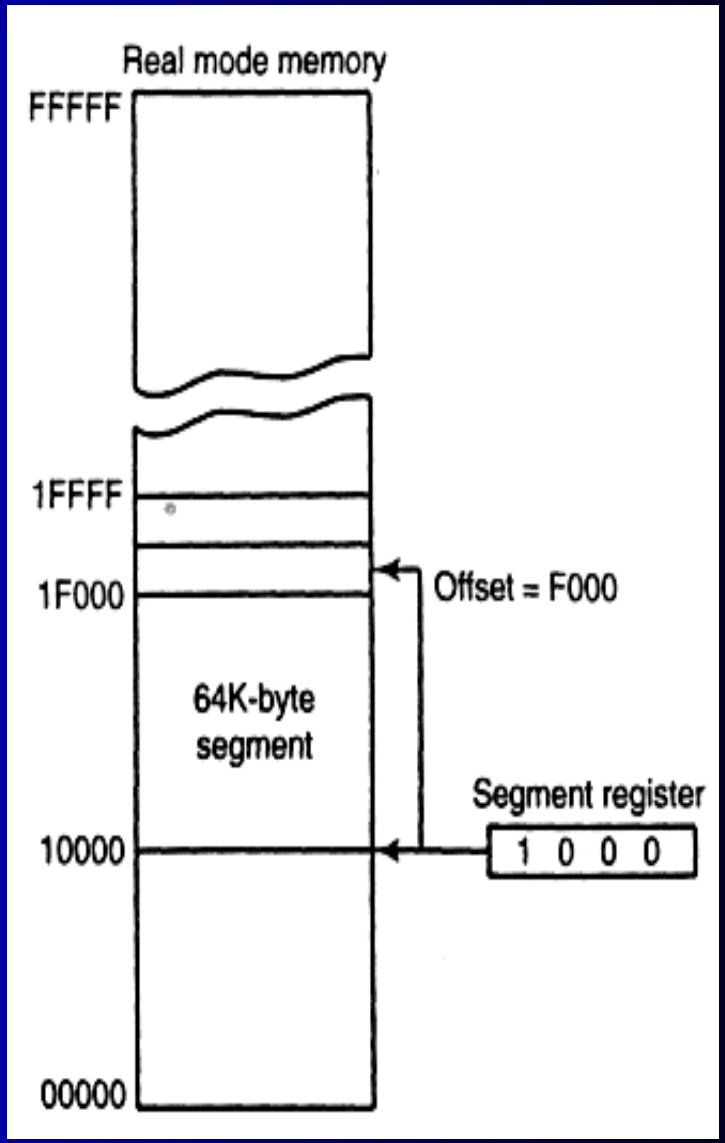
Segments and Offsets

- A combination of a segment address and an offset address access a memory location in the real mode.
- All real mode memory addresses must consist of a segment address plus an offset address.
- The **segment address,** located within one of the segment registers, defines the beginning address of any 64K-byte memory segment.
- The **offset address** selects any location within the 64K byte memory segment.
- Segments in the real mode always have a length of 64K bytes.

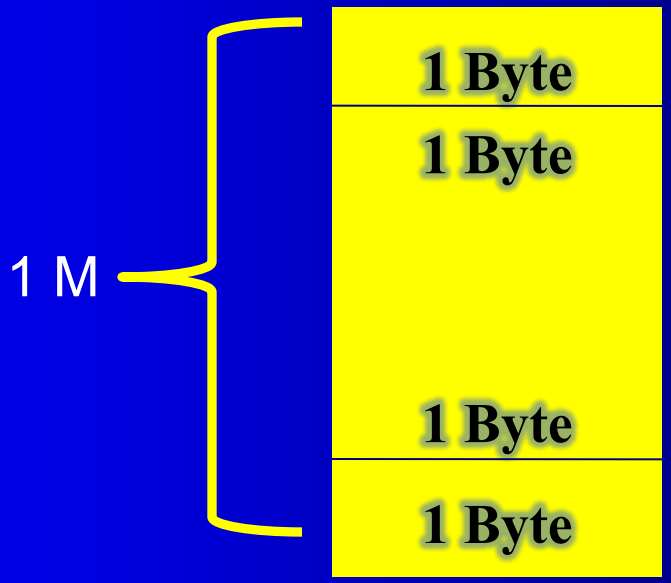
Example: The real mode memory-addressing, using a segment address plus an offset.

- The segment register = 1000H,
- The starting address= 10000H.
- In the real mode, each segment register is internally appended with a 0H on its rightmost end.
- a memory segment that begins at location 10000H and ends at location 1FFFFH— 64K bytes in length.
- It also shows how an offset address, sometimes called a **displacement**, and of F000H selects location 1F000H in the memory system.

