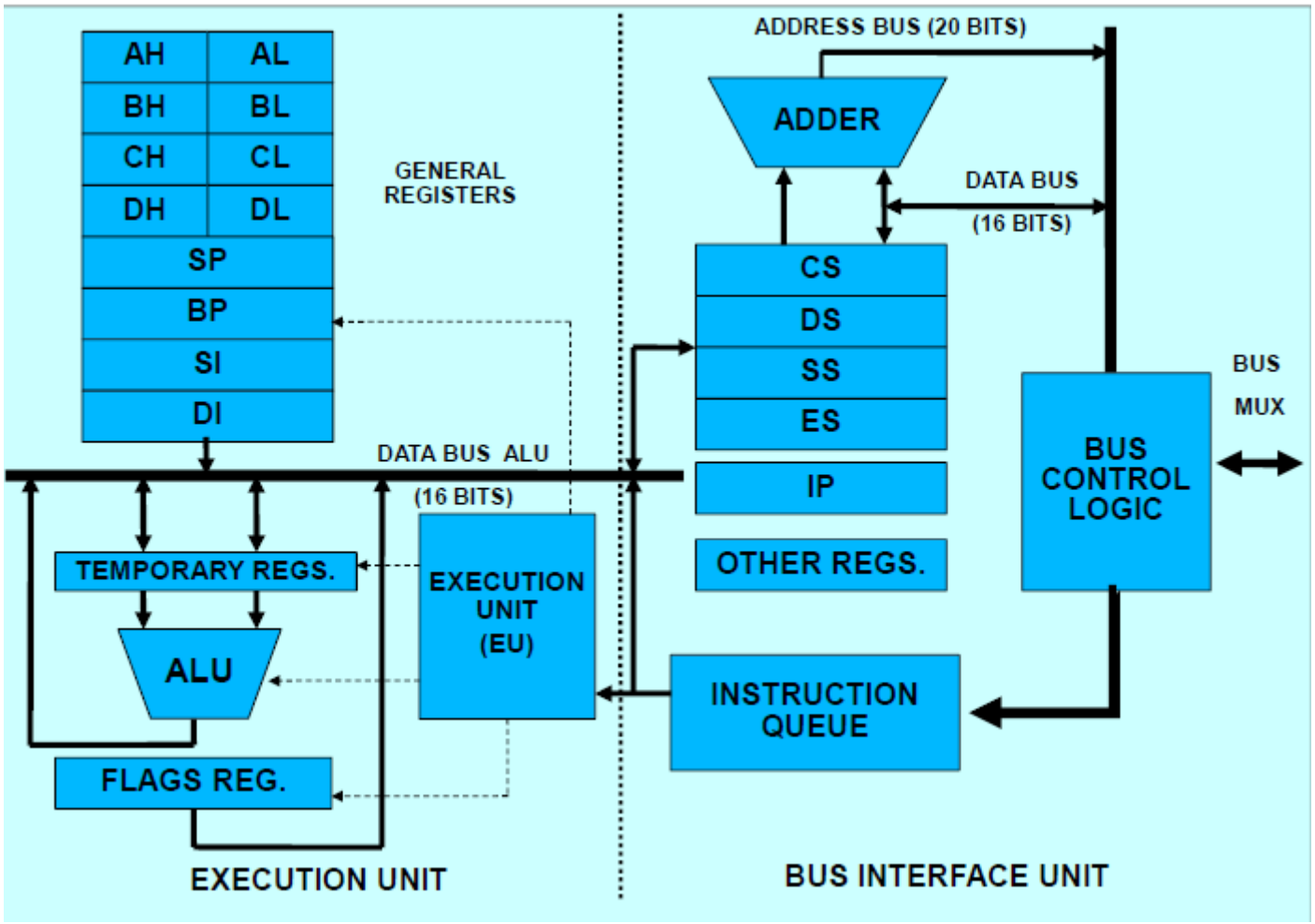
P1: Addressing modes. Directives and operators..

