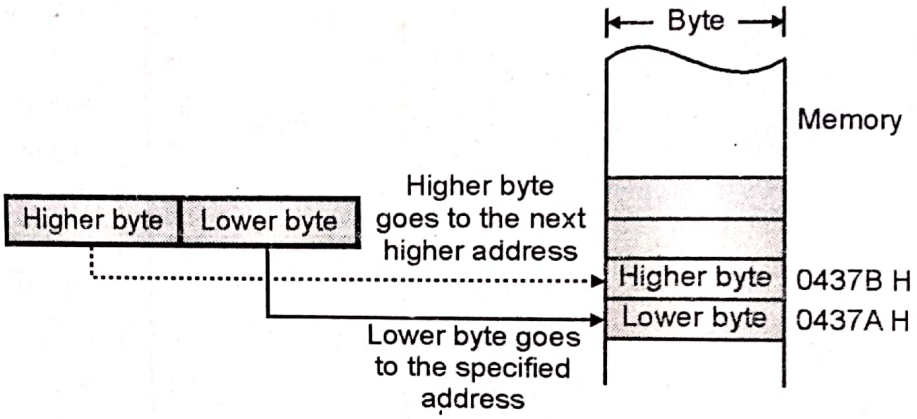
2.8 Even and Odd Memory Banks for 8086

PU - May 2005, May 2006, May 2007

- We know that the 8086 has a 20-bit address bus, so it can address 2^{20} or 1,048, 576 addresses.
- At each address we can store an 8-bit data (1 byte). Hence the total memory capacity of 8086 is 1 M byte.
- However the data bus of 8086 is 16 bit and the processor is capable of processing 16 bit data i.e. words.
- The question is how to write a word (16 bit data) into a memory which is segmented to store the data in the byte form.



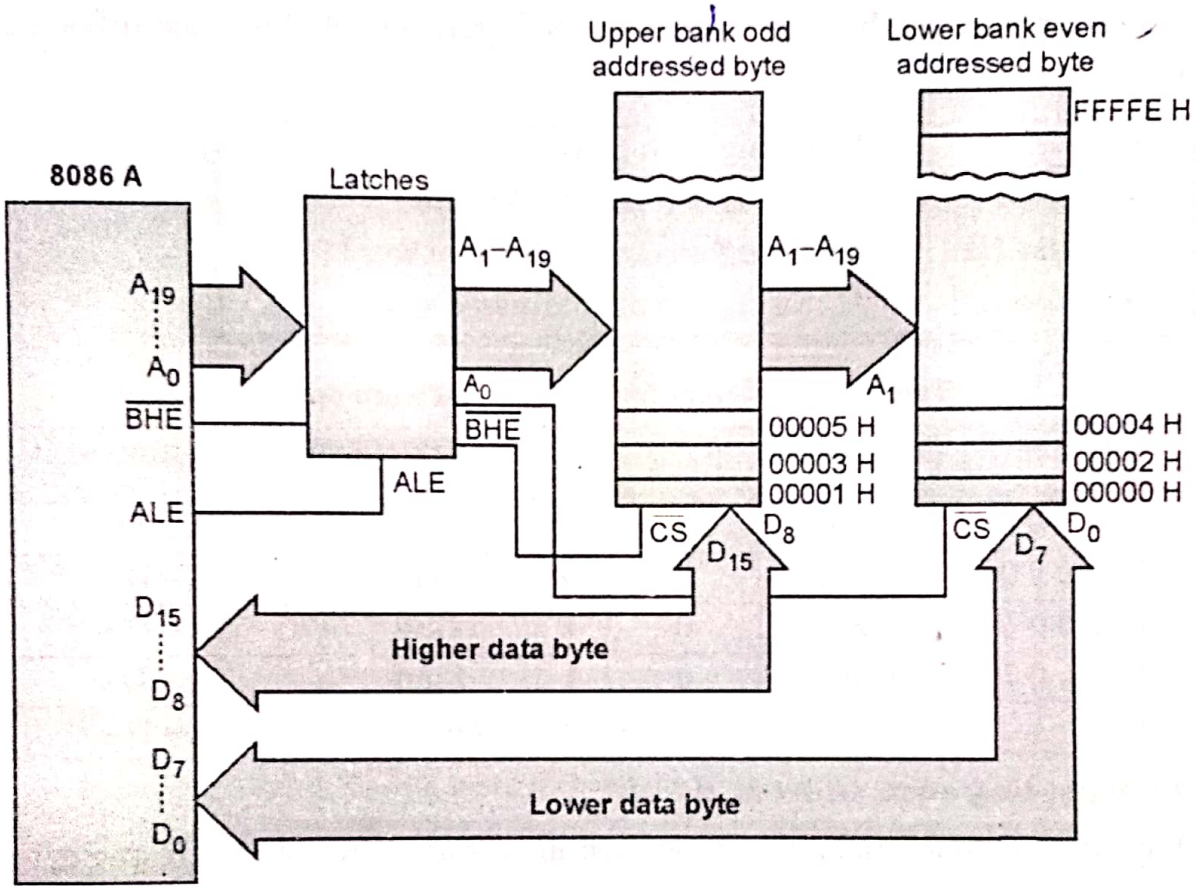
m(15.9) **Fig. 2.8.1 : How is a word stored in the 8086 memory ?**

