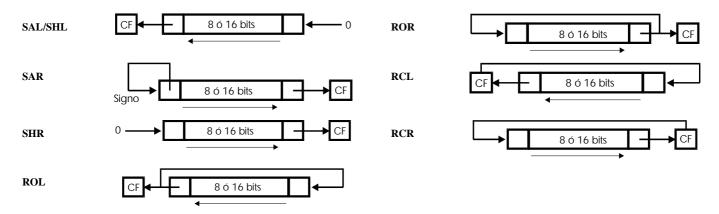
| NE- MÓ- | OPERANDOS | | FLAGS | DESCRIPCIÓN |
|------------|---------------------------|-----------------------|--|---|
| NIC. | DESTINO, FUENTE | | TLAGS | |
| MOV | REG | MEM/REG | No altera flage | INSTRUCCIONES DE TRANSFERENCIA COPIA EL OPERANDO FUENTE EN EL OPERANDO DESTINO Ambos operandos deben ser del mismo lipo (BYTE O PALABRA). |
| MOV | MEM/REG MEM/REG | REG NUM | No altera flags | 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 | MÊTE 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 | | 5 / 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 específica 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 específicado 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 específicado 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 flaqs 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. |
| | DEC | MEM/DEC | 05.05.75.45.05.05 | 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 lipo 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 | | /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. Solo 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 | | 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/REG | OF,SF,ZF,AF,PF,CF OF,DF,SF,ZF,AF,PF,CF. | FORMAR EL COMPLEMENTO A 2. Invierte el signo del operando destino. El operando puede ser una posición de memoría o un registro. COMPARAR DOS OPERANDOS. Lo hace mediante la resta del operando fuente del destino. El resultado NO es almacenado, y sólo se actualiza el |
| CMP | MEM/REG MEM/REG | REG NUM | AF,CF. (Se activan si el nº en | 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. AJUSTE ASCII PARA LA RESTA. Corrige el resultado en AL de la resta de dos números decimales desempaquelados, convirtiéndolo en un valor decimal |
| AAS | | | AL antes de AAS es > 9) SF,ZF,AF,PF,CF | desempaquetado. AJUSTE DECIMAL PARA LA RESTA. Corrige el resultado almacenado en AL correspondiente a la resta de dos números BCD empaquetados, |
| DAS | PEG | MEM | OF,CF(se activan cuando la | convirtiéndole 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. MULTIPLICAR SIN SIGNO. Multipica un nº sin signo tamaño byte por un nº sin signo tamaño byte contenido en AL quardando el resultado en AX |
| MUL | | MEM | mitad superior del resultado -en DX ó AH- no sea 0) | (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 ó 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 | BEC. | ALEM | 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 7fffh<cociente<8001h="" o="" respectivamente.<="" td=""></cociente<81h> |
| AAD | | | SF,ZF,PF | AJÚSTE 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. |
| NOT | REG | MEM | No altera flaca | INSTRUCCIONES LÓGICAS NO LÓGICO. Forma el complemento a uno del operando, esto es, cambia los ceros por unos y los unos por ceros. |
| NOT AND | REG MEM/REG | MEM/REG REG | No altera flags OF=0,SF,ZF,PF,CF=0 (AF indefinida) | To Coloco T unin a completient a unit de repeaturo, sea es canina a secreta por una y los unos por cetos. El operando puede ser un registro o una posición de memoria de tamaño byte o word. 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 | MEM/REG REG MEM/REG | NUM MEM/REG REG | 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. |
| | MEM/REG | NUM | (Al Illucilliud) | , |

