INTRODUCCIÓN A LA PROGRAMACIÓN EN ENSAMBLADOR DEL 8086

1. EL MICROPROCESADOR

En 1978 Intel sacó al mercado el 8086, con un bus de datos de 16 bits y capaz de direccionar hasta 1 Mb de memoria. La importancia del 8086 se debe a que fue elegido por IBM para desarrollar el Personal Computer, que fue tomado como estándar por casi todos los fabricantes de ordenadores. Posteriormente Intel fabricó sucesivamente el 80186, el 80286, el 80386, el i486 y los Pentiums, manteniendo en todos ellos la compatibilidad software con los anteriores.

1.1. MODELO DE PROGRAMACIÓN

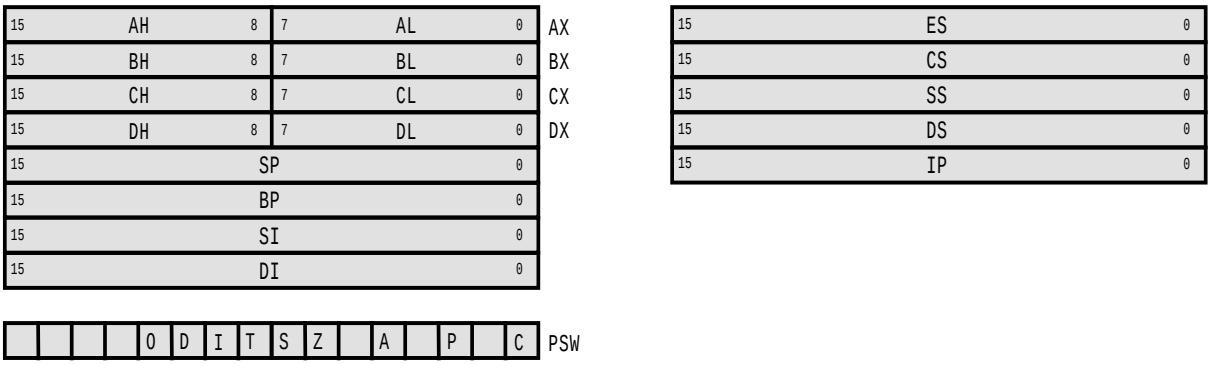

Figura 1.1. Registros del 8086

1.2. TERMINALES EN EL 8086

- Alimentación única de +5V y dos terminales de GND.
- Entrada única de reloj CLK que requiere una relación cíclica del 30%.
- Una línea de entrada de RESET.
- Terminal de entrada MN/MX# para seleccionar los modos mínimo o máximo en la CPU. En el modo mínimo la CPU genera las señales de control del sistema, está pensado para sistemas sencillos. En el modo máximo, pensado para aplicaciones complejas, es necesario un circuito controlador de bus (8288).
- Bus de datos de 16 bits, cuyas líneas están multiplexadas con las 16 líneas de menor peso del bus de direcciones (AD0-AD15).
- Bus de direcciones de 20 bits. Las cuatro líneas de mayor peso (D16-D19) están multiplexadas con cuatro de las 8 líneas indicadoras del estado del procesador.
- Ocho líneas de salida indicadoras del estado del procesador (S0-S7), multiplexadas con distintas señales.
- Terminal de salida BHE# (multiplexada con S7) que habilita la parte alta (D8-D15) del bus de datos.
- Terminal de entrada READY utilizada para sincronizar el procesador con periféricos o memorias mas lentas.