- The answer is that a word is written into two consecutive memory addresses.
- That means the lower byte is written into the specified memory address say 0437AH and the higher byte is written into the next higher address as illustrated in Fig. 2.8.1.

- In order to make it possible to read or write a word with one machine cycle, the memory of 8086 is divided into two "banks" of upto 524, 288 bytes i.e. 500 K bytes each, as shown in Fig. 2.8.2.
- The two memory banks are known as :

  - The even addressed memory bank
  - The odd addressed memory bank

- The even addressed memory bank contains all the bytes which have even addresses such as 00000, 00002, 00004, ... etc.



m(15.10)**Fig. 2.8.2 : 8086 memory banks block diagram**

- The odd addressed memory bank contains all the bytes which have odd addresses such as 00001, 00003, 00005 etc.
- The even addressed bank is also called as lower bank and the data lines of this bank are connected to the lower eight data lines i.e. D_0 to D_7 of 8086 as shown in Fig. 2.7.2.
- The odd addressed bank is also called as the upper bank and the data lines of this bank are connected to the upper eight data lines i.e. D_8 to D_{15} of 8086 as shown in Fig. 2.7.2.
- The address line A_0 is used to enable the lower memory bank because A_0 is connected to the $\overline{CS}$ input of the lower bank.

- Address lines A_1 through A_{19} are used for selecting the desired memory device in the bank and to address the desired byte in that device.
- The bus high enable ($\overline{BHE}$) is used for enabling the memory in the upper bank.
- $\overline{BHE}$ is multiplexed out on a single line from 8086 at the same time when an address is sent out.
- The ALE signal is used to strobe in the address into the external latch. The same latch stores the $\overline{BHE}$ signal as well and holds it stable for the remaining machine cycle.
- Table 2.8.1 gives the logic levels on the $\overline{BHE}$ and A_0 lines for different types of memory accesses.

