ASSEMBLER DIRECTIVES :

Assembler directives are the directions to the assembler which indicate how an operand or section of the program is to be processed. These are also called pseudo operations which are not executable by the microprocessor. The various directives are explained below.

1. ASSUME : The ASSUME directive is used to inform the assembler the name of the logical segment it should use for a specified segment.

Ex: ASSUME DS: DATA tells the assembler that for any program instruction which refers to the data segment ,it should use the logical segment called DATA.

2.DB -Define byte. It is used to declare a byte variable or set aside one or more storage locations of type byte in memory.

For example, CURRENT_VALUE DB 36H tells the assembler to reserve 1 byte of memory for a variable named CURRENT_ VALUE and to put the value 36 H in that memory location when the program is loaded into RAM .

3. DW -Define word. It tells the assembler to define a variable of type word or to reserve storage locations of type word in memory.

4. DD(define double word) :This directive is used to declare a variable of type double word or restore memory locations which can be accessed as type double word.

5.DQ (define quadword) : This directive is used to tell the assembler to declare a variable 4 words in length or to reserve 4 words of storage in memory .

6.DT (define ten bytes): It is used to inform the assembler to define a variable which is **10** bytes in length or to reserve 10 bytes of storage in memory.

7. EQU –Equate It is used to give a name to some value or symbol**.** Every time the assembler finds the given name in the program, it will replace the name with the value or symbol we have equated with that name

8.ORG -Originate : The ORG statement changes the starting offset address of the data.

It allows to set the location counter to a desired value at any point in the program.For example the statement ORG 3000H tells the assembler to set the location counter to 3000H.

9 .PROC- Procedure: It is used to identify the start of a procedure or subroutine.

10. END- End program .This directive indicates the assembler that this is the end of the program module.The assembler ignores any statements after an END directive.

11. **ENDP**- End procedure: It indicates the end of the procedure (subroutine) to the assembler.

12.ENDS-End Segment: This directive is used with the name of the segment to indicate the end of that logical segment.

Ex: CODE SEGMENT : Start of logical segment containing code

CODE ENDS : End of the segment named CODE.

Basic Peripherals and Their Interfacing with 8086

Interfacing with RAM And ROM

The **figure 2.1** shows a general block diagram of an 8086 memory array. In this, the 16-bit word memory is partitioned into high and low 8-bit banks on the upper halves of the data bus selected by BHE, and AO.