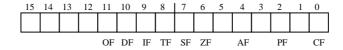
| NE- MÓ- NIC. | OPERA DESTINO, | | FLAGS | DESCRIPCIÓN |
|------------------------|----------------------------|--|---------------------------------------|--|
| XOR | REG MEM/REG MEM/REG | MEM/REG REG NUM | OF=0,SF,ZF,PF,CF=0 (AF indefinida) | O LÓGICO EXCLUSIVO. Realiza la operación lógica OR exclusiva 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. |
| _ | | | INST | R <mark>u</mark> cciones de desplazamiento y rotación |
| ROL | MEM / REG MEM / REG | 1 CL | OF, CF | ROTAR A LA IZOUIERDA 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 específicado en el operando fuente (los bits se van moviendo una posición hacia la derecha). Si el nº de veces es 1, se puede específicar 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 bils 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 específicar 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 consequir comparaciones múltiples, quardando 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, solo los flags. Tras la comparación, SI y DI son automáticamente actualizados para apuntar al siguiente elemento de la cadena. Si el flag DF es '1', SI y DI decrementan dos unidades. NOTA: Se puede usar REP para conseguir comparaciones múltiples, quardando 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 multiples, guardando en CX el n' de comparaciones a |
| LODS LODSB LODSW | | | No altera flags | realizar. Al final, D1 guardará la dirección siguiente a aquella en la que se encontró un elemento igual al acumulador. 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:S1 a AL o AX respectivamente. Tras la transferencia, S1 es automáticamente actualizado para apuntar al siguiente elemento de la cadena . Si el flag DF es '0', S1 incrementan y si el flag DF es '1', S1 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 | CA | ICCIÓN DE DENA ICCIÓN DE | 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. REPITE LA INSTRUCCIÓN DE CADENA SIGUIENTE. REPE y REPZ son dos nemónicos para la misma instrucción máquina. Se suelen usar con las |
| REPE REPZ | CA | DENA | No altera flags | Instrucciones CMPS y SCAS. Repiten la instrucción de cadena mientras las cadenas sean iguales, ZF='1' y/o CX distinto de '0' in NOTA!!: SÓLO ACTÚAN SOBRE LA SIGUIENTE INSTRUCCIÓN DE CADENA. PARA REPETIR UN BLOQUE DE INSTRUCCIONES SE USARÁ LOOP. |
| REPNE REPNZ | | ICCIÓN DE DENA | 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' . ijNOTA!!: SÓLO ACTÚAN SOBRE LA SIGUIENTE INSTRUCCIÓN DE CADENA. PARA REPETIR UN BLOQUE DE INSTRUCCIONES SE USARÁ LOOP. |
| | - | LDD1 | | NSTRUCCIONES DE CONTROL DE PROGRAMA |
| CALL | REG: F (solo 127 pos | ADR] OFFSET REG siciones arriba o | No altera flags | LLAMADA A SUBRUTINA Transflere la ejecución del programa principal a una subrutina. Se salva la dirección de la instrucción siguiente para continuar cuando termine la subrutina. Hay dos lipos 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. SEU la usua en SEU la estructión de contenido de la cubritatio. |
| RET | | oajo)) opcional | No altera flags | 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). RETORNO DE SUBRUTINA. Devuelve el control al programa principal. Carga la dirección completa de la instrucción siguiente a la CALL que origino la lamada. 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 | DIRE | ECCIÓN | 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 |
| JA/JNBE | DESPLA | ZAMIENTO | No altera flags | registro o dirección que sigue a JMP). Dependiendo del segmento, es un salto NEAR o FAR. 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. |
| JAE/JNB/ JNC | DESPLA | ZAMIENTO | No altera flags | NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo. 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 | DESPLA | ZAMIENTO | 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 | DESPLA | ZAMIENTO | 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 | DESPLA | ZAMIENTO | 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 | DESPLA | ZAMIENTO | 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- | OPERANDOS | | |
|------------------|----------------------|---------------------------------------|---|
| MÓ- NIC. | DESTINO, FUENTE | FLAGS | DESCRIPCIÓN |
| JG | 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 |
| JNLE | | · · | NOTA: Usaremos MAYOR Ý MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo. |
| JGE | 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 |
| JNL JL | DESPLAZAMIENTO | No altera flags | NOTA: Usaremos MAYOR Y MENOR con números con signo, y SUPERIOR e INFERIOR para números sin signo. SALTO SI MENOR. Salta si se cumple SF distinto de OF. El salto está comprendido entre 128 posiciones hacia atrás y 127 hacia delante. |
| JNGE | | ino altera liays | 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 | 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. |
| JPE | | J | |
| 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. |
| | | IN | ISTRUCCIONES 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 |
| | | INSTRUC | CCIONES 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. Solo 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 pelición 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 |] |
|--------|---|---|-------|-------|---------|------------|----------|---|
| COD_01 | - | | 11102 | 1 120 | 10, 111 | DEGI ELLE. | I "ILLON | ı |

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 | СН |
| 110 | SI | DH |
| 111 | DI | BH |

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

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

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 |

Funciones de la interrupción 21h.

