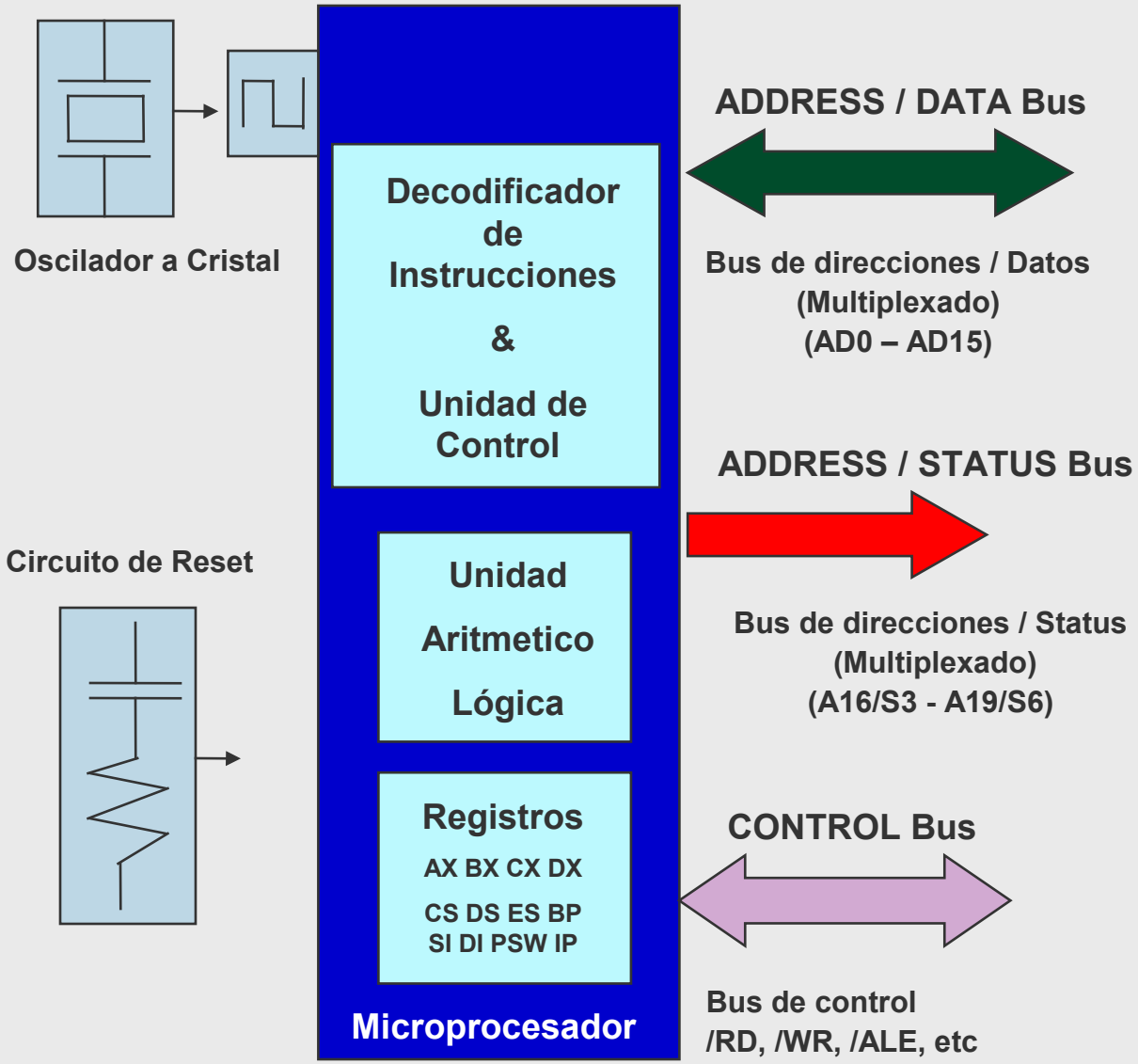
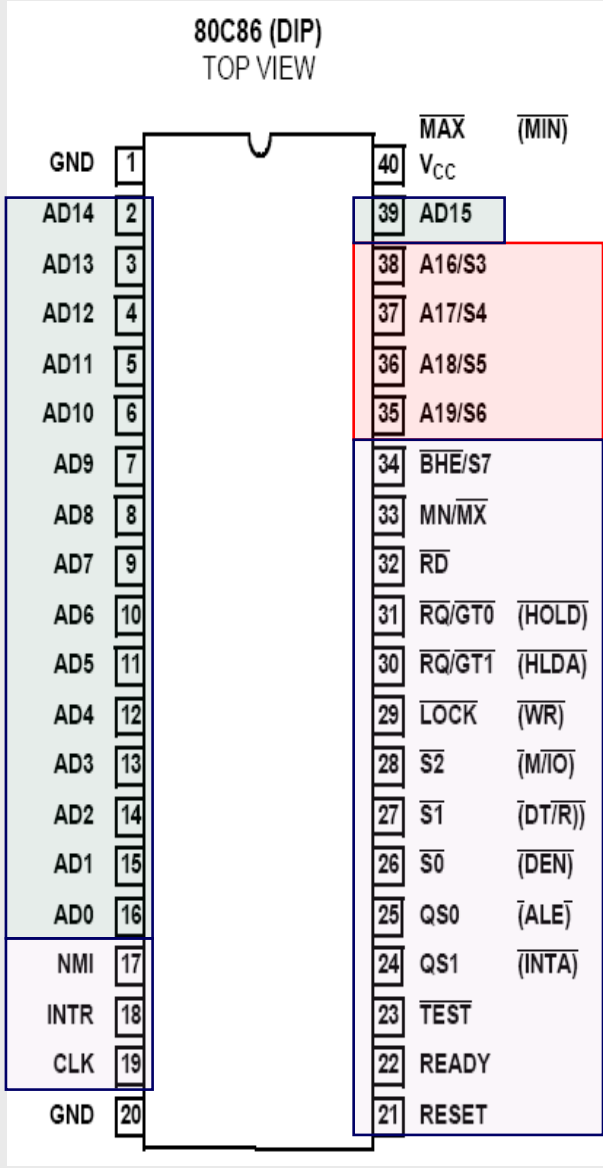
Digital III

