Ferhat Abbas Setif-1 University
College of Sciences/ Computer science department

First year computer science engineering
Computer organisation and architecture

جامعة فرحات عباس سطيف1 كلية
العلوم / قسم الاعلام الالي

السنة الأولى مهندس
هندسة الحاسوب

COMPUTER ARCHITECTURE AND ORGANIZATION

Tutorial 01 - Solution

Exercice 1 :

- **How many bits of information can be transmitted simultaneously on an electrical wire?**

  1 bit
- **How can 4 bits be transmitted simultaneously?**

  Using 4 electrical wires
- **Given an 8-bit data bus:**
  - What is the smallest binary number that can be represented? **00000000 = 0**
  - And the largest? **$2^8 - 1 = 255$**
  - Represent these numbers in base 2, 10, and 16: **$11111111_2 = 255_{10} = FF_{16}$**
- **Given a 2-bit address bus:**
  - How many different addresses can be represented? **2^2 different addresses (00, 01, 10, 11)**
  - And for 20 bits? **2^{20} different addresses**
- **What is the addressable space of a processor with 16-bit words and a 32-bit address bus?**

  Addressable space = 2^{32} * (16 bits / 8 bits) = 8 GB
- **How many "address" pins are there on a 1 MB memory module (with 8-bit words)?**

  $\log_2(2^{20}) = 20$ pins
- **In a Von Neumann architecture, where are the data and programs stored?**

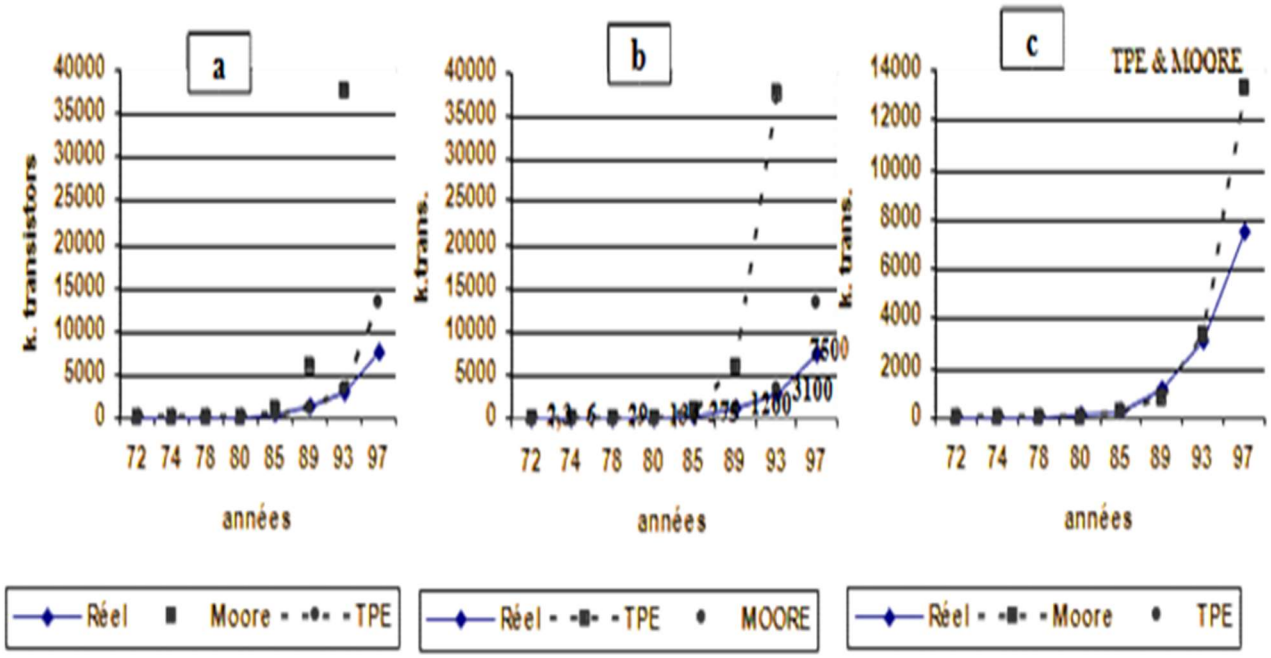
  Data and programs are stored in the same memory (Main Memory - MC).
- **In a Harvard architecture, where are the data and programs stored?**

  Data and programs are stored in two separate memories: Data Memory and Instruction Memory.
- **How does the CPU know where the next instruction to execute is located?**

  Through the Program Counter register (PC), also known as the Instruction Pointer (IP) for Intel.

Exercice 2 :

Année	1971	1979	1982	1985	1989	1993
Processeur	4004	8088	80286	80386	I486	Pentium
Nbre réel de transistors	2300	29000	134000	275000	1200000	3300000
Nbre de transistors calculé avec la relation de Moore	2300	58147	234206	930353	5 921 036	37 683 200
Nbre de transistors calculé avec la relation de Moore modifiée	2300	26 022	73 600	208 172	832 689	3 330 756



The most accurate of the three figures is (a):

- In (b), TPE cannot diverge so quickly because it was introduced to correct Moore's Law.
- In (c), both Moore's Law and TPE cannot have the same curve.

➔ The correct figure is (a), where Moore's curve diverges rapidly from reality, while the TPE curve converges slightly toward reality.

