# Topic 3

# Memory Management and Access

*Department of Electronics*

*Academic Year 14/15*

*(ver 25-10-2014)*

# Index

# 3.1. Memory maps

## Memory map

- It refers to the organization of the different memory units in the uP address space

## uP address space

- Number of addresses: $2^m$ positions (addresses), being *m* the size of the address bus
- Width word *n* bits, being *n* the number of lines (bits) of the data bus
- uP address space: $2^m$ X *n*

# 3.1. Memory maps

- By means of the memory map, it is described which addresses are occupied by the different devices assigned to specific functions
- Conceptual elements included in the map:
  - **Program memory** (typically non-volatile memories)
  - **Data** memory (can be volatile and non-volatile memories)
  - **Input/output** space. All the interfaces according to the peripherals that will be needed have to be included in the map

To sum up, the memory map typically includes ROM, RAM and I/O interfaces

The memory map specification can be carried out as follows:

**Functional:** it describes the allocation (addresses) of the different elements (hardware or software) of the digital system, according to their function: the location of the program sections, general data and data tables, interface registers, etc.

**Physical:** it describes the address correspondence between the map and the physical device which implement it. All the connections between the devices will be carried out (taking into account the structure of the address bus and data bus, the method of selecting devices, etc.)

# 3.1. Memory maps

- In the design of the map, it is specified which **addresses** are occupied by the different devices, indicating:
  - The starting address, usually called **base address**
  - The last address. It is calculated as: **base address+number of positions occupied by the device in memory**
- Considerations about the **base address**, Could it be any value?
  1. It has to belong to the uP address space (logically)
  2. It has to be an appropriate value, so the last address is inside the address space
  3. It has to be **multiple** of the size of the memory block that has to be addressed: if it is needed to **map** a M bytes block, the address base has to be multiple of M

**Example:** for a 8Kbytes block, the address base could be 0x0000, 0x2000, 0x4000 …

- The 0 address is multiple of any block
- There is a “trick”. The address base finishes always in a determined number of 0 (**how many?**)

# 3.1. Memory maps

## Example of a memory map

- 8 bits uP (width of the data bus), 16 bits address bus
- Which is the address space of the uP?
- Which are the selected base addresses?
- Which is the size of each block?

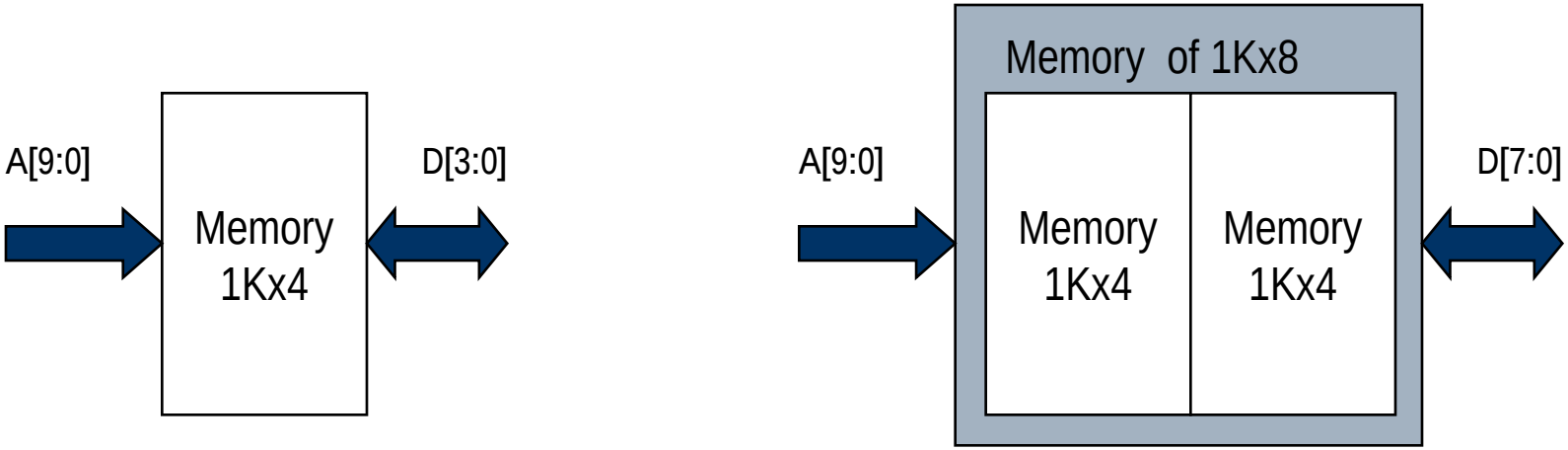
| MEMORY MAP | | |
|---|---|---|
| Functional Map | Physical Map | Addr. |
| Program and constant data tables | Chip ROM, 8Kbytes | 0x0000<br>0x1FFF |
| *Empty area* | ----- | 0x2000<br>0x3FFF |
| I/O devices | Integrated circuits, LCD displays, etc. | 0x4000<br>0x41FF |
| *Empty area* | ------- | 0x4200<br>0x8FFF |
| Hex keyboard | Keyboard model | 0x9000<br>0x9003 |
| *Empty area* | -------- | 0x9004<br>0xBFFF |
| Variables and temporary data | Chip RAM, 8Kbytes | 0xC000<br>0xCFFF |
| Serial transfer data | | 0xD000<br>0xDFFF |

# 3.2. Memory Expansion

- Goal: To obtain a block of memory of larger capacity by using basic memory chips (usually of the same capacity), with two purposes:
  - To provide a memory block of larger capacity.
  - To adapt to the specifications of the selected uP (data bus and/or address space)
- Reminder:
  - Capacity=number of words X size of each word
    - Capacity increases if one of the parameters (or both) increases
- Types of memory expansions:
  - To increase the size of the word (it means to increase the number of bits per word)
  - To increase the number of words
  - Both of them

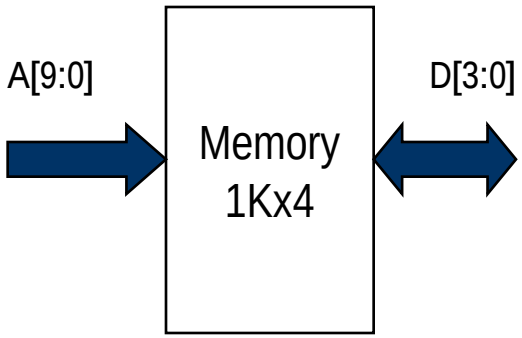
# 3.2. Memory Expansion

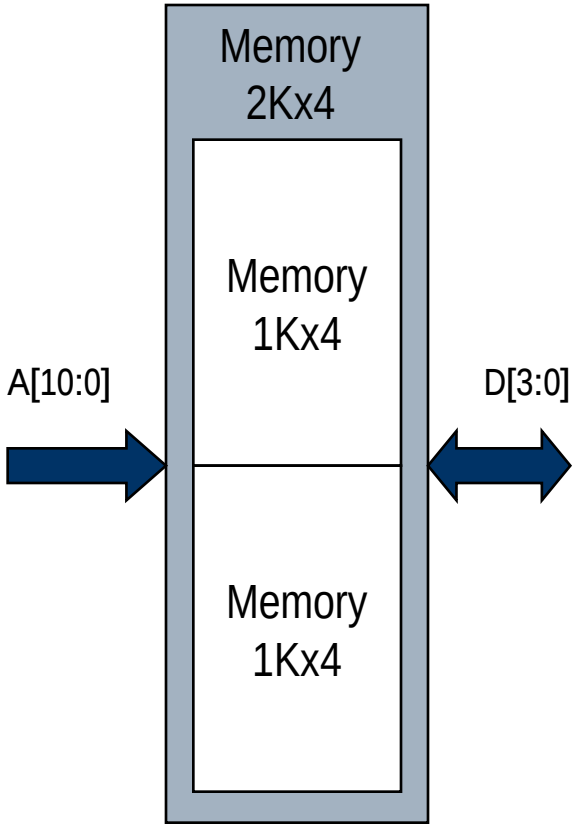
- Expansion of the number of bits per word (word length)
  - Conceptually it means to increase the width of the data bus in the new memory block. How many bits?
  - Each time the memory block is accessed, all the memory chips are simultaneously activated (the data is “shared” among all the chips)
  - The size of the address bus is held



# 3.2. Memory Expansion

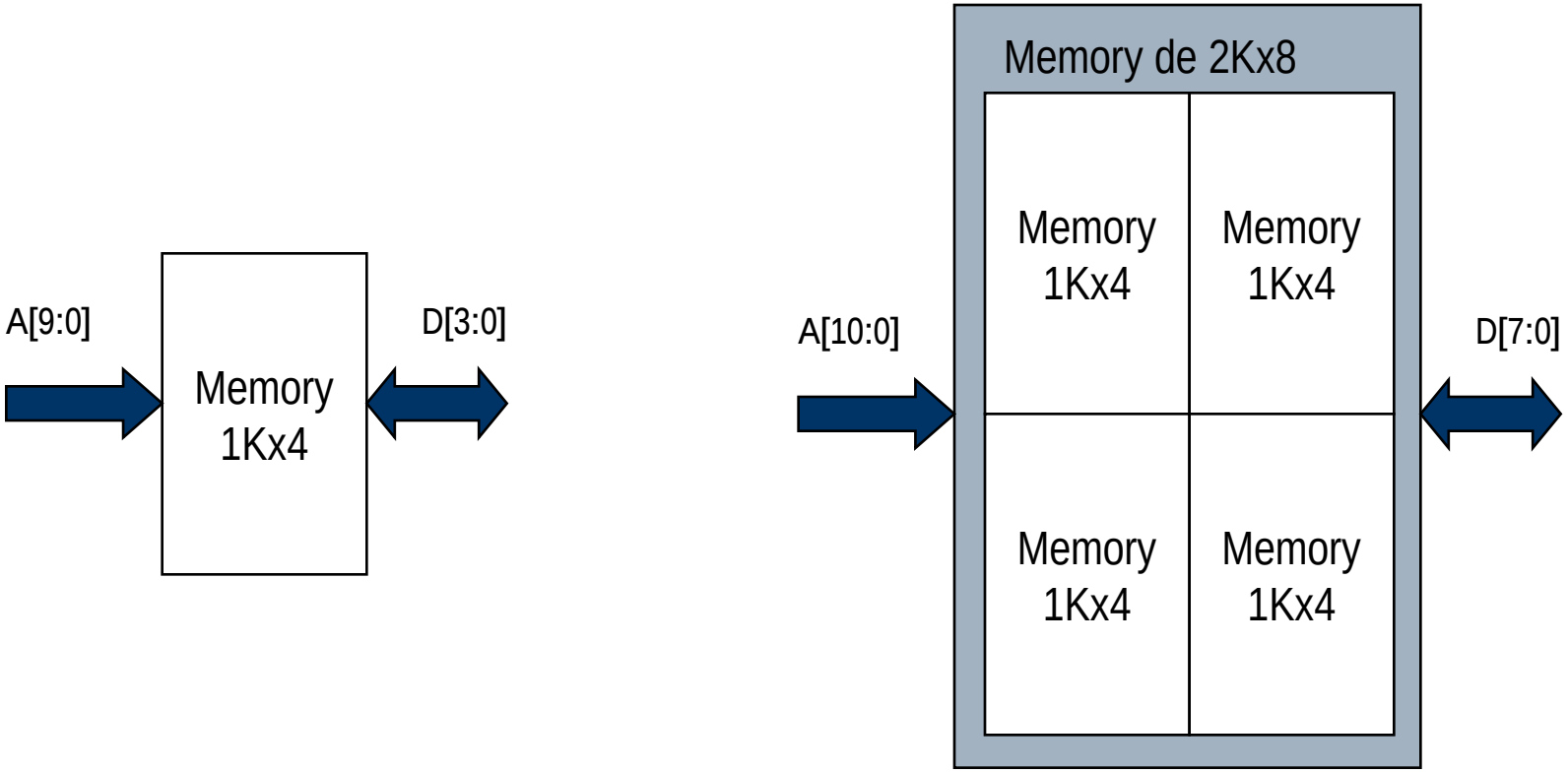
- Expansion of the **number of words**
  - Conceptually it means to increase the size of the address bus
    How many bits? (**the number of bits is increased always in power of two**)
  - The width of the data bus is held



# 3.2. Memory Expansion

- Combination of both expansions, to increase the number of bits per word, and the number of words:
  - The width of the data bus is increased
  - The address bus size is increased

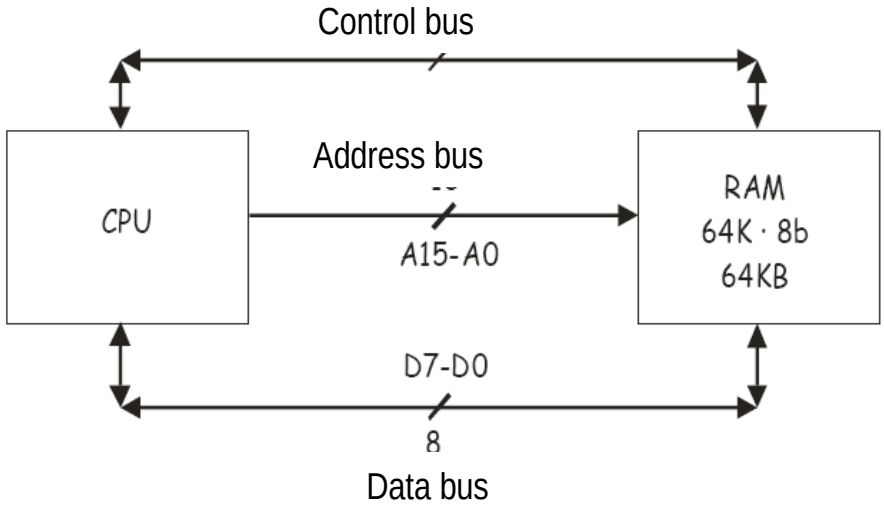


# 3.2. Memory Expansion

- How is the memory connected to the uP?
  - The resulting expanded memory has to be connected as if it was only one chip, connecting the suitable lines of the data, address and control busses.