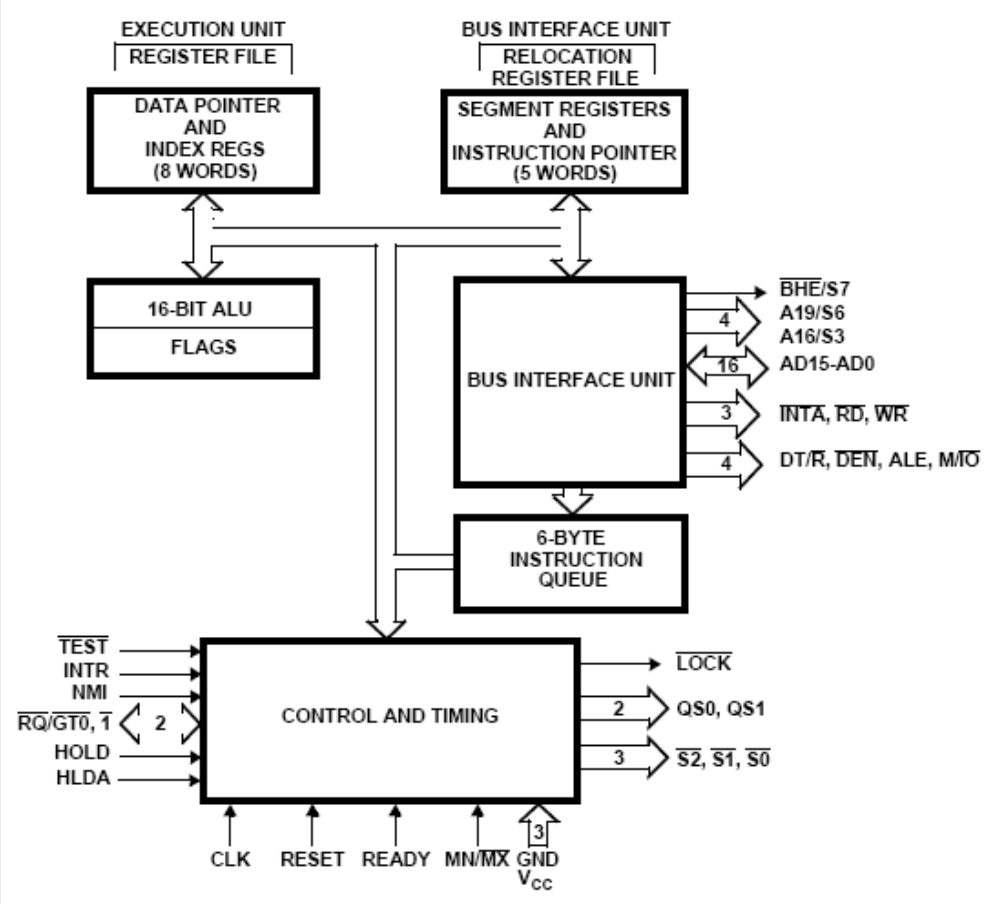
El Microprocesador i80c86

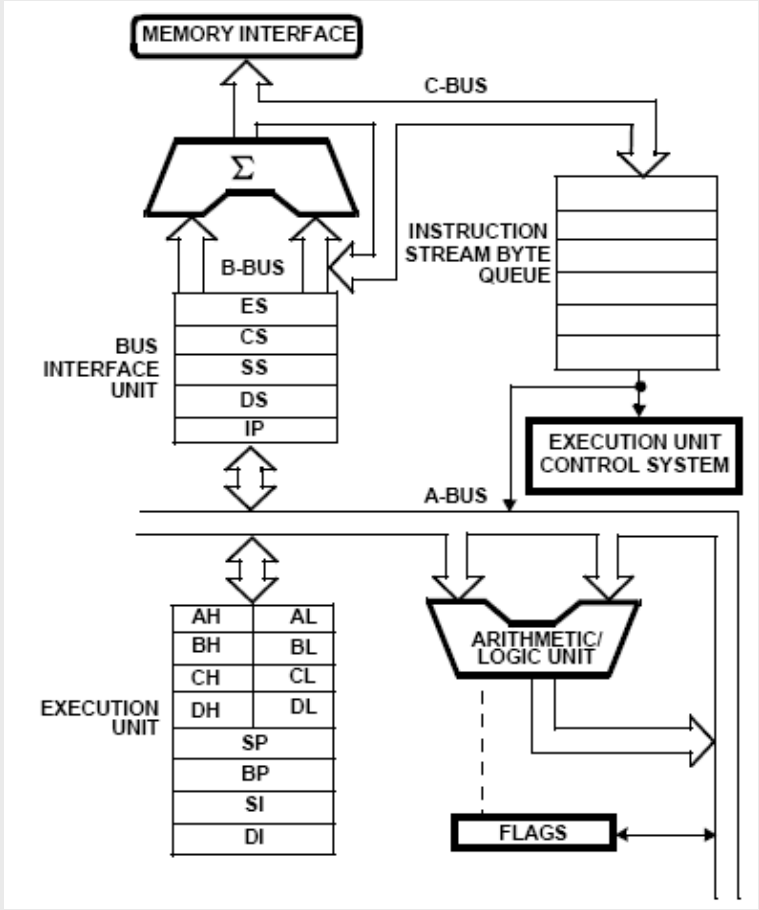
Pinout del Microprocesador i80c86





Arquitectura Interna i80c86





Generación de direcciones Físicas

(con segmentación)

Las direcciones constituyen el mecanismo que permite al microprocesador comunicarse con el resto de los dispositivos (memorias, periféricos, etc)

Address Bus (A0 - A19)

16 bits | 4 bits

SEGMENT Register | 0000

+

16 bits

OFFSET register

Physical Address

20 bits

Ej: jmp 0001h

(CS: 8000h)

(OFFSET : 0001h)

+

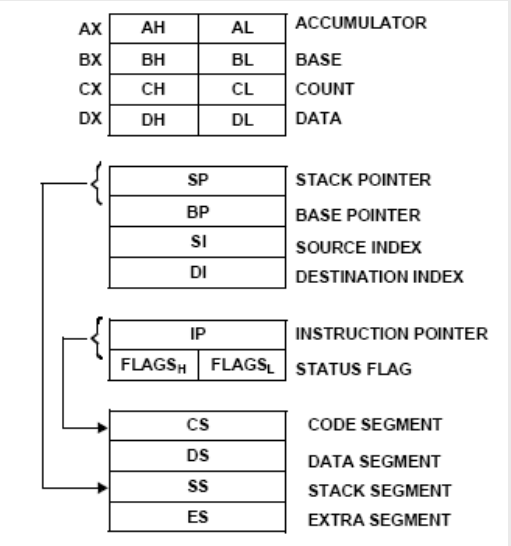
CS (8000 h) | 0 h

OFFSET (0001h)

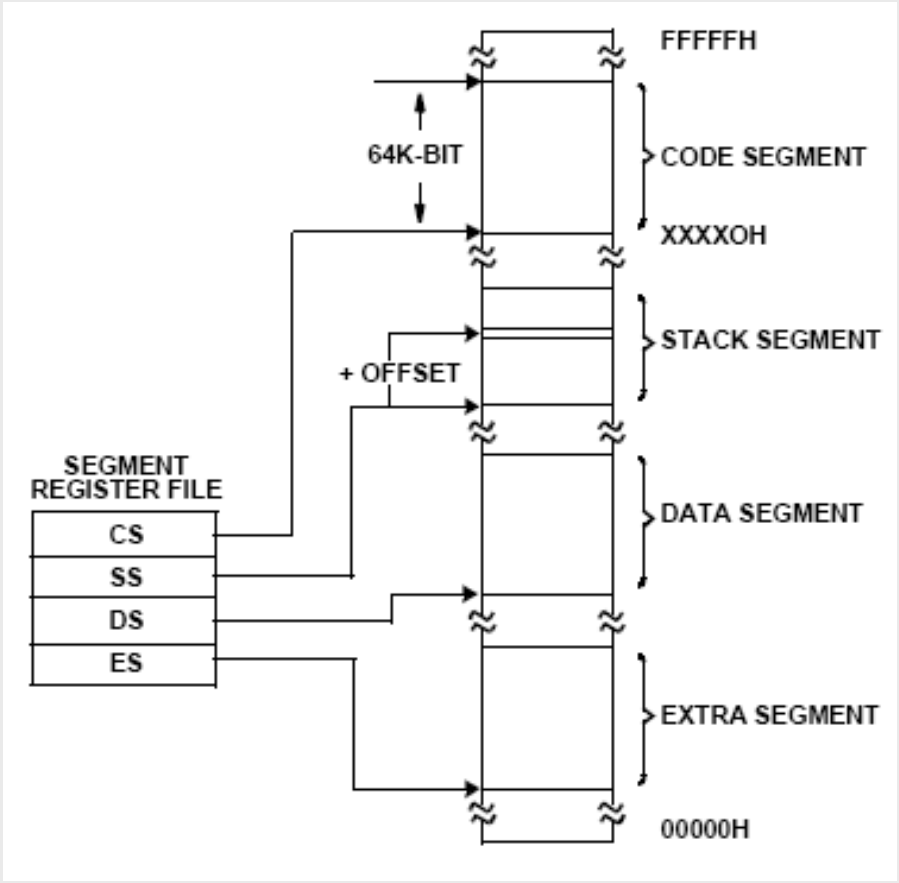
Dirección física (80001 h)

Generación de direcciones

Registros Internos



Registros de Segmento



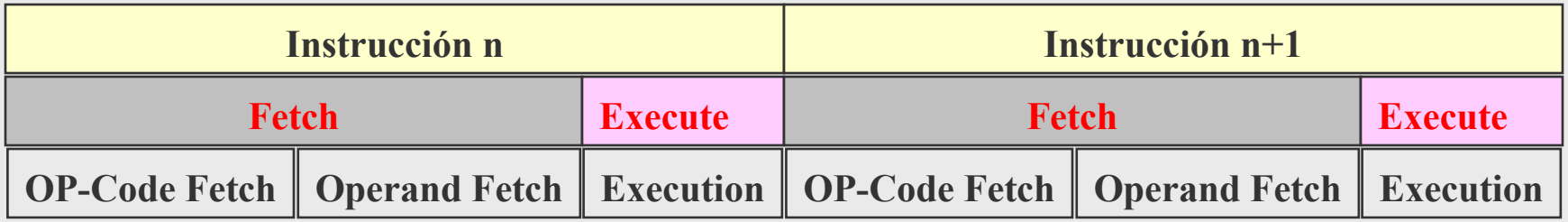
TYPE OF MEMORY REFERENCE	DEFAULT SEGMENT BASE	ALTERNATE SEGMENT BASE	OFFSET
Instruction Fetch	CS	None	IP
Stack Operation	SS	None	SP
Variable (except following)	DS	CS, ES, SS	Effective Address
String Source	DS	CS, ES, SS	SI
String Destination	ES	None	DI
BP Used As Base Register	SS	CS, DS, ES	Effective Address

Digital III

Ejecucion de Instrucciones

Ejecución de una instrucción

(sin Pre-Fetch)



OP-CODE Fetch

- Lectura del OP-CODE desde memoria de programa (Apuntado por CS : IP)
- Decodificación del OP-CODE de la instrucción
- Incremento del IP.
- Determinación de búsqueda de operandos extras para la ejecución

OPERAND Fetch (si existe)

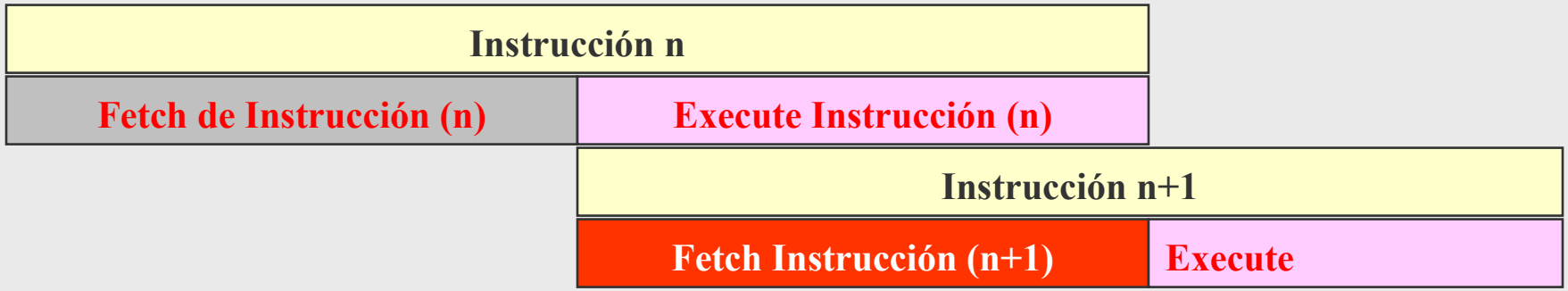
- Lectura del (o los) operando(s) desde memoria de programa (CS : IP + ...)
- Almacenamiento interno de los operandos.

Execution

- Ejecución de la instrucción.

Ejecución de instrucciones

(con Pre-Fetch)



OP-CODE Fetch (Instrucción n)

- Lectura del OP-CODE desde memoria de programa (CS : IP)
- Decodificación del OP-CODE de la instrucción
- Incremento del IP.
- Determinación de búsqueda de operandos extras para la ejecución
- Determina si en la ejecución hará uso de los buses.

Execution (Instrucción n)

- Ejecución de la instrucción (n).

... SI LA INSTRUCCIÓN (n) NO UTILIZA LOS BUSES.

