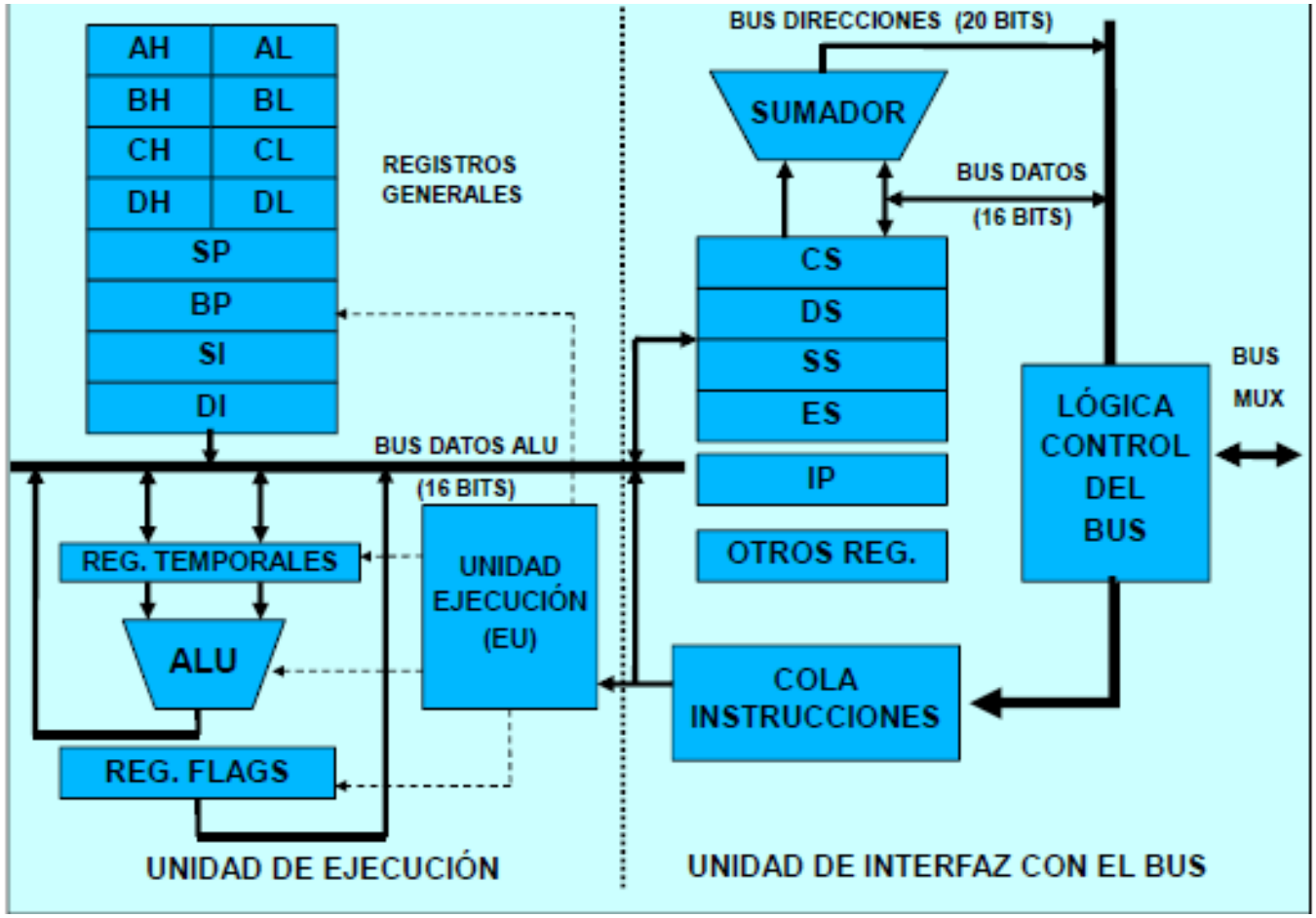
Practica 1. Modelo de programación del 80x86 de Intel.

