Diccionario de Ensamblador (resumen del capítulo 28 del Abel). FOC 2017

-***AAA.*** Corrige la suma de dos bytes ASCII en el AL. **F**: AF, CF (I: OF, PF, SF, ZF). **S:** AAA (nop). **O:** 00110111

-***AAD.*** Corrige el dividendo a un valor binario en AL para una división binaria posterior. **F:** PF, SF, ZF (I: AF, CR, OF). **S:** AAD(nop). **O:** |11010101|00001010|.

-***AAM.*** Corrige el producto de multiplicar dos valores BCD desempacados. Divide AL entre 10 y almacena cociente en AH y el residuo en AL. **F:** PF, SF, ZF (I: AF, CF, OF). **S:** AAM(nop). **O:** |11010100|00001010|.

-***AAS.*** Corrige la diferencia de dos bytes en el AL. Si los bits más a la derechas mayor que 9, o si CF es 1, resta 6 del AL y 1 el AH, enciende AD y CF. Si no, AF y CF se limpian. Siempre limpia los 4 bits más a la izq. del AL. **F:** AF, CF (I: OF, PF, SF, ZF). **S:** ASS(nop). **O:** 00111111.

-***ADC.*** Suma el contenido del 1° op. al 2° op, luego suma el 2° al 1°. **F:** AF, CF, Of, PF, SF, ZF. **S:** ADC{reg/mem}, {reg/mem/in}. **O:** r/m w r: |000100dw|mod reg r/m|despl|desph| in to ac: |0001010w|-data-|data if w=1| in to r/m: |100000sw|mod 010 r/m|-data-|data if sw = 01 | despl|desph|

-***ADD.*** Suma dos números. **F:** AF, CF, OF, PF, SF, ZF. **S:** {reg/mem}, {reg/mem/in}. **O:** r/m w r: |000000dw|mod reg r/m|despl|desph| in to ac: |0000010w|-data|data if w=1|in to r/m: |100000sw|mod 000 r/m|-data-|data if sw=01|

-***AND.*** Operación logica AND, bit a bit. **F:** CF (0), OF (0), PF, SF, ZF (I: AF). **S:** {reg/mem}, {reg/mem/in}. **O:** r/m w r: |00100dw|mod reg r/m|despl|desph| in to ac: |0010010w|-data-|data if w=1| in to r/m: |100000sw|mod 100 r/m|-data-|data if w=1|despl|desph||

-***BSF/BSR.*** Escanea un string. BSF de der a izq. y BSR de izq. a der. El 2° op contiene el string a escanear. Retorna la posición del bit en el 1° op. **F:** ZF. **S:** BSF/BSR {reg}, {reg/mem}.**O:** *BSF:* |00001111|1011110x| mod reg r/m|despl|desph| x=0 if BSF, x=1 if BSR.

-***BT/BTC/BTR/BTS:*** copia un bit especificado a CF. El 1° op contiene el string por probar, y el 2° su posición. **F:** CF. **S:** BT/BTC/BTR/BTS {reg/mem}, {reg/in}. **O:** in to r: |00001111 |10111010|mod *** r/m| r/m to r: |00001111| 10***010|mod reg r/m|despl|desph| *** BT= 100, BTC=11, BTR=110, BTS=101.

-***CALL.*** Llama a un procedimiento. Si es NEAR, coloca el IP en la pila. Si es FAR, coloca el CS en la pila y carga el puntero del intersegmento en la pila. **F:** none. **S:** CALL {reg/mem}. **O:** directo dentro del segmento: |11101000|despl|desph| indirecto dentro del segmento: |11111111|mod 010 r/m|despl|desph| intersegmento indirecto: |11111111|mod 010 r/m|despl|desph| inter-segmento directo: |10011010|offset-low|offset-high|seg-low|seg-high|

-***CBW.*** Convierte de byte a word. Duplica el bit del signo del AL a cada uno de los bits del AH. **F:** none. **S:** CBW (nop). **O:** 10011000. *Ver: CWD, CWDE, CDQ.*

-***CDQ.*** Convierte de doublword a quadword. Duplica el bit del signo del EAX a cada uno de los bits del EDX. **F:** none. **S:** CDQ (nop). **O:** 10011001. *Ver: CBW, CWDE, CWD.*

-***CLC.*** Limpia la CF. **F:** CF (en 0). **S:** CLC (nop). **O:** 11111000.

