MATERIA : LENGUAJE ENSAMBLADOR

INDICE

I.- DESCRIPCION GENERAL DE LA ARQUITECTURA DEL MICROPROCESADOR

MICROPROCESADOR. 8088------>8 BITS

8086------>16 BITS

VELOCIDAD SEGUN CASOS EL 8086 ES DE 2 A 4,000 VECES MAYOR QUE LOS DE 8 BITS.

TRATA LA INFORMACION EN MULTIPLOS DE 16 BITS EN VEZ DE 8 BITS.

TECNICA HMOS PARA TRATAMIENTO DE SILICIO 8086 => 70,000 O MAS TRANSISTORES POR CHIP.

CHIP MATEMATICO INTEL 8087 µP (PROCESADOR PARALELO) QUE REALIZA OPERACIONES MATEMATICAS.

INTERNAMENTE EL 8086 Y 8088 SON IGUALES.

EXTERNAMENTE EL 8086 UTILIZA UN BUS DE 8 BITS QUE LE PERMITE CONECTAR LAS MEMORIAS Y UNIDADES DE E/S NORMALES DE 8 BITS.

EXTERNAMENTE EL 8088 SE CONECTA A UN BUS DE 8 BITS Y EL 8086 A UN BUS DE 16 BITS (BUS DE DATOS).

COLA DE INSTRUCCION. GUARDA LOS BYTES DE LA INSTRUCCION CUANDO LA COMPUTADORA ESTA PREPARADA PARA OBTENER LA SIGUIENTE INSTRUCCION, NO NECESITA CARGAR LOS BYTES DESDE MEMORIA, SINO QUE TOMA LA INSTRUCCION SIGUIENTE DE LA COLA ANTES CITADA, ASI NO ESTA CONTINUAMENTE TENIENDO ACCESO A MEMORIA.

LA COLA DEL 8086 ES DE 6 BYTES.

LA COLA DEL 8088 ES DE 4 BYTES.

EL 8086 PUEDE ACCEDER A 1 MEGABYTE DE MEMORIA DE LECTURA-ESCRITURA (220 BYTES). SIN EMBARGO, UTILIZA UN ESQUEMA DE DIRECCIONAMIENTO DE MEMORIA LLAMADO DE **SEGMENTACION**, EN EL CUAL CIERTOS **REGISTROS DE SEGMENTACION** SUMINISTRAN UNA DIRECCION BASE QUE SE AÑADE AUTOMATICAMENTE A CADA DIRECCION DE 16 BITS QUE DEFINE EL USUARIO.

PARTE DE LA DIRECCION Y TODO EL BUS DE DATOS ESTAN MULTIPLEXADOS

EN 16 TERMINALES (PINS).

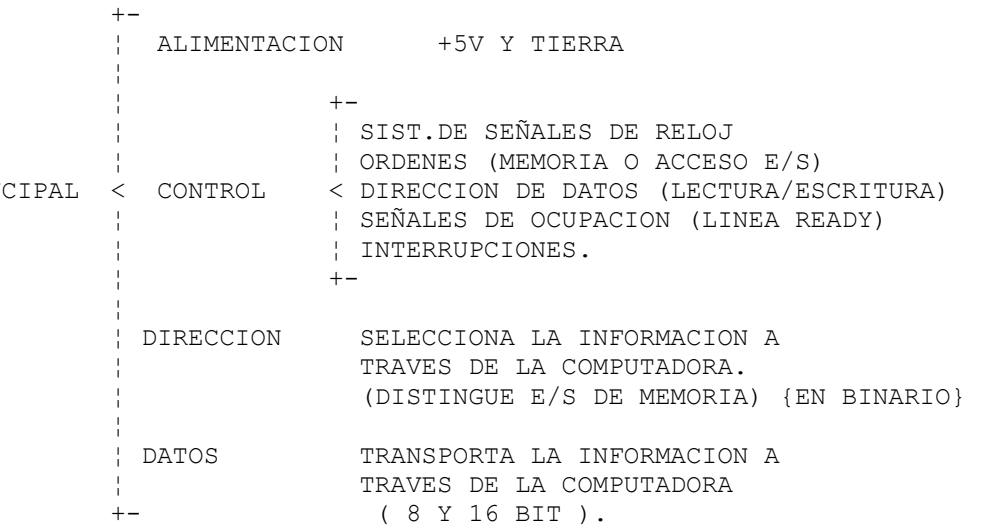
REGISTROS DE SEGMENTACION. (CS,DS,SS Y ES). A TRAVES DE ELLOS SE PUEDE DECIR A LA COMPUTADORA, SEPARADA Y DINAMICAMENTE LA DIRECCION DE UN PROGRAMA, DATO O PILA, DENTRO DE UN MEGABYTE DE MEMORIA.

HAY 4 REGISTROS MAS DE 16 BITS: EL PUNTERO DE PILA (SP), EL PUNTERO BASE (BP) Y DOS REGISTROS INDICE. EL REGISTRO INDICE FUENTE (SI) Y EL REGISTRO INDICE DESTINO (DI).

EL 8086 ES MUY RAPIDO. CON UN RELOJ DE 5 MHZ, TARDA 2µSEG.

EN CARGAR EL ACUMULADOR DESDE CUALESQUIER POSICION DE MEMORIA DENTRO DEL MBYTE. A 8MHZ SON NECESARIOS SOLO 1.25 µSEG.

1.1.- BUSES



EL 8086 Y EL 8088 SON PRACTICAMENTE IDENTICOS EXCEPTO POR EL TAMAÑO DE SU BUS DE DATOS EXTERNO.

1.2.- EU (EXECUTION UNIT) Y BIU (BUS INTERFASE UNIT)

LA IGUALDAD INTERNA Y DESIGUALDAD EXTERNA SE TRATA DIVIDIENDO CADA PROCESADOR 8086 Y 8088 EN 2 SUB-PROCESADORES:

UNIDAD DE EJECUCION (EU: EXECUTION UNIT)

UNIDAD DE INTERFAZ DE BUS (BIU: BUS INTERFACE UNIT)

EU => ES LA ENCARGADA DE REALIZAR TODAS LAS OPERACIONES.

BIU => ES LA ENCARGADA DE ACCEDER A DATOS E INSTRUCCIONES DEL MUNDO EXTERIOR.

LA VENTAJA DE ESTA APROXIMACION MODULAR ES EL AHORRO DE ESFUERZO NECESARIO PARA PRODUCIR EL CHIP 8088.

EL 8088 COMPARTE LOS 8 BITS MAS BAJOS DEL BUS DE DIRECCIONES CON LOS 8 BITS DEL BUS DE DATOS, DE MANERA QUE BASTAN 16 TERMINALES PARA DIRECCION Y DATOS, EN VEZ DE LOS 24 PREVISIBLES.

TANTO EL 8086 COMO EL 8088 TIENE UN BUS DE DIRECCIONAMIENTO DE 20 BITS, LO QUE PROVEE LA CAPACIDAD DE DIRECCIONAR 1 MBYTE DE MEMORIA = 220 = 1,048,576 BYTES = 1 MBYTE.

EL 8086 / 8088 TIENE OTRA MEMORIA SEPARADA, LLAMADA ESPACIO DE E/S QUE PUEDE CONSIDERARSE UNA MEMORIA EXTRA PARA DIRECCIONAR EN EL CUAL SE ENCUENTRAN LOS APARATOS DE E/S (CABLEADAS LAS DIRECCIONES). LOS APARATOS PUEDEN CONECTARSE A LA MEMORIA PRINCIPAL O AL ESPACIO E/S. ASIMISMO LA MEMORIA PUEDE CONECTARSE A LA MEMORIA PRINCIPAL O AL ESPACIO E/S.

EL JUEGO DE REGISTROS DEL 8086 Y DEL 8088 SON EXACTAMENTE IGUALES, CON 14 REGISTROS INTERNOS DE 16 BITS.

EL 8086/8088 TIENE 25 MODALIDADES DE DIRECCIONAMIENTO DIFERENTES LAS CUALES SON UN CONJUNTO DE REGLAS QUE ESPECIFICAN LA LOCALIZACION (POSICION) DE UN DATO USADO DURANTE LA EJECUCION DE

UNA INSTRUCCION.

EL 8086/8088 REQUIEREN UNA UNICA SEÑAL DE RELOJ (5 Y 8MHZ).

CON **MULTIPLEXEADO EN EL TIEMPO Y CODIFICACION** SE OPTIMIZA LAS 40 LINEAS DEL µP 8088/8086.

MULTIPLEXEADO EN EL TIEMPO. ES USAR EL MISMO CONJUNTO DE LINEAS, PERO EN PERIODOS DE TIEMPO DISTINTOS PARA ENVIAR CONJUNTOS DE SEÑALES DIFERENTES (DIRECCION/DATOS).

CODIFICACION. SIGNIFICA CONVERTIR UN CONJUNTO DE ESTADOS POSIBLES EN NUMEROS Y ENVIAR ESTOS NUMEROS POR UNAS POCAS LINEAS EN VEZ DE USAR UN LINEA PARA CADA ESTADO DIFERENTE (EJEM. 8 ESTADOS CON 3 LINEAS).

LA VELOCIDAD DEL 8086 NO ES EL DOBLE DEL 8088 POR:

1.- MUCHAS OPERACIONES NECESITAN TRANSFERIR DATOS EN 8 BITS.

2.- EL PROCESADOR TIENE QUE HACER BASTANTE MAS COSAS QUE UNICAMENTE TRANSFERIR DATOS.

3.- EL PROCESADOR INTERNO DUAL Y LA COLA DE INSTRUCCIONES DE ESTRUCTURA TUBULAR (PIPELINE).

PIPELINE. REALIZA AL MISMO TIEMPO LAS PRINCIPALES FUNCIONES INTERNAS DE TRANSFERENCIA DE DATOS Y BUSQUEDA DE INSTRUCCIONES.

PARA LOGRAR EL PIPELINE EL 8086/8088 CONSTAN DE 2 PROCESADORES INTERCONECTADOS EN LA MISMA PIEZA DEL SILICIO. UNA UNIDAD ESTA ENCARGADA DE BUSCAR INSTRUCCIONES (BIU) Y LA OTRA DE EJECUTARLAS (EU).

EL BIU UTILIZA EL METODO POR COLA PARA ALMACENAR NUEVAS INSTRUCCIONES HASTA QUE SE NECESITEN.

EL EU ES EL PROCESADOR PRINCIPAL Y SE ENCARGA DE CODIFICAR Y EJECUTAR TODAS LAS INSTRUCCIONES.

EL BIU ESTA ENCARGADO DE LOCALIZAR LAS INSTRUCCIONES Y DE

TRANSFERIR TODOS LOS DATOS ENTRE LOS REGISTROS Y EL MUNDO EXTERIOR (EL BIU DEL 8088 ES MAS COMPLEJO YA QUE DEBE TRANSFERIR DATOS ENTRE EL BUS DE DATOS INTERNO DE 16 BITS Y EL BUS DE DATOS EXTERNO DE 8 BITS).

CUANDO LA BIU LOCALIZA EN MEMORIA UN BYTE DE CODIGO DE MAQUINA LO COLOCA EN UNA LINEA DE ESPERA ESPECIAL LLAMADA COLA DE INSTRUCCIONES (PRIMERO QUE ENTRA, PRIMERO QUE SALE).

EL 8086 TIENE UNA LONGITUD DE 6 BYTES Y EL CODIGO DE MAQUINA SE ALMACENA EN MEMORIA DE 2 EN 2 BYTES.

EL 8088 TIENE UNA COLA DE INSTRUCCIONES DE SOLO 4 BYTES Y EL CODIGO DE MAQUINA SE GUARDA DE BYTE EN BYTE.

LA DIVISION DEL TRABAJO ENTRE LA EU Y LA BIU AHORRA UN TIEMPO CONSIDERABLE Y AYUDA A QUE EL RENDIMIENTO DEL 8088 DE 8 BITS SEA MAS COMPARABLE AL 8086 DE 16 BITS (ANALOGIA JEFE-SECRETARIA).

1.3.- REGISTROS

EL 8086/8088 CONTIENE 14 REGISTROS DE 16 BITS ALGUNOS PERTENECEN AL EU Y OTROS AL BIU (EU DIRECCIONA LA PARTE BAJA).

REGISTROS GENERALES DE 16 BITS (AX, BX , CX, DX) X=EXTENDIDO

EL REGISTRO AX ES EL ACUMULADOR DE 16 BITS. USANDOLO A VECES SE PROVOCA UN LENGUAJE MAQUINA CODIFICADO EN MUY POCOS BYTES (DIRECCIONAMIENTOS LO IMPLICAN). BYTES SUPERIOR E INFERIOR PUEDEN SER ACCESADOS SEPARADAMENTE.

EL REGISTRO BX PUEDE SER USADO COMO UN REGISTRO BASE EN EL CALCULO DE DIRECCIONES.

EL REGISTRO CX ES USADO COMO UN CONTADOR IMPICADO POR CIERTAS INSTRUCCIONES.

EJEM. LOOP UTILIZA CX PARA ALMACENAR EL NUMERO DE ITERACIONES DE EL LAZO.

DESPLAZAMIENTOS Y ROTACIONES (UTILIZAN CL).

EL REGISTRO DX ES USADO PARA RETENER LA DIRECCION DE I/O DURANTE CIERTAS OPERACIONES DE I/O . SE UTILIZA PARA ALMACENAR DATOS DE 16 BITS. PUEDE PENSARSE QUE ES UNA EXTENSION DEL REGISTRO AX PARA MULTIPLICACIONES Y DIVISIONES DE 16 BITS.

REGISTROS PUNTEROS Y DE INDICE (SP , BP , SI Y DI) Y

PUNTERO DE INSTRUCCIONES (IP)

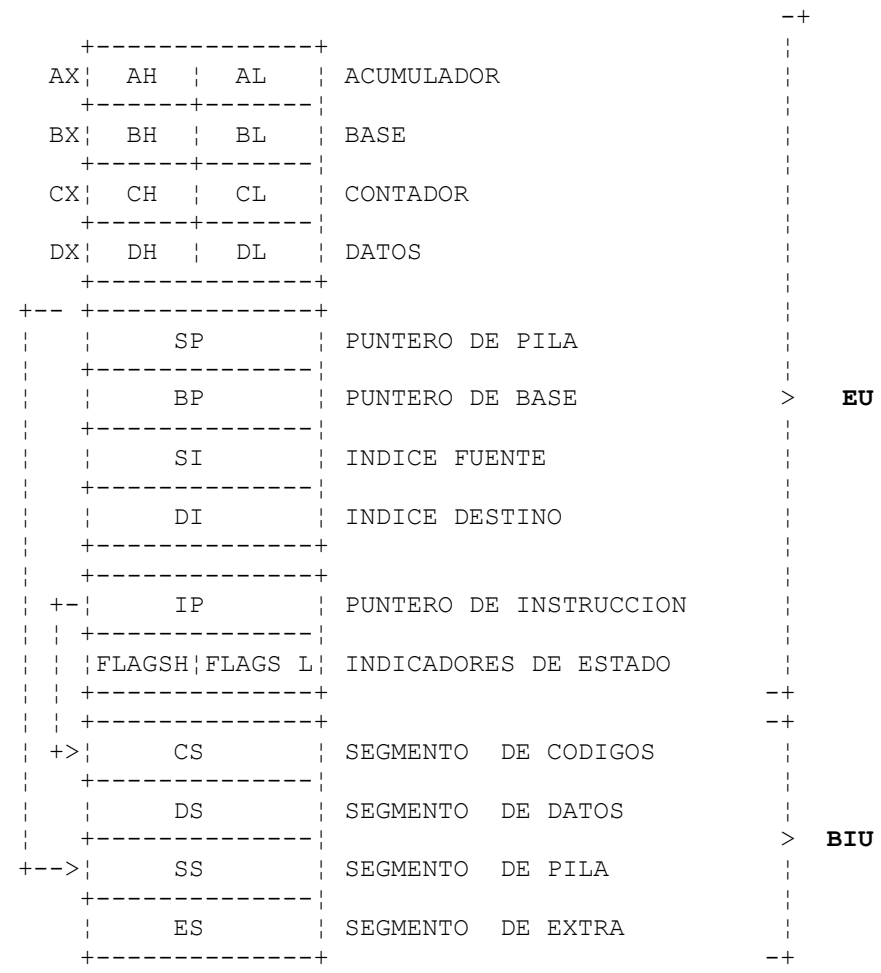
APUNTADOR DE STACK (SP)

PUNTERO DE BASE (BP)

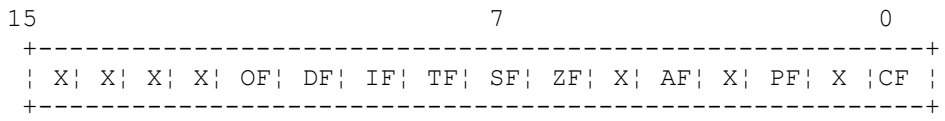
INDICE FUENTE (SI)

INDICE DESTINO (DI)

X = EXTENDED
H = HIGH
L = LOW



JUEGO DE REGISTROS DEL 8086 / 8088



X = REDUNDANTE

REGISTRO DE INDICADORES (BANDERAS) EN EL 8086 / 8088.

EL APUNTADOR DE INSTRUCCIONES (IP) Y EL REGISTRO SP SON ESCENCIALMENTE LOS CONTADORES DEL PROGRAMA Y LOS REGISTROS DE APUNTADOR DEL STACK (STACK O PILA PARA RETENER LA DIRECCION DE RETORNO DE UNA O VARIAS SUBRUTINAS) SE ENCARGAN DEL CONTROL DE FLUJO PROPIO DEL PROGRAMA.

LOS REGISTROS INDICE FUENTE (SI), INDICE DESTINO (DI) Y EL PUNTERO DE BASE (BP) SE UTILIZAN COMO PARTES DE LOS MODOS DE DIRECCIONAMIENTO O SEA, SE UTILIZAN PARA AYUDAR A LA LOCALIZACION DE DATOS.

REGISTRO INDICADOR DE ESTADO (PSW)

BANDERA DE ZERO	(ZF)	BANDERA DE INTERRUPCION	(IF)
BANDERA DE SIGNO	(SF)	BANDERA DE OVERFLOW (DESBORDAMIENTO)	(OF)
BANDERA DE PARIDAD	(PF)		
BANDERA DE ACARREO	(CF)	BANDERA DE DESVIO	(TF)
BANDERA AUXILIAR	(AF)	BANDERA DE DIRECCION	(DF)

CADA BANDERA OCUPA UN BIT ESPECIFICO EN EL REGISTRO.

REGISTROS DE SEGMENTOS. (CS, DS, SS Y ES). SUS CODIGOS REPRESENTAN RESPECTIVAMENTE A LOS SEGMENTOS DE CODIGO, DATOS STACK (PILA) Y EXTRA.

LA INSTRUCCION COMPLETA Y DIRECCION DEL STACK SON FORMADOS POR MEDIO DE SUMAR EL CONTENIDO DE IP Y SP A EL CONTENIDO DE LOS SEGMENTOS DE CODIGO(CS) Y SEGMENTOS DE STACK(SS) RESPECTIVAMENTE . EL DS (DATA SEGMENT) Y ES (EXTRA SEGMENT) FORMAN PARTE DE LA DIRECCION FINAL DE DATOS.

1.4.- MEMORIAS

DISPOSITIVO QUE SE ENCUENTRA CONECTADO AL BUS PRINCIPAL, DISTRIBUIDO EN UNA O MAS PLACAS DEL SISTEMA. SU MISION CONSISTE EN ALMACENAR, DE UNA FORMA BASICAMENTE TEMPORAL, DATOS Y PROGRAMAS. LA MEMORIA PUEDE VERSE COMO UNA COLECCION DE CELDAS INDIVIDUALES, CADA UNA DE LAS CUALES LLEVA ASOCIADO UN NUMERO AL QUE SE LE DA EL NOMBRE DE DIRECCION. TODAS LAS TRANSFERENCIAS DE INFORMACION DE CELDA A CELDA O ENTRE CELDAS Y CPU SE HACEN VIA EL BUS DE DATOS , UTILIZANDO EL BUS DE DE DIRECCIONES PARA SELECCIONAR LAS CELDAS Y EL BUS DE CONTROL PARA INICIAR Y REALIZAR EL PROCESO.

EN LA ACTUALIDAD, PRACTICAMENTE TODAS LAS MEMORIAS SON MEMORIAS DE SEMICONDUCTORES (CIRCUITOS DE SEMICONDUCTORES INTEGRADOS Y ENCAPSULADOS EN UN CHIP).

RAM = MEMORIA DE ACCESO ALEATORIO

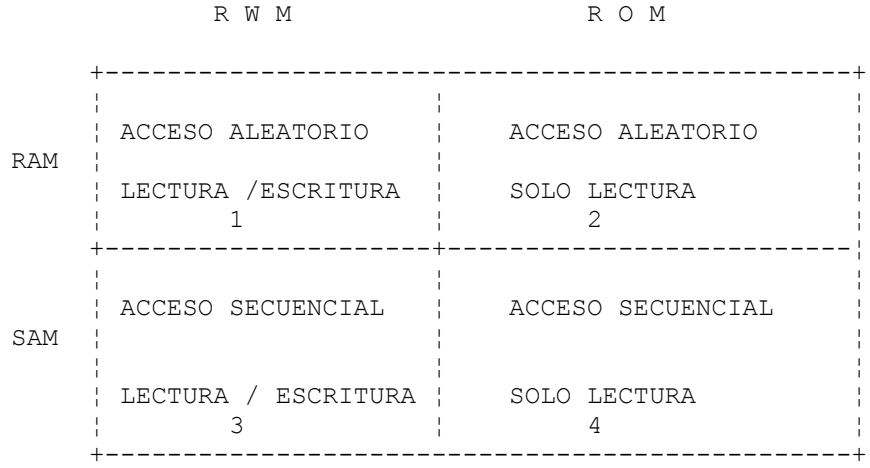
ROM = MEMORIA DE SOLO LECTURAS

SEGMENTACION. ES UN METODO DE ACCESO A MEMORIAS EN EL CUAL TODA DIRECCION SE COMPONE DE DOS CANTIDADES; UN IDENTIFICADOR DE SEGMENTO Y UN DESPLAZAMIENTO.

IDENTIFICADOR DE SEGMENTO. APUNTA A UN AREA GENERAL DE MEMORIA (EL SEGMENTO), MIENTRAS EL DESPLAZAMIENTO APUNTA A UNA DIRECCION DENTRO DE ESE SEGMENTO.

RWM = READ / WRITE MEMORY

SAM = SEQUENTIAL ACCESS MEMORY



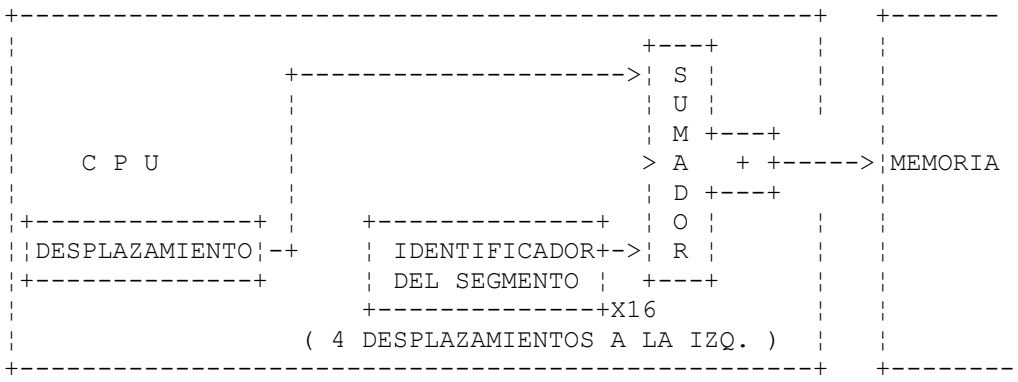
EJEMPLOS:

1.- MEMORIA PRINCIPAL QUE SE PUEDE MODIFICAR DIRECTAMENTE.

2.- MEMORIA PRINCIPAL QUE NO SE PUEDE MODIFICAR DIRECTAMENTE

3.- CINTA MAGNETICA

4.- CINTA DE PAPEL (SOLO PUEDE PERFORARSE UNA VEZ).



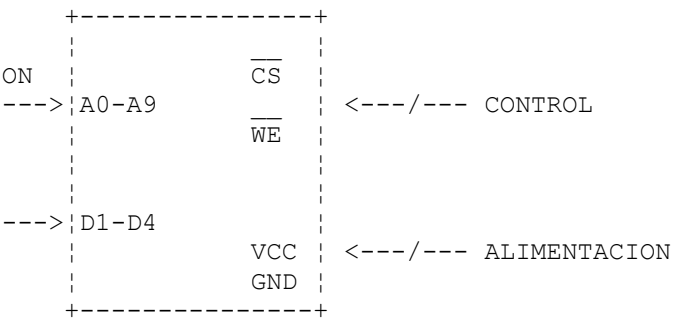
GESTION DE MEMORIA EN LA SEGMENTACION (INTEL)

PAGINACION. LA MEMORIA SE DIVIDE EN PAGINAS LOGICAS,CADA UNA DE LAS CUALES TIENE UNOS CUANTOS MILES DE CELDAS.
CADA ACCESO A MEMORIA SE HACE A PARTIR DE UN IDENTIFICADOR DE PAGINA QUE SELECCIONA LA PAGINA DESEADA, Y UN DESPLAZAMIENTO QUE LOCALIZA LA POSICION DE LA CELDA EN LA PAGINA.

DIFERENCIAS ENTRE PAGINACION Y SEGMENTACION.

1.- LAS PAGINAS TIENEN UN TAMAÑO FIJO, MIENTRAS QUE EL TAMAÑO DE LOS SEGMENTOS PUEDE VARIAR.

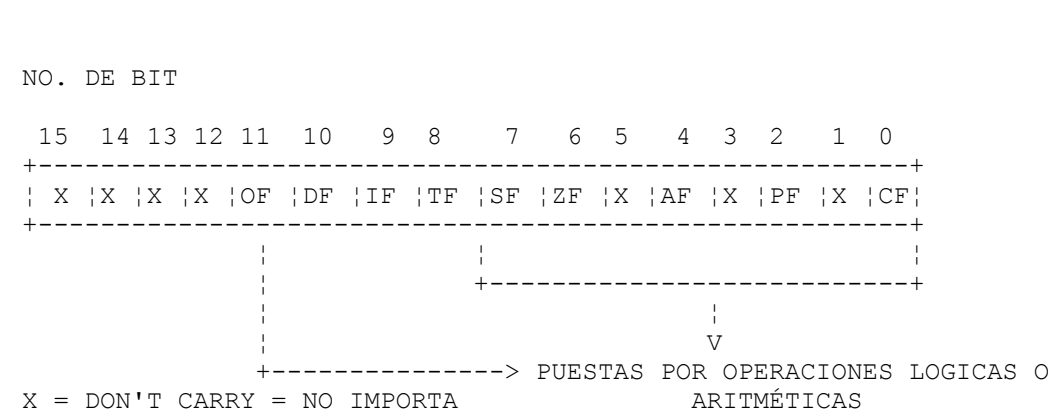
2.- LOS SEGMENTOS SE PUEDEN SUPERPONER Y LAS PAGINAS NO.



TERMINALES DE SALIDA DE UN CHIP DE MEMORIA TIPICO.

1.5.- REGISTRO DE BANDERAS (PSW PROCESSOR STATUS WORD).

ESTE REGISTRO CONTIENE INFORMACION QUE ES USADA UN BIT A UN TIEMPO, EN VEZ DE UN NUMERO DE 16 BITS. LOS BITS EN EL REGISTRO DE BANDERAS TIENEN SIGNIFICADO INDIVIDUAL PARA EL MICROPROCESADOR. ALGUNOS DE ESOS BITS CONTIENEN EL CODIGO DE CONDICION PUESTO POR LA ULTIMA INSTRUCCION EJECUTADA. EL PROGRAMA USA ESOS CODIGOS PARA MODIFICAR LA EJECUCION DEL PROGRAMA. EL PROGRAMA PUEDE PROBAR LOS CODIGOS DE CONDICION EN EL PROGRAMA, Y EL PROGRAMA PUEDE SELECCIONAR UNA TRAYECTORIA DE EJECUCION ALTERNA BASADA EN ESOS VALORES. OTROS BITS DENTRO DE EL REGISTRO DE BANDERAS INDICAN EL ESTADO DE OPERACION CORRIENTE DEL PROCESADOR, INSTRUCCIONES INDIVIDUALES CONTROLAN ESOS BITS.



LAS BANDERAS EN EL BYTE DE ALTO ORDEN (BANDERAS DE CONTROL) REFLEJAN EL ESTADO DEL PROCESADOR 8088 Y AFECTAN LA MANERA EN EL CUAL EL PROGRAMA SE EJECUTE. LAS BANDERAS EN EL BYTE DE BAJO ORDEN (BANDERAS CONDICIONALES) SON CODIGOS DE CONDICION, Y PUEDEN SER INTERROGADOS POR INSTRUCCIONES DE BRINCO CONDICIONALES PARA CAMBIAR EL ORDEN DE EJECUCION DEL PROGRAMA.

LAS BANDERAS CONDICIONALES SON:

SF (BANDERA DE SIGNO).- INDICA CUANDO EL RESULTADO DE LA

ULTIMA OPERACION ARITMETICA O LOGICA FUE POSITIVA O NEGATIVA. LO PUESTO EN EL BIT DE SIGNO REFLEJA EL VALOR DEL BIT MAS ALTO DE EL ULTIMO RESULTADO. SI EL ULTIMO RESULTADO ES NEGATIVO, EL BIT DE SIGNO ES PUESTO A UNO, SI ES POSITIVO O ZERO, EL BIT DE SIGNO ES BORRADO A ZERO.