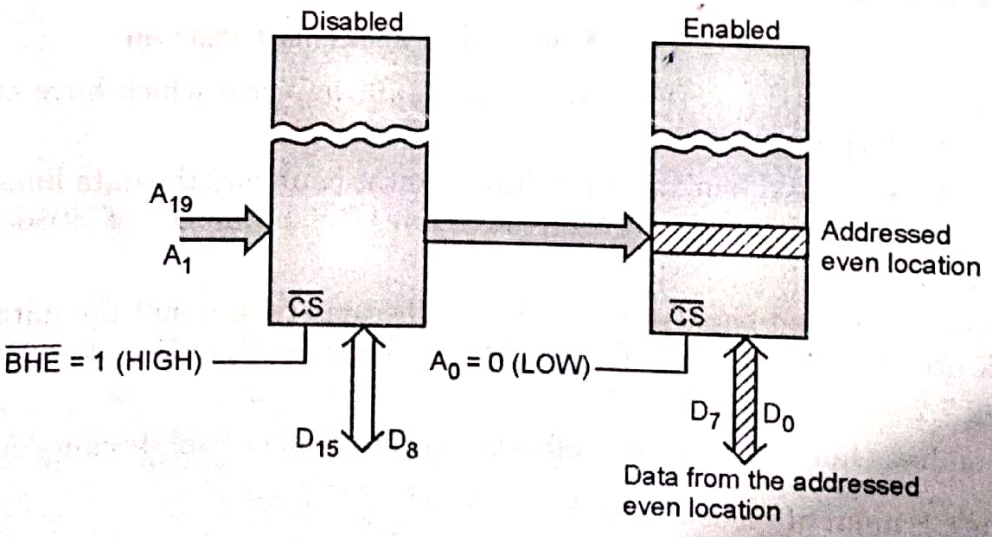
- Case I : Accessing even addressed byte
- Case II : Accessing odd addressed byte
- Case III : Accessing an even addressed word
- Case IV : Accessing an odd addressed word

Table 2.8.1 : Signals for the byte and word operations

Address	Data type	$\overline{BHE}$	A_0	Bus cycles	Data lines used
0 0 0 0	Byte	1	0	One	$D_0 - D_7$
0 0 0 0	Word	0	0	One	$D_0 - D_{15}$
0 0 0 1	Byte	0	1	One	$D_8 - D_{15}$
0 0 0 1	Word	0	1	First	$D_0 - D_7$
		1	0	Second	$D_8 - D_{15}$

Case I : Accessing even addressed byte :

- Fig. 2.8.3(a) shows the process involved in accessing an even addressed byte.