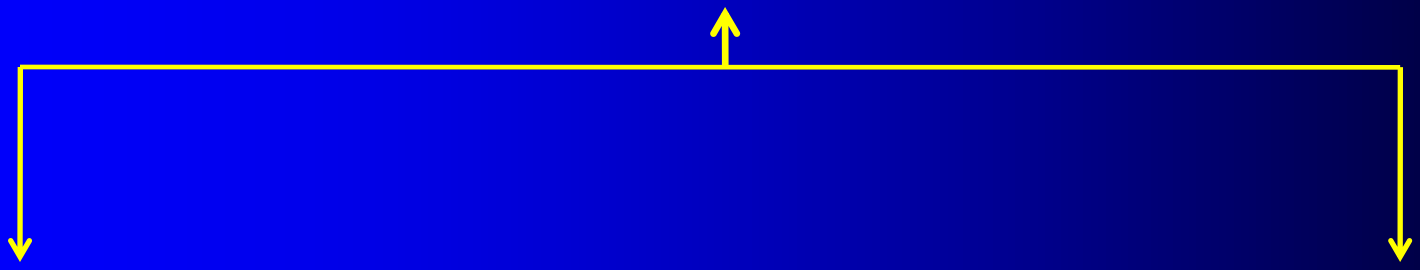


Memory System & Segmentations

- 8088/8086 has 1 M Byte Memory
- 1 M Byte=2^{20} Byte
- 20 address lines to Reach
- 2^{20} location , Each location is a Byte address location range from



Memory is Viewed by different ways



1- Logical Memory

“as it appears in

a program for

programmer ”

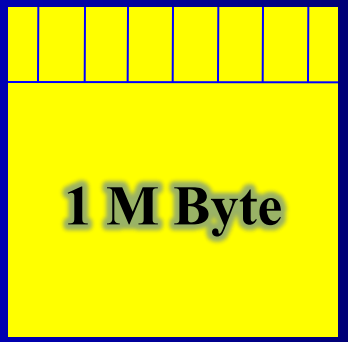
2- Physical Memory

“as it is Built in reality”

“in hardware Design”

Logical Memory

- Logical Memory : is what appears in program not as H.W , so L.M for 8086 is the same as 8088



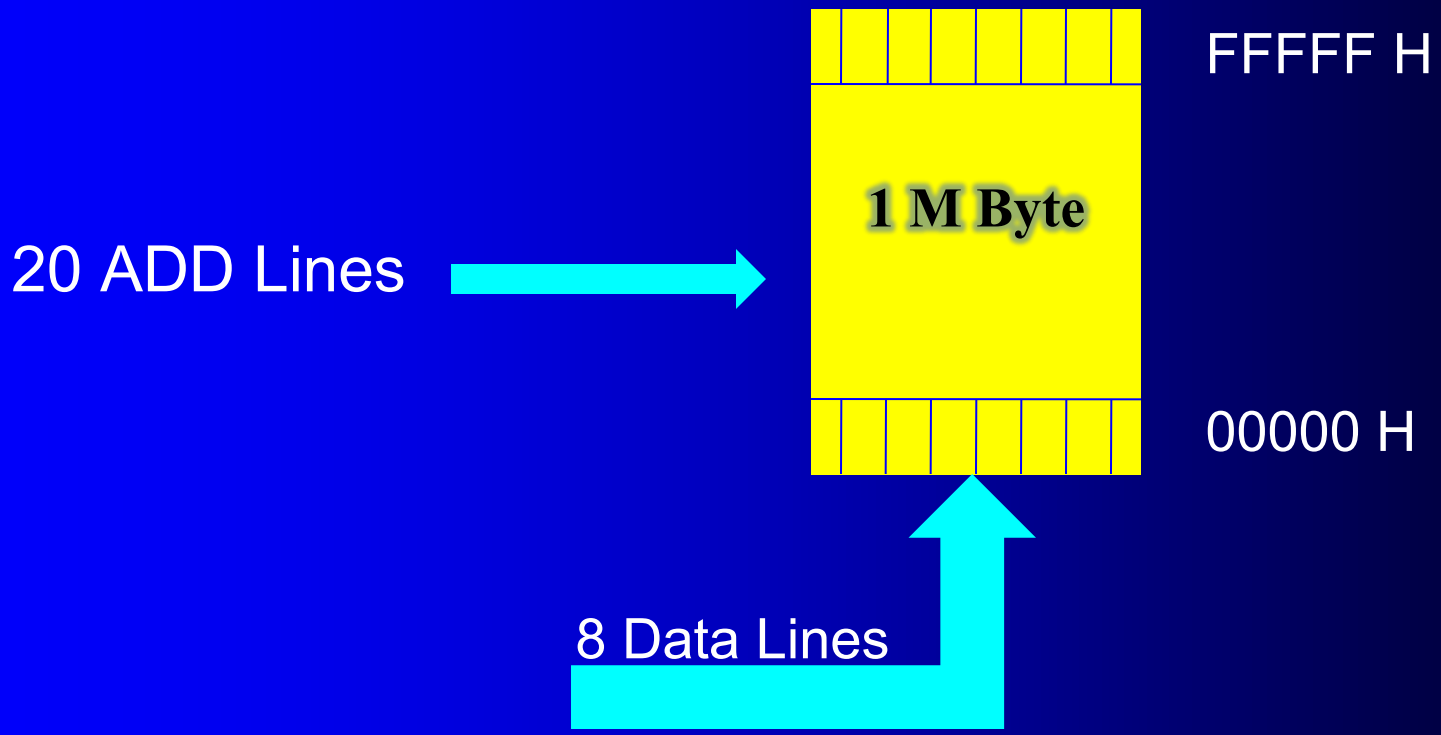


- Some Address Spaces are reseved for Special usages (interrupts , stack,…..)

