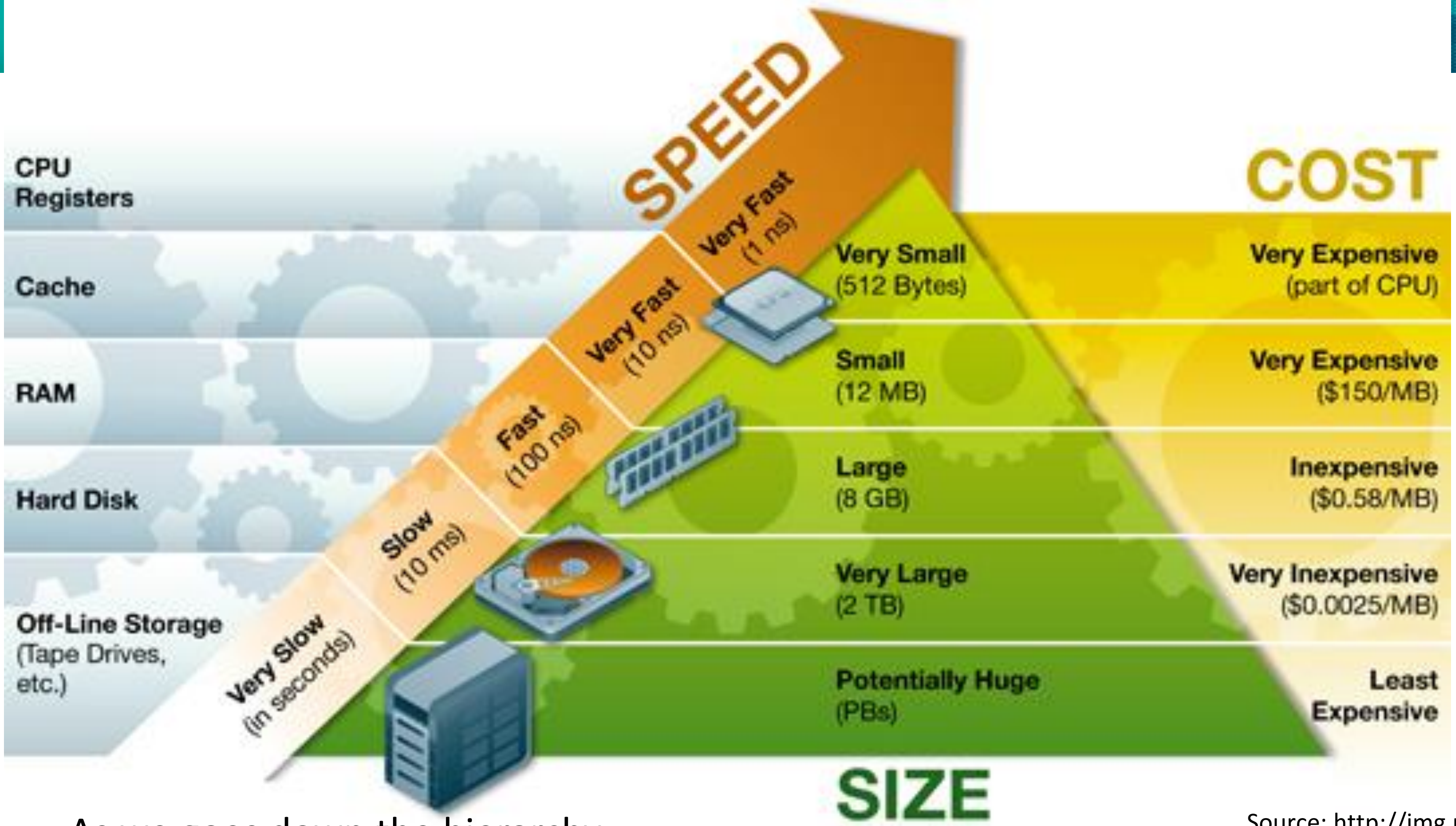
- **Volatile memory**: loss of electricity will cause the data in memory to be erased (i.e. computer shutdown/reboot).
  – Referred to as main or primary memory.