Programming Model of the Intel 80x86 (Unit 2)

- The 80x86 family as a particular case.
- Internal registers and architecture of the 80x86.
- **Memory access and organization.**
- **Addressing modes.**
- **Directives and operators of the 80x86 assembler.**
- **Structure of an assembly program.**
- Assembly instructions.
- Memory map of the PC system.
- Interrupts: mechanism and interrupt vectors.

- **Internal registers and architecture of the 80x86**



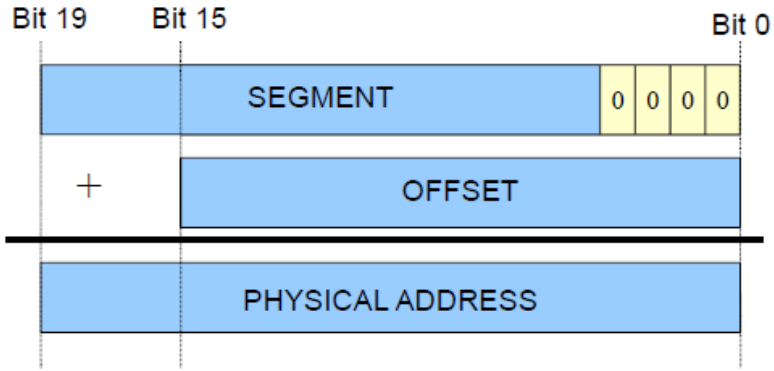
- Data registers: **AX, BX, CX, DX**
  - AX: Multiply, divide and I/O operations.
  - BX: Base register for indirect addressing (pointer to the base of a table)
  - CX: Loop counter.
  - DX: Multiply, divide and I/O operations..
- Pointer registers: **SP , BP , SI , DI**
  - Involved in the memory addressing as displacements (offsets) with respect to the memory zones indicated in segment registers.
  - SP (Stack Pointer): Used in conjuntction with the stack segment register SS. Involved in:
    - Subroutine calls
    - Interrupts
    - Stack management instructions
  - BP (Base Pointer): Used in conjunction with the stack segment register SS. Useful for accessing the parameters of subroutines passed through the stack.
  - SI (Source Index): Used for indexing memory tables (reading). For any use if it is free.
  - DI (Destination Index): Used for indexing memory tables (writing). For any use if it is free.
- Segment registers: **CS , SS , DS , ES**
  - Involved in the memory addressing pointing to 64KB areas of memory (segments).
  - CS (Code Segment): Pointer to the machine code segment (program). Along with the instruction pointer IP, it constitutes the program counter.
  - SS (Stack Segment): Pointer to the stack segment. Along with SP or BP, it points to an absolute memory position in the stack.
  - DS (Data Segment): Pointer to the main data segment (global variables).
  - ES (Extra Segment): Pointer to the additional data segment (global variables).

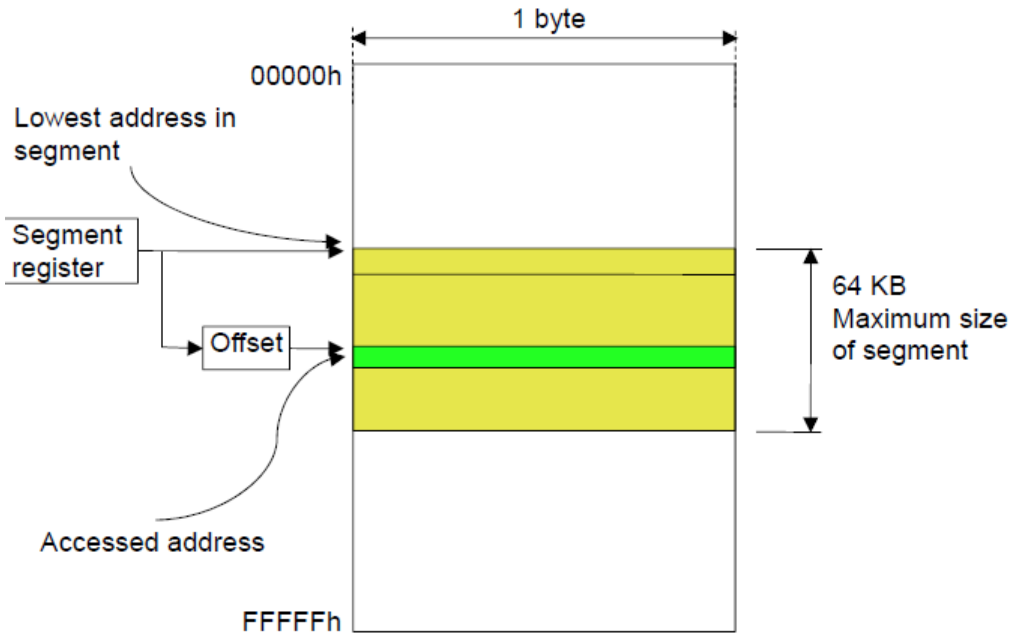
Memory access (real mode)