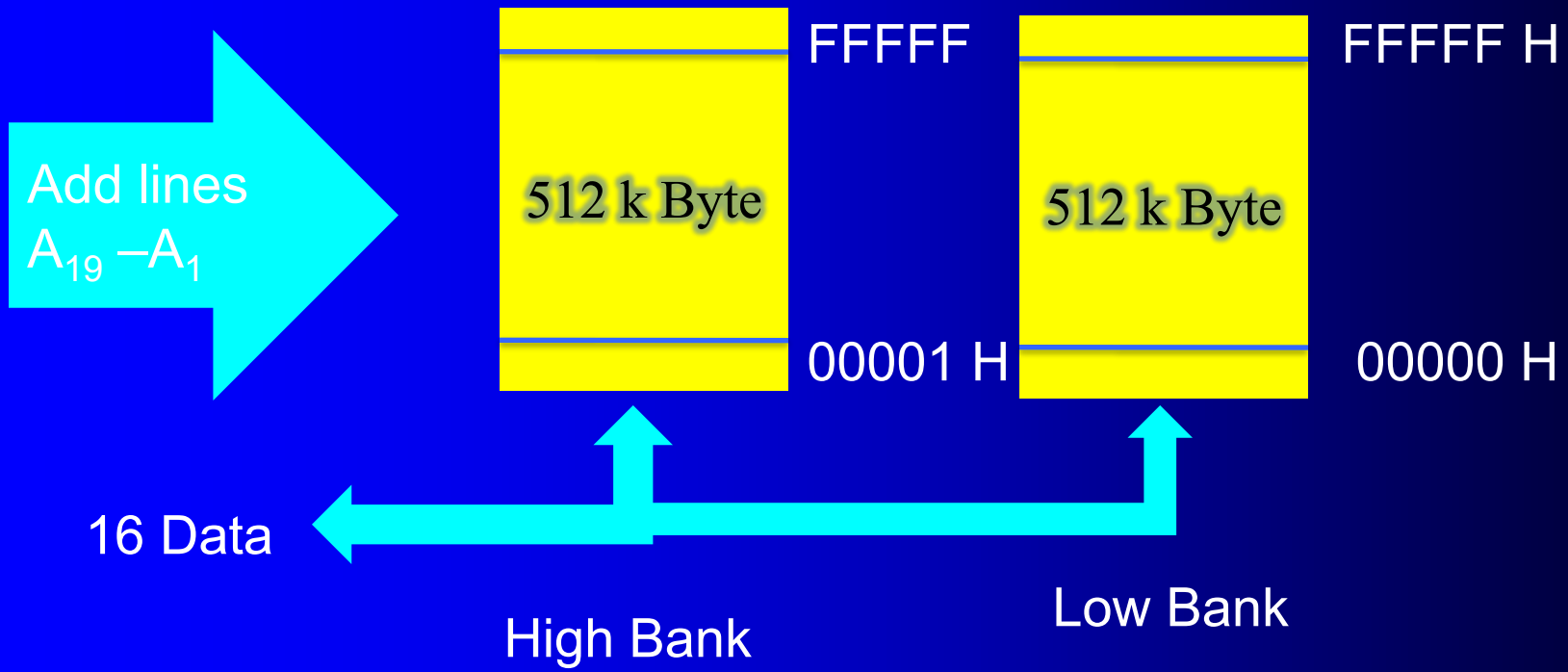
Physical Memory

- Memory as it is found in hardware
- For 8088 : Data Bus → 8 bits
  add Bus → 20 lines

Memory can be made by only (1) Bank



- For 8086 :-Data Bus → 16 bits add Bus → 20 lines
- It Shoult be Bytes compatible memory can be made by (2) Banks.



Different Between 8086 and 8088

- 8088 is Byte compatible , it can deal with Byte (8 bits Data).
- 8086 is more efficient than 8088 in Dealing with 16 bits
- because 8086 need 1 clock cycle to read – write 16 bits.
- While 8088 need 2 clock cycle to read – write 16 bits.

How to get a location Address ?

