Memory

- Introduction
- Concepts and definitions
- Semi conducting main memory
- Cache memory
- Associative memory
- Shared memory
The memory contains programs that are executed in the computer and the data they work with.

The memory is a simple element, however, it has a great number of varieties, technologies, structures, services and costs.

Every computer has a hierarchy of memory elements where some of them are internally located while others are externally located.
Requirements of the memory

A memory system must have the following elements:

- **Means or support.** It must have an element where the different states that codify the information are stored.
- **Transducer.** It is an element that allows to turn an energy into another one, this is, to transform physical magnitudes to electrical (sensor) or electrical magnitudes to physical (actuator). Static memory and dynamic memory.
- **Addressing Mechanism.** It must provide a way to read/write information in the place and wished time.
Characteristic of the memory

◆ **Location**

Depending on where the memory is physically located there are three different types:

◆ Internal to processor memory. Hi-speed memory which is temporally used.
◆ Internal memory (Main Memory).
◆ External memory (Secondary Memory).
Characteristic of the memory

◆ Capacity

Amount of information that the memory system is able to store.
The memory capacity is measured using multiples of bit units.

1 bit 1 Mb = 1024 Kb = 2^{20} bits
1 nibble = 4 bits 1 Gb = 1024 Mb = 2^{30} bits
1 byte = 1 octeto = 8 bits 1 Tb = 1024 Gb = 2^{40} bits
1 Kb = 1024 bits = 2^{10} bits
Characteristic of the memory

◆ **Unit of transference**
   It is equal to the number of input and output data lines of the memory module.

Associated concepts:

◆ **Word.** The size of a word is generally equal to the number of bits used to represent an integer and the instruction length.
◆ **Addressable unit.** It is the minimum size that we can address in the memory.
◆ **Transference unit.** For the main memory it is the number of bits that can be read or written at the same time.
Characteristic of the memory

- **Access method**
  Way to locate the information into memory.

Types:

- SAM: Sequential Access Memory.
- DAM: Direct Access Memory.
- RAM: Random Access Memory.
- CAM: Content Addressable Memory.
Characteristic of the memory

**Speed**

In order to measure the yield three parameters are used:

- **Access time** ($T_A$)
  - RAM: time that passes from the moment at which a direction of the memory appears until the data either has been stored or is available for its use.

  Another one: time that is used in locating the read/write mechanism in the wished position.
Characteristic of the memory

- Memory cycle time ($T_C$)
  Time that passes since a read/write order operation occurs until another read/write order can be given.

![Diagram showing memory cycle time ($T_C$) and transaction time ($T_A$)]
Characteristic of the memory

- **Transference speed** ($V_T$).
  
  It is the speed at which data can be transferred to or from a memory unit.

  In the case of random access
  
  \[ V_T = \frac{1}{T_C} \]

  In the case of non-random access
  
  \[ T_N = T_A + \frac{N}{V_T} \]

  - $T_N$ Average time of read/write of $N$ bits
  - $T_A$ Access time
  - $N$ Number of bits
  - $V_T$ Transference speed (bits/second)
Characteristic of the memory

**Phisical device**

Memory systems in computers use different phisical devices.

Commonly used types are:

- Main memory uses semiconductive memory.
- For secondary memory:
  - Magnetic memory, discs, tapes, etc.
  - Optical memory.
  - Magnetic-optical memory.
Characteristic of the memory

◆ Physical facts

Main physical facts to consider to work with different types of memory are:

◆ Changeability. This property makes reference to the possibility to alter the content of a memory. ROM and RWM.
**Characteristic of the memory**

- Permanence of the information. Related to the durability of the information stored into memory:
  - Destructive reading memories. DRO (Destructive ReadOut) and NDRO (Non Destructive ReadOut).
  - Volatility. This characteristic references the possible destruction of the information stored into certain memory device when an electrical powerdown occurs. Memory can be volatile and non volatile.