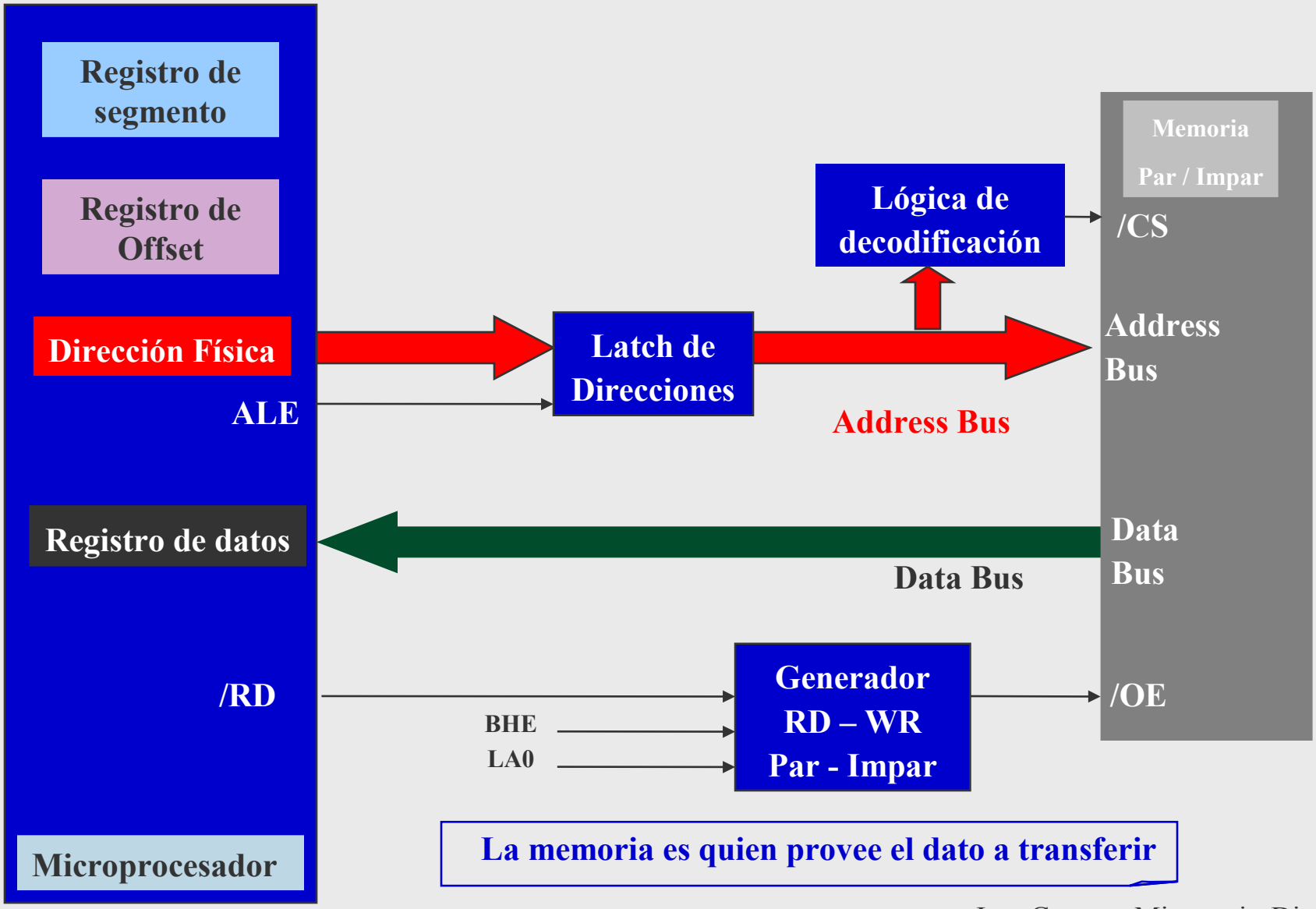
Pre-Fetch (Instrucción n + 1)

- Busqueda de las instrucción siguiente.
- Armado de la cola interna de ejecución.

Digital III

Accesos de Lectura y Escritura

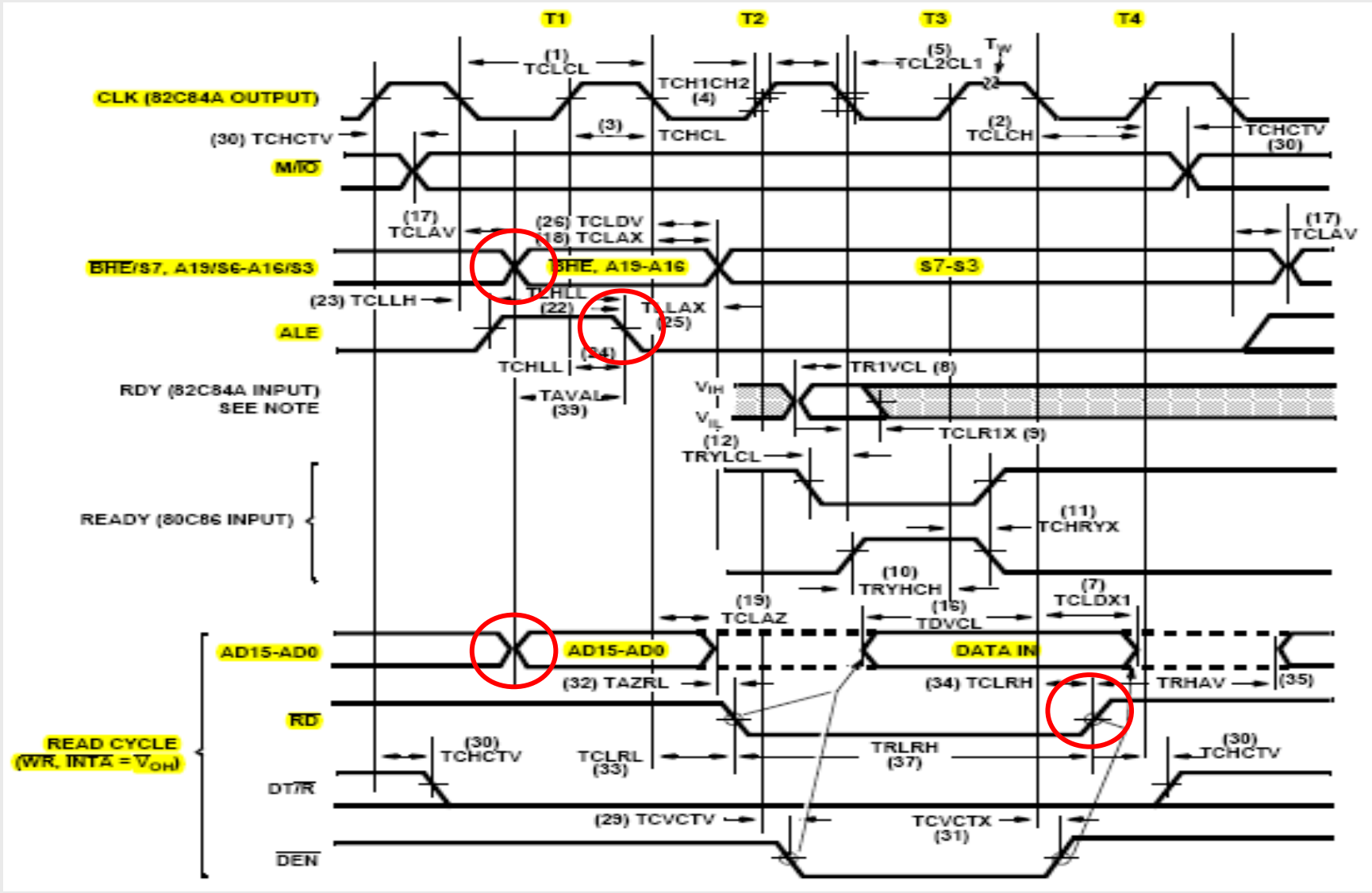
Acceso a memoria en lectura



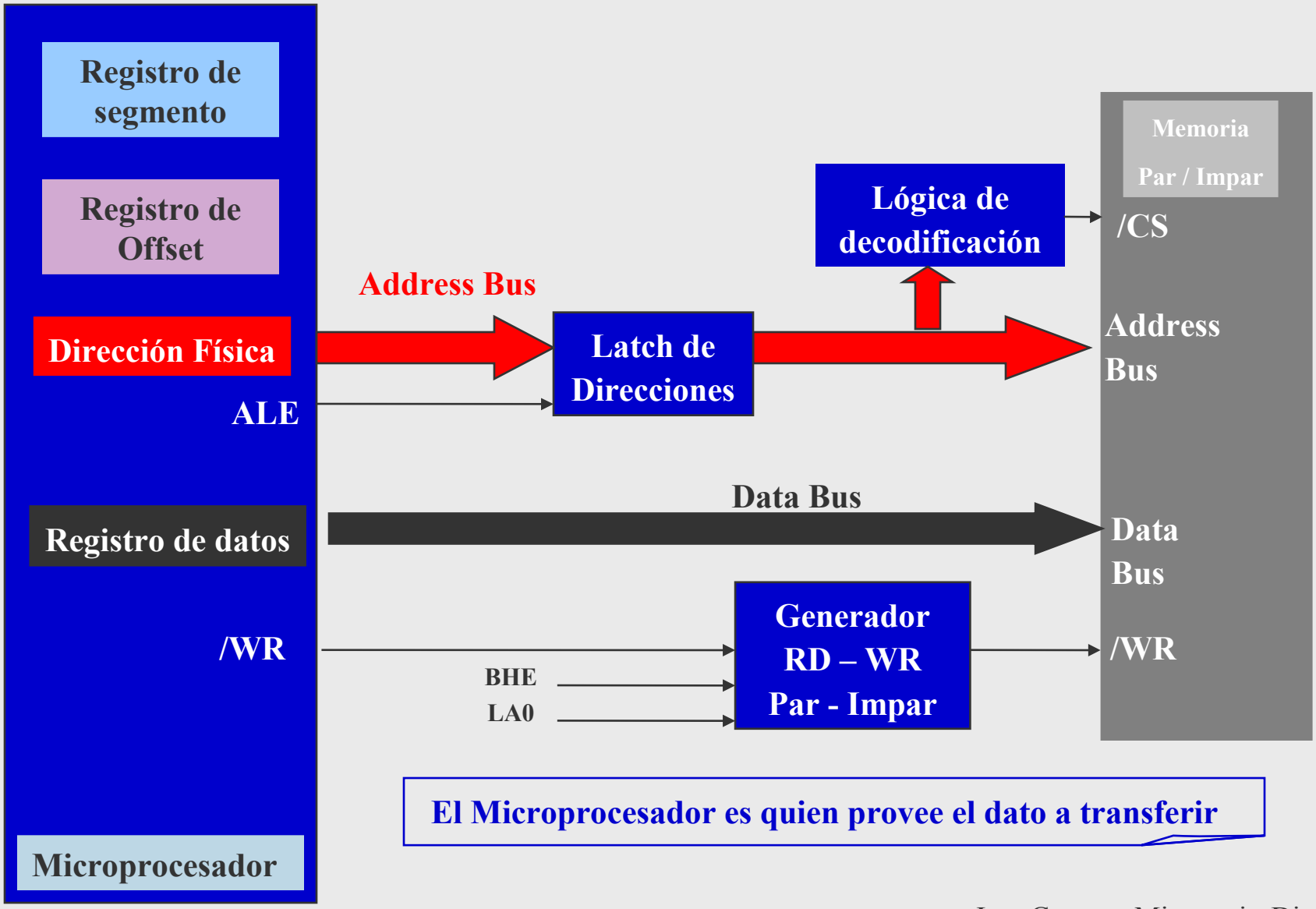
La memoria es quien provee el dato a transferir