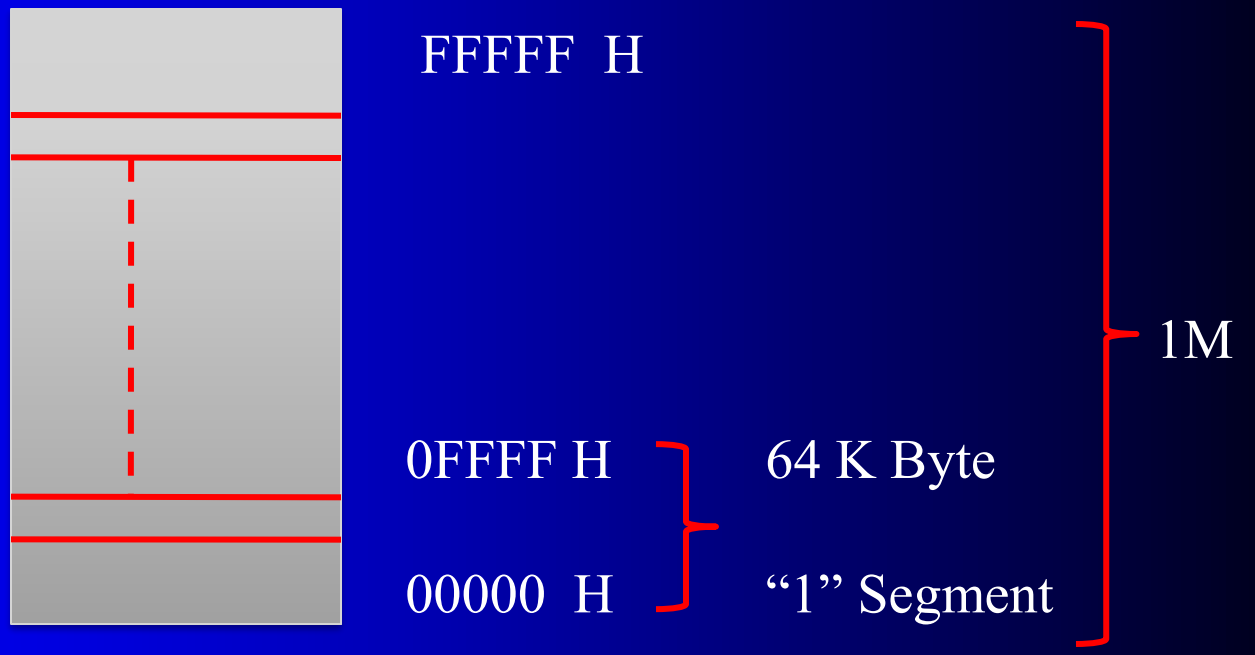
- By using segment Register with Pointer/index reg we can find any location in memory.
- CS ⟶ Code Segmant Start Location.
- DS ⟶ Data Segmant Start Location.
- ES ⟶ Extra Segmant Start Location.
- SS ⟶ Stack Segmant Start Location.
- Reg are 16 bits only , Then any Reg can be used to Address 2^{16} location only.
- 1 M Byte can be Divided into 16 Segmant part.
- each Segmant =64 k Byte.



How to reach all locations in memory ?

We can save the start add. In reg and more from it till 64 Kbyte.

EX: If we want to go From 00000H till 0FFFFH



SOL:

We save Base “start” add in segment Reg and use another reg. as an offset to reach 64 Kbyte from this segment start.

- Only [4] segments are active at the same time.

How to reach all locations in memory ?

All Segments starting address will have add. Locations ends by o^H

EX: 050F0, FEC00, 9FAA0

to get a specific Location Add we use:

- **CS with IP**
- **DS with BX, SI or DI**
- **SS with BP, SP**
- **ES with DI**

Segment Reg holds the Start of the segment Location of pointer Reg. holds the offset index.

Generating a Memory address

[1] **Logical address :**

Segment : offset EX: CS: 0100^H

[2] **Physical Address:**

- Physical address should be 20 bits.
- Physical address = Segment*10^H + offset.
- Starting add of segment = segment Reg * 10^H .
- Ending add of segment = Seg. Reg * 10^H + FFFF H

Ex.1 Segment Reg = 1000^H

offset Reg = 1234^H

PA = 1000^H * 10^H + 1234^H = 11234^H

LA = 1000^H : 1234^H

- Start add. For Seg = 10000^H .

  End add. For this Seg = 10000^H + $FFFF^H$ = $1FFFF^H$.

Ex.2

CS=1000^H , DS=3000^H , SS=$A000^H$, ES=8000^H , Bx=0200^H

what is the PA pointed By BX ?

Sol

We Know BX work with DS

- PA = DS × 10^H + BX = 30000^H + 0200^H = 30200^H

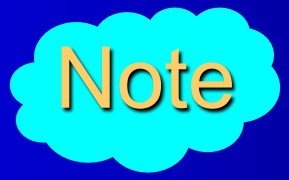
Ex.3

Find Address Range of Extra Seg If ES=$F000^H$

Sol

Start Add = Es × 10^H = $F0000^H$

End Add = Es × 10^H + FFFF = $FFFFF^H$



- This way of getting start Address of End add is **Ok** with code, Data and Extra Seg. note Stack Segment
- Stack Segment is LIFO memory it is upside down "Later when we talk about it"

Memory Segmentation & Data Formats

- **According to Value of Cs, Ds, Es, Ss We can only refer to 4 active Segments.**

  Recall: Cs ⟹ Define the start of code segment in memory

  Ds ⟹ Define start of Data seg im memory.

  Es ⟹ Start of Extra seg.

  Ss ⟹ Start of Stack seg.

- **To Reach only M.L we need Address of (20 bits).**

  .. Registers inside MP are 16 bits only.
  so we used Seg. Reg of offset Registers (index of pointer Reg)

  so to get any 20 bit Add. To reach any M.L

  Physical Add = Seg. Reg $*$ 10^H + offset

Memory Segmentation & Data Formats

EX.4

to get the First M.L of this Code Segment

start = Cs x 10^H = 10000



How to reach any Location inside this Seg?

- Note that this is a block of code

  To get any Location in this block of locations we use offset relative to the start of the block.