- Terminal de salida RD# para indicar operación de lectura y WR# para indicar operación de escritura en memoria o puerto de entrada/salida.
- Terminal de salida M/IO# que indica si el acceso es a memoria o a un puerto.
- Terminal de entrada TEST# que es examinado en la instrucción WAIT para determinar si se para o no el procesador hasta una interrupción.
- Dos líneas de entrada de interrupciones: INTR (enmascarable) y NMI (no enmascarable).
- Línea de salida INTA# de reconocimiento de petición de interrupción enmascarable.
- Línea de salida ALE que indica dirección válida en el bus de direcciones/datos.
- Terminal de salida DEN# que valida el traspaso de datos en el bus de datos.
- Línea de salida DT/R# que indica el sentido de la transferencia en el bus de datos.
- Terminal de entrada HOLD para que otro dispositivo tome el control del bus y terminal de salida HLDA de cesión del citado bus.
- Terminales de salida que informan sobre el estado del procesador: S0-S2 indican la operación que se está ejecutando, S3-S3 están multiplexadas con A16-A17 e indican el registro de segmento que se está utilizando y S5 (multiplexada con A18) reproduce el flag I.
- Las señales RQ/GT0# RQ/GT1# permiten la comunicación con otros procesadores del mismo bus local.
- Las señal de salida LOCK# se activa con las intrucciones que incluyen el prefijo LOCK, para indicar que no es posible la cesión del bus a ningún otro procesador.
- Las líneas QS0-QS1 informan sobre el estado de la cola interna de instrucciones.

Las señales M/IO#, WR#, INTA#, ALE, DT/R#, DEN#, HOLD y HLDA son exclusivas del modo mínimo. Las señales S0-S2, RQ/GT0#, RQ/GT1#, LOCK# y QS0-QS1 solo están disponibles en modo máximo.

1.3. REGISTRO DE ESTADO (PSW)

- C (*Carry*) Flag de acarreo.
- P (*Parity*) Flag indicador de paridad.
- A (*Auxiliar*) Flag de acarreo auxiliar (del bit 3 al bit 4).
- Z (*Zero*) Flag indicador de cero.
- S (*Sign*) Flag indicador de signo del resultado.
- T (*Trap*) Flag indicador de ejecución paso a paso.
- I (*Interrup*) Flag habilitación de interrupciones enmascarables.
- D (*Direction*) Flag de dirección ascendente/descendente en instrucciones de cadenas.
- O (*Overflow*) Flag indicador de desbordamiento.

1.4. INTERRUPCIONES

El 8086 puede tratar 256 tipos diferentes de interrupciones, que se numeran en hexadecimal entre 0x00 y 0xFF y, según el modo en el que se producen, pueden ser *internas* o *externas*. Las **interrupciones externas** son aquellas producidas por señales eléctricas procedentes del exterior del microprocesador, normalmente producidas por los periféricos. Las **interrupciones internas** son las producidas por la propia CPU en la ejecución de un programa (por ejemplo ante el intento de división por cero) o por una

instrucción específica (son las *interrupciones programadas* que se producen cuando se ejecuta la instrucción INT).

Para poder realizar la gestión de todas las interrupciones, los primeros 1024 bytes de memoria están reservadas para los *vectores de interrupción* que almacenan los valores que se deben cargar en los registros IP y CS, para ejecutar la rutina de atención a la interrupción correspondiente (la interrupción N tendrá su vector de interrupción a partir del byte de dirección Nx4).

El 8086 dispone de dos líneas de petición de interrupción hardware:

- La línea **NMI**, que provoca una interrupción no enmascarable que es atendida siempre mediante la rutina de vector 2.
- La línea **INTR**, que provoca una interrupción enmascarable, que solo será atendida si el flag F está a '1', recibiendo en las 8 líneas de menor peso del bus de datos el número del vector de interrupción. Esta línea es controlada por el circuito PIC (*Programmable Interrupt Controller*) que se encarga también de situar el número de interrupción en el bus de datos.

La propia CPU del 8086 es capaz de generar automáticamente tres interrupciones internas:

- La **interrupción 0** que se genera cuando se produce un intento de división por cero.
- La **interrupción 1** que se genera después de ejecutarse cada instrucción si el flag T del registro de estado está a '1', lo que permite la ejecución paso a paso y la depuración de programas.
- La **interrupción 4** es la interrupción de overflow y se genera cuando se produce una operación aritmética con desbordamiento.

Mediante la instrucción INT *num* se genera la interrupción representada por num, que será atendida siempre, independientemente del estado del flag I. El vector correspondiente a la interrupción provocada proporcionará la dirección de comienzo de la rutina o el programa de atención a esa interrupción, que finalizará siempre con la instrucción IRET.