Hardware: 20 address bits (A19-A0)

Software: 32 bits (16 bits for Segment + 16 bits for Offset)

PHYSICAL ADDRESS = **Segment** x **16** + **Offset**





- **Addressing modes.**

Immediate addressing

The source operand is always a value and the destination a register.

Examples:

```
mov CL, 3Fh        ; 3Fh ⇒ CL
mov SI, 4567h      ; 4567h ⇒ SI
```

Register addressing

Both operands are always registers.

Examples:

```
mov DX, CX         ; CX ⇒ DX
mov BH, CL         ; CL ⇒ BH
```

Direct addressing

The *offset* of the memory position to be accessed is specified in the instruction. By default, the segment indicates **DS**.

Examples if DS = **3000h**:

mov DX, DS:[678Ah] Load **DL** with the content of memory ; position **3678Ah** and **DH** with the content of memory position **3678Bh**.
mov AL, DS:[32h] Load **AL** with the content of memory position **30032h**
mov [800h], BL Write the content of **BL** into the memory position **30800h**

Register indirect addressing

The effective address of the operand is contained in registers **BX**, **BP**, **SI** or **DI**.

Example:

```
mov AX, [BX]
```

Based addressing

The effective address is obtained by adding a displacement to register **BX** or **BP**.

Equivalent examples if *offset* of TABLE is 4:

```
mov AX, [BX]+4
mov AX, 4[BX]
mov AX, TABLE[BX]
mov AX, [BX+4]
```

Indexed addressing

The effective address is calculated by adding a displacement to the content of **SI** or **DI**.

Equivalent examples if *offset* of TABLE is 4:

```
mov AX, [SI]+4
mov AX, 4[SI]
mov AX, TABLE[SI]
mov AX, [SI+4]
```

Based-indexed addressing

The effective address is obtained by adding **BX** or **BP** plus **SI** or **DI** and/or a direct offset.

Examples:

```
mov AX, TABLE[BX][SI]
mov AX, TABLE+[BX]+[SI]
```

Relative addressing

Used in conditional jumps: The operand is a displacement of 8 bits with sign (-128 to 127) that is added to the instruction pointer **IP**.

Example:

```
jnc 26
jz label      ; Whenever the label is at a distance larger
              ; than or equal to -128 and lower than 128
```

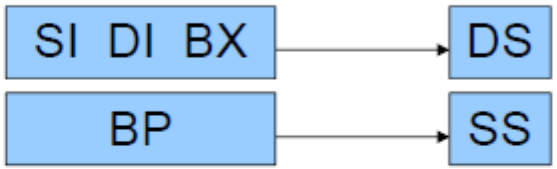
Implicit addressing

It is not necessary to specify the operand (it is implicit).

Examples:

```
cli    ; Set the interrupt flag to 0
stc    ; Set the carry flag to 1
```

Default segment registers for indirect, relative and indexed addressing:



Forced use of another segment register:

The address is prefixed with the desired register.

