8086
BIU-EU+Segmentation

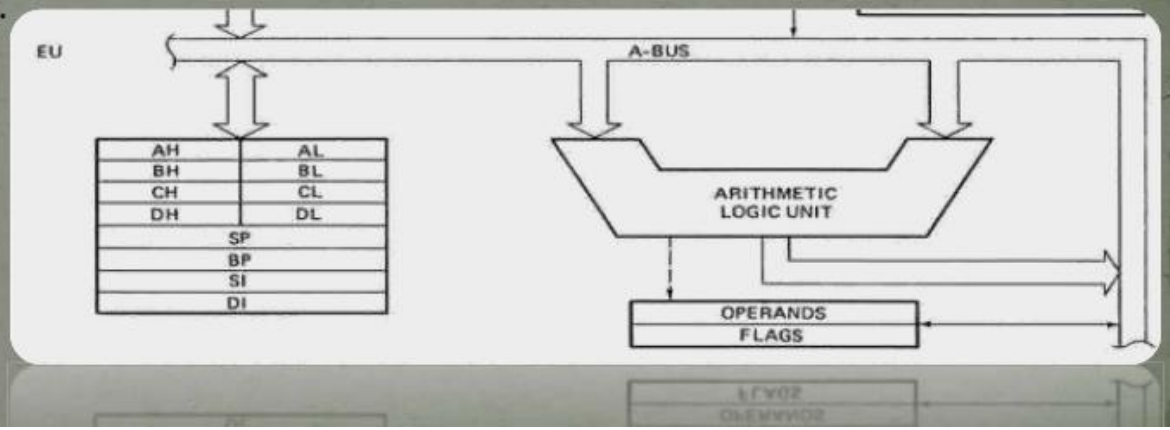
www.edutalks.org

EXCECUTION UNIT

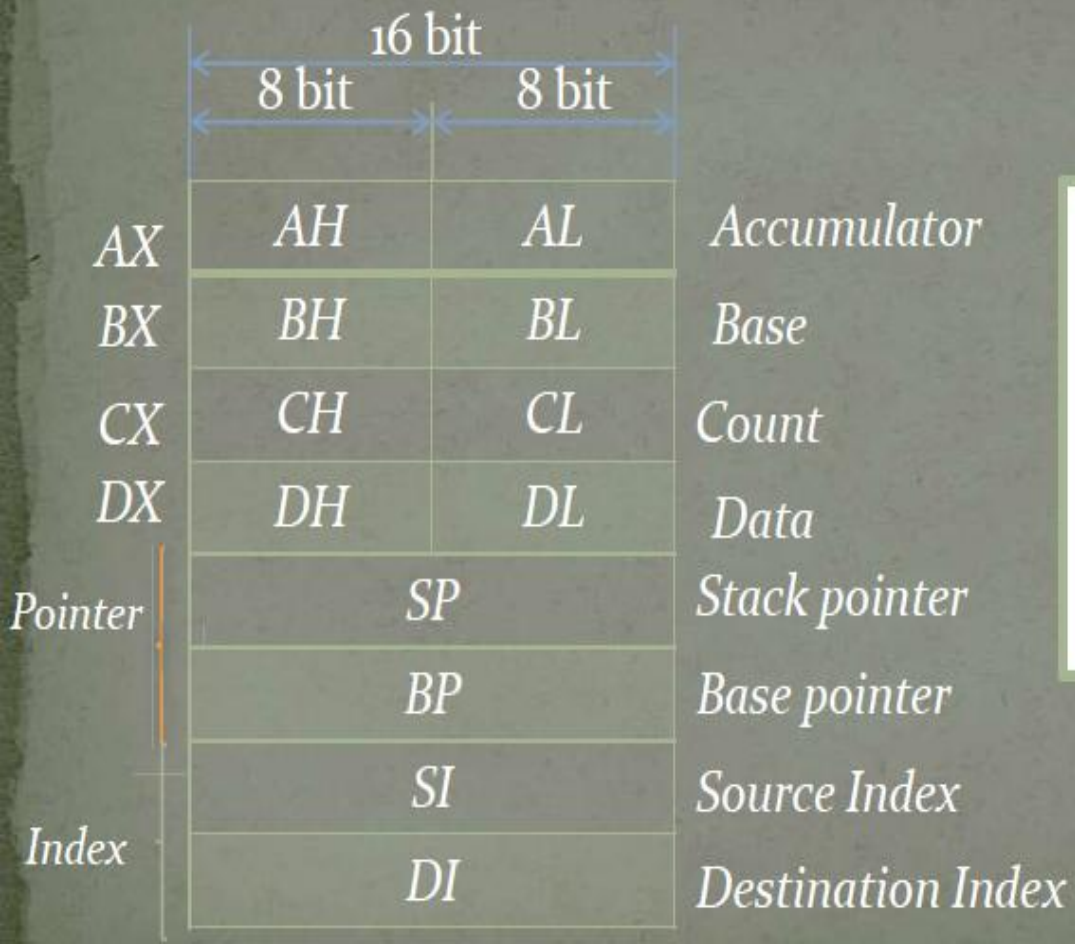
- ✓Decodes instructions fetched by the BIU
- ✓Generate control signals,
- ✓Executes instructions.