2. ORGANIZACIÓN DE LA MEMORIA EN EL 8086

Aunque el 8086 dispone de 20 líneas en el bus de direcciones, la dirección de memoria se obtiene a partir de dos registros de 16 bits mediante un sistema de *segmentación de la memoria*, que equivale a subdividir la memoria en **segmentos** de 64 kb, de modo que un *registro de segmento* determina la dirección de comienzo de ese segmento de memoria y la posición relativa (off-set o desplazamiento) la fija otro registro también de 16 bits. Así mediante la combinación de ambos registros es posible direccionar cada una de las 65536 posiciones del segmento de 64 kb.

Mediante la segmentación, los 5 dígitos hexadecimales necesarios para direccionar hasta 1 Mb, se obtienen desplazando un dígito a la izquierda el número representado en el registro de segmento y sumándolo al número que representa el *desplazamiento* con respecto al origen, es decir:

DIRECCIÓN FÍSICA = SEGMENTO * 16 + DESPLAZAMIENTO

Cuando se utiliza el método segmentado, las direcciones se expresan mediante el número base y el *offset* o *desplazamiento*, ambos de 4 dígitos hexadecimales y separados por dos puntos. Por ejemplo la dirección física 56F7A se puede representar escribiendo 56F7:000A ó 5000:6F7A ó 5600:0F7A, ... Con este procedimiento, toda la memoria puede dividirse en segmentos, los cuales comienzan siempre en una dirección múltiplo de 16 y que, por supuesto, pueden solaparse.

A la vista de su modelo de programación, el 8086 puede operar con datos de tipo byte y de tipo palabra que se almacenan en memoria según el convenio de Intel, es decir, con la parte menos significativa siempre en la dirección más baja, de modo que, por ejemplo, si suponemos que AX=1234 y que TABLA representa a la dirección 100h (dentro del segmento de datos), la instrucción MOV TABLA,AX hará que se almacene 34h en la dirección 100h y 12 en la 101h.

4. MODOS DE DIRECCIONAMIENTO

En el 8086 las direcciones se construyen mediante la combinación de un registro de segmento y un registro de offset o desplazamiento (SEG:DESP), de acuerdo con las posibilidades que se muestran en la tabla 4.1.

Tabla 4.1. Combinación de registros

TIPO DE REFERENCIA A MEMORIA	SEGMENTO POR DEFECTO	SEGMENTO ALTERNATIVO	OFFSET O DESPLAZAMIENTO
Búsqueda de instrucción	**CS**	**--**	**IP**
Operación sobre la pila (stack)	**SS**	**--**	**SP**
Variables	**DS**	**CS - ES - SS**	**DIRECCIÓN EFECTIVA**
Cadenas fuente	**DS**	**CS - ES - SS**	**SI**
Cadenas destino	**ES**	**--**	**DI**
BP como registro base	**SS**	**CS - ES - SS**	**DIRECCIÓN EFECTIVA**

- INMEDIATO: No existe dirección efectiva pues el dato viene incluido en la propia instrucción.
  MOV AX, 1234h Carga en el registro AX el dato 1234h.

- POR REGISTRO: No existe dirección efectiva pues el dato está en el registro indicado.
  MOV BX,AX Copia el contenido de AX en BX

- DIRECTO: La dirección efectiva está incluida en la propia instrucción, bien mediante una expresión numérica o bien mediante una etiqueta. El segmento por defecto es DS.

MOV AX, [1234h] Guarda en el registro AL el contenido de la dirección 1234h y en AH el dato contenido en la dirección 1235h.
MOV BL, DATO Guarda en BL el byte apuntado por la etiqueta DATO.

- INDIRECTO MEDIANTE REGISTRO: La dirección efectiva es el contenido de uno de los registros BX, SI, DI ó BP. Si se utilizan BX, DI o SI el registro de segmento por defecto es DS y si se utiliza BP el registro de segmento es SS.
  MOV DX,[BX] Lleva el dato de contenido en la posición de memoria contenida en BX a DL y el contenido en la siguiente a DH.

