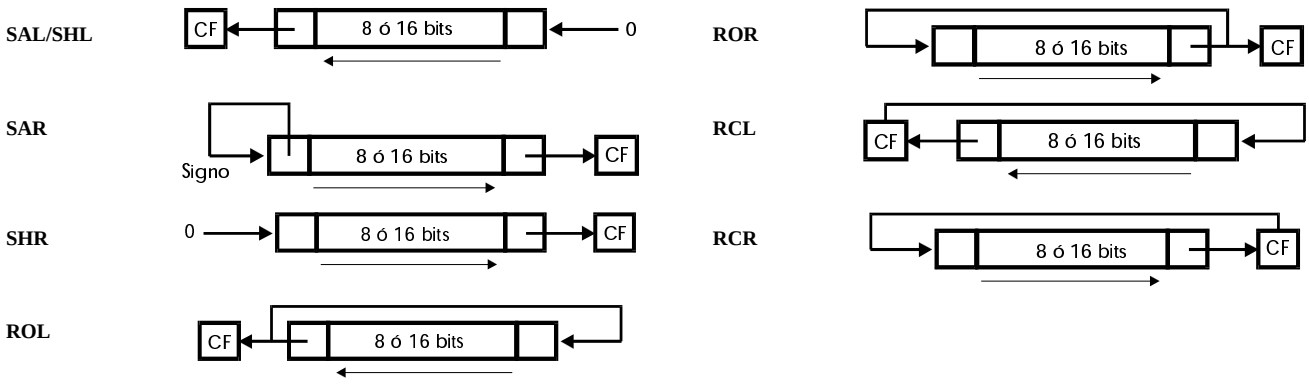
NE-MÓ-NIC.	OPERANDOS DESTINO,	FUENTE	FLAGS	DESCRIPCIÓN
			INSTRUCCIONES DE TRANSFERENCIA	
MOV	REG MEM/REG MEM/REG	MEM/REG REG NUM	No altera flags	COPIA EL OPERANDO FUENTE EN EL OPERANDO DESTINO.Ambos operandos deben ser del mismo tipo (BYTE O PALABRA). RESTRICCIONES: No se pueden mover datos entre dos elementos de memoria. No se puede usar MOV para transferir un valor inmediato a un registro de segmento. Ante cualquiera de estos dos casos, si es necesario, se usará un registro auxiliar. El registro CS no puede ser usado como destino.
PUSH	REG 16 / MEM 16		No altera flags	METE EL OPERANDO EN LA PILA. Decrementa en dos unidades el contenido del Puntero de Pila (SP) y después almacena la palabra especificada por el operando en la posición indicada por el puntero de pila (SS:SP). RESTRICCIONES: Solo se pueden guardar en la pila operandos tipo palabra, y antes hay que inicializar los registros SS y SP.
POP	REG 16 / MEM 16		No altera flags	SACA UNA PALABRA DE LA PILA. Transfiere la palabra de la pila, direccionada por SS:SP al operando destino (tipo palabra), y despues incrementa en dos unidades el registro SP. El contenido de la pila no se borra, sino que el puntero es modificado. RESTRICCIONES: El registro CS no puede ser especificado como destino.
XCHG	MEM/REG	REG	No altera flags	INTERCAMBIA EL CONTENIDO DE LOS DOS OPERANDOS. Pueden ser byte o palabra, pero los DOS del mismo tamaño RESTRICCIONES: No se puede intercambiar dos posiciones de memoria. Para ello, se usa un registro auxiliar. No pueden ser operandos registros de segmento
XLAT			No altera flags	TRADUCE. Convierte caracteres de un código en otro. Reemplaza un byte contenido en AL, por un byte de una tabla de conversión (256 bytes de long. max.) creada por el usuario. AL es el índice de la tabla. BX apunta al comienzo de la tabla. **AL= DS: BX+AL**
IN	AL / AX	PORT / DX	No altera flags	COPIA EL CONTENIDO DE UN PUERTO DE ENTRADA EN EL ACUMULADOR. Tamaño byte o palabra, se almacena en AL o AX respectivamente. Si la dirección del puerto está entre 0 y 255, se especifica directamente en la instrucción. En general, para direcciones entre 0h y FFFFh, se carga en DX la dirección (variable)
OUT	PORT / DX	AL / AX	No altera flags	TRANSFIERE UN BYTE O PALABRA (ubicado en AL o AX respect.) AL PUERTO DE SALIDA ESPECIFICADO. La dirección del puerto puede ser un valor fijo de un byte (de 0 a FFh), o un valor variable que puede ser modificado por el programa, almacenado en DX (de 0 a FFFFh).