The main parts are

- ✓Control Circuitry
- ✓Instruction decoder
- ✓ALU



EXCECUTION UNIT-General Purpose Registers



- Normally used for storing temporary results
- Each of the registers is 16 bits wide (AX, BX, CX, DX)
- Can be accessed as either 16 or 8 bits AX, AH, AL

EXCECUTION UNIT-General Purpose Registers

Register	Purpose
AX	Word multiply, word divide, word I /O
AL	Byte multiply, byte divide, byte I/O, decimal arithmetic
AH	Byte multiply, byte divide
BX	Store address information
CX	String operation, loops
CL	Variable shift and rotate
DX	Word multiply, word divide, indirect I/O

Pointer And Index Registers

- used to keep offset addresses.
- Used in various forms of memory addressing.
- In the case of SP and BP the default reference to form a physical address is the Stack Segment(SS-will be discussed under the BIU)
- The index registers (SI & Di) and the BX generally default to the Data segment register(DS).

- SP: Stack pointer
  - Used with SS to access the stack segment
- BP: Base Pointer
  - Primarily used to access data on the stack
  - Can be used to access data in other segments

- SI: Source Index register
  - is required for some string operations
  - When string operations are performed, the SI register points to memory locations in the data segment which is addressed by the DS register. Thus, SI is associated with the DS in string operations.
- DI: Destination Index register
  - is also required for some string operations.
  - When string operations are performed, the DI register points to memory locations in the data segment which is addressed by the ES register. Thus, DI is associated with the ES in string operations.
- The SI and the DI registers may also be used to access data stored in arrays

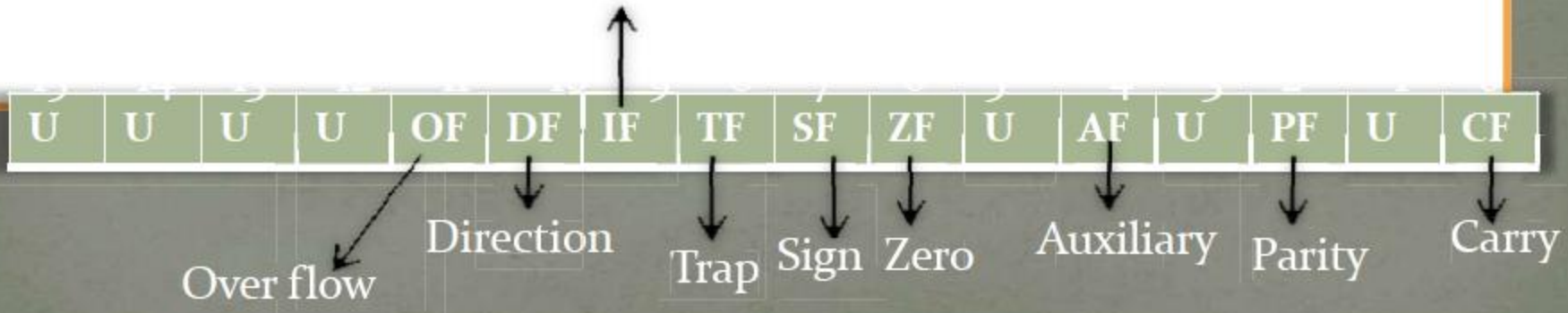
www.edutalks.org

EXCECUTION UNIT-Flag register

A flag is a flip flop which indicates some conditions produced by the execution of an instruction or controls certain operation of the EU .