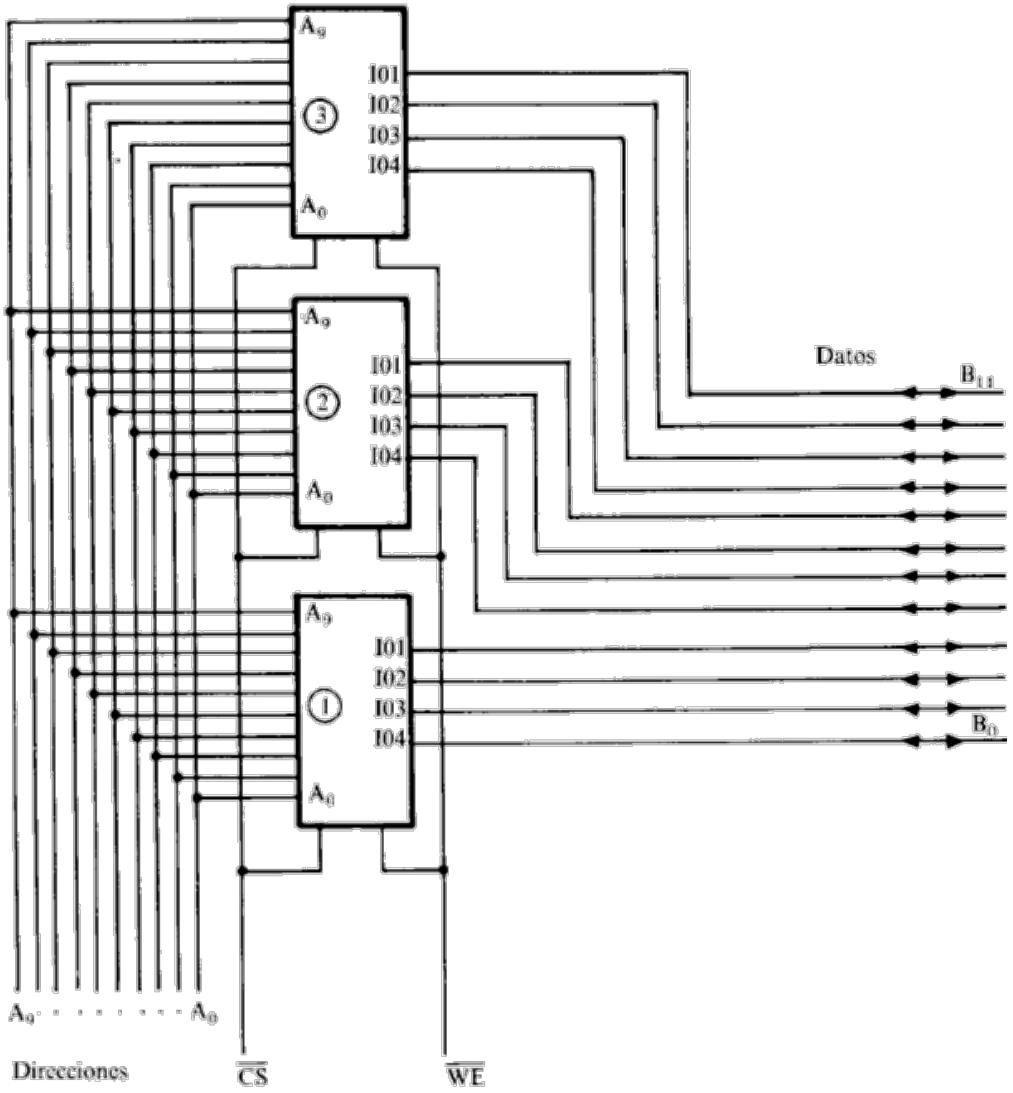
**Example:** 64Kbytes memory RAM

- Address: 16 lines, e.g. A0-A15 (there are other solutions)
- Data: 8 lines
- Control: operation line(R/#W), address validation (#AS)



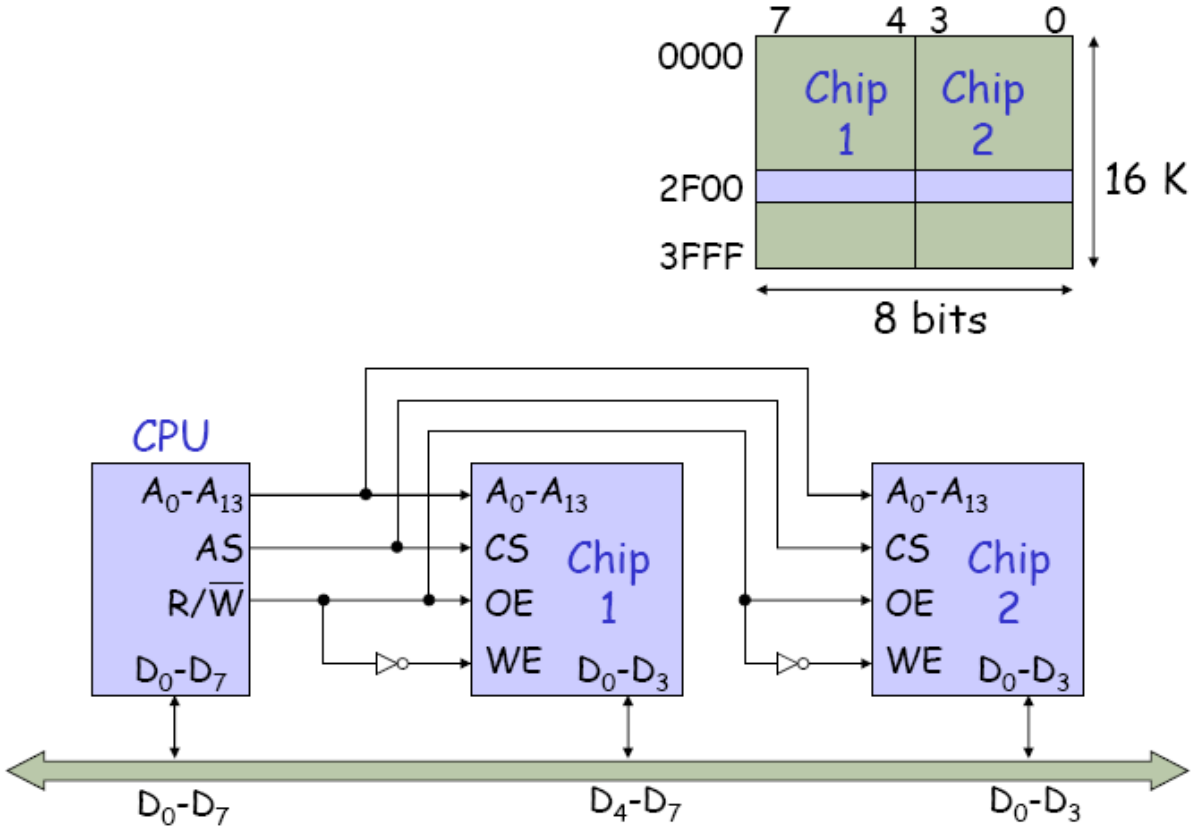
# 3.2. Memory Expansion: word length

- 1Kx12 memory by means of 1Kx4 memories



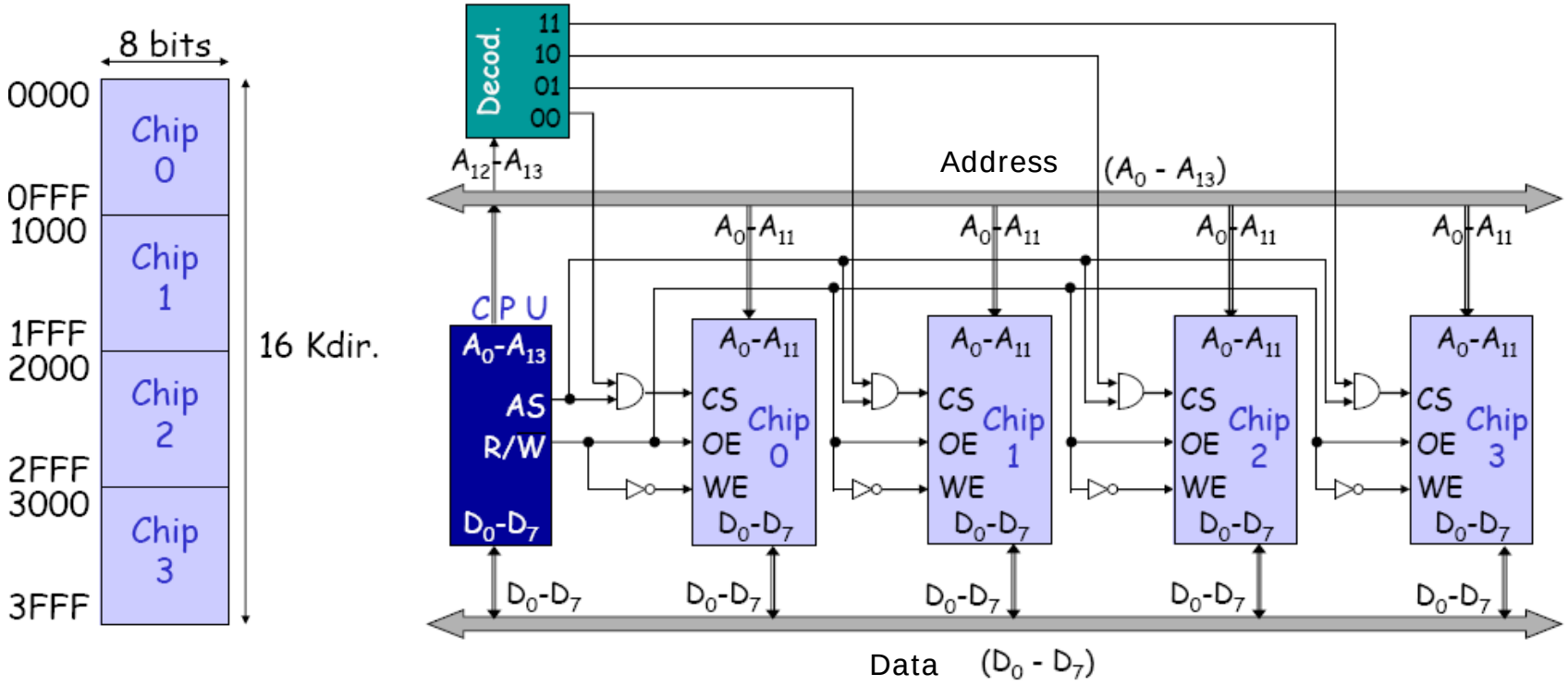
# 3.2. Memory Expansion: word length

- Example of a whole system: 16Kx8 from 16Kx4



# 3.2. Memory Expansion: number of words

- Example of a whole system: 16Kx8 from 4Kx8




**Question**: Deduct the address range that implements each chip, if the memory address busses are connected to $A_2$-$A_{13}$, and for decoding are used $A_0$ and $A_1$

# 3.2. Memory Expansion: number of words

## Example 1

Design a 2Kx4 memory block by means of 1Kx4 memory chips

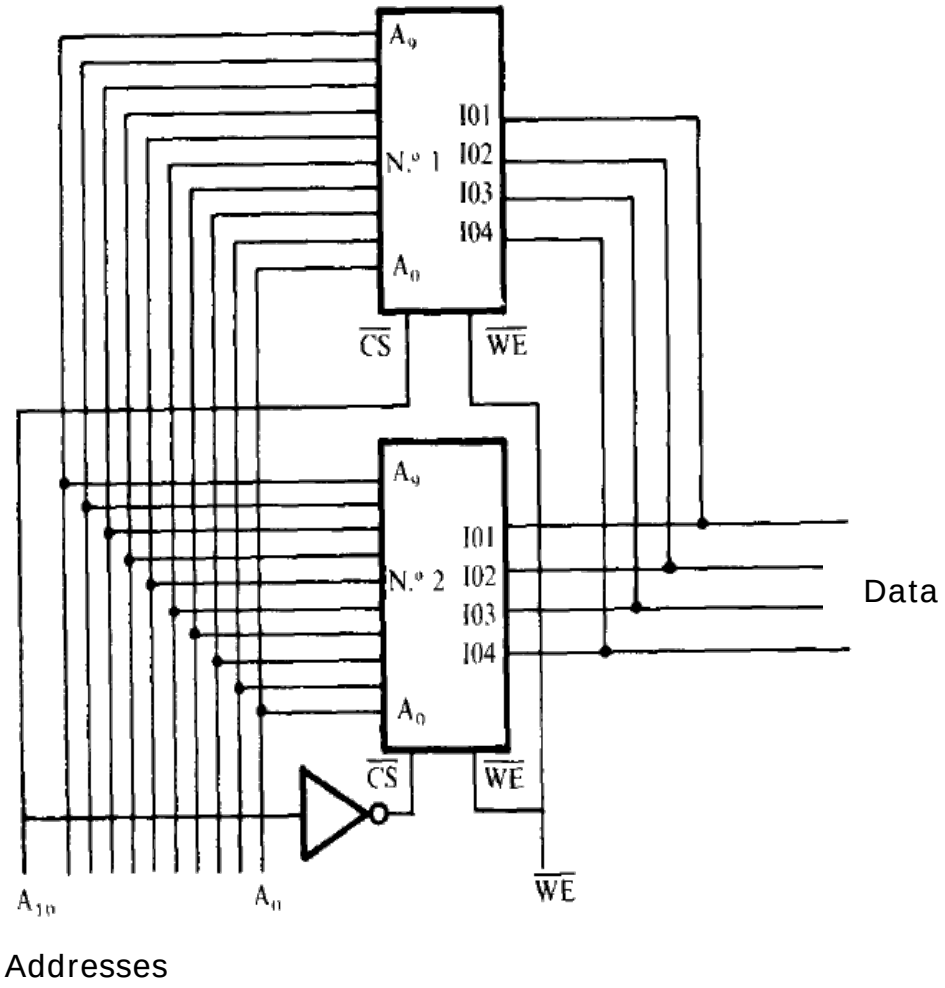
- Which is the address range of each chip?
- If A0 is used for selecting each chip (the memory address bus is connected from the uP A1 line), which is the address range of each chip?