LEA	REG 16	MEM	No altera flags	CARGA LA DIRECCIÓN EFECTIVA. Carga el desplazamiento (offset) de la dirección de memoria origen en el registro de 16 bits indicado como destino. RESTRICCIONES: No puede usarse como destino ningún registro de segmento. NOTA: (LEA=dirección, MOV=contenido de la dirección)
LDS	REG 16	MEM	No altera flags	CARGA EL SEGMENTO DE DATOS. Carga el registro de 16 bits especificado como destino con el contenido de la palabra almacenada en la dirección MEM. Las posiciones involucradas son: (byte bajo de REG16)=MEM, (byte alto de REG16)=MEM+1, (byte bajo de DS)=MEM+2, (byte alto de DS)=MEM+3 RESTRICCIONES: No se puede usar como destino un registro de segmento. NOTA: LDS se suele usar para inicializar SI y DS antes de usar instrucc. de cadena
LES	REG 16	MEM	No altera flags	CARGA EL SEGMENTO EXTRA. Carga el registro de 16 bits especificado como destino con el contenido de la palabra almacenada en la dirección MEM. Las posiciones involucradas son: (byte bajo de REG16)=MEM, (byte alto de REG16)=MEM+1, (byte bajo de ES)=MEM+2, (byte alto de ES)=MEM+3 RESTRICCIONES: No se puede usar como destino un registro de segmento. NOTA: LES se suele usar para inicializar DI y ES antes de usar instrucc. de cadena
PUSHF			No altera flags	GUARDA EN LA PILA LA PALABRA DE ESTADO. Decrementa en dos unidades el registro SP, y guarda en la dirección de pila indicada por SS:SP el registro de flags PSW
POPF			OF,DF,IF,TF,SF,ZF,AF,PF, CF.	SACA UNA PALABRA DE LA PILA AL REGISTRO PSW. Saca la palabra de la dirección más baja de la pila (SS:SP) y la copia sobre el registro de estado del procesador (PSW). Posteriormente incrementa en dos unidades el puntero de pila SP
LAHF			No altera flags	ALMACENA EL BYTE BAJO DE PSW (flags SF, ZF,AF,PF y CF) EN AH. Sirve para ejecutar correctamente sobre un 8088/86 programas escritos para un 8080 o un 8085.
SAHF			SF,ZF,AF,PF,CF.	ALMACENA EL REGISTRO AH EN EL BYTE MENOS SIGNIFICATIVO DE PSW. Sirve para ejecutar correctamente sobre un 8088/86 programas escritos para un 8080 o un 8085.
			INSTRUCCIONES ARITMETICAS	
ADD	REG MEM/REG MEM/REG	MEM/REG REG NUM	OF,SF,ZF,AF,PF,CF.	SUMAR.Suma el operando fuente al operando destino, almacenando el resultado en el operando destino. Los operandos pueden ser de tipo byte o palabra, pero AMBOS DEL MISMO TIPO. RESTRICCIONES: No se permite la suma de dos posiciones de memoria.
ADC	REG MEM/REG MEM/REG	MEM/REG REG NUM	OF,SF,ZF,AF,PF,CF.	SUMAR CON ACARREO. Suma el operando fuente al operando destino, almacenando el resultado en el operando destino. Los operandos pueden ser de tipo byte o palabra, pero AMBOS DEL MISMO TIPO. Si el flag de CARRY (CF) está activado, suma uno al resultado. RESTRICCIONES: No se permite la suma de dos posiciones de memoria.
INC	REG/MEM		OF,SF,ZF,AF,PF (no afecta al flag DF)	INCREMENTA EL DESTINO EN UNA UNIDAD. Suma uno a la dirección de memoria o registro especificado. Si dicho operando de destino era FFFFh pasará a valer 0000h después de la ejecución de INC, y no se pondrá a '1' el flag CF
AAA			AF,CF.(Se activan si el nº en AL antes de AAA es > 9)	AJUSTE ASCII PARA LA SUMA. Sólo opera sobre el registro AL, corrigiendo el resultado de una suma de dos números BCD desempaquetados (cada nº BCD viene representado por 8 bits) y convirtiéndolo en un número BCD desempaquetado.
DAA			SF,ZF,AF,PF,CF.	AJUSTE DECIMAL PARA LA SUMA. Corrige el resultado almacenado en AL correspondiente a la suma de dos números BCD empaquetados, convirtiéndolo en un número BCD empaquetado. Si el nibble (cuatro bits) de menor peso es superior a 9 o bien la bandera AF se activó durante la suma, la instrucción DAA le suma 6. Si el nuevo valor del nibble de mayor peso es ahora mayor que 9 o bien el flag CF está a '1', se suma 60h al registro AL.
SUB	REG MEM/REG MEM/REG	MEM/REG REG NUM	OF,SF,ZF,AF,PF,CF.	RESTAR. Resta el operando fuente del operando destino, almacenando el resultado en el operando destino. Los operandos pueden ser de tipo byte o palabra, pero AMBOS DEL MISMO TIPO. RESTRICCIONES: No se permite la resta de dos posiciones de memoria.