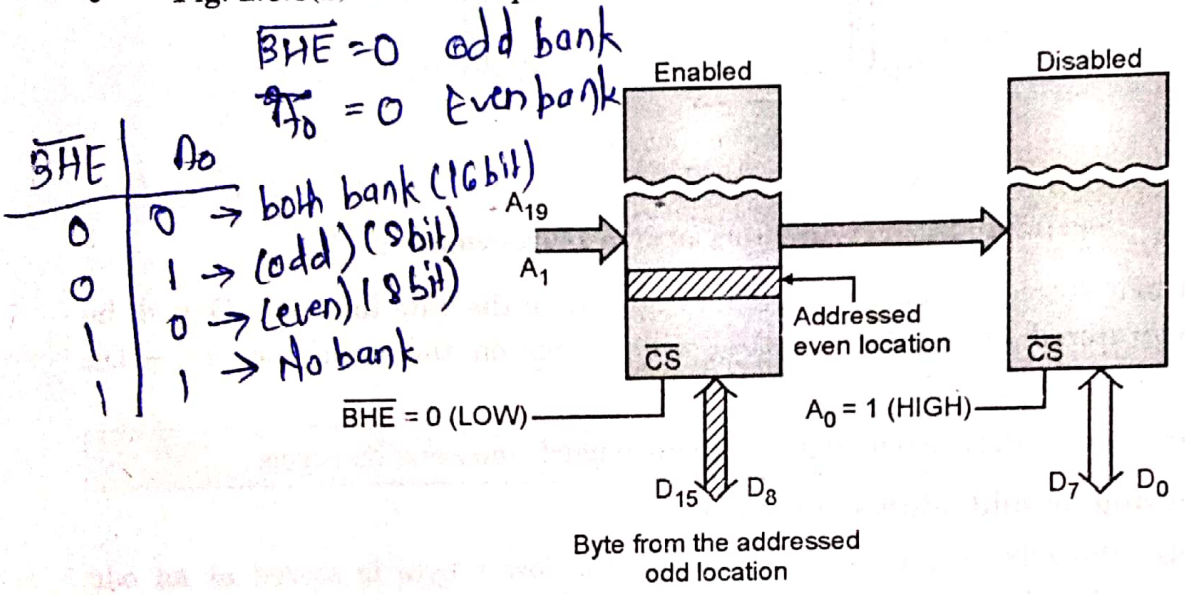


m(15.11)**Fig. 2.8.3(a) : Accessing an even addressed byte**

- The 8086 forces A_0 line LOW and $\overline{BHE}$ HIGH.
- This will enable the lower memory bank and disable the higher memory bank.
- The 8086 outputs the address of the desired even memory location on the address lines A_1 to A_{19}.
- The data stored (byte) at the addressed memory location appears on the data lines $D_0 - D_7$ as shown by the hatched lines in Fig. 2.8.3(a).

Case II : Accessing the odd addressed byte :

- Fig. 2.8.3(b) shows the process involved in accessing an odd addressed byte.

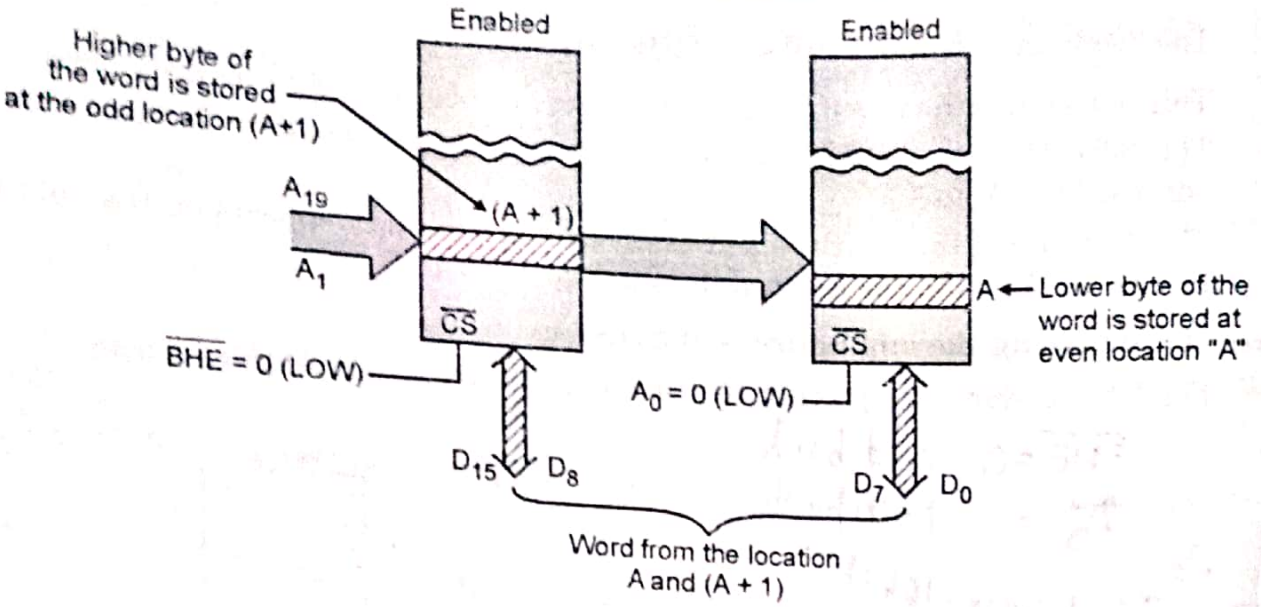


m(15.11)**Fig. 2.8.3(b) : Accessing an odd byte**

- The 8086 forces $A_0 = 1$ (HIGH) and $\overline{BHE} = 0$ (LOW).
- This will disable the lower (even) memory bank and enable the upper memory bank.
- The 8086 will output the address of desired odd memory location on A_1 to A_{19} address lines.
- The data byte on the addressed odd location appears on the data lines $D_8 - D_{15}$, as shown in Fig. 2.8.3(b).