ZF (BANDERA DE CERO).- INDICA CUANDO EL RESULTADO DE LA ULTIMA OPERACION FUE IGUAL A CERO. ES PUESTA A "1" SI EL RESULTADO ES CERO Y "0" SI EL RESULTADO ES DIFERENTE DE CERO. EL PROGRAMA UTILIZA ESTA BANDERA PARA SABER SI DOS NUMEROS SON IGUALES.

PF (BANDERA DE PARIDAD).- INDICA CUANDO EL NUMERO DE UNOS EN EL RESULTADO ES PAR O IMPAR. LA BANDERA DE PARIDAD ES PUESTA A "1" SI EL RESULTADO DE LA OPERACION TIENE UN NUMERO PAR DE UNOS. LA BANDERA ES BORRADA A "0" SI EL NUMERO DE UNO EN EL RESULTADO ES IMPAR.

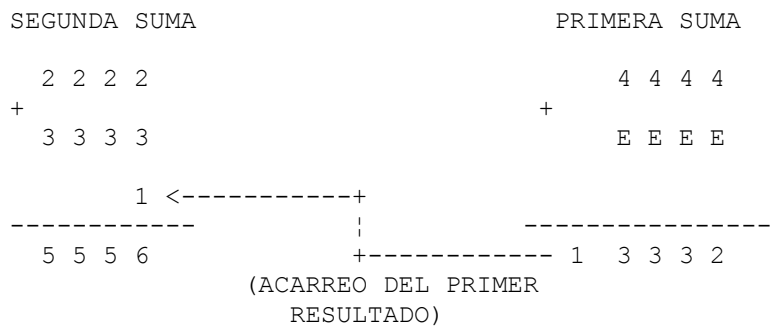
CF (BANDERA DE ACARREO).- ES EL ENLACE DEL PROCESADOR PARA ARITMETICA DE ALTA PRECISION. UNA SUMA CAUSA QUE ESTA BANDERA SEA PUESTA SI HAY UN ACARREO EN EL MSB, Y EN UNA RESTA CAUSA QUE SEA PUESTA SI UN PRESTAMO ES NECESARIO.

EJEMPLO

EN LA RESTA SI CF = 1 INDICA

QUE EL SUSTRAENDO ES MAYOR QUE EL MINUENDO.

```
    MINUENDO
  -
    SUSTRAENDO
-------------------
    CF = 1
SUSTRAENDO > MINUENDO
O  CF = 0  MINUENDO > = SUSTRAENDO.
```



SUMA DE 32 BITS CON CARRY

AF (BANDERA DE ACARREO AUXILIAR).- ES PUESTA SI HAY UN ACARREO HACIA AFUERA DE LOS 3 BITS DURANTE UNA SUMA O UN BORROW (PRESTAMO) POR EL BIT 3 DURANTE UNA RESTA. ESTA BANDERA ES USADA EXCLUSIVAMENTE POR ARITMETICA BCD.

EJEMPLO

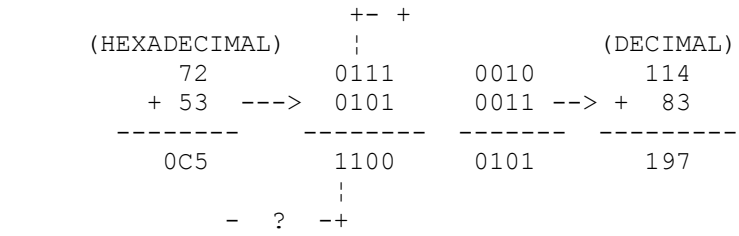
```
   38
 + 29
 ----
   61    ADICION DECIMAL

         => RESULTADO INCORRECTO.

         .
        . . SE REQUIERE AJUSTE DECIMAL PARA ADICION (DAA)

        AF = 1 ------+
                     |
                     |
                     |
                     |
                61 <-+ AJUSTE
             +   6
             ---------
                67     RESULTADO CORRECTO
```

OF (BANDERA DE OVERFLOW (SOBRE FLUJO)).- ES LA UNICA EN BYTE DE ALTO ORDEN DE EL REGISTRO DE BANDERAS QUE ES PUESTA P

UNA OPERACION ARITMETICA NORMAL. LA BANDERA DE SOBREFLUJO ES VITAL PARA LA ARITMETICA DE 2'S COMPLEMENTO.

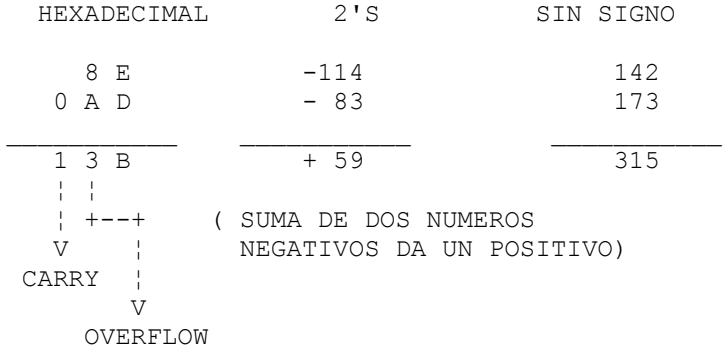
LA ARITMETICA DE 2'S UTILIZA EL MSB (BIT MAS SIGNIFICATIVO) PARA INDICAR EL SIGNO DE EL NUMERO. LA UNIDAD DE ADICION DE EL PROCESADOR MANEJA AMBOS NUMEROS SIGNADOS Y SIN SIGNO. NORMALMENTE LA SUMA DE NUMEROS SIGNADOS ES CORRECTA. SIN EMBARGO HAY CIERTOS NUMEROS EN 2'S QUE CUANDO SON SUMADOS JUNTOS PRODUCEN UN RESULTADO INCORRECTO.

EJEM. SUMA EN HEXADECIMAL



SI EL 72 HEXADECIMAL Y 53 HEXADECIMAL SON NUMEROS SIN SIGNO, EL RESULTADO ES CORRECTO. SI, SIN EMBARGO LOS NUMEROS SON NUMEROS SIGNADOS 2'S EL RESULTADO ES INCORRECTO.

SI EL RESULTADO DE LA SUMA NO PUEDE SER REPRESENTADO EN EL RANGO DE 2'S DE 8 BITS (127 A -128). ESTE EFECTO ES CONOCIDO COMO OVERFLOW (SOBREFLUJO), ENTONCES LA ADICION TIENE SOBREFLUJO EN EL RANGO DE NUMEROS DE 2'S (OVERFLOW NO ES IGUAL A CARRY Y TIENE SIGNIFICADO DIFERENTE).

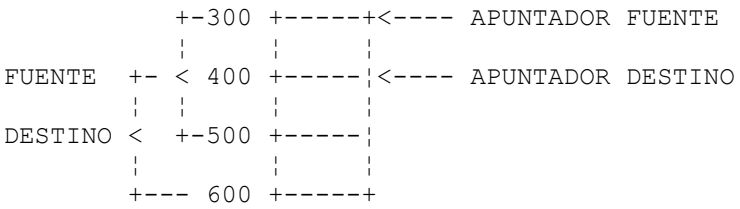


EJEMPLOS:

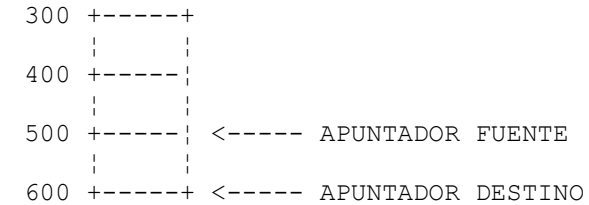
```
                        128 64 32 16  8  4  2  1

          35              0  0  1  0  0  0  1  1
        +
          12              0  0  0  0  1  1  0  0
      ----------      ---------------------------------
          47              0  0  1  0  1  1  1  1


          35           35
        +(-12)       - 12
      ----------  =  ---------
                       23


           0 0 0 0        1 1 0 0

           1 1 1 1        0 0 1 1      12 EN 1'S
         +
                              1
      -------------------------------
           1 1 1 1        0 1 0 0    <--- (-12)   EN    2'S
         +
           0 0 1 0        0 0 1 1         (+35)
      -------------------------------  ------------
           0 0 0 1        0 1 1 1          +23
```

LAS BANDERAS DE CONTROL SON:

DF (BANDERA DE DIRECCION). LAS INSTRUCCIONES DEL 8088 CONTIENEN VARIAS INSTRUCCIONES DE STRINGS QUE OPERAN SOBRE UN GRAN BLOQUE DE DATOS. ESAS INSTRUCCIONES MANIPULAN BLOQUES DE DATOS UN BYTE O PALABRAS A UN TIEMPO. EL REGISTRO INDICE APUNTA A EL BLOQUE DE DATOS. SIGUIENDO LAS OPERACIONES SOBRE UN BYTE O PALABRAS, EL PROCESADOR CAMBIA EL REGISTRO INDICE PARA APUNTAR AL SIGUIENTE ELEMENTO EN EL BLOQUE. OPERACIONES DE CARACTERES USAN LA BANDERA DE DIRECCION PARA INDICAR LA DIRECCION DE MOVIMIENTO A TRAVEZ DEL BLOQUE DE DATOS. CUANDO LA BANDERA DE DIRECCION ES BORRADA A "0" LAS OPERACIONES DE CARACTERES INCREMENTAN EL REGISTRO INDICE. SI LA BANDERA DE DIRECCION ES PUESTA A "1", OPERACIONES CON CARACTERES DECREMENTAN EL REGISTRO INDICE.



A).- DF = 0
(INCORRECTO)



B).- DF= 1
(CORRECTO)

IF (BANDERA HABILITADORA DE INTERRUPCIONES.- CONTROLA INTERRUPCIONES EXTERNAS. DURANTE AQUELLA SECCION DEL PROGRAMA DEL USUARIO DONDE ES INDESEABLE PERMITIR INTERRUPCIONES EXTERNAS, EL PROGRAMA PUEDE BORRAR LA BANDERA DE INTERRUPCION. MIENTRAS LA BANDERA DE INTERRUPCION ES BORRADA A "0" NO PUEDEN OCURRIR INTERRUPCIONES EXTERNAS. CUANDO EL PROGRAMA PONE LA BANDERA DE INTERRUPCION A "1", DISPOSITIVOS EXTERNOS PUEDEN GENERAR INTERRUPCIONES. LOS PROGRAMAS DE USUARIO CONTROLAN LAS BANDERAS DE INTERRUPCION.

TF (BANDERA DE TRAMPA). ESTA BANDERA AYUDA A SEGUIR (DEBUG) PASO A PASO LOS PROGRAMAS. ESTA BANDERA NO ES PUESTA POR UNA OPERACION DEL PROCESADOR. EL PROGRAMA PONE ESTA BANDERA, TAMBIEN LLAMADA LA BANDERA DE TRAZO O LA BANDERA PASO POR PASO, CON UNA INSTRUCCION ESPECIFICA. CUANDO ESTA BANDERA ESTA ACTIVA, UNA INTERRUPCION OCURRE DESPUES DESPUES DE CADA INSTRUCCION.

ASI PARA EJECUTAR UN PROGRAMA SIN HACERLO PASO POR PASO ES NECESARIO QUE EL PROGRAMA DEL USUARIO BORRE LA BANDERA DE TRAMPA.

II. METODOS DE DIRECCIONAMIENTO

REGLAS PARA LOCALIZAR UN OPERANDO (CODIGO DE OPERACIÓN O DATO) DE UNA INSTITUCION.

REFERENCIA A REGISTROS = REGISTRO ESPECIFICO

REFERENCIA A MEMORIAS

- DIRECCION DE SEGMENTO
- +
- DIRECCION DE BASE
- +
- CANDIDAD INDICE
- +
- DECALAGE

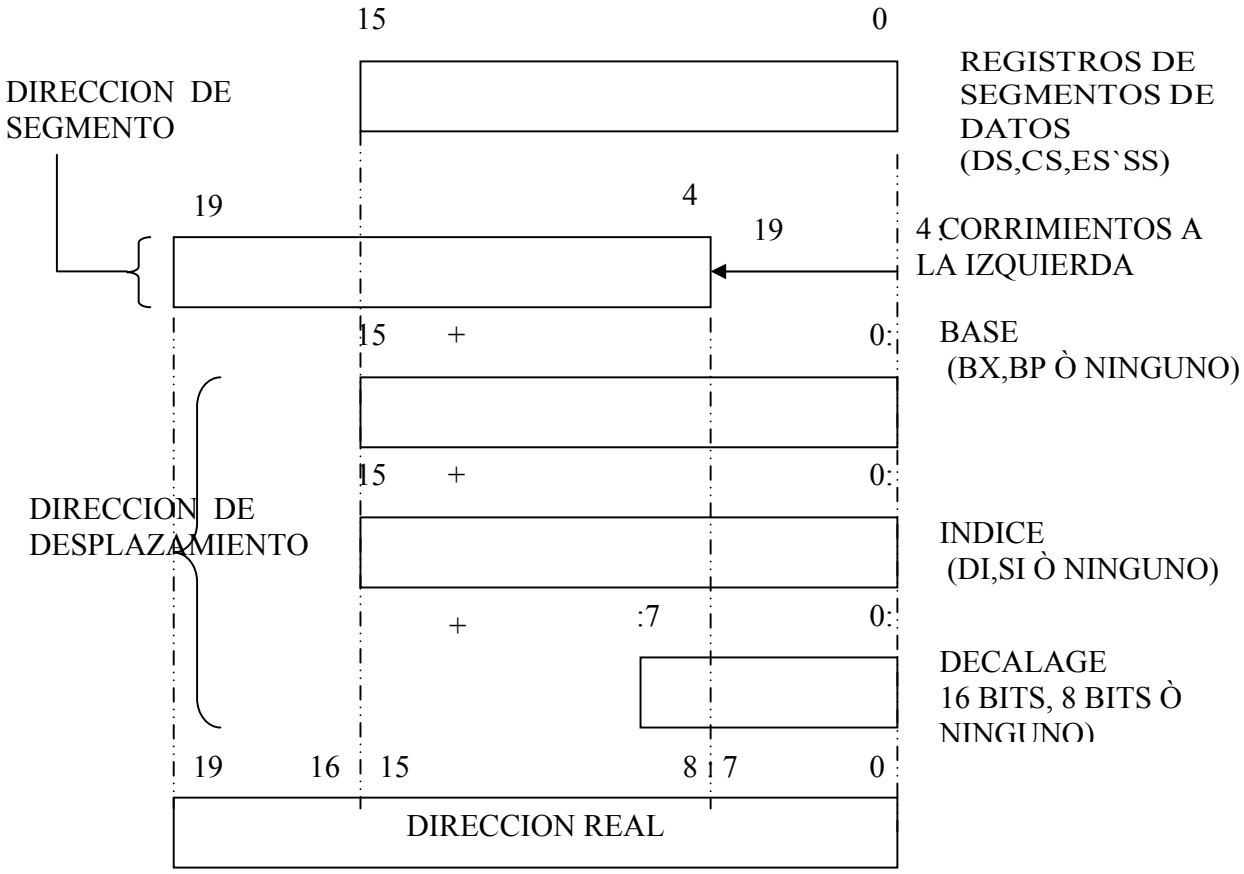


GRAFICO DE MODOS DE DIRECCIONAMIENTO EN EL 8086

DECALAGE= DESPLAZAMIENTOS=CANTIDAD ESTATICA. SE FIJA EN EL TIEMPO DE ENSAMBLAJE (PASO DE CODIGO FUENTE A CODIGO MAQUINA) Y NO PUEDE CAMBIARSE DURANTE LA EJECUCION DEL PROGRAMA (NI DEBE CAMBIARSE).
POR EL NOMBRE SE DA A ENTENDER QUE LA BASE SE MODIFICA MENOS QUE EL INDICE.

POR ESTAR LA BASE Y EL INDICE DENTRO DEL CPU SE PUEDEN MODIFICAR FACILMENTE.

EJEMPLO. INSTRUCCION INC (INCREMENTAR) Y EL UNICO OPERANDO ESTA EN MEMORIA.

MODO DE DIRECCIONAMIENTO: USA LA BASE, EL INDICE Y UN DECALAGE (OFFSET O DESPLAZAMIENTO) DE 8 BITS.

LA DIRECCION DEL SEGMENTO EN EL SEGMENTO DE DATOS (DS), LA BASE EN EL REGISTRO DE BASE (BX) Y EL INDICE ESTA EN EL REGISTRO INDICE DE DATOS (DI) Y EL OFFSET ES 6.

```
      INC  6 [ BX] [ DI ]                ; INCREMENTA EL
                                         ; CONTENIDO
                                         ; DE BX  + DI  + 6
```

SUPONGAMOS BX = 4000 H, DI = 20 H Y DS = 3000 H

EL DESPLAZAMIENTO SE DETERMINA EN LA SIGUIENTE FORMA:

DESPLAZAMIENTO = OFFSET + (BX) + (DI) = 6 + 4000 + 20 = 4026H

EL REGISTRO DE SEGMENTACION SE MULTIPLICA POR 16 (DECIMAL) ANTES DE AÑADIRSELE EL RESTO. DADO QUE 16 EN DECIMAL ES 10 EN HEXADECIMAL, ESTO PRODUCE UN DESPLAZAMIENTO DE UN DIGITO HEXADECIMAL (4 BITS) A LA IZQUIERDA. COMBINADO ESTO CON LA DIRECCIONA RELATIVA EN EL SEGMENTO NOS DA:

```
DIRECCION REAL = 10 * (DS ) + DESPLAZAMIENTO  = 3000 * 10 + DESPL.
      O EFECTIVA                                =30000 + 4026
                                                =34026 H
```

ASI 34026 H ES LA DIRECCION REAL DE MEMORIA DONDE ENCONTRAMOS OPERANDO, MIENTRAS 4026 ES LA DIRECCION DEL SEGMENTO, QUE ES LO QUE INTERESA AL PROGRAMADOR.

< < INSTRUCCIONES EN EL LENGUAJE MAQUINA > >

UNA INSTRUCCIÓN ESTA DIVIDIDA EN GRUPOS DE BITS O CAMPOS, CON UN CAMPO LLAMADO EL CODIGO DE OPERACIÓN (OP CODE), EL CUAL INDICA A LA COMPUTADORA QUE HACER, Y LOS OTROS CAMPOS LLAMADOS LOS OPERANDOS, QUE INDICAN LA INFORMACION NECESARIA POR LA INSTRUCCIÓN PARA LLEVAR A CABO SU TAREA. UN
OPERANDO PUEDE CONTENER UN DATO, AL MENOS PARTE DE LA DIRECCION DE UN DATO, UN APUNTADOR INDIRECTO A UN DATO, U OTRA INFORMACION PERTENECIENTE A EL DATO PARA SER ACTIVADO POR LA INSTRUCCIÓN.

OP CODE	OPERANDO	. . . . .	OPERANDO

FORMATO GENERAL DE LA INSTRUCCIÓN

LAS INSTRUCCIONES PUEDEN CONTENER VARIOS OPERANDOS, PERO MAS OPERANDOS Y MAS LONGITUD DE ESOS OPERANDOS SON, MAS ESPACIO DE MEMORIA QUE ELLOS OCUPARAN Y MAS TIEMPO QUE ELLOS TOMARAN PARA TRANSFERIR CADA INSTRUCCIÓN HACIA EL CPU.
EN ORDEN PARA MINIMIZAR EL NUMERO TOTAL DE BITS EN UNA INSTRUCCIÓN, MAS INSTRUCCIONES , PARTICULARMENTE PARA AQUELLAS COMPUTADORAS DE 16 BITS, SON LIMITADAS A UNO O DOS OPERANDOS CON AL MENOS UN OPERANDO EN UNA INSTRUCCIÓN DE 2 OPERANDOS INVOLUCRANDO UN REGISTRO UNO DE ELLOS. PORQUE LA MEMORIA Y/O ESPACIOS I/O SON RELATIVAMENTE GRANDES, MEMORIA Y DIRECCION I/O REQUIERE VARIOS BITS, PERO PORQUE EL NUMERO DE REGISTROS ES PEQUEÑO, ESTE TOMA SOLO UNOS POCOS BITS ESPECIFICAR UN REGISTRO, POR LO

TANTO UN MEDIO PARA CONSERVAR BITS DE INSTRUCCIONES ES USAR REGISTROS LO MAS POSIBLE.

LA LIMITACION A DOS OPERANDOS HACE REDUCIR LA FLEXIBILIDAD DE MUCHAS INSTRUCCIONES, PERO NORMALMENTE LA FLEXIBILIDAD ESTRA NO ES NECESARIA.

POR EJEMPLO UNA INSTRUCCIÓN DE SUMA COMO LA QUE SE MUESTRA A CONTINUACION:

```
MOV  [ 300 ] , AX
ADD  AX , [ 200 ]
```

LA CUAL INVOLUCRA LOS DOS NUMEROS A SER SUMADOS Y EL RESULTADO, ES REDUCIDO A 2 OPERANDOS POR MEDIO DE PONER LA SUMA DENTRO DE LA LOCALIZACION QUE CONTIENE EL SUMANDO. ESTO SIGNIFICA QUE EL SUMANDO ES PERDIDO, PERO ESTO ES USUALMENTE NO IMPORTANTE.

LO IMPORTANTE ES QUE EL SUMANDO PUEDA ESTAR DUPLICADO (POR MEDIO DE ALMACENARLO EN OTRA PARTE) ANTES DE QUE LA SUMA SEA EJECUTADA (EL RESULTADO DE LA SUMA SE GUARDA EN EL ACUMULADOR).

MODO DE DIRECCIONAMIENTO, LA MANERA EN LA CUAL UN OPERANDO ES ESPECIFICADO Y ES LLAMADO.

2.1.- DIRECCIONAMIENTO INMEDIATO

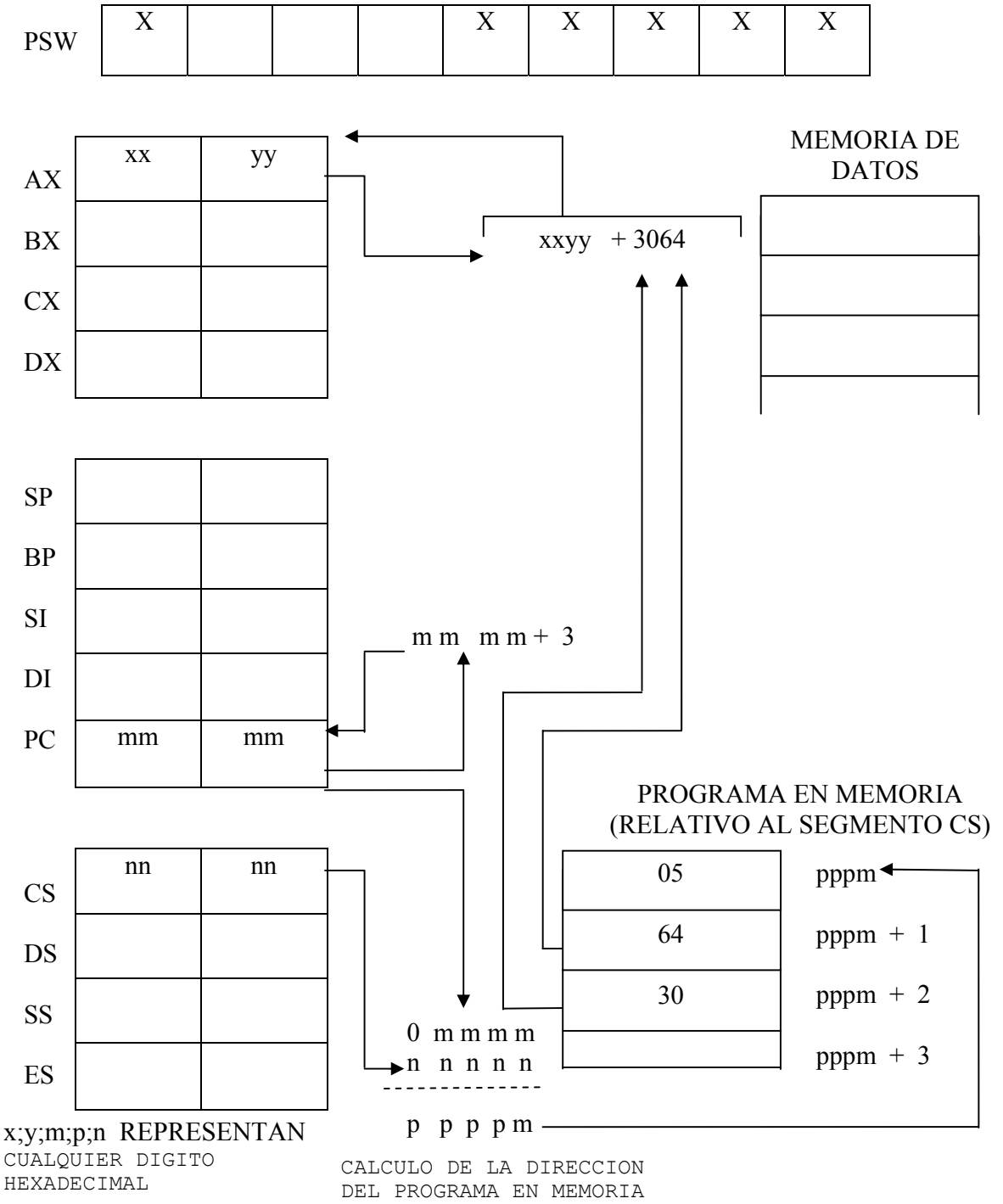
EL DATO ES CUALQUIERA 8 BITS O 16 BITS DE LONGITUD Y ES PARTE DE LA INSTRUCCIÓN.

INSTRUCCIÓN

EN ESTA FORMA DE DIRECCIONAMIENTO, UNO DE LOS OPERANDOS ESTA PRESENTE INMEDIATAMENTE EN EL BYTE SIGUIENTE AL OP CODE (INSTRUCCIÓN OBJET CODE).

POR EJEMPLO : ADD AX , 3064 H

O D I T S Z A P C

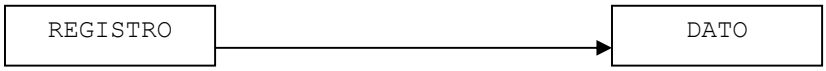


NOTE QUE LOS 16 BITS DEL OPERANDO INMEDIATO, CUANDO SE ALMACENEN EN UN PROGRAMA EN MEMORIA, TIENEN EL BYTE DE BAJO ORDEN PRECEDIENDO AL BYTE DE ALTO ORDEN.

2.2.- DIRECCIONAMIENTO A REGISTROS

EL DATO ES EL REGISTRO QUE ES ESPECIFICADO POR LA INSTRUCCIÓN. POR UN OPERANDO DE 16 BITS UN REGISTRO PUEDE SER AX, BX, DX, CX, SI, DI , SP, O BP Y PARA UN OPERANDO DE 8 BITS UN REGISTRO PUEDE SER AL, AH, BL, BH, CL ,CH, DL O DH.

INSTRUCCIÓN REGISTRO

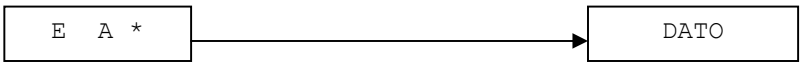


EJEMPLO: ADD AX, BX => EL CONTENIDO DE (BX) ES SUMADO A AX

2.3.- DIRECCIONAMIENTO DIRECTO

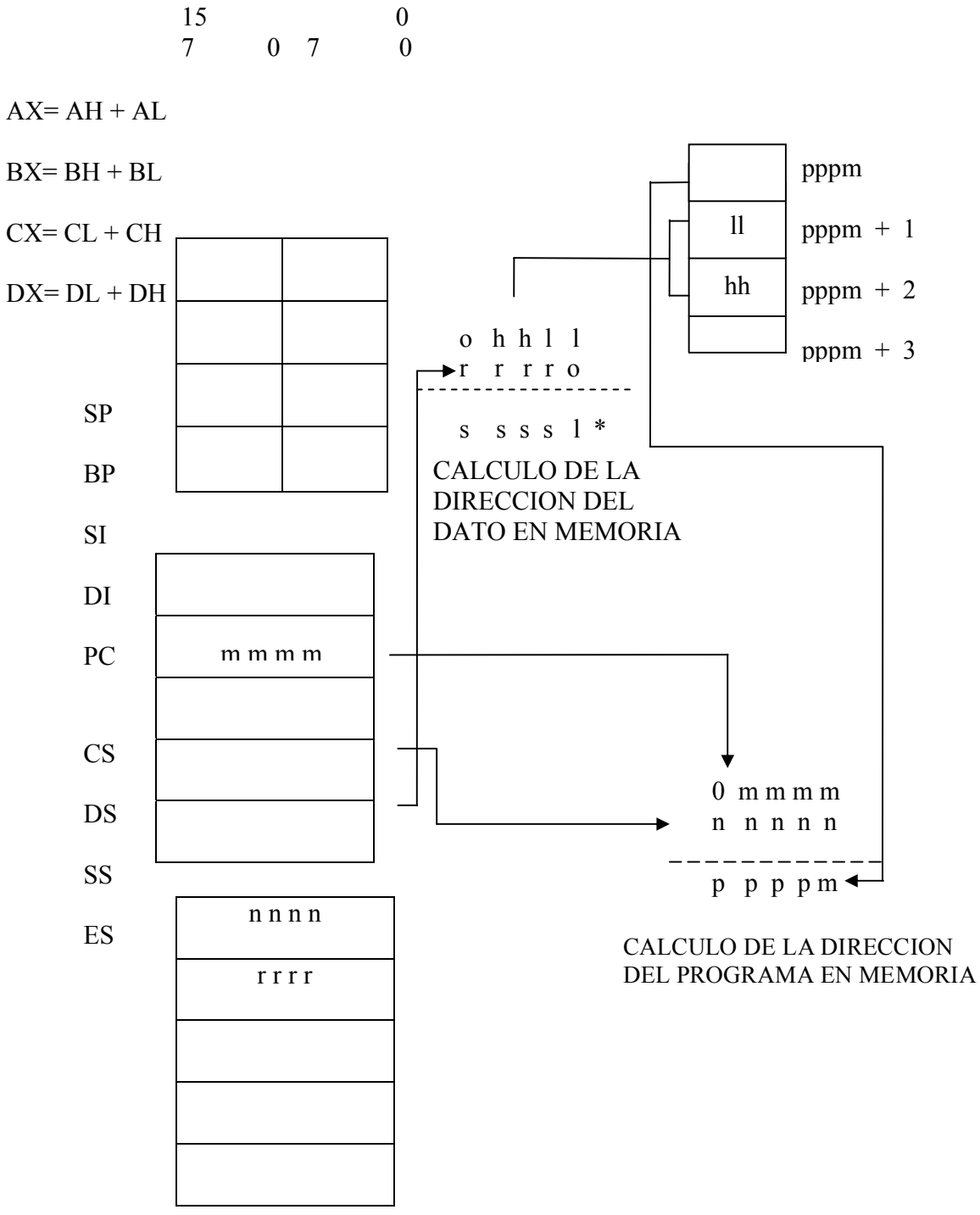
LOS 16 BITS DE DIRECCION EFECTIVA DE EL DATO ES PARTE DE LA INSTRUCCIÓN.