In 8086 The EU contains

- a 16 bit flag register
- 9 of the 16 are active flags and remaining 7 are undefined.
  - 6 flags indicates some conditions- status flags
  - 3 flags –control Flags



www.edutalks.org

EXECUTION UNIT-Flag register

Flag	Purpose
Carry (CF)	Holds the carry after addition or the borrow after subtraction. Also indicates some error conditions, as dictated by some programs and procedures .
Parity(PF)	PF=0;odd parity ,PF=1;even parity.
Auxiliary (AF)	Holds the carry(half – carry) after addition or borrow after subtraction between bit positions 3 and 4 of the result (for example, in BCD addition or subtraction.)
Zero (ZF)	Shows the result of the arithmetic or logic operation. Z=1;result is zero. Z=0; The result is 0
Sign (SF)	Holds the sign of the result after an arithmetic/logic instruction execution. S=1;negative S=0;positive

www.edutalks.org

EXCECUTION UNIT-Flag register

Flag	Purpose
Trap(TF)	A control flag. Enables the trapping through an on-chip debugging feature.
Interrupt (IF)	A control flag. Controls the operation of the INTR (interrupt request) I=0;INTR pin disabled. I=1;INTR pin enabled.
Direction(DF)	A control flag. It selects either the increment or decrement mode for DI and /or SI registers during the string instructions.
Overflow (OF)	Overflow occurs when signed numbers are added or subtracted. An overflow indicates the result has exceeded the capacity of the machine

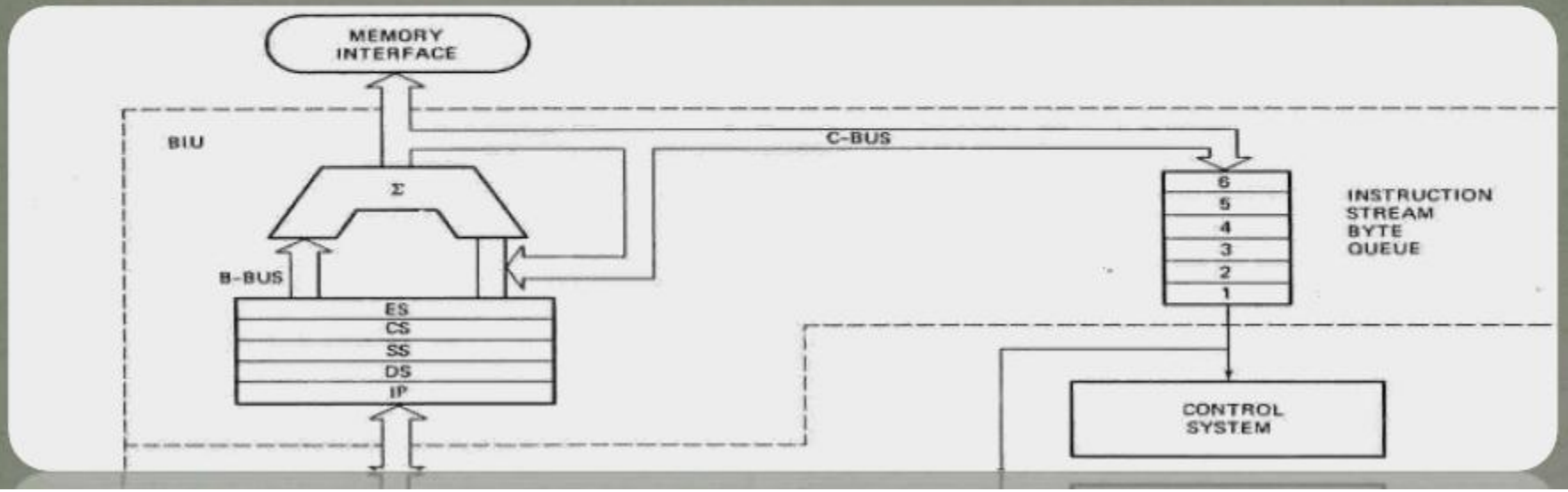
www.edutalks.org

EXCECUTION UNIT-Flag register

- Six of the flags are status indicators reflecting properties of the last arithmetic or logical instruction.
- For example, if register AL = 7Fh and the instruction ADD AL,1 is executed then the following happen