-***CLD.*** Limpia la DF. Provoca que las operaciones con strings como *MOVS* se procesen de izq. a der. **F:** DF (en 0). **S:** CLD (nop). **O:** 11111000. *Ver STD.*

-***CLI.*** Limpia la IF, de modo que deshabilita una interrupción externa. **F:** IF (en 0). **S:** CLI (nop). **O:** 11111010. *Ver STI.*

-***CMC.*** Complementa la CF. **F:** CF. **S:** CMC (nop). **O:** 11110101

-***CMP.*** Compara el contenido de dos datos. Resta el 2° al 1°, no guarda el resultado, pero afecta las flags. **F:** AF, CF, OF, PF, SF, ZF. **S:** CMP {reg/mem}, {reg/mem/in}. **O:** r/m w r: |001110sw|mod reg r/m |despl|desph| in to ac: |0011110w|-data-|data if w=1| in to r/g: |100000sw|mod 111 r/m|-data-|data if sw=0| despl|desph|

-***CMP/CMPBS/CMPSW/CMPSD.*** Compara un string. El DS: si registra la dirección del 1° op. y el ES: di registra la dirección del 2°. **F:** AF, CF, OF, PF, ZF. **S:** [REPnn] CMPS/CMPSB/CMPSW/CMPSD (nop). **O:** 1010011w.

-***CWD.*** Convierte de word a doubleword en el DX. Duplica el bit del signo de AX en cada uno de los bits del DX. **F:** none. **S:** CWD (nop). **O:** 10011001.

-***CWDE.*** Convierte de word a doubleword extendido en el EAX. Duplica el bit del signo de AX en cada uno de los bits del EAX. **F:** none. **S:** CWDE (nop). **O:** 110011000.

-***DAA.*** Corrige el resultado de sumar dos BCD empacados en el AL. **F:** AF, CF, PF, SF, ZF (I: OF). **S:** DAA (nop). **O:** 00100111

-***DAS.*** Corrige el resultado de restar dos BCD empacados en el AL. **F:** AF, CF, PF, SF, ZF (I: OF). **S:** DAS (nop). **O:** 00101111

-***DEC.*** Decrementa en un reg o mem. **F:** AF, OF, PF, SF, ZF. **S:** DEC {reg, mem}. **O:** Register|01001reg| Reg/mem |1111111w|mod 001 r/m|despl|desph|

-***DIV.*** Divide dos valores sin signo. Operaciones:

bits	1 OP	2 OP	Cociente	Res.
16	AX	8 reg/mem	AL	AH
32	DX:AX	16reg/mem	AX	DX
64	EDX:EAX	32reg/mem	EAX	EDX

F: AF, CF, OF, PF, SF. **S:** DIV {reg/mem} **O:** |111101w|mod 110 r/m|despl|desph|

-***ESC.*** Proporciona al procesador una instrucción y un operando para ejecución**:** none. **S:** ESC in, {reg/mem}. **O:** |11011xxx|mod xxx r/m|despl|desph|

-***HLT.*** Provoca que se detenga al procesador mientras esperar una interrupción. **F:** none. **S:** HLT (nop). **O:** 11110100

-***IDIV.*** Divide números con signo. El MSB es el signo, con 1=neg. Operaciones: *ver tabla de DIV.* **F:** *ver DIV*. **S:** IDIV {reg/mem}. **O:** |1111011w|mod 111 r/m|despl|desph|

-***IMUL.*** Multiplica dos números con signo. EL MSB es el signo, con1=neg. Operaciones: *ver tabla de DIV, se debe intercambiar los valores de la columna 1 op con coef, donde cociente se convierte en el producto.* **F:** CF, OF (I: AF, PF, SF, ZF). **S:** IMUL {reg/mem} **O:** |111011w|mod 101 r/m|despl|desph|

-***IN.*** Transfiere de un puerto de entrada un byte al AL o un word al AX. Usar DX, si #puerto>256. **F:** none. **S:** IN{AL/AX}, {#port/DX} **O:** VarPort: |1110110w| Fixed port: |1110010w|

-***INC.*** Incrementa en 1 un registro o memoria. **F:** AF, OF, PF, SF, ZF. **S:** INC {reg/mem} **O:** reg: |0100reg| reg/mem: |1111111w|mod 000 r/m|despl|desph|

-***INT.*** Interrumpe el procesamiento y transfiere el control a una de las 256 direcciones de interrupción. **F:** Limpia IF y TF. **S:** INT number. **O:** |1100110v|-type-| if v =0, type is 3