SBB	REG MEM/REG MEM/REG	MEM/REG REG NUM	OF,SF,ZF,AF,PF,CF.	RESTA CON BORROW. Resta el operando fuente del operando destino, almacenando el resultado en el operando destino. Los operandos pueden ser de tipo byte o palabra, pero AMBOS DEL MISMO TIPO. Si la bandera de arrastre está activada (CF=1), se resta 1 al resultado. RESTRICCIONES: No se permite la resta de dos posiciones de memoria.
DEC	REG / MEM		OF,SF,ZF,AF,PF (no afecta a CF)	DECREMENTA EL DESTINO EN UNA UNIDAD. Resta uno al operando destino. Dicho operando puede estar almacenado en una posición de memoria o en un registro, y su tamaño puede ser byte o palabra.
NEG	REG / MEM		OF,SF,ZF,AF,PF,CF	FORMAR EL COMPLEMENTO A 2. Invierte el signo del operando destino. El operando puede ser una posición de memoria o un registro.
CMP	REG MEM/REG MEM/REG	MEM/REG REG NUM	OF,DF,SF,ZF,AF,PF,CF.	COMPARAR DOS OPERANDOS. Lo hace mediante la resta del operando fuente del destino. El resultado NO es almacenado, y sólo se actualiza el contenido de los flags. Los operandos pueden ser de tipo byte o palabra, pero AMBOS DEL MISMO TIPO. RESTRICCIONES: No se permite la comparación entre dos posiciones de memoria.
AAS			AF,CF. (Se activan si el nº en AL antes de AAS es > 9)	AJUSTE ASCII PARA LA RESTA. Corrige el resultado en AL de la resta de dos números decimales desempaquetados, convirtiéndolo en un valor decimal desempaquetado.
DAS			SF,ZF,AF,PF,CF	AJUSTE DECIMAL PARA LA RESTA.Corrige el resultado almacenado en AL correspondiente a la resta de dos números BCD empaquetados, convirtiéndolo en un número BCD empaquetado. Si el nibble (4 bits) de menor peso es superior a 9 o bien el flag AF se activó durante la resta, la instrucción DAS le resta 6. Si el nuevo valor del nibble de mayor peso es ahora mayor que 9 o bie el flag CF está a 1, se resta 60h al registro AL.
MUL	REG / MEM		OF,CF(se activan cuando la mitad superior del resultado -en DX o AH- no sea 0)	MULTIPLICAR SIN SIGNO. Multipica un nº sin signo tamaño byte por un nº sin signo tamaño byte contenido en AL guardando el resultado en AX **(AX= AL * operando byte)** o multiplica un nº sin signo tamaño word por un nº sin signo tamaño word contenido en AX guardando el resultado en DX (palabra más significativa) y en AX (palabra menos significativa)**(DX AX = AX * operando word)**.
IMUL	REG / MEM		OF,CF(se activan cuando la mitad superior del resultado -en DX o AH- no sea 0)	MULTIPLICA CON SIGNO. Multipica un nº con signo tamaño byte por un nº con signo tamaño byte contenido en AL guardando el resultado en AX **(AX= AL * operando byte)** o multiplica un nº con signo tamaño word por un nº con signo tamaño word contenido en AX guardando el resultado en DX (palabra más significativa) y en AX (palabra menos significativa)**(DX AX = AX * operando word)**.
AAM			SF,ZF,PF	AJUSTE ASCII PARA LA MULTIPLICACIÓN. Corrige el resultado en AX del producto de dos números decimales desempaquetados,convirtiéndolo en un valor desempaquetado. **(AH=cociente de AL / 10)** y **(AL=resto de AL / 10)**
DIV	REG / MEM		Todos los flags quedan INDEFINIDOS.	DIVIDIR SIN SIGNO. Divide el contenido del acumulador y su extensión (AH AL si el operando es de tipo byte, o DX AX si el operando es de tipo word) entre el operando fuente. Hay dos posibilidades: Dividir 16 bits entre 8 bits **(Dividendo=AX, Divisor=fuente 8bits, Cociente=AL, Resto=AH)**o dividir 32 bits entre 16 bits **(Dividendo=DX AX, Divisor=operando 16bits, Cociente=AX, Resto=DX)**. Se genera una interrupcion de tipo 0 si el cociente supera **FFh** o **FFFFh** respectivamente.
IDIV	REG / MEM		Todos los flags INDEFINIDOS. Signo resto=signo dividendo	DIVIDIR CON SIGNO. Divide el contenido del acumulador y su extensión (AH AL si el operando es de tipo byte, o DX AX si el operando es de tipo word) entre el operando fuente. Hay dos posibilidades: Dividir 16 bits entre 8 bits **(Dividendo=AX, Divisor=fuente 8bits, Cociente=AL,Resto=AH)** o dividir 32 bits entre 16 bits **(Dividendo=DX AX, Divisor=operando 16bits, Cociente=AX, Resto=DX)**. Int tipo 0 si **7Fh<cociente<81h** ó **7FFFh<cociente<8001h** respectivamente.
AAD			SF,ZF,PF	AJUSTE ASCII PARA LA DIVISIÓN. Convierte dos dígitos en código BCD desempaquetado almacenados en AH y AL en el correspondiente nº binario. El resultado es almacenado en AL.