  AL = 80h
  CF = 0; there is no carry out of bit 7
  PF = 0; 80h has an odd number of ones
  AF = 1; there is a carry out of bit 3 into bit 4
  ZF = 0; the result is not zero
  SF = 1; bit seven is one
  OF = 1; the sign bit has changed
  Can be used to transfer program control to a new memory location;
  for example: ADD AL,1
  JNZ 0100h

BUS INTERFACE UNIT (BIU)



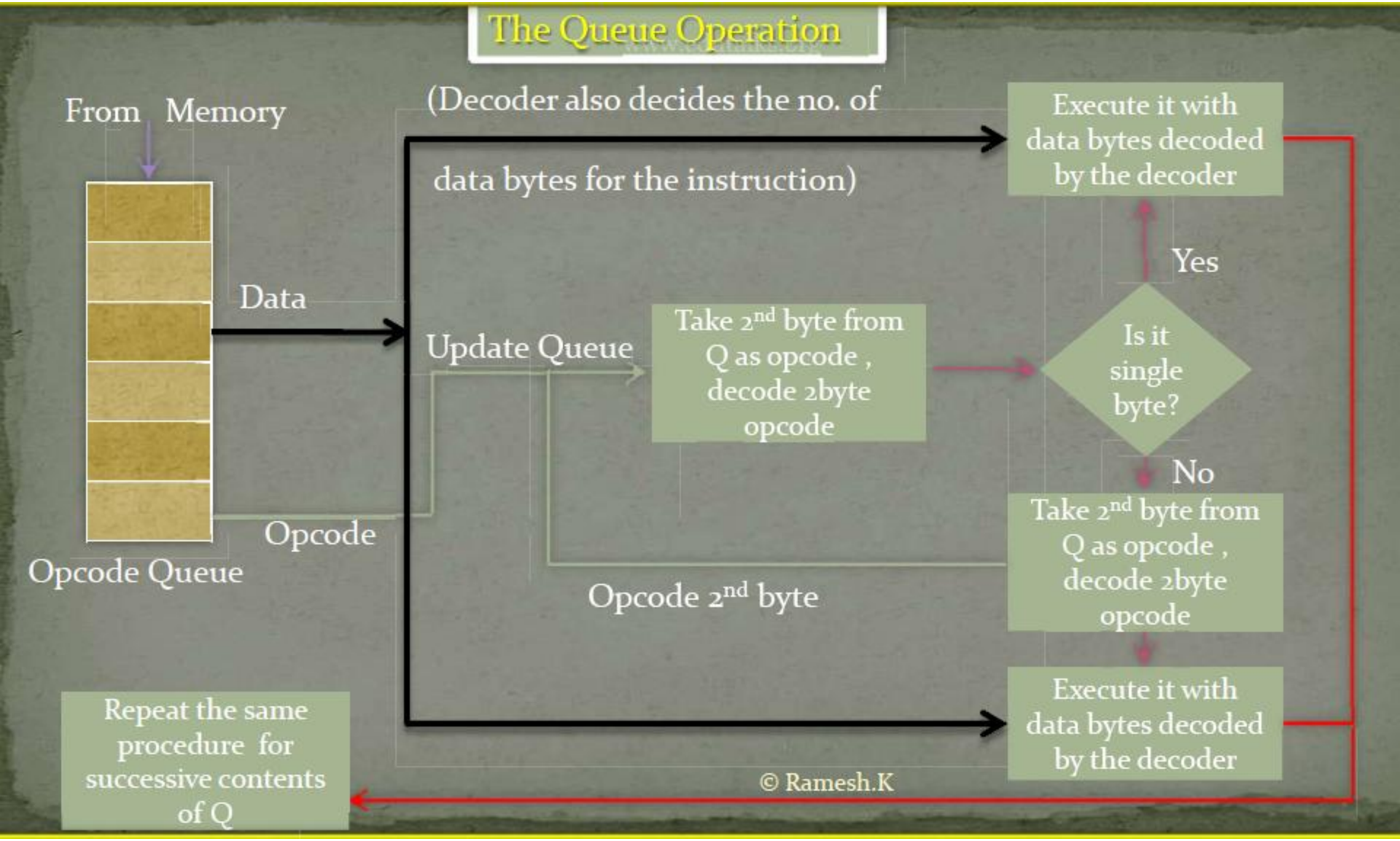
BUS INTERFACE UNIT (BIU)