- Static/dinamic storage. A memory is static if the information contained doesn’t vary along the time. A memory is dinamic if the information stored is lost along the time. To avoid data of being lost, an information recharge or refresh is needed. SRAM (Static RAM) and DRAM (Dynamic RAM).
Characteristic of the memory

**Organization**

It makes reference to the physical disposition of the bits to form words.

The organization depends on the type of memory.

For a semiconducting memory we find three types of organization:
- 2D Organization
- 2½D Organization
- 3D Organization
Characteristic of the memory

2D Organization

RAM with $2^m$ words of n bit each. The matrix of cells is formed by $2^m$ rows and n columns.

It is used in memories of reduced capacity.

Great rapidity of access.
Characteristic of the memory

- **2½D Organization**
  It uses 2 decoders with m/2 inputs and $2^{m/2}$ outputs.

- It requires less logical gates.
Characteristic of the memory

- **3D Organization**
  Similar to $2^{1\frac{1}{2}}$D organization but n bit word is stored in n planes and inside each plane positions x and y are selected.

More complicated design. Slower Access.
Memory hierarchy

Fundamental parameters that define the types of memory of the computer:

- Cost.
- Speed. The memory shouldn’t cause wait states to the processor.
- Capacity.

The ideal configuration: fast memory, great capacity and little cost.

It is not necessary to use a single type of memory, but to use different types from memory, this is, to use a memory hierarchy.
If we lower towards the inferior levels of the hierarchy it happens that:

- The cost by information unit (bit) diminishes.
- Capacity increases.
- Access time increases.
- The frequency of accesses to the memory by the CPU diminishes.

The principle of **reference locality** depends on the frequency of accesses.
Types of memory

Depending on the type of operation that a memory allows we can distinguish the following types:

- **Read Only memory**
  - **ROM (Read Only Memory).** Usually they are used in microprogramming of systems, library routines of frequent use, etc. The manufacturers use it when they produce components in a massive way.
  
  - **PROM (Programable Read Only Memory).** The writing process is carried out electrically and can be made either by the provider or by the client after the manufacturing of the original chip, unlike the ROM that is recorded when it is assembled. PROM allows just a single recording and it is more expensive than ROM.
Types of memory

- Read-mostly memory
  - **EPROM (Erasable Programable Read Only Memory)**. EPROM can be written several times using electrical power. However, using ultra-violet rays all the data can be erased. This type of memory is more expensive than PROM.
  - **EEPROM (Electricaly Erasable Programable Read Only Memory)**. Data can be erased at byte level in a selective way using electrical power. It’s more expensive than EPROM.
  - **Flash Memory**. Denominated thus by the speed with which it can be reprogrammed. It uses selective electrical erasure at block of bytes level. It’s cheaper than EEPROM.
Types of memory

- Read/Write memory
  
  **RAM (Random Access Memory).** Like the previous memory, it’s a random access memory. Main RAM types are:

  - Dynamic RAM (DRAM). Data is stored in a similar way as in the load of a condenser. Because it tends to unload is necessary to refresh it periodically. Simpler and cheaper than SRAM.
  
  - Static RAM (SRAM). Data is stored forming bistable so it is not necessary to refresh it. It’s faster and more expensive than DRAM.
### Types of memory

#### Summary chart

<table>
<thead>
<tr>
<th>Tipo</th>
<th>Clase</th>
<th>Borrado</th>
<th>Escritura</th>
<th>Volatilidad</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>Lectura/Escritura</td>
<td>Eléctricamente por bytes</td>
<td>Eléctricamente</td>
<td>Volátil</td>
</tr>
<tr>
<td>ROM</td>
<td>Sólo lectura</td>
<td>No</td>
<td>Mediante máscaras</td>
<td></td>
</tr>
<tr>
<td>PROM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPROM</td>
<td>Sobre todo lectura</td>
<td>Luz violeta, chip completo</td>
<td>Eléctricamente</td>
<td>No Volátil</td>
</tr>
<tr>
<td>FLASH</td>
<td></td>
<td>Eléctricamente por bloques</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EEPROM</td>
<td></td>
<td>Eléctricamente por byte</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Design

Memory chip
It is organized internally like a matrix of memory cells of nxm, where n is the number of words that can be stored in the memory chip and m is the number of bits by word.