CBW			No altera flags	CONVIERTE BYTE EN WORD. Copia el bit de signo del byte almacenado en AL sobre todos los bits de AH. A esta operación se denomina *extensión del signo de AL*. Si es sin signo, se rellena los bits + significativos con 0. Si es con signo, se rellena los bits + significativos con el bit de mayor peso.
CWD			No altera flags	CONVIERTE WORD EN DOUBLEWORD. Copia el bit de signo de la palabra almacenada en AX sobre todos los bits de DX. A esta operación se denomina *extensión del signo de AX en DX*.
			INSTRUCCIONES LÓGICAS	
NOT	REG / MEM		No altera flags	NO LÓGICO. Forma el complemento a uno del operando, esto es, cambia los ceros por unos y los unos por ceros. El operando puede ser un registro o una posición de memoria de tamaño byte o word.
AND	REG MEM/REG MEM/REG	MEM/REG REG NUM	OF=0,SF,ZF,PF,CF=0 (AF indefinida)	Y LÓGICO. Realiza la operación lógica AND entre los operandos fuente y destino, realizada bit a bit y almacenada en el destino. RESTRICCIONES: No se puede realizar la operación entre dos posiciones de memoria.
OR	REG MEM/REG MEM/REG	MEM/REG REG NUM	OF=0,SF,ZF,PF,CF=0 (AF indefinida)	O LÓGICO. Realiza la operación OR inclusiva entre los operandos fuente y destino, realizada bit a bit y almacenada en el destino. RESTRICCIONES: No se puede realizar la operación entre dos posiciones de memoria.

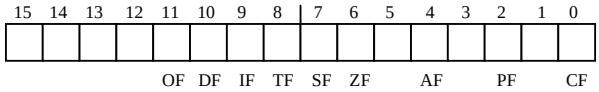
NE-MÓ-NIC.	OPERANDOS DESTINO,	FUENTE	FLAGS	DESCRIPCIÓN
XOR	REG MEM/REG MEM/REG	MEM/REG REG NUM	OF=0,SF,ZF,PF,CF=0 (AF indefinida)	O LÓGICO EXCLUSIVO. Realiza la operación lógica OR exclusiva entre los operandos fuente y destino, realizada bit a bit y almacenada en el destino. RESTRICCIONES: No se puede realizar la operación entre dos posiciones de memoria.
TEST	REG MEM/REG MEM/REG	MEM/REG REG NUM	OF=0,SF,ZF,PF,CF=0 (AF indefinida)	AND LÓGICA. Realiza la operación lógica AND entre los operandos fuente y destino, realizada bit a bit pero sin almacenar el destino. Se actualizan los flags. RESTRICCIONES: No se puede realizar la operación entre dos posiciones de memoria.
INSTRUCCIONES DE DESPLAZAMIENTO Y ROTACIÓN				
ROL	MEM / REG MEM / REG	1 CL	OF, CF	ROTAR A LA IZQUIERDA los bits del operando destino el número de veces especificado en el operando fuente (los bits se van moviendo una posición hacia la izquierda). Si el nº de veces es 1, se puede especificar directamente. Si no, hay que usar CL. CF copia en cada rotación el bit más significativo. (ver esquema de DESPLAZAMIENTOS)
RCL	MEM / REG MEM / REG	1 CL	OF, CF	ROTA A LA IZQUIERDA USANDO EL ACARREO. Rota hacia la izquierda los bits del operando destino y la bandera de acarreo un número de veces especificado en el operando fuente. Si el nº de veces es 1, se puede especificar directamente. Si no, hay que usar CL. Los bits se van moviendo una posición hacia la izquierda. El bit más significativo se almacena en CF y el contenido de CF pasa a ser el bit menos significativo. (ver esquema de DESPLAZAMIENTOS)
ROR	MEM / REG MEM / REG	1 CL	OF, CF	ROTAR A LA DERECHA los bits del operando destino el número de veces especificado en el operando fuente (los bits se van moviendo una posición hacia la derecha). Si el nº de veces es 1, se puede especificar directamente. Si no, hay que usar CL. CF copia en cada rotación el bit más significativo. (ver esquema de DESPLAZAMIENTOS)
RCR	MEM / REG MEM / REG	1 CL	OF, CF	ROTA A LA DERECHA USANDO EL ACARREO. Rota hacia la derecha los bits del operando destino y la bandera de acarreo un número de veces especificado en el operando fuente. Si el nº de veces es 1, se puede especificar directamente. Si no, hay que usar CL. Los bits se van moviendo una posición hacia la derecha. El bit menos significativo se almacena en CF y el contenido de CF pasa a ser el bit más significativo. (ver esquema de DESPLAZAMIENTOS)
SAL	MEM / REG MEM / REG	1 CL	OF,SF,ZF,PF,CF	DESPLAZAMIENTO ARITMÉTICO A LA IZQUIERDA. Desplaza hacia la izquierda los bits del operando destino el número de veces indicado por el operando fuente, siendo colocado un cero en el bit menos significativo en cada desplazamiento. Si el nº de veces es 1, se puede especificar directamente. Si no, hay que usar CL. (ver esquema de DESPLAZAMIENTOS). NOTA: SAL y SHL son la misma instrucción máquina. Actúan igual.