INSTRUCCIÓN MEMORIA



*E A (EFFECTIVE ADDRESS = DIRECCION EFECTIVA) ES SUMADO A 16 10 VECES EL CONTENIDO DE EL REGISTRO DEL SEGMENTO APROPIADO.

EL 8086 IMPLEMENTA DIRECCIONAMIENTO EN LINEA RECTA A UNA DIRECCION DE MEMORIA POR MEDIO DE SUMAR UN DESPLAZAMIENTO DE 16 BITS, PROVISTOS POR 2 BYTES DE EL CODIGO OBJETO A EL REGISTRO DE SEGMENTO DE DATOS. LA SUMA SE CONVIERTE EN LA DIRECCION ACTUAL DE MEMORIA.



h;l;m;n;p;r;s REPRESENTAN DIGITOS HEXADECIMALES.
LA DIRECCION DE DATOS EN MEMORIA ACTUAL DE SALIDA PARA DIRECCIONAMIENTO A MEMORIA DIRECTA.

EJEMPLO GRAFICO DEL DIRECCIONAMIENTO DIRECTO

2.4.-DIRECCIONAMIENTO INDIRECTO CON INDICE E INDIRECTO CON BASE.

LA DIRECCION EFECTIVA DEL DATO ESTA EN EL REGISTRO BASE BX O UN REGISTRO INDICE, ESTO ES ESPECIFICADO POR LA INSTRUCCION.

E A= { (BX) (DI) (SI) }

INSTRUCCIÓN REGISTRO MEMORIA

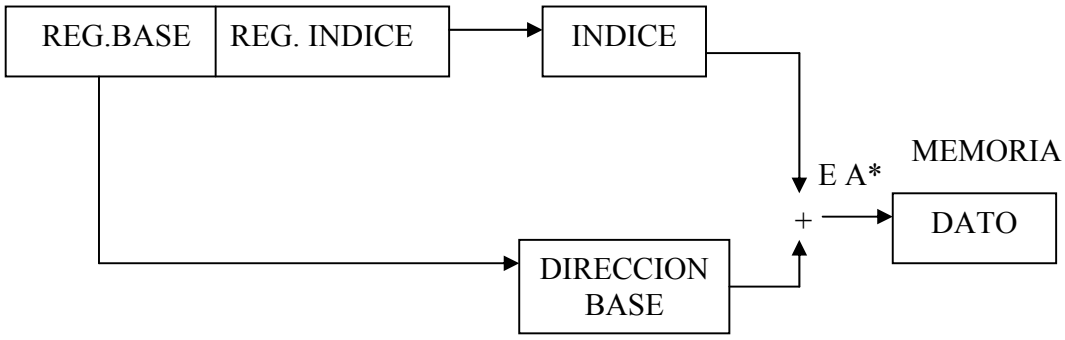
- E A ES SUMADO A 1610 VECES EL CONTENIDO DE UN REGISTRO DE SEGMENTO APROPIADO.

2.5.- DIRECCIONAMIENTO INDIRECTO CON BASE E INDICE.

LA DIRECCIONA EFECTIVA ES LA SUMA DE UN REGISTRO BASE Y UN REGISTRO INDICE, AMBOS SON ESPECIFICADOS POR LA INSTRUCCIÓN:

E A = { (BX) (BP) } + { (S I) (DI) }

INSTRUCCIÓN REGISTRO

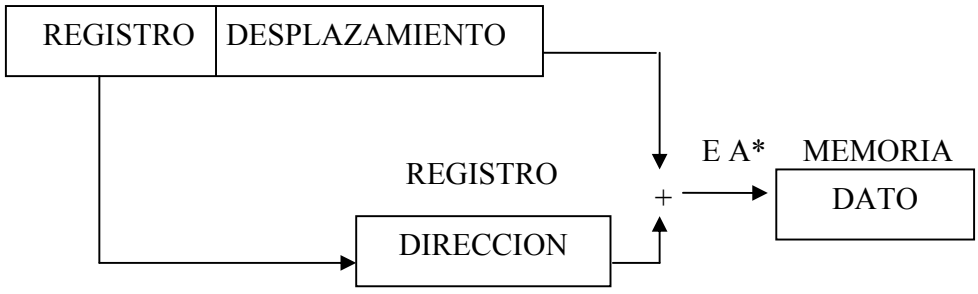


2.6.- DIRECCIONAMIENTO INDIRCTO CON BASE O INDICE Y CONSTANTE.

LA DIRECCION EFECTIVA ES LA SUMA DE UN DESPLAZAMIENTO DE 8 O 16 BITS Y EL CONTNIDO DE UN REGISTRO BASE O UN REGISTRO INDICE.

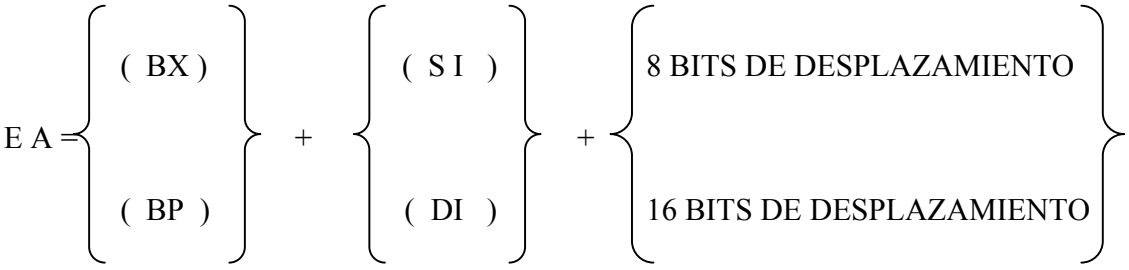
$$E\,A = \left\{ \begin{array}{l} (BX) \\ (BP) \\ (SI) \\ (DI) \end{array} \right\} + \left\{ \begin{array}{l} 8\ BITS\ DE\ DESPLAZAMIENTO \\ 16\ BITS\ DE\ DESPLAZAMIENTO \end{array} \right\}$$

INSTRUCCIÓN

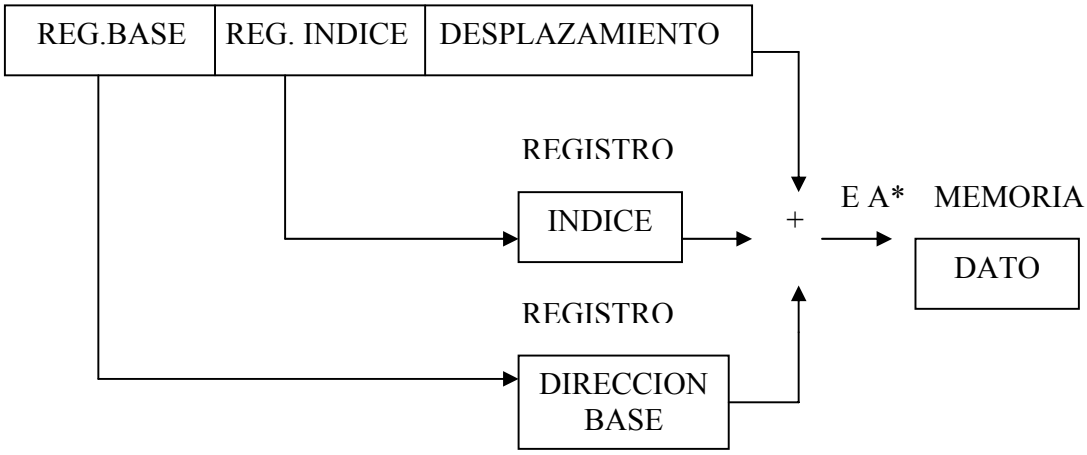


2.7.- DIRECCIONAMIENTO INDIRECTO CON BASE E INDICE Y CONSTANTE

LA DIRECCION EFECTIVA ES LA SUMA DE UN DESPLAZAMIENTO DE 8 O 16 BITS Y UNA DIRECCION CON BASE E INDICE.



INSTRUCCIÓN



POR EJEMPLO , SI

(BX) = 0158 DI = 10A5 DESPLAZAMIENTO = 1B57 (DS) = 2100 Y DS ES USADO COMO EL REGISTRO SEGMENTO, ENTONCES LA DIRECCION EFECIVA Y FISICA PRODUCIDA POR ESTAS CANTIDADES Y LOS VARIOS MODO DE DIRECCIONAMIENTO SERA :

DIRECCIONAMIENTO DIRECTO :

E A = 1B57

DIRECCION FISICA = 1B57 + 21000 = 22B57

DIRECCIONAMIENTO A REGISTRO: NO HAY DIRECCION EFECTIVA EL DATO ESTA EN EL REGISTRO ESPECIFICADO.

DIRECCIONAMIENTO INDIRECTO CON BASE: ASUMIENDO REG. BASE BX.

E A = 0158

DIRECCION FISICA = 0158 + 21000 = 21158

DIRECCIONAMIENTO INDIRECTO CON BASE Y CONSTANTE: ASUMIENDO REG. BASE BX.

E A = 0158 + 1B57 = 1CAF

DIRECCION FISICA = 1CAF + 21000 = 22CAF

DIRECCIONAMIENTO INDIRECTO CON BASE E INDICE: ASUMIENDO REGISTRO BASE BX E INDICE DI.

E A = 0158 + 10A5 = 11FD

DIRECCION FISICA = 11FD + 21000 = 22IFD

DIRECCIONAMIENTO INSDIRECTO CON BASE E INDICE Y CONSTANTE: ASUMIENDO REG. BASE. BX E INDICE DI.

E A = 0158 + 10A5 + 1B57 = 2D54

DIRECCION FISICA = 2D54 + 21000 = 23D54

III.- SET (CONJUNTO) DE INSTRUCCIONES

FORMA GENERAL DE UNA INSTRUCCION EN ASSEMBLER (ENSAMBLADOR) ES

ETIQUETA: NEMONICO OPERANDO, OPERANDO
^
¦
EVITA AMBIGUEDAD ENTRE EL NEMONICO Y EL 1er. OPERANDO

ETIQUETA: IDENTIFICA A LA DIRECCION ASIGNADA EN EL PRIMER BYTE DE LA INSTRUCCION EN LA CUAL APARECE.

NEMONICO: TODAS LAS INSTRUCCIONES DEBEN CONTENER UN NEMONICO (AYUDA A LA MENTE A RECORDAR SU OPERACION).

OPERANDOS: SU PRESENCIA DEPENDE DE LA INSTRUCCION. ALGUNAS INSTRUCCIONES NO TIENEN OPERANDO, ALGUNAS TIENEN UNO Y ALGUNAS TIENE DOS. SI SON DOS , ELLOS SON SEPARADOS POR COMAS. SI HAY DOS OPERANDOS, EL OPERANDO DESTINO APARECE PRIMERO Y EL OPERANDO FUENTE SEGUNDO.

CONSTANTE: UN NUMERO CUYA BASE ES IDENTIFICADO CON UN SUFIJO COMO SIGUE:

- B - BINARIO
- D - DECIMAL
- O - OCTAL
- H - HEXADECIMAL

EL DEFAULT ES HEXADECIMAL PARA EL DEBUG Y DECIMAL PARA ASSEMBLER ,NUMEROS HEXADECIMALES DEBEN TENER COMO PREFIJO UN 0.

10112 = 1011B

22310 = 223D = 223

B25A16 = 0B25AH

CONSTANTES ALFANUMERICAS. UN CARACTER ALFANUMERICO DEBE SER ENCERRADO EN UNAS SIMPLES CUOTAS ('). EJEM. MOV AL,'E'

3.1.- INSTRUCCIONES PARA MOVIMIENTO DE DATOS (INFORMACION)

PARA CUALESQUIER MEDIDA, LAS INSTRUCCIONES DEL MOVIMIENTO DE DATOS SON LAS MAS FRECUENTEMENTE USADAS EN EL SET DE INSTRUCCIONES DE CUALESQUIER COMPUTADORA. MUCHOS DE LOS PROBLEMAS DE PROCESAMIENTO DE DATOS CONSISTEN DE MOVIMIENTO DE INFORMACION DE UNA LOCALIDAD A OTRA. UN PROGRAMA PUEDE HACER ALGUN PROCESAMIENTO DE LA INFORMACION CUANDO ESTA ES MOVIDA, PERO EL GRUESO DEL TRABAJO ES JUSTAMENTE EL MOVIMIENTO.

MOVE: LA INSTRUCCION MOV ES PARA MOVER UN BYTE O UNA PALABRA (2 BYTES) DENTRO DEL CPU O ENTRE EL CPU Y MEMORIA.
DEPENDIENDO DE LOS MODOS DE DIRECCIONAMIENTO ESTE PUEDE TRANSFERIR INFORMACION DE LA SIGUIENTE MANERA :

REGISTRO A REGISTRO

OPERANDO INMEDIATO A REGISTRO

OPERANDO INMEDIATO A UNA LOCALIDAD DE MEMORIA

LOCALIDAD DE MEMORIA A UN REGISTRO

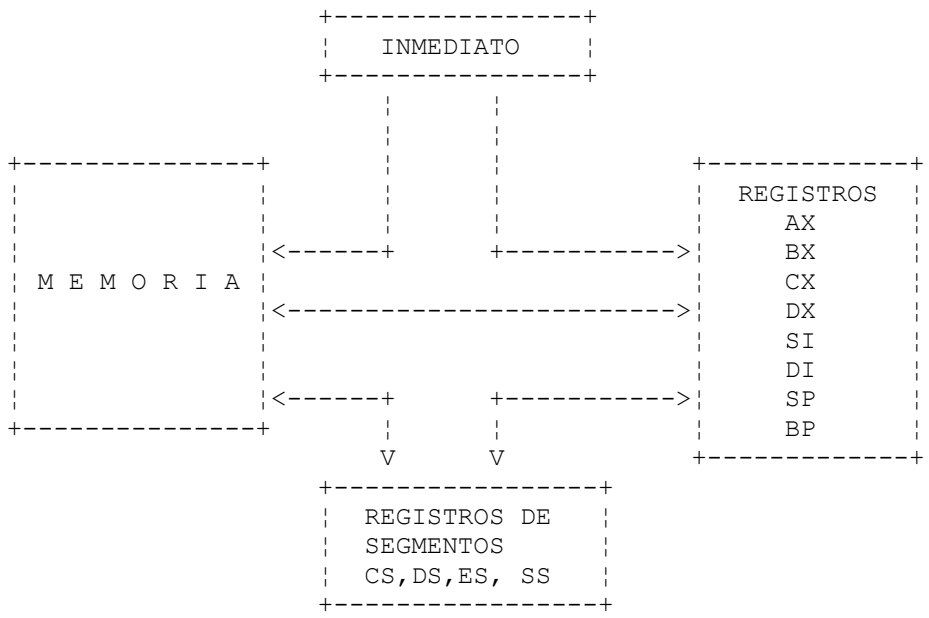
REGISTRO A UNA LOCALIDAD DE MEMORIA

LOCALIDAD DE MEMORIA/REGISTRO A UN REGISTRO SEGMENTO(EXCEPTO CS O IP)

REGISTRO SEGMENTO A UNA LOCALIDAD DE MEMORIA / REGISTRO
PARA UN OPERANDO INMEDIATO A REGISTRO SEGMENTO EN FORMA INDIRECTA SOLAMENTE POR MEDIO DE UN REGISTRO O LOCALIDAD DE MEMORIA.

ESTA INSTRUCCION MUEVE UN BYTE O UNA PALABRA DE DATOS DE UNA LOCALIDAD DE MEMORIA A UN REGISTRO, DE UN REGISTRO A UNA LOCALIDAD DE MEMORIA, O DE UN REGISTRO A UN REGISTRO; TAMBIEN PUEDE ALMACE-

NAR UN VALOR DETERMINADO EN UN REGISTRO O UNA LOCALIDAD DE MEMORIA.



MOVIMIENTO DE DATOS EN OPERACIONES CON MOV

LA INSTRUCCION MOV NO AFECTA BANDERAS .

LAS INSTRUCCIONES 8088 PUEDEN ESPECIFICAR SOLAMENTE UN OPERANDO EN MEMORIA EN CUALESQUIER INSTRUCCION. POR LO TANTO LA INSTRUCCION ADD REQUIERE AL MENOS QUE UNO DE LOS OPERANDOS SEA UN REGISTRO. EL 8088 NO TIENE LA CAPACIDAD DE SUMAR UNA LOCALIDAD DE MEMORIA A OTRA EN UNA SIMPLE INSTRUCCION.

NOTESE QUE NO HAY SENTIDO (FLECHAS) PARA ALMACENAR INFORMACION EN UNA INSTRUCCION, POR LO TANTO EL FLUJO DE DATOS PARA UNA INSTRUCCION INMEDIATA ES EN UNA SOLA DIRECCION.

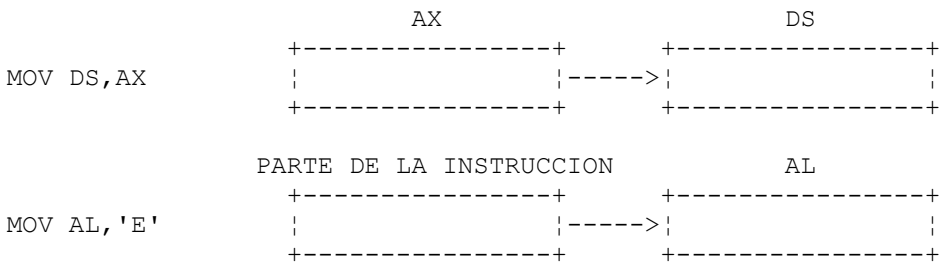
INSTRUCCIONES MOV

```
0100 89 D8           MOV AX,BX      ; REGISTRO BX-->REGISTRO AX

0102 89 C3           MOV BX,AX      ; REGISTRO AX-->REGISTRO BX


0104 8B 0E 0000 R    MOV CX,EXWORD  ; MEMORIA    -->REGISTRO

0108 89 16 0000 R    MOV EXWORD,DX  ; REGISTRO   -->MEMORIA


010C 8A 2E 0000 R    MOV CH,EXBYTE  ; MEMORIA    -->REGISTRO(BYTE)

0110 88 36 0000 R    MOV EXBYTE,DH  ; REGISTRO   -->MEMORIA(BYTE)


0114 BE 03E8         MOV SI,1000    ; INMEDIATO  -->REGISTRO

0117 B3 17           MOV BL,23      ; INMEDIATO  -->REGISTRO(BYTE)
0119 C7 06 0000 D007 MOV EXWORD,2000; INMEDIATO  -->MEMORIA

011F C6 06 0000 R 2E MOV EXBYTE,46  ; INMEDIATO  -->MEMORIA(BYTE)


0124 A1 0000 R       MOV AX,EXWORD  ; MEMORIA    -->ACUMULADOR

0127 A0 0000 R       MOV AL,EXBYTE  ; MEMORIA    -->ACUM.(BYTE)

012A A3 0000 R       MOV EXWORD,AX  ; ACUMULADOR -->MEMORIA

012D A2 0000 R       MOV EXBYTE,AL  ; ACUMULADOR -->MEMORIA(BYTE)


0130 8E 1E 0000 R    MOV DS,EXWORD  ; MEMORIA    -->REG. SEGMENTO

0134 8E D8           MOV DS,AX      ; REGISTRO   -->REG. SEGMENTO

0136 8C 1E 0000 R    MOV EXWORD,DS  ; REG.SEGMENTO->MEMORIA

013A 8C CD           MOV AX,ES      ; REG.SEGMENTO->REGISTRO
```

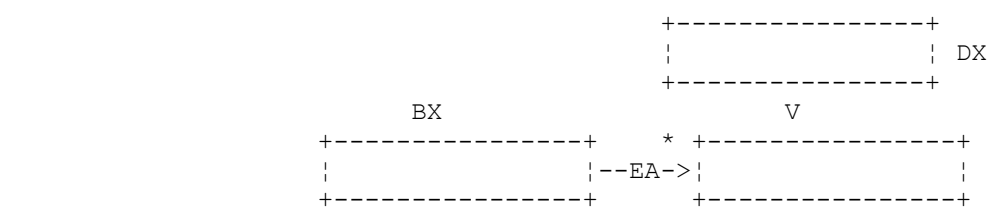
VALOR INMEDIATO A UN REGISTRO SEGMENTO

```
013C B8 LLHH R       MOV AX,CODE    ; TOMA VALOR INMEDIATO

013F 8E D8           MOV DS,AX      ; VALOR INMEDIATO A EL
                                      REGISTRO SEGMENTO
```

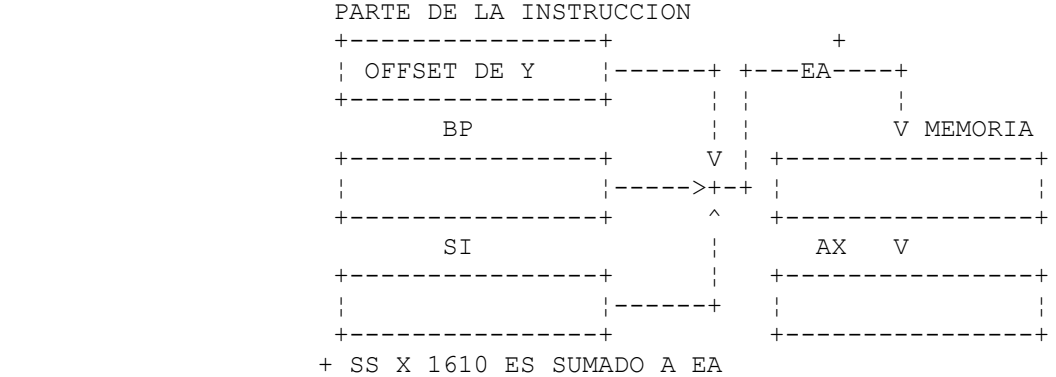
EJEMPLOS DE MOVIMIENTOS DE DATOS :



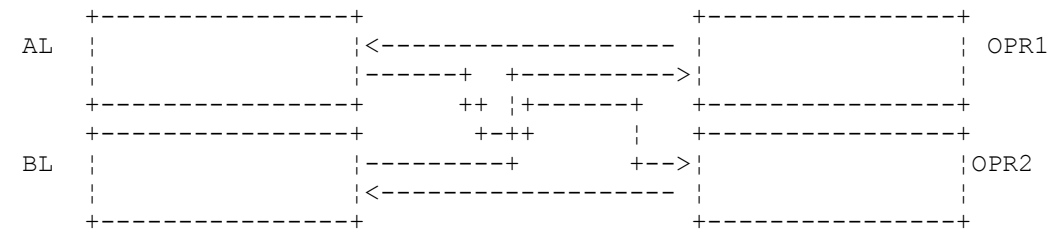
MOV [BX],DX MUEVE EL CONTENIDO DE DX HACIA LA LOCALIDAD INDICADA POR BX



MOV AX,Y[BP][SI] MUEVE HACIA AX EL CONTENIDO DE LA LOCALIDAD ENCONTRADA POR SUMAR (BP) ; (SI) Y EL OFFSET Y




```
MOV AL,OPR1
MOV BL,OPR2                          SECUENCIA DE  PROGRAMA  PARA
MOV OPR2,AL                          INTERCAMBIAR EL CONTENIDO DE
MOV OPR1,BL                                 2 LOCALIDADES
```



<< EXCHANGE (INTERCAMBIO) >>

LA INSTRUCCION EXCHANGE (XCHG) ES UN SIMPLE INTERCAMBIO DE DOS LOCACIONES.- LA INSTRUCCION PUDE CAMBIAR DOS REGISTROS O UN REGISTRO Y UNA LOCALIDAD DE MEMORIA.- UN REGISTRO DE SEGMENTO NO PUEDE SER UNA DE LAS LOCALIDADES DEL INTERCAMBIO.

LA INSTRUCCION INTERCAMBIO TOMA EL LUGAR DE TRES INSTRUCCIONES DE MOVER, Y NO REQUIERE UNA LOCALIDAD DE ALMACENAMIENTO TEMPORAL.- SI NO EXISTIERA LA INSTRUCCION INTERCAMBIO, UN PROGRAMA REQUERIRA DE 3 MOVIMIENTOS PARA HACER QUE LOS VALORES DE AX Y BX SE INTERCAMBIEN.- PRIMERO SE DEBERA MOVER AX A UNA LOCALIDAD TEMPORAL, LUEGO MOVER BX A AX Y FINALMENTE MOVER LA LOCALIDAD TEMPORAL A BX.- EL XCHG LLEVA A CABO ESTA OPERACION EN UNA SIMPLE INSTRUCCION.

```
XCHG BX,CX      ;  REGISTRO BX <-> REGISTRO CX
XCHG BX,EXWORD  ;  REGISTRO BX <-> MEMORIA
XCHG AX,BX      ;  REGISTRO AX <-> REGISTRO BX
```

<< INPUT Y OUTPUT (ENTRADA Y SALIDA) >>

EL 8088 TIENE INSTRUCCIONES IN Y OUT PARA LLEVAR A CABO OPERACIONES DE ENTRADA Y SALIDA . CADA DISPOSITIVO I/O DE LA IBM PC TIENE UNO O MAS REGISTROS CONSTRUIDOS Y QUE PUEDEN SER MANIPULADOS POR ESAS INSTRUCCIONES DE I/O .- CADA DISPOSITIVO DE I/O TIENE UNA DIRECCION PARA CADA REGISTRO EN EL DISPOSITIVO .- ESAS DIRECCIONES SON DIFERENTES DE LA DIRECCION DE MEMORIA DEL SISTEMA .- EL ESPACIO DE DIRECCIONES DE I/O ES SEPARADO DE EL ESPACIO DE MEMORIA .- HAY UN TOTAL DE 216 O 65,536 DIRECCIONES DE I/O DISPONIBLES EN EL 8088 .- LA IBM PC LOCALIZA 512 DE ESAS DIRECCIONES PARA LOS

CANALES DEL SISTEMA I/O PARA USARSE POR LOS VARIOS ADAPTADORES DE I/O .- OTRAS 256 DIRECCIONES SON UTILIZADAS POR EL SISTEMA DE TECLADO PARA CONTROLAR LAS DISPOSICIONES DE I/O DE AHI.

LA INSTRUCCION IN TRANSFIERE DATOS DE EL DISPOSITIVO I/O A EL REGISTRO AL .- LA INSTRUCCION PUEDE ESPECIFICAR LA DIRECCION DE EL DISPOSITIVO I/O EN 2 DIFERENTES MANERAS .- SI LA DIRECCION DE I/O ESTA EN EL RANGO DE 0 A 255, LA DIRECCION PUEDE TENER UN VALOR INMEDIATO .- SI LA DIRECCION ES MAYOR QUE 255, LA INSTRUCCION NOMBRA LA DIRECCION DE I/O INDIRECTAMENTE .- EL REGISTRO DX RETIENE LA DIRECCION I/O PARA LA INSTRUCCION INDIRECTA IN .- EL REGISTRO DX PUEDE PROVEER UNA DIRECCION INDIRECTA I/O PARA TODOS LOS DISPOSITIVOS I/O, INCLUYENDO AQUELLOS MENORES DE 256.

LA INSTRUCCION OUT OPERA EN UNA MANERA SIMILAR, EXCEPTO QUE ESTA ES ALMACENADA EN EL REGISTRO AL LO QUE SE VA A ENVIAR A EL REGISTRO DEL DISPOSITIVO I/O .- LA INSTRUCCION OUT ESPECIFICA LA DIRECCION EN LA MISMA MANERA QUE LA INSTRUCCION IN.

```
IN AL,020H      ;      PUERTO 20   ->  AL
IN AL,DX        ;      PUERTO (DX) ->  AL
OUT 021H,AL     ;      AL          ->  PUERTO 021H
OUT DX,AL       ;      AL          ->  PUERTO (DX)
```

<< TRANSFERENCIA DE BANDERAS >>

LA INSTRUCCION LAHF TOMA LOS 8 BITS MAS BAJOS DEL REGISTRO DE BANDERAS Y LOS MUEVE HACIA EL REGISTRO AH.

