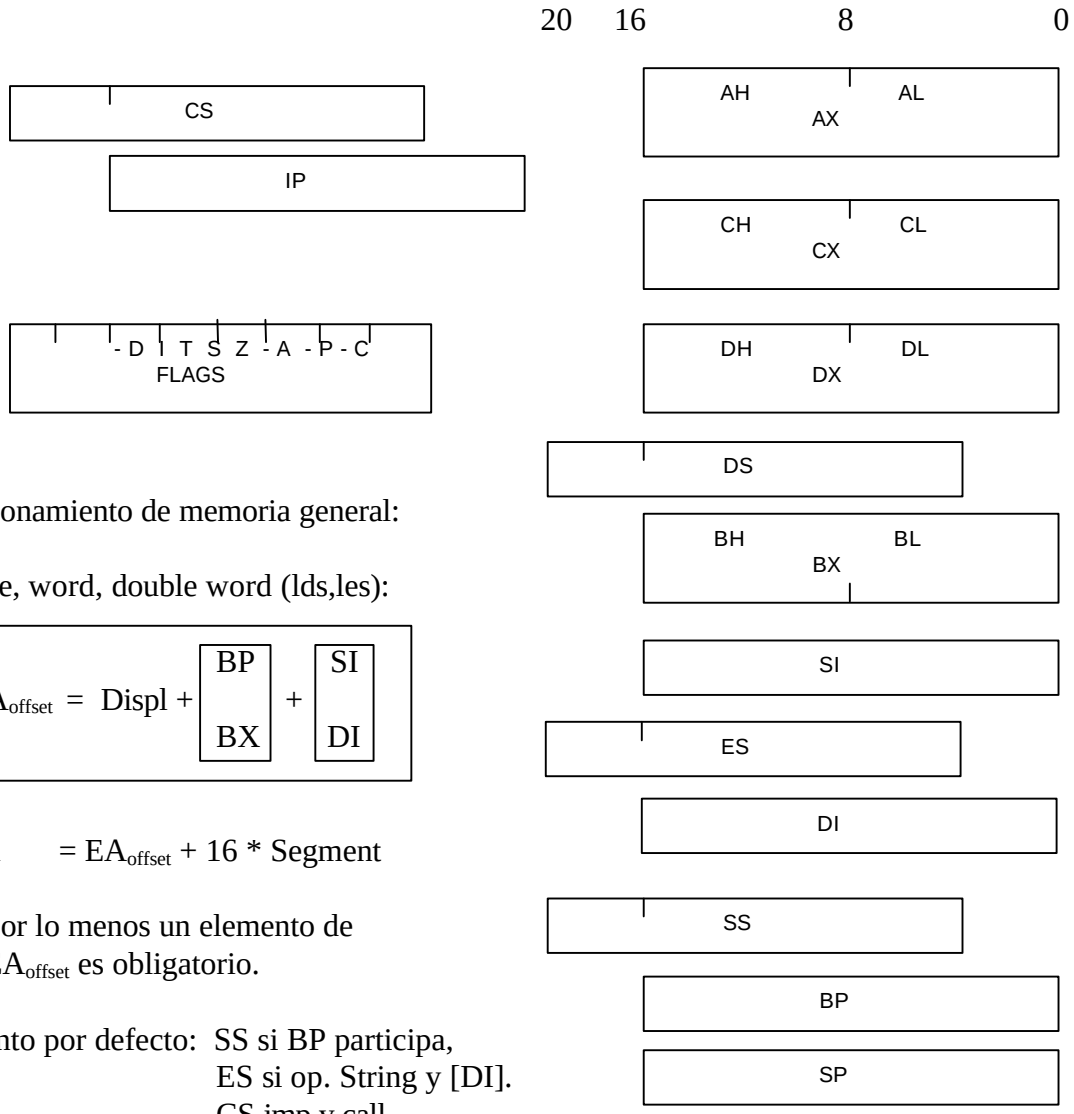
Intel 8086 modelo básico (primera parte)

Registros:

Uso general: AX, BX, CX, DX, SI, DI, BP.
Uso con direccionamento especial: SP, IP.
Registros de segmento: CS, SS, DS, ES.