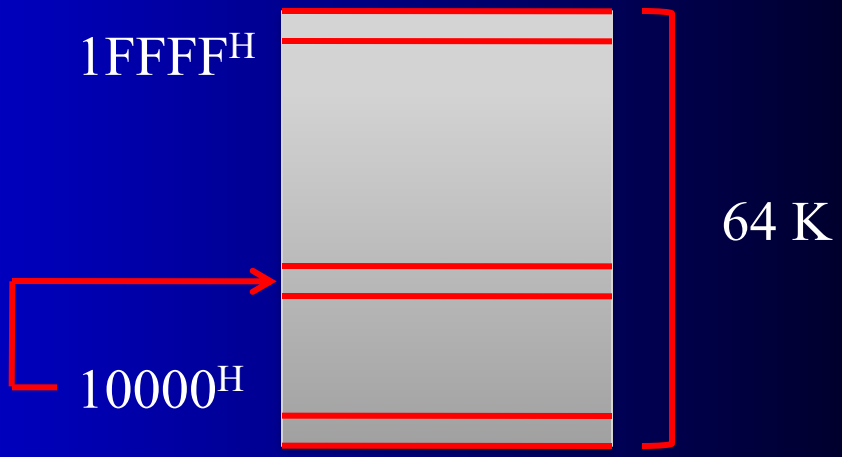
EX.5

It want to reach 10200^H M.L

Start = 1000 to reach 10200

= 10000 + offset

.. Value offset Reg = 0200^H



Memory Segmentation & Data Formats

Important:

We move inside a block of memory by offset.

Offset Min value = 0000^H

Offset Max value = $FFFF^H$ } 64 Kbyte

This offset can cover 2^{16} Value of Add = 2^{16} Location with respect.

To a start "Referred by Seg. Reg"

So Segment Size = 2^{16} byte = 64 Kbyte

CS – DS – ES – SS defining start of 4 segments

So Only 4 Segments are active At the same time

. . Each segment size = 64 Kbyte

So 4 Seg = 4*64 kbyte = 256 Kbyte

So ¼ Total Memory is Active at any time.

Memory Segmentation & Data Formats



To Change the code Seg. (for Ex.) we change value of **CS.**

So ¼ Memory is active at a time, this Means we can have up to 16 Seg.

- Only 4 are Active and accessable at the same time and rest of them are not reachable until we change the Seg. Register Value.

  So we may have 4 code segments, 4 Data Seg. 4 Extra Segment. And **4** Stack Segment.

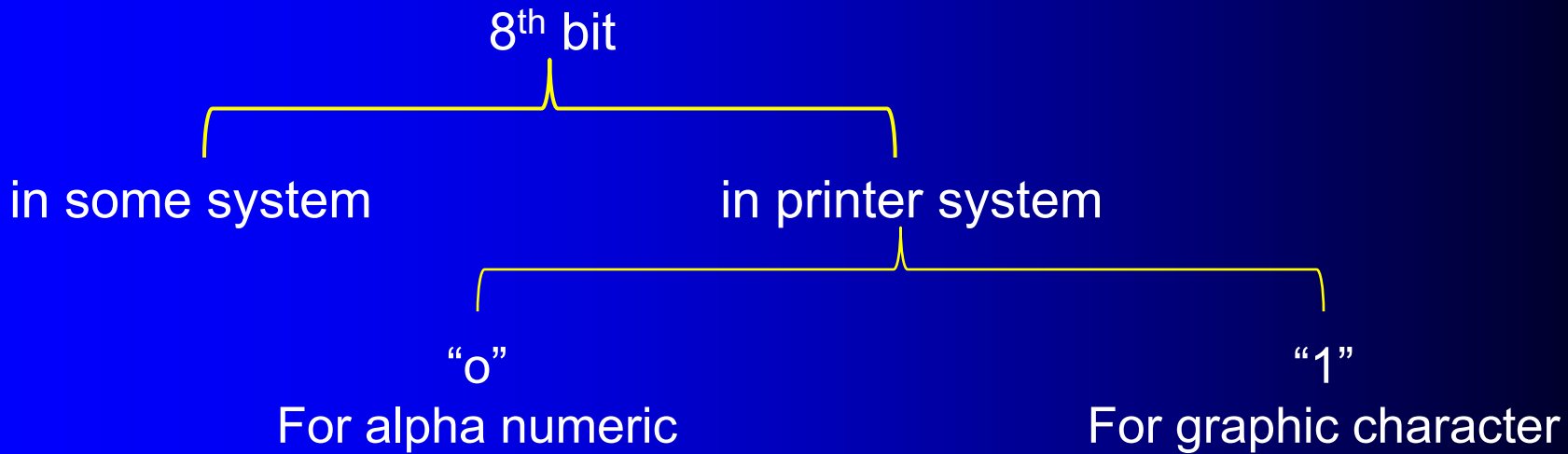
DATA FORMATS

Data can be represented in :

(1) ASCII (American standard code for info interchange).
(2) BCD
(3) 8 bit signed and unsigned integers .
(4) 16 bit signed and unsigned integers .
(5) 32 bit signed and unsigned integers .
(6) Short & long real numbers (floating .point numbers).