LA INSTRUCCION SAHF HACE EL PROCESO INVERSO. EL VALOR EN EL REGISTRO AH ES PUESTO EN LOS 8 BITS MAS BAJOS DEL REGISTRO DE BANDERAS.

<< OPERACIONES DEL STACK (PILA) >>

EL 8088 LOCALIZA EL STACK CON EL PAR DE REGISTROS SS:SP .- INFORMACION EMPUJADA HACIA EL STACK CAUSA QUE EL STACK CREZCA HACIA UNA DIRECCION MAS BAJA DE MEMORIA. EL STACK SIRVE COMO UNA LOCACION PARA GUARDAR LA LOCALIDAD DE RETORNO DE LA SUBRUTINA Y GUARDAR LOS REGISTROS O INFORMACION DESEADA.

PUSH: GUARDAR UN REGISTRO.

POP: SACAR UN REGISTRO QUE FUE GUARDADO CON PUSH.

CUALESQUIER LOCALIDAD DE MEMORIA QUE PUEDE SER ESPECIFICADA POR EL PROGRAMA CON LOS MODOS DE DIRECCIONAMIENTO TAMBIEN PUEDE SER PUSH(ED) O POP(ED).

PARA TODAS LAS OPERACIONES DE STACK CON EL 8088, LA UNIDAD BASICA DE INFORMACION ES LA PALABRA DE 16 BIT. TODAS LAS COSAS QUE SEAN GUARDADAS EN EL STACK O SACADAS DE EL SON DE UNA O MAS PALABRAS DE LONGITUD.NO HAY OPERACIONES DE UN BYTE DISPONIBLES CON PUSH O POP, POR EJEMPLO PARA GUARDAR AL SE REQUIERE GUARDAR AX.

EJEMPLO DE OPERACIONES CON STACK

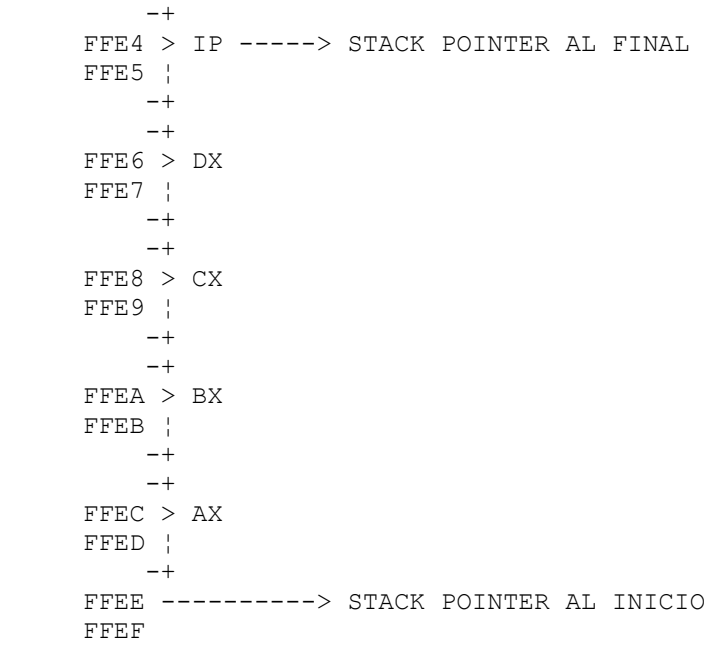
```
0000 50            PUSH AX        ;  PUSH UN REGISTRO
0001 56            PUSH SI        ;
0002 OE            PUSH CS        ; PUEDE GUARDAR REG. SEGMENTO
0003 FF 36 0000 R  PUSH EXWORD    ; PUEDE GUARDAR LOC. MEMORIA

0007 8F 06 0000 R  POP EXWORD     ; PUEDE SACAR CUALESQUIER COSA
000B 07            POP ES         ; QUE GUARDE
000C 5F            POP DI         ; POP NO NECESITA HACER JUEGO
000D 5B            POP BX         ; CON PUSH

000E 9C            PUSHF          ; DIFERENTE NEMONICO PARA
000F 9F            POPF           ; BANDERAS
```

EJEMPLO QUE MUESTRA PASE DE PARAMETROS

```
0010 50            PUSH AX        ; SALVA 4 PARAMETROS EN STACK
0011 53            PUSH BX
0012 51            PUSH CX
0013 52            PUSH DX
```

```
0014 E8 0017 R     CALL SUBRUTINE

 0017 8B EC         MOV BP,SP      ; ESTABLECE LA DIRECCION STACK
 0019 8B 46 02      MOV AX,[BP+2]  ; TOMA EL ULTIMO [DX]
 001C 8B 5E 04      MOV BX,[BP+4]  ; TOMA EL   3er. [CX]
 001F 8B 4E 06      MOV CX,[BP+6]  ; TOMA EL   2o.  [BX]
 0022 8B 56 08      MOV DX,[BP+8]  ; TOMA EL   1er. [AX]
```

MOVIMIENTO DEL STACK EN EL PASE DE PARAMETROS



EL USO PRIMARIO DE EL STACK ES GUARDAR INFORMACION TEMPORALMENTE.

```
 PUSH DX            EJEM: UN PROGRAMA NECESITA ENTRAR UN
                    VALOR EN LA DIRECCION I/O 3DAH , PERO
 MOV DX,3DAH        EL REGISTRO DX TIENE UN DATO IMPORTANTE.

 IN AL,DX
POP DX


 PUSH BX            EFECTO NETO ES INTERCAMBIAR LOS VALORES
                    DE LOS REGISTROS BX Y CX.
 PUSH CX            ESTRUCTURA DE DATOS LIFO.
  .
  .
 POP BX

 POP CX
```

```
MOV AX,CS           ; MUEVE CS A AX          ¦
                                             ¦
 MOV DS, AX          ; TOMA EL VALOR A DS    ¦
                                             ¦
                                            > EFECTO NETO
                                             ¦
 PUSH CS             ; REGISTRO CS AL STACK  ¦ IDENTICO
                                             ¦
 POP DS              ; PONE EL VALOR EN DS   ¦
                                       ------+
```

3.2.- INSTRUCCIONES ARITMETICAS

AUNQUE SON PEQUEÑAS EN NUMERO, ELLAS HACEN EL GRUESO DE LA TRANSFORMACION DE DATOS EN EL PROCESADOR.

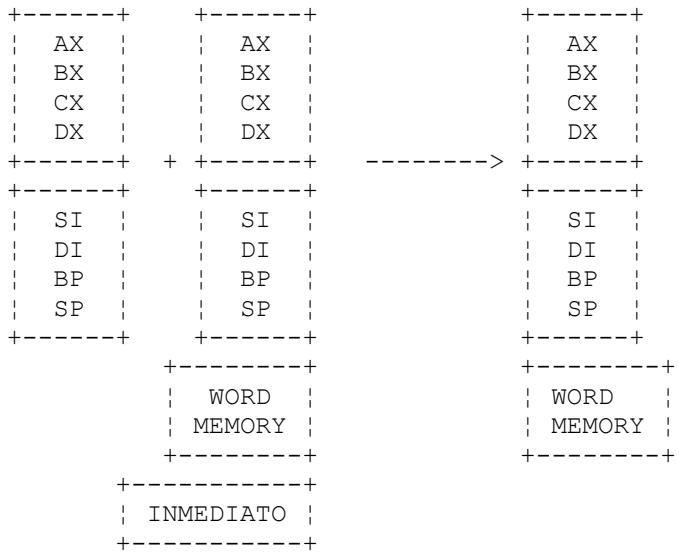
<< ADICION >>

LA INSTRUCCION ADD LLEVA A CABO LA ADICION EN DOS COMPLEMENTO DE LOS OPERANDOS ESPECIFICADOS. EL PROCESADOR COLOCA EL RESULTADO EN EL PRIMER OPERANDO, DESPUES DE SUMAR LOS DOS OPERANDOS JUNTOS. EL SEGUNDO OPERANDO PERMANECE SIN CAMBIO. LA INSTRUCCION PONE AL DIA EL REGISTRO DE BANDERAS PARA REFLEJAR EL RESULTADO DE LA ADICION. POR EJEMPLO LA INSTRUCCION:

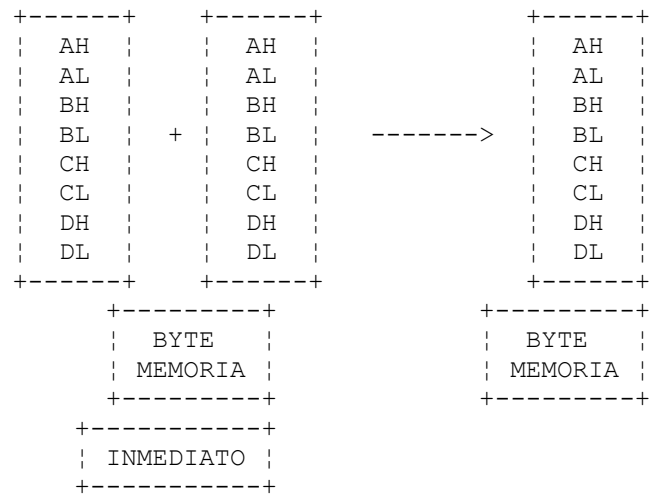
```
ADD  AX,BX

[AX] + [BX] -----> [AX]
```

EL REGISTRO DE BANDERAS NOS INDICA CUANDO EL RESULTADO ES ZERO, NEGATIVO, PARIDAD Y SI TIENE CARRY O OVERFLOW Y LA BANDERA ARITMETICA.

* UN REGISTRO SEGMENTO NO PUEDE SER DESTINO(SUMAS Y RESTAS)
* EL TAMAÑO DE LOS DOS OPERANDOS DEBEN SER IGUALES



ADICION CON 16 BIT



ADICION CON 8 BIT

<< OPERACIONES DE ADICION >>

ADC (ADD WITH CARRY) ES LO MISMO QUE LA INSTRUCCION ADD EXCEPTO QUE LA BANDERA DE CARRY ES INCLUIDA EN LA ADICION.

UN PROGRAMA PUEDE USAR LA BANDERA DE CARRY PARA OPERACIONES DE PRESICION MULTIPLE.

EJEMPLO:

```
0008 A1 0000 R      MOV AX,WORD PTR VALUE 1

000B 01 06 0004 R   ADD WORD PTR VALUE 2,AX     ; LLEVA ACABO UNA
                                                ; SUMA DE 16 BITS
000F A1 0002 R      MOV AX,WORD PTR VALUE 1 + 2

0012 11 06 0006 R   ADC WORD PTR VALUE 2 + 2,AX ; SUMA LOS 16 BITS
                                                ; MAS ALTOS
0000      ?? ?? ?? ??     VALUE 1

0004      ?? ?? ?? ??     VALUE 2
```

EJEMPLO ANTERIOR CON:

CARRY = 0		CARRY = 1	
0015	25AO	0015	65AO
+ 0021	B79E	0021	B79E
0036	DD3E	0037	ID3E

DOBLE PRESICION ACOMODA 232 = 22 (210)3 ¸ 4 BILLONES

<< SUBSTRACCION >>

LAS INSTRUCCIONES DE SUBSTRACCION (SUB Y SBB) SON IDENTICAS A LAS INSTRUCCIONES DE SUMA EXCEPTO QUE ELLAS HACEN LA SUBSTRACCION EN VEZ DE LA ADICION.

LA SUBSTRACCION PONE LAS BANDERAS DE STATUS DE ACUERDO A EL RESULTADO DE LA OPERACION, CON LA BANDERA DE CARRY AHORA REPRESENTA UN BORROW (PRESTAMO). POR EJEMPLO LA INSTRUCCION:

```
SUB  AX , BX

[AX] - [BX] ---> [AX]
```

LAS BANDERAS DE STATUS SON CAMBIADAS PARA REFLEJAR EL RESULTADO SBB INCLUYE LA BANDERA BORROW COMO UNA PARTE DE LA SUBSTRACCION. EL VALOR DE LA BANDERA BORROW ES RESTADO DE EL RESULTADO OBTENIDO CON LA SUBSTRACCION NORMAL.

```
0016 A1 0000R       MOV AX,WORD PTR VALUE 1

0019 29 06 0004R    SUB WORD PTR VALUE2,AX     ; SUBSTRAE PORCION
                                                 DE BAJO ORDEN
001D A1 0002R       MOV AX,WORD PTR VALUE1 + 2

0020 19 06 0006R    SBB WORD PTR VALUE 2 + 2 , AX ; SUBSTRAE
                                                  ; PORCION DE
                                                  ; ALTO ORDEN
```

EJEMPLO DE SUBSTRACCION DE MULTIPLE PRESICION

OPERACIONES DE SUBSTRACCION CON 32 BITS

```
  0 0 0 6   A 9 2 F          0 0 0 6   A 9 2 F
-                          -
  0 0 0 4   9 8 3 7          0 0 0 4   B 8 3 7
 ------------------         ------------------
  0 0 0 2   1 0 F 8          0 0 0 2   F 0 F 8
-                          -
BORROW        0            BORROW        1
 ------------------         ------------------
  0 0 0 2   1 0 F 8          0 0 0 1   F 0 F 8
```

EJEMPLO DE SIMPLE PRESICION

W <--- X + Y + 24 - Z

```
MOV AX,X            ; PONE [ X ] EN AX

ADD AX,Y            ; SUMA [ Y ] A AX

ADD AX,24           ; SUMA 24 A LA SUMA

SUB AX,Z            ; RESTA Z DE X + Y + 24

MOV W,AX            ; ALMACENA EL RESULTADO EN W
```

EJEMPLO NUMERICO DE SIMPLE PRESICION X + Y + 24 - Z

```
              X = A92F
            +
              Y = 9837
                 -------
             C=1   4166
            +
                   0024
                  -------
                   418A
            -
              Z = 3A2F
                  -------
                   075B
```

EJEMPLO DE DOBLE PRESICION

DPSUM <----- DP1 + DP2

```
          MOV AX,DP1          ; SUMA EL BAJO ORDEN
           ADD AX,DP2          ; [ DP1 ] + [ DP2 ] --> AX

           MOV DPSUM,AX        ; PONE LA SUMA EN DPSUM

           MOV AX,DP1+2        ; SUMA LAS PALABRAS DE ALTO ORDEN

           ADC AX,DP2+2        ; C + [ DP1+2 ] + [ DP2+2 ] --> AX

           MOV DPSUM+2,AX      ; PONE LA SUMA EN DPSUM+2
```

EJEMPLO DE DOBLE PRESICION

W <---- X + Y + 24 - Z   EN DOBLE PRESICION

```
           MOV AX,X            ;SUMA EN DOBLE PRESICION

           MOV DX,X+2          ; LOS NUMEROS EN X Y X+2

           ADD AX,Y            ; A LOS NUMEROS DOBLE PRESICION

ADC DX,Y+2          ; EN Y  Y  Y+2

ADD AX,24           ; SUMA EL NUMERO 24 EN

ADC DX,0            ; DOBLE PRESICION DEL RESULTADO

SUB AX,Z            ; RESTA EL NUMERO Z Y Z+2 EN DOBLE

SBB DX,Z+2          ; PRESICION DE LA SUMA

MOV W, AX           ; ALMACENA EL TOTAL
```

```
          MOV W+2, DX         ; EN W Y W+2
```

EJEMPLO NUMERICO DE DOBLE PRESICION X + Y + 24 - Z

```
              X = 0006  A92F
            +
              Y = 0004  9837
             -----------------
                   000B  4166
            +
                   0000  0024
             -----------------
                   000B  418A
            -
              Z = 0003  6A2F
             -----------------
                   07    D75B
```

<< ARITMETICA DE UN SIMPLE OPERANDO >>

LA INSTRUCCION DE NEGACION (NEG) ES LA OPERACION DE SIGNO INVERSO. LA INSTRUCCION CAMBIA EL SIGNO DOS COMPLEMENTO DE UN OPERANDO DE UN BYTE O UNA PALABRA.

LA INSTRUCCION INCREMENTAR (INC) SUMA UNO A EL OPERANDO, MIENTRAS LA INSTRUCCION DECREMENTA (DEC) RESTA UNO DE EL OPERANDO.

```
NEGACION       NEG EXWORD     ;  [EXWORD] <-- - [EXWORD]
INCREMENTAR    INC EXBYTE     ;  [EXBYTE] <-- [EXBYTE] + 1
DECREMENTAR    DEC SI         ;  SI <--  SI - 1
```

PUEDEN INCREMENTAR UN CONTADOR DE UN LAZO. CUANDO EL VALOR ES CERO, EL LAZO ESTA HECHO.

OPERANDO => BYTE O PALABRA

TODAS LAS BANDERAS SON AFECTADAS EXCEPTO QUE INC Y DEC NO CAMBIAN LA BANDERA DE CARRY.

NEG = RESTAR DE ZERO.

<< COMPARE >>

LA INSTRUCCION COMPARE (CMP), COMPARA DOS VALORES POR MEDIO DE RESTARLOS. CMP DESCARTA EL RESULTADO, PERO PONE LAS BANDERAS DE STATUS DE ACUERDO A EL RESULTADO. LA INSTRUCCION MODIFICA UNICAMENTE LAS BANDERAS. UN PROGRAMA PUEDE UTILIZAR LA INSTRUCCION COMPARE IGUAL QUE LA INSTRUCCION SUBSTRAER SIN UTILIZAR LA INSTRUCCION CON BORROW.

```
     VALOR 1                          VALOR 1
   -                                -
     VALOR 2                          VALOR 2
   ------------                     ------------

 CARRY =1  VALOR 2 > VALOR 1        Z = 1  VALOR 2 = VALOR 1
CMP OPR1,OPR2    ( OPR1 ) - ( OPR2 )
```

MODOS DE DIRECCIONAMIENTO. INC, DEC Y NEG NO DEBEN USAR EL MODO INMEDIATO. AL MENOS QUE EL OPR2 SEA INMEDIATO, UNO DE LOS OPERANDOS PARA UN CMP DEBE ESTAR EN UN REGISTRO; EL OTRO OPERANDO PUEDE TENER CUALQUIER MODO DE DIRECCIONAMIENTO EXCEPTO QUE OPR1 NO PUEDE SER INMEDIATO.

```
+-------------------------------------------------------------+
¦    0010 VALOR 1   ¦       0011        ¦        0011        ¦
¦  -                ¦     -             ¦      -             ¦
¦    0011 VALOR 2   ¦       0010        ¦        0011        ¦
¦  ------           ¦     ------        ¦      -------       ¦
¦  1 1111           ¦     0 0001        ¦      0 0000        ¦
¦                   ¦                   ¦                    ¦
¦   CARRY = 1       ¦ CARRY = 0  Z = 0  ¦       Z = 1        ¦
¦ VALOR 2 > VALOR 1 ¦ VALOR 1 > VALOR 2 ¦ VALOR 1 = VALOR 2  ¦
+-------------------------------------------------------------+
```

PARA HACER COMPARACIONES DE MAS DE 16 BIT UTILIZE LA INSTRUCCION DE RESTAR CON MULTIPLE PRESICION.

<< AJUSTE DECIMAL >>

UN PROGRAMA USA EL MISMO CONJUNTO DE INSTRUCCIONES ARITMETI-

CAS PARA MANIPULAR NUMEROS BCD (DECIMAL CODIFICADO EN BINARIO) COMO LO HACE PARA NUMEROS EN DOS COMPLEMENTO. SIN EMBARGO, EL RESULTADO DE LA ARITMETICA PUEDE PRODUCIR UN RESULTADO EL CUAL ES INCORRECTO PARA REPRESENTACION BCD. LAS INSTRUCCIONES DE AJUSTE DECIMAL CORRIGEN EL RESULTADO DESPUES DE LA ARITMETICA DOS COMPLEMENTO.

AJUSTE PARA ADICION DECIMAL (DAA) Y AJUSTE PARA SUBSTRACCION DECIMAL (DAS) SON USADAS POR OPERACIONES BCD EMPACADAS SOLAMENTE. EN EL BCD EMPACADO, HAY DOS DIGITOS DECIMALES EMPACADOS EN CADA BYTE DE DATOS . DAA Y DAS OPERAN SOLAMENTE EN UN BYTE DE DATOS EN EL REGISTRO AL. CON ESAS LIMITACIONES CONSTRUIDAS EN LA INSTRUCCION , NO HAY OPERANDOS CON DAA O DAS.

EJEMPLO: BCD3<----- BCD1 + BCD2

INSTRUCCION	ACCION	CONTENIDO DE AL	CF	AF
MOV AL,BCD1	(AL) <-- 34	34	--	--
ADD AL,BCD2	(AL) <-- 34 + 89	BD	0	0
DAA	AJUSTE	23	1	*
MOV BCD3,AL	(BCD3) <-- 23	23	1	*
MOV AL,BCD1+1	(AL) <-- 18	18	1	*
ADC AL,BCD2+1	(AL) <-- 18 + 27 + CF	40	0	1
DAA	AJUSTE	46	0	*
MOV BCD3+1,AL	(BCD3+1) <-- 46	46	0	*

```
                                  1<-- CF

BCD1 = 1834                     1 8     3 4
                              +
 BCD2 = 2789                      2 7     8 9
                                 -----   -----
                                  4 0     B D
                              +         +
                                  0 6     6 6  <-- AJUSTE
                                 -----   -----
                                  4 6     2 3
```

INSTRUCCION	ACCION	CONTENIDO DE AL	CF	AF
MOV AL,BCD1	(AL) <-- 34	34	--	--
SUB AL,BCD2	(AL) <-- 34 - 12	22	0	0
DAS	AJUSTE	22	0	*
MOV BCD3,AL	(BCD3) <-- 22	22	0	*
MOV AL,BCD1+1	(AL) <-- 12	12	0	*
SBB AL,BCD2+1	(AL) <-- 12 - 46 - CF	CC	1	1
DAS	AJUSTE	66	1	*
MOV BCD3+1,AL	(BCD3+1) <-- 66	66	1	*

```
                          1<-CF  0<-- CF

BCD1 = 1234                    1 2      3 4
                             -
BCD2 = 4612                    4 6      1 2
                               -----    -----
                               C C      2 2
                             -
                                6 6           <-- AJUSTE
                               -----    -----
                                6 6      2 2
```

RESULTADO = 6622 Y COMO EL CF=1 EL RESULTADO CORRECTO ES NEGATIVO E IGUAL A - 3378 EN 10'S

PARA OPERACIONES BCD, LA BANDERA DE SIGNO Y OVERFLOW NO TIENEN SIGNIFICADO.

SI LA ADICION DE 2 DIGITOS RESULTA EN UN NUMERO BINARIO ENTRE 1010 Y 1111, LAS CUALES NO SON VALIDAS EN DIGITOS BCD, ENTONCES ES UN CARRY HACIA EL SIGUIENTE DIGITO, POR LO TANTO 6 (0110) DEBEN SER SUMADOS AL DIGITO CORRIENTE.

```
    0                          0
  + 7                        0111
    6                        0110
 ------                    --------
   13                       1101
                            0110 <--- AJUSTE
                           --------
CARRY AL SIGUIENTE DIGITO 1  0011
                           ¦
    1                      +---> 1
  + 9                         1001
    9                        + 1001
------                     --------
   19                          0011
                             + 0110 <--- AJUSTE
                           --------
CARRY AL SIGUIENTE DIGITO 1  1001
                           ¦
                           +---> 1
                  +--------------+
                  ¦      97      ¦
                  ¦    + 96      ¦
                  ¦   -------    ¦
                  ¦     193      ¦
                  ¦              ¦
                  +--------------+
```

<< SUMA Y RESTA CON AJUSTE ASCII >>

LAS INSTRUCCIONES DE AJUSTE ASCII SON MUY SIMILARES A LAS INSTRUCCIONES DE AJUSTE DECIMAL. ELLOS SIGUEN LAS OPERACIONES DE NUMEROS BCD SIN EMPACAR. CUANDO UN PROGRAMA UTILIZA LAS INSTRUCCIONES DE AJUSTE DECIMAL (DAA & DAS) CON REPRESENTACION EMPACADA BCD, ESTE USA AJUSTE ASCII PARA NUMEROS BCD SIN EMPACAR. EN NUMEROS BCD SIN EMPACAR UN SIMPLE BYTE REPRESENTA LOS DECIMALES ENTRE 0 Y 9. EL ESQUEMA DE NUMEROS ES REFERIDO COMO ASCII DECIMAL PORQUE DEBIDO A LO FACIL CON LO CUAL LOS PROGRAMAS PUEDEN CONVERTIR LOS NUMEROS A Y DE REPRESENTACION ASCII (SUMAR Y RESTAR 30H RESPECTIVAMENTE (NUMEROS 0-9 EN ASCII HEXADECIMAL 30H->39H)).

DESPUES DE LA ADICION DE DOS NUMEROS DECIMALES SIN EMPACAR , UN PROGRAMA EJECUTA LA INSTRUCCION DE AJUSTE ASCII PARA ADICION

(AAA) LA CUAL CONVIERTE EL RESULTADO A LA REPRESENTACION CORRECTA SIN EMPACAR. LAS REGLAS PARA ADICION SON IDENTICAS PARA AQUELLAS CON NUMEROS EMPACADOS DECIMALES. ENTONCES LA ADICION PARA NUMEROS DECIMALES SIN EMPACAR PUEDE RESULTAR EN UN NUMERO MAYOR QUE 9, LAS INSTRUCCIONA AAA Y AAS REQUIEREN MAS QUE DE EL REGISTRO AL. PARA AAA , LOS BAJOS DIGITOS DE EL RESULTADO CORRECTO PERMANECE EL AL. SI LA ADICION DECIMAL RESULTA EN UN CARRY DE EL DIGITO BAJO , LA INSTRUCCION AAA INCREMENTA EL REGISTRO AH POR UNO. SI ESTO PASA, AAA PONE LA AF Y CF A UNO. DE OTRA MANERA SON RESETEADAS A CERO.

LAS INSTRUCCIONES DE AJUSTE ASCII DIFIEREN DE LA INSTRUCCIÓN DECIMAL EN QUE ELLAS AFECTAN EL VALOR EN EL REGISTRO AH TAMBIEN COMO UNA PUESTA DE LA BANDERA DE CARRY CUANDO ES UN CARRY DE EL DIGITO DE BAJO ORDEN.

UN PROGRAMA USA AJUSTE ASCII PARA SUBSTRACCION (AAS) SIGUIENDO A LA SUBSTRACCION DE UN NUMERO DECIMAL DE EMPACAR DE OTRO. LOS RESULTADOS DE LA OPERACION DE UN BYTE DEBEN SER COLOCADOS EN EL REGISTRO AL. EL RESULTADO DE LA INSTRUCCION DE AJUSTE ASCII PERMANECE EN EL REGISTRO AL. SI DE LA SUBSTRACCION RESULTA UN BORROW , AAS DECREMENTA EL REGISTRO AH Y PONE LAS BANDERAS AF Y CF, DE OTRA MANERA LAS BANDERAS SON BORRADAS.

```
AAA             ( AL )<--- SUMA EN AL AJUSTADA AL FORMATO
                           BCD SIN EMPACAR

                ( AH )<--- AH + CARRY DEL AJUSTE

AAS             ( AL )<--- DIFERENCIA EN AJUSTE A AL AL
                           FORMATO BCD SIN EMPACAR

                ( AH )<--- AH - BORROW DEL AJUSTE
```

EJEMPLO QUE INVOLUCRA ADICION Y SUBSTRACCION DE BCD SIN EMPACAR.