Intel 80c86 en lectura

(Lectura de datos desde la memoria (o I/O) al uP)

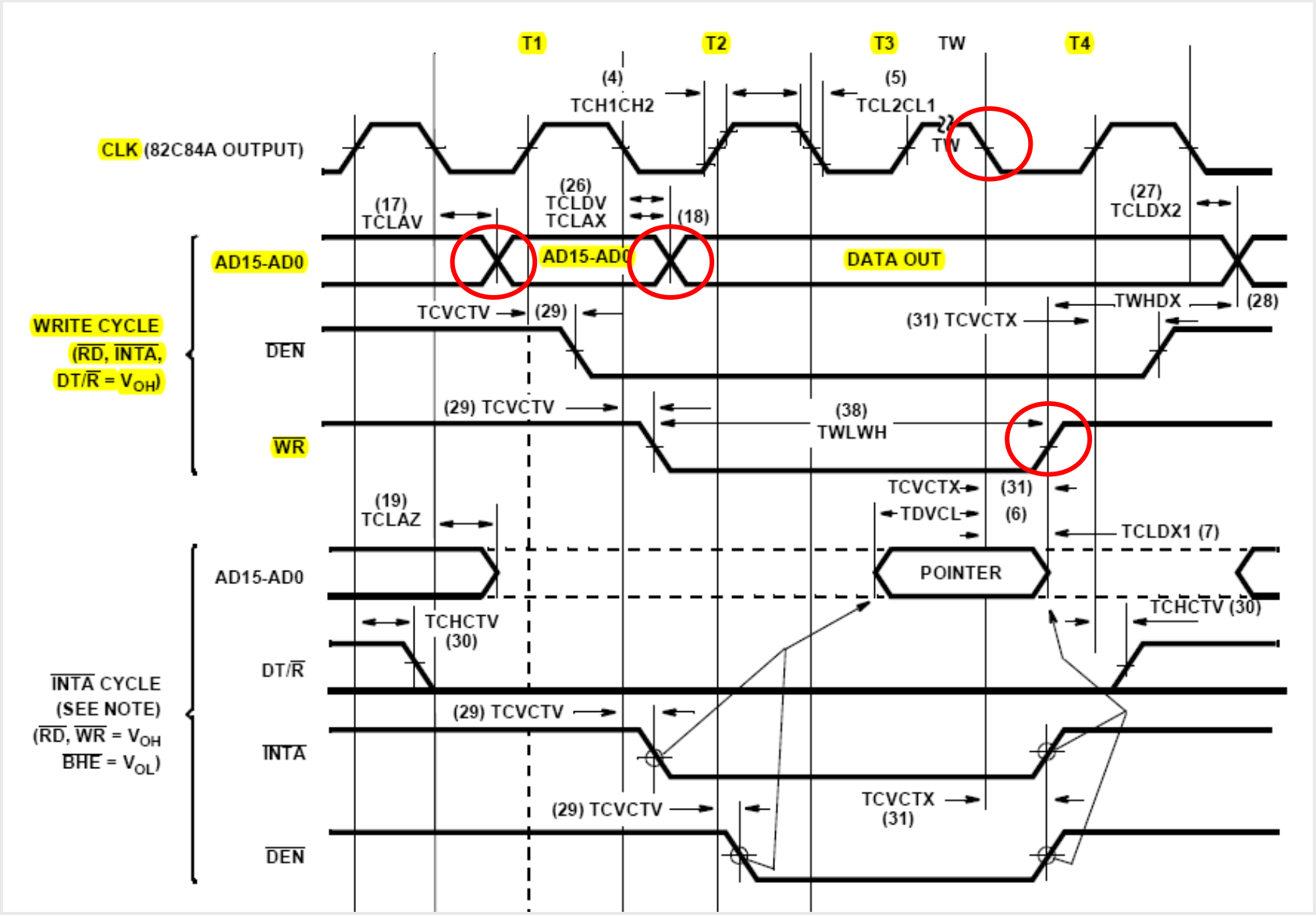


Acceso a memoria en escritura



Intel 80c86 en escritura

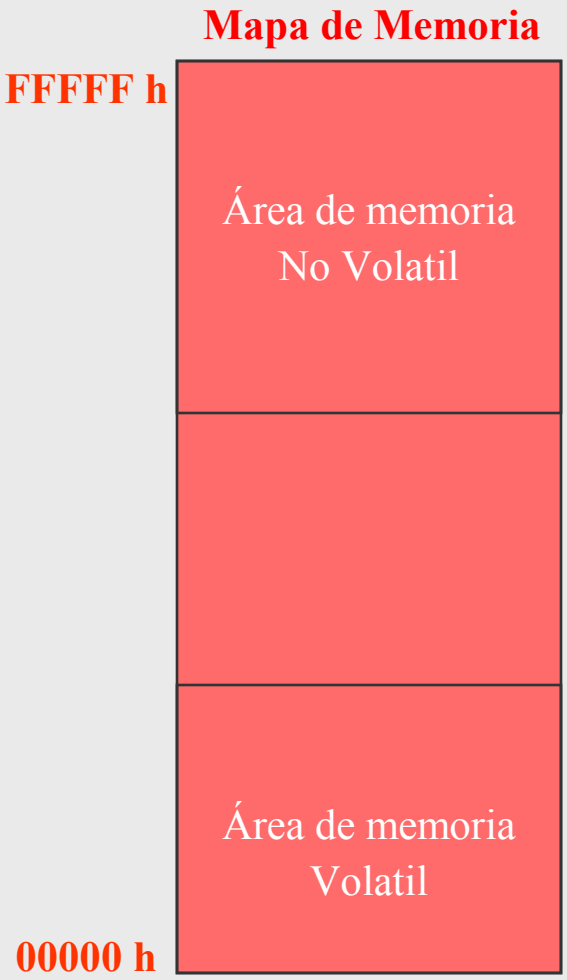
(Escritura de datos desde el uP a la memoria (o I/O))



Digital III

Mapeo de Dispositivos

Mapeo de dispositivos

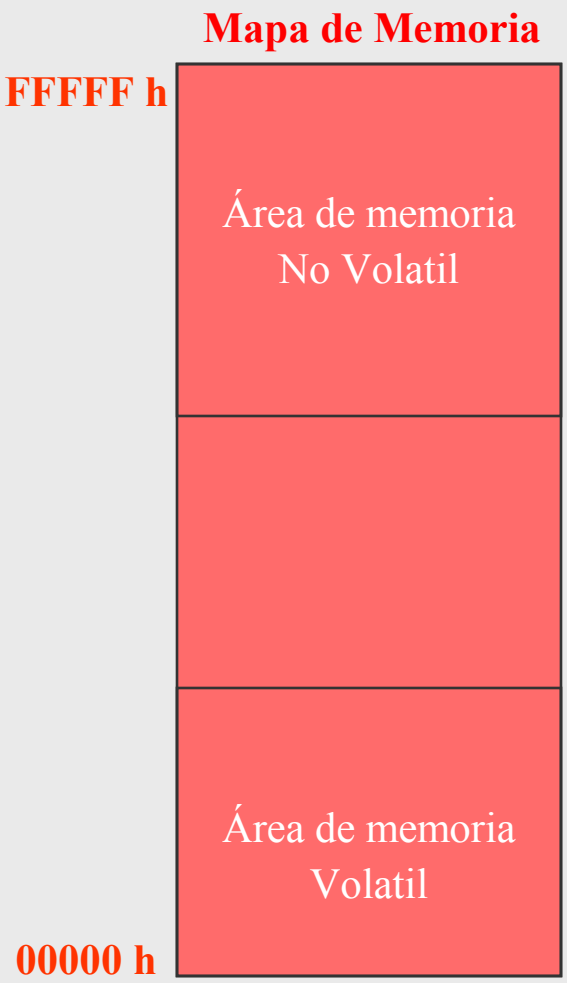


❇ ***Nota:***

El mapeo de un dispositivo consiste en asignarle un rango de direcciones dentro del mapa de memoria (o Entrada/Salida) donde se pueda acceder a sus registros internos.