4Kx8 memory
The interconnection of a memory chip is made through its pins:

- $n$ pins for the address bus, where $2^n$ words can be addressed.
- $m$ pins for the data bus indicating that in each access $m$ bits will be used.
- W/R (Write/Read). This pin indicates the type of operation: read or write. There are chips having one pin for writing WE (Write Enable) and other one for reading OE (Output Enable).
- CS (Chip Selection) or CE (Chip Enable). Selects the chip of memory to be accessed.
- $V_{CC}$. Power.
- $V_{SS}$. Ground.
For the correct operation of the memory it is necessary to incorporate an additional circuitry like decoders, multiplexors, buffers, etc.
**Design**

**Memory map**
Space that a computer is able to address.

<table>
<thead>
<tr>
<th>Direcciones en decimal</th>
<th>Direcciones en hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>.....</td>
<td></td>
</tr>
<tr>
<td>.....</td>
<td></td>
</tr>
<tr>
<td>$2^{16} - 1$</td>
<td>FFFF</td>
</tr>
</tbody>
</table>

Example of a 16 bit buffer computer.
Design

The physical implementation of the memory map is made using one or several chips of memory.

In the market we can find different memory chips configuration:

zKx1, zKx4, zKx8, zKx16, zKx32, zMx1, zMx4, zMx8, zMx16, zMx32, etc. where z is a multiple of 2.

This way, for instance, a 1Kx8 chip indicates that it can store 1024 words of 8 bit each one.
Design

Example 1:
If we want to design a 128 Kword main memory.

(1) How many 32Kx8 memory chips do we need if we suppose an 8 bit word?

(2) How many 64Kx4 memory chips do we need if we suppose an 8 bit word?
Design

Solution 1:

(1) We need to address 128K from 32K so we will need 4 chips. As the length of the word is equal to the content of each chip address we won’t be needing anymore.

(2) To address 128K we will need 2 chips. With both chips we will have a 128Kx4 memory so we will need another 2 chips to reach the 128Kx8 desired memory.
If we want to design an n bit memory and we have t bit chips we will need n/t parallel disposed chips to reach the desired word width.

Example: Let’s suppose we want to design a 64Kbyte (n=8) memory and we have 64Kx4 (t=4) chips, then we will need 2 chips (8/4). We can observe that there’s 1 chip row and 2 chip columns.
If we want a cK word capacity and we have zK capacity chips, we will need c/z chips to reach the desired capacity.

Example: We want to design a memory with 64Kbytes and we have 32Kx8 chips, then we will need 2 chips. When line $A_{15}$ is at 1 it enables the upper chip, when it’s at 0 it enables the lower chip. In this interconnection we can observe that there’s 2 chip rows and 1 chip column.
Example 2:
Let’s obtain the memory map and the connections diagram of a 16 bit computer that can address 1 Mword and has 128Kwords installed from 64Kx1 chips.

1. We must obtain the number of bits of the address bus.
2. Figure out the number of bits needed to address the memory chip we’re going to use.
3. Calculate the number of chips we need.
4. Obtain the number of bits of the address bus that allows to select the memory chips.
5. Draw the connections diagram by the selection logic.
Solution 2:

(1) We must obtain the number of bits of the address bus.

As they say that it can address 1Mword, we can see that it’s a 20 bit bus ($1M = 2^{20}$).

(2) Figure out the number of bits needed to address the memory chip we’re going to use.

As it’s a 64K memory chip, we will need 16 bits ($64K=2^{16}$). The bits we will use to address the memory chip are the smaller weight ones, so in this particular case $A_{15}A_{14}...A_1A_0$. 
Design

Solution 2:

(3) Calculate the number of chips we need.