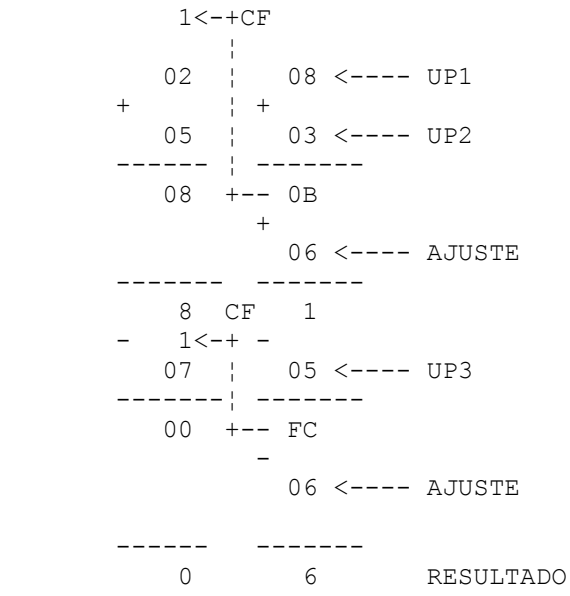
(DX)<-- UP1 + UP2 - UP3 UP = UNPACKED (SIN EMPACAR)

```
MOV AL,UP1              ; SUMA DIGITOS BAJO ORDEN

ADD AL,UP2              ; Y

AAA                     ; AJUSTA

MOV DL,AL               ; PONE LA SUMA EN DL

MOV AL,UP1 + 1          ; SUMA LOS DIGITOS DE ALTO ORDEN

ADC AL,UP2 + 1          ; CON CARRY

AAA                     ; Y AJUSTA

XCHG AL,DL              ; INTERCAMBIA AL Y DL

SUB AL,UP3              ; RESTA DIGITOS DE BAJO ORDEN
AAS                     ; DE LA SUMA Y AJUSTA

XCHG AL,DL              ; INTERCAMBIA AL Y DL

SBB AL,UP3 + 1          ; RESTA LOS DIGITOS DE ALTO ORDEN

AAS                     ; DE LA SUMA Y AJUSTA

MOV DH,AL               ; MUEVE AL A DH
```

-
-
-
-
-
-
-
-
-
-
-
-
-

EJEMPLO NUMERICO DE ADICION Y SUBSTRACCION BCD SIN EMPACAR

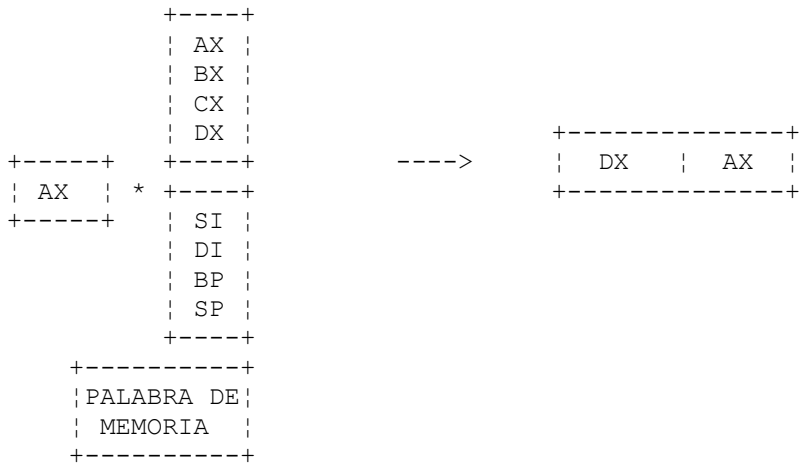
UP1 + UP2 - UP3 UP1 = 02 08 , UP2 = 05 03 Y UP3 = 07 05



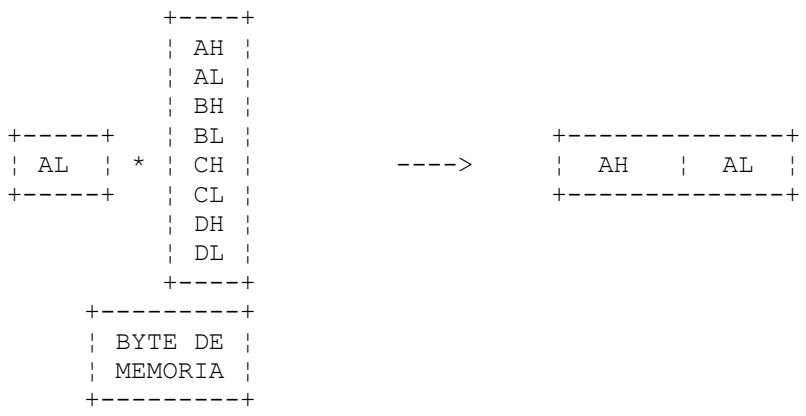
<< MULTIPLICAR >>

HAY DOS FORMAS DE LA INSTRUCCION DE MULTIPLICAR. MULTIPLICAR (MUL) MULTIPLICA DOS ENTEROS SIN SIGNO PARA PRODUCIR UN RESULTADO SIN SIGNO. LA INSTRUCCION DE MULTIPLICACION ENTEROS (IMUL) MULTIPLICA ENTEROS SIGNADOS. LA MULTIPLICACION DE ENTEROS TOMA LOS NUMEROS DOS COMPLEMENTO COMO LOS OPERANDOS, Y PRODUCE UN RESULTADO QUE TIENE LA MAGNITUD Y SIGNO CORRECTO.

OPERACIONES DE MULTIPLICACIÓN



MULTIPLICACION DE PALABRAS



MULTIPLICACION DE BYTES

PARA MULTIPLICAR 8 BITS, UN OPERANDO DEBE SER EL REGISTRO AL Y EL RESULTADO ESTA SIEMPRE EN EL REGISTRO AX. EL RESULTADO DE LA MULTIPLICACION PUEDE SER DE HASTA 16 BITS EN LONGITUD (EL MAXIMO VALOR PUEDE SER 255 x 255 = 65, 535). PARA MULTIPLICAR 16 BITS POR 16 BITS, UN OPERANDO DEBE SER EL REGISTRO AX. EL RESULTADO EL CUAL PUEDE SER DE HASTA 32 BITS DE LONGITUD (EL MAXIMO VALOR ES 65,535 x 65535 < 2 32) ES COLOCADO EN UN PAR DE REGISTROS . EL REGISTRO DX CONTIENE LOS 16 BITS MAS ALTOS DEL RESULTADO , MIENTRAS AX RETIENE LA PORCION MENOS SIGNIFICATIVA. NOTESE QUE LA MULTIPLICACION NO PERMITE UN OPERANDO INMEDIATO.

LAS BANDERAS DE REGISTROS PUESTOS SON ALGO DIFERENTES PARA MULTIPLICACION YA QUE ELLOS SON PARA OTRAS OPERACIONES ARITMETICAS . LAS UNICAS BANDERAS CON SIGNIFICADO VALIDO SON LAS BANDERAS DE CARRY Y OVERFLOW. ELLAS SON PUESTAS EN FORMA DIFERENTE POR LAS DOS INSTRUCCIONES.

MULTIPLICACION SIN SIGNO (MUL) PONE AMBAS BANDERAS SI LA PARTE ALTA DE EL RESULTADO ES DIFERENTE DE ZERO. SI DOS BYTES SON MULTIPLICADOS, LA PUESTA DE LA CF y OF SIGNIFICA QUE EL RESULTADO ES MAYOR DE 255 Y NO PUEDE SER CONTENIDO EN UN SOLO BYTE. PARA UNA MULTIPLICACION DE PALABRAS, LAS BANDERAS SON PUESTAS SI EL RESULTADO ES MAYOR DE 65, 535.

MULTIPLICACION ENTERA SIGNADA (IMUL) PONE LAS BANDERAS DE CARRY Y OVERFLOW DE ACUERDO CON EL MISMO CRITERIO. ESTO ES, LAS BANDERAS SON PUESTAS SI EL RESULTADO NO PUEDE SER REPRESENTADO EN LA PARTE BAJA DE EL RESULTADO. IMUL PONE LAS BANDERAS SI LA PARTE ALTA DE EL RESULTADO NO ES LA EXTENSION DE SIGNO DE LA PARTE BAJA DE EL RESULTADO. ESTO SIGNIFICA QUE SI EL RESULTADO ES POSITIVO,

LA PRUEBA ES LA MISMA QUE PARA MUL, ESTA PONE LAS BANDERAS SI LA PARTE ALTA DE EL RESULTADO ES DIFERENTE DE ZERO (EL BIT MAS SIGNIFICATIVO ES CERO INDICANDO RESULTADO POSITIVO). SI EL RESULTADO ES NEGATIVO, IMUL PONE LAS BANDERAS SI LA PARTE ALTA DE EL RESULTADO ES NO TODOS UNOS (PERO EL BIT MAS SIGNIFICATIVO ES UN UNO INDICANDO QUE EL RESULTADO ES NEGATIVO). POR EJEMPLO, UNA MULTIPLICACION DE UN BYTE CON UN RESULTADO NEGATIVO PONE LAS BANDERAS SI EL RESULTADO ES MENOR DE -128, ENTONCES ESTO ES QUE LOS NUMEROS MAS PEQUEÑOS SON REPRESENTABLES EN UN SIMPLE BYTE.

<< AJUSTE ASCII : MULTIPLICACION >>

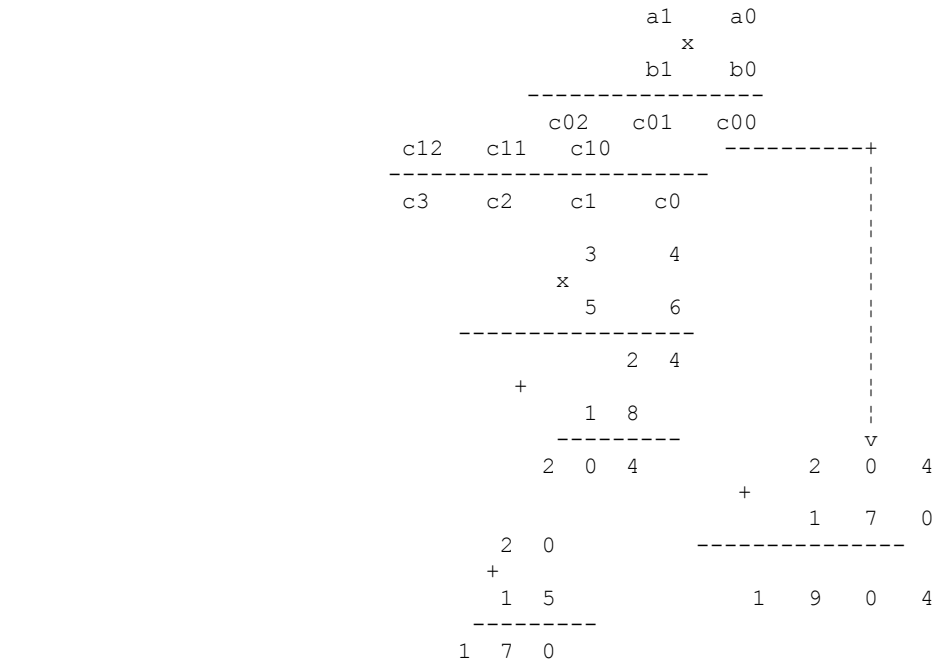
CUANDO UN PROGRAMA MULTIPLICA DOS NUMEROS DECIMALES SIN EMPACAR, EL RESULTADO EN EL REGISTRO AL ES UN NUMERO BINARIO. ENTONCES EL NUMERO BCD SIN EMPACAR MAS GRANDE ES 9, EL MAXIMO RESULTADO DE MULTIPLICACION BCD SIN EMPACAR ES 81. SIN EMBARGO, ESTE RESULTADO NO ES UNA REPRESENTACION BCD SIN EMPACAR VALIDA DE EL NUMERO. EL AJUSTE ASCII PARA MULTIPLICACION (AAM) CONVIERTE EL RESULTADO BINARIO EN UN RESULTADO DECIMAL SIN EMPACAR. LA ACCION DE AAM ES COLOCAR LOS DIGITOS DECIMALES MAS SIGNIFICATIVOS DENTRO DEL REGISTRO AH Y DEJAR LOS DIGITOS DECIMALES MENOS SIGNIFICATIVOS EN EL REGISTRO AL. POR EJEMPLO SIN UN PROGRAMA MULTIPLICA LOS VALORES 6 y 7, EL RESULTADO EN AL ES 2AH; LA NSTRUCCION AAM CONVIERTE EL RESULTADO DEJANDO 04H EN EL REGISTRO AH Y 02H EN EL REGISTRO AL O LOS NUMEROS DECIMALES SIN EMPACAR 42 EN AH : AL.

AAM PONE LAS SF Y ZF DE ACUERDO A EL RESULTADO EN AL. (UNPACKED BCD ZF INDICA QUE EL RESULTADO ES MULTIPLO DE 10

LSBYTE = 0)

AAM (AX) <---- PRODUCTO EN AL AJUSTADO AL FORMATO BCD SIN EMPACAR CON AH CONTENIENDO DIGITO DE ALTO ORDEN.

ANALIZAR EJEMPLO (A) = 34 Y (B) = 56



MULTIPLICACION

AAM AJUSTE PARA BCD SIN EMPACAR PARA LA MULTIPLICACION

(AX) <---- PRODUCTO EN AL AJUSTADO AL FORMATO BCD SIN EMPACAR. CON AH CONTENIENDO EL DIGITO DE ALTO ORDEN.

AAA AJUSTE PARA BCD SIN EMPACAR PARA LA ADICION.

(AL) <---- SUMA EN AL ES AJUSTADA A EL FORMATO EL DIGITO DE ALTO ORDEN.

AAA AJUSTE PARA BCD SIN EMPACAR PARA LA ADICION.

(AL) <---- SUMA EN AL ES AJUSTADA A EL FORMATO BCD SIN EMPACAR.

(AH) <---- (AH) + CARRY DE EL AJUSTE.
(SI AH SE INCREMENTA POR UNO AF y CF SON UNOS)

INSTRUCCION	ACCION	CONTENIDO DE AX
MOV AL,A	(AL) <--- 4	-- 04
MUL B	(AX) <--- 4 * 6	00 18
AAM	AJUSTE	02 04
MOV WORD PTR CO,AX	(CO+1 : C0)<--- 24	02 04
MOV AL,A+1	(AL) <--- 3	02 03
MUL B	(AX) <--- 3 * 6	00 12
AAM	AJUSTE	01 08
ADD AL,CO+1	(AL) <--- 2 + 8	01 0A
AAA	AJUSTE	02 00
MOV WORD PTR CO+1,AX	(CO+2 : CO+1)<--- 20	02 00
MOV AL,A	(AL) <--- 4	02 04
MUL B+1	(AX) <--- 4 * 5	00 14
AAM	AJUSTE	02 00
MOV WORD PTR C1,AX	(C1+1 : C1)<--- 20	02 00
MOV AL,A+1	(AL) <--- 3	02 03
MUL B+1	(AX) <--- 3 * 5	00 0F
AAM	AJUSTE	01 05
ADD AL,C1+1	(AL) <--- 5 + 2	01 07
AAA	AJUSTE	01 07
MOV WORD PTR C1+1,AX	(C1+2 : C1+1)<--- 17	01 07
MOV AL,CO	(AL)<--- 4	01 04
MOV AL,CO	(AL)<--- 4	01 04
MOV C,AL	(C)<--- 4	01 04
MOV AL,CO+1	(AL)<--- 0	01 00
ADD AL,C1	(AL)<--- 0 + 0	01 00
AAA	AJUSTE	01 00
MOV C+1,AL	(C + 1)<--- 0	01 00

INSTRUCCION	ACCION	CONTENIDO DE AX
MOV AL,CO+2	(AL)<--- 2	01 02
ADC AL,C1+1	(AL)<--- 2 + 7 +(CF)	01 09
AAA	AJUSTE	01 09
MOV C+2,AL	(C + 2)<--- 9	01 09
MOV AL,0	(AL)<--- 0	01 00
ADC AL,C1+2	(AL)<--- 0 + 1 +(CF)	01 01
AAA	AJUSTE	01 01
MOV C+3,AL	(C + 3)<--- 1	01 01

MULTIPLICACION DE DOS DIGITOS BCD SIN EMPACAR
<< DIVISION >>

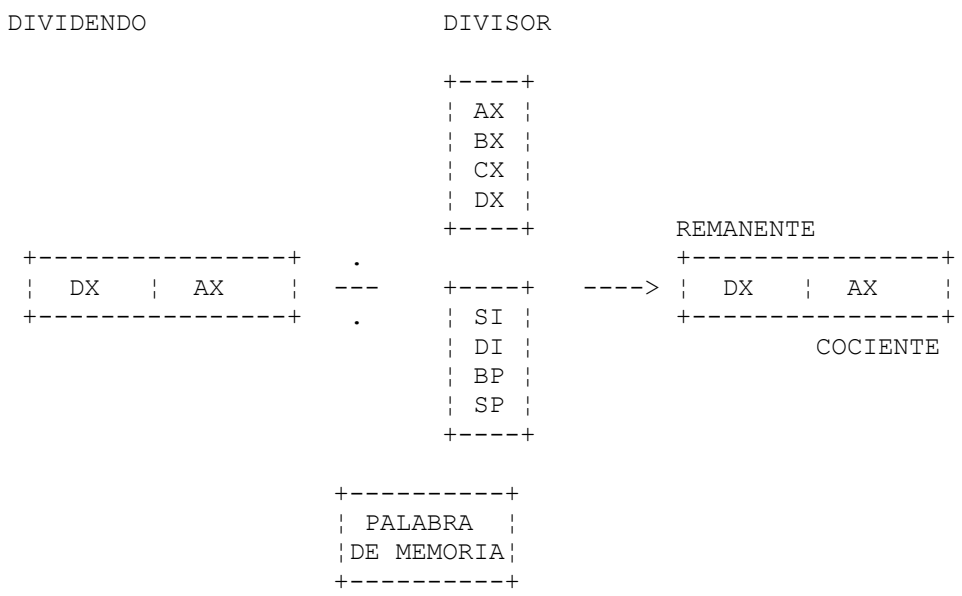
EL SET DE INSTRUCCIONES DEL 8088 CONTIENE LA DIVISION COMO PARTE DE LAS FUNCIONES ARITMETICAS. COMO CON LA MULTIPLICACION HAY DOS FORMAS DE LA DIVISION, UNA PARA NUMEROS BINARIOS SIN SIGNO, LA OTRA PARA NUMEROS SIGNADOS EN DOS COMPLEMENTOS (DIV Y IDIV RESPECTIVAMENTE). CUALESQUIER FORMA DE DIVISION PUEDE SER APLICADA A OPERACIONES DE UN BYTE O PALABRA.

LA INSTRUCCION DIVidir (DIV) LLEVA A CABO UNA DIVISION SIN SIGNO DE EL DIVIDENDO Y PRODUCE AMBOS UN COCIENTE Y UN REMANENTE. DEL MISMO MODO QUE LA MULTIPLICACION, LOS OPERANDOS DEBEN APARECER EN LOCALIDADES ESPECIFICAS DE MEMORIA. TAMBIEN COMO EN LA MULTIPLICACION, UNO DE LOS VALORES ES DOS VECES EL TAMAÑO DE EL OPERANDO NORMAL. PARA DIVISION, EL DIVIDENDO ES UN OPERANDO DE DOBLE LONGITUD. OPERACIONES DE UN BYTE DIVIDE UN DIVIDENDO DE 16 BIT POR UN DIVISOR DE 8 BIT. LA DIVISION PRODUCE 2 RESULTADOS. LA DIVISION COLOCA EL COCIENTE EN EL REGISTRO AL, Y EL REMANENTE EN

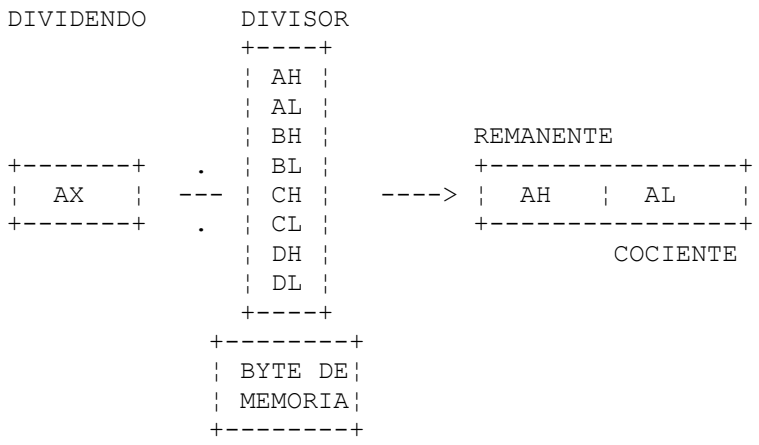
EL REGISTRO AH. ESTA COLOCACION DE LOS OPERANDOS HACE LA INSTRUCCION DE DIVISION EL COMPLEMENTO DE LA MULTIPLICACION. ESTO SIGNIFICA QUE MULTIPLICANDO EL REGISTRO AL POR UN OPERANDO DE UN BYTE, Y ENTONCES DIVIDIENDO EL REGISTRO AX POR EL MISMO OPERANDO, REGRESA AL A SU ESTADO ORIGINAL. SI EL REGISTRO AH ES PUESTO A CERO, ENTONCES NO HAY REMANENTE.

LA OPERACION CON PALABRAS DIVIDE UN DIVIDENDO DE 32 BIT POR UN DIVISOR DE 16 BIT. EL DIVIDENDO ESTA EN EL PAR DE REGISTROS DX:AX CON DX RETENIENDO LA PORCION MAS SIGNIFICATIVA Y AX RETENIENDO LA MENOS SIGNIFICATIVA. LA DIVISION DE PALABRAS COLOCA EL COCIENTE EN EL REGISTRO AX Y EL REMANENTE EN DX.

OPERACION DE DIVISION



DIVISION DE UNA PALABRA



DIVISION DE UN BYTE

```
                1  <---- COCIENTE
              +-----
DIVISOR ---> 3 ¦ 5  <---- DIVIDENDO
                2  <---- REMANENTE
```

NINGUNA DE LAS BANDERAS DE STATUS SON DEFINIDAS SIGUIENDO UNA INSTRUCCION DE DIVISION. SIN EMBARGO, UN ERROR SIGNIFICATIVO PUEDE OCURRIR DURANTE UNA DIVISION. SI UNA DIVISION POR CERO OCURRE O SI EL COCIENTE ES MAYOR DE LO QUE PUEDE CABER EN EL REGISTRO DE RESULTADOS, EL PROCESADOR NO PUEDE DAR EL RESULTADO CORRECTO. PARA UNA DIVISION DE UN BYTE EL COCIENTE DEBE SER MENOR DE 256 Y MENOR DE 65,356 PARA UNA OPERACION DE UNA PALABRA. EL PROCESADOR NO TIENE NINGUNA BANDERA DE STATUS PARA SEÑALAR ESTE ERROR. EN VEZ DE ESO, EL PROCESADOR EJECUTA UNA INTERRUPCION POR SOFTWARE LA INTERRUPCION A NIVEL 0. COMO CON LAS INTERRUPCIONES POR SOFTWARE, ESTA INTERRUPCION DE DIVISION POR CERO GUARDA LAS BANDERAS, REGISTRO DE SEGMENTO DE CODIGO, Y EL APUNTADOR DE INSTRUCCIONES EN EL STACK. EL PROCESADOR TRANSFIERE EL CONTROL A LA LOCALIDAD INDICADA POR EL APUNTADOR EN LA LOCALIDAD 0. LA RUTINA DE DIVISION POR CERO DEBERA TOMAR LA ACCION APROPIADA PARA MANEJAR LA

CONDICION DE ERROR.

LA DIVISION DE ENTEROS SIGNADOS (IDIV) DIFIERE DE LA INSTRUCCION DIV SOLAMENTE EN QUE ESTE TOMA EN CUENTA EL SIGNO DE LOS DOS OPERANDOS. SI EL RESULTADO ES POSITIVO, LAS ACCIONES SON COMO LAS DESCRITAS PARA LA INSTRUCCION DIV, EXCEPTO QUE EL MAXIMO VALOR DE EL COCIENTE ES 127 O 32,767 PARA OPERACIONES DE UN BYTE O PALABRA RESPECTIVAMENTE. SI EL RESULTADO ES NEGATIVO, EL COCIENTE ES TRUNCADO, Y EL REMANENTE TIENE EL MISMO SIGNO QUE EL DIVIDENDO. LOS COCIENTES MINIMOS PARA RESULTADOS NEGATIVOS SON -128 Y -32,768 PARA OPERACIONES DE UN BYTE O PALABRA.

EJEMPLOS DE DIVISION SIGNADA

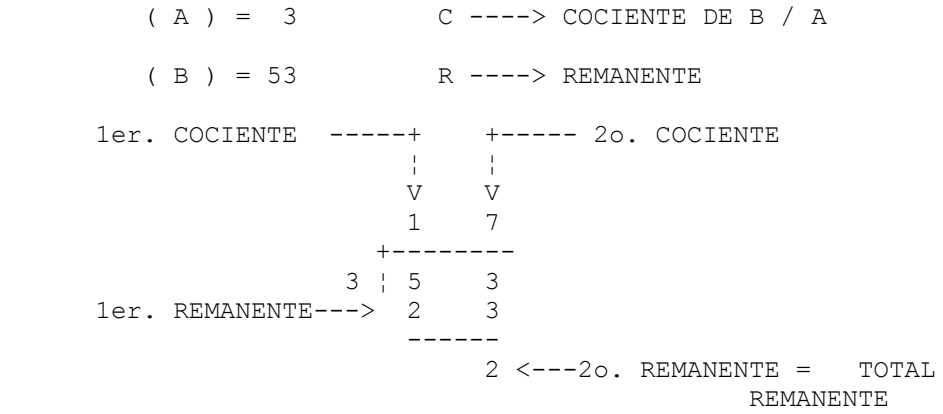
DIVIDENDO (AX)	DIVISOR	COCIENTE (AL)	REMANENTE (AH)
7	2	3	1
7	-2	-3	1
-7	2	-3	-1
-7	-2	3	-1
^			^
+---------- MISMO SIGNO ----------+			

<< AJUSTE ASCII PARA DIVISION >>

A DIFERENCIA DE LAS OTRAS INSTRUCCIONES ARITMETICAS, EL PROGRAMA DEBE EJECUTAR LA INSTRUCCION DE AJUSTE ASCII PARA DIVISION (AAD) ANTES DE LA DIVISION. LA INSTRUCCION AAD TOMA LOS DOS DIGITOS DECIMALES SIN EMPACAR EN EL REGISTRO AX (DIGITOS MAS SIGNIFICATIVOS EN AH) Y CONVIERTE ESTE A UN NUMERO BINARIO EN EL REGISTRO AL , DEJANDO UN CERO EN AH. ESTO PONE AX CON UN VALOR APROPIADO PARA DIVISION POR UN DIGITO SIMPLE DE UN NUMERO DECIMAL SIN EMPACAR. AAD PONE LOS CODIGOS DE CONDICION DE ACUERDO A EL RESULTADO EN EL REGISTRO AL. LAS BANDERAS DE PARIDAD, SIGNO Y CERO REFLEJAN EL VALOR EN AL, MIENTRAS LAS OTRAS SON DESCONOCIDAS.

SIGUIENDO CON LA DIVISION, EL SIGUIENTE PUEDE SER UN NUMERO DECIMAL SIN EMPACAR DE UN SIMPLE DIGITO. ESTO ES PORQUE NO HAY SEÑALAMIENTO DE OVERFLOW ES LA DIVISION EN ESTE CASO. EL PEOR CASO ES LA DIVISION DE 99 POR 1, PRODUCIENDO UN UN COCIENTE DE 99. EL NUMERO ES MENOR QUE EL MAXIMO PARA AMBOS DIV E IDIV, POR LO TANTO NO HAY OVERFLOW POR LA DIVISION. SIN EMBARGO ESTE NUMERO ES MAYOR QUE EL MAXIMO NUMERO DECIMAL SIN EMPACAR, EL CUAL ES 9. HAY DOS METODOS PARA MANEJAR ESTO. PRIMERO, UNA PRUEBA PUEDE SEGUIR A TODAS LAS SECUENCIAS AAD-DIV. PARA DETECTAR UN COCIENTE MAYOR QUE 9, Y ENTONCES MANEJAR APROPIADAMENTE EL OVERFLOW, O EL PROGRAMA PUEDE USAR LA INSTRUCCION AAM SIGUIENDO A LA DIVISION PARA CONVERTIR EL COCIENTE EN UN NUMERO DIGITAL DE 2 NUMEROS SIN EMPACAR. SIN EMBARGO EL PROGRAMA DEBE GUARDAR EL REMANENTE SIEMPRE ANTES DE EJECUTAR LA AAM, YA QUE AAM DESTRUYE EL REGISTRO AH. ESTA ALTERNATIVA GENERA UN RESULTADO DIGITAL DE DOS NUMEROS DE LA DIVISION DE UN VALOR DE 2 DIGITOS POR UN NUMERO DE UN SIMPLE DIGITO.

EJEMPLO DE DIVISION BCD SIN EMPACAR



```
INSTRUCCION              ACCION            CONTENIDO DE AX

 MOV AH,0                ( AH )<--- 0           00  --

 MOV AL,B+1              ( AL )<--- 5           00  05

 DIV A                   DIVIDE                 02  01

 MOV C+1,AL              ( C+1 )<--- 1          02  01

 MOV AL,B                ( AL )<--- 3           02  03

 AAD                     AJUSTE                 00  17

 DIV A                   DIVIDE                 02  07

 MOV C,AL                ( C )<--- 7            02  07

 MOV R,AH                ( R )<--- 2            02  07
```

EJEMPLO ARITMETICO

$$X = \frac{A * 2 + B * C}{D - 3}$$

```
 0000  ????   X   DW      ALMACENAMIENTO DE LAS VARIABLES
 0002  ????   A   DW
 0004  ????   B   DW
 0006  ????   C   DW
 0008  ????   D   DW

 000A  B8  0002       MOV AX,2     ; LEVANTA LA CONSTANTE
 000D  F7 2E 0002R    IMUL A       ; DX:AX = A*2
 0011  8B DA          MOV BX,DX
 0013  8B C8          MOV CX,AX    ; BX:CX = A*2
 0015  A1 0004R       MOV AX,B
 0018  F7 2E 0006R    IMUL C       ; DX:AX = B*C
 001C  03 C1          ADD AX,CX
 001E  13 D3          ADC DX,BX    ; DX:AX = A*2 + B*C

0020  8B 0E 0008R   MOV CX,D
0024  83 E9 03      SUB CX,3     ; CX = D - 3
0027  F7 F9         IDIV CX      ; AX = ( A*2 + B*C )/( D - 3 )
0029  A3 0000R      MOV X, AX    ; ALMACENA EL RESULTADO COCIENTE
```

3.3.- INSTRUCCIONES LOGICAS
AND, OR, OR EXCLUSIVO (XOR) Y NOT

VALOR	NOT (VALOR)
0	1
1	0

TABLA DE VERDAD DE LA OPERACION NOT

X	Y	X AND Y	X OR Y	X XOR Y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

TABLA DE VERDAD DE LAS OPERACIONES LOGICAS AND, OR Y XOR

OBSERVACIONES A LAS TABLAS DE VERDAD:

- EN EL OPERADOR AND LOS VALORES CON BIT 0 PRODUCE COMO RESULTADO COLOCAR 0 EN EL DESTINO Y CON BIT 1 REFLEJA EL VALOR.
- XOR PRODUCE 1 DE SALIDA UNICAMENTE SI UNO DE LOS OPERANDOS ES UNO Y EL OTRO ES UN CERO.
- CF Y OF SON PUESTAS A 0 YA QUE NO SON OPERACIONES ARITMETICAS.
- SF, ZF Y PF REFLEJAN EL RESULTADO DE LA OPERACION.
- OPERADOR NOT NO AFECTA NINGUNA DE LAS BANDERAS.