La tabla siguiente representa las funciones más empleadas de la interrupción 21h.

| Función | Descripción |
|------------------------------|--|
| $\mathbf{AH} = 01\mathbf{h}$ | Entrada desde el teclado |
| | Esta función espera a que se teclee un carácter por teclado. Escribe el carácter en pantalla y |
| | devuelve el código ASCII en el registro AL |
| | Modifica AL con el código ASCII del carácter leído. |
| $\mathbf{AH} = 02\mathbf{h}$ | Salida a la pantalla |
| | Muestra un carácter en pantalla. Se debe guardar en DL el código ASCII del carácter que se |
| | desea sacar por pantalla. |
| | Devuelve en AL el código ASCII del carácter impreso |
| $\mathbf{AH} = 06\mathbf{H}$ | Envía carácter a la salida estándar sin comprobar ctrl-C. Se debe guardar en DL el código |
| | ASCII del carácter que se desea sacar por pantalla (distinto de 0FFh). |
| | Lee carácter de la entrada estándar sin esperar pulsación cuando DL=0FFh. |
| | Salida: si había algún carácter: ZF='0' y AL=código ASCII del carácter leído. Si no había |
| | carácter: ZF='1'. |
| $\mathbf{AH} = 08\mathbf{h}$ | Entrada desde el teclado sin reproducir el carácter por la pantalla |
| | Lee un carácter por pantalla pero no lo muestra por pantalla. |
| | Modifica AL con el código ASCII del carácter leído. |
| $\mathbf{AH} = 09\mathbf{h}$ | Muestra cadena |
| | Muestra por pantalla la cadena a la que apunta la pareja de registros DS:DX. El final de la |
| 1 | cadena se debe marcar con el carácter \$. |
| $\mathbf{AH} = 0\mathbf{Ah}$ | Lee cadena |
| | Lee una cadena desde el teclado. El buffer de almacenamiento debe estar apuntado por DS:DX. |
| | El primer byte del buffer debe contener el número máximo de caracteres a leer. En el segundo byte se devuelve el número real de caracteres pulsados (a excepción del retorno de carro) y a |
| | partir de segundo byte se encuentra la cadena de caracteres leida, finalizada con un retorno de |
| | carro (0Dh). |
| AH = 25h | Cargar el vector de interrupción |
| AII = 25II | Entradas: AL = tipo de interrupción; DS:DX apunta a la rutina de atención a la interrupción. |
| AH = 2Ch | Obtener la hora del sistema |
| 7 H1 = 2CH | Devuelve: |
| | CH = horas (desde 0 a 23) CL = minutos (desde 0 a 59) |
| | DH = segundos (desde 0 a 59) DL = centésimas de segundo (desde 0 a 99) |
| AH = 35h | Obtener vector de interrupción |
| | Obtiene la dirección de la rutina de atención de la interrupción especificada. |
| | Entradas: AL = tipo de interrupción. |
| | Devuelve: ES:BX = segmento y desplazamiento de la rutina de atención a la interrupción. |
| $\mathbf{AH} = 4\mathbf{Ch}$ | Sale al DOS. Devuelve el control al DOS, igual que la interrupción 20h. Devuelve en AL el |
| | código de retorno al DOS que se desee. |
| | Modifica AL con el valor que se desea devolver al DOS para ser usado con el IF |
| | ERRORLEVEL. |

Funciones de la interrupción 10h.

| Función | Descripción | | | | |
|------------------------------|---|--|--|--|--|
| $\mathbf{AH} = 00\mathbf{h}$ | Establece el modo de la pantalla | | | | |
| | AL = 0 40 x 25 blanco y negro alfanumérico. $AL = 5$ 320 x 200 blanco y negro gráfica. | | | | |
| | $AL = 1$ 40×25 color (16) alfanumérico. $AL = 6$ 640×200 blanco y negro gráfica. | | | | |
| | AL = 2 80 x 25 blanco y negro alfanumérico. $AL = 0Eh$ 640x200 color(16) gráfica. | | | | |
| | AL = 3 80 x 25 blanco y negro alfanumérico. AL=12h 640x480 color (16) gráfica. | | | | |
| | AL = 4 320 x 200 color (4) gráfica. AL=13h 320x200 color(256) gráfica. | | | | |
| $\mathbf{AH} = 01\mathbf{h}$ | Establecer las líneas del cursor | | | | |
| | CH (bits 0-4) Línea inicial. CL (bits 0-4) Línea final. | | | | |
| | CH (bits 5-7) deben ser 0. CL (bits 5-7) deben ser 0. | | | | |
| $\mathbf{AH} = 02\mathbf{h}$ | Posición del cursor | | | | |
| | DH = Fila (0-24) DL = Columna (0-79). | | | | |
| $\mathbf{AH} = 03\mathbf{h}$ | Leer posición del cursor | | | | |
| | BH=número de página (0 en modo gráfico) Devuelve: | | | | |
| | DH (fila) DL (columna) CH (bits 0-4) línea inicial CH(bits 5-7) 0 CL (bits 0-4) línea final y CL (bits 5-7) 0 | | | | |