# 3.2. Memory Expansion: number of words

## Solution 1

- 2 memories of capacity 1Kx4 are required

- Address range
  - Nº1 0x000-0x3FF
  - Nº2 0x400-0x7FF
- If A0 is used, one chip will use even addresses and the other the odd ones in the range 0x000 -0x7FF h



# 3.2. Memory Expansion: number of words

## Example 2

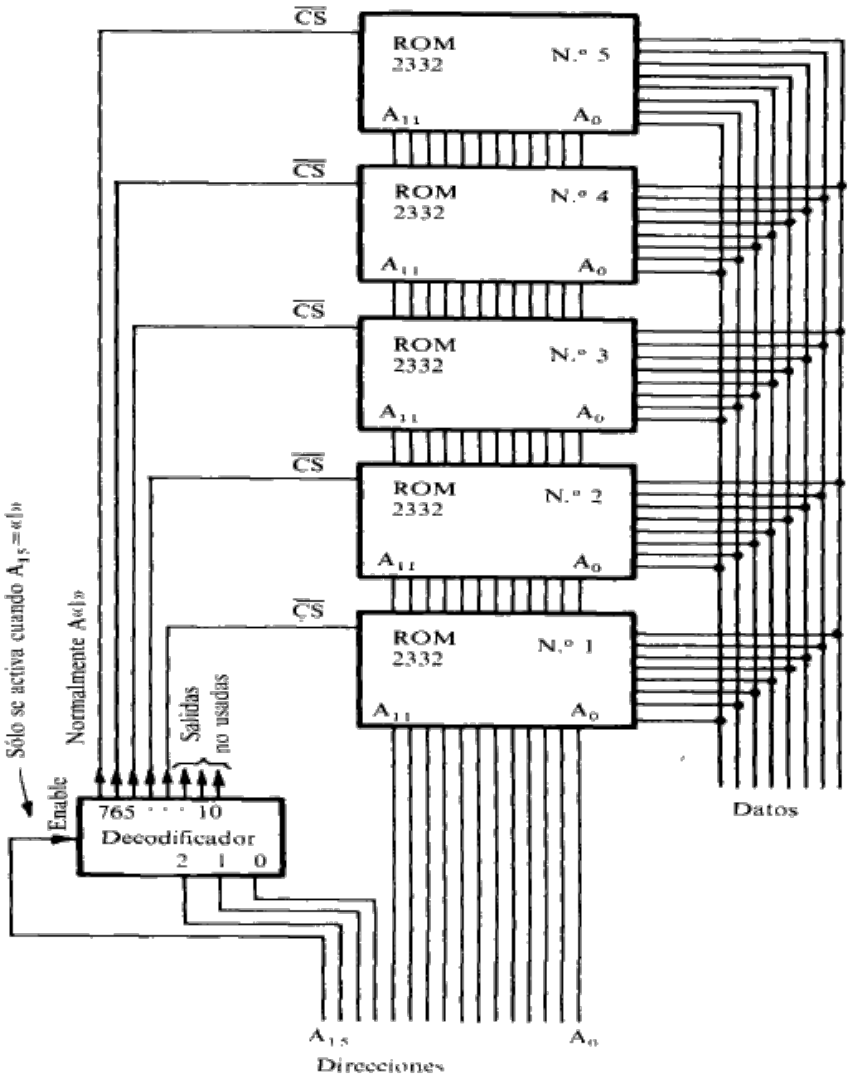
Design a 20Kx8 memory block by means of 4Kx8 memory chips.
For decoding, a 3/8 decoder has to be used

- Which is the address range of each chip?
- For which capacity has the expansion been designed?

# 3.2. Memory Expansion: number of words

## Solution 2

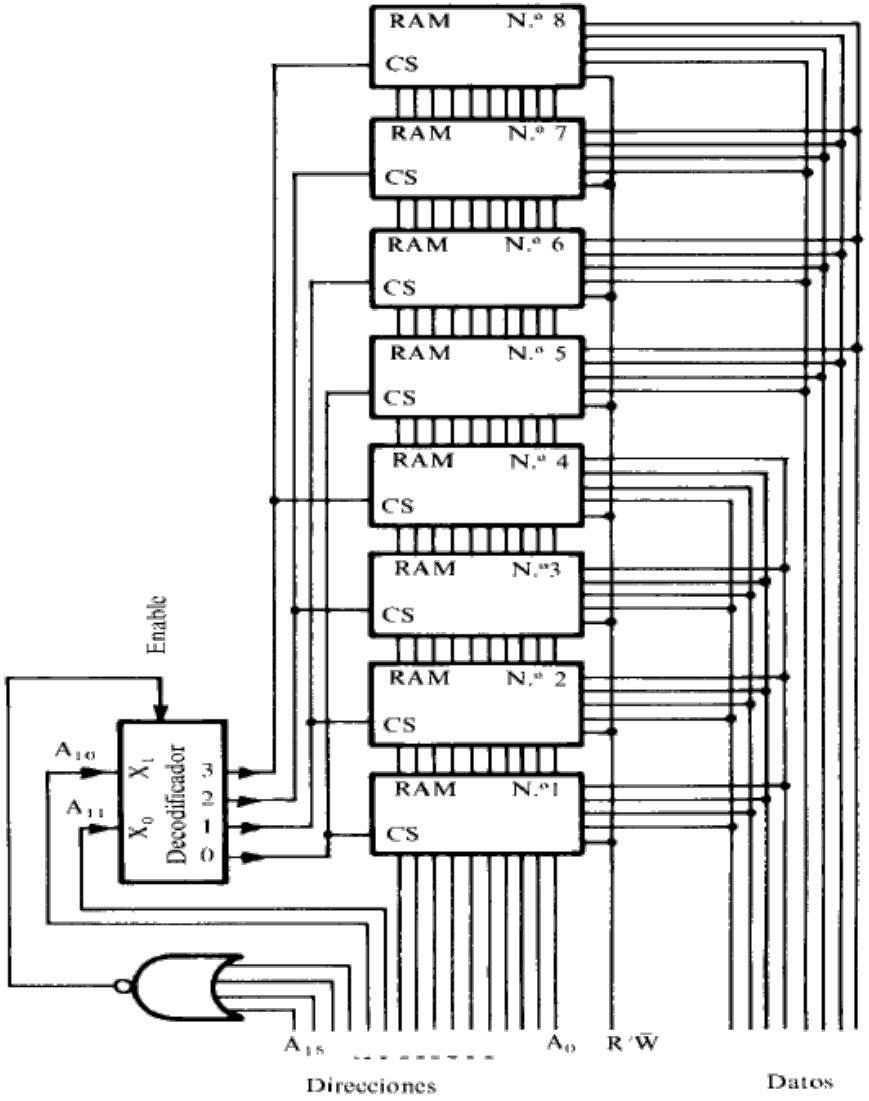
- The expansion is prepared for 32Kx8 capacity
- Address range
  - 0xB000-0xBFFF
  - 0xC000-0xCFFF
  - 0xD000-0xDFFF
  - 0xE000-0xEFFF
  - 0xF000-0xFFFF



# 3.2. M. Expansion: nº of words + word length

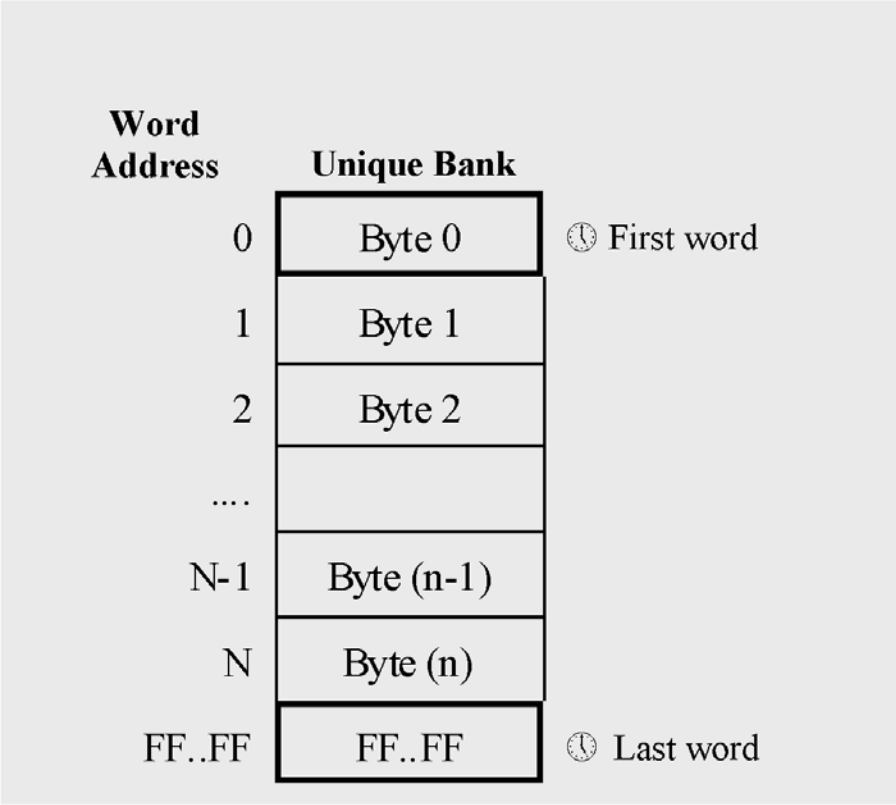
## Example 3

Design a 4Kx8 memory block by means of 1Kx4 memory chips



# 3.3. Memory Management & Data Alignement

- The memory structure depends on the number of bits of the uP external data bus
  - 8-bit data bus, only a bank (group) of byte size



# 3.3. Memory Management & Data Alignement

- With an external data bus of 16 bits (although internally it can be a 32-bit bus), memory can write or read 16 bits by 16 bits. Then, memory is structured in two banks: the odd one and the even one
  - Access to data of one and two bytes is allowed
  - In this case, control lines are added to indicate if only a byte is accessed (odd or even) or two bytes are simultaneously accessed. (i.e. 68000 by Motorola)

| 2 bytes Word address | (Even) Bank 0 (2N) | (Odd) Bank 1 (2N+1) | | |
|---|---|---|---|---|
| 0 | Byte 0 | Byte 1 | » | First word |
| 2 | Byte 2 | Byte 3 | | |
| .... | | | | |
| N-2 | Byte (n-2) | Byte (n-1) | | |
| N | Byte (n) | Byte (n+1) | | |
| | | | | |
| FF..FE | FF..FE | FF..FF | » | Last word |

# 3.3. Memory Management & Data Alignement

- External data bus of 32 bits, memory is arranged in 4 banks
  - Accesses to one, two or four bytes are allowed. Indicative control lines are included (i.e. the ColdFire by Freescale)

| 4 bytes Word address | Bank 0 (4N) | Bank 1 (4N+1) | Bank 2 (4N+2) | Bank 3 (4N+3) |
|---|---|---|---|---|
| 0 | Byte 0 | Byte 1 | Byte 2 | Byte3 |
| 4 | (Half) Word 4 | | (Half) Word 6 | |
| 8 | (Double) Word (Longword) 8 | | | |
| …. | --- | --- | --- | --- |
| N-4 | Byte N-4 | Byte N-3 | Byte N-2 | Byte N-1 |
| N | (Half) Word N | | (Half) Word N+2 | |
| N+4 | (Double) Word (Longword) N+4 | | | |
| …. | | | | |
| FF…F8 | Byte FF…F8 | Byte FF…F9 | Byte FF…FA | Byte FF…FB |
| FF…FC | Byte FF…FC | Byte FF…FD | Byte FF…FE | Byte FF…FF |

# 3.3. Memory Management & Data Alignement

## Summary

- If the data bus size is 8 bits, maximum size of data transfer is also 8 bits. Memory structure will have only one bank
- If the data bus size is 16 bits, and if the min. size of data transfer is 8 bits; then, memory structure will have two banks (even -2N- and odd -2N+1- banks). At least one control line is needed for distinguishing the access
- If the data bus size is 32 bits, and if the min. size of data transfer is 8 bits; then, memory structure will have four banks (4N, 4N+1, 4N+2, 4n+3 banks). At least two control lines are needed for distinguishing the access