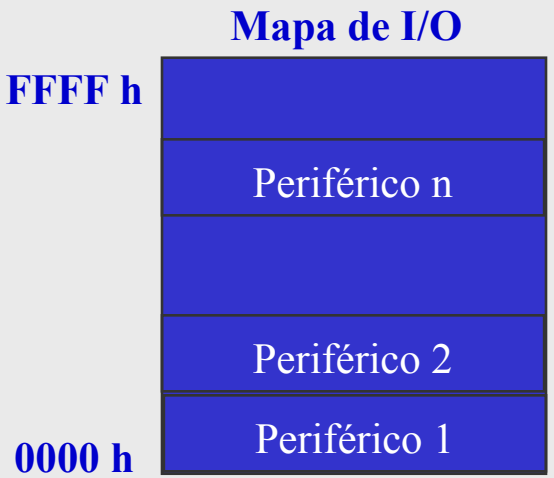
El Mapa de Memoria es el vínculo entre el software y el hardware.

Mapas de memoria y de I/O



❇ *Observación:*

Los espacios de direcciones de **MEMORIA** y de **ENTRADA/SALIDA** son espacios DISTINTOS y son accesibles con instrucciones distintas.



Accesible con instrucciones MOV, PUSH, etc.

Accesible con instrucciones IN y OUT

Ejemplos de mapeo en Microcontroladores

Dir. de registro	BANCO 0	BANCO 1	Dir. de registro
00h	Dir. Ind.[1]	Dir. Ind.[1]	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	-	-	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2[1]	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	68 REGISTROS DE PROPÓSITO GENERAL	MAPEADOS (ACCESO) EN EL BANCO 0	8Ch
4Fh			CFh
50h			D0h
7Fh			FFh

Localización de memoria no implementada, se lee como '0'

Nota 1: No es un registro físico

Microchip PIC16F84

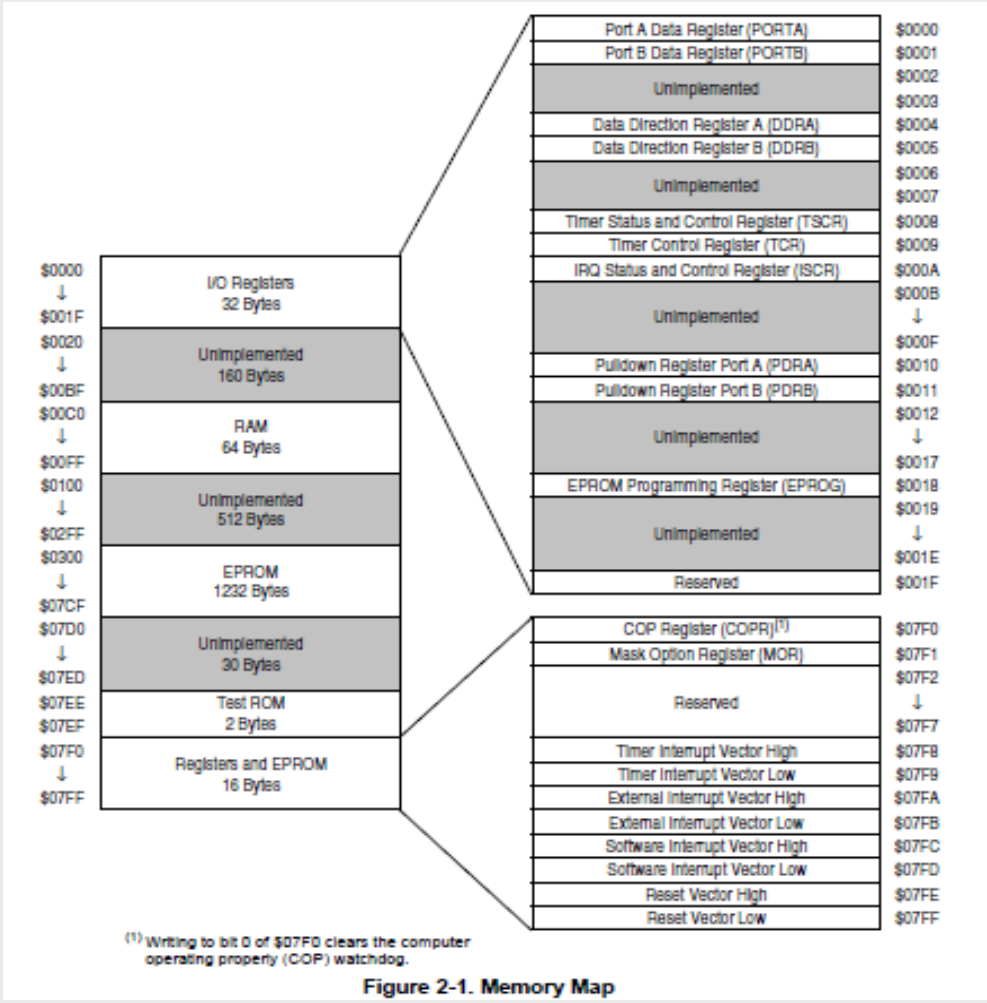


Figure 2-1. Memory Map

Motorola MC68HC705

Mapeo de dispositivos con decod. completa

Ej. Sistema con:

1 EPROM de 128K x 8. (17 líneas de direcciones)
(8 líneas de datos)

2 RAMs de 128K x 8. (17 líneas de direcciones)
(8 líneas de datos)

Decodificación total de direcciones									
	Líneas de direcciones del microprocesador								
	Rango	LA19	LA18	LA17	LA16	LA15	LA14 a LA2	LA1	LA0
EPROM *128 K*	FFFFFh a E0000h	1	1	1	A16	A15	A14 a A2	A1	A0
Area Libre	40000h a DFFFFh	0	1	x	x	x	xxxxxxx	x	x
RAM 1 *128 K*	20000h a 3FFFFh	0	0	1	A16	A15	A14 a A2	A1	A0
RAM 0 *128 K*	00000h a 1FFFFh	0	0	0	A16	A15	A14 a A2	A1	A0

Mapeo de dispositivos con espejado

Ej. Sistema con:

1 EPROM de 128K x 8. (17 líneas de direcciones)
(8 líneas de datos)

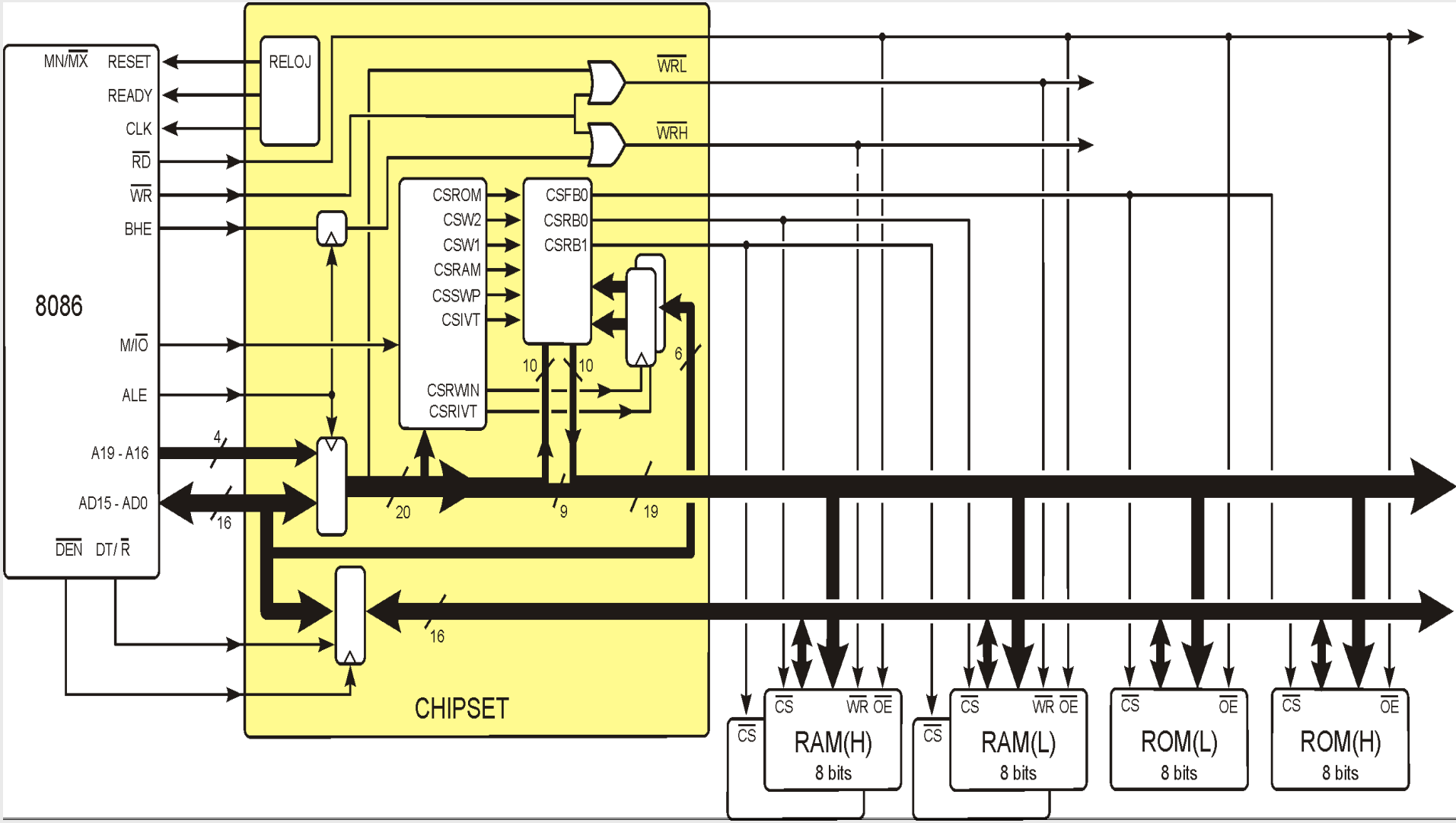
2 RAMs de 128K x 8. (17 líneas de direcciones)
(8 líneas de datos)

Decodificación parcial de direcciones									
	Líneas de direcciones del microprocesador								
	Rango	A19	A18	A17	A16	A15	A14 a A2	A1	A0
EPROM 128 K	FFFFFh a E0000h	1	X	X	A16	A15	A14 a A2	A1	A0
Espejo EPROM		1	X	X	A16	A15	A14 a A2	A1	A0
Espejo EPROM		1	X	X	A16	A15	A14 a A2	A1	A0
Espejo EPROM		1	X	X	A16	A15	A14 a A2	A1	A0
Espejo RAM 1		0	X	1	A16	A15	A14 a A2	A1	A0
Espejo RAM 0		0	X	0	A16	A15	A14 a A2	A1	A0
RAM 1 128 K	20000h a 3FFFFh	0	X	1	A16	A15	A14 a A2	A1	A0
RAM 0 128 K	00000h a 1FFFFh	0	X	0	A16	A15	A14 a A2	A1	A0

Digital III

Chipset Intel 8086
Implementación VHDL

Implementacion VHDL – D.B.