- RELATIVO A BASE: La dirección efectiva se obtiene al sumar un desplazamiento que se incluye en la instrucción al contenido del registro BX o BP. El registro de segmento por defecto para BX es DS y para BP es SS.
  MOV DL,[BX+7Ah] Guarda en DL el contenido de la posición de memoria cuya dirección resulta de sumar 7Ah al contenido de BX. Puede escribirse MOV DL,7Ah[BX] ó MOV DL,[BX+7Ah]

- DIRECTO INDEXADO: La dirección efectiva se obtiene al sumar un desplazamiento que se incluye en la instrucción al contenido del registro DI o SI. El registro de segmento por defecto es DS.
  MOV AL,TABLA[DI] Guarda en AL el contenido de la posición de memoria cuya dirección se obtiene al sumar a la dirección representada por tabla el contenido del registro DI.

- INDEXADO RELATIVO A BASE: La dirección efectiva se obtiene un desplazamiento que se incluye en la instrucción al contenido del registro SI o DI y al contenido del registro BX o BP. Cuando se utiliza BX el registro de segmento por defecto es DS y si se utiliza BP entonces es SS.
  MOV AH,[BP+SI+7] Copia en el registro AH el contenido de la posición de memoria cuya dirección resulta de sumar los contenidos de BP y SI y el número 7. También puede escribirse MOV AH,7[BP+SI] ó MOV AH,7[BP][SI]

- DIRECCIONAMIENTO DE CADENAS: Se utiliza en instrucciones sobre cadenas. La dirección efectiva de los elementos de la cadena fuente viene dada por DS:SI y la dirección efectiva de la cadena destino viene dada por ES:DI.
  MOVSB Copia la cadena desde cuya dirección se encuentra en DS:[SI] a la posición de memoria cuya dirección se encuentra en ES:[DI].

5. LENGUAJE ENSAMBLADOR DEL 8086

El ensamblador del 8086 establece para las líneas de programa la siguiente sintaxis:

Etiqueta: Código_de_Operación Operandos ;Comentarios

El campo **Etiqueta** es opcional y debe finalizar con dos puntos «**:**». En el campo correspondiente al **Código_de_Operación** debe incluirse el mnemónico correspondiente a la instrucción. El campo **Operandos** incluirá uno o dos operandos, según la instrucción. Si son dos los operandos necesarios, se incluirán separados por una coma de acuerdo con la sintaxis Destino,Fuente. El campo **Comentarios** debe ir precedido del signo *punto y coma*.

A continuación se muestra la estructura general que debe darse a un programa en ensamblador, con las **directivas del ensamblador** de uso más frecuente.

```
                TITLE ESTRUCT.ASM

                COMMENT % Comentarios en varias líneas %
                ;Es comentario también todo lo que sigue a un signo de punto y coma

                DOSSEG
                .MODEL SMALL            ;Tipo de programa

                .STACK 100h             ;Segmento de pila: Reserva de memoria para la pila

                .DATA                   ;Comienzo del segmento de datos del programa
nombre1         EQU   000h              ;El símbolo nombre1 será sustituido por 000h
nombre2         EQU   001h
variable1       DB    00h               ;Reserva un byte de memoria lo inicializa a 00h
variable2       DB    ?                 ;Reserva un byte de memoria sin inicializar
variable3       DW    0FFFFh            ;Reserva una palabra y la inicializa a FFFFh
array1  DB      10 DUP (?)              ;Reserva 10 bytes sin inicializar a partir de array1
tabla1  DB      00,01,02,03             ;Reserva bytes de memoria y los inicializa
mensaje1        DB    "Mensaje:...",$"  ;El servicio 09h de la interrupción 21h
                                        ;escribe cadenas de caracteres en pantalla hasta
                                        ;que encuentra el signo $.
buffer  DB 255,?, 255 DUP(?)            ;Reserva un buffer de 255 caracteres preparado
                                        ;para el servicio 0Ah de la interrupción 21h.

                .CODE                   ;Comienzo del segmento de código del programa
NomMacro1  MACRO param1, param2, ...    ;DEFINICIÓN DE UNA MACRO
                LOCAL eti1              ;Etiquetas propia de la macro
                ...
eti1:           [código]
                ...
                ENDM            ;FINAL DE LA MACRO

NomProc1        PROC            ;DEFINICIÓN DE UN PROCEDIMIENTO O SUBRUTINA
                [código]
                ...
                RET
NomProc1        ENDP            ;FINAL DEL PROCEDIMIENTO

inicio:         MOV AX, @DATA   ;Comienzo del módulo principal del program
                MOV DS, AX      ;Inicialización del registro de segmento de datos
etiq1:  [código]
                ...
```

```
                NomMacro1       ;Llamada a la macro
                ...
bucle:  [código]
                ...
                CALL  NomProc1  ;Llamada a subrutina (procedimiento)
                JMP bucle       ;Instrucción de salto incondicional
                ...
FIN:            MOV AH, 4CH     ;Función 4Ch-INT 21h para finalizar regresando al DOS.
                INT 21h

                END inicio      ;Directiva para indicar por donde debe comenzar
                                ;la ejecución del programa.
```