| AH | Función | | | |
|------------------------------|---|---|--|--|
| $\mathbf{AH} = 06\mathbf{h}$ | Desplazamiento (scroll) hacia arriba | | | |
| | AL es el número de líneas. Si AL=0 se borra la ventana. | | | |
| | CH fila esquina superior izquierda. | DH fila esquina inferior derecha. BH relleno. | | |
| | CL columna esquina superior izquierda. | DL columna esquina inferior derecha. | | |
| $\mathbf{AH} = 07\mathbf{h}$ | Desplazamiento (scroll) hacia abajo | | | |
| | AL es el número de líneas. Si AL=0 se bo | orra la ventana. | | |
| | CH fila esquina superior izquierda. | DH fila esquina inferior derecha. BH relleno. | | |
| | CL columna esquina superior izquierda. | DL columna esquina inferior derecha. | | |
| $\mathbf{AH} = 08\mathbf{h}$ | Leer carácter y atributo de la posición | actual | | |
| | BH es el número de página. | | | |
| | Devuelve: AL el carácter leído AH atributo del carácter leído. | | | |
| $\mathbf{AH} = 09\mathbf{h}$ | Escribir el carácter y el atributo en la posición actual del cursor | | | |
| | BH es el número de página. | CX es el número de caracteres a escribir. | | |
| | BL es el atributo del carácter o el color. | AL es el carácter a escribir. | | |
| $\mathbf{AH} = 0\mathbf{Eh}$ | Escribir el carácter en la pantalla y avanzar el cursor | | | |
| | AL es el carácter a escribir. | | | |
| | BL es el color del carácter o su atributo. | | | |
| | BH es el número de la página. | | | |
| $\mathbf{AH} = 0$ Fh | Leer el estado actual de la pantalla | | | |
| | Devuelve: | | | |
| | AL el modo. | | | |
| | AH número de columnas de la pantalla. | | | |
| | BH número de página activa. | | | |

Funciones de la interrupción 16h.

| AH | Función | | | |
|------------------------------|--|--|--|--|
| $\mathbf{AH} = 00\mathbf{h}$ | Leer pulsación del teclado | | | |
| | Salida: AH = Código BIOS de rastreo; AL = Código ASCII de la tecla. | | | |
| | Sólo reconoce 84 teclas, elimina del buffer las del teclado extendido. Espera tecla. | | | |
| | Elimina tecla del buffer al leerla. | | | |
| $\mathbf{AH} = 01\mathbf{h}$ | Comprobar pulsación del teclado | | | |
| | Salida: si no hay tecla en el buffer ⇒ Flag de cero activado. | | | |
| | si hay tecla ⇒ Flag de cero desactivado; AH = Código BIOS de rastreo. | | | |
| | AL = Código ASCII de la tecla. | | | |
| | No espera tecla. No reconoce teclado extendido. No elimina tecla del buffer. | | | |
| $\mathbf{AH} = 02\mathbf{h}$ | Obtener estado del teclado | | | |
| | Salida: AL = Estado del teclado. | | | |
| $\mathbf{AH} = 10\mathbf{h}$ | Leer pulsación del teclado extendido | | | |
| | Salida: AH = Código BIOS de rastreo. | | | |
| | AL = Código ASCII de la tecla. | | | |
| | Espera tecla. Elimina tecla del buffer. | | | |
| $\mathbf{AH} = 11\mathbf{h}$ | Comprobar pulsación del teclado extendido | | | |
| | Salida: Si no hay tecla en el buffer ⇒ Flag de cero activado | | | |
| | Si hay tecla ⇒ Flag de cero desactivado | | | |
| | AH = Código BIOS de rastreo | | | |
| | AL = Código ASCII de la tecla | | | |
| | No elimina tecla del buffer. | | | |
| $\mathbf{AH} = 12\mathbf{h}$ | Obtener estado del teclado extendido | | | |
| | Salida: AH = Estado del teclado extendido. | | | |
| | AL = Estado del teclado. | | | |

Tabla 1. Operación de lectura en la PPI

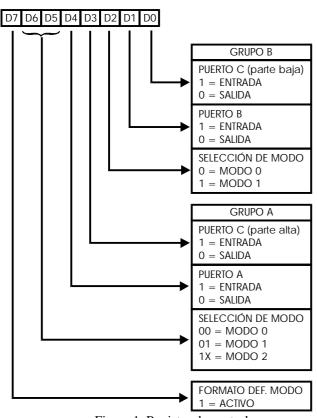
| A1 | A0 | RD# | WR# | CS# | Operación de entrada (Lectura) |
|----|----|-----|-----|-----|-------------------------------------|
| 0 | 0 | 0 | 1 | 0 | Puerto A \rightarrow Bus de Datos |
| 0 | 1 | 0 | 1 | 0 | Puerto $B \rightarrow Bus$ de Datos |
| 1 | 0 | 0 | 1 | 0 | Puerto $C \rightarrow Bus$ de Datos |
| 1 | 1 | 0 | 1 | 0 | Palabra de control → Bus de datos |