-***INTO.*** Provoca una interrupción si ocurre in overflow y ejecuta INT 04H. **F:** IF, TF. **S:** INTO (nop). **O:** 11001110

-***IRET/IRETD.*** Proporciona un reterno FAR de una rutina de interrupción. Deshace los pasos que hizo la interrupción original. **F:** todas. **S:** IRET/IRETD. **O:** 11001111

-***JA/JNBE.*** Salta a la instrucción indicada si CF es 0 y ZF es 0. **F:** todas. **S:** JAEJNBE label. **O:** |01110111|disp|

-***JAE/JNB.*** Salta a la instrucción indicada si CF es 0. **F:** todas. **S:** JAE/JNB label. **O:** |01110011|disp|

-***JB/JNAE.*** Salta a la instrucción indicada si CF es 1. **F:** todas. **S:** JAE/JNAE label. **O:** |01110010|disp|

-***JBE/JNA.*** Salta a la instrucción indicada si CF es 1 o AF es 1. **F:** todas. **S:** JBE/JNA label. **O:** |01110110|disp|

-***JC.*** *Ver JB/JNAE.*

-***JCXZ/JECXZ.*** Salta a la instrucción indicada si CX o ECX contiene cero es 1. **F:** todas. **S:** JCXZ/JECXZ label. **O:** |11100011|disp|

-***JE/JZ.*** Salta a la instrucción indicada si ZF es 1. **F:** todas. **S:** JE/JZ label. **O:** |01110100|disp|

-***JG/JNLE.*** Salta a la instrucción indicada si ZF es 0 y SF es igual a OF. **F:** todas. **S:** JG/JNLE label. **O:** |01111111|disp|

-***JGE/JNL.*** Salta a la instrucción indicada si SF es a OF. **F:** todas. **S:** JGE/JNL label. **O:** |01111101|disp|

-***JL/JNGE.*** Salta a la instrucción indicada si SF es igual a OF. **F:** todas. **S:** JL/JNGE label. **O:** |0111100|disp|

-***JLE/JNG.*** Salta a la instrucción indicada si ZF es 1 y SF no es igual que OF. **F:** todas. **S:** JLE/JNG label. **O:** |01111110|disp|

-***JMP.*** Incondicional. **F:** todas. **S:** JMP{reg/mem} label. **O:** direct w seg short: |1101011|disp| direct w segment: |11101001|despl|desph| indirect w segment: |11111111|mod 100 r/m|despl|desph| indirect intersegment: |11111111|mod 101 r/m|despl|desph| direct intersegment: |11101010|offset-low|offset-high| seg-low|seg-high|

-***JNC.*** *Ver JAE/JNB.*

-***JNE/JNZ.*** Salta a la instrucción indicada si ZF es 0. **F:** todas. **S:** JNE/JNZ label. **O:** |01110101|disp|

-***JNO.*** Salta a la instrucción indicada si OF es 0. **F:** todas. **S:** JNO label. **O:** |01110001|disp|

-***JNP/JPO.*** Salta a la instrucción indicada si PF es 0. **F:** todas. **S:** JNP/JPO label. **O:** |01111011|disp|

-***JNS.*** Salta a la instrucción indicada si SF es 0. **F:** todas. **S:** JNS label. **O:** |01111001|disp|

-***JO.*** Salta a la instrucción indicada si OF es 1. **F:** todas. **S:** JO label. **O:** |01110000|disp|

-***JP/JPE.*** Salta a la instrucción indicada si PF es 1. **F:** todas. **S:** JP/JPE label. **O:** |01111010|disp|

-***JS.*** Salta a la instrucción indicada si SF es 1. **F:** todas. **S:** JS label. **O:** |01111000|disp|

-***LAHF.*** Carga el bit más a la der. del registro de banderas. **F:** none. **S:** LAHF (nop). **O:** |10011111|

-***LDS/LED/LFS/LGS/LSS.*** Inicializa una dirección FAR y el desplazamiento de un dato de modo que las siguientes instrucciones puedan accesarlo. **F:** todas. **S:** LDS/LED/LFS/LGS/ LSS {reg}, {mem}. **O:** LDS |11000101| LES |11000100| LFS |00001111| 10110100| LGS |00001111|10110101| LSS |00001111|10110010|. C/u de los anteriores tiene al final: |mod reg r/m|despl|desph|

-***LEA.*** Carga una dirección FAR (offset) a un registro. **F:** none. **S:** LEA {reg}, {mem}. **O:** |1001101|

