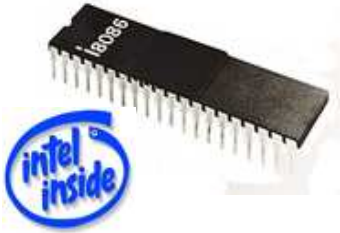


# LA FAMÍLIA i80x86

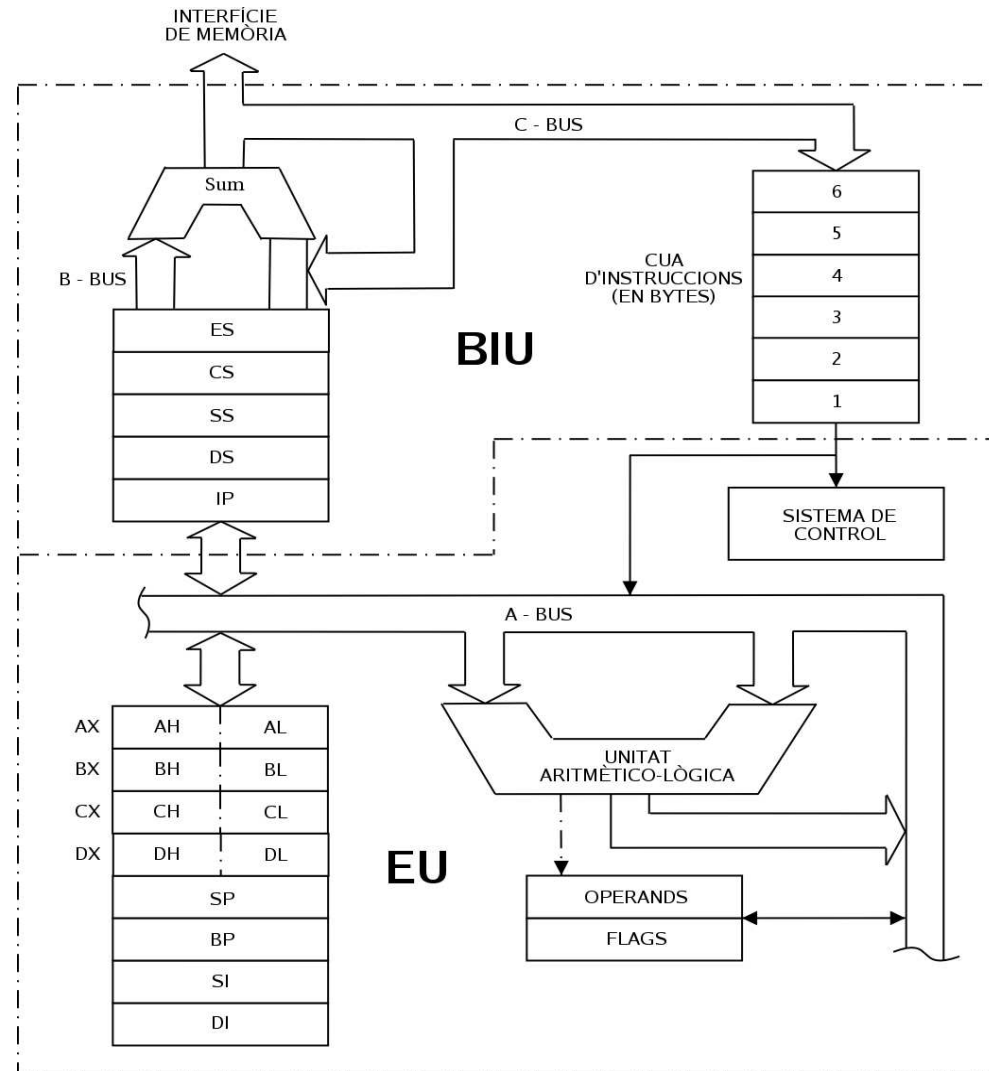
- Són microprocessadors del fabricant **Intel**
- Arquitectura CISC
- Little Endian
- Assemblador de 2 operands
- Accés segmentat (Segment:Offset) a memòria
- Evolució