SHL	MEM / REG MEM / REG	1 CL	OF,SF,ZF,PF,CF	DESPLAZAMIENTO LÓGICO A LA IZQUIERDA. Desplaza hacia la izquierda los bits del operando destino el número de veces indicado por el operando fuente, siendo colocado un cero en el bit menos significativo en cada desplazamiento. Si el nº de veces es 1, se puede especificar directamente. Si no, hay que usar CL. (ver esquema de DESPLAZAMIENTOS). NOTA: SAL y SHL son la misma instrucción máquina. Actúan igual.
SAR	MEM / REG MEM / REG	1 CL	OF,SF,ZF,PF,CF	DESPLAZAMIENTO ARITMÉTICO A LA DERECHA. Desplaza hacia la derecha los bits del operando destino el número de veces indicado por el operando fuente, siendo colocado una copia del bit más significativo en el bit más significativo en cada desplazamiento. Así, el bit de signo inicial se mantiene. Si el nº de veces es 1, se puede especificar directamente. Si no, hay que usar CL. El bit que sale por la derecha va a CF. (ver esquema de DESPLAZAMIENTOS)
SHR	MEM / REG MEM / REG	1 CL	OF,SF,ZF,PF,CF	DESPLAZAMIENTO LÓGICO A LA DERECHA. Desplaza hacia la derecha los bits del operando destino el número de veces indicado por el operando fuente, siendo colocado un cero en el bit más significativo en cada desplazamiento. Así, el bit de signo inicial se mantiene. Si el nº de veces es 1, se puede especificar directamente. Si no, hay que usar CL. El bit que sale por la derecha (el menos significativo) va a CF. (ver esquema de DESPLAZAMIENTOS)
INSTRUCCIONES DE CADENA				
MOVSB			No altera flags	MUEVE CADENAS de BYTES. Transfiere un byte de una dirección de memoria dada por DS:SI a otra posición de memoria dada por ES:DI. Tras la transferencia, SI y DI son automáticamente actualizados para apuntar al siguiente elemento de la cadena . Si el flag DF es '0', SI y DI incrementan una unidad. Si el flag DF es '1', SI y DI decrementan una unidad. NOTA: Se puede usar REP para conseguir transferencias múltiples, guardando en CX el nº de transferencias a realizar.
MOVSW			No altera flags	MUEVE CADENAS de PALABRAS. Transfiere una palabra de una dirección de memoria dada por DS:SI a otra posición de memoria dada por ES:DI. Tras la transferencia, SI y DI son automáticamente actualizados para apuntar al siguiente elemento de la cadena . Si el flag DF es '0', SI y DI incrementan dos unidades. Si el flag DF es '1', SI y DI decrementan dos unidades. NOTA: usar REP para transferencias múltiples, guardando en CX el nº de transferencias a realizar.
CMPSB			OF, SF, ZF, AF, PF, CF	COMPARA CADENAS BYTE A BYTE. Compara la cadena ubicada en la dirección de memoria dada por DS:SI con otra cadena situada en la posición de memoria dada por ES:DI. No se modifica ninguna cadena, sólo los flags. Tras la comparación, SI y DI son automáticamente actualizados para apuntar al siguiente elemento de la cadena . Si el flag DF es '0', SI y DI incrementan una unidad. Si el flag DF es '1', SI y DI decrementan una unidad. NOTA: Se puede usar REP para conseguir comparaciones múltiples, guardando en CX el nº de comparaciones a realizar.
CMPSW			OF, SF, ZF, AF, PF, CF	COMPARA CADENAS WORD A WORD. Compara la cadena ubicada en la dirección de memoria dada por DS:SI con otra cadena situada en la posición de memoria dada por ES:DI. No se modifica ninguna cadena, sólo los flags. Tras la comparación, SI y DI son automáticamente actualizados para apuntar al siguiente elemento de la cadena . Si el flag DF es '0', SI y DI incrementan dos unidades. Si el flag DF es '1', SI y DI decrementan dos unidades. NOTA: Se puede usar REP para conseguir comparaciones múltiples, guardando en CX el nº de comparaciones a realizar.
SCAS SCASB SCASW			OF, SF, ZF, AF, PF, CF	EXPLORA UNA CADENA (DE BYTES O PALABRAS) COMPARANDO SUS ELEMENTOS CON EL ACUMULADOR. Compara la cadena ubicada en la dirección de memoria dada por ES:DI con el acumulador (AL si es byte, AX si es word). No se modifica la cadena, sólo los flags. Tras la comparación, DI es automáticamente actualizado para apuntar al siguiente elemento de la cadena . Si el flag DF es '0', DI incrementan y si el flag DF es '1', DI decrementa (una unidad si es byte, dos si es word). NOTA: Se puede usar REP para conseguir comparaciones múltiples, guardando en CX el nº de comparaciones a realizar. Al final, DI guardará la dirección siguiente a aquella en la que se encontró un elemento igual al acumulador.