Implementación – Top Level

```
entity Top_ST_chipset is
  port (
        clk50           : in        std_logic;
--  Lado MICROPROCESADOR
        clk             : out       std_logic;
        ready           : out       std_logic;
        reset           : out       std_logic;
        rd              : in        std_logic;
        wr              : in        std_logic;
        mio             : in        std_logic;
        bhe             : in        std_logic;
        ale             : in        std_logic;
        adds            : in        std_logic_vector (3 downto 0);
        addata          : inout std_logic_vector (15 downto 0);
        den             : in        std_logic;
        dtr             : in        std_logic;
-- Lado SISTEMA
        csfb0           : out       std_logic;                  -- CS ROM
        csrb0           : out       std_logic;                  -- CS RAM banco 0
        csrb1           : out       std_logic;                  -- CS RAM banco 1
        wrL             : out       std_logic;
        wrH             : out       std_logic;
        address         : out       std_logic_vector (18 downto 0);
        data            : inout std_logic_vector (15 downto 0));

end Top_ST_chipset;
```

Implementación – Def. de Señales

```
architecture Behavioral of Top_ST_chipset is
------------------------------------------------------------
-- Definicion de Señales
------------------------------------------------------------
signal      csram                       : std_logic;
signal      csrom                       : std_logic;
signal      csw1, csw2, csivt, csswp    : std_logic;
signal      cs_rwin, cs_rivt            : std_logic;
signal      csivt_rom, csw1_rom, csw2_rom : std_logic;
signal      csivt_ram                   : std_logic;
signal      csw1_ram, csw2_ram          : std_logic;
signal      bhe_sg                      : std_logic;
signal      sig_addr                    : std_logic_vector (19 downto 0);

signal      iowr                        : std_logic;
signal      s_rwin                      : std_logic_vector (5 downto 0) := "000000";
signal      s_rivt                      : std_logic_vector (2 downto 0) := "000";
```

Implementación – Generador Clk

```
-----------------------------------------------------------------
-- Generador de reloj del Microprocesador
-----------------------------------------------------------------
process (clk50)
variable estado : integer := 0;
variable res_cnt : integer := 8;
begin
        if clk50 = '1' and clk50'event then              -- Flanco ascendente
                case estado is
                        when 0 =>
                                estado := 1;
                                clk  <= '1';
                        when 1 =>
                                estado := 2;
                        when 2 =>
                                estado := 3;
                                clk  <= '0';
                        when 3 =>
                                estado := 4;
                        when 4 =>
                                estado := 5;
                        when 5 =>
                                estado := 6;
                        when others =>
                                estado := 0;
                end case;
```

Implementación – Generador Reset

```
-----------------------------------------------------------------
-- Generador de Reset del Microprocesador
-----------------------------------------------------------------

                if res_cnt = 0 then
                        reset <= '1';
                else
                        res_cnt := res_cnt - 1;
                        reset <= '0';
                end if;
        end if;

end process;
```

Implementacion – Interfase Bus

```
-------------------------------------------------------
-- Cerrojo de  de direcciónes y manejo de BUS
-------------------------------------------------------

sig_addr    <= adds & addata        when        ale = '1';
bhe_sg      <= bhe                  when        ale = '1';

addata      <=    data when (den = '0') and (dtr = '0') else
                                          "ZZZZZZZZZZZZZZZZ";

data        <=    addata when (den = '0') and (dtr = '1') else
                                          "ZZZZZZZZZZZZZZZZ";
```

Implementación
Lógica de Selección Memorias I

```
-------------------------------------------------------------
--      CS de los bancos de memoria
-------------------------------------------------------------

csivt_rom       <= '0' when (csivt = '0') and not(s_rivt(1 downto 0) = "11") else '1';
csw1_rom        <= '0' when (csw1 = '0') and (s_rwin(2) = '0') else '1';
csw2_rom        <= '0' when (csw2 = '0') and (s_rwin(5) = '0') else '1';
csfb0           <= csrom and csivt_rom and csw1_rom and csw2_rom;

csivt_ram       <= '0' when (csivt = '0') and (s_rivt(1 downto 0) = "11") else '1';
csrb0           <= csram and csivt_ram and csswp;

csw1_ram        <= '0' when (csw1 = '0') and (s_rwin(2 downto 1) = "10") else '1';
csw2_ram        <= '0' when (csw2 = '0') and (s_rwin(5 downto 4) = "10") else '1';
csrb1           <= csw1_ram and csw2_ram;
```

Implementación
Lógica de Selección Memorias II

```
-------------------------------------------------------------------------
--      Control de acceso a BYTE par o impar mediante WR
-------------------------------------------------------------------------
wrH <= wr or bhe_sg;
wrL <= wr or sig_addr (0);


-------------------------------------------------------------------------
--      Generación de direcciones para los bancos
-------------------------------------------------------------------------
address(8 downto 0) <= sig_addr (9 downto 1);
address(18 downto 9) <=  '0' & sig_addr (18 downto 10) when csrom = '0'                  else
                         '0' & sig_addr (18 downto 10) when csram = '0'                  else
                         '1' & s_rwin(1 downto 0) & sig_addr (16 downto 10) when csw1_rom = '0' else
                         '1' & s_rwin(4 downto 3) & sig_addr (16 downto 10) when csw2_rom = '0' else
                         "00" & s_rwin(0) & sig_addr (16 downto 10) when csw1_ram = '0'  else
                         "01" & s_rwin(2) & sig_addr (16 downto 10) when csw2_ram = '0'  else
                         "10000000" & s_rivt(1 downto 0) when csivt_rom = '0'            else
                         "0000000000"  when csivt_ram = '0'                              else
                         "000000000" & s_rivt(2) when csswp = '0';
```

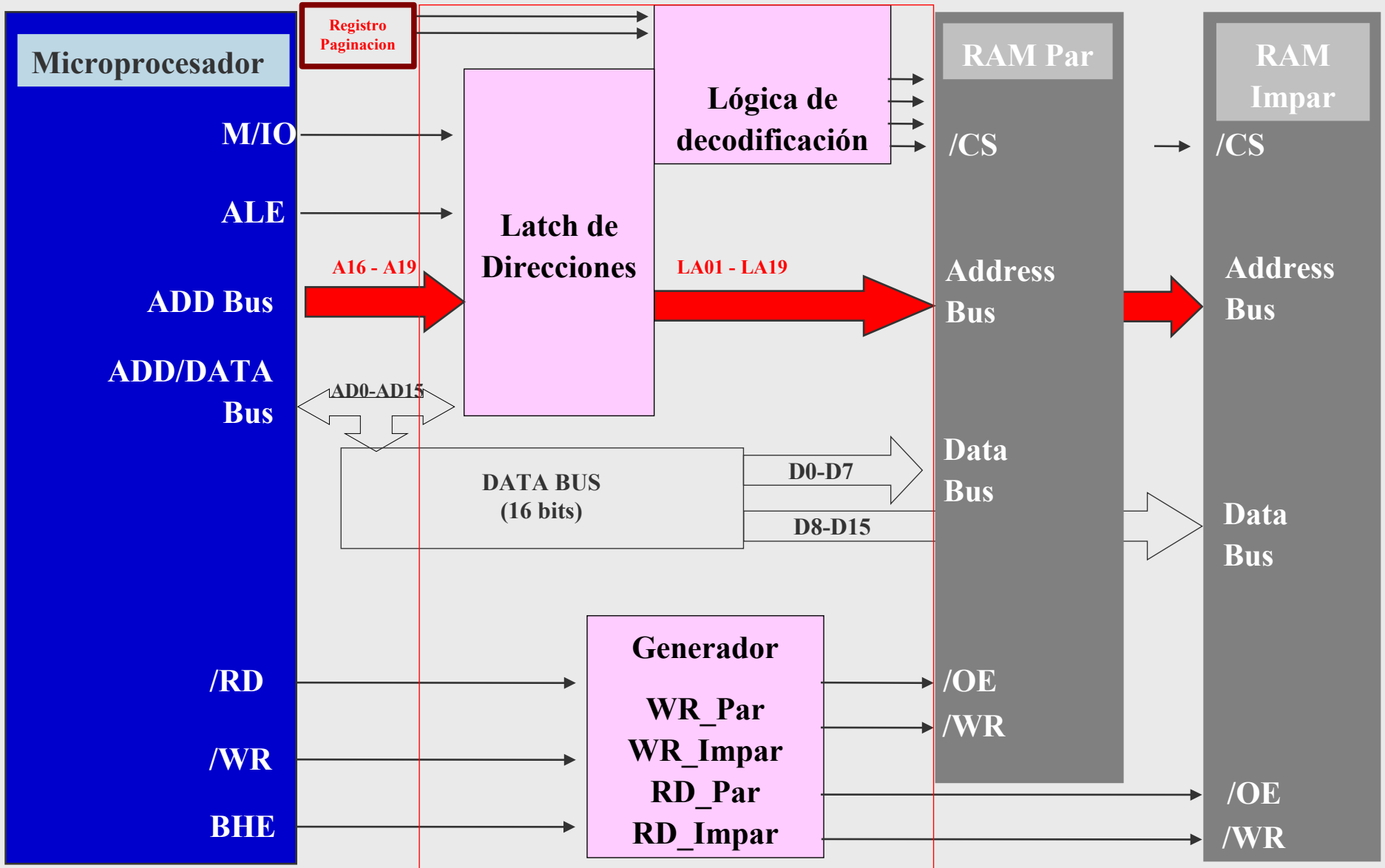
Implementación – Registros IO

```
-------------------------------------------------
--  Registros y mapeo de entrada - salida
-------------------------------------------------
cs_rwin    <= '0' when (mio = '0')and(sig_addr (15 downto 12) = "0000") else '1';
cs_rivt    <= '0' when (mio = '0')and(sig_addr (15 downto 12) = "0001") else '1';
iowr <= mio or wr;

process (iowr)          -- Este  proceso se activa con las escrituras sobre entrada - salida
begin
      if (iowr = '1' and iowr'event)  then          -- Flanco de ascendente
            if cs_rwin = '0' then
                  s_rwin <= addata (5 downto 0);
            end if;
            if cs_rivt = '0' then
                  s_rivt <= addata (2 downto 0);
            end if;
      end if;
end process;
end Behavioral;
```