5.1. PSEUDOINSTRUCCIONES DE USO MÁS FRECUENTE

PSEUDOINSTRUCCIÓN	DESCRIPCIÓN
=	Signo igual. Sirve para redefinir constantes. Es parecida a EQU, pero permite definir asignaciones en el curso de la ejecución. **Sintaxis:** Label = Expresión
ASSUME	Asigna un registro de segmento a un segmento.Una orden ASSUME NOTHING anula las asignaciones anteriores. **Sintaxis**: ASSUME registro:Segmento
COMMENT	Permite introducir comentarios sin colocar ';' en las líneas. Debe ir precedido de un delimitador cualquiera, el cual se repetirá al final del comentario. **Ejemplo**: COMMENT *El delimitador es el asterisco, en este caso. He aquí el final del comentario.*
.CREF	.XREF. Crea o suprime el fichero de referencias cruzadas
DB	"Define Byte", define un byte: atribuye un byte. Sirve para reservar espacios en memoria. **Sintaxis**: [nombre de la variable] DB expresión
DD	"Define Double Word", define una palabra doble byte: atribuye 4 bytes. Sirve para reservar espacios en memoria. **Sintaxis**: [nombre de la variable] DD expresión
DQ	"Define QuadWord", define una palabra cuádruple: atribuye 8 bytes. Sirve para reservar espacios en memoria. **Sintaxis**: [nombre de la variable] DQ expresión
DT	"Define Tenbytes", define 10 bytes en BCD. Sirve para reservar espacios en memoria. **Sintaxis**: [nombre de la variable] DT expresión

DW	"Define Word", define una palabra: atribuye 2 bytes. Sirve para reservar espacios en memoria. **Sintaxis**: [nombre de la variable] DW expresión
ELSE	Alternativa a una operación condicional.
END	Fin. Marca el final del programa fuente. "Expresión" proporciona la dirección de comienzo del programa. **Sintaxis:** END [Expresión]
ENDIF	Fin de la secuencia IF.
ENDM	Fin de la macroinstrucción. **Sintaxis:** ENDM
ENDP	Fin del procedimiento.
ENDS	Fin del segmento.
EQU	Equivalente. Asigna el valor de una "Expresión" a un "Nombre". **Sintaxis:** Nombre EQU Expresión
IF Expresión	Si la expresión es verdadera (no 0)
IFE Expresión	Si la expresión es falsa (= 0)
INCLUDE	Incluir. Para ensamblar en el programa una secuencia exterior. **Sintaxis:** INCLUDE programa.asm
LABEL	Define el atributo de un "Nombre". Su tipo puede ser BYTE, WORD, DWORD, un nombre de registro, estructura, NEAR, FAR. **Sintaxis:** Nombre LABEL Type
LOCAL	Crea un símbolo único para el parámetro especificado.
MACRO	Marca el principio de una macroinstrucción **Sintaxis:** Nombre de la macro MACRO [parámetro]
NAME	Nombre de un módulo **Sintaxis:** NAME Nombreatribuido
ORG	Origen. Especifica la dirección de comienzo del código **Sintaxis:** ORG Expresión o dirección