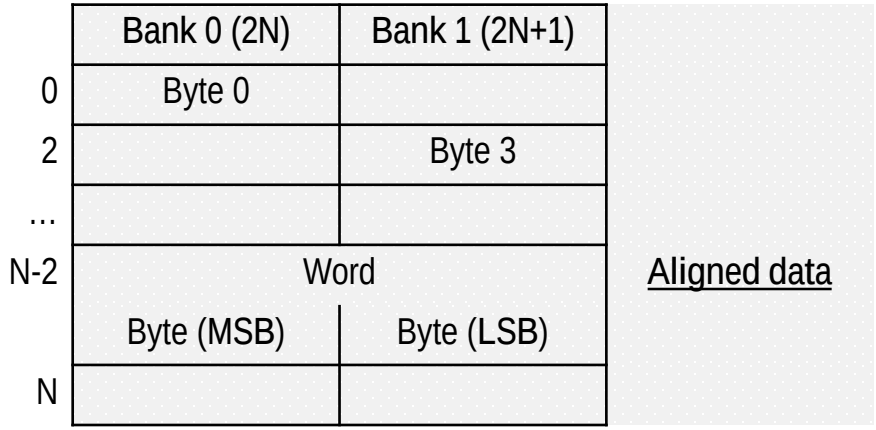
Pay attention to the proposed exercises for doing in class

# 3.3. Memory Management & Data Alignement

- How are the data bytes aligned in memory? Two proposals
  - Big endian or Little endian

**Big endian.** The least significant byte is stored in the memory cell corresponding to the highest address, of ones which contain the data

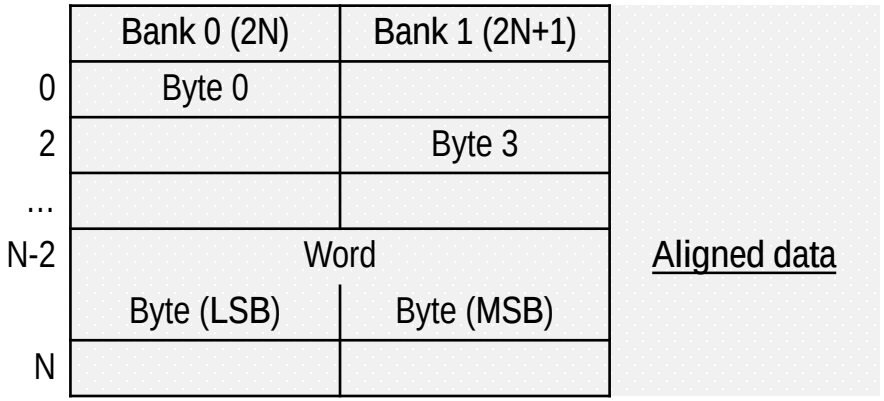
In any case, the memory cells, which contain the data, must be consecutive



# 3.3. Memory Management & Data Alignement

**Little endian.** The least significant byte is stored in the memory cell corresponding to the lowest address, of ones which contain the data

In any case, the memory cells, which contain the data, must be consecutive

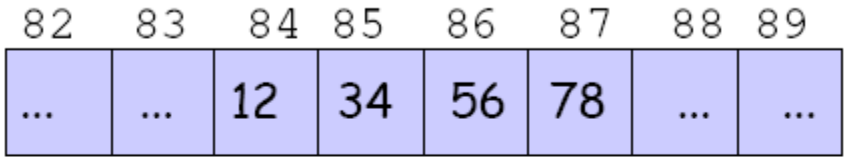



**Attention:** In practice, *big or little endian do not differentiate*, but it is needed to know the alignment model for the interchange of data between systems

# 3.3. Memory Management & Data Alignement

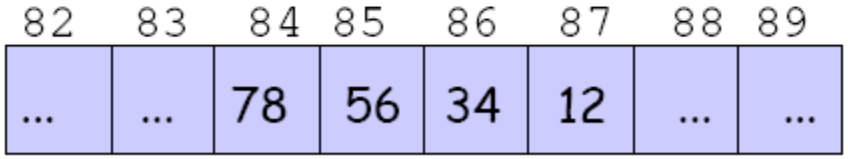
- Example of data storage 0x12345678 from address 0x84 in both formats:

Data: 12345678H at address 84




Big-Endian

The Least Significant Byte at the highest address




Little-Endian

The Most Significant Byte at the highest address

# 3.3. Memory Management & Data Alignement

- Aligned and misaligned data
  - This concept is only applicable if the data is formed by more than one byte (2 bytes, 4 bytes, etc.) and the uP allows the access to different sizes: 8 bits, 16 bits, 32 bits, etc.
  - Aligned data: data is aligned if its access only requires a binary combination of the address bus
  - Misaligned data: data is misaligned if its access requires two or more binary combinations of address bus. This leads to different accesses (as many binary combinations of address bus)

# 3.3. Memory Management & Data Alignement

- Aligned and misaligned data (example by using little endian)

| Word address | Bank 0 (4N) | Bank 1 (4N+1) | Bank 2 (4N+2) | Bank 3 (4N+3) | |
|---|---|---|---|---|---|
| 0 | Byte 0 | Byte 1 | Byte 2 | Byte3 | |
| 4 | Halfword 4 LSB | Halfword 4 MSB | Halfword 6 LSB | Halfword 6 MSB | Aligned data |
| 8 | Byte 8 | Halfword 9 LSB | Halfword 9 MSB | Halfword A LSB | Misaligned data |
| C | Halfword A MSB | | | | |
| 10 | Longword 10 LSB | Longword 10 | Longword 10 | Longword 10 MSB | Aligned data |
| 14 | | Longword 15 LSB | Longword 15 | Longword 15 | Misaligned data |
| 18 | Longword 15 MSB | | | | |

# 3.4. Design of a memory maps

Stages of the design of a memory map:

1. Analyze the specifications of the system (capacity of memory, types of chips, ...)
2. Plan the functional map, decide in which addresses to allocate different systems
3. Design of the physical map. The selection logic (decoding system) has to be designed for accessing to the desired memory position, and no to others:
   - Through the address that appears in the address bus, the chip select line belonging to the chip that implements the address has to be activated. **Prepare a decoding table!**
   - **Pay attention**: the address bus is divided into two groups of lines. One group (the lowest address lines) is directly connected to the memory address bus (the same number of lines as the memory has). The remainder of the lines (the highest one) can be used for the selection logic.
   - The selection logic can be implemented by means of logic gates, decoders, programmable devices (PLD, PAL, …), etc.
4. Connection to the uP (address, data and control busses)

Important: In addition, very often it is required a memory expansion as well for obtaining the desired capacity.

# 3.4. Design of a memory maps

## Example of design 1:

A given uP has an 8-bit data bus and a 16-bit address bus, with and address validation line, AS#. The memory map to be designed has the following features:

- Permanent memory for code and data, 8Kbytes, starting from address 0x0000
- Volatile memory for temporal data and variables, 4Kbytes, starting from address 0xC000
- Volatile memory for serial transfers, 4Kbytes, after the later memory block
- I/O devices, 512 bytes starting from address 0x4000
- Hexadecimal keyboard, occupying 4 memory positions, mapped at address 0x9000

## Memory requirements:

According to the specifications, it is needed 8Kbytes of ROM mapped at address 0x0000, 512 bytes for I/O mapped at address 0x4000, 4 bytes mapped at address 0x9000 for the keyboard and 8 Kbytes of RAM, mapped at address 0xC000

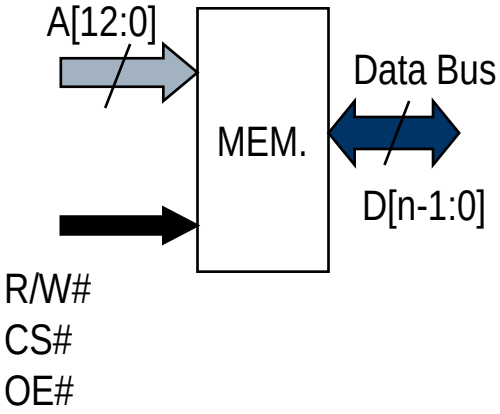
# 3.4. Design of a memory maps

- Maps:
  - Functional
  - Physical

| MEMORY MAP | | |
|---|---|---|
| Functional Map | Physical Map | Addr. |
| Program and constant data tables | Chip ROM, 8Kbytes | 0x0000 0x1FFF |
| *Empty area* | ----- | 0x2000 0x3FFF |
| I/O devices | Integrated circuits, LCD displays, etc. | 0x4000 0x41FF |
| *Empty area* | ------- | 0x4200 0x8FFF |
| Hex keyboard | Keyboard model | 0x9000 0x9003 |
| *Empty area* | -------- | 0x9004 0xBFFF |
| Variables and temporary data | Chip RAM, 8Kbytes | 0xC000 0xCFFF |
| Serial transfer data | | 0xD000 0xDFFF |

# 3.4. Design of a memory maps

- Design of the selection logic: Decoding table
  - Consider the base address
  - Identify the function of the different address bits
    - Which bits are used for accessing to one position of the device?
    - Which bits are used for the selection logic?




Example:

- For the RAM of 8 Kbytes and the ROM of 8 Kbytes, the address lines A[12..0] are needed for selecting the desired memory position in the chip (for accessing to the block), so, they can have any value, thus it is represented by "XX...X"
- The remainder lines A[13..13], have a different value depending on the chip. This is:
  - "**0 0 0" for ROM**→ *0x0000 - 0x1FFF*
  - **"1 1 0" for RAM** → *0xC000 - 0xDFFF*

Pay attention: the address base has to be a multiple of the block size
(It has to finish in so many zeros "…000" as address lines has the chip)

# 3.4. Design of a memory maps

## Decoding table

| Address Lines | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Highest Lines (high byte) | | | | | | | | Low byte | | Selection Lines | | | | |
| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7..A4 | A3..A0 | CS1 | CS2 | CSES1 | CSES2 | Selected device |
| 0 | 0 | 0 | x | x | x | x | x | X | X | 0 | 1 | 1 | 1 | ROM |
| 1 | 1 | 0 | 0 | x | x | x | x | X | X | 1 | 0 | 1 | 1 | RAM Variable |
| 1 | 1 | 0 | 1 | x | x | x | x | X | X | | | | | RAM Serial Data |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | x | X | X | 1 | 1 | 0 | 1 | Input/Output |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0000 | 00xx | 1 | 1 | 1 | 0 | Keyboard |

| Initial/Final address | Selected Device | |
|---|---|---|
| 0x0000 / 0x1FFF | ROM | |
| 0xC000 / 0xCFFF | RAM | Variable |
| 0xD000 / 0xDFFF | | Serial Data |
| 0x4000 / 0x41FF | Input/Output | |
| 0x9000 / 0x9003 | Keyboard | |

# 3.4. Design of a memory maps

## Selection Logic: two alternatives

### Full address decoding:

The access to one memory position can be done **only by one combination** of the bit lines of the address bus:

*{one physical position = one logical address}*

All the bit lines are considered in the decoding, and **it complicates the decoding circuitry**

### Partial address decoding:

The access to one memory position can be done for **several combinations** of the bit lines of the address bus:

*{one physical position = several logical addresses}*

This decoding is easier, but it can create ambiguity:

*With how many addresses can a memory position be accessed?*

$$\text{Number of addresses} = 2^{\text{number of the address lines NO considered in the decoding}}$$

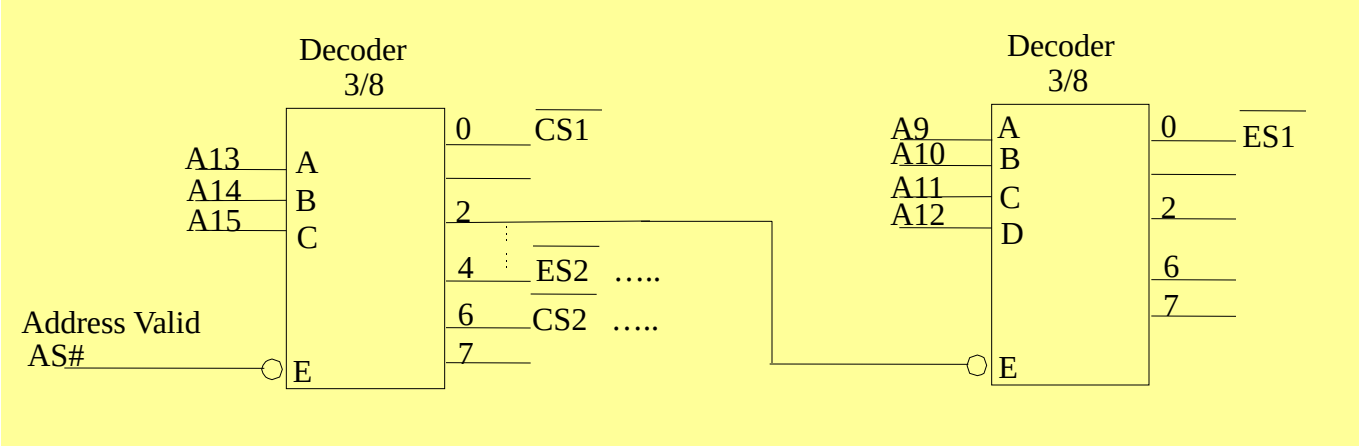
*How is the minimum number of address lines to be used?*

*At least,* $\log_2$ *(number of the devices to be selected)*

# 3.4. Design of a memory maps

## Selection logic

### Full address decoding



# 3.4. Design of a memory maps

## Exercise

- Design the selection logic belonging to the previous example by using partial address decoding

- How many addresses are used for accessing to each device?