As we want 128Kx16 we will need 16 chips to obtain a complete word (16 bit). With these first 16 bits we will have 64Kx16 so we need another 64Kx16, this is, 16 chips more. In conclusion, we will need 32 chips of 64Kx1 to store 128Kx16.
Design

Solution 2:

(4) Obtain the number of bits of the address bus that allows to select the memory chips.

As we have 2 rows of 16 chips each, we will need 1 bit to difference one row from another. So we will use the bit $A_{16}$ to select the memory chips. The rest of the addresses will be used for further computer memory extensions.
### Solution 2:

(4) Obtain the number of bits of the address bus that allows to select the memory chips.

#### Diagram

- **20 bits**
  - $A_{19}A_{18}A_{17}A_{16}A_{15}A_{14}...A_0$

#### Addressing

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Address Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-15</td>
<td>00000</td>
</tr>
<tr>
<td>16-17</td>
<td>0FFFF</td>
</tr>
<tr>
<td>18-20</td>
<td>1FFFF</td>
</tr>
</tbody>
</table>

#### Memory Map

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>Fila 0: 16 chips de 64Kx1</td>
</tr>
<tr>
<td>0FFFF</td>
<td>Fila 1: 16 chips de 64Kx1</td>
</tr>
</tbody>
</table>

#### Table

<table>
<thead>
<tr>
<th>Address (A19 A18 A17 A16 A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0)</th>
<th>Memory Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>Fila 0: 16 chips de 64Kx1</td>
</tr>
<tr>
<td>0FFFF</td>
<td>Fila 1: 16 chips de 64Kx1</td>
</tr>
</tbody>
</table>

---

**Unit 4. Memory** 38
Solution 2:

(5) Draw the connections diagram by the selection logic.

Design
CPU memory is faster than main memory.

The ideal would be that the main memory was the same technology than the registries of the CPU, but due to its high cost the tendency is to find intermediate solutions.

A solution would be to take advantage of the locality principle and to place a very fast memory between the CPU and the main memory the way the CPU accedes more times to that memory than it does to main memory.

This very fast memory will have to be small so the costs won’t be excessive. This memory is denominated cache memory.
The way cache memory works is based on the transference of parts (blocks) between the main memory and the cache memory and the transference of words between cache memory and the CPU.
- $2^n$ words main memory is organized in $M$ blocks of fixed length ($K$ words/block) where $M = 2^n/K$ blocks.
- Cache memory is divided in $C$ lines or partitions of $K$ words ($C << M$).
Concept

Rate of success

- When the CPU looks for a memory word and it finds it in the cache memory we name it a hit. If the word is not in the cache memory a miss is entered.

- The relation between the number of successes and the total number of references to memory (hits + misses) is the rate of success. Well designed systems usually obtain a rate of success of 0.9.

\[
Tasa \ de \ acierto = \frac{N^\circ \ aciertos}{N^\circ \ referencias}
\]
Design parameters

Cache memory size

It raises a certain commitment:

- It should be small enough to make the average cost by bit of stored data in the computer's internal memory be close to the one of the main memory.
- It should be big enough to make the total average access time as close as possible to the one of the cache memory.

According to empirical studies, it is suggested that the size of a cache memory varies from 1Kb to 1Mb.
Design parameters

Block size

When the block size is increased from very small values the rate of success initially will increase. From a certain size of the block the rate of success begins to diminish.

Two effects arise:

- The bigger the size of the block is, less blocks will fit in the cache memory and more times the replacement of blocks algorithm will be executed.

- When the block size increases, each new word added to the block will be at a bigger distance from the word required from the CPU, therefore it is less probable that it would be needed in the short term.

A size of 4-8 addressable units is suggested.
**Design parameters**

**Cache memory levels**

- **Internal cache. Level 1.** It is physically located in the same chip as the processor. The accesses to this cache are very fast. The capacity of this cache is quite small.