Modelo de los registros:



Direccionamiento de memoria general:

byte, word, double word (lds,les):

$$EA_{offset} = Displ + \boxed{BP \;\; BX} + \boxed{SI \;\; DI}$$

$EA = EA_{offset} + 16 * Segment$

Por lo menos un elemento de EA_{offset} es obligatorio.

Segmento por defecto: SS si BP participa,
ES si op. String y [DI].
CS jmp y call
DS todos los otros casos.

Instrucciones básicas (por frecuencia del uso):

Mnemonica	Ingles	Castellano		Flags
Movimiento de datos				
MOV	**A, DE**	Move	Mover	A <- DE
MOV	**As, Des**	Move	Mover	As <- DEs
XCHG	**REG, A**	Exchange	Reemplazar	REG <-> MEM
LEA	**reg16, M**	Load Efective Address		reg16<-EA
PUSH	**DATA16**	Push	Push	DATA16 -> [SS:SP], SP<-SP-2
POP	**DIR16**	Pop	Pop	SP<-SP+2, DIR16<- [SS:SP]
Aritmetica				
ADD	**A, DE**	Add	Añadir	A <- A+DE OSCZAP
ADC	**A, DE**	add with carry	Añadir	A <- A+DE+C OSCZAP
SUB	**A, DE**	subtract	Resta	A <- A-DE OSCZAP
SBB	**A, DE**	Subtact with borrow	Resta	A <- A-DE-C OSCZAP
CMP	**A, DE**	Compare	Compara	. <- A-DE OSCZAP
NEG	**A**	Negate	Negativo	A <- -A
INC	**A**	Increment	Incremento	A++
DEC	**A**	Decrement	Decremento	A--
Logica				
AND	**A, DE**	And	And	A <- A DE SZP/COA
OR	**A, DE**	Or	Or	A <- A DE SZP/COA
XOR	**A, DE**	Exclusive Or	XOr	A <- A DE SZP/COA
TEST	**A, DE**	Test And	Test	. <- A DE SZP/COA
NOT	**A**	Not	-	A <- -1 DE

Otras instrucciones aritméticas de uso frecuente:

MUL	**DE**	Multiply	Multiplicar	ACC <- ACC1 * SRC
IMUL	**DE**	Int. Multiply	Multiplicar	; con signo
DIV	**DE**	Divide	Dividir	ACC1 <- ACC / SRC ; ACC2 <- ACC % SRC
IDIV	**DE**	Divide	Dividir	; Con signo
Control				
JMP	**DEST**	Jump	Ir a DEST	IP <- DEST (near) o CS:IP <-DEST (far)
CALL	**DEST**	Call	Llamar a DEST	PUSH CS; (far) PUSH IP; JMP DEST
RET		Return	Volver	POP IP; POP CS (far);
RET	**imm**			SP <- SP+imm; POP CS (far); POP IP
Jcc	**DISPL8**	Jump if cc	Ir si cc	IP <- IP+DISPL8(con signo)

Sumario de Jcc:

Estimación de flags:

Cc= { N } +

C	Carry
Z	Zero
O	Overf
S	Sign
P	Parity (**PE, PO**)

Aritmética con números sin signo (Bellow/Above) CMP X, Y // Jcc dest:

Jcc = Jump if | **N** Not | **B** Below (X<Y) / **A** Above (X>Y) | **E** (or) equal

Aritmética con números sin signo (Greater/Less) CMP X,Y // Jcc dest:

Jcc = Jump if | **N** Not | **G** Greater X>Y / **L** Less X<Y | **E** (or) equal

Alias dentro de Jcc:

Sinónimos	Antónimos	Sinónimos	Antónimos
JP, JPE	JNP, JPO	JZ, JE	JNZ, JNE
JC, JB, JNAE	JNC, JNB, JAE	JG, JNLE	JNG, JLE
JA, JNBE	JNA, JBE	JL, JNGE	JNL, JGE

Leyenda:

Reg8: AH, AL, BH, BL, CH, CL, DH, DL;
Reg16 : AX, BX, CX, DX, SI, DI, BP, SP;
Seg : DS, ES, SS, CS
Imm8: 8 bits de data inmediatos
Imm16: 16 bits de data inmediatos
Mem8: direccionamiento general de un BYTE.
Mem16: direccionamiento general de un WORD.
Mem32: direccionamiento general de un DWORD.
Acc8: AL
Acc16: AX