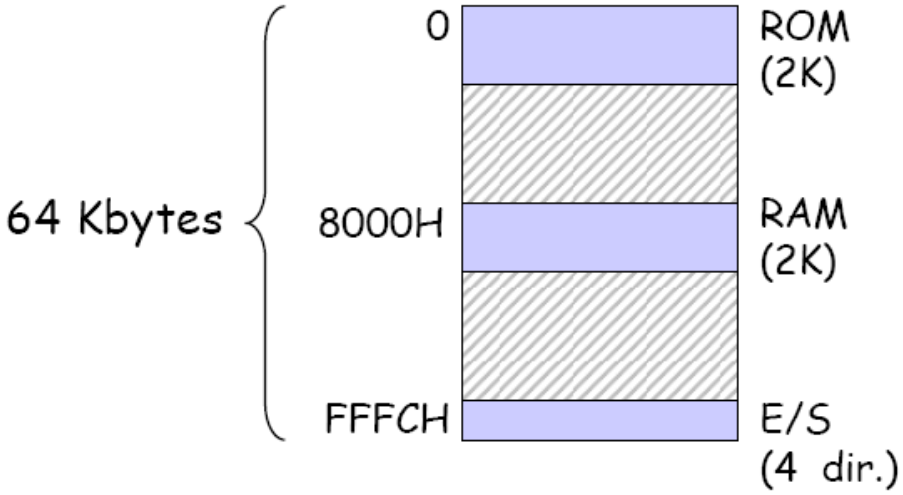
# 3.4. Design of a memory maps

## Example of design 2:

A system has a space address of 64 K x 8, occupied for 2 Kbytes of memory ROM starting from address 0, and other 2 Kbytes of RAM starting from address 8000H. Besides, the 4 last addresses are used for selecting 4 registers belonging to a peripheral.

- Draw the address space and indicate the range address occupied for each device
- Design the selection logic for full and partial address decoding

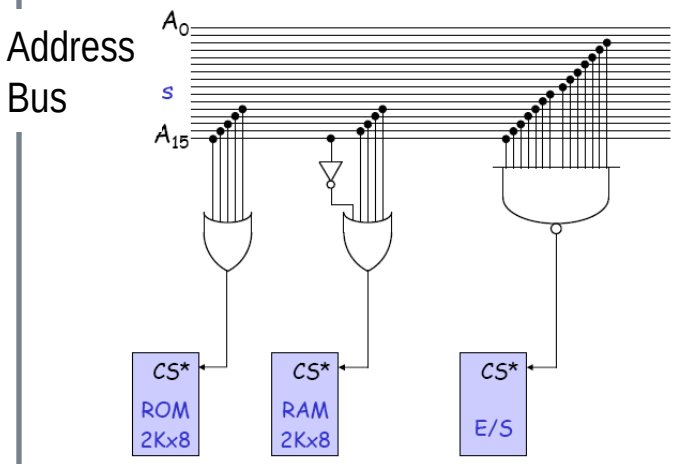
## Example of design 2: Solution



# 3.4. Design of a memory maps

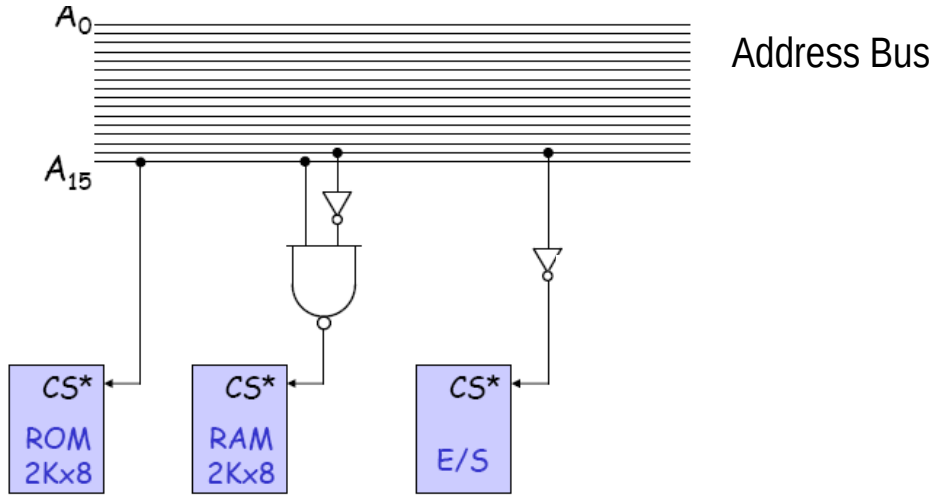
## Example of design 2: Solution

### Solution by using full address decoding




### Solution by using partial address decoding

| | | | | |
|---|---|---|---|---|
| Addresses ROM | 0000 | 0xxx | xxxx | xxxx |
| Addresses RAM | 1000 | 0xxx | xxxx | xxxx |
| Input / Output | 1111 | 1111 | 1111 | 11xx |




Pay attention: each ROM position is accessed with 16 different addresses; each RAM position with 8, and I/O with $2^{13}$

# 3.5. Access management and timing

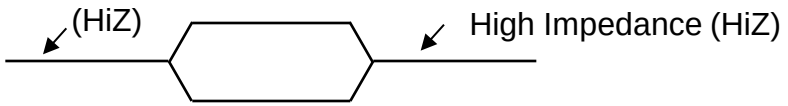
- Access management and timing
  - When analyzing uP connected to a memory device:
    both uP + memory timings have to be studied and compared
- Access time to memory is a limiting factor of the performance of a digital system
  - Access time is always slow, compared with the uP
  - Access time depends on memory type, technology and manufacturing
  - It also depends on interconnection to the uP
- Approach working with slow memories:
  - Solution: extend the uP read/write cycle according to each memory
    - How? By adding wait cycles, until all data are read/written.
    - How can wait cycles be added?

# 3.5. Access M+T: timing diagram

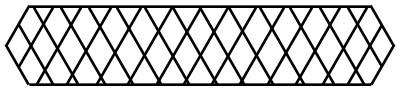
## Notation:

- Bus: Signal formed by a set of lines

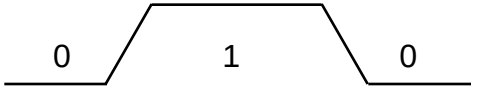

- High Impedance (HiZ)



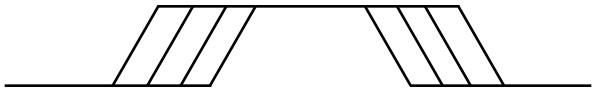

- Unknown value (irrelevant)



- Logic ‘0’ and ‘1’



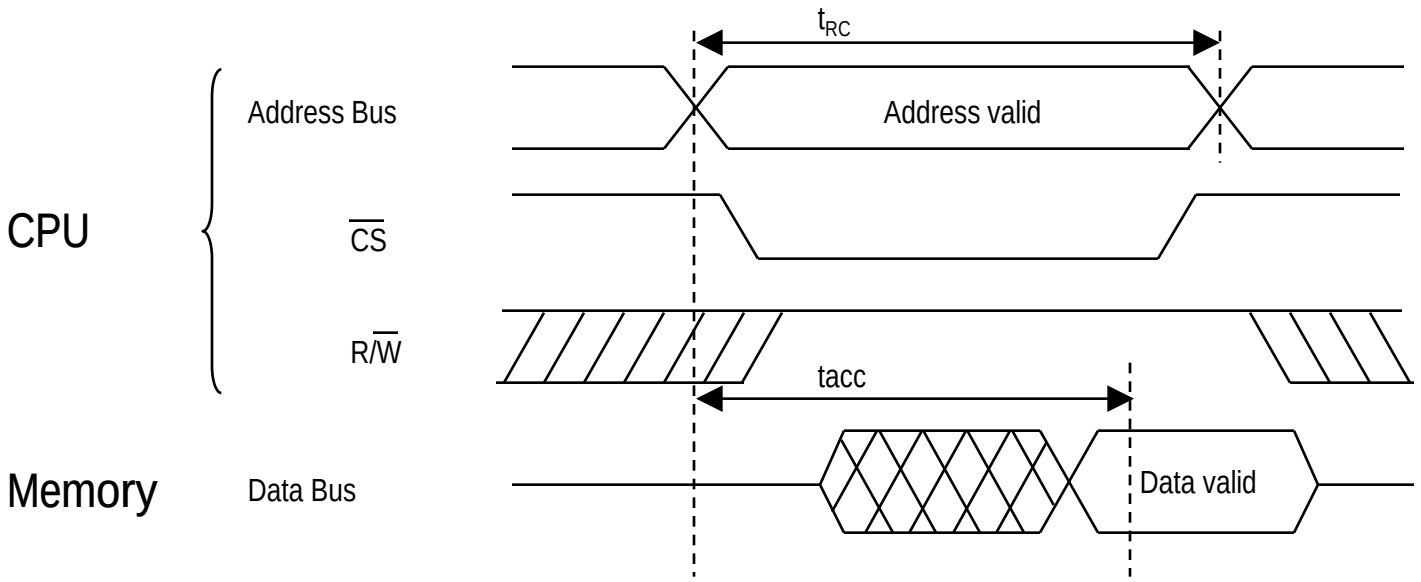

- Undetermined moment of change in a signal



# 3.5. Access M+T: Read timing diagram

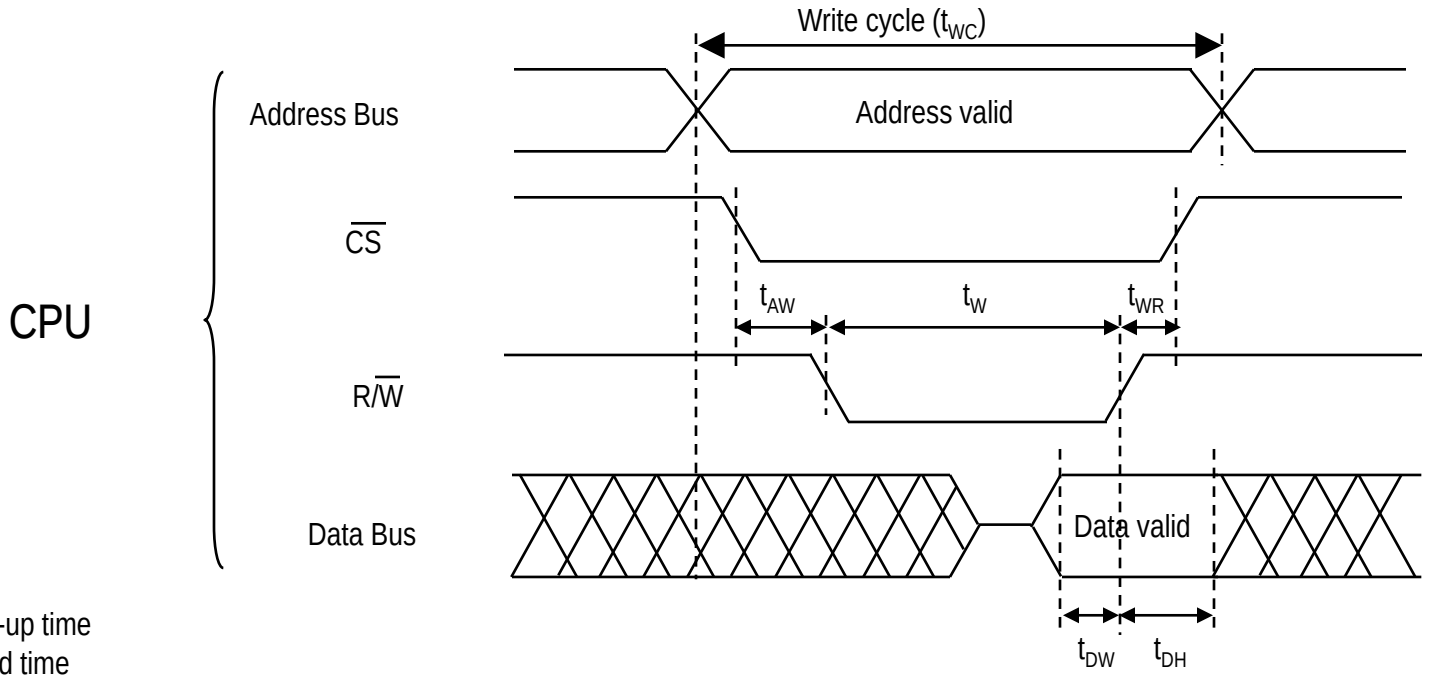
- Access time ($t_{tacc}$): from the transaction start (address valid at its bus) to the moment data is valid at its bus
- Read cycle ($t_{RC}$): minimum time between two consecutive accesses

$$t_{acc} \approx t_{RC}$$



# 3.5. Access M+T: Write timing diagram

- Write cycle ($t_{WC}$): minimum time between two consecutive accesses
- Writing pulse ($t_W$): minimum time of write enable to perform the write




$t_{WC}$ Write cycle
$t_W$ Writing pulse
$t_{DW}$ Writing set-up time
$t_{DH}$ Writing hold time
$t_{AW}$ Writing address set-up time
$t_{WR}$ Writing address hold time

# 3.5. Access M+T: example SRAM (1/5)

CYPRESS PERFORM

**CY7C1069AV33**

## 2M x 8 Static RAM

### Features

- **High speed**
  - **— $t_{AA}$ = 10, 12 ns**
- **Low active power**
  - **— 990 mW (max.)**
- **Operating voltages of 3.3 ± 0.3V**
- **2.0V data retention**
- **Automatic power-down when deselected**
- **TTL-compatible inputs and outputs**
- **Easy memory expansion with $\overline{CE}_1$ and $CE_2$ features**
- **Available in Pb-free and non Pb-free 54-pin TSOP II , non Pb-free 60-ball fine-pitch ball grid array (FBGA) package**

### Functional Description

The CY7C1069AV33 is a high-performance CMOS Static RAM organized as 2,097,152 words by 8 bits. Writing to the device is accomplished by enabling the chip (by taking $\overline{CE}_1$ LOW and $CE_2$ HIGH) and Write Enable ($\overline{WE}$) inputs LOW.