PROC	Procedimiento. Identifica un bloque de código. El atributo es FAR o NEAR. La ausencia de atributo equivale a NEAR. **Sintaxis:** Nombre del procedimiento PROC Atributo
PUBLIC	Indentifica que los símbolos pueden compartirse con otros programas. **Sintaxis:** PUBLIC símbolo
RECORD	Sirve para definir un formato para los bytes o las palabras.
SEGMENT	Marca la entrada en un segmento. Los "Types" pueden ser: PARA (tomado por defecto: el segmento debe comenzar en una dirección divisible por 16, en decimal), BYTE (el segmento puede empezar en cualquier lugar), WORD (debe comenzar por una palabra en una dirección par), PAGE (debe comenzar en una dirección divisible por 256 en decimal), PUBLIC, COMMON (todos los restantes segmentos enlazados juntos comenzarán en la misma dirección), STACK (pila: exigido por LINK), MEMORY (el segmento deberá almacenarse en memoria encima de todos los demás). **Sintaxis:** Nombre del segmento SEGMENT Type
STRUC	Similar a RECORD, pero puede procesar todas las longitudes en bytes.
SUBTIL Texto	El "Texto" es un subtítulo".
TITLE Texto	El "Texto" es un título.
.XALL	No se listan más que las macros que haya engendrado código o datos.
.XCREF	Suprime el listado del fichero de referencias.
.XLIST	Suprime el listado del código fuente.
&	Símbolo "ampersand". Sirve para concatenar texto o símbolos.
;;	Símbolo "punto y coma" doble. El comentario no aparecerá en el listado.
!	Signo de exclamación. El carácter siguiente se introduce como tal.
%Expresión	Convertir la expresión en un número del sistema de numeración actual

5.2. REPERTORIO DE INSTRUCCIONES DEL 8086

MNEMÓNICO	OPERACIÓN	MNEMÓNICO	OPERACIÓN
AAA	ASCII ADJUST AFTER ADDITION	JS	JUMP IF SIGN
AAD	ASCII ADJUST BEFORE DIVISION	JZ	JUMP IF ZERO
AAM	ASCII ADJUST AFTER MULTIPLY	LAHF	LOAD REGISTER AH FROM
AAS	ASCII ADJUST AFTER SUBTRACTION	LDS	LOAD POINTER USING DS
ADC	ADD WITH CARRY	LEA	LOAD EFFECTIVE ADDRESS
ADD	ADDITION	LES	LOAD POINTER USING ES
AND	LOGICAL AND	LOCK	LOCK THE BUS
CALL	CALL PROCEDURE	LODS	LOAD STRING (BYTE OR WORD)
CBW	CONVERT BYTE TO WORD	LODSB	LOAD STRING BYTE
CLC	CLEAR CARRY FLAG	LODSW	LOAD STRING WORD
CLD	CLEAR DIRECTION FLAG	LOOP	LOOP ON COUNT
CLI	CLEAR INTERRUPT-ENABLE FLAG	LOOPE	LOOP WHILE EQUAL
CMC	COMPLEMENT CARRY FLAG	LOOPNE	LOOP WHILE NOT EQUAL
CMP	COMPARE	LOOPNZ	LOOP WHILE NOT ZERO
CMPS	COMPARE STRING (BYTE OR WORD)	LOOPZ	LOOP WHILE ZERO
CMPSB	COMPARE STRING BYTE	MOV	MOVE (BYTE OR WORD)
CMPSW	COMPARE STRING WORD	MOVS	MOVE STRING (BYTE OR WORD)
CWD	CONVERT WORD TO DOUBLEWORD	MOVSB	MOVE STRING BYTE