	Acc	Acc1	Acc2
8 bits	AX	AL	AH
16 bits	DX:AX	AX	DX

Combinaciones validos de direccionamiento y ejemplos:

DEST	EA	Ejemplo
DISPL8	IP+DISPL8	JMP SHORT L1 // JMP L1 // JA L1 // JNGE MYLOOP
DISPL16	IP+DISPL16	JMP NEAR L1 // JMP L2
MEM16	WORD PTR [EA]	JMP [BX] // JMP NEAR CS:[BX+DI]
MEM32	DWORD PTR [EA]	JMP DWORD PTR [BX] // JMP DWORD PTR R

RET y CALL se compilan del ensamblador según la declaración de PROG. Los PROGs declarados como NEAR se llaman con NEAR y los declarados con FAR como FAR.

A	**DE**	Ejemplos ASM	
		I8	EQU 10h
		I16	EQU 222h
		DB	R DUP (20)
		DW	X
Reg8	reg8	ADD	BH,AL
		XOR	BL,BL
		MOV	BH,AL
Reg8	mem	AND	AL, R /
		OR	CL,BYTE PTR X
		MOV	AL, R
		MOV	CL,BYTE PTR X
Reg8	imm8	ADD	BL, I8
		SUB	CL, 231
		MOV	BL, I8
		MOV	CL, 231
Reg16	reg16	SUB	BX,SI
		XOR	AX,AX
		MOV	BX,SI
Reg16	mem	TEST	SI, X[BX]
		XOR	AX, WORD PTR R+3[SI]
		MOV	SI, X[BX]
		MOV	BX, WORD PTR R+3[SI+BX]
Reg16	imm16	CMP	BX, I16
		TEST	BX, I8
		SBB	CX, OFFSET R
		MOV	BX,I16
		MOV	BX,I8
		MOV	CX, OFFSET R

DEs **As**

Seg	reg16	MOV	DS,AX
reg16	seg	MOV	CX,ES
mem16	seg	MOV	X, ES // MOV WORD PTR R, DS
seg	mem16	MOV	ES,X // MOV ES,WORD PTR R[BX+SI]

DATA16 Ejemplo

Reg16		PUSH	AX
Seg		PUSH	ES // PUSH CS // POP DS
Mem16	PUSH	X // PUSH WORD PTR R+2	
Imm16		PUSH	234 // 80286 y superiores.

DIR16

reg16	POP	SI
mem16	POP	WORD PTR R+6 // POP X
seg	POP	DS

Referencia completa de Jcc

Comprobación de flags:

Jcc	Descripción	Condición
JC	Jump if carry	C
JNC	Jump if no carry	/C
JZ	Jump if zero	Z
JNZ	Jump if not zero	/Z
JS	Jump if sign	S
JNS	Jump if no sign	/S
JO	Jump if overflow	O
JNO	Jump if no Overflow	/O
JP	Jump if parity	P
JPE	Jump if parity even	P
JNP	Jump if no parity	/P
JPO	Jump if parity odd	/P

Jcc instrucciones para números sin signo (CMP A,B // Jcc DEST):
(Below/Above)

Jcc	Descripción		Condición		Flags
JA	Jump if above	A>B		/C./Z	
JNBE	Jump if not below or equal	A>B		/C./Z	
JAE	Jump if above or equal	A>=B		/C	
JNB	Jump if not below	A>=B		/C	
JE	Jump if equal		A==B		Z
JNE	Jump if not equal ()		A != B		/Z
JBE	Jump if below or equal	A<=B		C or Z	
JNA	Jump if not above	A<=B		C or Z	
JB	Jump if below		A<B		C
JNAE	Jump if not above or equal	A<B		C	

Jcc instrucciones para números con signo (CMP A, B // Jcc DEST):
(Greater / Less)

Jcc	Descripción		Condición		Flags
JG	Jump if greater	A>B		/S or Z	
JNLE	Jump if not less than or equal		A>B		/S or Z
JGE	Jump if greater than or equal	A>=B		/S	
JNL	Jump if not less than		A>=B		/S
JE	Jump if equal		A==B		Z
JNE	Jump if not equal	A !=B		/Z	
JLE	Jump if less than or equal	A<=B		S O or Z	
JNG	Jump if not greater than	A<=B		S O or Z	
JL	Jump if less than	A<B		S.O	
JNGE	Jump if not greater or equal		A<B		S.O