- **External cache. Level 2.** It is physically located outside the processor so it will be slower than level 1 cache but will remain faster than main memory. Being outside, the capacity could be bigger than level 1 cache.
Design parameters

Cache content

A cache memory containing both, data and instructions, has the following advantages:

- It has a higher rate of success because it levels the load, this is, if an execution pattern implies more pick up of instructions than data, cache memory will tend to be filled in with instructions and vice versa.
- It’s only needed to implement and design one cache memory, therefore the cost will be more reduced.

Nowadays, the parallel execution of instructions is used and so is the design pipelining in which the use of two cache gives better benefits. Advantage: It eliminates the competition between the instructions processor and the unit of execution.
Design parameters

Writing strategy

Before a block which is in the cache can be replaced, it is necessary to know if it’s been modified or not (clean block and modified or dirty block).

- Write through

All writing operations are executed in both, cache memory and main memory, which makes sure that the contents are always valid. The main disadvantage is that it generates a lot of traffic with the main memory and it can originate a bottle neck.
Design parameters

Writing strategy

- Write back

  Writing is only executed in cache memory. Associated to each cache line there is a modification bit. When a writing order is executed in cache memory, the modification bit is put to 1. When a cache line has to be replaced and it’s at 1, writing will be made in the main memory line while if it’s at 0 it won’t be made.

  Disadvantage: I/O modules are forced to access the main memory through the cache memory. This complicates the circuitry and generates a bottle neck.

  Advantage: It uses less main memory bandwidth making it suitable for its use in multiprocessors.

Problem of data coherence.
Design parameters

Correspondence function

Due to the existence of less lines than blocks, an algorithm (function) to make main memory blocks correspond to cache memory lines is needed.

- Direct correspondence. Main memory block 12 can be only stored in line 4 (=12 mod 8).
- Associative correspondence. Storage can be made in any line.
- Associative correspondence by sets. Storage can be made in any line of the set 0 (=12 mod (8/2)).

<table>
<thead>
<tr>
<th>Número Línea</th>
<th>Directa</th>
<th>Asociativa</th>
<th>Asociativa por conjuntos</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>Conjunto 0</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Correspondence function

- Next we are going to describe the three techniques before enumerated.
- In each case we will see the general structure and a concrete example.
- For the three cases, we are going to work with a system with the following characteristics:
  - Cache memory size is 4Kb.
  - Data is transferred between main memory and cache memory in 16 bytes (K) blocks. This indicates that cache memory is organized in 256 (4096/16) lines (C).
  - Main memory is 64Kb, so addresses bus is 16 bits ($2^{16} = 64K$). This indicates that main memory is composed by 4096 blocks (M).
Design parameters

Direct correspondence
It consists of making correspond each block of main memory to only one line of cache memory. Correspondence function is expressed by the following function:

Cache L. Number = M.M. B. Number mod C.M. number of lines
Design parameters

Direct correspondence

Each Main memory address is divided in three different fields:

\[
\begin{array}{c|c|c}
\text{Etiqueta} & \text{Línea} & \text{Palabra} \\
\hline
s \text{ bits} & w \text{ bits} \\
\end{array}
\]

where

- \( w \text{ bits} \) = Identifies each word inside a block
- \( s \text{ bits} \) = Identifies the block number
Design parameters

Direct correspondence

The use of the part of an address as a line number provides a unique allocation of each block of main memory in cache memory.
Design parameters

Direct correspondence

Example:

Memoria Principal

<table>
<thead>
<tr>
<th>Dirección</th>
<th>Dato</th>
<th>Memoria Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>34</td>
<td>0</td>
</tr>
<tr>
<td>0002</td>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>000F</td>
<td>78</td>
<td>0</td>
</tr>
<tr>
<td>F800</td>
<td>AB</td>
<td>1</td>
</tr>
<tr>
<td>F801</td>
<td>0F</td>
<td>1</td>
</tr>
<tr>
<td>F802</td>
<td>FF</td>
<td>1</td>
</tr>
<tr>
<td>F80F</td>
<td>54</td>
<td>1</td>
</tr>
<tr>
<td>FFF0</td>
<td>FF</td>
<td>1</td>
</tr>
<tr>
<td>FFFD</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>FFFE</td>
<td>33</td>
<td>1</td>
</tr>
<tr>
<td>FFFF</td>
<td>55</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Etiqueta</th>
<th>Línea</th>
<th>Palabra</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bits</td>
<td>8 bits</td>
<td>4 bits</td>
</tr>
</tbody>
</table>

Etiqueta | Datos | Nº |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>123456........78</td>
<td>0 (00)</td>
</tr>
<tr>
<td>......</td>
<td>...............</td>
<td>1 (01)</td>
</tr>
<tr>
<td>......</td>
<td>...............</td>
<td>128 (80)</td>
</tr>
<tr>
<td>F</td>
<td>AB0FFF........54</td>
<td></td>
</tr>
<tr>
<td>......</td>
<td>...............</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>FF........213355</td>
<td>255 (FF)</td>
</tr>
</tbody>
</table>

4 bits | 16 bytes | 8 bits
Design parameters

Direct correspondence

- The direct correspondence technique is simple and not very expensive to implement.

- Disadvantage: there is a concrete position of cache memory for each given block.
Design parameters

Associative correspondence

It allows a main memory block to be loaded in any cache memory line.

The logic of control of the cache memory interprets a memory direction as a label and a word field.

\[
\begin{array}{c|c}
\text{s bits} & \text{w bits} \\
\hline
\text{Etiqueta} & \text{Palabra}
\end{array}
\]

where

Word = Identifies each word inside a MM block
Label = Identifies univocally a block as MM
Associative correspondence

In order to determine if a block is in the Cache memory, we have to examine simultaneously all labels of the cache memory lines.
Design parameters

Associative correspondence

Example:

Memoria Principal

<table>
<thead>
<tr>
<th>Dirección</th>
<th>Dato</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>12</td>
</tr>
<tr>
<td>0001</td>
<td>34</td>
</tr>
<tr>
<td>0002</td>
<td>56</td>
</tr>
<tr>
<td>000F</td>
<td>78</td>
</tr>
<tr>
<td>F800</td>
<td>AB</td>
</tr>
<tr>
<td>F801</td>
<td>0F</td>
</tr>
<tr>
<td>F802</td>
<td>FF</td>
</tr>
<tr>
<td>F80F</td>
<td>54</td>
</tr>
<tr>
<td>FFF0</td>
<td>FF</td>
</tr>
<tr>
<td>FFFD</td>
<td>21</td>
</tr>
<tr>
<td>FFFE</td>
<td>33</td>
</tr>
<tr>
<td>FFFF</td>
<td>55</td>
</tr>
</tbody>
</table>

Bloque 0
Bloque 3968
Bloque 4095

Memoria Cache

<table>
<thead>
<tr>
<th>Etiqueta</th>
<th>Datos</th>
<th>Nº Línea</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>123456....78</td>
<td>0</td>
</tr>
<tr>
<td>....</td>
<td>............</td>
<td>1</td>
</tr>
<tr>
<td>....</td>
<td>............</td>
<td>.........</td>
</tr>
<tr>
<td>F80</td>
<td>AB0FFF....54</td>
<td>128</td>
</tr>
<tr>
<td>....</td>
<td>............</td>
<td>.........</td>
</tr>
<tr>
<td>FFF</td>
<td>FF.........213355</td>
<td>255</td>
</tr>
</tbody>
</table>

Etiqueta | Palabra
---------|--------
12 bits | 4 bits

12 bits | 16 bytes
Design parameters

Associative correspondence

- Advantage: very fast access.
- Disadvantage: a quite complex circuitry is needed.
Design parameters

Associative correspondence by sets

This technique is a commitment that tries to combine the advantages of the techniques previously described. CM divided in T sets of L lines.
The relations we have are

\[ C = T \times L \]