Contains

- 4-byte instruction Queue (Q)
- The segment registers (CS, DS, ES, SS).
- The instruction Pointer (IP).
- The Address summing block(∑)

THE QUEUE (Q)

- The BIU uses a mechanism known as an **instruction stream queue** to implement a ***pipeline architecture.***
- This queue permits pre-fetch of up to **6 bytes** of instruction code. Whenever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by pre-fetching the next sequential instruction.
- These pre-fetching instructions are held in its **FIFO queue**. With its **16** bit data bus, the BIU fetches two instruction bytes in a single memory cycle.
- After a byte is loaded at the input end of the queue, it automatically shifts up through the **FIFO** to the empty location nearest the output.
- The **EU accesses the queue from the output end**. It reads one instruction byte after the other from the output of the queue.
- The intervals of no bus activity, which may occur between bus cycles are known as ***Idle state.***


The Queue Operation
From Memory
Data
Opcode
Opcode Queue
(Decoder also decides the no. of
data bytes for the instruction)
Update Queue
Take 2nd byte from Q as opcode , decode 2byte opcode
Is it single byte?
Yes
No
Execute it with data bytes decoded by the decoder
Take 2nd byte from Q as opcode , decode 2byte opcode
Opcode 2nd byte
Execute it with data bytes decoded by the decoder
Repeat the same procedure for successive contents of Q
© Ramesh.K

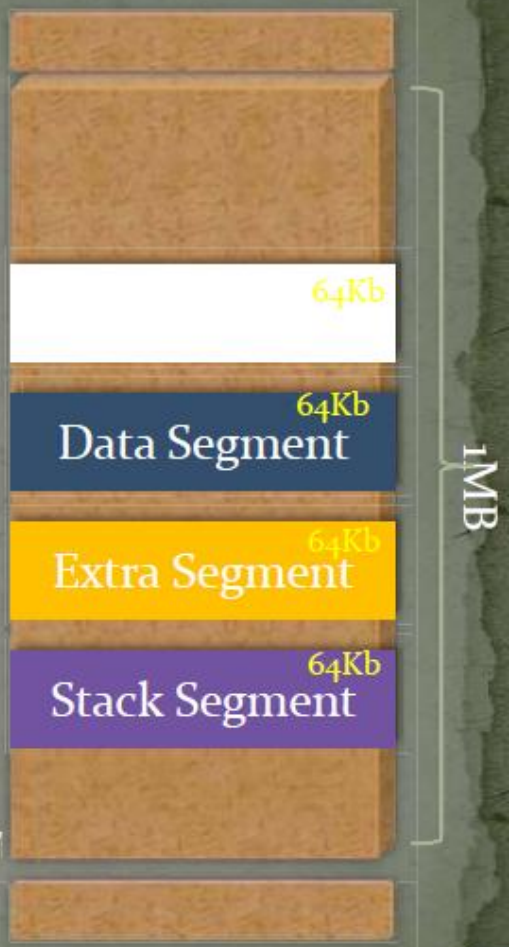
www.edutalks.org

Segmented Memory

- The memory in an 8086/88 based system is organized as segmented memory.
- The CPU 8086 is able to address 1Mbyte of memory.
- The Complete physically available memory may be divided into a number of logical segments.
- The 4 segments are Code, Data, Extra and Stack segments.
- A Segment is a 64kbyte block of memory.
- The 16 bit contents of the segment registers in the BIU actually point to the starting location of a particular segment.
- Segments may be overlapped or non-overlapped

Advantages of Segmented memory Scheme

- Allows the memory capacity to be 1Mb although the actual addresses to be handled are of 16 bit size.
- Allows the placing of code, data and stack portions of the same program in different parts(segments) of the m/y, for data and code protection.
- Permits a program and/or its data to be put into different areas of memory each time program is executed, i.e. provision for relocation is done .