POR EJEMPLO SI (DL) = 10001010

NOT DL

ES EJECUTADO, ENTONCES

NUEVO (DL) = 01110101

NOT NO PUEDE TENER DIRECCIONAMIENTO INMEDIATO.

NOMBRE	NEMONICO Y FORMATO	DESCRIPCION
NOT	NOT OPR	(OPR) <--- ($\overline{OPR}$)
OR	OR DST,SRC	(DST) <--- (DST) V (SRC)
AND	AND DST,SRC	(DST) <--- (DST) /\ (SRC)
OR EXCLUSIVO	XOR DST,SRC	(DST) <--- (DST) $\overline{V}$ (SRC)
TEST	TEST OPRI, OPR2	OPR1 /\ OPR2 ;EL RESULTADO NO SE PONE, SOLAMENTE LAS BANDERAS SON AFECTADAS

TABLA DE UTILIZACION DE LAS FUNCIONES LOGICAS

AL MENOS QUE EL OPERANDO FUENTE (SRC) SEA INMEDIATO, ENTONCES

UNO DE LOS OPERANDOS DEBE SER UN REGISTRO. EL OTRO OPERANDO PUEDE TENER CUALQUIER MODO DE DIRECCIONAMIENTO.

SI : (AL) = 10010110

(LOG DATA) = 01011010

ENTONCES LA INSTRUCCION :

OR AL, LOG DATA

PRODUCIRA :

NUEVO (AL) = 11011110

LA INSTRUCCION :

AND AL, LOG DATA

PRODUCIRA :

NUEVO (AL) = 00010010

Y :

XOR AL, LOG DATA

RESULTARA EN :

NUEVO (AL) = 11001100

LA INSTRUCCION TEST ES LO MISMO QUE LA AND EXCEPTO QUE ESTA NO PONE EL RESULTADO EN NINGUN LADO. IGUAL QUE LA INSTRUCCION CMP ES UTILIZADA SOLAMENTE PARA PONER BANDERAS.

INSTRUCCIONES LOGICAS SON USADAS PARA DE ACUERDO AL PATRON DE BITS EN EL OPERANDO DESTINO.

- PONER SELECTIVAMENTE (OR)
- CAMBIAR (NOT)
- BORRAR (AND)
- PROBAR BITS EN EL OPERANDO (TEST)
- COMPLEMENTAR BITS (XOR)

```
          +------------------------ BITS A SER PUESTOS
          ¦  ¦     ¦        ¦
       0  1  1  0  1  0  0  1  <---- MASCARILLA
  OR
       1  0  1  0  0  0  0  1  <---- DESTINO
    --------------------------
       1  1  1  0  1  0  0  1  <---- NUEVO CONTENIDO DE DESTINO
       ¦        ¦     ¦  ¦
       +---------------------------- BITS DEJADOS SIN CAMBIO


       +---------------------------  BITS A SER BORRADOS
       ¦     ¦        ¦
       0  1  0  1  1  0  1  1  <---- MASCARILLA
AND
       1  0  0  1  0  1  0  1  <---- DESTINO
     ------------------------
       0  0  0  1  0  0  0  1  <---- NUEVO CONTENIDO DEL DESTINO
          ¦     ¦  ¦     ¦  ¦
          +------------------------ BITS DEJADOS SIN CAMBIO


      0  1  1  0  1  1  0  1  <---- DESTINO

TEST  1  0  0  1  0  0  1  0  <---- COMPLEMENTACION DEL DESTINO
(AND)
      0  1  1  0  0  1  0  0  <---- MASCARILLA
    ------------------------
      0  0  0  0  0  0  0  0  <---- RESULTADO
         ¦  ¦        ¦
         +------------------------ BITS PROBADOS (Z=1 INDICA QUE
                                   LOS BITS 2, 5 y 6 = 1)
```

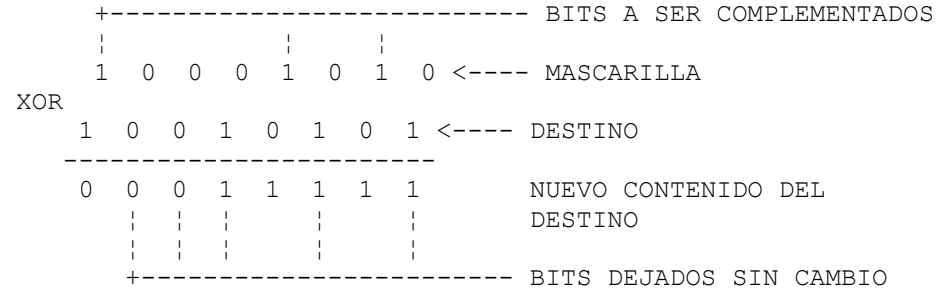
```
OR DL,00000101B          ; PONE LOS BITS 2 Y 0

XOR DL,01000010B         ; COMPLEMENTA LOS BITS 6 Y 1

AND DL,11100111B         ; BORRA LOS BITS 3 Y 4

MOV AL,DL                ; DUPLICA EL RESULTADO EN AL

NOT AL                   ; Y TOMA LA RAMA

TEST AL,10000010B        ; PARA SALIR SI AMBOS

JZ EXIT                  ; BITS 7 Y 1 SON PUESTOS
```

EJEMPLO DE UTILIZACION DE LAS FUNCIONES LOGICAS

TABLA QUE MUESTRA COMO LA FUNCION XOR ES UTILIZADA PARA

COMPLEMENTAR BITS SELECTIVAMENTE

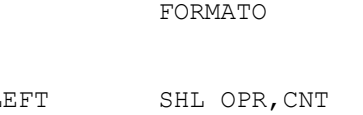
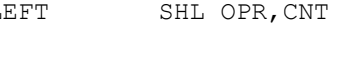
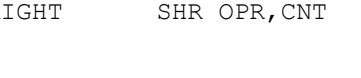
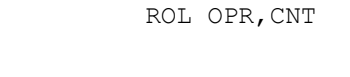
	A	B	A XOR B	
	0	0	0	
SIN CAMBIO	0	1	1	
	1	0	1	COMPLEMENTA
	1	1	0	



INSTRUCCIONES DE CORRIMIENTO Y ROTACION

UNA OPERACION DE CORRIMIENTO MUEVE TODOS LOS BITS EN UNA AREA DE DATOS, YA SEA A LA DERECHA O A LA IZQUIERDA (EJEMPLO DE UN BANCO DE IGLESIA SI LA BANCA ESTA LLENA SE CAE UNO).

SHIFT (CORRIMIENTOS) Y ROTACIONES OPERAN SOBRE CUALESQUIERA YA SEA UN BYTE O UNA PALABRA DE DATOS. EL OPERANDO PUEDE SER UN REGISTRO O UNA LOCALIDAD DE MEMORIA.

NOMBRE	NEMONICO Y FORMATO	DESCRIPCION *
SHIFT LOGICAL LEFT	SHL OPR,CNT	
SHIFT ARITHMETIC LEFT	SAL OPR,CNT	
SHIFT LOGICAL RIGHT	SHR OPR,CNT	
SHIFT ARITHMETIC RIGHT	SAR OPR,CNT	
ROTATE LEFT	ROL OPR,CNT	
ROTATE RIGHT	ROR OPR,CNT	
ROTATE LEFT THROUGH CARRY	RCL OPR,CNT	
ROTATE RIGHT THROUGH CARRY	RCR OPR,CNT	

* NUMEO DE BITS DE POSICION CORRIDOS ES DETERMINADO POR CNT.

SAR => EL SHIFT ARITMETICO SELECCIONA EL BIT DE ENTRADA PARA PRESERVAR EL BIT DE SIGNO DEL OPERANDO.

SAR Y SAL => EL TERMINO ARITMETICO ES PORQUE UN CORRIMIENTO EQUIVALE A MULTIPLICAR O DIVIDIR POR 2.

EN EL SISTEMA DECIMAL, O BASE 10, AGREGAR UN CERO AL FIN DE UN NUMERO ES MULTIPLICAR POR 10.

CORRIMIENTO A LA IZQUIERDA INTRODUCIENDO UN 0 EN EL BIT MAS BAJO EQUIVALE A MULTIPLICAR POR 2. UN CORRIMIENTO A LA IZQUIERDA DE 3 BITS EQUIVALE A MULTIPLICAR POR 8.

CORRER UN NUMERO A LA DERECHA EQUIVALE A DIVIDIR POR 2.

LOS CORRIMIENTOS AFECTAN LA OF Y CF. LA OF ES INDEFINIDA PARA CORRIMIENTOS MAYORES QUE UNO. PERO PARA CORRIMIENTOS DE UNO EN UNO, UNA INSTRUCCION DE CORRIMIENTO/ROTACION PONE LA OF SI EL SIGNO DE EL NUMERO CAMBIA CON EL RESULTADO. SI EL BIT DE ALTO ORDEN NO HA CAMBIADO OF = 0.

PF, SF Y ZF SON PUESTAS POR INSTRUCCIONES DE CORRIMIENTO PERO DEJADAS SIN CAMBIO POR INSTRUCCIONES DE ROTACION.

MODOS DE DIRECCIONAMIENTO. OPR PUEDE TENER CUALESQUIER MODO DE DIRECCIONAMIENTO EXCEPTO INMEDIATO; CNT DEBE SER 1 O CL.

OF = 0 SI LOS 2 MSB DE EL OPERANDO SON IGUALES.

OF = 1 EN OTRO CASO.

EJEMPLO DE INSTRUCCIONES VALIDAS Y NO VALIDAS

```
SHR QUAN[SI],1              ES VALIDA

MOV CL,6

SAL DATA[BX][DI],CL         ES VALIDA

SHR AL,3                    NO ES VALIDA
                            ( CNT = 1 SOLAMENTE )
```

```
          SHR AL, BL                    NO ES VALIDA
                                        ( CL UNICO UTILIZADO )
```

ASUMIR INICIALMENTE (DL) = 8D, (CL) = 3 Y CF = 1 E INDIQUE EL RESULTADO DE VARIOS CORRIMIENTOS Y ROTACIONES.

INSTRUCCION		RESULTADO	
SHL DL,CL	CF = 0	0110 1000	
SAL DL,CL	CF = 0	0110 1000	
SHR DL,CL		0001 0001	CF = 1
SAR DL,CL		1111 0001	CF = 1
ROL DL,CL	CF = 0	0110 1100	
ROR DL,CL		1011 0001	CF = 1
RCL DL,CL	CF = 0	0110 1110	
RCR DL,CL		0111 0001	CF = 1

INICIALMENTE (DL) = 1000 1101

(CL) = 0000 0011

(CF) = 1

CONVERSION DE UN NUMERO BCD SIN EMPACAR DE 16 DIGITOS A UN NUMERO EMPACADO BCD.

```
          MOV DX,8      ; PONE EL CONTADOR DEL LAZO DX A 8
          MOV CL,4      ; PONE EL CONTADOR DE CORRIMIENTOS A 4
          MOV SI,0      ; INDICE A CERO
          MOV DI,SI     ; EN SI Y DI
CONVERT:  MOV AX, WORD THE UNPACKED [SI]; TRAE DOS DIGITOS HACIA
                                        ; AX.
          SHL AL,CL     ; Y LOS EMPACA
          SHR AX,CL
```

```
        MOV PACKED[DI],AL ; ALMACENA RESULTADOS

        ADD SI,2      ; INCREMENTA INDICES

        INC DI        ; EN SI Y DI

        DEC DX        ; DECREMENTA EL CONTADOR DEL LAZO

        JNZ CONVERT   ; REPITE SI EL CONTADOR ES DIFERENTE A
                      ; CERO
```

UNPACKED ES UTILIZADA COMO VARIABLE DE DOS BYTES Y PACKED ES DEFINIDA COMO VARIABLES DE UN BYTE.

3.4.- INSTRUCCIONES PARA MANEJO DE STRINGS (ALFANUMERICOS)

UN TIPO COMUN DE DATOS ES UN STRING (HILERA ALFANUMERICA) DE CARACTERES O NUMEROS QUE UN PROGRAMA MANIPULA COMO UN GRUPO. EL PROGRAMA MUEVE EL STRING DE UN LUGAR A OTRO, COMPARA ESTE CON OTRO STRING Y BUSCA ESTE POR VALORES ESPECIFICOS.

UN PROGRAMA PUEDE HACER OPERACIONES CON STRING EN CUALESQUIER BYTE O PALABRA DE DATOS. LAS INSTRUCCIONES STRING NO USAN LOS MODOS DE DIRECCIONAMIENTO QUE EL RESTO DE LAS INSTRUCCIONES DE MANEJO DE DATOS USAN. EN EFECTO, LAS INSTRUCCIONES STRINGS SON MUY PRECISAS EN SU FORMA DE DIRECCIONAMIENTO, NO SON PERMITIDAS LAS VARIACIONES.

INSTRUCCIONES STRINGS DIRECCIONA LOS OPERANDOS CON CUALESQUIERA DE LA COMBINACION DE REGISTROS DS:SI O ES:DI . PARA OPERANDOS FUENTE USE DS:SI Y PARA OPERANDOS DESTINO USE ES:DI, DE AQUI LOS NOMBRES DE REGISTRO INDICE FUENTE Y REGISTRO INDICE DESTINO. MUCHOS TROZOS DE DATOS COMPONEN UN STRING, PERO LAS INSTRUCCIONES STRING TRABAJAN SOBRE SOLAMENTE UN ELEMENTO A LA VEZ. ESTO HACE NECESARIO QUE EL PROGRAMA TRABAJE A TRAVEZ DEL STRING UN ELEMENTO A LA VEZ. EL INCREMENTO AUTOMATICO O DECREMENTO PERMITEN UN RAPIDO PROCESAMIENTO DE DATOS STRINGS. LA BANDERA DE DIRECCION EN EL REGISTRO DE STATUS CONTROLA LA DIRECCION DE LA MANIPULACION DE STRINGS. CUANDO LA BANDERA DE DIRECCION ES PUESTA A "1" LA DIRECCION SE DECREMENTA, MIENTRAS QUE ELLA SE INCREMENTA SI LA BANDERA ES BORRADA A "0". EL TAMAÑO DE LOS OPERANDOS DETERMINA LA CANTIDAD DE INCREMENTO/DECREMENTO.

INSTRUCCIONES STRING DE UN BYTE CAMBIAN LA DIRECCION POR UNO

DESPUES DE CADA OPERACION STRING, MIENTRAS QUE LAS INSTRUCCIONES STRING DE PALABRAS CAMBIAN LA DIRECCION POR DOS. ESTO DEJA EL APUNTADOR APUNTADO A EL SIGUIENTE ELEMENTO EN LA SIGUIENTE OPERACION STRING.

B = BYTE
W = WORD

INSTRUCCIONES STRINGS

NOMBRE	NEMONICO Y FORMATO	DESCRIPCIÓN
MOVER STRINGS	MOVS DST,SRC	((DI))<-- ((SI))
		OPERANDO SOBRE UN BYTE
MOVER UN STRING DE UN BYTE	MOVSB	(SI)<-(SI) ± 1 ; (DI)<-(DI) ± 1
		OPERANDO SOBRE UNA PALABRA
MOVER UN STRING DE UNA PALABRA	MOVSW	(SI)<-(SI) ± 2 ; (DI)<-(DI) ± 2
COMPARAR STRINGS	CMPS SRC,DST	((SI))-((DI))
		OPERANDO SOBRE UN BYTE
COMPARANDO STRING DE UN BYTE	CMPSB	(SI)<-(SI) ± 1 ; (DI)<-(DI) ± 1
		OPERANDO SOBRE UNA PALABRA
COMPARANDO STRING DE UNA PALABRA	CMPSW	(SI)<-(SI) ± 2 ; (DI)<-(DI) ± 2
BUSQUEDA DE STRINGS	SCAS DST	OPERANDO SOBRE UN BYTE
BUSQUEDA DE UN STRING DE UN BYTE	SCASB	(AL)- ((DI)), (DI) <- (DI) ± 1
		OPERANDO SOBRE UNA PALABRA
BUSQUEDA DE UN STRING DE UNA PALABRA	SCASW	(AX)-((DI)), (DI) <- (DI) ± 2

NOMBRE	NEMONICO Y FORMATO	DESCRIPCION
CARGANDO STRINGS	LODS SRC	OPERANDO SOBRE UN BYTE
CARGANDO UN STRING DE UN BYTE	LODSB	(AL)<-((SI)), (SI)<-(SI) ± 1
		OPERANDO SOBRE UNA PALABRA
CARGANDO UN STRING DE UNA PALABRA	LODSW	(AX)<-((SI)), (SI)<-(SI) ± 2
ALMACENANDO STRINGS	STOS DST	OPERANDO SOBRE UN BYTE
ALMACENANDO UN STRING DE UN BYTE	STOSB	((DI))<-(AL), (DI) <- (DI) ± 1
		OPERANDO SOBRE UNA PALABRA
ALMACENANDO UN STRING	STOSW	((DI))<-(AX),
DE UNA PALABRA		(DI) <- (DI) ± 2

INCREMENTO (+) ES USADO SI DF=0 Y DECREMENTO (-) ES USADO SI DF=1

BANDERAS: CMPS Y SCAS AFECTAN TODAS LAS CONDICIONES DE (CF,OF Y ZF) LAS BANDERAS Y MOVS, LODS Y STOS NO AFECTAN LAS BANDERAS.

MODOS DE DIRECCIONAMIENTO: OPERANDOS SON IMPLICADOS.

<< REPITA LO PREFIJADO >>

EJEMPLO: REP STOSB
EL STOBS SE REPITE HASTA QUE CX ES DECREMENTADO A 0.
STOBS OPERA NORMALMENTE (EN ESTE CASO LLENADO).
(CX) <---- (CX) - 1
LOCALIDAD APUNTADA POR ES:DI

MOVS => COMBINACION DE LODS Y STOS.

MOVS Y CMPS SON LAS UNICAS DOS INSTRUCCIONES DE MEMORIA QUE PUEDEN TRATAR CON 2 OPERANDOS DE MEMORIA.

REP MOVSB <= INSTRUCCION MUY PODEROSA PARA MOVER BLOQUES DE DATOS

REPEAT WHILE EQUAL (REPE) PARA USARSE CON SCAS O CMPS.

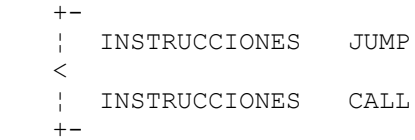
REPEAT WHILE NO EQUAL (REPNE) PARA RAPIDA BUSQUEDA EN UNA TABLA.

SI REPE ES ESPECIFICADO, LA INSTRUCCION EJECUTA HASTA QUE AL (O AX) NO SEAN IGUALES A LA LOCALIDAD DE MEMORIA O HASTA QUE CX SEA IGUAL A CERO, LO QUE OCURRA PRIMERO ENTRE LA INSTRUCCION.

REPNE ES JUSTAMENTE LO OPUESTO, LA BUSQUEDA CONTINUA HASTA QUE EL ACUMULADOR IGUALE EL VALOR EN EL REGISTRO O CX SEA A IGUAL A CERO SIN ALCANZAR LA IGUALDAD.

3.5.- INSTRUCCIONES DE TRANSFERENCIA DE CONTROL

LAS INSTRUCCIONES DE TRANSFERENCIA DE CONTROL SON USADAS PARA MANDAR EL PROGRAMA A EJECUTARSE EN UNA DIFERENTE SECCION DEL PROGRAMA.



INSTRUCCIONES JUMP (BRINCAR). TRANSFIERE EL CONTROL A LA LOCALIDAD NOMBRADA SIN GUARDAR DIRECCION DE RETORNO.

INSTRUCCIONES CALL (LLAMADA). INVOLUCRA UNA SUBRUTINA.

LAS INSTRUCCIONES JUMP PERMITEN A LA COMPUTADORA PENSAR. LAS INSTRUCCIONES CONDICIONALES PUEDEN PROBAR EL RESULTADO DE UNA OPERACION PREVIA, Y MODIFICAR EL FLUJO DEL PROGRAMA BASADO EN ESTE RESULTADO. SI LAS INSTRUCCIONES JUMP CONDICIONAL NO EXISTIERAN, LOS PROGRAMAS DE COMPUTADORA SERIA MUCHO MAS SIMPLES, PERO TAMBIEN MUCHO MENOS PRODUCTIVOS.

UNA INSTRUCCION JUMP MODIFICA EL APUNTADOR DE INSTRUCCIONES (IP) Y POSIBLEMENTE EL REGISTRO DEL SEGMENTO DE CODIGO (CS). ESTOS REGISTROS INDICAN LA SIGUIENTE INSTRUCCION A SER EJECUTADA.

SI UNA INSTRUCCION JUMP MODIFICA SOLAMENTE EL REGISTRO IP, ESTE ES UN JUMP INTRASEGMENTO (64K), ENTONCES EL BRINCO TOMA LUGAR DENTRO DEL SEGMENTO (JUMP NEAR (CERCA)). SI EL JUMP MODIFICA EL REGISTRO CS, ESTE ES UN INTERSEGMENTOS O JUMP FAR (LEJOS).

SIMILARMENTE UN CALL A UN PROCEDIMIENTO LEJOS (FAR) GUARDA AMBOS VALORES EL CS E IP, MIENTRAS UN CALL A UNA SUBRUTINA CERCA

(NEAR) DEJA SOLAMENTE EL VALOR DE IP EN EL STACK.

LA INSTRUCCION RETURN (REGRESO) DEBE ESPECIFICAR LA MANERA COMO EL CONTROL VINO A LA RUTINA PARA QUE REGRESE A LA LOCALIDAD CORRECTA (CERCA O LEJOS).

DIRECCIONAMIENTO DEL JUMP

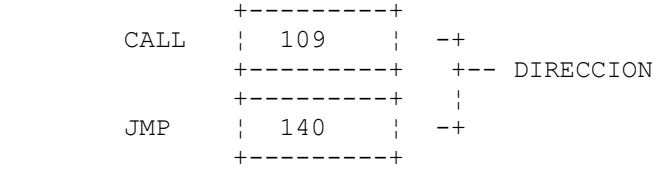
SI LA INSTRUCCION CONTIENE LA DIRECCION DEL BLANCO DE EL JUMP O CALL COMO PARTE DE LA INSTRUCCION (COMO EL DATO EN UNA INSTRUCCION INMEDIATA), ESTE ES UN JUMP DIRECTO. SI LA INSTRUCCION TIENE LA DIRECCION EN UN REGISTRO O LOCALIDAD DE MEMORIA, ESTE ES UN JUMP INDIRECTO, ENTONCES LA INSTRUCCION REQUIERE TRAER LA DIRECCION BUSCADA DE ALGUNA LOCALIDAD INTERMEDIA. EL PROGRAMA NO PUEDE TOMAR LA RAMA DIRECTAMENTE A LOCALIDAD DESEADA, PERO DEBE IR AHI INDIRECTAMENTE.

CUANDO HAY UN JUMP DIRECTO A LA LOCALIDAD CERCANA COMO 127 BYTES HACIA ADELANTE O 128 BYTES HACIA ATRAS DENTRO DEL PROGRAMA LA COMPUTADORA HACE UN JUMP RELATIVO.

TRANSFERENCIA INCONDICIONAL (JUMP'S Y CALL'S)

UNA INSTRUCCION INCONDICIONAL ES UNA QUE TRANSFIERE EL CONTROL CADA VEZ QUE ESTA SE EJECUTA. EN CONTRASTE UNA INSTRUCCION CONDICIONAL PRUEBA EL ESTADO CORRIENTE DE LA MAQUINA PARA DETERMINAR SI SE TRANSFIERE O NO EL CONTROL.

TODAS LAS INSTRUCCIONES CALL SON INCONDICIONALES:



INSTRUCCION RETURN (REGRESO DE UNA SUBRUTINA)

```
RET
```

JUMPS CONDICIONALES

PRUEBA BANDERAS. PRUEBA EL RESULTADO DE UNA OPERACION ARITMETICA O LOGICA PREVIA.

PRUEBA DE LOOP. CONTROLA LA INTERACCION DE UNA SECCION DEL CODIGO.

TODOS LOS JUMP CONDICIONALES TIENEN UN DESPLAZAMIENTO DE UN BYTE. SI UN JUMP CONDICIONAL ES A UNA LOCALIDAD A MAS DE 128 BYTES, UNA DIFERENTE CONSTRUCCION DEBE SER USADA. COMO UN EJEMPLO SUPONER QUE EL PROGRAMA NECESITA EJECUTAR LA ETIQUETA CERO SI LA BANDERA DE CERO ES PUESTA. ESTA ETIQUETA ES DE MAS DE 128 BYTES. EL CODIGO PARA MANEJAR ESTA OCURRENCIA SE MIRA ASI:

```
          JNZ       CONTINUE

          JMP       ZERO

CONTINUE   ¨
           ¨
           ¨
```

CODIGOS PARA PROBAR LA CONDICION

INSTRUCCIONES DE RAMA CONDICIONALES

NOMBRE	NEMONICO Y FORMATO	NEMONICO ALTERNO	CONDICION A PROBAR *
BRANCH ON ZERO, OR EQUAL	JZ OPR	JE	ZF = 1
BRANCH ON NONZERO, OR NOT EQUAL	JNZ OPR	JNE	ZF = 0
BRANCH ON SIGN SET	JS OPR		SF = 1
BRANCH ON SIGN CLEAR	JNS OPR		SF = 0
BRANCH ON OVERFLOW	JO OPR		OF = 1

NOMBRE	NEMONICO Y FORMATO	NEMONICO ALTERNO	CONDICION A PROBAR *
BRANCH ON NO OVERFLOW	JNO OPR		OF = 0
BRANCH ON PARITY SET, OR EVEN PARTY	JP OPR	JPE	PF = 1
BRANCH ON PARITY CLEAR, OR ODD PARITY	JNP OPR	JPO	PF = 0
BRANCH ON BELOW, OR NOT ABOVE OR EQUAL (UNSIGNED)	JB OPR	JNAE, JC	CF = 1
BRANCH ON NOT BELOW,OR ABOVE OR EQUAL (UNSIGNED)	JNB OPR	JAE, JNC	CF = 0
BRANCH ON BELOW OR EQUAL ,OR NOT ABOVE (UNSIGNED)	JBE OPR	JNA	CF V ZF = 1
BRANCH ON NOT BELOW OR EQUAL, OR ABOVE (UNSIGNED)	JNBE OPR	JA	CF V ZF = 0
BRANCH ON LESS,OR NOT GREATER OR EQUAL (SIGNED)	JL OPR	JNGE	SF $\overline{V}$ OF = 1
BRANCH ON NOT LESS, OR GREATER OR EQUAL (SIGNED)	JNL OPR	JGE	SF $\overline{V}$ OF = 0
BRANCH ON LESS OR EQUAL, OR NOT GREATER (SIGNED)	JLE OPR	JNG	(SF $\overline{V}$ OF) V ZF = 1
BRANCH ON NOT LESS OR, EQUAL, OR GREATER (SIGNED)	JNLE OPR	JG	(SF $\overline{V}$ OF) V ZF = 0

V = OR , $\overline{V}$ = XOR (OR EXCLUSIVO)

* SI LA CONDICION DE PRUEBA ES ALCANZADA (IP) <-- (IP) MAS EXTENSION DE SIGNO, DE OTRA MANERA (IP) PERMANECE SIN CAMBIO Y EL PROGRAMA CONTINUA EN SECUENCIA.

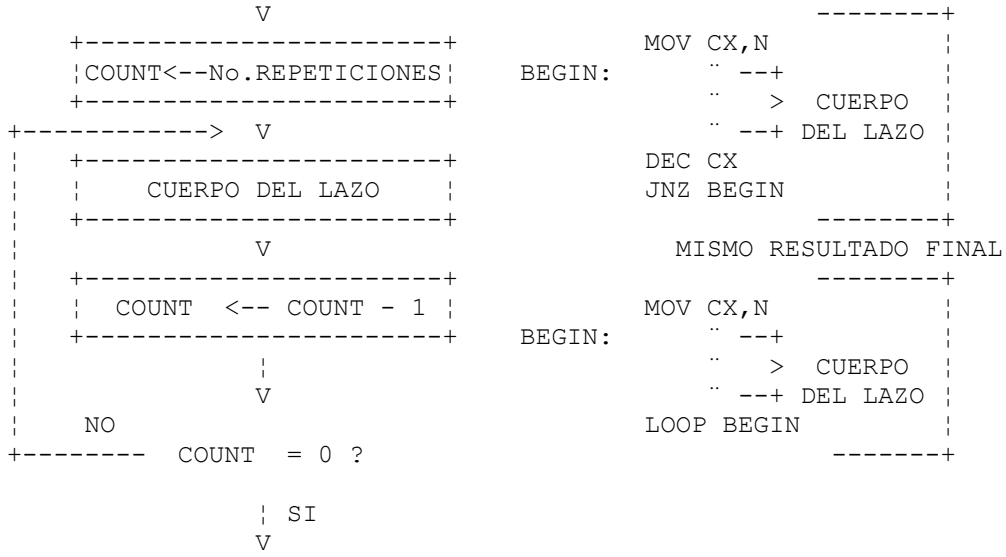
FLAGS: NINGUNA BANDERA ES AFECTADA.