Case III : Accessing an even addressed word :

- Here the 8086 wants to read a 16 bit word.
- As stated earlier, the lower byte of this word is stored first and the higher byte is stored at the next higher.
- This is illustrated in Fig. 2.8.3(c). The lower byte of the word has been stored at the even memory location A and the higher byte of the same word has been stored at the next location (odd memory location A + 1).
- Both the memory banks are enabled because 8086 will force A_0 as well as $\overline{BHE}$ low.
- The address on the lines $A_{19} - A_1$ will point towards the locations A and A + 1 simultaneously.

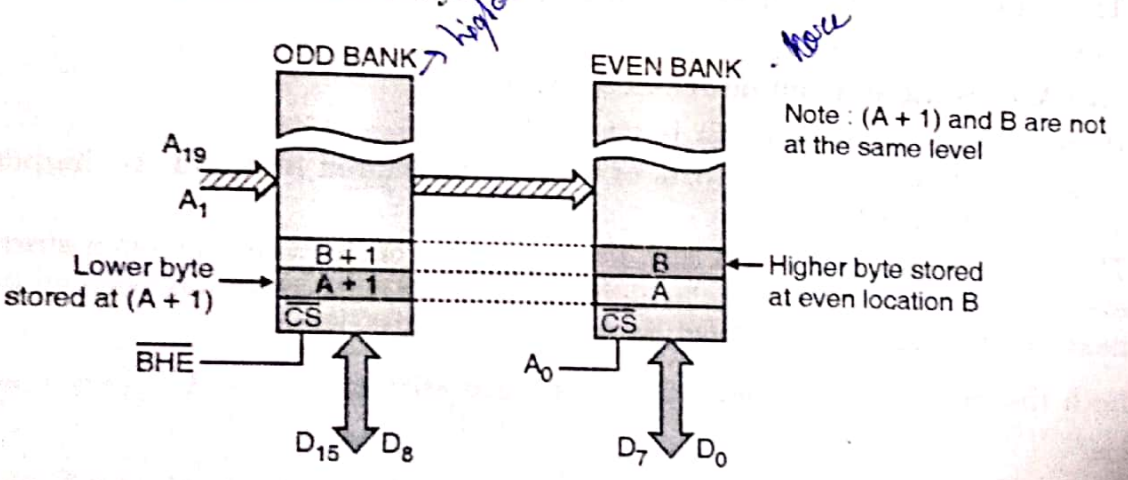


m(15.11)**Fig. 2.8.3(c) : Accessing an even addressed word**

- The lower byte stored at location A will now appear on the data lines $D_0 - D_7$ and the higher byte stored at location (A + 1) will appear on the data lines $D_8 - D_{15}$ simultaneously.
- Thus to read a word from even address the 8086 needs only one bus cycle.

Case IV : Accessing an odd addressed word :

- In this case the 8086 wants to read a word, the lower byte is stored at an odd address. (say A + 1) and the upper byte of the word is at even address (i.e. at B immediately next to A + 1).
- Now refer Fig. 2.8.3(d) and notice that the locations (A + 1) and B are not at the same level.
- Therefore the 8086 cannot use the same address for both these locations, (A + 1) and B.
- Therefore it is not possible for 8086 to output a single address and read all the 16 bits of the word simultaneously.