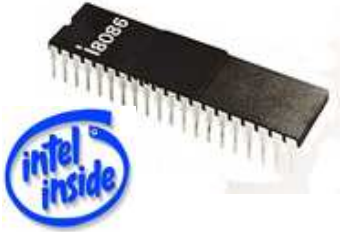
<i>Nom</i>	<i>Any</i>	<i>Bus de Dades</i>	<i>Espai d'adreces</i>	<i>Comentaris</i>
<b>8086</b>	<b>1978</b>	<b>16</b>	<b>1 Mb</b>	<b>Primer de 16 bits</b>
8088	1980	8	1 Mb	Bus extern 8 bits
<b>80186</b>	<b>1982</b>	<b>16</b>	<b>1 Mb</b>	<b>Microcontrolador 8086</b>
80188	1982	8	1 Mb	Microcontrolador 8088
<b>80286</b>	<b>1982</b>	<b>16</b>	<b>16 Mb</b>	<b>Suport al multiprocés</b>
<b>80386</b>	<b>1985</b>	<b>32</b>	<b>4 Gb</b>	<b>Primer 32 bits</b>
80386SX	1988	16	4 Gb	Bus extern 16 bits
80386SL	1990	16	4 Gb	80386SX de baix consum
<b>80486</b>	<b>1989</b>	<b>32</b>	<b>4 Gb</b>	<b>80486 + coprocessador i cache</b>
80486SX	1991	32	4 Gb	80486 sense coprocessador
<b>Pentium</b>	<b>1993</b>	<b>32</b>	<b>4 Gb</b>	<b>Nucli RISC amb segmentació</b>



# Intel 8086

- Registres
  - Dedicats
  - Solapats
  - 16 i 8 bits
  - Segment
  - Flags
- Cua d'instruccions
- Dos blocs: BIU i EU
- Tres grans busos





# Intel 8086 - Adreçament

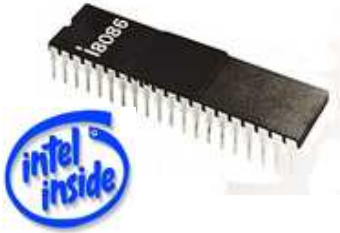
---

- Adreçament Segmentat

$$\text{@Final} = \text{Registre Segment} * 16 + \text{Offset}$$

- Exemple

CS * 16	1001011011110011	16 Bits
+ IP	0001001011001001	16 Bits
<hr/>		
	10011000000111111001	20 Bits



# Intel 8086 - Registres (I)

- Combinacions típiques de registres

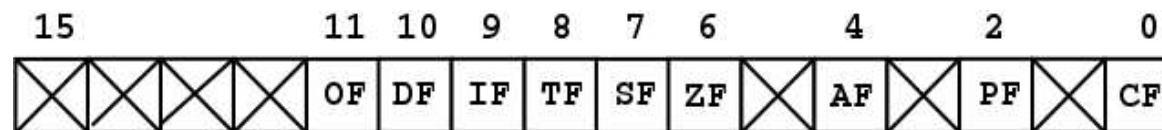
Punter de programa: CS:IP

Accés a dades de memòria: DS:SI ES:DI

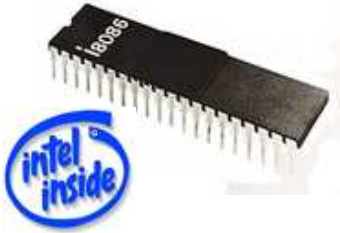
Punters a la pila: SS:SP SS:BP

- Registre de Flags

– Accés implícit a través d'instruccions (per exemple, de salt)



Flags: Overflow, Direction, Interrupt, Trap, Sign, Zero, Auxiliar, Parity i Carry



# Intel 8086 - Registres (II)

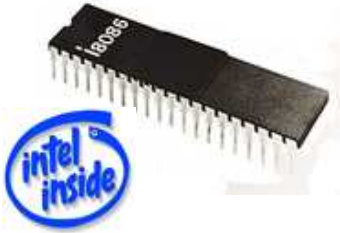
---

- Registres

- AX : **A**cumulator                      BX : **B**ase
- CX : **C**ounter                              DX : **D**ata
- SI : **S**ource **I**ndex              DI : **D**estination **I**ndex
- SP : **S**tack **P**ointer              BP : **B**ase **P**ointer
- IP : **I**nstruccion **P**ointer

- Registres de Segment

- CS : **C**ode **S**egment              DS : **D**ata **S**egment
- SS : **S**tack **S**egment              ES : **E**xtra **S**egment