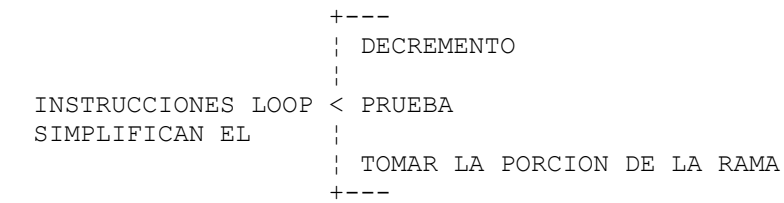
MODO DE DIRECCIONAMIENTO. MODO ES RELATIVO A IP. OPR DEBE REPRESENTAR UNA ETIQUETA QUE ESTE DENTRO DE -128 A 127 BYTES DE LA INSTRUCCION SIGUIENTE A LA INSTRUCCION DE RAMA.

```
     0050      AGAIN     INC CX
     0052                ADD AX,[BX]
```

```
      0054                JNS AGAIN

      0056      NEXT:     MOVE RESULT,CX

 DIRECCION EFECTIVA DE LA RAMA                        0050

 ( IP ) CUANDO LA DESICION DE RAMA JNS ES HECHA       0056
                                                     -------
                                                        -6
           0000  0110    + 6
           1111  1001    + 6 EN 1'S
         +           1
         ---------------
           1111  1010    + 6 EN 2'S
            F     A         = FA ( HEXADECIMAL )

                          CANTIDAD A SUMAR A LA DIRECCION 0056 = IP
                          PARA QUE IP SE CAMBIE A 0050 ( AGAIN )
```

NOMBRE	INSTRUCCIONES DE LAZO NEMONICO Y FORMATO	NEMONICO ALTERNO	CONDICION DE PRUEBA
LOOP	LOOP OPR		CX Ï 0
LOOP WHILE ZERO, OR EQUAL	LOOPZ OPR	LOOPE	ZF=1 AND (CX) Ï 0
LOOP WHILE NONZERO, OR NOT EQUAL	LOOPNZ OPR	LOOPNE	ZF=0 AND (CX) Ï 0
BRANCH ON CX	JCXZ OPR		(CX) = 0





MODO DE DIRECCIONAMIENTO : RELATIVO

OPR DEBE ESTAR EN UN RANGO DE +127 A -128 (RELATIVO) = 1 BYTE

CX <--- CX - 1 , SI LA CONDICION DE PRUEBA ES ALCANZADA ENTONCES

IP <--- IP + OPERANDO

SI LA CONDICION NO ES ALCANZADA IP NO CAMBIA

EJEMPLO : BUSQUEDA DE UN ESPACIO EN BLANCO EN UN STRING
(CARACTER 32 DECIMAL = 20H)

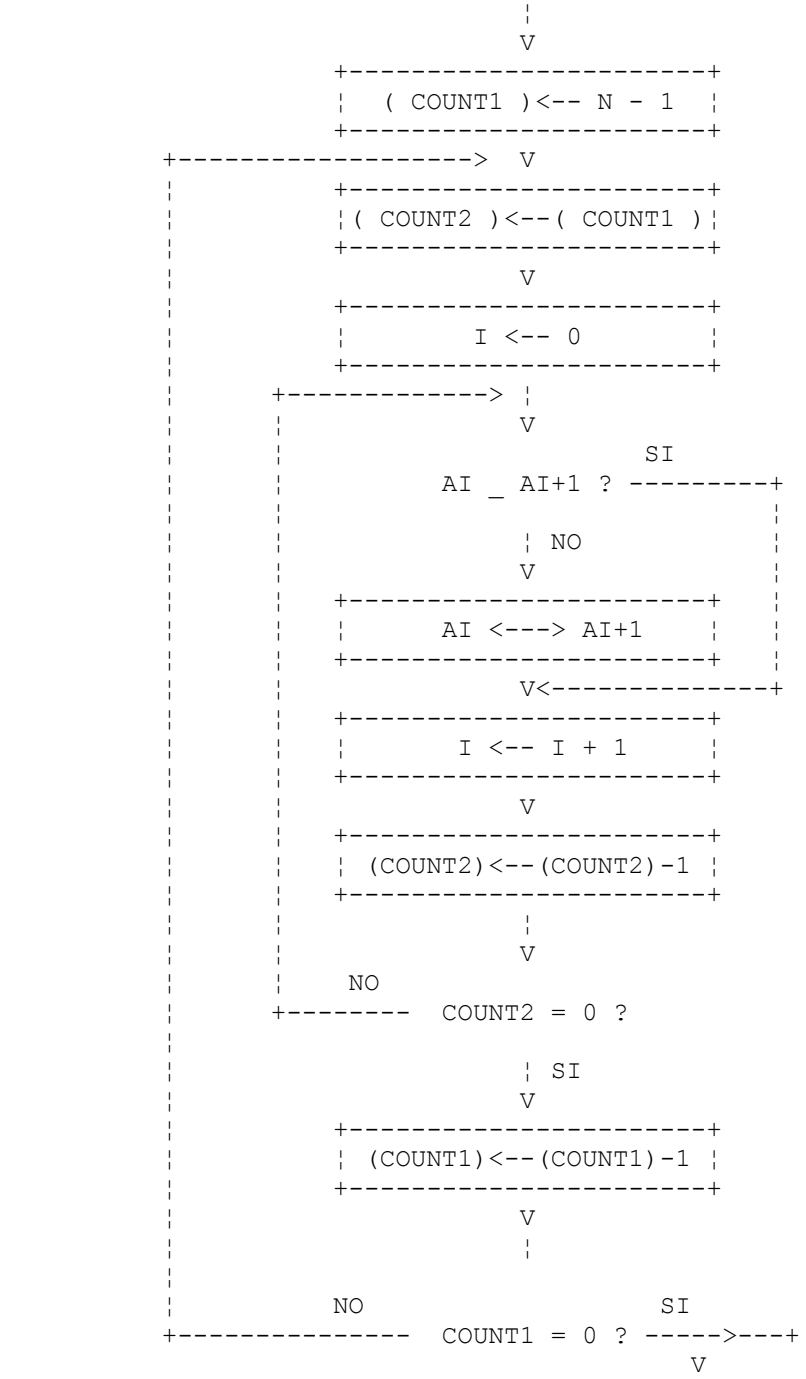
```
        MOV CX,L              ; PONE EL TAMAÑO DEL ARREGLO EN CX

        MOV SI,-1             ; INICIALIZA INDICE Y

        MOV AL,20             ; PONE EL CODIGO PARA ESPACIO EN AL

NEXT:   INC SI                ; INCREMENTA EL INDICE

        CMP AL,ASCII STR[SI]  ; PRUEBA PARA ESPACIO

        LOOPNE NEXT           ; LOOP SI NO ENCUENTRA EL ESPACIO Y
                              ; EL CONTADOR ES Ï 0

        JNZ NOT_FOUND         ; SI EL ESPACIO NO FUE ENCONTRADO,
                              ; TOMA LA RAMA A NOT FOUND
```

SORTEO DE BURBUJA

N ----> No. DE ELEMENTOS EN EL ARREGLO

A ----> NOMBRE DE EL ARREGLO

I ----> INDICE DENTRO DEL ARREGLO



```
          MOV CX,N                ; PONE EL CONTADOR 1
          DEC CX                  ; A N - 1
LOOP1:    MOV DI,CX               ; GUARDA EL CONTADOR 1 EN DI
          MOV BX,0                ; BORRA BX
LOOP2:    MOV AX,A[BX]            ; CARGA A(I) DENTRO DE AX Y
          CMP AX,A[BX+2]          ; COMPARA CON A(I+1)
          JGE CONTINUE            ; SWAP IF
          XCHG AX,A[BX+2]         ; A(I)<A(I+1) Y
          MOV A[BX],AX            ; ALMACENE EL NUMERO MAYOR
CONTINUE: ADD BX,2                ; INCREMENTA EL INDICE
          LOOP  LOOP2             ; SI NO ES EL FIN DE UN PASO,
                                  ; REPITA
          MOV CX,DI               ; RESTAURE EL CONTADOR 1 PARA
                                  ; EL LAZO EXTERNO
          LOOP  LOOP1             ; SI NO ES EL PASO FINAL,
                                  ; REPITA
```

SECUENCIA DE PROGRAMA PARA LLEVAR A CABO UN SORTEO DE BURBUJA.

SORTEO EN ORDEN DESCENDENTE

A (O) = 15		A (O) = 23	
A (1) = 23	COUNT1 = 4	A (1) = 15	FINAL
			=>
A (2) = 12	COUNT2 = 4	A (2) = 12	1a RONDA
A (3) = 7	I = 0	A (3) = 18	
A (4) = 18		A (4) = 7	
		I = 1	

ANALIZAR LOS DEMAS CICLOS PARA EFECTO BURBUJA CASO DEL PROGRAMA.

MANIPULACION DE BANDERAS

ALGUNAS VECES ES NECESARIO TENER CONTROL DIRECTO SOBRE LAS BANDERAS. LA HABILIDAD PARA PONER, BORRAR O CAMBIAR LA BANDERA CF ES DESEABLE Y TAMBIEN LA MANIPULACION DIRECTA DE LAS BANDERAS DE CONTROL DF E IF ES REQUERIDA.

INSTRUCCIONES PARA MANIPULACION DE BANDERAS

NOMBRE	NEMONICO Y FORMATO	DESCRIPCION
CLEAR CARRY	CLC	CF <---- 0
COMPLEMENT CARRY	CMC	CF <---- $\overline{CF}$
SET CARRY	STC	CF <---- 1
CLEAR DIRECTION	CLD	DF <---- 0
SET DIRECTION	STD	DF <---- 1
CLEAR INTERRUPT	CLI	IF <---- 0
SET INTERRUPT	STI	IF <---- 1
LOAD AH FROM FLAGS *	LAHF	(AH)<--- (LOW ORDER BYTE OR PSW)
STORE AH TO FLAGS *	SAHF	(LOW ORDER BYTE OR PSW)<---(AH)

BANDERAS: SOLAMENTE LAS BANDERAS INDICADAS SON AFECTADAS.

MODOS DE DIRECCIONAMIENTO: NINGUNO (IMPLICADO)

CLD Y STD PARA MANIPULACION DE STRINGS.

BCD MULTIPLE PRESICION (SUMA)

SUMA LOS NUMEROS BCD EMPACADOS EN LA LOCALIDAD NUMBER ONE A LOS NUMEROS BCD EMPACADOS EN LA LOCALIDAD NUMBER TWO Y DEJA EL RESULTADO EN LA LOCALIDAD NUMBER TWO .

```
            MOV CX,LONGITUD NUMERO    ; DETERMINA NUMERO DE BYTES
                                      ; A SUMAR

            MOV SI,OFFSET NUMBER ONE + LONGITUD NUMERO - 1

            MOV DI,OFFSET NUMBER TWO + LONGITUD NUMERO - 1

            CLC                       ; INICIA CON NO CARRY

LAZO DE SUMA

            MOV AL,[SI]               ; TOMA EL VALOR ONE

            ADC AL,[DI]               ; SUMA EL VALOR TWO

            DAA                       ; AJUSTE BCD

            MOV [DI],AL               ; ALMACENA EL RESULTADO

            PUSHF                     ; GUARDA LA BANDERA DE
                                      ; CARRY

            DEC SI                    ; APUNTA AL SIGUIENTE BYTE
                                      ; DE ONE

            DEC DI                    ; APUNTA EL SIGUIENTE BYTE
                                      ; DE TWO

            POPF                      ; REGRESA LAS BANDERAS

            LOOP LAZO DE SUMA         ; HACE EL SIGUIENTE BYTE
                                      ; SUPERIOR
```

INSTRUCCIONES ESPECIALES

INSTRUCCIONES NOP Y HLT

NOMBRE	NEMONICO Y FORMATO	DESCRIPCION
NO OPERACION	NOP	CAUSA NO ACCION
HALT	HLT	CAUSA QUE LA ACCION DE LA COMPUTADORA CESE

FLAGS: NINGUNA BANDERA ES AFECTADA.

MODO DE DIRECCIONAMIENTO: NINGUNO (IMPLICADO).

NOP => PUEDE USARSE PARA TOMAR TIEMPO (PARA PREVEER ALGUN FALTANTE DEL PROGRAMA).

HLT => EL PROCESADOR PARA LA SIGUIENTE INSTRUCCION DESPUES DE EJECUTARLA. SI LAS INTERRUPCIONES SON DESHABILITADAS CUANDO EL PROCESADOR PARA , LA COMPUTADORA SE COMPORTA COMO UN VEGETAL. SIN EMBARGO, SI LAS INTERRUPCIONES SON HABILITADAS CUANDO EL PROCESADOR PARA, LA INTERRUPCION SERA ACEPTADA Y EL CONTROL PASARA AL MANEJO DE INTERRPCIONES (AYUDA A SABER QUE TAREA ESTA HACIENDO).

* EL UNICO RECURSO PARA REINICIAR ES APAGARLA Y VOLVERLA A ENCENDER.

IV.- PROGRAMACION INTERNA

4.1 DESCRIPCION DEL DEBUG.

COMANDOS DEBUG : D = DUMP (VACIAR, DESCARGAR)

F = FILL (LLENAR)

PROGRAMAS EN LENGUAJE DE ALTO NIVEL. ESTAN AISLADOS DE LO QUE ESTA PASANDO REALMENTE DENTRO DE LA MAQUINA COMPUTADORA Y ASI ELLOS PUEDEN CONCENTRARSE EN LAS FORMULAS (EJEM A = 3).

EN CONTRASTE EL LENGUAJE ENSAMBLADOR OPERA EN UN NUVEL MUY CONCRETO. ESTE TRATA CON BITS, BYTES, PALABRAS (DOS BYTES JUNTOS), CON REGISTROS, LOS CUALES ESTAN COLOCADOS FISICAMENTE EN EL MICROPROCESADOR DONDE LOS BYTES Y PALABRAS SON ALMACENADOS Y CON LOCALIDADES DE MEMORIA, LAS CUALES TIENEN DIRECCION NUMERICA ESPECIFICA Y LOCALIZACION FISICA ESPECIFICA EN LOS CHIPS DE MEMORIA DENTRO DE LA PC.

COMPILADOR. CAMBIA EL ARCHIVO FUENTE HACIA INSTRUCCIONES EN LENGUAJE MAQUINA.

UN ARCHIVO FUENTE EN ENSAMBLADOR CONSISTE DE EL TEXTO DE EL PROGRAMA QUE FUE CREADO PRIMERO. ESTE ES ENTONCES ENSAMBLADO HACIA INSTRUCCIONES EN LENGUAJE MAQUINA POR UN PROGRAMA ENSAMBLADOR (SIMILAR A UN COMPILADOR) EXCEPTO QUE HAY UNA MAYOR CORRESPONDENCIA ENTRE INSTRUCCIONES EN LENGUAJE ENSAMBLADOR Y LENGUAJE MAQUINA.

DEBUG ES UN PROGRAMA MAS FACIL DE OPERAR QUE LOS PROGRAMAS ENSAMBLADORES ASM (O MASM).

PARA TECLEAR Y EJECUTAR UN PROGRAMA UTILIZANDO DEBUG REQUIERE SOLAMENTE LLAMAR AL MISMO DEBUG: UN PROCESO SIMPLE. USANDO

ASSEMBLER INVOLUCRA USAR UN EDITOR DE TEXTOS, EL ASSEMBLER MISMO Y UN PROGRAMA LLAMADO LINK Y CON FRECUENCIA OTRO PROGRAMA LLAMADO EXE2BIN. CADA UNO DE ESOS PROGRAMAS REQUIERE UNA SERIE DE COMANDOS COMPLEJOS PARA TRABAJAR.

DEBUG NO REQUIERE " OVERHEAD " (ENCABEZADOS) COMO LOS PROGRAMAS ESCRITOS EN ENSAMBLADOR.

UTILIZANDO DEBUG ESTA EN CONTACTO CERCANO CON LO QUE ESTA SUCEDIENDO REALMENTE EN SU COMPUTADORA

DEBUG VERSUS ASSEMBLER

DEBUG	ASSEMBLER
FACIL CORRERLO.	DURO PARA CORRER.
POCOS ENCABEZADOS EN LOS PROGRAMAS.	MUCHOS ENCABEZADOS EN LOS PROGRAMAS.
MAS CERCANIA A LA MAQUINA.	AISLADO DE LA MAQUINA.
NO MUY VERSATIL.	MUY VERSATIL.
BUENO EN PROGRAMAS CORTOS.	BUENO EN PROGRAMAS LARGOS.

" LOS OJOS SON LAS VENTANAS DE EL ALMA " => EL DEBUG VENTANA DEL 8088.

DEBUG. ES UTILIZADO PARA EXAMINAR Y MODIFICAR LOCALIDADES DE MEMORIA; CARGAR, ALMACENAR E INICIAR PROGRAMAS; Y PARA EXAMINAR Y MODIFICAR REGISTROS (TOCAR VARIOS REGISTROS FISICAMENTE).

<< LEVANTANDO EL DEBUG >>

```
          +-------+
A > DEBUG ¦ ENTER ¦
          +-------+
-   <-- CARACTER PROMPT DEL DEBUG.
```

COMANDO " D " DUMP (VACIAR).

- d100 <---- TECLEE ESTO.

```
08F1:0100  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00
```

```
08F1:0110  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00

08F1:0120  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00

O8F1:0130  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00

08F1:0140  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00

08F1:0150  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00

08F1:0160  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00

08F1:0170 00 -------------- 8 BITS ( HEXADECIMAL )
 ¦    +-------------------------------+
 +--------------+                      DIRECCION DE ( CONTRAPESO )
                SEGMENTO O SECCION            FOCET
+-------------------------------------------------------------+
¦ PARA ENCONTRAR LA DIRECCION REAL, TOME LA DIRECCION DE EL   ¦
¦                                                             ¦
¦ EL SEGMENTO, CORRALA UN LUGAR A LA IZQUIERDA Y SUMELE  LA   ¦
¦                                                             ¦
¦ DIRECCION DEL OFFSET.                                       ¦
+-------------------------------------------------------------+
```

COMANDO " F " FILL (LLENAR).

```
- f120 14f ff
   ¦   ¦    ¦
   ¦   ¦    +-------- CONSTANTE A SER LLENADA.
   ¦   ¦
   ¦   +--------- DIRECCION FINAL.
   ¦
   +------ DIRECCION DEL INICIO.
```

PARA VER EL EFECTO TECLEE -d100

CODIGOS ASCII EJEM : "a" = 61 , "$" = 24

CODIGOS ASCII EJEM : "a" = 61 , "$" = 24

```
- f 100 117 61

- f 178 17f 24

08F1:0100 61 61 61 61  ¨ ¨ - ¨    61  aaaaaaaaaaaaaaaa
     0110 61 61 61 61  ¨ ¨ -00 ¨ ¨00  aaaaaaaa

08F1:0170                 -24 ¨ ¨24          $$$$$$$$
```

4.2.- COMANDOS GENERALES DEL DEBUG

E = ENTER (ENTER UN BYTE O BYTES)

A = ASSEMBLE

U = UNASSEMBLE

G = GO

```
-E100  <----  TECLEE ESTO
- 04B5:0100   61 _  EL CURSOR ESPERA AQUI PARA QUE USTED
                    TECLEE UN NUMERO DE DOS DIGITOS.
```

PARA "ENTRAR" UN NUMERO HEXADECIMAL DE 2 DIGITOS EN ESTA LOCALIDAD TECLEAMOS EL NUMERO SEGUIDO POR LA BARRA ESPACIADORA (NO LA TECLA ENTER). LA BARRA ESPACIADORA TIENE EL EFECTO DE ENTRAR UN NUMERO E IR POR EL SIGUIENTE. LA TECLA ENTER, ENTRA EL NUMERO Y TERMINA LA SECUENCIA DE ENTRADA "E" (COMANDO) Y LA REGRESA A EL PROMPT DEL DEGUG.

EJEMPLO DE UN PROGRAMA TECLEADO CON EL COMANDO "E"

```
B2
1
B4
2
CD
21
CD
20
```

ESCRIBIENDO EL PROGRAMA CON EL COMANDO " A "

* UTILIZA NEMONICOS (INSTRUCCIONES EN LENGUAJE ENSAMBLADOR)

```
+-----------------------------------------------------------+
¦ PROGRAMAS ESCRITOS EN DEBUG DEBERAN SIEMPRE INICIAR EN LA ¦
¦                                                           ¦
¦ DIRECCION CON UN OFFSET DE 0100 H.                        ¦
+-----------------------------------------------------------+

-a100

08F1:100 _
```

```
0100 MOV DL,1  <---- DESPUES DE CADA NEMONICO TECLEE ENTER

0102 MOV AH,2

0104 INT 21

0100 INT 20

08F1:0108 <----- TECLEE ENTER PARA REGRESAR AL DEBUG ( PROMPT )
```

CORRIENDO (GO) EL PROGRAMA

```
-G   <----- TECLEE ESTO
O
PROGRAMA TERMINADO NORMALMENTE
-
```

(SI HAY UN ERROR DE LOGICA TECLEE ALT - CTRL Y DEL SIMULTANEAMENTE).

- d100 PARA VERIFICAR QUE HAY LO MISMO QUE SE TECLEO CON EL COMANDO E.

COMANDO " U "

```
- U100,106  <----  TECLEE ESTO
                          --+
08F1:0100 B2 01  MOV DL,01  ¦
                            ¦
    :0102 BA 02  MOV AH,02  ¦ PROGRAMA IMPRIME HACIA
                           >
    :0104 CD 21  INT 21     ¦ AFUERA TODO ESTO
                            ¦
    :0106 CD 20  INT 20     ¦
         ¦     ¦¦       ¦ --+
         +--V--++----V----+
        LENGUAJE LENGUAJE
        MAQUINA  ENSAMBLADOR
          <----------  ENSAMBLAR
          --------->   DESENSAMBLAR
+-----------------------------------------------------------+
¦ PROGRAMA ENSAMBLADOR TRANSLADA EL LENGUAJE ENSAMBLADOR    ¦
¦                                                           ¦
¦ A EL LENGUAJE MAQUINA.                                    ¦
+-----------------------------------------------------------+

REGISTRO       DL = 01
               AH = 02

INT = JUMP A SUBRUTINA.
```

SUBRUTINA 21H DEPENDE DE LO QUE HAYA EN AH.

COMO HAY UN 2 EN AH TRANSFIERE EL CONTROL A LA FUNCION " DISPLAY DE SALIDA " , CUYO PROPOSITO ES ESCRIBIR UN SIMPLE CARACTER EN LA PANTALLA (FUNCION DOS).

PONE EN LA PANTALLA LO QUE HAYA EN EL REGISTRO DL = 01 (EQUIVALENTE A LA CARITA FELIZ EN UN CODIGO IBM INVENTADO).

INT 20 => INTERRUPCION PARA TERMINACION DEL PROGRAMA. REGRESA AL DEBUG.

```
- E101
08F1:0101  01.58 <----  TECLEE "58"  PRESIONE LA TECLA ENTER
-G
X
PROGRAMA TERMINADO NORMALMENTE.
-
```

<< LAZO SIN FIN >>

```
-a106
08F1:0106  JMP 100  <---- TECLEE JMP 100
08F1:0108  <---- TECLEE ENTER
```

CTRL	-	C

PARA SALIRSE DEL LAZO SIN FIN UNA VEZ SE CORRA EL PROGRAMA

<< COMANDOS DEBUG >>

R = REGISTROS

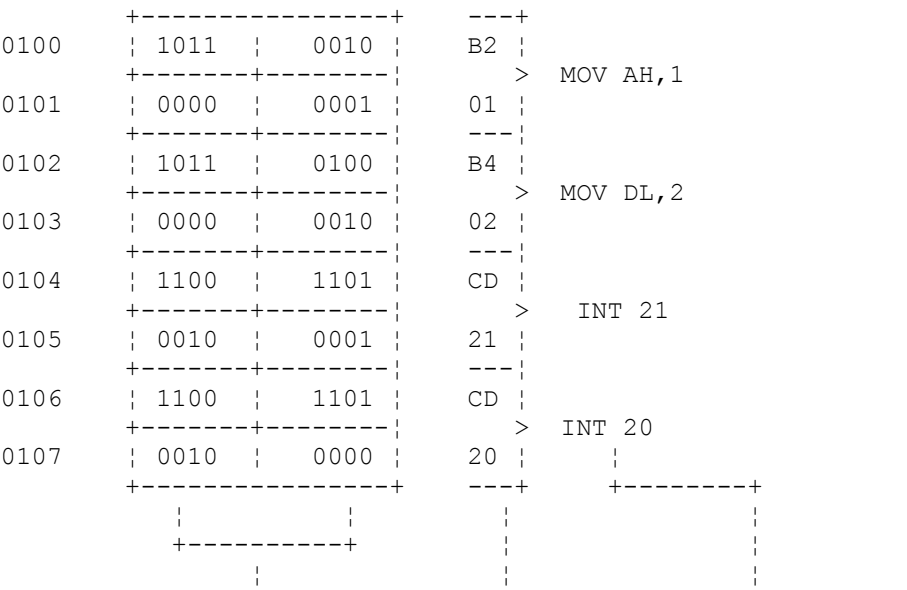
N = NOMBRAR

W = WRITE (ESCRIBIR)

Q = QUIT (ABANDONAR)

L = LOAD (CARGAR)

```
- U 100,107

08F1:0101 B201 MOV DL,01

08F1:0102 B402 MOV AH,02

08F1:0104 CD21 INT 21

08F1:0106 CD20 INT 20
```



MANIPULACION DE REGISTROS CON DEBUG.

- R (PARA VER TODOS LOS REGISTROS).

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=08F1 ES=08F1 SS=08F1 CS=08F1 IP=0100 NV UP DI PL NZ NA PO NC


- RAX   <---- TECLEE ESTO PARA ENTRAR A EL REGISTRO AX

AX 0000 <---- CONTENIDO CORRIENTE (ACTUAL) ES 0000
              DEBUG ESPERA PARA QUE USTED ENTRE (TECLEE) UN NUEVO
              VALOR. SUPONGAMOS QUE USTED TECLEE "1234"

:1234   <---- CAMBIELO ESTE A 1234H ( TECLEARLO ).

-       <---- REGRESA AL PROMPT DEL DEBUG
```

PARA VERIFICAR LO ANTERIOR

```
 - R  +----------- VERIFICADO
      V
 AX=1234 BX=0000 CX=FFFF DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

 DS=08F1 ES=08F1 SS=08F1 CS=08F1 IP=0100 NV UP DI PL NZ NA PO NC


 - RCX   <---- TECLEE ESTO PARA ENTRAR A EL REGISTRO CX

 CX 0000 <---- CONTENIDO CORRIENTE (ACTUAL) ES 0000

 :FFFF   <---- CAMBIELO ESTE A FFFFH ( TECLEARLO ).

 -


 PARA VERIFICAR LO ANTERIOR
 - R                 +---------- VERIFICADO
                     V
AX=1234 BX=0000 CX=FFFF DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=08F1 ES=08F1 SS=08F1 CS=08F1 IP=0100 NV UP DI PL NZ NA PO NC
```

PUEDE HACER ESTO CON LOS REGISTROS AX, BX, CX Y DX.

4.3.- PRACTICAS CON EL DEBUG

PROGRAMA QUE MUESTRA TODOS LOS CARACTERES ASCII

```
A >DEBUG
        +------------------------------+
- A100  ¦ O A ( ASUME LOCALIDAD 100 ) ¦
        +------------------------------+
08F1:0100 MOV DL,0  <-- PONE EL PRIMER CARACTER EN DL

08F1:0102 MOV AH,2  <-- ESPECIFICA FUNCION DE SALIDA EN EL
                        DISPLAY

08F1:0104 INT 21    <-- LLAMA FUNCION DOS PARA IMPRIMIR CARACTER

08F1:0106 INC DL    <-- CAMBIA AL SIGUIENTE CARACTER

08F1:0108 JMP 102   <-- REGRESA PARA MOSTRAR EL SIGUIENTE
                        CARACTER

08F1:010A           <-- PRESIONE ENTER PARA TERMINAR EL
                        ENSAMBLADOR
```

```
-U 100,108

08F1:0100 B2 00     MOV DL,00

08F1:0102 B4 02     MOV AH,02

08F1:0104 CD 21     INT 21

08F1:0106 FE C2     INC DL

08F1:0108 EB F8     JMP 0102
```

GUARDANDO EL PROGRAMA EN DISQUETE.

PROCESO DE 3 PASOS

1) DECIR A EL DEBUG EL NOMBRE DEL PROGRAMA QUE DESEAMOS GUARDAR.

2) DECIRLE QUE TAN LARGO ES EL PROGRAMA.

3) DECIRLE QUE ESCRIBA EL PROGRAMA A EL DISQUETE.

1) PARA ESPECIFICAR EL NOBRE COMANDO "N" (POR NOMBRE), SEGUIDO INMEDIATAMENTE (NO ESPACIO) POR EL NOMBRE DEL ARCHIVO QUE DESEAMOS GUARDAR. EL NOMBRE DEL ARCHIVO PUEDA SER CUALESQUIERA, PERO LA EXTENSION DEBE SER COM SI DESEAMOS EJECUTARLO DIRECTAMENTE DESDE DOS.

```
-NASCII.COM
```

2) DECIRLES CUANTOS BYTES DESEAMOS GUARDAR (REGISTROS BX Y CX).

CX RETIENE LOS BYTES DE BAJO ORDEN (MENOS SIGNIFICATIVOS)

BX RETIENE LOS BYTES DE ALTO ORDEN (MAS SIGNIFICATIVOS)

```
-RBX      <--- TECLEE ESTO PARA VER EL REGISTRO BX
-RBX      <--- TECLEE ESTO PARA VER EL REGISTRO BX

BX 0000   <--- ESTA PUESTO A CEROS.

:         <--- TECLEE ENTER YA QUE ESTA LISTO EN CEROS

-rcx      <--- TECLEE ESTO PARA VER EL REGISTRO CX.