Reading from the device is accomplished by enabling the chip ($\overline{CE}_1$ LOW and $CE_2$ HIGH) as well as forcing the Output Enable ($\overline{OE}$) LOW while forcing the Write Enable ($\overline{WE}$) HIGH. See the truth table at the back of this data sheet for a complete description of Read and Write modes.

The input/output pins ($I/O_0$ through $I/O_7$) are placed in a high-impedance state when the device is deselected ($\overline{CE}_1$ HIGH or $CE_2$ LOW), the outputs are disabled ($\overline{OE}$ HIGH), or during a Write operation ($\overline{CE}_1$ LOW, $CE_2$ HIGH, and $\overline{WE}$ LOW).

The CY7C1069AV33 is available in a 54-pin TSOP II package with center power and ground (revolutionary) pinout, and a 60-ball fine-pitch ball grid array (FBGA) package.

# 3.5. Access M+T: example SRAM (2/5)

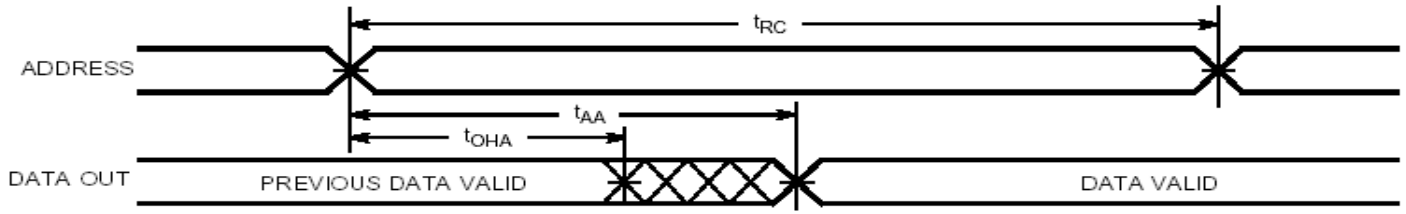
| Parameter | Description | –10 | | –12 | | Unit |
|---|---|---|---|---|---|---|
| | | Min. | Max. | Min. | Max. | |
| **Read Cycle** | | | | | | |
| $t_{power}$ | $V_{CC}$(typical) to the First Access[8] | 1 | | 1 | | ms |
| $t_{RC}$ | Read Cycle Time | 10 | | 12 | | ns |
| $t_{AA}$ | Address to Data Valid | | 10 | | 12 | ns |
| $t_{OHA}$ | Data Hold from Address Change | 3 | | 3 | | ns |
| $t_{ACE}$ | $\overline{CE}_1$ LOW/$CE_2$ HIGH to Data Valid | | 10 | | 12 | ns |
| $t_{DOE}$ | $\overline{OE}$ LOW to Data Valid | | 5 | | 6 | ns |
| $t_{LZOE}$ | $\overline{OE}$ LOW to Low-Z[9] | 1 | | 1 | | ns |
| $t_{HZOE}$ | $\overline{OE}$ HIGH to High-Z[9] | | 5 | | 6 | ns |
| $t_{LZCE}$ | $\overline{CE}_1$ LOW/$CE_2$ HIGH to Low-Z[9] | 3 | | 3 | | ns |
| $t_{HZCE}$ | $\overline{CE}_1$ HIGH/$CE_2$ LOW to High-Z[9] | | 5 | | 6 | ns |
| $t_{PU}$ | $\overline{CE}_1$ LOW/$CE_2$ HIGH to Power-up[10] | 0 | | 0 | | ns |
| $t_{PD}$ | $\overline{CE}_1$ HIGH/$CE_2$ LOW to Power-down[10] | | 10 | | 12 | ns |

# 3.5. Access M+T: example SRAM (3/5)

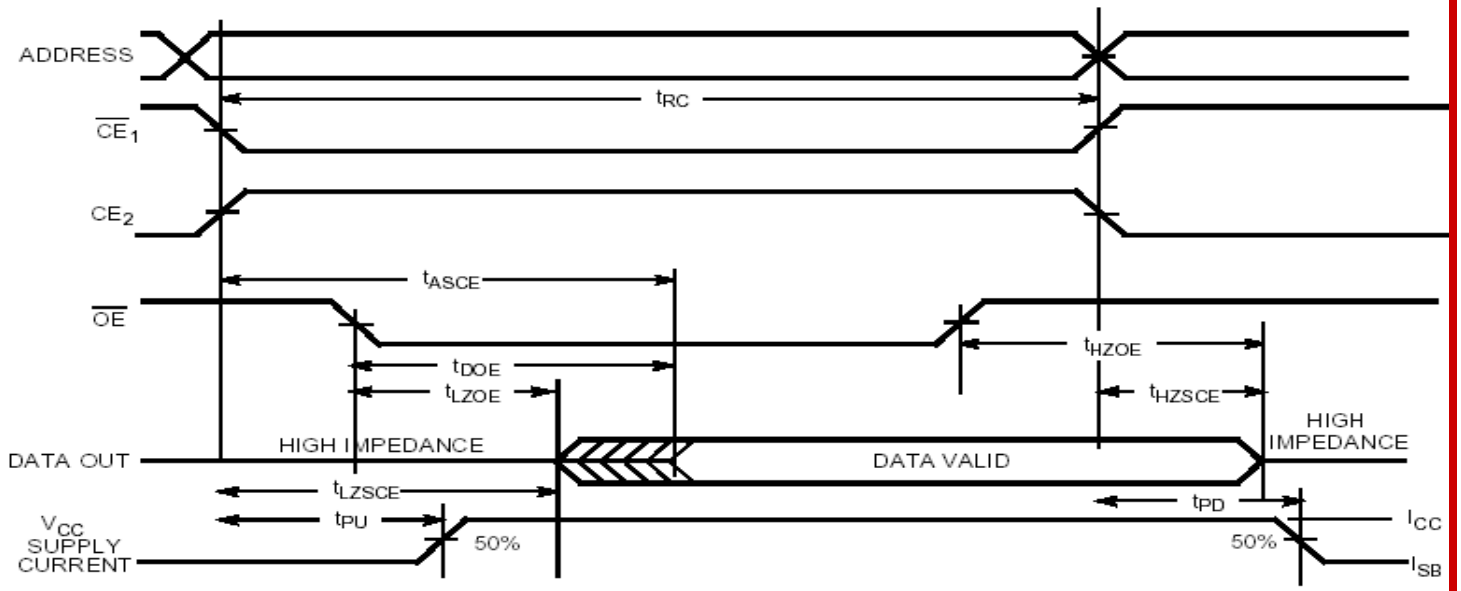
| Parameter |
|---|
| Read Cycle |
| $t_{power}$ |
| $t_{RC}$ |
| $t_{AA}$ |
| $t_{OHA}$ |
| $t_{ACE}$ |
| $t_{DOE}$ |
| $t_{LZOE}$ |
| $t_{HZOE}$ |
| $t_{LZCE}$ |
| $t_{HZCE}$ |
| $t_{PU}$ |
| $t_{PD}$ |

## Switching Waveforms

**Read Cycle No. 1**[13, 14]




**Read Cycle No. 2 ($\overline{OE}$ Controlled)**[14, 15]




**Notes:**
13. Device is continuously selected. $\overline{CE}_1 = V_{IL}$, $CE_2 = V_{IH}$.
14. $\overline{WE}$ is HIGH for Read cycle.
15. Address valid prior to or coincident with $\overline{CE}_1$ transition LOW and $CE_2$ transition HIGH.

# 3.5. Access M+T: example SRAM (4/5)

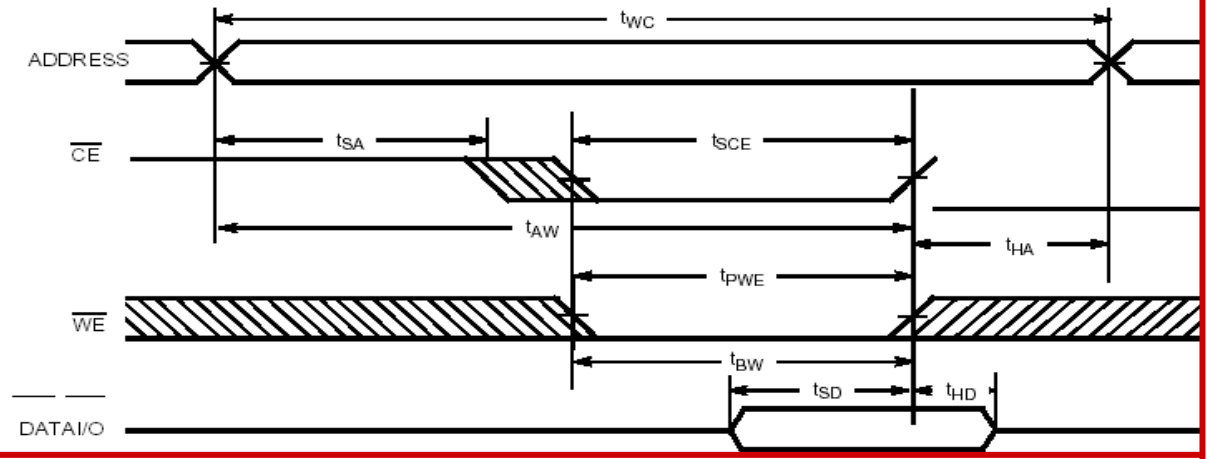
| **Write Cycle**[10, 11] | | | | | | |
|---|---|---|---|---|---|---|
| $t_{WC}$ | Write Cycle Time | 10 | | 12 | | ns |
| $t_{SCE}$ | $\overline{CE}_1$ LOW/$CE_2$ HIGH to Write End | 7 | | 8 | | ns |
| $t_{AW}$ | Address Set-up to Write End | 7 | | 8 | | ns |
| $t_{HA}$ | Address Hold from Write End | 0 | | 0 | | ns |
| $t_{SA}$ | Address Set-up to Write Start | 0 | | 0 | | ns |
| $t_{PWE}$ | $\overline{WE}$ Pulse Width | 7 | | 8 | | ns |
| $t_{SD}$ | Data Set-up to Write End | 5.5 | | 6 | | ns |
| $t_{HD}$ | Data Hold from Write End | 0 | | 0 | | ns |
| $t_{LZWE}$ | $\overline{WE}$ HIGH to Low-Z[9] | 3 | | 3 | | ns |
| $t_{HZWE}$ | $\overline{WE}$ LOW to High-Z[9] | | 5 | | 6 | ns |

# 3.5. Access M+T: example SRAM (5/5)

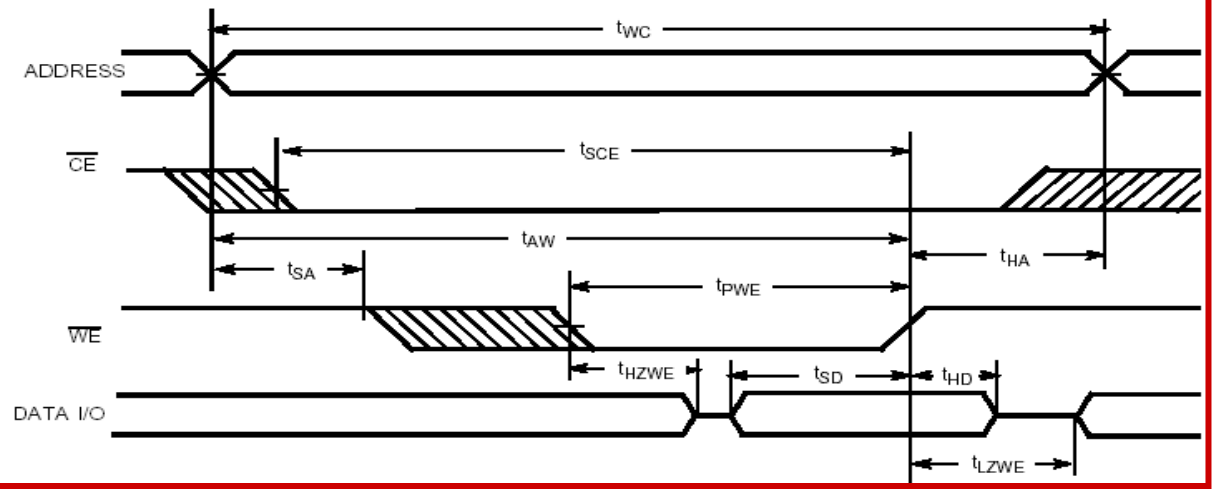
| Write Cycle[10, 11] |
|---|
| $t_{WC}$ |
| $t_{SCE}$ |
| $t_{AW}$ |
| $t_{HA}$ |
| $t_{SA}$ |
| $t_{PWE}$ |
| $t_{SD}$ |
| $t_{HD}$ |
| $t_{LZWE}$ |
| $t_{HZWE}$ |

**Switching Waveforms** (continued)

**Write Cycle No. 1 ($\overline{CE}_1$ Controlled)**[16, 17, 18]




**Write Cycle No. 2 ($\overline{WE}$ Controlled, $\overline{OE}$ LOW)**[16, 17, 18]



# 3.5. Access M+T: example EPROM (1/5)

FINAL

AMD

## Am27C040

### 4 Megabit (512 K x 8-Bit) CMOS EPROM

### DISTINCTIVE CHARACTERISTICS

- **Fast access time**
  - — Available in speed options as fast as 90 ns
- **Low power consumption**
  - — <10 µA typical CMOS standby current
- **JEDEC-approved pinout**
  - — Plug-in upgrade for 1 Mbit and 2 Mbit EPROMs
  - — Easy upgrade from 28-pin JEDEC EPROMs
- **Single +5 V power supply**
- **±10% power supply tolerance standard**
- **100% Flashrite™ programming**
  - — Typical programming time of 1 minute
- **Latch-up protected to 100 mA from –1 V to $V_{CC}$ + 1 V**
- **High noise immunity**
- **Compact 32-pin DIP, PDIP, PLCC packages**

### GENERAL DESCRIPTION

The Am27C040 is a 4 Mbit ultraviolet erasable programmable read-only memory. It is organized as 512K bytes, operates from a single +5 V supply, has a static standby mode, and features fast single address location programming. The device is available in windowed ceramic DIP packages and plastic one-time programmable (OTP) packages.