-***LES/LFS/LGS.*** *Ver LDS.*

-***LOCK.*** Previene que se cambie un ítem de información al mismo tiempo que lo hace e;

procesador. **F:** none. **S:** LOCK instruction. **O:** 1111 0000

-LODS/LODSB/LODSW/LODSD. Carga el registro acumulador con un valor de memoria. Se afectan los registros: AL, AX, EAX. **F:** none. **S:** LOSDB /LODSW/LODSD (nop) **O:** 1010110w

-LOOP/LOOPW/LOOPD. Ejecuta una rutina un número de veces n. El CX es el contador, debe contener un valor antes de comenzar el loop; CX se decremente en 1 en cada ciclo **F:** todas. **S:** LOOP label. **O:** |11100010|disp|

-LOOPE/LOOPZ. Igual que LOOP, solo que hace un ciclo si CX es nonzero y ZF es 1. **F:** todas. **S:** LOOPE/LOOPZ label. **O:** |111000001|disp|

-LOOPNE/LOOPNZ. Igual que LOOP, solo que hace un ciclo si CX es nonzero y ZF es 0. **F:** todas. **S:** LOOPNE/LOONPZ label. **O:** |11100000|disp|

-LSS. *Ver LDS.*

-MOV. Transfiere los datos del 2° operando al 1°. **F:** todas. **S:** MOV {reg/mem}, {reg/mem/in}. **O:** r/m to/from reg: |100010dw|mod reg r/m|despl|desph| in to r/m: |1100011w|mod 000 r/m|-data-|data if w=1|despl|desph| in to reg |1011w reg|-data-|data if w=1| m to ac |1010000w|addr-low|addr-high| ac to m |1010001w|addr-low|addr-high| r/m to seg reg |1000 1110|mod 0 sg r/m| (sg= seg reg) seg reg to r/m |100001100|mod 0 sg r/m| (sg=seg reg)

-MOVS/MOVSB/MOVSW/MOVSD. Mueve datos entre localizaciones de memoria. El primer operando se direcciona por el ES: DI, el segundo por el DS:SI. **F:** none. **S:** [REP] MOVS/MOVSB/MOVSW /MOVSD (nop). **O:** 1010010w

-MOVSX/MOVZX. Copia un operando fuente de 8 o 16 de bits a uno destino de 16 o 32 bits. MOVSX llena con el bit del signo a los dígitos de más a la izquerda, MOVZX llena con 0 bits. **F:** none. **S:** MOVSX/MOVZX {reg/mem}, {reg/mem,in} (nop). **O:** MOVSX |00001111|1011111w|mod reg r/m|despl|desph| MOVZX |00001111|1011011w|mod reg r/m|despl|desph|

-MUL. Multiplica dos valores sin signo. Operaciones: *ver tabla de IMUL* **F:** OF, OF (I: AF, PF, SF, ZF). **S:** MUL {reg, mem}. **O:** |1111 011w|mod 100 r/m|despl|desph|

-NEG. Obtiene el complemento de un valor binario. **F:** AF, CF, OF, PF, SF, ZF. **S:** NEG {reg/mem} **O:** |1111011w|mod 011 r/m|despl|desph|

-NOP. Ninguna operación. Se usa para borrar o insertar cod. maq. o retrasar la ejecución por propósitos de tiempo. NOP ejecuta una operación nula XCHG AX, AX. **F:** none. **S:** NOP (nop). **O:** 10010000

-NOT. Cambia los 0s por 1s y viceversa. **F:** none. **S:** NOT {reg/mem}. **O:** |1111011w|mod 010 r/m|despl|desph|

-OR. Realiza la operación lógica OR con los bits de dos operandos. **F:** CF (0), OF (0), PF, SF, ZF (I: AF). **S:** OR {reg/mem}, {reg/mem/in} **O:** r/m w r |000010dw|mod reg r/m|despl|desph| in to ac |0000110w|-data-|data if w=1| in to r/m |100000sw|mod 001 r/m|-data-|data if w=1|

-OUT. Transfiere un byte del AL o un Word del AX a un puerto de salida. El puerto es un operando numérico fijo o una variable en el DX. Usar DX si #port>256. **F:** none. **S:** {#port/DX}, {AL/AX}**O:** VarPort |111011w| Puerto fijo |1110011w|-port-|

-POP. Desapila un word o db anteriormente apilado a un destino (mem, reg, seg-menos CS-) SP apunta al Word actual en la cima de pila; POP tlo transfiere al destino e incrementa al SP en 2 o 4. **F:** none. **S:** POP {reg/mem}. **O:** reg |01011reg| seg reg |000 sg 11| (sg implica segment leg) r/m |10001111|mod 000 r/m|despl|desph|