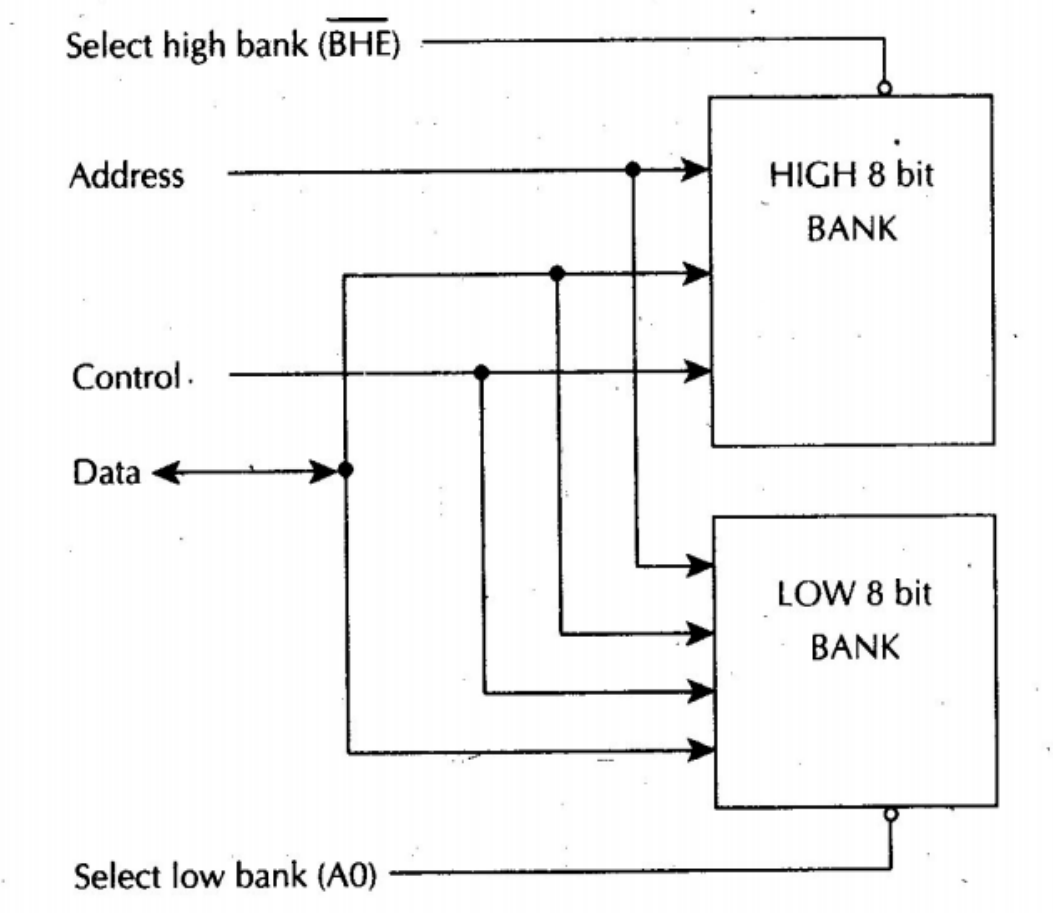


FIGURE 2.1– 8086 MEMORY ARRAY

a) ROM and EPROM

ROMS and EPROMs are the simplest memory chips to interface to the 8086. Since ROMs and EPROMs are read-only devices, A0 and BHE are not required to be part of the chip enable/select decoding. The 8086 address lines must be connected to the ROM/EPROM chip chips starting with A1 and higher to all the address lines of the ROM/EPROM chips. The 8086 unused address lines can be used as chip enable/select decoding. To interface the ROMs/RAMs directly to the 8086-multiplexed bus, they must have output enable signals. The **figure 3.5.2** shows the 8086 interfaced to two 2716s.

Byte accesses are obtained by reading the full 16-bit word onto the bus with the 8086 discarding the unwanted byte and accepting the desired byte.

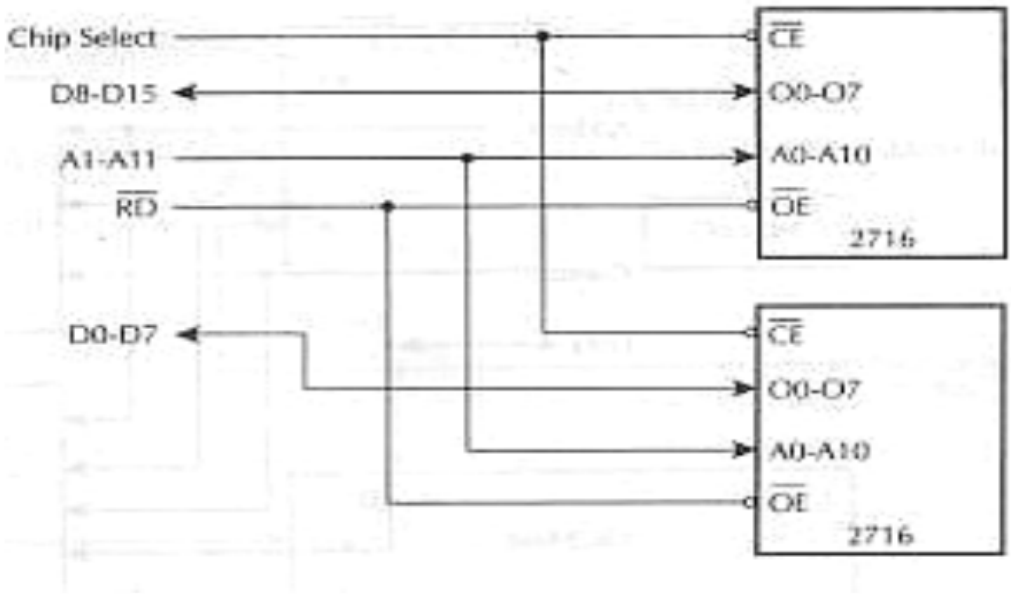


FIGURE 3.5.2

b) Static RAMS

Since static RAMs are read/write memories, both A0 and BHE must be included in the chip select/enable decoding of the devices and write timing must be considered in the compatibility analysis.