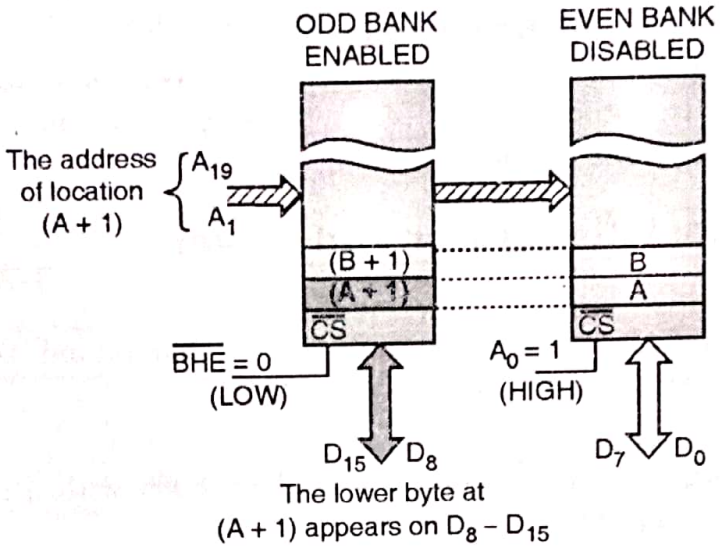


m(15.11)**Fig. 2.8.3(d) : Accessing an odd addressed word**

- Hence the 8086 needs two bus cycles, to read the 16 bit word. One bus cycle is required to read the contents of location (A + 1) and other to read the contents of location B. The sequence of operations in these two bus cycles is as explained below.

First bus cycle :

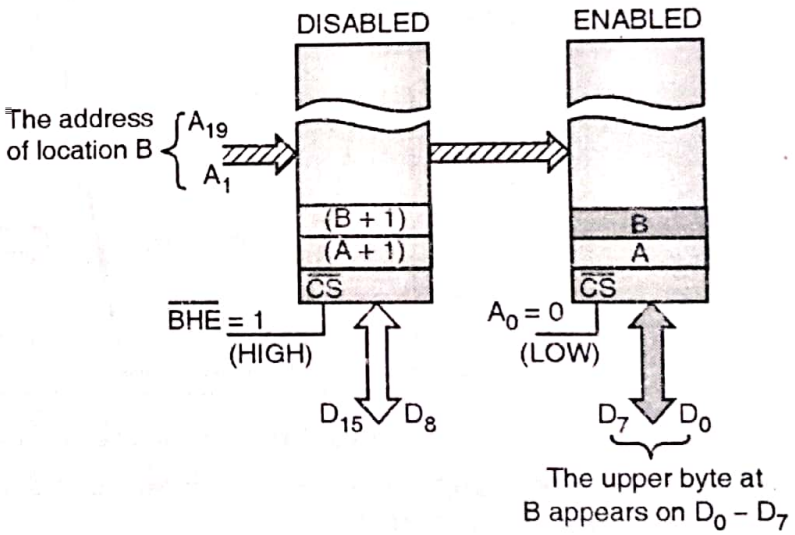
- In the first bus cycle, the 8086 forces $A_0 = 1$ (HIGH) and $\overline{BHE} = 0$ (LOW) so as to enable the odd bank.
- It then outputs the address on $A_1 - A_{19}$ lines to point at the location (A + 1).
- The contents of location (A + 1) will appear on the data lines D_8 to D_{15}, as shown in Fig. 2.8.3(e).



m(15.11)**Fig. 2.8.3(e) : First bus cycle**

Second bus cycle :

- In the second bus cycle the 8086 forces $A_0 = 0$ (LOW) and $\overline{BHE} = 1$ (HIGH) so as to enable the lower (even) bank.
- It then outputs the new address on the $A_1 - A_{19}$ lines to point at location B.
- The contents of location B will appear on $D_0 - D_7$ as shown in Fig. 2.8.3(f).



m(15.11)**Fig. 2.8.3(f) : Second bus cycle**

Conclusion :

Accessing a word which starts at the odd address takes two machine cycles i.e. longer time. Therefore generally it is preferred that the word operand should have lower byte in the even bank so that the microprocessor can read 16 bits at a time in single machine cycle.

- This can be achieved by using the assembler directive EVEN.