© Ramesh.K

www.edutalks.org

Segmented Memory

- The memory in an 8086/88 based system is organized as segmented memory.
- The CPU 8086 is able to address 1Mbyte of memory.
- The Complete physically available memory may be divided into a number of logical segments.
- The 4 segments are Code, Data, Extra and Stack segments.
- A Segment is a 64kbyte block of memory.
- The 16 bit contents of the segment registers in the BIU actually point to the starting location of a particular segment.
- Segments may be overlapped or non-overlapped

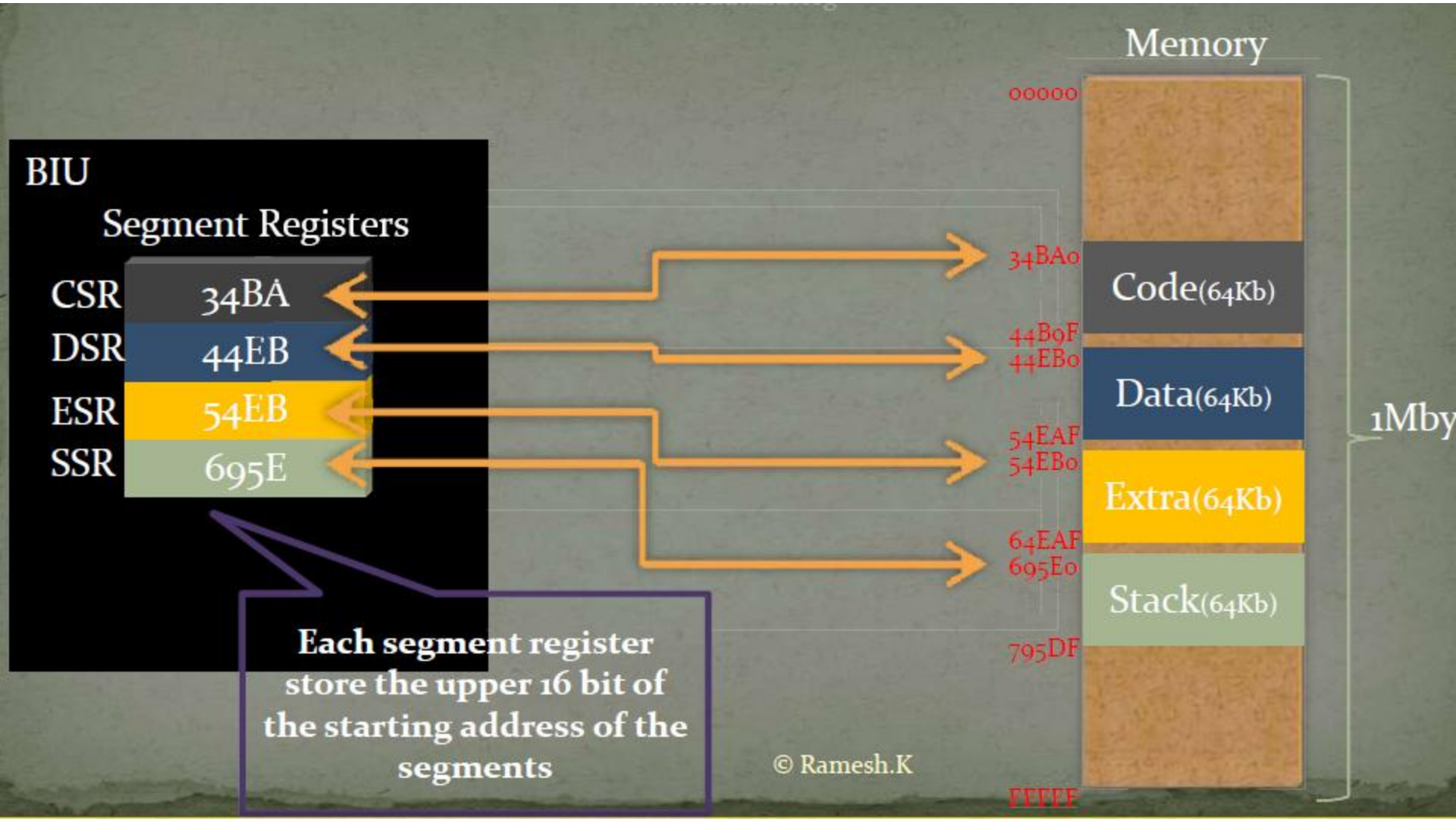
Advantages of Segmented memory Scheme

- Allows the memory capacity to be 1Mb although the actual addresses to be handled are of 16 bit size.