Data can be typically accessed in less than 90 ns, allowing high-performance microprocessors to operate without any WAIT states. The device offers separate Output Enable (OE#) and Chip Enable (CE#) controls, thus eliminating bus contention in a multiple bus microprocessor system.

AMD's CMOS process technology provides high speed, low power, and high noise immunity. Typical power consumption is only 100 mW in active mode, and 50 µW in standby mode.

All signals are TTL levels, including programming signals. Bit locations may be programmed singly, in blocks, or at random. The device supports AMD's Flashrite programming algorithm (100 µs pulses) resulting in typical programming time of 1 minute.

# 3.5. Access M+T: example EPROM (2/5)

| Family Part Number | Am27C040 | | | |
|---|---|---|---|---|
| Speed Options ($V_{CC}$ = 5.0 V ± 10%) | **-90** | **-120** | **-150** | **-200** |
| Max Access Time (ns) | 90 | 120 | 150 | 200 |
| CE# (E#) Access (ns) | 90 | 120 | 150 | 200 |
| OE# (G#) Access (ns) | 40 | 50 | 65 | 75 |

## PIN DESIGNATIONS

A0–A18 = Address Inputs

CE# (E#)/PGM# (P#)= Chip Enable/Program Enable Input

DQ0–DQ7 = Data Inputs/Outputs

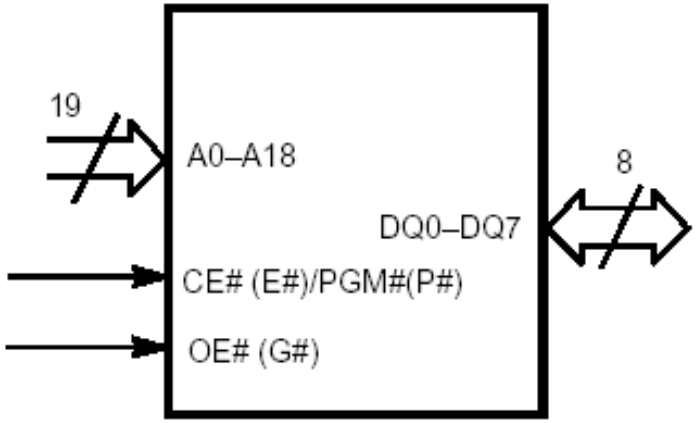
OE# (G#) = Output Enable Input

$V_{CC}$ = $V_{CC}$ Supply Voltage

$V_{PP}$ = Program Voltage Input

$V_{SS}$ = Ground

## LOGIC SYMBOL



# 3.5. Access M+T: example EPROM (3/5)

## FUNCTIONAL DESCRIPTION

### Device Erasure

In order to clear all locations of their programmed contents, the device must be exposed to an ultraviolet light source. A dosage of 30 W seconds/cm$^2$ is required to completely erase the device. This dosage can be obtained by exposure to an ultraviolet lamp — wavelength of 2537 Å — with intensity of 12,000 µW/cm$^2$ for 30 to 35 minutes. The device should be directly under and about one inch from the source and all filters should be removed from the UV light source prior to erasure.

Note that all UV erasable devices will erase with light sources having wavelengths shorter than 4000 Å, such as fluorescent light and sunlight. Although the erasure process happens over a much longer time period, exposure to any light source should be prevented for maximum system reliability. Simply cover the package window with an opaque label or substance.

### Device Programming

Upon delivery, or after each erasure, the device has all of its bits in the "ONE", or HIGH state. "ZEROs" are loaded into the device through the programming procedure.

The programming mode is entered when 12.75 V ± 0.25 V is applied to the $V_{PP}$ pin, CE#/PGM# is at $V_{IL}$ and OE# is at $V_{IH}$.

For programming, the data to be programmed is applied 8 bits in parallel to the data output pins.

The flowchart in the EPROM Products Data Book, Programming section (Section 5, Figure 5-1) shows AMD's Flashrite algorithm. The Flashrite algorithm reduces programming time by using a 100 µs programming pulse and by giving each address only as many pulses to reliably program the data. After each pulse is applied to a given address, the data in that address is verified. If the data does not verify, additional pulses are given until it verifies or the maximum pulses allowed is reached. This process is repeated while sequencing through each address of the device. This part of the algorithm is done at $V_{CC}$ = 6.25 V to assure that each EPROM bit is programmed to a sufficiently high threshold voltage. After the final address is completed, the entire EPROM memory is verified at $V_{CC} = V_{PP} = 5.25$ V.

Please refer to the EPROM Products Data Book, Section 5 for the programming flow chart and characteristics.

### Program Inhibit

Programming different data to multiple devices in parallel is easily accomplished. Except for CE#/PGM#, all like inputs of the devices may be common. A TTL low-level program pulse applied to one device's CE#/PGM# input with $V_{PP}$ = 12.75 V ± 0.25 V will program that particular device. A high-level CE#/PGM# input inhibits the other devices from being programmed.

### Program Verify

A verification should be performed on the programmed bits to determine that they were correctly programmed. The verify should be performed with OE# at $V_{IL}$, CE#/PGM# at $V_{IH}$, and $V_{PP}$ between 12.5 V and 13.0 V.

### Auto Select Mode

The autoselect mode provides manufacturer and device identification through identifier codes on DQ0–DQ7. This mode is primarily intended for programming equipment to automatically match a device to be programmed with its corresponding programming algorithm. This mode is functional in the 25°C ± 5°C ambient temperature range that is required when programming the device.

To activate this mode, the programming equipment must force $V_H$ on address line A9. Two identifier bytes may then be sequenced from the device outputs by toggling address line A0 from $V_{IL}$ to $V_{IH}$ (that is, changing the address from 00h to 01h). All other address lines must be held at $V_{IL}$ during the autoselect mode.

Byte 0 (A0 = $V_{IL}$) represents the manufacturer code, and Byte 1 (A0 = $V_{IH}$), the device identifier code. Both codes have odd parity, with DQ7 as the parity bit.

### Read Mode

To obtain data at the device outputs, Chip Enable (CE#/PGM#) and Output Enable (OE#) must be driven low. CE#/PGM# controls the power to the device and is typically used to select the device. OE# enables the device to output data, independent of device selection. Addresses must be stable for at least $t_{ACC}$–$t_{OE}$. Refer to the Switching Waveforms section for the timing diagram.

### Standby Mode

The device enters the CMOS standby mode when CE#/PGM# is at $V_{CC}$ ± 0.3 V. Maximum $V_{CC}$ current is reduced to 100 µA. The device enters the TTL-standby mode when CE#/PGM# is at $V_{IH}$. Maximum $V_{CC}$ current is reduced to 1.0 mA. When in either standby mode, the device places its outputs in a high-impedance state, independent of the OE# input.

### Output OR-Tieing

To accommodate multiple memory connections, a two-line control function is provided to allow for:

- Low memory power dissipation, and
- Assurance that output bus contention will not occur

CE#/PGM# should be decoded and used as the primary device-selecting function, while OE# be made a common connection to all devices in the array and connected to the READ line from the system control bus. This assures that all deselected memory devices are in their low-power standby mode and that the output pins are only active when data is desired from a particular memory device.

### System Applications

During the switch between active and standby conditions, transient current peaks are produced on the rising and falling edges of Chip Enable. The magnitude of these transient current peaks is dependent on the output capacitance loading of the device. At a minimum, a 0.1 µF ceramic capacitor (high frequency, low inherent inductance) should be used on each device between $V_{CC}$ and $V_{SS}$ to minimize transient effects. In addition, to overcome the voltage drop caused by the inductive effects of the printed circuit board traces on EPROM arrays, a 4.7 µF bulk electrolytic capacitor should be used between $V_{CC}$ and $V_{SS}$ for each eight devices. The location of the capacitor should be close to where the power supply is connected to the array.

## MODE SELECT TABLE

| Mode | | CE#/PGM# | OE# | A0 | A9 | $V_{PP}$ | Outputs |
|---|---|---|---|---|---|---|---|
| Read | | $V_{IL}$ | $V_{IL}$ | X | X | X | $D_{OUT}$ |
| Output Disable | | $V_{IL}$ | $V_{IH}$ | X | X | X | HIGH Z |
| Standby (TTL) | | $V_{IH}$ | X | X | X | X | HIGH Z |
| Standby (CMOS) | | $V_{CC}$ + 0.3 V | X | X | X | X | HIGH Z |
| Program | | $V_{IL}$ | $V_{IH}$ | X | X | $V_{PP}$ | $D_{IN}$ |
| Program Verify | | $V_{IL}$ | $V_{IL}$ | X | X | $V_{PP}$ | $D_{OUT}$ |
| Program Inhibit | | $V_{IH}$ | X | X | X | $V_{PP}$ | HIGH Z |
| Auto Select (Note 3) | Manufacturer Code | $V_{IL}$ | $V_{IL}$ | $V_{IL}$ | $V_H$ | X | 01h |
| | Device Code | $V_{IL}$ | $V_{IL}$ | $V_{IH}$ | $V_H$ | X | 9Bh |

***Note:***

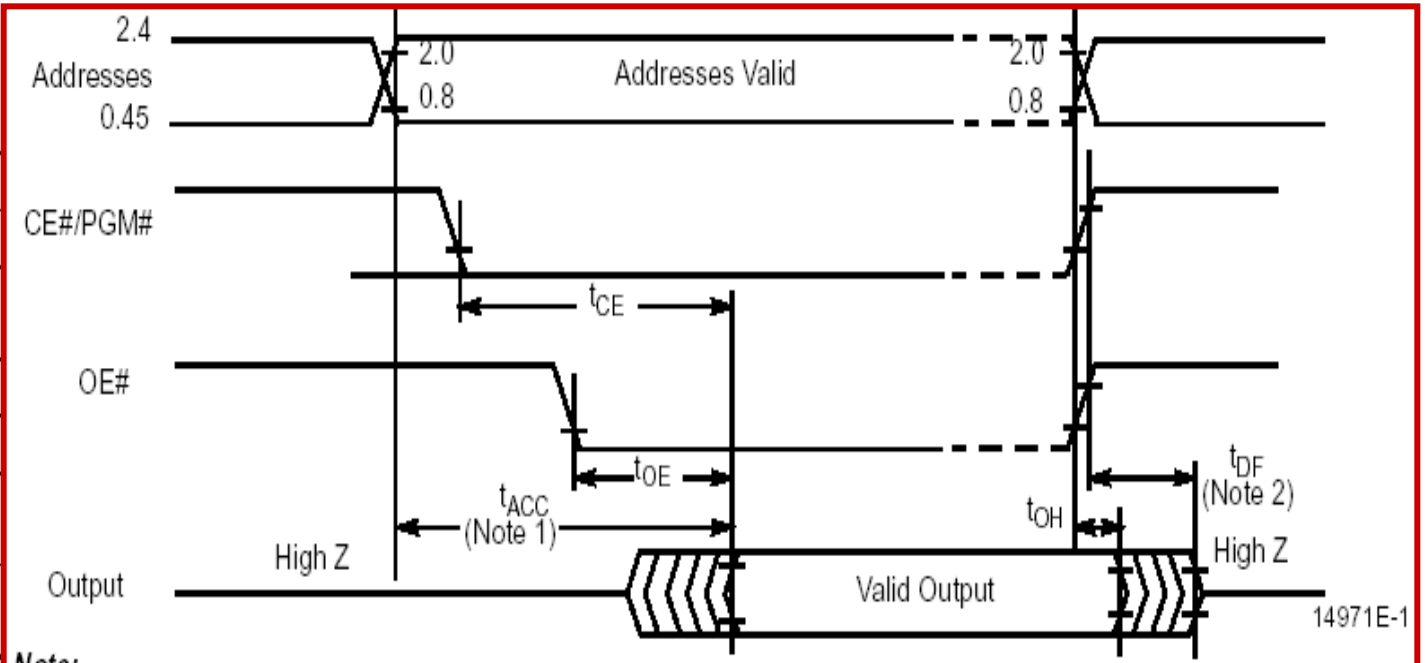
1. *$V_H$ = 12.0 V ± 0.5 V.*
2. *X = Either $V_{IH}$ or $V_{IL}$*
3. *A1 – A8 = A10 – A18 = $V_{IL}$*
4. *See DC Programming Characteristics in the EPROM Products Data Book for $V_{PP}$ voltage during programming*

# 3.5. Access M+T: example EPROM (4/5)

| Parameter Symbols | | Description | Test Setup | | Am27C040 | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|
| JEDEC | Std. | | | | -90 | -120 | -150 | -200 | |
| $t_{AVQV}$ | $t_{ACC}$ | Address to Output Delay | CE# = OE# = $V_{IL}$ | Max | 90 | 120 | 150 | 200 | ns |
| $t_{ELQV}$ | $t_{CE}$ | Chip Enable to Output Delay | OE# = $V_{IL}$ | Max | 90 | 120 | 150 | 200 | ns |
| $t_{GLQV}$ | $t_{OE}$ | Output Enable to Output Delay | CE# = $V_{IL}$ | Max | 40 | 50 | 65 | 75 | ns |
| $t_{EHQZ}$ $t_{GHQZ}$ | $t_{DF}$ (Note 2) | Chip Enable High or Output Enable High, Whichever Occurs First, to Output High Z | | Max | 30 | 30 | 30 | 40 | ns |
| $t_{AXQX}$ | $t_{OH}$ | Output Hold Time from Addresses, CE# or OE#, Whichever Occurs First | | Min | 0 | 0 | 0 | 0 | ns |