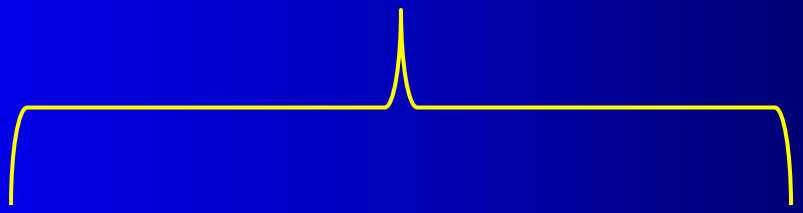
[1] ASCII Code

- Used to represent alpha numeric characters.
- 7 bit representations.
- In some system, 8th bit hold the parity.



[2] BCD “Binary coded Decimal”

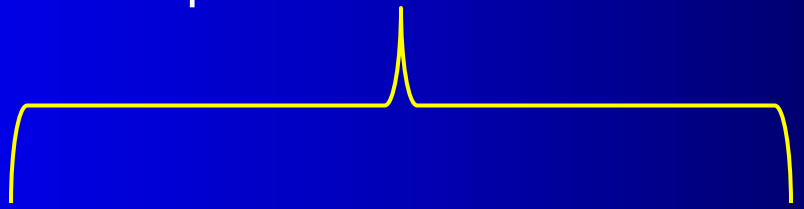
* 4 bits binary code 0000(0) 1001 → 1001(9)
BCD



packed
2 BCD digit / byte

unpacked
1 BCD digit / byte

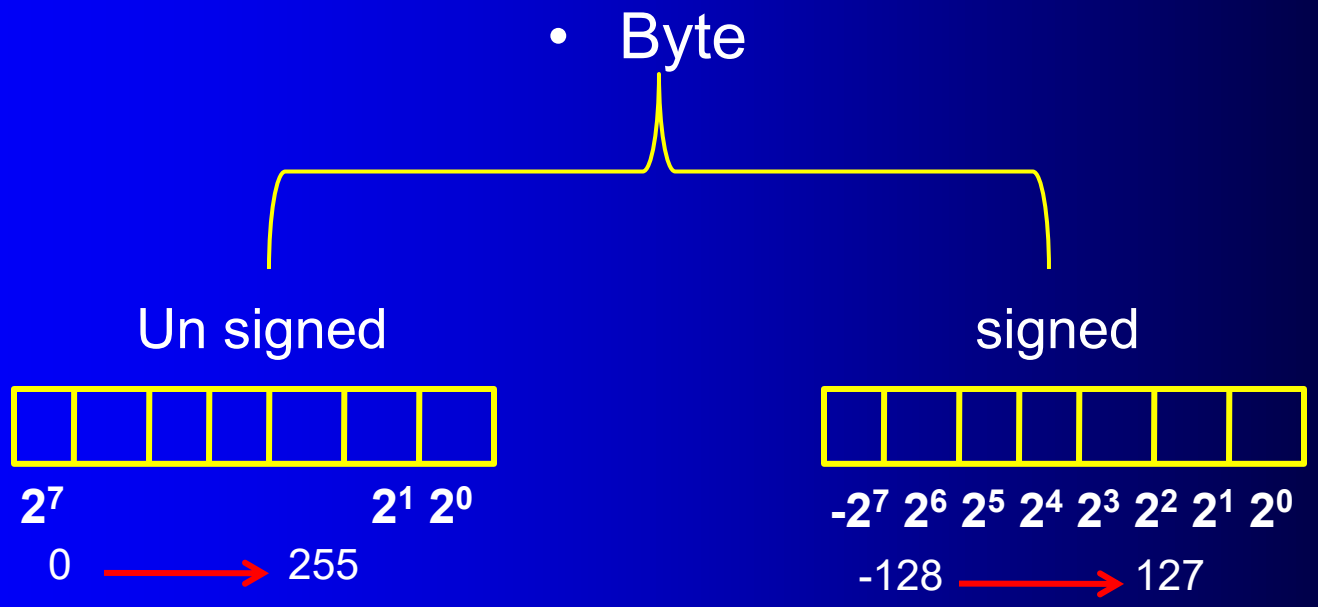
EX.12 → representation in **BCD**



paked
(0001) (0010)
(1) (2)

unpacked
(0 000 0001) (00 000010)
(1) (2)