  Example: Random-access-memory (RAM), info that haven't saved is destroyed.

- **Non-volatile memory**: holds its data even when power is not turned on. All data stored in this type of memory will retain (even when computer is shutdown).
  – Referred to as secondary memory.

  Example: Hard drive or flash drive.

Memory - speed and cost



Source: http://img.mp.itc.cn

As we goes down the hierarchy:

a) Increasing capacity
b) Increasing access time
c) Decreasing cost per bit
d) Decreasing frequency of access of the memory by the processor

Why there is a memory hierarchy?

- Size/Capacity? Cost? Speed?
- Relationships in memory implementation:
  - Faster access time, greater cost per bit
  - Greater capacity, smaller cost per bit
  - Greater capacity, slower access time
- Therefore, computer system is equipped with a hierarchy of memory subsystems, some internal to the system (directly accessible by the processor) and some external (accessible by the processor via an I/O module).

Memory Organization

1. Error Checking

- Memory chips with built-in error-checking typically use a method known as **parity** to check for errors.
- **error-correction code** (ECC).
- ROM differ from RAM with no write control
- **Parity Checking**
  - Parity bits for every 8 bits of data
  - Parity can be divided into even parity and odd parity.

- Eg. for **Even Parity**
  - If the total number of 1s is odd, the parity bit is set to 1
  - E.g five 1s, the parity bit is 0
- **Odd parity**, the parity bit is set to 1 when total number of 1s in the byte are even
- Parity error can be caused by RAM chips that are unable to hold data reliably. This happens when the chips overheat or power falters.
- The problem with parity is that it discovers errors but does nothing to correct them.

Error-detecting and correcting

- Memories error → voltage spikes .
- Solution: Error-detecting or error-correcting → extra bits are added
- The Hamming distance between two strings of bits (binary integers) is the number of corresponding bit positions that differ. This can be found by using XOR on corresponding bits or equivalently, by adding corresponding bits (base 2) without a carry. For example, in the two bit strings that follow:

```
          A = 0 1 0 0 1 0 1 0 0 0
          B = 1 1 0 1 0 1 0 1 0 0
    A XOR B = 1 0 0 1 1 1 1 1 0 0
```

- The Hamming distance (*H*) between these 10-bit strings is 6, the number of 1's in the XOR string.

Hamming code algorithm

how to compute the check / parity bits for the above 8-bit data word Hamming code algorithm:

- Each check / parity bit operates on every data bit whose position number contains a 1 in the same bit position as the position number of that check bit. Thus,
- For calculating P_1 , since the bit position starts at 1, consider 1 bit, skip 1 bit, consider 1 bit, skip 1…. So, the XOR operation on the data bit positions 3, 5, 7, 9, and 11 (D1, D2, D4, D5, D7)

Bit positions	12	11	10	9	8	7	6	5	4	3	2	1
Check/ Parity bits					P_8				P_4		P_2	P_1
Data bits	D_8	D_7	D_6	D_5		D_4	D_3	D_2		D_1		
	0	0	1	1	P_8	1	0	0	P_4	1	P_2	P_1

Cont..

- $P_1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$

- Similarly for calculating,
- P_2, since the bit position starts at 2, consider 2 bit positions, skip 2 bits, consider 2 bits, skip 2….. So, XOR is performed on bit positions 3, 6, 7, 10, and 11

- $P_2 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$

- P_4, since the bit position starts at 4, consider 4 bit positions, skip 4 bits, consider 4 bits, skip 4….. So, XOR is performed on bit positions 5, 6, 7, and 11

- $P_4 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$

Cont..

- $P_{8\,,}$ since the bit position starts at 8, consider 8 bit positions, skip 8 bits, consider 8 bits, skip 8….. So, XOR is performed on bit positions 9,10, 11, 12

- $P_8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$

- So, check bits stored would be, $P_8\, P_4\, P_2\, P_1$ = 0111

Exercise

- Actual data: **1110**. Determine the code word?