- Familia 80x86 como caso particular.
- Registros internos y arquitectura del 80x86.
- **Acceso y organización de la memoria.**
- **Modos de direccionamiento.**
- **Directivas y operadores del ensamblador del 80x86.**
- **Estructura de un programa en ensamblador.**
- Instrucciones del ensamblador
- Mapa de Memoria del sistema PC.
- Interrupciones: mecanismo y vectores de interrupción.

- **Acceso y organización de la memoria.**



- Registros de datos
  - **AX**: Multiplicar, dividir y operaciones de E/S.
  - **BX:** Registro base para direccionamiento indirecto (apunta a la base de una tabla)
  - **CX:** Contador de bucles.
  - **DX**: Multiplicar, dividir, operaciones de E/S.
- Registros punteros: **SP , BP , SI , DI**
  - Intervienen en el direccionamiento de memoria como desplazamientos (*offsets*) respecto a las zonas de memoria indicadas en registros de segmento.
  - **SP** (*Stack Pointer*): Usado junto al registro de segmento de pila **SS**
  - **BP** (*Base Pointer*): Usado junto al registro de segmento de pila **SS**. Útil para acceder a los parámetros de subrutinas pasados por pila.
  - **SI** (*Source Index*): Usado para indexar tablas en memoria (lectura).
  - **DI** (*Destination Index*): Usado para indexar tablas en memoria (escritura).
- Registros de segmento: **CS , SS , DS , ES**
  - Intervienen en el direccionamiento de memoria indicando zonas de 64KB de memoria (segmentos).
  - **CS** (*Code Segment*): Indica el segmento de código máquina (programa). Junto con el puntero de instrucciones **IP** constituye el *contador de programa*.
  - **SS** (*Stack Segment*): Indica el segmento de pila. Junto con **SP** o **BP** indica una posición absoluta de memoria en la pila.
  - **DS** (*Data Segment*): Indica el segmento principal de datos (variables globales).
  - **ES** (*Extra Segment*): Indica el segmento adicional de datos (variable globales).

Memoria física de un sistema basado en 8086 organizada como 2^{20} posiciones de 1 byte (1 MB).
Memoria física de 1 MB dividida a nivel lógico en “segmentos” de 64 KB.

Los segmentos empiezan en direcciones múltiplo de 16. Dos segmentos consecutivos están separados por 16 bytes.

En un programa, las instrucciones suelen estar en un segmento, los datos en uno o varios segmentos distintos y la pila en otro (hay casos en que esto no se cumple).

La CPU puede acceder a la vez hasta a cuatro segmentos distintos (registros **CS**, **DS**, **ES** y **SS** con valores distintos).

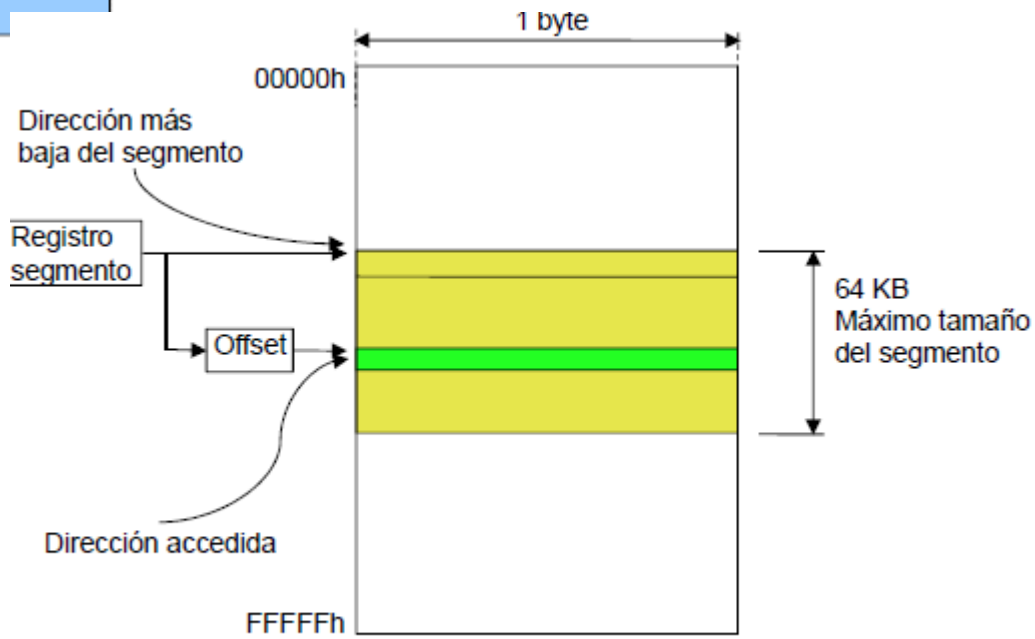
Puede haber solapamiento total o parcial de segmentos (caso extremo: **CS**, **DS**, **ES** y **SS** con mismo valor).