# Intel 8086 - Registres (III)

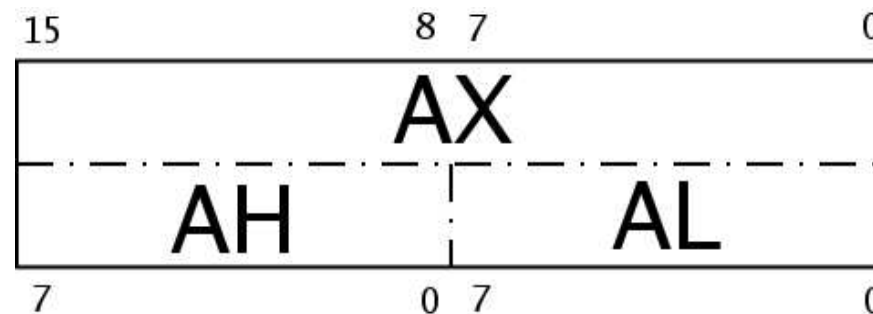
- Els registres de 8 bits són solapats

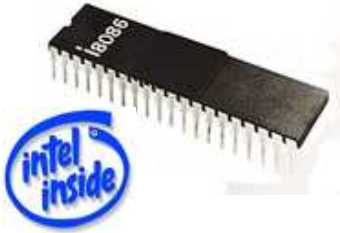
AX = AH : AL

BX = BH : BL

CX = CH : CL

DX = DH : DL

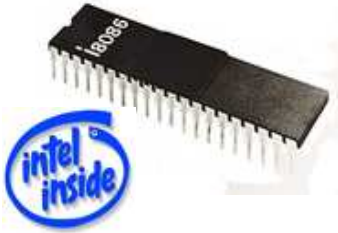




# Intel 8086 - Instruccions (I)

---

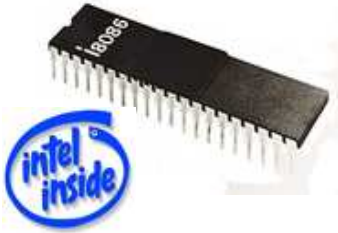
- Instruccions de Moviment
  - De dades (MOV, XCHG)
  - De pila (PUSH, POP)
  - Conversió de tipus (CBW)
- Instruccions Aritmètiques
  - Suma i diferència (ADD, ADC, INC, SUB, SBB, DEC)
  - Comparació (CMP, NEG)
  - Productes i divisions (MUL, IMUL, DIV, IDIV)



# Intel 8086 - Instruccions (II)

---

- Instruccions Lògiques
  - Booleanes (AND, OR, XOR, NOT)
  - Desplaçament i Rotació (SAL, SHL, SAR, SHR, ROL, ROR, RCL, RCR)
  - De Test (TEST)
- Instruccions de Control
  - Incondicional (JMP, CALL, RET, IRET)
  - Condicional (JA, JNBE, JAE, JNB, JG, JGE, ...)



# Intel 8086 - Instruccions (III)

---

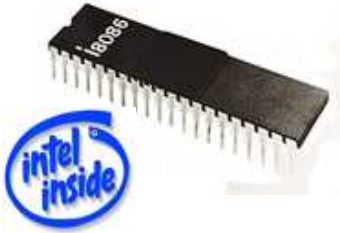
- Control de Flags
  - Carry (STC, CLC, CMC)
  - Interrupcions (STI, CLI)
  - Direcció (STD, CLD)
- Accés a Entrada / Sortida
  - Entrada (IN)
  - Sortida (OUT)
- Altres
  - NOP, LES, LDS, ...



# Intel 8086 - Modes d'adreçament (I)

---

- Operands
  - Cap operand: NOP
  - Un Operand: STC (Flag de carry)
  - Dos Operands: MOV AX, BX (Registres AX i BX)
- Operands explícits o implícits
  - MUL BX (  $DX:AX = \mathbf{AX} * BX$  )
  - CWD ( Expandir el signe de AX a DX:AX )
- Tipus d'operands (instruccions de dos operands)
  - Registre - Registre, Registre - Memòria, Memòria - Registre, Registre - Immediat, Memòria - Immediat



# Intel 8086 - Modes d'adreçament (II)

---

- Immediat
  - MOV AH, 0
  - MOV AL, -2
- Registre
  - MOV AX, BX
  - MOV DS, BX
- Registre Indirecte
  - LEA BX, Offset\_Variable
  - MOV AX, [BX]