- Actual data: **100101101011**. Determine the code word?

2. Byte ordering - little endian

- "Little Endian" - low-order byte of the number is stored in memory at the lowest address, and the high-order byte at the highest address.
- E.g, a 4 byte Long-Int

  Byte3 Byte2 Byte1 Byte0

  will be arranged in memory as follows:

Base Address+0	Byte0
Base Address+1	Byte1
Base Address+2	Byte2
Base Address+3	Byte3

- Intel processors (those used in PC's) use "Little Endian" byte order."

Byte ordering - Big endian

- Big Endian- high-order byte of the number is stored in memory at the lowest address, and the low-order byte at the highest address.
- E.g, a 4 byte Long-Int

  Byte3 Byte2 Byte1 Byte0

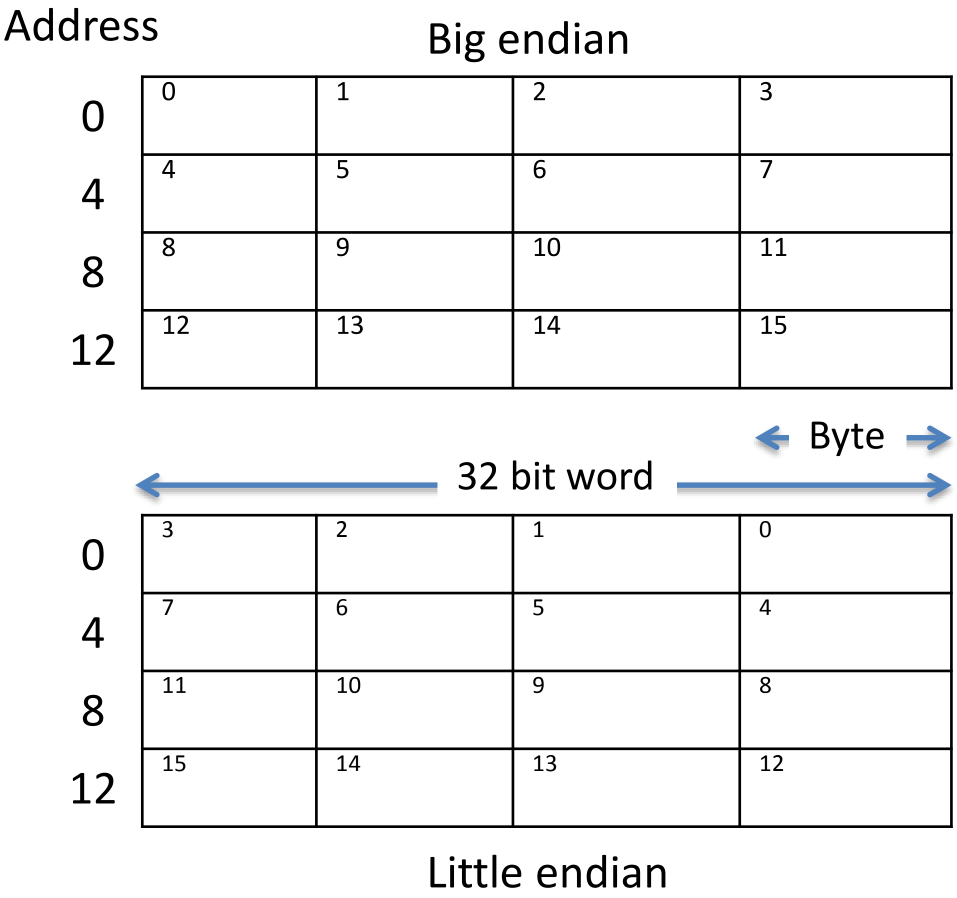
  will be arranged in memory as follows:

Base Address+0	Byte3
Base Address+1	Byte2
Base Address+2	Byte1
Base Address+3	Byte0