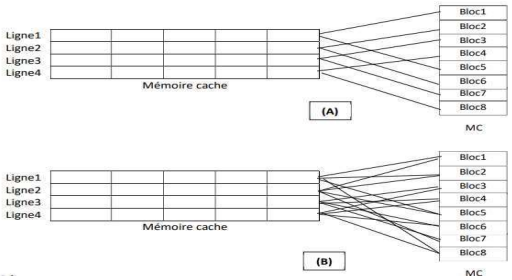
Exercice 3 :

Given the ASCII codes **48_{16}, 45_{16}, $4C_{16}$, and $4F_{16}$**, which correspond respectively to the characters:

H, E, L, O

The ASCII code for the word **HELLO** is **$48454C4C4F_{16}$**.

Exercice 4 :



(A) is DIRECT MAPPED

(B) is 2 WAYS SET ASSOCIATIVE

Exercice 5 :

$@_{MM}$ = Main Memory Address

8 Bits TAG			Line number on x bits (Index)				Offset Cache line			
bt-1	.....	bx+4	bx+3	.....	b5	b4	b3	b2	b1	b0

Calculate t, x, and n.

n = Number of cache lines = Cache Size / Line Length

- Cache Size = 256 KB
- Line Length = 128 bits = 16 B
- Line Offset = $\log_2$(16 B) = 4 bits

n = 256 KB / 16 B
n = 16K lines

x = Length of a cache line number = $\log_2$(Number of Cache Lines)
x = $\log_2$(16K lines) = $\log_2(2^{14}$ lines) = 14
➔ x = 14

t = Number of bits for the Main Memory Address managed by the cache
t = Line Offset + Cache Line Number Size + 14

t = 4 + 14 + 14 = 32 bits

Size of the Main Memory managed by the cache

Memory Size = 2^t bytes
Memory Size = 2^{32} bytes = 4 GB

Can two consecutive addresses be loaded into such a cache?

Yes, two consecutive addresses can be loaded into this cache, either in the same line or in two consecutive lines.

Let's assume we have two pieces of information, **Inf1** and **Inf2**, stored in **main memory (MC)** at consecutive addresses:

- **Case 1:**
  - **@Inf1 = $000001F_{16}$**
  - **@Inf2 = 0000020_{16}**
  - The two pieces of information will be stored in **two different lines**: **line 1 and line 2**.
- **Case 2:**
  - **@Inf1 = $000001E_{16}$**
  - **@Inf2 = $000001F_{16}$**
  - The two pieces of information will be stored in **the same line (line 1)**, but in **two different words**: **word E and word F**.

What is the effect of this organization on the ratio of cache hits to cache misses?

Let's assume we have a program that manipulates consecutive pieces of information spaced **256 KB apart**.

For example, consider three pieces of information **Inf1, Inf2, and Inf3** stored in three different **MC addresses**:

- **@Inf1 = $000001C_{16}$**
- **@Inf2 = $004001C_{16}$**
- **@Inf3 = $008001C_{16}$**

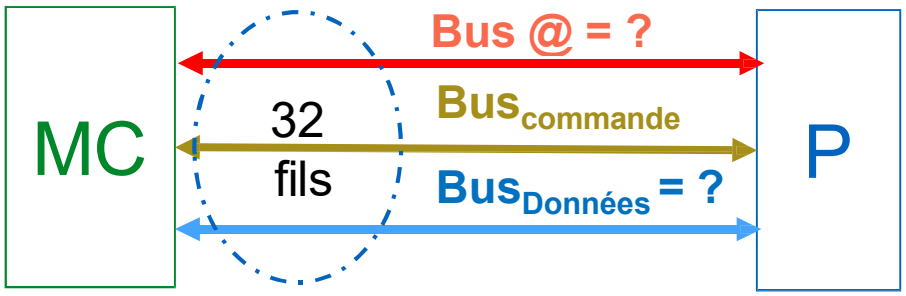
These three pieces of information will be mapped to the **same cache line and the same word**:
Line: $00\ 0000\ 0000\ 0001_2$ and **Word: C_{16}**.

This type of program results in the following behavior:

1. **Inf1** is loaded into the cache and used by the processor.
2. The processor then requests **Inf2**, which is **not found in the cache** (cache miss). It is fetched from **main memory** and stored in the cache.
3. When **Inf3** is needed, it is also **not found in the cache** (another cache miss), so it is fetched from **main memory** and stored in the cache.
4. This process repeats, causing frequent cache misses.

Since the processor **relies more on RAM than on the cache in this scenario**, the **cache miss/hit ratio is high (more misses than hits)**.

Exercice 6 :



Memory size = Number of locations * Memory word length
Number of locations = $2^{\text{Address Bus Width}}$

Address Bus Width = 32 - (Control Bus Width + Data Bus Width)

Bus Transfer Rate Formula:
Bus Transfer Rate = Bus Width * Clock Frequency

➔ Data Bus Width = Data Bus Transfer Rate / Clock Frequency
Data Bus Width = 2 GB/s / 1 GHz = 2 Bytes = 16 wires

Address Bus Width = 32 - 16 - 2 = 14 wires
Number of locations = 2^{14} locations

Memory Size = 16K * 1 byte = 16 KB