For each static RAM, the memory data lines must be connected to either the upper half AD15-AD0 or lower half AD7-AD0 of the 8086 data lines.

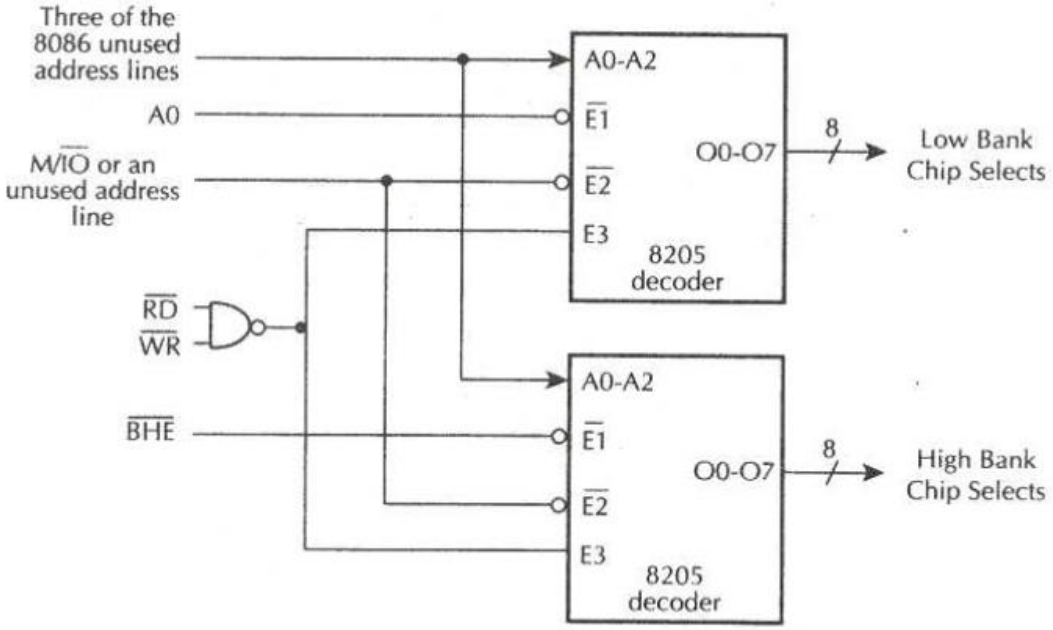
For static RAMs without output enable pins, read and write lines must be used as enables for chip select generation to avoid bus contention. If read and write lines are not used to activate the chip selects, static RAMs with common input/output data pins such as 2114 will face extreme bus contentions between chip selects and write active. The 8086 A0 and BHE pins must be used to enable the chip the chip selects. Note that Intel 8205 has three enables E1, E2, and E3, three inputs A0 and A2, and eight outputs O0-O7.

For devices with output enables such as 2142, one way to generate chip selects for the static RAMs is by gating the 8086 $\overline{WR}$ signal with BHE and A0 to provide upper and lower bank write strobes. A possible configuration is shown in the **figure 3.5.4**. Since the Intel 2142 is a 1024 * 4 bit static RAM, two chips for each bank with a total of 4 chips for 2K * 8 static RAM is required. Note that DATA is read from the 2142 when the output disable OD is low, WE is HIGH, and DATA is written into 2142. If multiple chip selects

are available with the static RAM, BHE and A0 may be used directly as the chip selects. A possible configuration for 2K * 8 array is shown in the **figure 3.5.5**.

c) Dynamic RAM

Dynamic RAM store information as charges in capacitors. Since capacitors can hold charges for a few milliseconds, refresh circuitry is necessary in dynamic RAMs for retaining these charges. Therefore, dynamic RAMs are complex devices to design a system. To relieve the designer of most of these complicated interfacing tasks, Intel provides the 8202 dynamic RAM controller as part of the 8086 families of peripheral devices. The 8202 can be interfaced with the 8086 to build a dynamic memory system.