Tabla 2. Operaciones de escritura en la PPI

| A1 | A0 | RD# | WR# | CS# | Operación de salida (Escritura) | | |
|----|----|-----|-----|-----|----------------------------------|--|--|
| 0 | 0 | 1 | 0 | 0 | Bus de Datos→ Puerto A | | |
| 0 | 1 | 1 | 0 | 0 | Bus de Datos→ Puerto B | | |
| 1 | 0 | 1 | 0 | 0 | Bus de Datos→ Puerto C | | |
| 1 | 1 | 1 | 0 | 0 | Bus de Datos→ Palabra de control | | |

Tabla 3. Selección de la PPI

| A1 | A0 | RD# | WR# | CS# | Deshabilitación |
|----|----|-----|-----|-----|--|
| X | X | X | X | 1 | Bus de datos \rightarrow Three-State |
| X | X | 1 | 1 | 0 | Bus de datos \rightarrow Three-State |



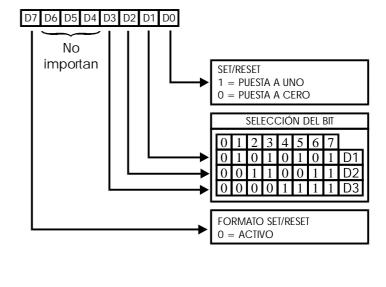


Figura 1. Registro de control

Figura 2. Palabra de control. Set/reset de los bits del puerto C

• FORMATO PUERTO C EN MODO1 ENTRADA (arriba) /SALIDA (abajo):

| I/O | I/O | IBFa | INTEa | INTRa | INTEb | IBFb | INTRb |
|------|-------|------|-------|-------|-------|------|-------|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| OBFa | INTEa | I/O | I/O | INTRa | INTEb | OBFb | INTRb |

• FORMATO DE LA PALABRA DE ESTADO O PUERTO C EN MODO 2:

| OBFa | INTE1 | IBFa | INTE2 | INTRa | X | X | X |
|------|-------|------|-------|-------|----|----|----|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

INTERFAZ SERIE USART 8251A

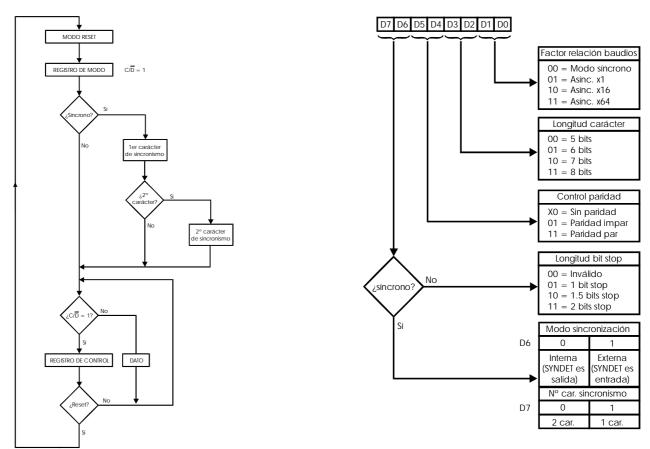
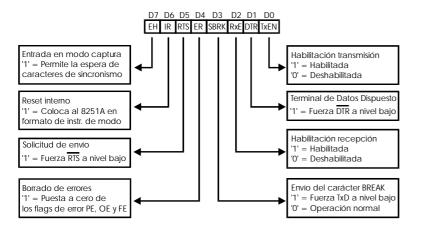


Figura 3. Secuencia de programación

Figura 4. Formato del registro de modo



| Registro | C/\overline{D} | Tipo |
|--------------------------------------|------------------|-----------|
| Modo | 1 | Escritura |
| Control | 1 | Escritura |
| Estado | 1 | Lectura |
| 1 ^{er} carácter sincronismo | 1 | Escritura |
| 2º carácter sincronismo | 1 | Escritura |
| Registro de transmisión | 0 | Escritura |
| Registro de recepción | 0 | Lectura |

Registros internos de la USART

Figura 5. Palabra de control