- Allows the placing of code, data and stack portions of the same program in different parts(segments) of the m/y, for data and code protection.
- Permits a program and/or its data to be put into different areas of memory each time program is executed, i.e. provision for relocation is done .

© Ramesh.K

Segment Registers

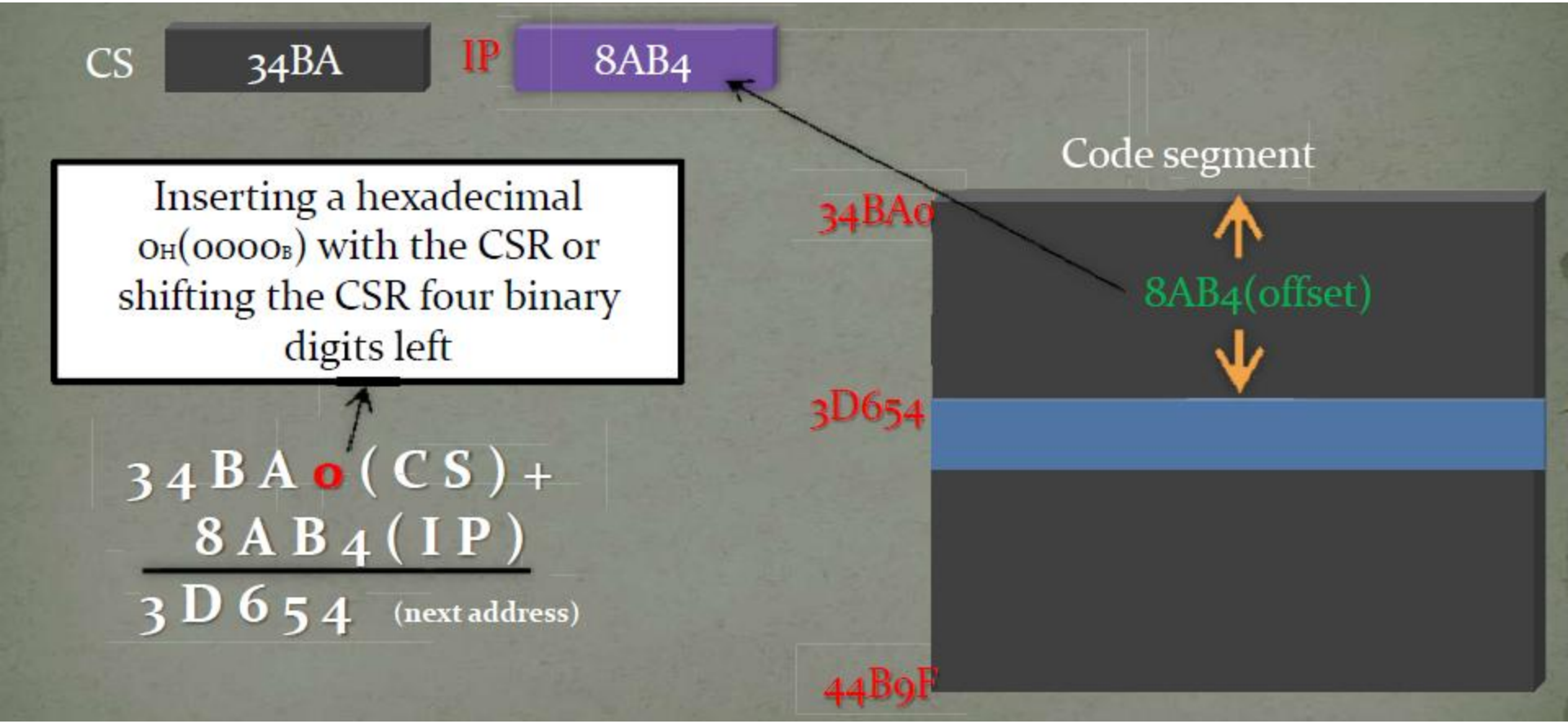
- In 8086/88 the processors have 4 segments registers
- Code Segment register (CS), Data Segment register(DS), Extra Segment register(ES) and Stack Segment (SS)register.
- All are 16 bit registers.
- Each of the Segment registers store the upper 16 bit address of the starting address of the corresponding segments.