DAA	**DECIMAL ADJUST AFTER ADDITION**	**MOVSW**	**MOVE STRING WORD**
DAS	**DECIMAL ADJUST AFTER SUBTRACTION**	**MUL**	**MULTIPLY, UNSIGNED**
DEC	**DECREMENT**	**NEG**	**NEGATE**
DIV	**DIVIDE, UNSIGNED**	**NOP**	**NO OPERATION**
ESC	**ESCAPE**	**NOT**	**LOGICAL NOT**
HLT	**HALT**	**OR**	**LOGICAL OR**
IDIV	**INTEGER DIVIDE, SIGNED**	**OUT**	**OUTPUT TO PORT**
IMUL	**INTEGER MULTIPLY, SIGNED**	**POP**	**POP A WORD FROM THE STACK**
IN	**INPUT BYTE OR WORD**	**POPF**	**POP FLAGS FROM THE STACK**
INC	**INCREMENT**	**PUSH**	**PUSH WORD ONTO STACK**
INT	**INTERRUPT**	**PUSHF**	**PUSH FLAGS ONTO STACK**
INTO	**INTERRUPT ON OVERFLOW**	**RCL**	**ROTATE THROUGH CARRY LEFT**
IRET	**INTERRUPT RETURN**	**RCR**	**ROTATE THROUGH CARRY RIGHT**
JA	**JUMP IF ABOVE**	**REP**	**REPEAT**
JAE	**JUMP IF ABOVE OR EQUAL**	**REPE**	**REPEAT WHILE EQUAL**
JB	**JUMP IF BELOW**	**REPNE**	**REPEAT WHILE NOT EQUAL**
JBE	**JUMP IF BELOW OR EQUAL**	**REPNZ**	**REPEAT WHILE NOT ZERO**
JC	**JUMP IF CARRY**	**REPZ**	**REPEAT WHILE ZERO**
JCXZ	**JUMP IF CX REGISTER ZERO**	**RET**	**RETURN FROM PROCEDURE**
JE	**JUMP IF EQUAL**	**ROL**	**ROTATE LEFT**
JG	**JUMP IF GREATER**	**ROR**	**ROTATE RIGHT**
JGE	**JUMP IF GREATER OR EQUAL**	**SAHF**	**STORE REGISTER AH INTO**
JL	**JUMP IF LESS**	**SAL**	**SHIFT ARITHMETIC LEFT**

JLE	**JUMP IF LESS OR EQUAL**	**SAR**	**SHIFT ARITHMETIC RIGHT**
JMP	**JUMP UNCONDITIONALLY**	**SBB**	**SUBTRACT WITH BORROW**
JNA	**JUMP IF NOT ABOVE**	**SCAS**	**SCAN STRING (BYTE OR WORD)**
JNAE	**JUMP IF NOT ABOVE OR EQUAL**	**SCASB**	**SCAN STRING BYTE**
JNB	**JUMP IF NOT BELOW**	**SCASW**	**SCAN STRING WORD**
JNBE	**JUMP IF NOT BELOW OR EQUAL**	**SHL**	**SHIFT LOGICAL LEFT**
JNC	**JUMP IF NO CARRY**	**SHR**	**SHIFT LOGICAL RIGHT**
JNE	**JUMP IF NOT EQUAL**	**STC**	**SET CARRY FLAG**
JNG	**JUMP IF NOT GREATER**	**STD**	**SET DIRECTION FLAG**
JNGE	**JUMP IF NOT GREATER OR EQUAL**	**STI**	**SET INTERRUPT ENABLE FLAG**
JNL	**JUMP IF NOT LESS**	**STOS**	**STORE STRING (BYTE OR WORD)**
JNLE	**JUMP IF NOT LESS OR EQUAL**	**STOSB**	**STORE STRING BYTE**
JNO	**JUMP IF NO OVERFLOW**	**STOSW**	**STORE STRING WORD**
JNP	**JUMP IF NO PARITY**	**SUB**	**SUBTRACT**
JNS	**JUMP IF NO SIGN**	**TEST**	**TEST**
JNZ	**JUMP IF NOT ZERO**	**WAIT**	**WAIT**
JO	**JUMP IF OVERFLOW**	**XCHG**	**EXCHANGE REGISTERS**
JP	**JUMP IF PARITY**	**XLAT**	**TRANSLATE**
JPE	**JUMP IF PARITY EVEN**	**XOR**	**EXCLUSIVE OR**
JPO	**JUMP IF PARITY ODD**		

6. EL PROGRAMA MICROSOFT CODEVIEW

El programa **CodeView** es una utilidad para la depuración de programas para entornos tipo PC, que se incluye en el Microsoft Macro Assembler versión 5.1.