C.M. set Number = MM B Number mod CM C Number

Bj block can be associated to any of the i set lines.
Design parameters

Associative correspondence by sets

The logic of control of cache memory interprets a memory address with three fields

- **s-d bits**
- **d bits**
- **w bits**

| Etiqueta | Conjunto | Palabra |

where

- w bits of less weight = Word inside a block
- s bits = Identifies a main memory block
- d bits = Especifices one of the cache memory sets
- s-d bits = Label associated to the lines of d set bits
Design parameters

Associative correspondence by sets

To know whether an address is in cache memory or not, first thing is to apply direct correspondence and then associative correspondence.
**Design parameters**

**Associative correspondence by sets**

**Example:**

<table>
<thead>
<tr>
<th>Dirección</th>
<th>Dato</th>
<th>Bloque 0</th>
<th>Bloque 64</th>
<th>Bloque 4095</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000F</td>
<td>78</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0400</td>
<td>AB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0401</td>
<td>0F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0402</td>
<td>FF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>040F</td>
<td>54</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF0</td>
<td>FF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFFD</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFFE</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFFF</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Memoria Principal**

<table>
<thead>
<tr>
<th>Etiqueta</th>
<th>Datos</th>
<th>Nº Conjunto</th>
<th>Palabra</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>123456·78</td>
<td>0</td>
<td>··</td>
</tr>
<tr>
<td>··</td>
<td>··</td>
<td>··</td>
<td>··</td>
</tr>
<tr>
<td>··</td>
<td>··</td>
<td>64</td>
<td>··</td>
</tr>
</tbody>
</table>

**Memoria Cache**

<table>
<thead>
<tr>
<th>Etiqueta</th>
<th>Datos</th>
<th>Nº Conjunto</th>
<th>Palabra</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>AB0FF·54</td>
<td>0</td>
<td>··</td>
</tr>
<tr>
<td>··</td>
<td>··</td>
<td>··</td>
<td>··</td>
</tr>
<tr>
<td>3F</td>
<td>FF·213355</td>
<td>64</td>
<td>··</td>
</tr>
</tbody>
</table>

**Unit 4. Memory**

64
Design parameters

Replacement algorithms

When a new block is transferred to cache memory it has to replace one of the already existing ones if the line is occupied.

In the direct correspondence case this algorithms have no sense.

Between the different existing algorithms the following are remarkable:

- LRU
- FIFO
- LFU
Design parameters

Cache memory examples

Intel processors have been incorporating cache memory to arise their performance.

<table>
<thead>
<tr>
<th>Procesador</th>
<th>Nivel 1</th>
<th>Nivel 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inferior 80386</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>80386</td>
<td>NO</td>
<td>16K, 32K, 64K</td>
</tr>
<tr>
<td>80846</td>
<td>8K datos/instrucciones</td>
<td>64K, 128K, 256K</td>
</tr>
<tr>
<td>Pentium</td>
<td>8K datos y 8K instrucciones</td>
<td>256K, 512K, 1M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Procesador</th>
<th>Cache interna</th>
<th>Descripción</th>
</tr>
</thead>
<tbody>
<tr>
<td>80846</td>
<td></td>
<td>Tamaño de línea de 16 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Organización asociativa por conjuntos de 4 vías</td>
</tr>
<tr>
<td></td>
<td>Cache externa</td>
<td>Tamaño de línea de 32, 64 ó 128 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Organización asociativa por conjuntos de 2 vías</td>
</tr>
<tr>
<td>Pentium</td>
<td>Cache interna</td>
<td>Tamaño de línea de 32 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Organización asociativa por conjuntos de 2 vías</td>
</tr>
<tr>
<td></td>
<td>Cache externa</td>
<td>Tamaño de línea de 32, 64 ó 128 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Organización asociativa por conjuntos de 2 vías</td>
</tr>
</tbody>
</table>
An associative memory is characterized by the fact that the memory position attempted to access is made by specifying its content or part of it and not by its address.