# 3.5. Access M+T: example EPROM (5/5)

| Parameter Symbols | |
|---|---|
| JEDEC | Std. |
| $t_{AVQV}$ | $t_{ACC}$ |
| $t_{ELQV}$ | $t_{CE}$ |
| $t_{GLQV}$ | $t_{OE}$ |
| $t_{EHQZ}$ $t_{GHQZ}$ | $t_{DF}$ (Note 2) |
| $t_{AXQX}$ | $t_{OH}$ |

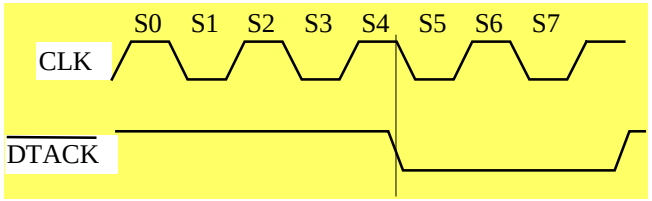



***Note:***

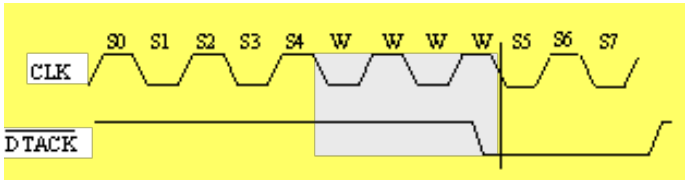
1. *OE# may be delayed up to $t_{ACC} - t_{OE}$ after the falling edge of the addresses without impact on $t_{ACC}$.*
2. *$t_{DF}$ is specified from OE# or CE#, whichever occurs first.*

# 3.5. Access management and timing: Cycle bus

- Cycle bus
  - It is the time the uP requires for carrying out one operation (read/write)
  - It is measured in clock cycles, and there is sometimes a control line informing that the operation is carried out in the minimum cycle bus
    - Example of a cycle bus: 4 clock cycles, and /DTACK control line




    - Example of an extended cycle bus: 2 extra clock cycles



# 3.5. Access management and timing: Cycle bus

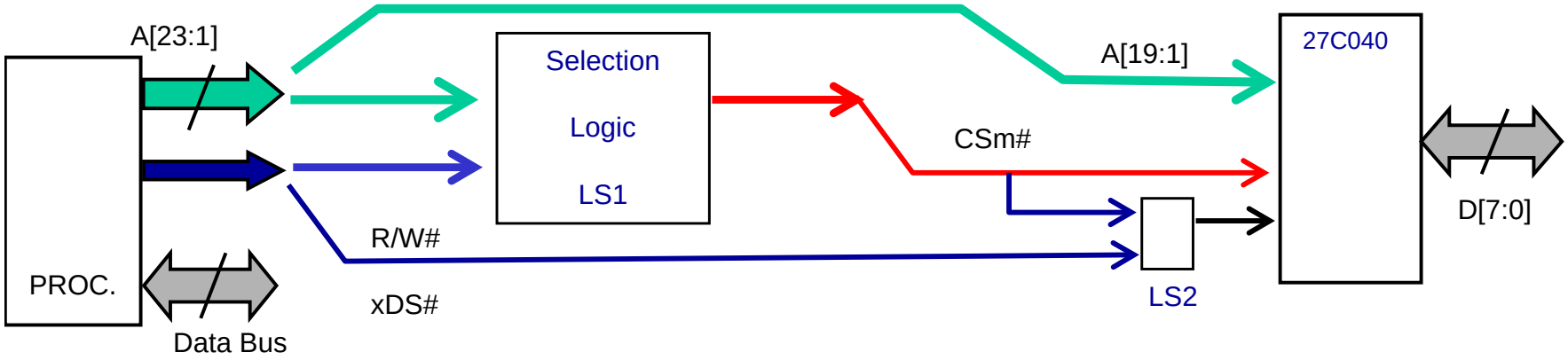
- How many wait cycles have to be added?
  - Comparing between two times:
    - Selected time ($t_{selection}$): time while the memory is selected by the uP
    - Access time memory ($t_{access}$): access time required by the memory for carrying out the selected operation
  - Two situations can happen:
    1. $t_{selection} \geq t_{access}$ ⇨ No extra cycles are required
    2. $t_{selection} < t_{access}$ ⇨ Extra cycles are required. How many?

Number of extra cycles=( $t_{access} - t_{selection}$)/$T_{clock}$

Round to the upper integer

# 3.5. Access management and timing: Cycle bus

- Example of a reading cycle
- Influence of circuit connections

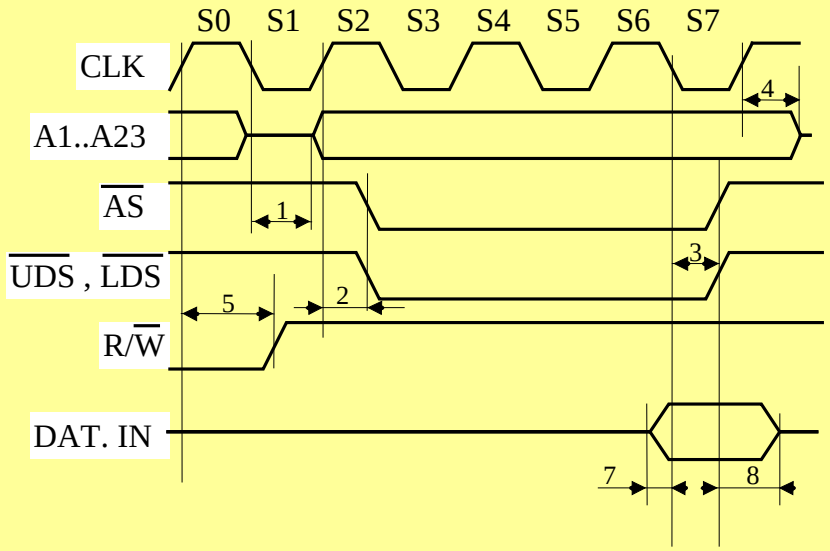



Delays: LS1, 15<td<29ns, and LS2 4<td<10ns

# 3.5. Access management and timing: Cycle bus

- Interpretation and management of timing schedule and uP signals (20 MHz)

**Reading cycle**

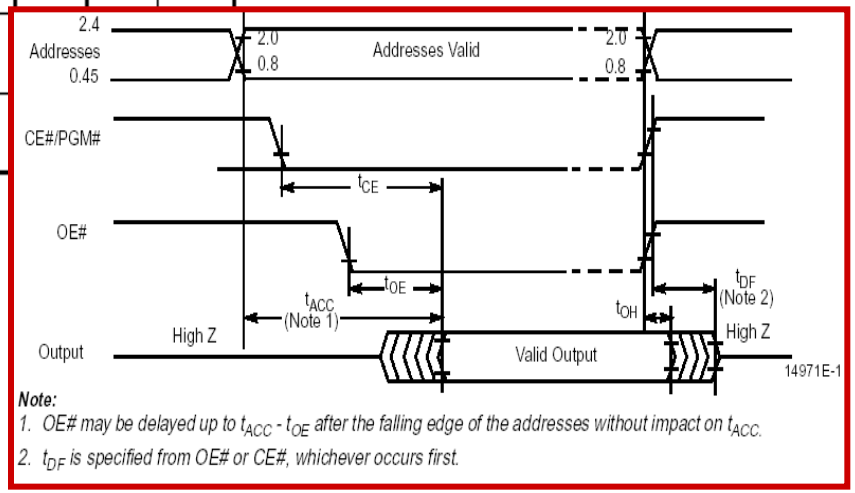



| | tmin (ns) | tmax (ns) |
|---|---|---|
| 1 | | 25 |
| 2 | 3 | 25 |
| 3 | 3 | 25 |
| 4 | 0 | |
| 5 | 0 | 25 |
| 6 | 0 | 25 |
| 7 | 5 | |
| 8 | 0 | |

# 3.5. Access management and timing: Cycle bus

- Checking in the memory datasheet the access time to load/store data ACCESS TIME

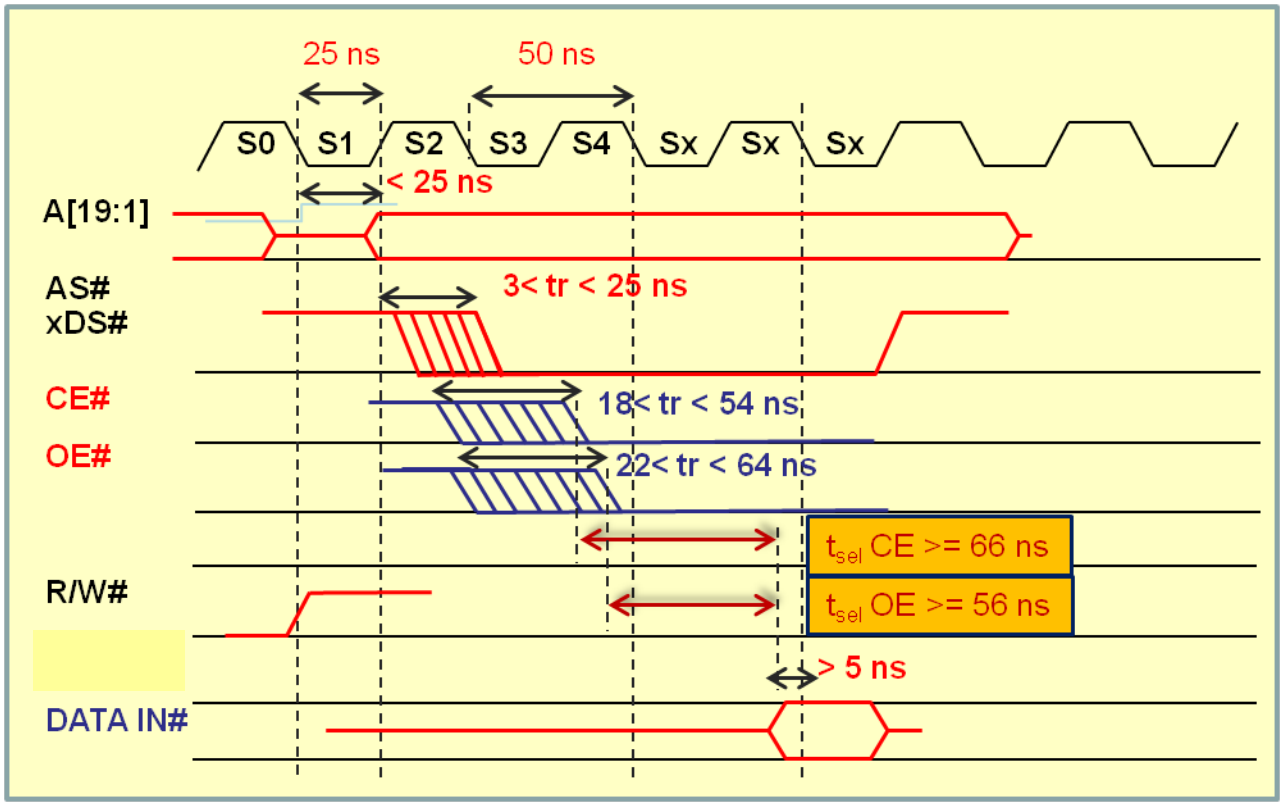
| Parameter Symbols | | Description | Test Setup | | Am27C040 | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|
| JEDEC | Std. | | | | -90 | -120 | -150 | -200 | |
| $t_{AVQV}$ | $t_{ACC}$ | Address to Output Delay | CE# = OE# = $V_{IL}$ | Max | 90 | 120 | 150 | 200 | ns |
| $t_{ELQV}$ | $t_{CE}$ | Chip Enable to Output Delay | OE# = $V_{IL}$ | Max | 90 | 120 | 150 | 200 | ns |
| $t_{GLQV}$ | $t_{OE}$ | Output Enable to Output Delay | CE# = $V_{IL}$ | Max | 40 | 50 | 65 | 75 | ns |
| $t_{EHQZ}$ $t_{GHQZ}$ | $t_{DF}$ (Note 2) | Chip Enable High or Output Enable High, Whichever Occurs First, to Output High Z | | Max | 30 | 30 | | | |
| $t_{AXQX}$ | $t_{OH}$ | Output Hold Time from Addresses, CE# or OE#, Whichever Occurs First | | Min | 0 | 0 | | | |




*Note:*
1. *OE# may be delayed up to $t_{ACC}$ - $t_{OE}$ after the falling edge of the addresses without impact on $t_{ACC}$.*
2. *$t_{DF}$ is specified from OE# or CE#, whichever occurs first.*

# 3.5. Access management and timing: Cycle bus

- Generate a timing schedule where intermediate uP signals in the circuit and chip select memory signal are included
- Compute the memory select time



# 3.5. Access management and timing: Cycle bus

- Check if memory selection time is larger than access time
- $t_{CE} < 120$ ns → $t_{OE} < 50$ ns

  - $t_{Sel\ OE} > t_{OE}$ (56-50 ns)
  - $t_{Sel\ CE} < t_{CE}$ (66-120= -54 ns)

As access time is larger than select time, it is necessary that CPU inserts wait cycles
The additional select time needed is 54 ns

As clock cycle is 50 ns, 2 wait clock cycles are inserted

**You must check in all types of memory in the circuit**