El programa puede cambiar en cualquier momento el valor de los registros de segmento.

Acceso a memoria (modo real)

Hardware: 20 bits de dirección (A19-A0)
Software: 32 bits (16 bits de Segmento y 16 bits de Offset)

DIRECCIÓN FÍSICA = **Segmento** x **16** + **Offset**



- ***CS* = A783h** ***(segmento)***
  IP* = 403Eh** ***(offset)

  Dirección física = **A783h** x **16** + **403Eh** =
  A783h x **10h** + **403Eh** =
  A7830h + **403Eh** = **AB86Eh**

- **Modos de direccionamiento.**

- Direccionamiento inmediato

  El operando fuente siempre es un valor y el destino un registro.

  Ejemplos:

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

- Direccionamiento por registro

  Ambos operandos son siempre registros.

  Ejemplos:

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

- Direccionamiento directo

  El *offset* de la posición de memoria a la que se quiere acceder se especifica en la instrucción. Por defecto, el segmento lo indica **DS**.

  Ejemplos si DS = **3000h**:

```
mov DX, [678Ah]  ; carga en DL el contenido de la posición
                 ; de memoria 3678Ah y en DH el
                 ; contenido de la posición de memoria
                 ; 3678Bh.
mov AL, [32h]    ; carga en AL el contenido de la posición
                 ; de memoria 30032h
mov [800h], BL   ; carga en la posición de memoria 30800h
                 ; el contenido de BL.
```

En TASM el direccionamiento directo "por defecto" no funciona.

mov DX, DS:[678Ah]

mov AL,DS:[32h]

mov DS:[800h],BL

- **Direccionamiento indirecto por registro**

  La dirección efectiva del operando está contenida en uno de los registros **BX**, **BP**, **SI** o **DI**.

  Ejemplo:

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

- **Direccionamiento relativo a base**

  La dirección efectiva se obtiene sumando un desplazamiento al registro **BX** o al **BP**.

  Ejemplos equivalentes si *offset* de la TABLA es 4:

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

- **Direccionamiento indexado**

  La dirección efectiva se calcula sumando un desplazamiento al contenido de **SI** o **DI**.

  Ejemplos equivalentes si *offset* de la TABLA es 4:

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

- **Direccionamiento indexado a base**

  La dirección efectiva se obtiene sumando **BX** o **BP** con **SI** o **DI** y/o un offset directo.

  Ejemplos:

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

- Direccionamiento relativo

  Usado en saltos condicionales: El operando es un desplazamiento de 8 bits con signo (-128 a 127) que se suma al puntero de instrucciones **IP**.

  Ejemplos:

```
jnc 26
jz etiqueta   ; si la etiqueta está a una distancia
              ; mayor o igual que -128 y menor que 128
```

- Direccionamiento implícito

  No es necesario indicar el operando (es implícito).