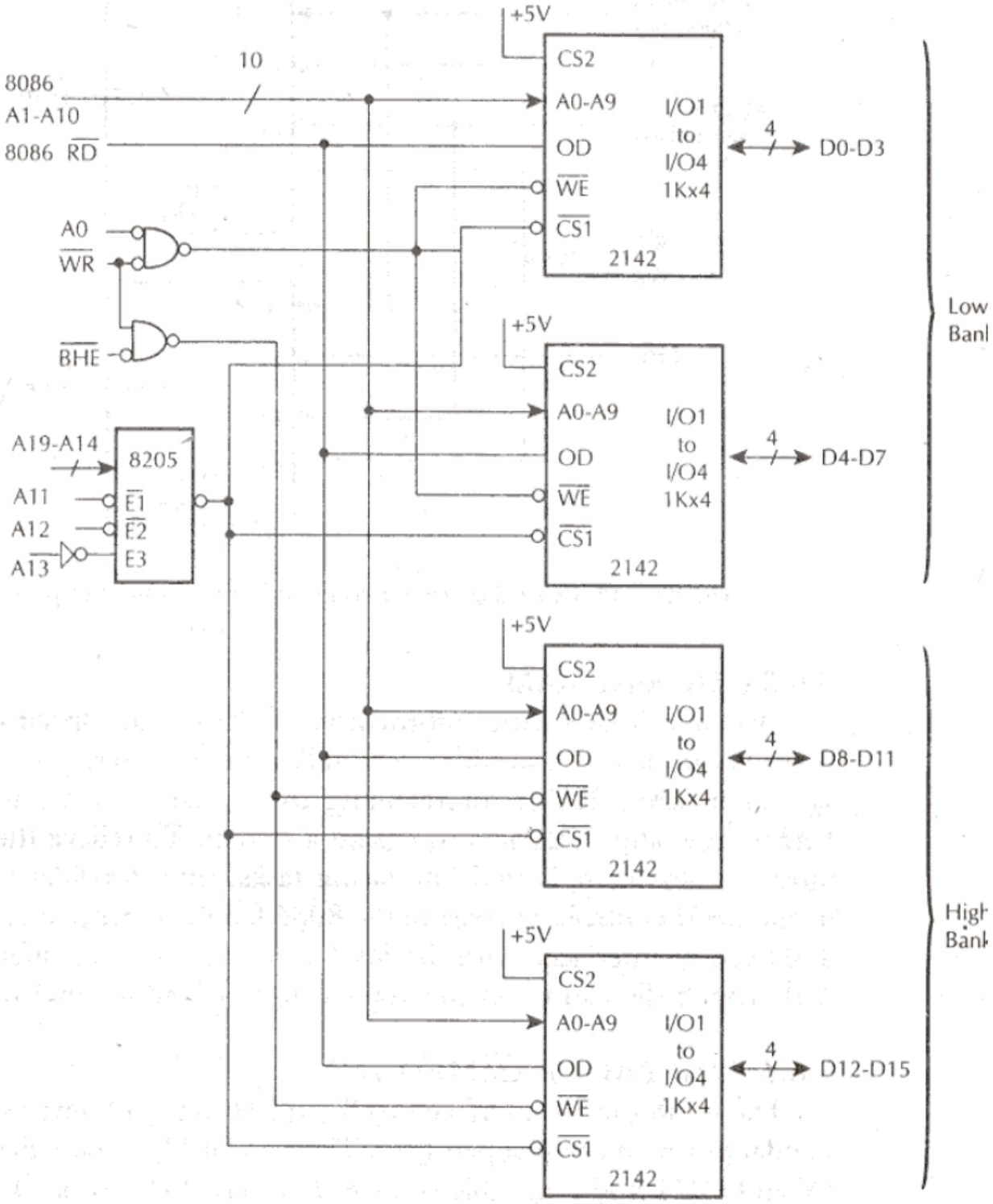
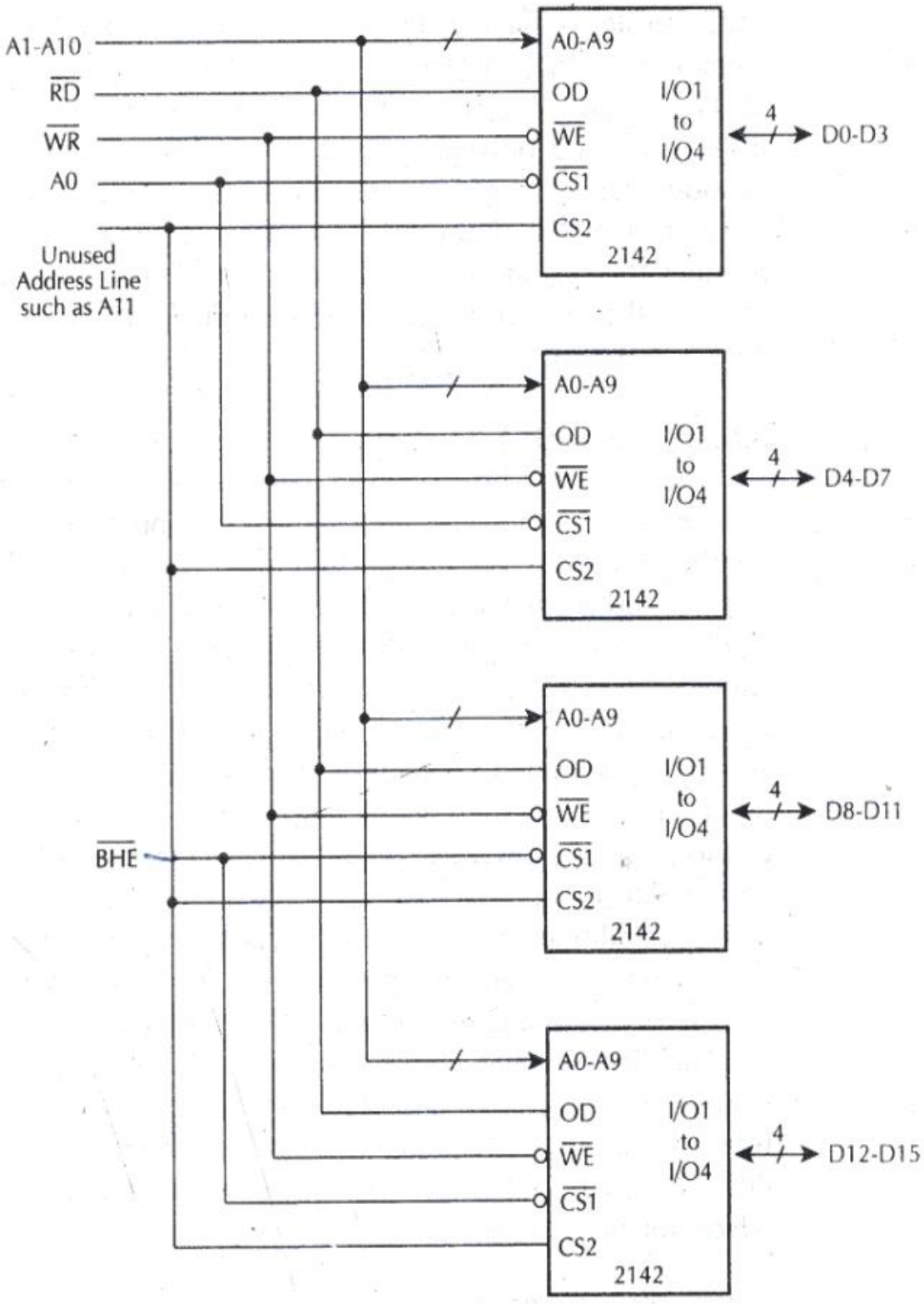


FIGURE 3.5.5 – 2K * 8 STATIC ARRAY WITH A0 and BHE AS DIRECT CHIP SELECT INPUTS


A1-A10
RD
WR
A0
Unused Address Line such as A11
BHE
A0-A9
OD
WE
CS1
CS2
2142
I/O1 to I/O4
4
D0-D3
D4-D7
D8-D11
D12-D15