LODS LODSB LODSW			No altera flags	CARGA CADENA (DE BYTES O PALABRAS) EN EL ACUMULADOR. Transfiere un byte o word de la cadena ubicada en la dirección de memoria dada por DS:SI a AL o AX respectivamente. Tras la transferencia, SI es automáticamente actualizado para apuntar al siguiente elemento de la cadena . Si el flag DF es '0', SI incrementan y si el flag DF es '1', SI decrementa (una unidad si es byte, dos si es word).
STOS STOSB STOSW			No altera flags	ALMACENA EL CONTENIDO DEL ACUMULADOR EN UNA CADENA. Transfiere el contenido del acumulador (AL o AX dependiendo de si la cadena es byte o word respect.) en la dirección de memoria dada por ES:DI. Tras la transferencia, DI es automáticamente actualizado para apuntar al siguiente elemento de la cadena . Si el flag DF es '0', DI incrementan y si el flag DF es '1', DI decrementa (una unidad si es byte, dos si es word). NOTA: Se puede usar REP para conseguir transferencias múltiples, guardando en CX el nº de transferencias a realizar.
REP	INSTRUCCIÓN DE CADENA		No altera flags	REPITE LA INSTRUCCIÓN DE CADENA SIGUIENTE. Se usa en combinación con el registro CX. Decrementa el contenido de CX en una unidad y se ejecuta la siguiente instrucción de cadena hasta que CX sea cero. ¡¡NOTA!!: SÓLO ACTÚAN SOBRE LA SIGUIENTE INSTRUCCIÓN DE CADENA.
REPE REPZ	INSTRUCCIÓN DE CADENA		No altera flags	REPITE LA INSTRUCCIÓN DE CADENA SIGUIENTE. REPE y REPZ son dos nemónicos para la misma instrucción máquina. Se suelen usar con las instrucciones CMPS y SCAS. Repiten la instrucción de cadena mientras las cadenas sean iguales, ZF='1' y/o CX distinto de '0' ¡¡NOTA!!: SÓLO ACTÚAN SOBRE LA SIGUIENTE INSTRUCCIÓN DE CADENA. PARA REPETIR UN BLOQUE DE INSTRUCCIONES SE USARÁ LOOP.
REPNE REPNZ	INSTRUCCIÓN DE CADENA		No altera flags	REPITE LA INSTRUCCIÓN DE CADENA SIGUIENTE. REPNE y REPNZ son dos nemónicos para la misma instrucción máquina. Se suelen usar con las instrucciones CMPS y SCAS. Repiten la instrucción de cadena mientras las cadenas sean DISTINTAS, ZF='0' y/o CX distinto de '0' . ¡¡NOTA!!: SÓLO ACTÚAN SOBRE LA SIGUIENTE INSTRUCCIÓN DE CADENA. PARA REPETIR UN BLOQUE DE INSTRUCCIONES SE USARÁ LOOP.
INSTRUCCIONES DE CONTROL DE PROGRAMA				
CALL	[ADR] REG:OFFSET REG (solo 127 posiciones arriba o abajo)		No altera flags	LLAMADA A SUBRUTINA:Transfiere la ejecución del programa principal a una subrutina. Se salva la dirección de la instrucción siguiente para continuar cuando termine la subrutina. Hay dos tipos de llamadas: NEAR y FAR. NEAR es para llamar a subrutinas en el mismo segmento de código (CALL NEAR decrementa SP en dos unidades y salva en la pila el IP correspondiente a la siguiente instrucción). FAR es para llamar a subrutinas en otro segmento (CALL FAR decrementa SP en dos unidades,salva en la pila el contenido del registro CS, decrementa SP otras dos unidades, y salva en la pila el IP de la siguiente instrucción (CS:IP que son 20 bits), y por último coloca en CS:IP la dirección de comienzo de la subrutina. RET termina la subrutina).
RET	(NUM) opcional		No altera flags	RETORNO DE SUBRUTINA. Devuelve el control al programa principal. Carga la dirección completa de la instrucción siguiente a la CALL que origino la llamada. Si la subrutina es NEAR, RET sustituye el contenido del registro IP por la palabra situada en la parte más baja de la pila (apuntada por SP). Tras esto, SP incrementa dos unidades. Si la subrutina es FAR, se sacan dos palabras de la pila. La primera se carga en el registro IP y la segunda en el CS. En total SP se incrementa en 4 unidades. Si se coloca un valor numérico tras RET, SP se incrementa de forma extra en la misma cantidad (esto es útil cuando se pasan parámetros a través de la pila)
JMP	DIRECCIÓN		No altera flags	SALTO INCONDICIONAL. Puede ser directo (lo que sigue a JMP es la dirección que se carga en IP) o indirecto (la dirección de salto está contenida en el registro o dirección que sigue a JMP). Dependiendo del segmento, es un salto NEAR o FAR.