Implementacion

```
-------------------------------------------------------------
-- Conexion de señales con lineas externas
-------------------------------------------------------------
sALE<= ALE;
sMIO<= MIO;             -- MIO: 1 -> M -  MIO: 0 -> IO
sINTA       <= INTA;
INTR<= sINTR;
NMI  <= sNMI;
sRD   <= RD;
sWR  <= WR;
--
CS     <= sCS;
PCS   <= sPCS;
--
LADD        <= sLADD;
RD_H        <= sRD_H;
RD_L        <= sRD_L;
WR_H        <= sWR_H;
WR_L        <= sWR_L;

CLK_uP     <= sCLK_uP;
RESET_uP<= sRESET_uP;
```

Implementacion con Paginacion



Para Analizar y Resolver

1. En función de la descripción VHDL dada,
realizar el **Mapa de Memoria** completo del sistema

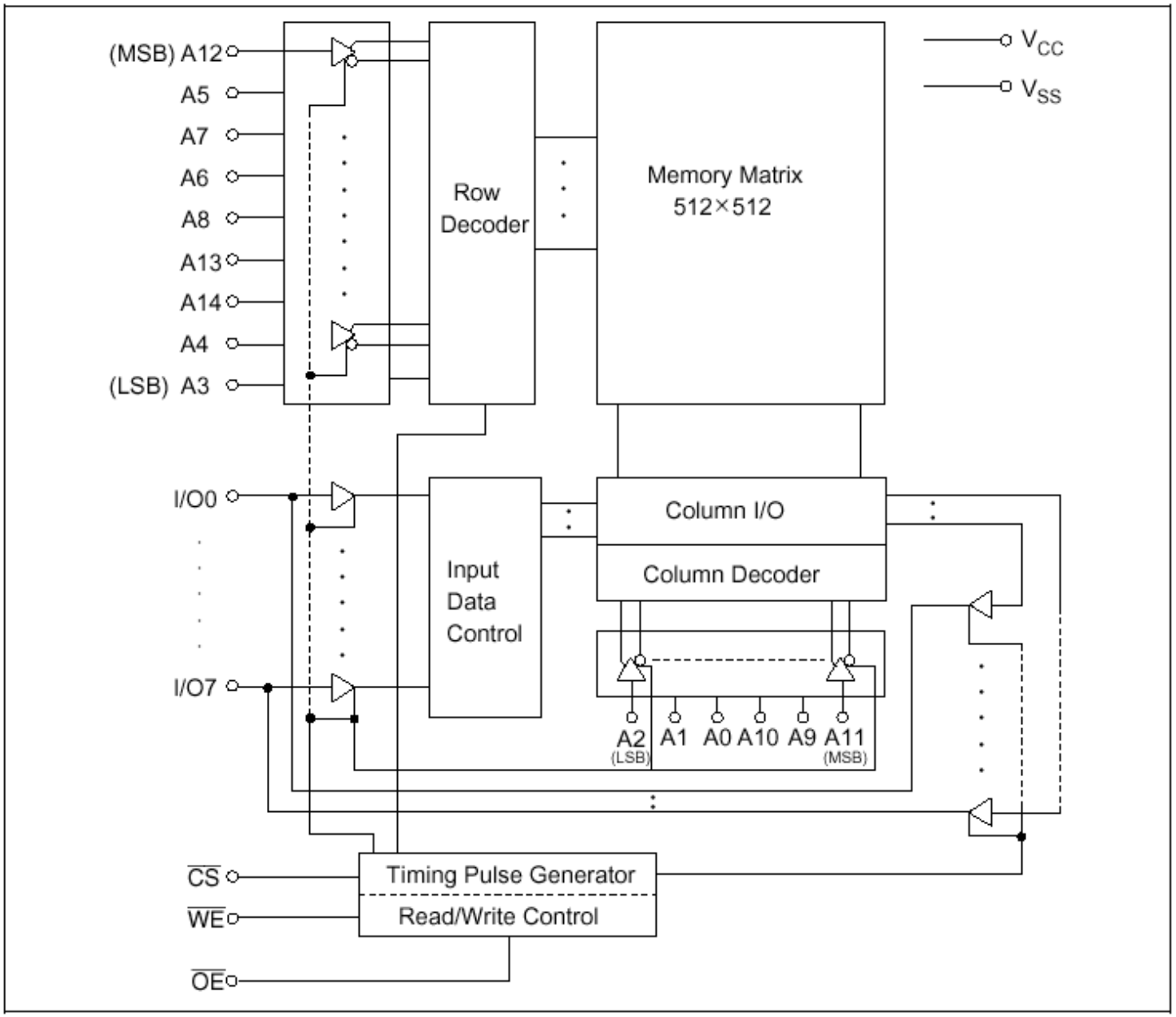
2. En función de la descripción VHDL dada,
realizar el **Mapa de Entrada/Salida** completo del sistema

ANALIZAR JUNTO CON LA HOJA DE DATOS DEL MICROPROCESADOR

Electrónica Digital III

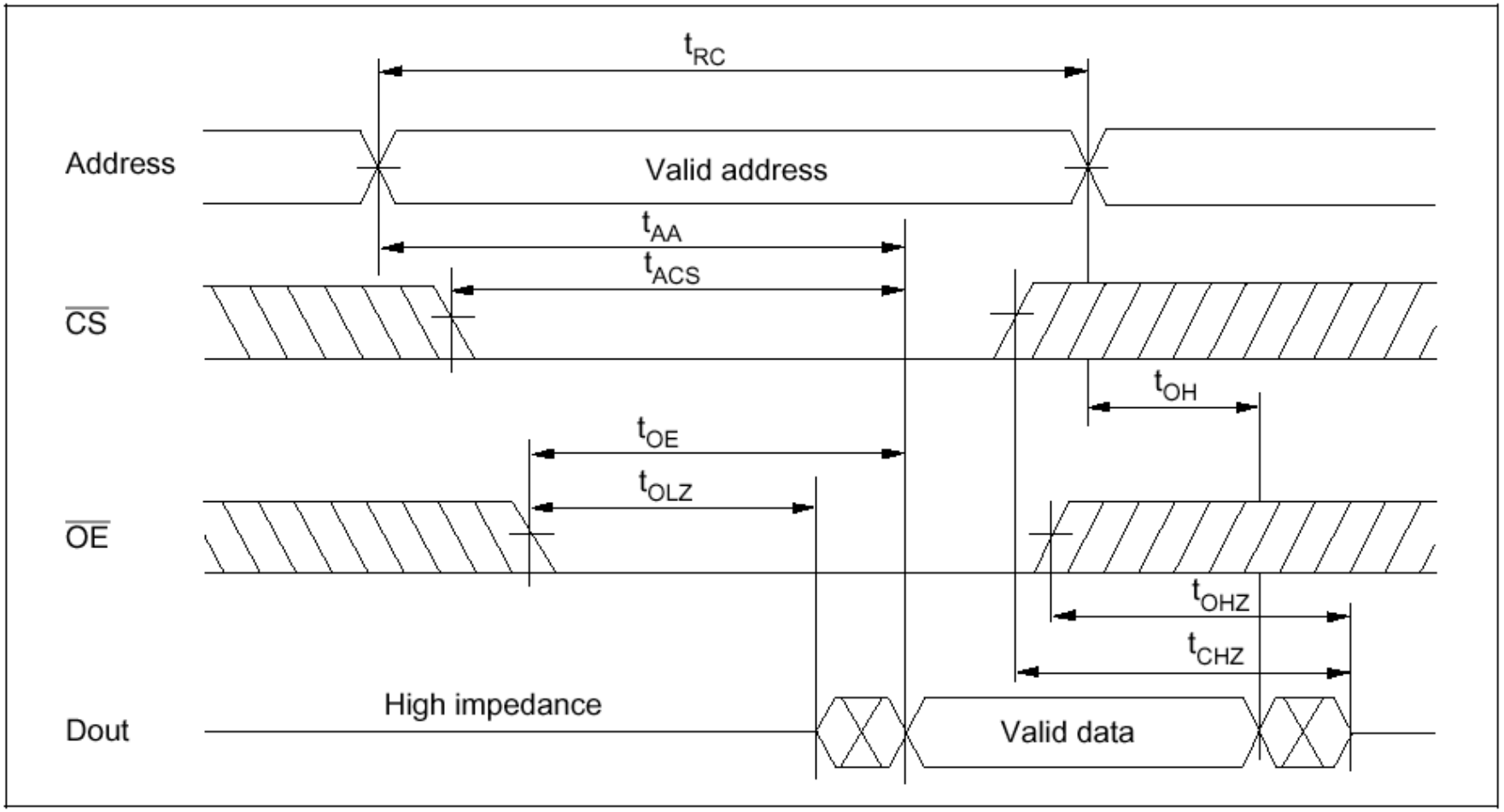
Ejemplo: Lectura de Memoria RAM con Intel 80C88