- [3] Byte – sized Data

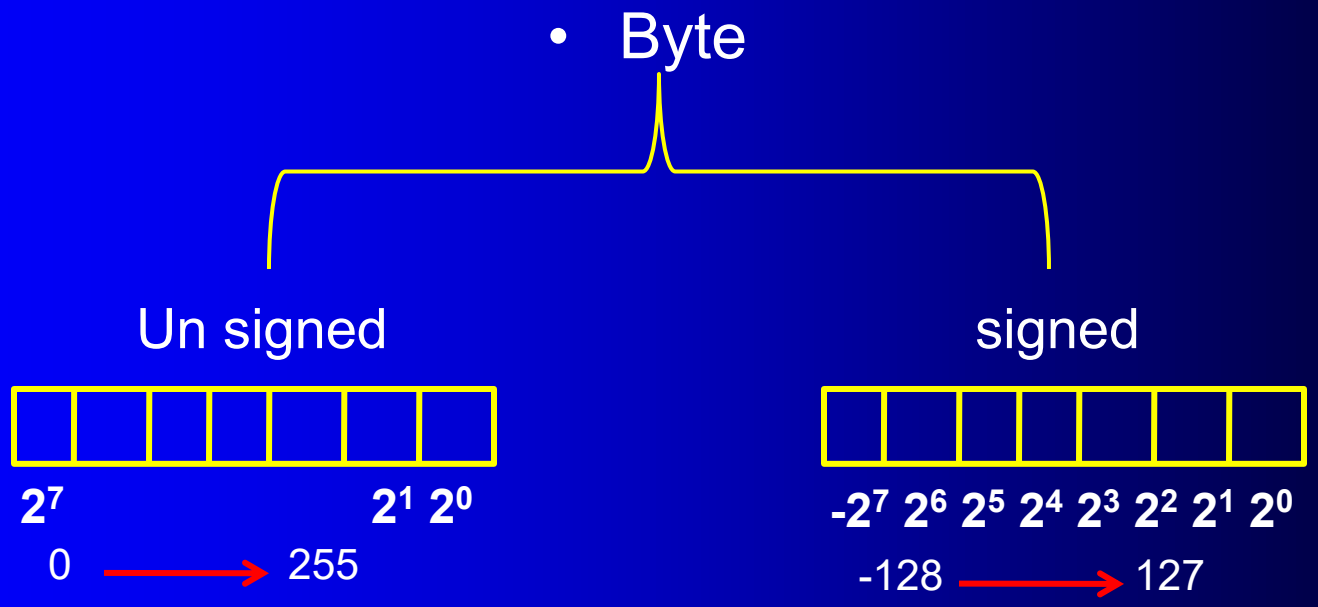


EX. if value is $8i^H$ ^{10000001}

Unsigned value. $81^H = 128^d + 1^d = 129^d$

Signed value $^{10000001} = -128+1 = 127^d$

- [3] Byte – sized Data



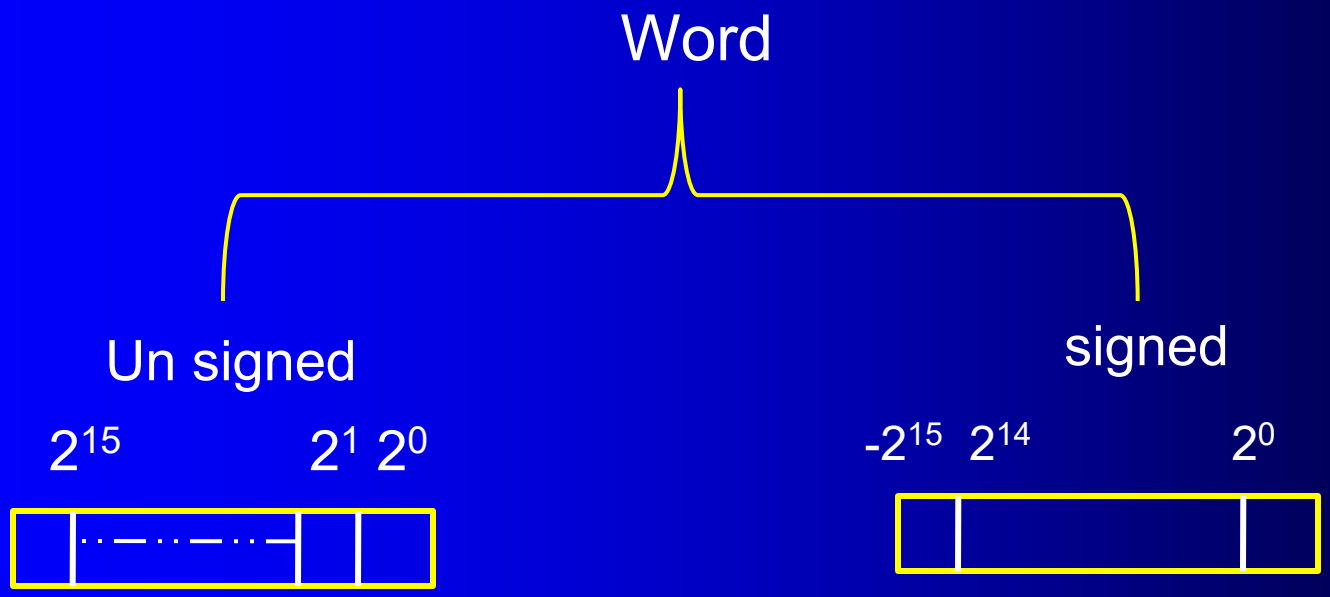
EX. if value is $8i^H$ ^{10000001}

Unsigned value. $81^H = 128^d + 1^d = 129^d$

Signed value $^{10000001} = -128+1 = 127^d$

4- Word – Sized - Data

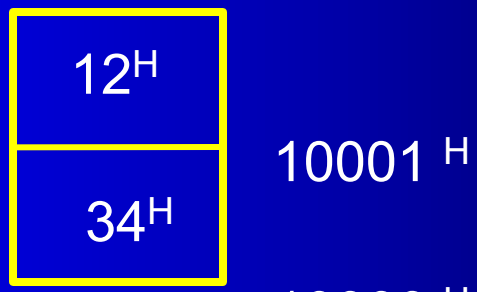
- Two Bytes



To Store word in memo , it takes Two Bytes.
(Two Memory location)