# Intel 8086 - Modes d'adreçament (III)

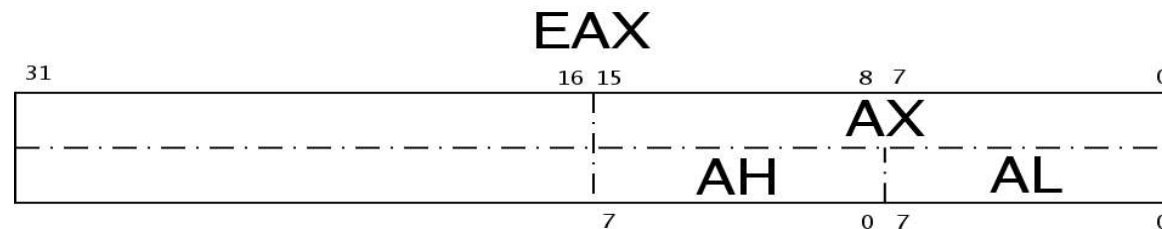
---

- Relatiu a Base
  - MOV AX, [BX + 4]
- Indexat Directe
  - MOV SI, 4
  - MOV AX, DS:[SI]
- Indexat amb Base
  - MOV BX, Offset\_Vector
  - XOR SI, SI
  - MOV AX, [BX+SI]



# Intel 80386 - Millores (I)

- Registres de 32 Bits
  - Tots els registres que no són de segment
    - EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP, EIP i EFLAGS



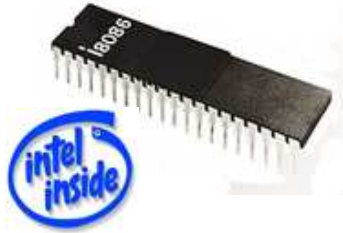
- Dos registres nous de segment (tots de 16 bits)
  - CS, DS, SS, ES, **FS** i **GS**
- 4 Gb d'espai físic d'adreces



# Intel 80386 - Millores (II)

---

- Suport Hardware per la Multitasca
  - **Mode Real:** La màquina opera com un 8086.
  - **Mode Virtual 86:** Mode multitasca on cada tasca veu un 8086 per ella sola.
  - **Mode Protegit:** Suport multitasca (protecció, memòria virtual, etc...)



# EXEMPLE

## TURBO ASSEMBLER (I)

---

- TASM (Borland) facilita la codificació
  - Poc críptic
  - Semblant als llenguatges d'alt nivell (C, Pascal)
- Estructura d'un programa

```
.model large
.386
    ; Definició de constants
    ; ...

.data          ; Definició de variables

.stack 100h   ; Definició de la pila (mida en bytes)

.code          ; Definició del codi
.startup
    ; Instruccions
.exit
end
```



# EXEMPLE

## TURBO ASSEMBLER (I)

- Declaració de constants

```
D    EQU    10           ; D = 10 en decimal
H1   EQU    10h         ; H1 = 16 decimal, 10 en hexadecimal
```

- Declaració de variables

```
Dada1  db  12h
Dada2  dw  ?
Dada3  dd  5 dup (0)
Dada4  db  1,2,3,4,5
Car     db  'c'
Cadena db  "Hola mon"
Punter dd  Dada1
```

- Declaració d'etiquetes i subrutines

```
Subrutina  proc
           ; Codi de la subrutina

Subrutina  endp

Loop:
           CALL Subrutina
           JNE  Loop
```



# Intel 80386 - Exemple

- Suma de tres enters

```
.model large
.386

.data                                ; Definició de variables
Nums    dd    12345678h, 9ABCDEF0h, 23h
Result  dd    ?

.stack 100h                          ; Definició de la pila (mida en bytes)

.code                                 ; Definició del codi
.startup

XOR     EAX, EAX                      ; Posa EAX a zero
LEA     ESI, Nums                     ; Carregar l'adreça efectiva
MOV     EAX, [ESI]                    ; Posar a EAX el primer valor
ADD     EAX, [ESI] + 4                 ; Sumar els 2 primers valors
ADD     EAX, [ESI] + 8                 ; Sumar el tercer valor
MOV     Result, EAX                   ; Guardar el resultat

.exit                                  ; Tornar al sistema operatiu
end
```