JA/JNBE	DESPLAZAMIENTO		No altera flags	SALTO SI SUPERIOR. Salta si se cumple CF='0' y ZF='0'. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo.
JAE/JNB/ JNC	DESPLAZAMIENTO		No altera flags	SALTO SI SUPERIOR O IGUAL. Salta si se cumple CF='0' . El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo.
JB/JC JNAE	DESPLAZAMIENTO		No altera flags	SALTO SI INFERIOR. Salta si se cumple CF='1'. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo.
JNA/JBE	DESPLAZAMIENTO		No altera flags	SALTO SI NO SUPERIOR (inferior o igual). Salta si se cumple CF='0' . El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo.
JCXZ	DESPLAZAMIENTO		No altera flags	SALTO SI CX ES CERO. Salta si se cumple CX='0' . El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo.
JE/JZ	DESPLAZAMIENTO		No altera flags	SALTO SI IGUAL. Salta si se cumple ZF='1'. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo.

NE-MÓ-NIC.	OPERANDOS DESTINO, | FUENTE	FLAGS	DESCRIPCIÓN
JG JNLE	DESPLAZAMIENTO	No altera flags	SALTO SI MAYOR. Salta si se cumple ZF='0' y SF=OF. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. El numero 00000011 es mayor que 11111000 NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo.
JGE JNL	DESPLAZAMIENTO	No altera flags	SALTO SI MAYOR O IGUAL. Salta si se cumple SF=OF. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. El numero 00000011 es mayor que 11111000 NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo.
JL JNGE	DESPLAZAMIENTO	No altera flags	SALTO SI MENOR. Salta si se cumple SF distinto de OF. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. El numero 00000011 es mayor que 11111000 NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo.
JLE JNG	DESPLAZAMIENTO	No altera flags	SALTO SI MENOR O IGUAL . Salta si se cumple ZF='1' o SF distinto de OF. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. El numero 00000011 es mayor que 11111000 NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo.
JNE JNZ	DESPLAZAMIENTO	No altera flags	SALTO SI NO IGUAL. Salta si se cumple ZF='0'. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo.
JNO	DESPLAZAMIENTO	No altera flags	SALTO SI NO HAY DESBORDAMIENTO. Salta si se cumple OF='0'. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante.
JNP JPO	DESPLAZAMIENTO	No altera flags	SALTO SI NO HAY PARIDAD, O SI ES PARIDAD IMPAR. Salta si se cumple PF='0'. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante.
JNS	DESPLAZAMIENTO	No altera flags	SALTO SI POSITIVO. Salta si se cumple SF='0'. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante.
JO	DESPLAZAMIENTO	No altera flags	SALTO SI HAY DESBORDAMIENTO. Salta si se cumple OF='1'. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante.
JP JPE	DESPLAZAMIENTO	No altera flags	SALTO SI HAY PARIDAD. Salta si se cumple PF='1' . El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante.
JS	DESPLAZAMIENTO	No altera flags	SALTO SI SIGNO. Salta si se cumple SF='1' . El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante.
LOOP	DIRECCIÓN	No altera flags	BUCLE. Provoca la repetición de una serie de instruciones un número determinado de veces, especificado por el registro contador CX. Si CX es distinto de '0', salta a la dirección. Esa dirección debe estar entre 128 posiciones hacia atrás y 127 hacia delante.
LOOPE LOOPZ	DIRECCIÓN	No altera flags	BUCLE SI IGUAL O CERO. Provoca la repetición de una serie de instruciones un número determinado de veces, especificado por el registro contador CX. Si CX es distinto de '0' y ZF='1', salta a la dirección. Esa dirección debe estar entre 128 posiciones hacia atrás y 127 hacia delante.
LOOPNE LOOPNZ	DIRECCIÓN	No altera flags	BUCLE SI NO IGUAL O NO CERO. Provoca la repetición de una serie de instruciones un número determinado de veces, especificado por el registro contador CX. Si CX es distinto de '0' y ZF='0', salta a la dirección. Esa dirección debe estar entre 128 posiciones hacia atrás y 127 hacia delante.
INT	TIPO DE INTERRUPCIÓN	IF=0, TF=0	INTERRUPCIÓN. Realiza una interrupción por software. Abandona el curso normal del programa para ejecutar la rutina de atención de la interrupción, salvando antes el contenido de IF e IP en la pila.El vector de interrupción está ubicado en la posición de mem (0000:4* int) a la (0000:4* int+3)
INTO		IF=0, TF=0	INTERRUPCIÓN SI OVERFLOW.Genera la interrupcion interna tipo 4 en el caso de que la bandera de overflow OF='1'
IRET		TODAS LAS BANDERAS SON RESTAURADAS	RETORNO DE UNA INTERRUPCIÓN. Devuelve el control del programa a la instrucción siguiente a donde se habia interrumpido, sacando su dirección de la pila.Se restauran los valores de CS, IP e IF. Todas las banderas son restauradas con los valores que tenían antes de la interrupción.