-POPA/POPAD. Desapila las 8 word de más arriba de la pila al DI, SI, BP, SP, BX, DX, CX y AX, en ese orden e incrementa SP en 16. POPAD usa dw. **F:** none. **S:** POP/POPAD (nop) **O:** 01100001

-POPF. Desapila los 10 words más arriba de la pila al registro de banderas e incrementa en SP en 2. **F:** todas. **S:** POPF/POFD (nop). **O:** 10011101

-PUSH. Apila un word o dw en la pila. El SP apunta al Word actual en la cima de la pila. PUSH decrementa a SP en 2 o 4. **F**: none. **S:** PUSH {reg/mem} **O:** reg |01010 reg| seg reg |00 sg 110|(sg implica segment reg) r/m |11111111|mod 110 r/m|despl|desph|

-PUSHA/PUSHAD. Apila el AX, CX, DX, BX, SP, BP, SI y DI, en ese orden; decrementa al SP en 16. **F:** none. **S:** PUSH/PUSHAD (nop) **O:** 01100000

-PUSHF. Apila el contenido del registro de banderas en la pila. Decrementa el SP en 2. **F:** none. **S:** PUSHF (nop). **O:** 10011100

-RCL/RCR. Rota los bits a través de la CF. **F:** CF, OF. **S:** RCL/RCR {reg/mem}, {CL/in} **O:** RCL |110100cw|mod 010 r/m| (if c=0, shift is 1) RCR |110100cw|mod 011 r/m| (if c=1, shift is in CL)

-REP. Repite una operación de string n veces. N debe cargarse en el CX. Por cada ejecución de la instrucción, REP decrementa a CX en 1 y repite la instrucción hasta que CX=0. **F:** *depende la instrucción string*. **S:** REP str-instruc **O:** 11110010

-REPE/REPZ/REPNE/REPNZ. Igual que REP, solo que se hacen condicionales. REPE, REPZ, REPNE se usan opcionalmente en SCAS y SCMAS. REPE/REPZ la operación se repite mientras ZF=1 y CX no es 0. REPNW/REPNZ se repite mientras ZF es 0 y el CX no es 0. **F:** *depende la instrucción string*. **S:** REPE/REPZ/REPNE/REPNZ str-instruc. **O:** REPNE/REPNZ 11110010 REPE/REPZ 11110011

-RET/RETN/RETF. Retorna de un proc. previamente ingresado por un CALL near o far. Para near, se mueve el Word a la cima de la pila al IP e incrementa el SP en 2. Para far, mueve la palabra a la cima de pila al IP y CS, e incrementa el SP en 4. **F:** none. **S:** RET/RETN/RETF [pop-value] **O:** dentro del segment |11000011| dentro del seg con pop value |11000010| data-low|data-high| intersgement |11001011| interseg con pop value |11001010|data-low|data-high|

-ROL/ROR. Rota el operando en un registro o memoria **F:** CF, OF. **S:** ROL/ROR {reg/mem}, {CL, in} **O:** ROL |110100cw|mod 000 r/m| (if c=0 count=1) ROR |110100cw|mod 001 r/m| (if c=1 count is in CL)

-SAHF. Guarda los bits del AH en los bits más a la derecha del registro de banderas. **F:** AF, CF, PF, ZF. **S:** SAHF (nop) **O:** |10011110|

-SAL/SAR. Shift aritmético. Corre los bits del operando fuente. **F:** CF, OF, PF, ZF (I: AF). **S:** SAL/SAR {reg/mem}, {CL, in} **O:** SAL |110100cw|mod 100 r/m| (if c=0 count=1) SAR |110100cw|mod 111 r/m|(if c=1 count in CL)

-SBB. Resta. Acarrea un 1 bit desbordado al siguiente paso de la aritmética. Primero resta CF del 2° operando, y luego resta este al 1°. **F:** AF, CF, OF, PF, SF, ZF **S:** SBB {reg/mem}, {reg, mem, in} **O:** r/m w r |000110dw|mod reg r/m|despldesph| in from ac |0001110w|-data-|data if w=1| in from r/m |100000sw|mod 011 r/m|-data-|data if sw=01|despl| desph|