Esquema Interno - Memoria RAM HM62256 (32K x 8)



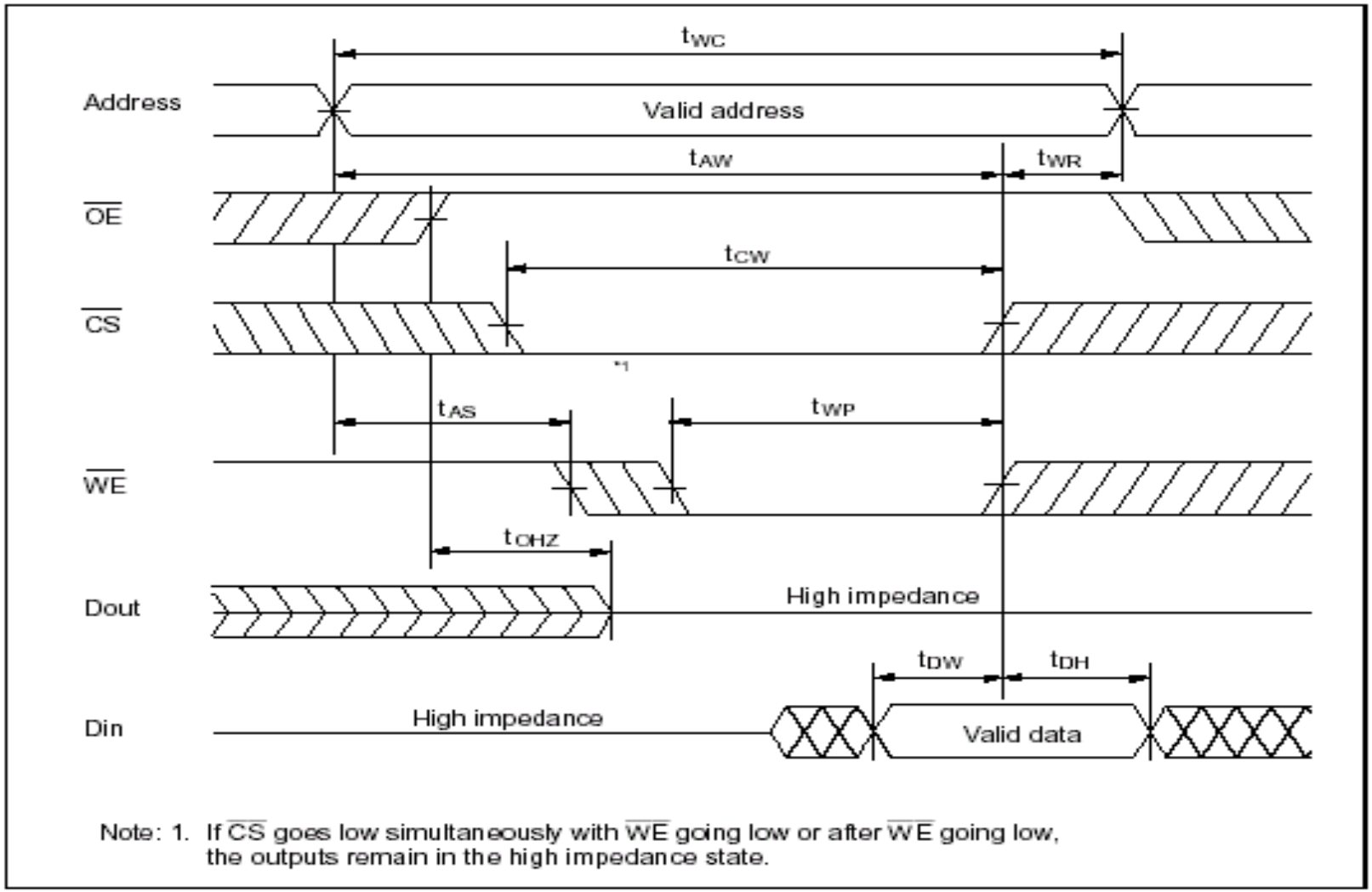
Ciclo de Lectura - Memoria RAM HM62256 (32K x 8)

Read Timing Waveform (1) ($\overline{WE}$ = V_{IH})



Ciclo de Escritura - Memoria RAM HM62256 (32K x 8)

Write Timing Waveform (1) ($\overline{OE}$ Clock)



Note: 1. If $\overline{CS}$ goes low simultaneously with $\overline{WE}$ going low or after $\overline{WE}$ going low, the outputs remain in the high impedance state.

Compatibilización en Lectura - uP -RAM (Con Latch Transparente)

Microprocesador

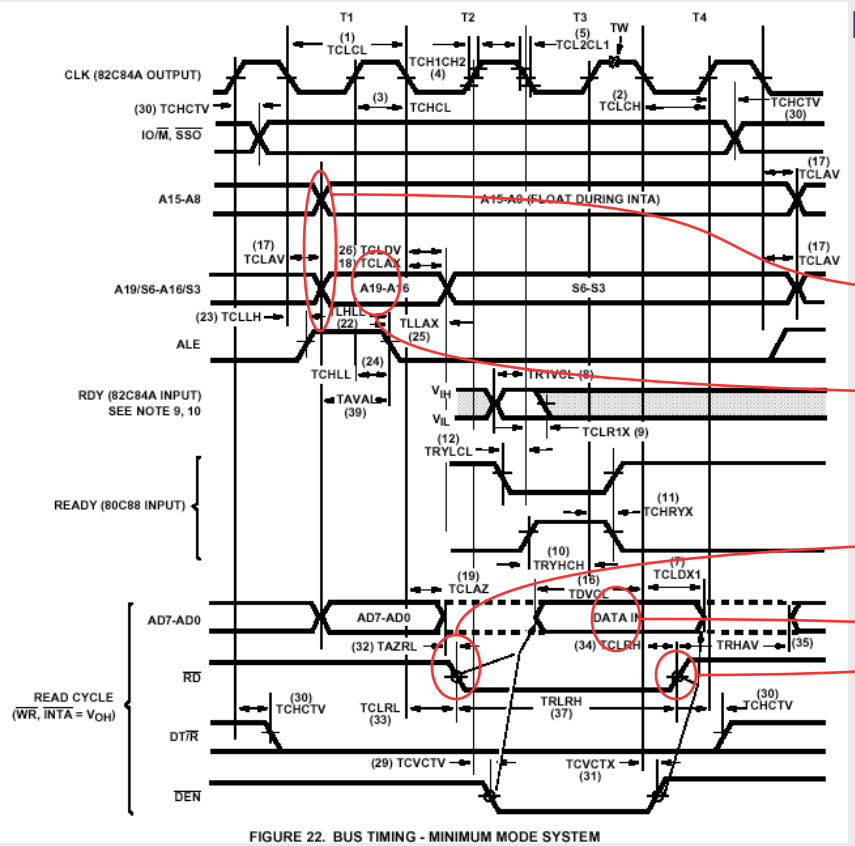
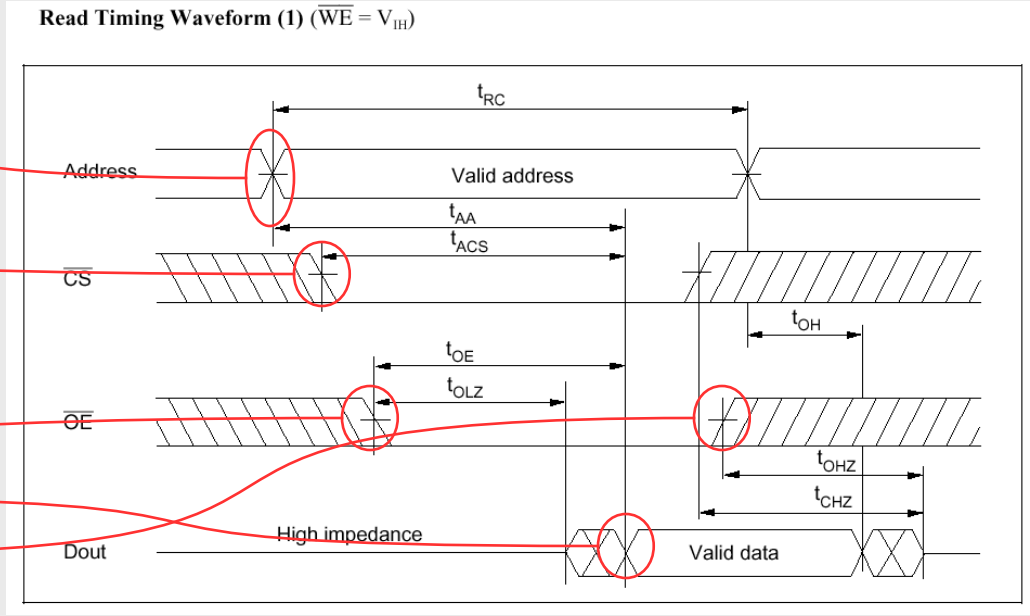


FIGURE 22. BUS TIMING - MINIMUM MODE SYSTEM

La compatibilidad depende del circuito intermedio !!

Memoria



- ✻ Direccionamiento uP
- ✻ Decodificación de direcciones
- ✻ Activación de /RD
- ✻ Lectura del dato
- ✻ Desactivación de /RD
- ✻ Direccionamiento uP

- ✻ Dirección estable para acceso
- ✻ Activación de Chip Select (Selección de celda)
- ✻ Activación de los buffers de salida
- ✻ Entrega del dato
- ✻ Desabilitación buffers de salida
- ✻ Deselección de dispositivo

Para Analizar y Resolver

Realizar el mismo análisis de Compatibilización para el Ciclo de Escritura (uP -RAM) (Utilizando Latch Transparente)