- Associative memory is also known as Content Addressable Memory (CAM).
Structure of a CAM

- An associative memory consists of a set of registries and a matrix of memory cells, with its logic associated, organized in n words with m bits/word.

- The set of registries is formed by a argument registry (A) of m bits, a mask registry (K) of m bits and a mark registry (M) of n bits.
Structure of a CAM

- Each word of the memory is simultaneously compared with the content of the argument registry, and the bit of the mark registry is put to 1 associated to those words whose content agrees with the one of the argument registry.

- At the end of this process, those bits of the mark registry that are to 1 indicate the coincidence of the corresponding words of the associative memory and the argument registry.
Structure of a CAM

The simultaneous comparison is made bit to bit. A \(_j\) bit (\(j=1,2,...,m\)) of the argument registry is compared with all the bits of the \(j\) column if \(K_j=1\). If there’s a coincidence between all the bits \(M_i=1\). \(M_i=0\) on the contrary.

<table>
<thead>
<tr>
<th></th>
<th>Alicante</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alicante</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nombre</th>
<th>Ciudad</th>
<th>Teléfono</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juan</td>
<td>Alicante</td>
<td>965254512</td>
</tr>
<tr>
<td>Pepe</td>
<td>Elda</td>
<td>965383456</td>
</tr>
<tr>
<td>Ana</td>
<td>Alicante</td>
<td>965907799</td>
</tr>
<tr>
<td>Laura</td>
<td>Elche</td>
<td>965442233</td>
</tr>
<tr>
<td>Pepe</td>
<td>Alicante</td>
<td>965223344</td>
</tr>
<tr>
<td>Paco</td>
<td>Elche</td>
<td>966664455</td>
</tr>
<tr>
<td>Paqui</td>
<td>Petrer</td>
<td>965375566</td>
</tr>
<tr>
<td>Pepi</td>
<td>Alicante</td>
<td>965286677</td>
</tr>
</tbody>
</table>
Structure of a CAM

- As a rule, in most of the applications the associative memory stores a table that does not have, for a given mask, two equal rows.

- The associative memory is used mainly with cache memory the way the identification of the label of each line is made simultaneously.

- TAG RAM is a clear example of associative memory used as a part of Cache memory in the Intel Pentium systems.
Necessity that different devices have access to a same unit of memory.

The referee is the element in charge of allowing the access to the memory unit, at a given moment, to each one of the elements that ask for this resource.
The referee is designed the way it assigns a service time, in average, analog to all the units that ask for the resource.

There are different strategies:

- Allocation of the smaller priority to the served element.
- Rotation of priorities. In any given state, the next state is calculated rotating the present order of priorities until the just given service element has the lowest priority.
Dual port memory

It is shared memory able to work with two elements at the same time. Based on duplicating the buses, decoders, etc.
Dual port memory

Dual port memory has practically all its components duplicated (left port (LEFT) y right port (RIGHT)).

<table>
<thead>
<tr>
<th>Puerto Izquierdo</th>
<th>Puerto Derecho</th>
<th>Descripción</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O_L</td>
<td>I/O_R</td>
<td>Bus de datos</td>
</tr>
<tr>
<td>A_L</td>
<td>A_R</td>
<td>Bus de direcciones</td>
</tr>
<tr>
<td>CE_L</td>
<td>CE_R</td>
<td>Selección de chip</td>
</tr>
<tr>
<td>R/W_L</td>
<td>R/W_R</td>
<td>Lectura/Escritura</td>
</tr>
<tr>
<td>OE_L</td>
<td>OE_R</td>
<td>Habilita lectura</td>
</tr>
<tr>
<td>SEM_L</td>
<td>SEM_R</td>
<td>Habilita semáforo</td>
</tr>
</tbody>
</table>

VRAM is a clear example of dual port memory. It can be accessed simultaneously by the monitor controller and the graphics card controller.