-SCAS/SCASB/SCASW/SCASD. Escanea un string en busca de un valor específico. SCASB carga el valor en el AL, SCASW lo carga en AX, SCASD lo cargar en EAX. El par ES:DI hace referencia al string en memoria por escanearse. **F:** AF, CF, OF, PF, SF, ZF. **S:** [REPnn] SCAS/SCASB/SCASW/SCASD **O:** 1010111w

-SETnn. Coloca un byte especificado basado en una condición. Incluye SET(N)E, SET(N)L, SET(N)C, SET(N)S, que son paralelas a los saltos condicionales. Si la condición probada es True, la operación coloca el byte operando a 1, sino 0. **F:** none. **S:** SETnn {reg/mem} **O:** |00001111|1001 cond|mod 000 r/m|

-SHL/SHR. *Ver SAL y SAR.* **F:** CF, OF, PF, SF, ZF (I: AF). **S:** SHL/SHR {reg/mem}, {CL, in} **O:** SHL |110100cw|mod 100 r/m|despl|desph|(if c=0, count =1) SHR |110100cw|mod 101 r/m|despl|desph| if c=1, count in CL)

-SHLD/SHRD. Hace shift de múltiples bits en un operando. 3 operandos: 1° es un registro o localización de memoria que contiene el valor por desplazar. El 2° es un registro que contiene los vslor por desplzar en el 1° operando. 3° es el CL o un inmediato que contiene el valor shift **F:** CF, OF, PF, SF, ZF (I: AF). **S:** SHLD/SHRD {reg, mem}, reg,{CL, in} **O:** |00001111|10100100|mod reg r/m|despl|desph|

-STC. Coloca la CF en 1. **F:** CF. **S:** STC (nop) **O:** 11111001

-STD. Coloca la DF en 1. Provoca que las operaciones string se hagan de der. a izq. **F:** DF. **S:** STD (nop) **O:** 11111101

-STI. Coloca la IF en 1. **F:** IF. **S:** STI (nop) **O:** 11111011

-STOS/STOSB/STOSW/STOSD. Guarda el contenido del acumulador en memoria. STOSB carga el valor en AL. STOSW carga el valor en AX. DTOSD carga el valor en EAX. EL par ES:DI hace referencia a la localización de memoria donde el valor es almacenado**:** none. **S:** [REP] STOS/STOSB/ STOSW/STOSD (nop) **O:** 1010101w.

-SUB. Resta dos valores binarios. **F:** AF, CF, OF, PF, SF, ZF. **S:** SUB {reg/mem}, {reg/mem/in} **O:** r/m w r: |001010dw|mod reg r/m|despl|desph| in from ac |0010110w|-data-|data if w=1| in from r/m |100000 sw|mod 101 r/m|-data-|data if sw=01|

-TEST. Prueba la operación lógica AND sin afectar el operando destino. Afecta las banderas. **F:** limpia CF y OF, afecta F, SF, ZF (I: AF). **S:** TEST {reg/mem}, {reg/mem/in} **O:** r/m and r |1000010w| mod reg r/m|despl|desph| in to ac |1010100w|-data-|data if w=1|in to r/m |1111011w|mod 000 r/m|-data-|data f w=1|despl|desph|

-WAIT. Permite al procesador principal permanecer en estado de espera hasta una interrupción externa ocurra, de modo que se sincronice con el coprocesador. **F:** none. **S:** WAIT (nop) **O:** 10011011

-XCHG. Intercambia datos entre dos operandos. **F:** none. **S:** XCHG {reg/mem}, {reg/mem} **O:** r w ac 10010 reg r/m w r |1000011w|mod reg r/m|despl |desph|

-XLAT/XLATB. Traduce bytes a un formato diferente. Se define una tabla, sse carga su dirección en el BX, y en AL se carga el valor por traducir. AL se usa como un offset dentro de la tabla. **F:** none. **S:** XLAT/XLATB [AL] **O:** 11010111

-XOR. Realiza el XOR lógico con dos operandos. **F:** CF (0), OF (0), PF, SF, ZF (I: AF) .**S:** XOR {reg/mem}, {reg/mem, in} **O:** r/m w r |001100dw|mod reg r/m|despl|desph| in to r/m |1000000w|mod 110 r/m|-data-|data if w=1|despl| desph| in to ac |0011010w|-data-|data if w=1|

Abreviauras:

F	Flags	**r/m w**	Reg/mem con
S	Source code	**in**	inmediato
O	Object code	**ac**	acumulador
despl	Disp-low	**desph**	Disp-high
seg	segmento	**I (*en F*)**	Indefinido