- Ex. Store 1234H at Address 10000^H

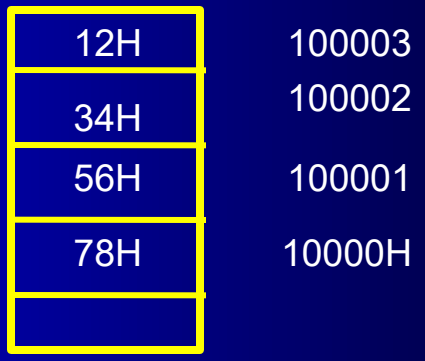
Sol



5- Double word Data:

- 4 Bytes Data
- Ex. Store 12345678H at address 10000^H

Sol



Note

Data are in Byte, word, Double word

3 Byte data will take storage of Double word.

Ex. store Dsi offset = 1000^{H} : 2356^{H}

at the address 10000^{H}

SOL

Data to sore is = 10000 + 2356 = 12356

in double word size .

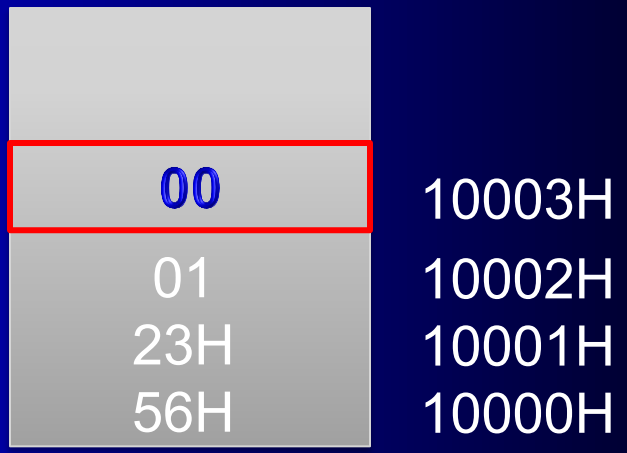
[6] Read Numbers:

Floating - point number

it composed of Exponent

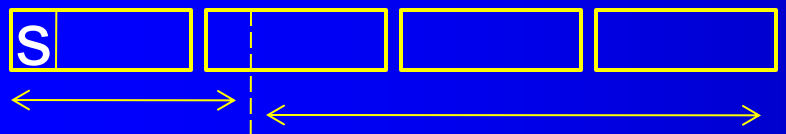
Mantissa “fraction part”

Ex. $12^{d} = 1100^{b}$ can be written as $1.1*2^{3}$



Real number

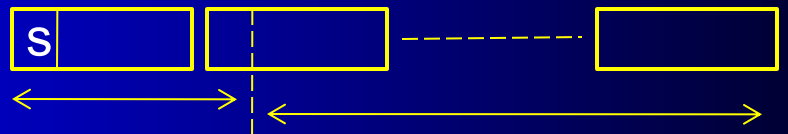
4 Bytes Short Form



8 bits Excess 127 exponent

32 bits Mantissa

8 Bytes long form



11 bits Excess 1023 exponent

52 bits Mantissa

Note

The exponent is stored by a biased exponent

With the single – precision form.

[4] **Decimal** – 1.75 ➡ - $1.11*2^{o}$

Biased exponent = 0+127 = 127

0	**0111 1111**	**110000 - - - - - - - 00**

In the real mode, show the starting and ending addresses of each segment located by the following segment register values:

(a) 1000H (b) 1234H (c) 2300H

a) segment register = 1000H
the starting address of the segment = 10000H,
the last address = 10000H + FFFFH = 1FFFFH.

Segment Register	Starting Address	Last Address
1000H	10000H	1FFFFH
1234H	12340H	2233FH
2300H	23000H	32FFFH

8086-80486 and Pentium-Pentium 4 default 16-bit segment and offset address combinations.

Segment	*Offset*	*Special Purpose*
CS (code segment)	IP (instruction pointer)	Instruction address
SS (stack segment)	SP (stack pointer) or BP (base pointer)	Stack address
DS (data segment)	BX(base index), DI(destination index), SI(source index), an 8-bit number, or a 16-bit number	Data address
ES (extra segment)	DI(destination index) for string instructions (Movs)	String destination address

80386 through Pentium 4 default 32-bit segment and offset address combinations.

Segment	*Offset*	*Special Purpose*
CS (code segment)	EIP (instruction pointer)	Instruction address
SS (stack segment)	ESP (stack pointer) or EBP (base pointer)	Stack address
DS (data segment)	EAX(accumulator segment)EBX(base index),ECX(cont segment), EDX EDI(destination index), ESI(source index), an 8-bit number, or a 32-bit number	Data address
ES (extra segment)	DI(destination index) for string instructions (Movs)	String destination address

Real Mode Memory Addressing

- Real mode memory addressing (80286 and above) allows access to data and programs located above the first 1M byte of memory.

END