INSTRUCCIONES DE MANIPULACIÓN DE FLAGS			
CLC		CF=0	PONE A CERO EL FLAG DE CARRY
CLD		DF=0	PONE A CERO EL FLAG DE DIRECCIÓN
CLI		IF	BORRA EL FLAG DE INTERRUPCIÓN. Desactiva el permiso de interrupción para las interrupciones enmascarables. Se seguirán tratando las interrupciones no enmascarables NMI y las software.
CMC		CF	COMPLEMENTAR EL FLAG DE ACARREO. Si CF='0' lo pone a '1', si CF='1' lo pone a '0'
STC		CF=1	PONER FLAG DE CARRY. Pone a '1' el flag de carry
STD		DF=1	PONER FLAG DE DIRECCIÓN. Pone a '1' el flag de dirección.
STI		IF	PONER FLAG DE INTERRUPCIÓN. Pone a uno el flag de interrupción, permitiendo las interrupciones enmascarables. No tiene efecto hasta que no se haya ejecutado la instrucción que sigue a la STI
INSTRUCCIONES DE CONTROL DEL MICROPROCESADOR			
ESC	COD. OP. | REG/MEM		ESCAPE. Se usa para pasar instrucciones a un coprocesador. El primer operando es un código de operación. El segundo operando indica dónde se encuentra el dato que se le va a pasar
HLT			PARADA DEL PROCESADOR. Cesa la actividad de búsqueda. Sólo saldra de este estado mediante una petición de interrupción externa enmascarable, no enmascarable, o una señal RESET. Útil para diagnosticar equipos.
LOCK			CIERRE DEL BUS. Usado en sistemas multiprocesador para compartir recursos, LOCK se usará delante de una instrucción crítica que deba tener prioridad absoluta en el control del bus.
NOP			NO OPERACIÓN. Sólo consume tres ciclos de reloj. Usado para ajustar retardos.
WAIT			ESPERA. El microprocesador entra en estado de letargo sin hacer nada. Se puede salir mediante la activación a nivel bajo de la entrada TEST, o con una petición de interrupción. Se usa para sincronizar.

ESQUEMA DE DESPLAZAMIENTOS:



PALABRA DE ESTADO PSW:



CODIFICACIÓN DE INSTRUCCIONES :

COD_OP	D	W	MOD	REG	R / M	DESPLAZA.	VALOR

D: 0 ..... REG = FUENTE
1 ..... REG = DESTINO

W: 0 ..... TAMAÑO BYTE
1 ..... TAMAÑO WORD

DIRECCIÓN DE REGISTRO	REGISTROS W=1	REGISTROS W=0
000	AX	AL
001	CX	CL
010	DX	DL
011	BX	BL
100	SP	AH
101	BP	CH
110	SI	DH
111	DI	BH

Dirección de Registro	Registro de Segmento
00	ES
01	CS
10	SS
11	DS

MOD / R / M	00	01	10	11 W=0	11 W=1
000 REG. DE SEGMENTO	(BX)+(SI) DS	(BX)+(SI)+D8 DS	(BX)+(SI)+D16 DS	AL	AX
001 REG. DE SEGMENTO	(BX)+(DI) DS	(BX)+(DI)+D8 DS	(BX)+(DI)+D16 DS	CL	CX
010 REG. DE SEGMENTO	(BP)+(SI) SS	(BP)+(SI)+D8 SS	(BP)+(SI)+D16 SS	DL	DX
011 REG. DE SEGMENTO	(BP)+(DI) SS	(BP)+(DI)+D8 SS	(BP)+(DI)+D16 SS	BL	BX
100 REG. DE SEGMENTO	(SI) DS	(SI)+D8 DS	(SI)+D16 DS	AH	SP
101 REG. DE SEGMENTO	(DI) DS	(DI)+D8 DS	(DI)+D16 DS	CH	BP
110 REG. DE SEGMENTO	D16 DS	(BP)+D8 DS	(BP)+D16 DS	DH	SI
111 REG. DE SEGMENTO	(BX) DS	(BX)+D8 DS	(BX)+D16 DS	BH	DI

USO DE LOS REGISTROS DE SEGMENTO:

TIPO DE OPERACIÓN	SEGMENTO POR DEFECTO	SEGMENTO OPCIONAL	OFFSET
Busqueda de instrucción	CS	NINGUNO	IP
Operación con la pila	SS	NINGUNO	SP
Operación con cadena fuente	DS	CS, ES, SS	SI
Operación con cadena destino	ES	NINGUNO	DI
Lectura / Escritura de datos	DS	CS, ES, SS	DIRECCIÓN EFECTIVA
BP Usado como registro base	SS	CS, DS, ES	DIRECCIÓN EFECTIVA