Examples with **DS = 3000h**, **CS = 2000h**, **ES = A000h**, **SS = E000h**, **SI = 100h** and **BP = 500h**

(caution: Tasm needs DS to be declared wrt slides)

mov DX, **DS:**[678Ah] ; [3678Ah] & [3678Bh] ⇒ DX
mov DX, **CS:**[678Ah] ; [2678Ah] & [2678Bh] ⇒ DX
mov **ES:**[SI], AL ; AL ⇒ [A0100h]
mov **SS:**[1000h+SI], CX ; CL ⇒ [E1100h] & CH ⇒ [E1101h]
mov SI, [BP] ; [E0500h] & [E0501h] ⇒ SI
mov **DS:**[BP], DI ; DI ⇒ [30500h] & [30501h]

If the instruction does not specify any registers, the number of bytes of the data transfer must be defined explicitly:

Examples with **DS** = **3000h**, **CS** = **2000h**, **ES** = **A000h**, **SS** = **E000h**, **SI** = **100h** and **BP** = **500h**

mov **BYTE PTR DS:**[3Ah], 4Fh ; 4Fh ⇒ [3003Ah]
mov **WORD PTR DS:**[3Ah], 4Fh ; 4Fh ⇒ [3003Ah] , 0 ⇒ [3003Bh]
mov **WORD PTR** ES:[3Ah], 2000h ; 0 ⇒ [A003Ah] , 20h ⇒ [A003Bh]
mov **BYTE PTR DS:**[3Ah+SI], 4 ; 4 ⇒ [3013Ah]

- Symbol definition directives

They assign symbolic names to expressions. After the assignment, the name can be utilized all over the program.

- EQU can be used to assign text or numerical expressions.
- = only allows numerical assignments and can be redefined.

Examples:

```
K           EQU     1024
TABLE       EQU     TABLE[BX+SI]
K2          EQU     K
COUNTER     EQU     CX
DOUBLEK     EQU     2*K
MIN_DAYS    EQU     60*24

CONSTANT =  20h
CONSTANT =  CONSTANT + 1
```

Data definition directives

They allocate memory space, assign a value and define a name for the variable.

- **DB** allocates 1 byte
- **DW** allocates 2 bytes (1 word)
- **DD** allocates 4 bytes (2 words)
- **DQ** allocates 8 bytes (4 words)

Examples:

```
NUMBERS     DB      4, 5*9, 10h+4, 23h, 'A'         ; 1 byte per element
TEXT        DB      "Final", 13, 0Ah
NUM         DW      1000, -200, 400/60, 80h         ; 2 bytes per element
NUMMM       DD      200000h                         ; 4 bytes per element
            DB      6 dup (10h)                     ; 10h six times
            DB      10h dup("Stack")                ; StackStackStack .....
LETTER      DB      ?           ; allocate 1 byte without assigning a value
LETTERS     DB      8 dup (?)
NEAR        DW      LETTER      ; store offset of LETTER
FAR         DD      LETTER      ; store offset and segment of LETTER
```

Addressing examples.

Physical address= segment*10h + offset.

DS=3000h
Offset=678Ah => Real = 3000h*10h+ 678Ah= **3678Ah**

Phisical address: 60006h = segment*10h+offset.
Segmento=6000h
Offset=6h.

Physical address 3678Fh

Segment=3678h
Offset= Fh

But also
Segment= 3000h
Offset= 678Fh

Addressing examples.

TD View:

4675:0000 45 33 28 19 23 FB 14 BB
…..
….
47FB:0000 00 12 34 56 BB FB 00 18

Registers DS=4675; ES=4675; SI=0002; BX=0001, SS= 47FB, BP=0002
The offset for TABLA in DS is 4.

The AX value will be:

Mov AX, SI   (AX=0002)
Mov AL, [SI] (AX=XX28)
Mov AX, ES:[SI] (AX=1928)
Mov AX, [BX] (AX=2833)
Mov AX, TABLA[SI] (AX=BB14)
Mov AH, TABLA[BX][SI] (AX=BBXX)

Mov AX, [BP] (AX=5634) (by default, the SS segment)