  Ejemplos:

```
cli     ; pone a 0 el flag de interrupciones
stc     ; pone a 1 el flag de acarreo
```

- Registros de segmento por defecto en direccionamiento indirecto, relativo e indexado:

- Uso forzado de otro registro de segmento:
  La dirección se prefija con el registro deseado.
- Ejemplos con **DS = 3000h**, **CS = 2000h**, **ES = A000h**, **SS = E000h**, **SI = 100h** y **BP = 500h**

(DS tiene que estar explícito en direccionamiento directo)

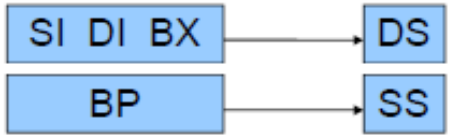
mov DX, **DS:**[678Ah] ; [3678Ah] & [3678Bh] ⇒ DX
mov DX, **CS:**[678Ah] ; [2678Ah] & [2678Bh] ⇒ DX
mov **ES:**[SI], AL ; AL ⇒ [A0100h]
mov **SS:**[1000h+SI], CX ; CL ⇒ [E1100h] & CH ⇒ [E1101h]
mov SI, [BP] ; [E0500h] & [E0501h] ⇒ SI
mov **DS:**[BP], DI ; DI ⇒ [30500h] & [30501h]

Si la instrucción no asocia el tamaño de la transferencia con un registro, el número de bytes debe ser definido explícitamente

Ejemplos con **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]

- Registros de segmento por defecto en direccionamiento indirecto, relativo e indexado:



- Uso forzado de otro registro de segmento:

  La dirección se prefija con el registro deseado.

- Ejemplos con **DS = 3000h**, **CS = 2000h**, **ES = A000h**, **SS = E000h**, **SI = 100h** y **BP = 500h**

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

- **Directivas y operadores**

Directivas de definición de símbolos

Asignan nombres simbólicos a expresiones. Después de la asignación, el nombre puede ser empleado en cualquier parte del programa.

- EQU puede usarse para asignar texto o expresiones numéricas.
- = sólo permite asignaciones numéricas y puede redefinirse.

Ejemplos:

```
K           EQU     1024
TABLA       EQU     TABLA[BX+SI]
K2          EQU     K
CONTADOR    EQU     CX
DOBLEK      EQU     2*K
MIN_DIAS    EQU     60*24

CONSTANTE   =       20h
CONSTANTE   =       CONSTANTE + 1
```

Directivas de definición de datos

Reservan espacio de memoria, asignan un valor y definen un nombre para la variable.

- **DB** reserva 1 byte
- **DW** reserva 2 bytes (1 palabra)
- **DD** reserva 4 bytes (2 palabras)
- **DQ** reserva 8 bytes (4 palabras)

Ejemplos:

```
NUMEROS     DB      4, 5*9, 10h+4, 23h, 'A'         ; 1 byte por elemento
TEXTO DB    "Final", 13, 0Ah
NUM         DW      1000, -200, 400/60, 80h         ; 2 bytes por elemento
NUMMM       DD      200000h                         ; 4 bytes por elemento
            DB      6 dup (10h)                     ; 10h seis veces seguidas
            DB      10h dup("Pila")                 ; PilaPilaPilaPila .....
LETRA       DB      ?               ; reserva 1 byte sin asignar valor
LETRAS      DB      8 dup (?)
CEERCANO    DW      LETRA           ; almacena offset de LETRA
LEJANO      DD      LETRA           ; almacena offset y segmento de LETRA
```

Ejemplos Direccionamiento.

Direccion física= segmento*10h + offset.

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

Direccion física: 60006h = segmento*10h+offset.
Segmento=6000h
Offset=6h.

Dirección física 3678Fh

Segmento=3678h
Offset= Fh

Pero también
Segmento= 3000h
Offset= 678Fh

Ejemplos Direccionamiento.

Si tenemos la siguiente información en memoria:

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

El offset TABLA es 4.
Y los registros DS=4675; ES=4675; SI=0002; BX=0001, SS= 47FB, BP=0002

La información en AX será:

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) (por defecto, segmento SS)