- Motorola processors (those used in Mac's) use "Big Endian" byte order.

example

- Common file formats and their endian order are as follows:
  - **Adobe Photoshop** -- Big Endian
  - **BMP (Windows and OS/2 Bitmaps)** -- Little Endian
  - **GIF** -- Little Endian
  - **IMG (GEM Raster)** -- Big Endian
  - **JPEG** -- Big Endian
  - **FLI (Autodesk Animator)** -- Little Endian
  - **MacPaint** -- Big Endian
  - **PCX (PC Paintbrush)** -- Little Endian
  - **QTM (Quicktime Movies)** -- Little Endian (on a Mac!)
  - **Microsoft RTF (Rich Text Format)** -- Little Endian
  - **SGI (Silicon Graphics)** -- Big Endian
  - **Sun Raster** -- Big Endian
  - **TGA (Targa)** -- Little Endian
  - **WPG (WordPerfect Graphics Metafile)** -- Big Endian (on a PC!)



Address

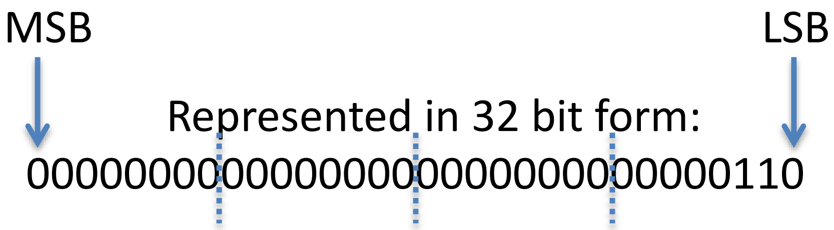
Big endian

Address				
0	0	1	2	3
4	4	5	6	7
8	8	9	10	11
12	12	13	14	15

Address				
0	3	2	1	0
4	7	6	5	4
8	11	10	9	8
12	15	14	13	12

Little endian

Example: number ‘6’



Represented in 32 bit form:

00000000000000000000000000000110

In big endian scheme:

The ‘110’ would be in byte 3 (or 7, 11 etc)

In little endian scheme:

The ‘110’ would be in byte 0 (or 4, 8 etc)

example

- For example, if a processor with an 8-bit data bus needs to store the 32-bit value 3A2B48CA$_{16}$
- Show the value in memory address 0 as in big endian and little endian scheme?

Address

Big endian

Address				
0	0	1	2	3
4	4	5	6	7
8	8	9	10	11
12	12	13	14	15

← 32 bit word → ← Byte →

Little endian

Address				
0	3	2	1	0
4	7	6	5	4
8	11	10	9	8
12	15	14	13	12

← 32 bit word → ← Byte →

- Tips: it uses four memory locations: one to store 3A$_{16}$, one for 2B$_{16}$, one for 48$_{16}$, and one for CA$_{16}$

Solution?

Question? 3A2B48CA$_{16}$

Big endian

Address				
0	0 3A	1 2B	2 48	3 CA
4	4	5	6	7
8	8	9	10	11
12	12	13	14	15

32 bit word

Byte

Little endian

0	3 3A	2 2B	1 48	0 CA
4	7	6	5	4
8	11	10	9	8
12	15	14	13	12

32 bit word

Byte

Problem on sending information

- What happen when we sent information between computer with different byte ordering? Big endian ←→ little endian

Address

Big endian

0	0 3A	1 2B	2 48	3 CA

32 bit word ← Byte →

Little endian

0	3 3A	2 2B	1 48	0 CA

32 bit word ← Byte →

- When sending big endian → little endian computer. The computer will read the value as: CA482B3A$_{16}$ wrong

0	3 CA	2 48	1 2B	0 3A

- When sending little endian → big endian computer. The computer will read the value as: CA482B3A$_{16}$ wrong

0	0 CA	1 48	2 2B	3 3A

- Solution: **no simple solution**. One way is to include a header in front of each data item telling what kind of data and how long it is.

Chapter 5

Chapter 5 will continue!