BIU
Segment Registers
CSR
34BA
DSR
44EB
ESR
54EB
SSR
695E
Each segment register store the upper 16 bit of the starting address of the segments
© Ramesh.K
Memory
00000
34BA0
Code(64Kb)
44B0F
44EB0
Data(64Kb)
54EAF
54EB0
Extra(64Kb)
64EAF
695E0
Stack(64Kb)
795DF
FFFFF
1Mby

www.edutalks.org

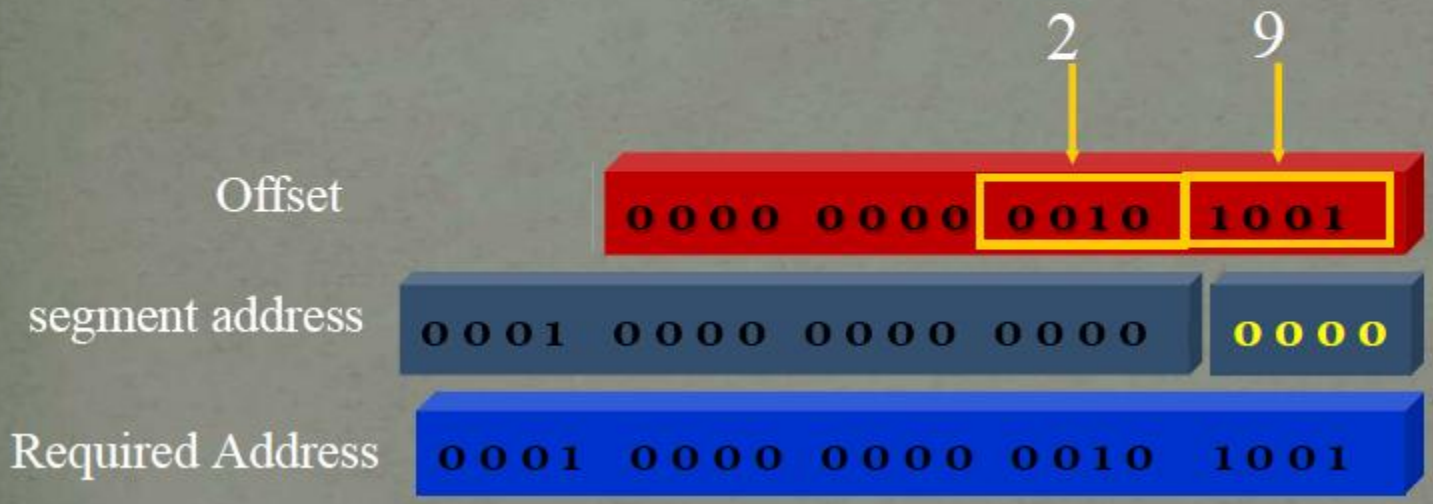
Instruction Pointer (IP) and the address Summing Block

- The IP register hold the 16 bit offset address or the offset, of the next code byte within the Code segment.
- An offset is the distance (in terms of address) from the beginning of a segment to a particular instruction or variable.
- The IP always references the Code segment register (CS).
- The physical address of the next instruction is formed by combining the CS and IP.
- To form a 20bit address of the next instruction, the 16 bit address of the IP is added (by the address summing block) to the address contained in the CS , which has been shifted four bits to the left.
- The following examples shows the CS:IP scheme of address formation


CS
34BA
IP
8AB4
Code segment
34BA0
8AB4(offset)
3D654
44B9F
Inserting a hexadecimal 0H(0000B) with the CSR or shifting the CSR four binary digits left
34BA0(CS) +
8AB4(IP)
3D654 (next address)

Example For Address Calculation (segment: offset)

- If the data segment starts at location 1000h and a data reference contains the address 29h where is the actual data?



Questions??????????????????

THANK YOU.