CX 0000   <--- ESTA EN CEROS TAMBIEN

: A       <--- TECLEE EL NUMERO DE BYTES DEL PROGRAMA (10 DECIMAL)
```

3) DECIRLE QUE ESCRIBA EL PROGRAMA W (WRITE ESCRIBIR)

```
- W       <--- TECLEE ESTO

WRITING 000A BYTES  <--- DEBUG NOS DICE ESTO
```

EL COMANDO "Q" (QUIT = ESCAPAR)

```
- Q  <---- TECLEE ESTO PARA ESCAPAR DEL DEBUG.

A > USTED REGRESA AL DOS

              << EJECUTANDO EL PROGRAMA DESDE DOS >>

A > ASCII
```

MOSTRARA TODOS LOS CARACTERES PARA PARAR EL PROGRAMA TECLEE

CRTL - BREAK

```
                << CARGANDO EL PROGRAMA EN DEBUG >>
                         ( HAY DOS FORMAS )
A > DEBUG ASCII.COM    ( NO OLVIDE LA EXTENSION )
              +-------- ARCHIVO COM A SER CARGADO CON DEBUG

 A > DEBUG

 - NASCII.COM

 -L
```

PARA VERIFICAR USE

```
 - U100, 108
```

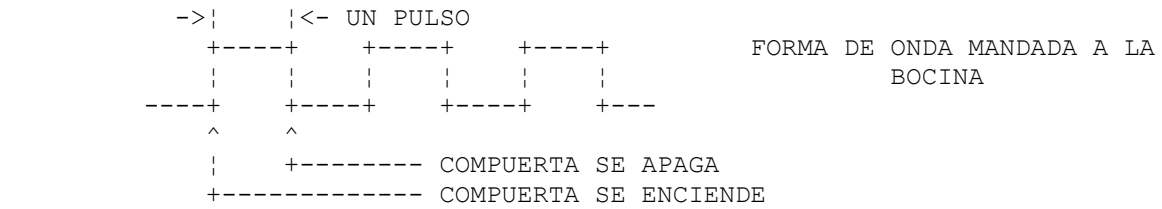
PARA HACER MAS ELEGANTE EL PROGRAMA.

```
 -a100

 08F1:0100 MOV CX,100     <-- LEVANTA EL CONTADOR PARA EL
                              LOOP
08F1:0103 MOV DL,0       <-- PONE EL PRIMER CARACTER EN DL

08F1:0105 MOV AH,2       <-- ESPECIFICA FUNCION DE SALIDA EN
                             EL DISPLAY

08F1:0107 INT 21         <-- LLAMA FUNCION DOS PARA IMPRIMIR
                             CARACTER

08F1:0109 INC DL         <-- CAMBIA AL SIGUIENTE CARACTER

08F1:010B LOOP 105       <-- LOOP HASTA QUE CX ES CERO (CX<--CX-1)

08F1:010D INT 20         <-- FIN ( SALIDA A DEBUG O DOS )
```

```
- U100, 10d

     : 0100 B90001  MOV CX,0100

     : 0103 B200    MOV DL,00

     : 0105 B402    MOV AH,02

     : 0107 CD21    INT 21

     : 0109 FEC2    INC DL

     : 010B E2F8    LOOP 0105

     : 010D CD20    INT 20

PARA GUARDARLO

-NSMASCII.COM  <-- PONE EL NOMBRE DEL PROGRAMA
     -------+
- RBX        ¦
            ¦
BX:0000     ¦
            ¦
:           ¦
           >  PONE LA LONGITUD A F EN HEXADECIMAL
- RCX        ¦
            ¦
CX:0000     ¦
            ¦
:F          ¦
     -------+
- W            ESCRIBE EL PROGRAMA A DISQUETE

- Q

A > SMASCII <---- PARA EJECUTARLO


EL PROGRAMA DEL SONIDO ( SOUND )
A > DEBUG

-a100                    U 100,10E

08F1:0100 IN AL,61       E4 61

08F1:0102 AND AL,FC      24 FC

08F1:0104 XOR AL,2       34 02

08F1:0106 OUT 61,AL      E6 61
```

```
08F1:0108 MOV CX,140     B9 40 01

08F1:010B LOOP 10B       E2 FE

08F1:010D JMP 104        EB F5
```

GUARDE EL ARCHIVO Y EJECUTELO.

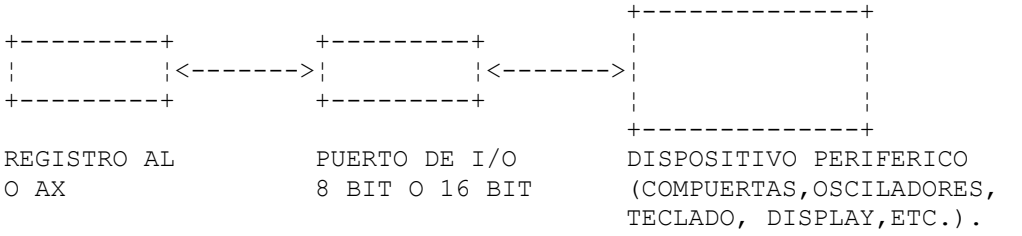


SI SE EJECUTA DESDE DEBUG CON ALT-CTRL-DEL NO SE PUEDE APAGAR

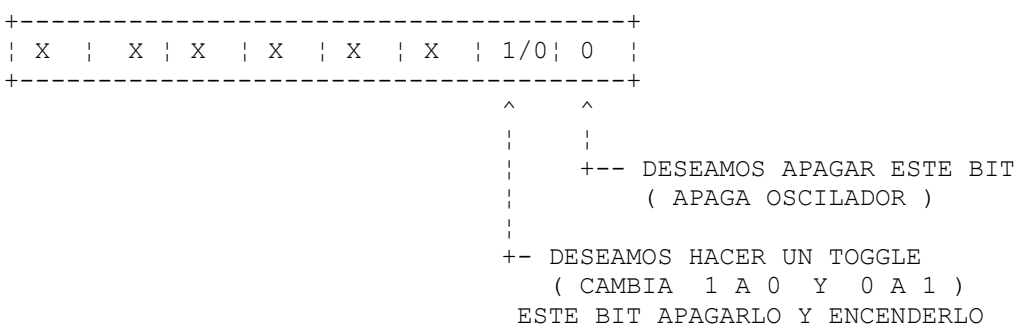
NECESITA APAGAR LA COMPUTADORA Y DESDE DOS CON ALT-CTRL-DEL SE APAGA.

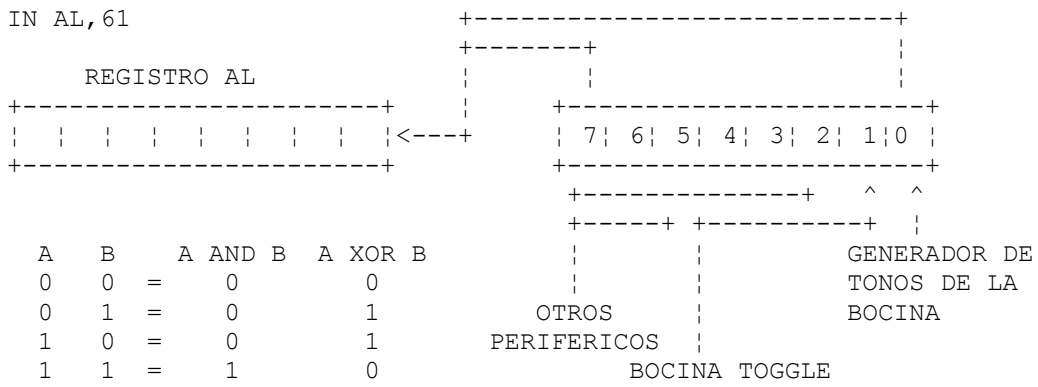
PARA ENCENDER Y APAGAR LA COMPUERTA OUT 61,AL

PUERTOS => ESTAN CONECTADOS A DISPOSITIVOS FISICOS EN EL MUNDO REAL.



PARA EL CASO DE LA BOCINA





FC = 1111 1100 => AND AL,FC HACE BIT 0 Y BIT 1 = 0

02 = 0000 0010 => XOR AL,2 HACE BIT 1 = TOGGLE

0 XOR 1 = 1
1 XOR 1 = 0
0 XOR 1 = 1

RECHAZO DE TIEMPO CX = 140 => PARA CAMBIAR EL SONIDO CAMBIAR

CX = 100 Y CORRERLO

V.- FUNCIONES PREDEFINIDAS DE D.O.S.

5.1.- FUNCIONES DE LECTURA DESDE TECLADO (CARACTER STRING)

LLAMADAS A FUNCIONES DOS

ENTRADA DESDE TECLADO

IMPRIMIENDO CARACTER STRING

ENTRADA AL BUFFER DEL TECLADO

SALIDA A IMPRESORA

PROMPT " A > " ES D.O.S.

PROMPT " A > " ES D.O.S.

SALIDA AL DISPLAY UTILIZANDO DOS.

```
MOV DL,1    <--- PONE EL CARACTER ASCII EN DL
MOV AH,2    <--- PONE EL NUMERO DE FUNCION DOS EN EL
                 REGISTRO AH
INT 21      <--- INTERRUMP No. 21 LLAMA AL DOS
```

ASI NO TENEMOS QUE SABER NADA ACERCA DE EN QUE MODO ESTA EL DISPLAY, CUANTO RETRAZO HORIZONTAL ESTA HACIENDO, DONDE OCURRIO EL RETORNO DE CARRO.... SIMPLEMENTE LLAMANDO UNA RUTINA EN DOS Y CON UNOS POCOS CONOCIMIENTOS ACERCA DEL SISTEMA OPERATIVO Y SOLAMENTE TRES INSTRUCCIONES.

```
+-------------------------------------------------------------+
¦ FUNCION DE ENTRADA AL TECLADO - NUMERO 01 HEXADECIMAL       ¦
¦                                                             ¦
¦ ENTRA CON :   REGISTRO AH = 1                               ¦
¦                                                             ¦
¦ EJECUTA   :   INT 21                                        ¦
¦                                                             ¦
¦ REGRESA CON : CARACTER TECLEADO EN EL REGISTRO AL           ¦
¦               +----++-----+                                 ¦
¦ COMENTARIO  : ¦CTRL¦¦BREAK¦ CAUSA SALIDA DE LA FUNCION      ¦
¦               +----++-----+                                 ¦
+-------------------------------------------------------------+
```

```
A > DEBUG

- A100
                      ------+
08F1:0100 MOV AH, 1     ¦
                        ¦
08F1:0102 INT 21        >  TECLEAR ESTAS INSTRUCCIONES
                        ¦
08F1:0104 INT 20        ¦
                  -------+
08F1:0106    PRESIONE "ENTER" PARA DEJAR EL COMANDO "A"

PARA ASEGURARSE

U 100,105

08F1:0100         B401      MOV AH, 01

08F1:0102         CD21      INT 21
08F1:0104         CD2O      INT 20

- G       PARA EJECUTARLO

- Z       TECLEE LA Z POR EJEMPLO

PROGRAMA TERMINADO NORMALMENTE

CARACTER ASCII DE LA Z GUARDADO EN AL

      +-------------------------------------------------+
      ¦ EL µP 8088 MANTIENE EL RASTRO DE DONDE ESTA CON EL ¦
      ¦                                                 ¦
      ¦ REGISTRO APUNTADOR DE INSTRUCCIONES ( IP )      ¦
      +-------------------------------------------------+

PROGRAMA ANTERIOR

-  G  <--- CORRA EL PROGRAMA OTRA VEZ.

^C        TECLEE ¦CTRL¦¦BREAK¦
                 +----++-----+
AX=0100 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=08F1 ES=08F1 SS=08F1 CS=08F1 IP=0104 NV UP DI PL NZ NA PO NC

08F1:0104  CD20  INT 20

-G   <--- CORRALO

PROGRAMA TERMINO NORMALMENTE
```

PARA CAMBIAR EL REGISTRO IP.

- RIP   <--- TECLEE ESTO PARA VER EL REGISTRO IP

IP 0104   <--- CONTENIDO DE IP

: 100   <--- TECLEE ESTO PARA CAMBIAR A 100

- G   <--- AHORA PRUEBE EL PROGRAMA OTRA VEZ

Z   <--- ESPERA PARA QUE USTED TECLEE UN CARACTER

PROGRAMA TERMINO NORMALMENTE

CONCLUSION. TENGA EL HABITO DE TECLEAR "RIP" A 100 SI ES NECESARIO

ANTES DE TECLEAR " G " PARA CORRER EL PROGRAMA.

PARA TECLEAR MAS DE 1 CARÁCTER
- A 100

```
08F1:0100           MOV AH,1

08F1:0102           INT 21

08F1:0104           JMP 100

08F1:0106

U100,105

08F1:0100           B401           MOV AH,01

08F1:0102           CD21           INT 21

08F1:0104           EBFA           JMP 0100
```

BUENOS DIAS A ^ C

```
          +----++-----+
```

SE TECLEA ¦CTRL¦¦BREAK¦ PARA INTERRUMPIR EL CICLO INFINITO

5.2.- FUNCIONES DE IMPRESION (MONITOR E IMPRESORA)

```
+-----------------------------------------------------------+
| FUNCION PRINT STRING NUMERO 09 HEXADECIMAL                |
|                                                           |
| ENTRE CON : REGISTRO AH = 9                               |
|           : DS:DX = DIRECCION DE INICIO DE STRING         |
|                                                           |
| EJECUTA CON : INT 21                                      |
|                                                           |
| COMENTARIOS : EL STRING DEBE TERMINAR CON $ ( SIGNO DEL   |
| DOLLAR )                                                  |
+-----------------------------------------------------------+
```

NOTESE : REGISTRO DS:DX = DIRECCION DE INICIO DEL STRING.

DIRECCION DEL SEGMENTO EN DS.

DIRECCION DEL OFFSET EN DX.

```
+-----------------------------------------------------------+
| STRINGS A SER IMPRIMIDOS POR LA FUNCION PRINT STRING      |
|                                                           |
| DEBEN TERMINAR CON EL SIGNO DE DOLAR ( $ )                |
+-----------------------------------------------------------+
```

EJEMPLO PARA USAR PRINT STRING.

```
- A 100

08F1:0100      MOV DX,109

08F1:0103      MOV AH,9

08F1:0105      INT 21

08F1:0107      INT 20

08F1:0109      DB 'GOOD MORNING ROBERT!$'

08F1:011F
```

EL PSEUDO CODIGO DE OPERACION "DB"

DB = "DEFINE BYTE"

LOS PSEUDOS CODIGO DE OPERACION LE DICEN A EL PROGRAMA

ESAMBLADOR (DEBUG) QUE HACER CUANDO EL PROGRAMA ESTA SIENDO

ENSAMBLADO.

+--+
| "INSTRUCCIONES" SON INSTRUCCIONES EN EL MICROPROCESADOR |
| |
| "PSEUDO-OPS" SON INSTRUCCIONES DE EL ENSAMBLADOR |
+--+

```
-U 100,108

0905:0100      BA0901    MOV DX,0109

0905:0103      B409      MOV AH,09

0905:0105      CD21     INT 21

0905:0107      CD20     INT 20
```

U NO ES DE MUCHA AYUDA YA QUE CON ESOS BYTES NO SON INSTRUCCIONES DEL PROGRAMA.

```
-D 100,11F
08F1:0100 BA 09 01 B4 09 CD 21 CD-20 47 6F 6F 64 C0 4D 6F ..4M!MGO
                                                           OD MO

08F1:0110 72 6E 69 6E 67 2C 20 52-6F 52 65 72 74 21 24 3A RNING RO
                                                           BERT!$

PROGRAMA OCUPA 9 BYTES

CARACTERES STRING HASTA 11E

GUARDANDO EL PROGRAMA EN DISCO.

-NWAKEVP.COM

-RBX

BX: 0000

:
 -RCX

 CX=0000

 :1F

 -W

 WRITING 001F BYTES
```

AHORA CORRA EL PROGRAMA


```
 -G     <----- EJECUTANDO EL PROGRAMA

 GOOD MORNING ROBERT !
```

PROGRAMA TERMINO NORMALMENTE
O
A > WAKEUP

GOOD MORNING ROBERT!

```
+-------------------------------------------------------------+
| ENTRADA AL BUFFER DEL TECLADO   FUNCION NUMERO 0A HEX       |
|                                                             |
| ENTRE CON:                                                  |
|      REGISTRO AH = 0A HEX.                                  |
|      REGISTRO DS:DX = DIRECCION DEL BUFFER                  |
|                                                             |
| EJECUTE CON : INT 21                                        |
|                                                             |
| REGRESE CON: CARACTERES TECLEADOS EN EL BUFFER.             |
|                                                             |
| COMENTARIOS: 1er BYTE DEL BUFFER=CONTADOR DEL No.MAXIMO DE  |
|              CARACTERES                                     |
|              2o BYTE=NUMERO ACTUAL DE CARACTERES TECLEADOS  |
+-------------------------------------------------------------+
```

EJEMPLO:

A> DEBUG.

```
-A100

08F1:0100  MOV DX,109

08F1:0103  MOV AH,A

08F1:0105  INT 21

O8F1:0107  INT 20

08F1:0109  DB 20
```

BUFFER = UNA SECUENCIA DE LOCALIDADES DE MEMORIA

```
+-----------------------------------+
|20 | 13| B | Y |......| I | N | G |
+-----------------------------------+
 ^    ^ +------------V-------------+
 |    |        MENSAJE EN ASCII
 |    |
 |    +-- NUMERO ACTUAL DE CARACTERES
 |        TECLEADOS
 |
 +- MAXIMO No. DE CARACTERES ( NUNCA
    PUEDE SER MAYOR DE 255 DECIMAL )
```

DESPUES DE TECLEAR EL MENSAJE TECLEE "
ENTER "

DEFINICION DE BUFFER 08F1:0109 DB 20 (20 HEX = 32 DECIMAL

LOCALIDADES SIN USAR).

EL PROGRAMA LE DICE DONDE ESTA COLOCADO EL BUFFER, COLOCANDO

SU DIRECCION EN EL REGISTRO DX CON EL MOV DX,109 LA FUNCION A HEX.

ESTA COLOCADA EN EL REGISTRO AH Y LUEGO SE LLAMA AL DOS CON NT 21.

-GUARDANDO EN DISQUETE

```
     -NBUFFIN.COM

     -RBX

     BX=0000

     :

     -RCX

     CX 0000

     : A

     -W
WRITING 000A BYTES

 -U 100,108

 0905:0100      BA0901    MOV DX,0109

 0905:0103      B4OA      MOV AH,0A

 0905:0105      CD21      INT 21

 0905:0107      CD20      INT 20

 -D 109,109

 08F1:0109 20
                                  +-------+
 BY BROOKS TOO BROAD FOR LEAPING¦ ENTER ¦
                                  +-------+
 PROGRAMA TERMINO NORMALMENTE
                                    MAXIMO No.DE  NUMERO ACTUAL DE
                                    CARACTERES    CARACTERES
-D 100,12F                             ¦  +----- TECLEADOS
                                       ¦  ¦
                    PROGRAMA           ¦  ¦  +---------- INICIO DEL
          +------------^-------------+V  V  V           MENSAJE
 0905:0100 BA 09 01 B4 0A CD 21 CD-20 20 IF 42 79 20 62 72 ..4M!MBY
                                                          BR

 0905:0110 6F 6F 6B 73 20 74 6F 6F-20 62 72 6F 61 64 20 66 OOKS TOO
                                                          BROAD F

 0905:0120 6F 72 20 66 65 61 70 69-6E 67 0D 00 00 00 00 00 OR LEAPI
                                          ^               NG
  CARACTER RETURN AL FIN DEL MENSAJE ----+
```



```
WRITING 000A BYTES

 -U 100,108
```

```
0905:0100      BA0901    MOV DX,0109

0905:0103      B4OA      MOV AH,0A

0905:0105      CD21      INT 21

0905:0107      CD20      INT 20

-D 109,109

08F1:0109 20
                                +-------+
BY BROOKS TOO BROAD FOR LEAPING¦ ENTER ¦
                                +-------+
PROGRAMA TERMINO NORMALMENTE
                                  MAXIMO No.DE  NUMERO ACTUAL DE
                                  CARACTERES    CARACTERES
-D 100,12F                            ¦  +----- TECLEADOS
                                      ¦  ¦
                   PROGRAMA           ¦  ¦  +--------- INICIO DEL
         +------------^-------------+V  V  V          MENSAJE
0905:0100 BA 09 01 B4 0A CD 21 CD-20 20 IF 42 79 20 62 72 ..4M!MBY
                                                        BR

0905:0110 6F 6F 6B 73 20 74 6F 6F-20 62 72 6F 61 64 20 66 OOKS TOO
                                                        BROAD F

0905:0120 6F 72 20 66 65 61 70 69-6E 67 0D 00 00 00 00 00 OR LEAPI
                                        ^               NG
 CARACTER RETURN AL FIN DEL MENSAJE ----+


                                        -
                                        -
                                        -
                                        -
                                        -
                                        -
                                        -
                                        -
                                        -
                                        -
                                        -
                                      -  130 -

- G

 QUESTION AUTORITY ! ---> No. ACTUAL DE CARACTERES
 TECLEADOS=19D=13H

 PROGRAMA TERMINO NORMALMENTE


                        << PROGRAMA ESPEJO >>

 -A100
```

```
                    -+
0905:0100 MOV DX,116 ¦
                     ¦
0905:0103 MOV AH,A   > ENTRADA AL BUFFER DEL TECLADO
                     ¦
0905:0105 INT 21    -+
                    -+
0905:0107 MOV DL,A   ¦
                     ¦ SALIDA AL DISPLAY
0905:0109 MOV AH,2   > ( LINEFEED )= A HEX
                     ¦
0905:010B INT 21    -+
                    -+
0905:010B MOV DX,118 ¦
                     ¦
0905:0110 MOV AH,9   > PRINT STRING
                     ¦
0905:0112 INT 21     ¦
                    -+
0905:0114 INT 20    <---- RETURN AL DEBUG O DOS

0905:0116 DB 30     <---- MAXIMO No. DE CARACTERES

0905:0117


-U 100,115

0905:0100  BA1601         MOV DX,0116

0905:0103  B40A           MOV AX,0A

0905:0105  CD21           INT 21

0905:0107  B20A           MOV DL,0A

0905:0109  BA02           MOV AH,02

0905:010B  CD 21          INT 21

0905:010D  BA1801         MOV DX,0118

0905:0110  B409           MOV AH,09
```



```
0905:0112  CD21           INT 21

0905:0114  CD20           INT 20


BUFFER COMINENZA EN LA 0116


-D 110,14F
                           +-------- MAXIMO No. DE CARACTERES
             ULTIMA PARTE  ¦  +----- NUMERO ACTUAL DE CARACTERES
             DEL PROGRAMA  ¦  ¦      TECLEADOS
```

```
          +------^--------+ V  V
 0905:0110 B4 09 CD 21 CD 20 30 00 00
                                  ^
 0905:0120 ....                   +- COMIENZO DEL MENSAJE
```

NOTA: ES IMPORTANTE TERMINAR SU MENSAJE TECLEANDO EL SIGNO " $ "

```
-G

 BUENOS DIAS I.T.N.L. $  <--- TECLEE ESTO

 BUENOS DIAS I.T.N.L.    <--- PROGRAMA IMPRIME ESTO.

 PROGRAMA TERMINO NORMALMENTE
```

```
   +-------------------------------------------------------------+
   ¦ SALIDA A IMPRESORA         FUNCION   NUMERO 05 HEX          ¦
   ¦                                                             ¦
   ¦ ENTRE CON: REGISTRO AH = 5                                  ¦
   ¦            REGISTRO DL = CARACTER A SER IMPRESO             ¦
   ¦                                                             ¦
   ¦ EJECUTE :  INT 21                                           ¦
   +-------------------------------------------------------------+
```

SIMILAR A LA FUNCION DE SALIDA AL DISPLAY.

AQUI NO HAY FUNCION QUE MANDE UN STRING DE CARACTERES A LA IMPRESORA A LA VEZ COMO LA FUNCION STRING LO HACE EN EL VIDEO.

PROBLEMA PARA MANDAR UN STRING DE CARACTERES A TRAVEZ DEL REGISTRO DL A UNA IMPRESORA.

<< MANERA NO DEMASIADO ELEGANTE DE IMPRIMIR UN STRING >>

ENVIAR " HI " A LA IMPRESORA.

```
 -A100

 0905:0100  MOV DL,[122] <--- MANDA "H" A LA IMPRESORA

 0905:0104  MOV AH,5

 0905:0106  INT 21

 0905:0108  MOV DL,[123] <--- MANDA "I" A LA IMPRESORA

 0905:010C  MOV AH,5
```

```
 0905:010E  INT 21

 0905:0110  MOV DL,[124] <--- MANDA RETORNO DE CARRO A LA IMPRESORA
0905:0104  MOV AH,5

 0905:0116  INT 21

 0905:0118  MOV DL,[125]<--- MANDA LINEFEED A LA IMPRESORA

 0905:011C  MOV AH,5

 0905:011E  INT 21

 0905:0120  INT 20      <--- RETURN A DEBUG O DOS

 0905:0122   DB "HI",0D,0A <--- STRING ASCII

 GUARDE EL PROGRAMA

 -NPRINTHI.COM

 -RBX

 BX:0000
:

 -RCX

 CX:0000

 :1E
 -W                                -133 -

WRITING 0026 BYTES ( 38 BYTES DECIMAL )
 -U 100,121

 0905:0100  8A162201 MOV DL,[0122]

 0905:0104  B405     MOV AH,05

 0905:0106  CD21     INT 21

 0905:0108  8A162301 MOV DL,[0123]

 0905:010C  B405     MOV AH,05

 0905:010E  CD21     INT 21

 0905:0110  8A162401 MOV DL,[0124]

 0905:0114  B405     MOV AH,05

 0905:0116  CD21     INT 21
0905:0118  8A162501 MOV DL,[0125]

 0905:011C  B405     MOV AH,05
```

```
 0905:011E  CD21     INT 21

 0905:0120  CD20     INT 20
```

Y "d" PARA MIRAR LOS CARACTERES ASCII

```
 -d 120,125

 08F1:0120    CD    20    68    69    0D    0A
                          ^     ^     ^     ^
                          ¦     ¦     ¦     ¦
                          h     i     ¦     ¦
                                   CARRIAGE +---  LINE
                                   RETURN         FEED
-G  <--- NADA SE IMPRIME EN LA PANTALLA PERO SE IMPRIME EN
          IMPRESORA.

 PROGRAMA TERMINO NORMALMENTE
```

```
        +-----------------------------------------------+
        ¦ DIRECCIONAMIENTO INDIRECTO SIGNIFICA REFERIRSE  ¦
        ¦                                                 ¦
        ¦ A LA DIRECCION DE ALGO,EN VEZ DE TOMAR EL ALGO  ¦
        +-----------------------------------------------+
```


<< UNA MANERA MAS ELEGANTE DE IMPRIMIR EL STRING >>

```
 -A100

 0905:0100 MOV CX,31     <--- NUMERO DE CARACTERES A IMPRIMIR

 0905:0103 MOV BX,111    <--- DIRECCION DEL PRIMER CARACTER

 0905:0106 MOV DL,[BX]   <--- PONE EL CARACTER EN DL

 0905:0108 MOV AH,5      <--- IMPRIME LA FUNCION DE SALIDA DOS

 0905:010A INT 21        <--- LLAMA LA FUNCION DOS

 0905:010C INC BX        <--- INCREMENTA EL APUNTADOR

 0905:010D LOOP 106      <--- LOOP HASTA QUE SE HACE

 0905:010F INT 20        <--- REGRESA AL DEBUG O DOS

 0905:0111 db"SHE IS MOST FAIR, THOUGH SHE BE MARBLE-HEARTED",0D,0A
CHECAR CON   U 100,105  Y    D 111,141
```

EJEMPLO DE IMPRESION DE :

LPRINT CHR$(27)"E" PARA APLICACIONES ESPECIALES

```
-A100

0905:0100 MOV CX,2

0905:0103 MOV BX,111                      + LA DIRECCION A LA
                                          ¦ CUAL APUNTA BX
0905:0106 MOV DL,[BX] <------CONTENIDO [BX]
                                            NO LO QUE HAY EN BX
0905:0108 MOV AH,5

0905:010A INT 21

0905:010C INC BX

0905:010D LOOP 106

0905:010F INT 20

0905:0111 DB 1B,"E"
             ¦   ¦
             ¦   +--------LETRA "E" = CODIGO 45¦
             +-1B=CODIGO DE ESCAPE
```


BIBLIOGRAFIA

1.- INTRODUCCION AL MICROPROCESADOR 8086 / 8088 (16 BITS)
CHRISTOPER L. MORGAN , MITCHEL WAITE
BYTE BOOKS , Mc. GRAW HILL

2.- MICROCOMPUTER SYSTEMS THE 8088 / 8086 FAMILY ARQUITECTURE
PROGRAMING AND DESIGN
YO-CHENG LIU , GLEEN A. GIBSON
PRENTICE HALL

3.- ASSEMBLY LANGUAGE PREMIER FOR THE IBM PC / XT
ROBERT LAFORE
PWME / WAITE BOOK

4.- ASSEMBLY LANGUAGE PROGRAMING FOR THE IBM PERSONAL COMPUTER
DAVID J. BREDLEY
PRENTICE HALL

5.- INTRODUCTION TO DIGITAL COMPUTER DESIGN
D.J.WOLLONS
Mc. GRAW HILL

6.- MICROCOMPUTER ARCHITECTURE AND PROGRAMING
JHON F. WAKERLY
JHON WILEY & SONS.

7.- INTRODUCCION A LA LOGICA COMPUTACIONAL
DR. JORGE OLVERA
IMPRESOS Y TESIS S.A.