Para el mejor aprovechamiento de las posibilidades del CodeView es preciso generar el programa ejecutable con las siguientes opciones:

- Ensamblar: MASM /Zi *PROGRAMA*;
- Enlazar: LINK /CO *PROGRAMA*;

La opción /Zi se emplea para especificar que se introduzcan números de línea e información de los símbolos en el fichero objeto. Esta opción se emplea para depuración.

La opción /CO (/CODEVIEW) se emplea en el momento de enlazar el programa para indicar que se desea que el enlazador incluya en el ejecutable información simbólica y sobre los números de línea. De este modo se prepara el ejecutable para la depuración con el CodeView.

Para monitores en blanco y negro conviene utilizar la opción /B (CV /B *PROGRAMA*).

6.1. LA PANTALLA DEL PROGRAMA CODEVIEW

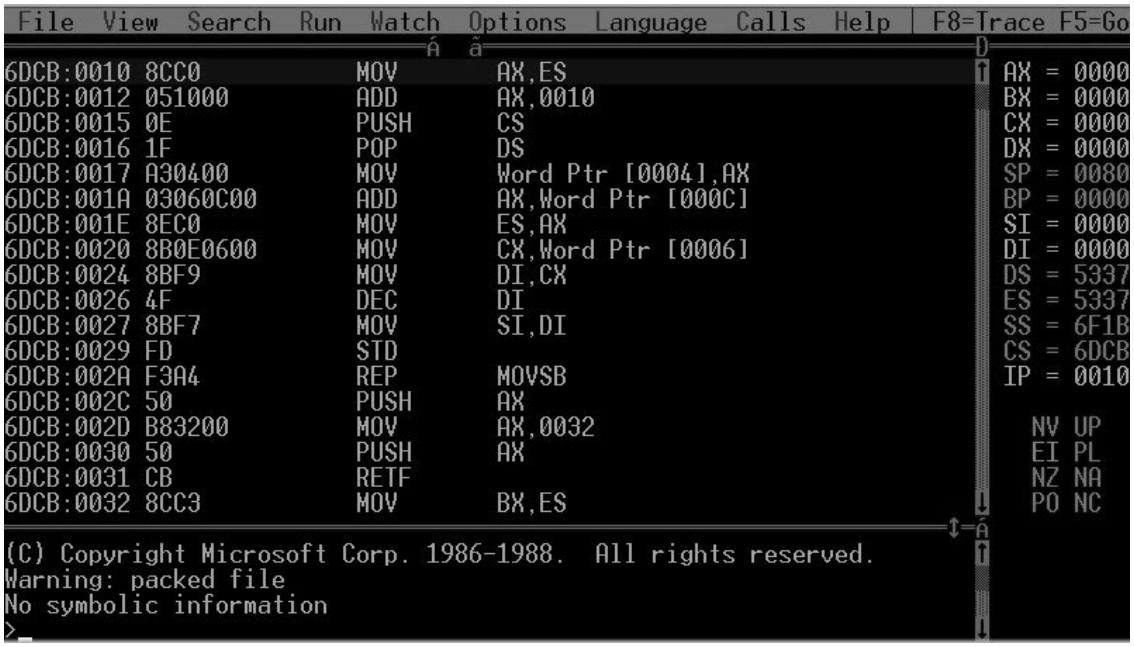

Figura 6.1. La pantalla del CodeView

6.2. LAS TECLAS

- F1 Ayuda del sistema.
- F2 Muestra/Oculta la ventana de registros.
- F3 Alterna la presentación del código fuente, mezclado y modo ensamblado.
- F4 Muestra la ventana de salida.
- F5 Ejecuta el programa hasta el final o hasta el siguiente *punto de parada*.
- F6 Cambia el cursor de la ventana de comandos a la ventana principal y viceversa.
- F7 Ejecuta el programa hasta la posición actual del cursor en la ventana principal.
- F8 Ejecuta la siguiente línea en modo paso a paso.
- F9 Pone/Quita un punto de parada en la línea actual del cursor.
- F10 Ejecuta la siguiente línea o rutina.
- Ctrl+G Aumenta el tamaño de la ventana en la que está el cursor.
- Ctrl+T Disminuye el tamaño de la ventana en la que está el cursor.
- PgUp Desplaza hacia arriba la ventana sobre la que está el cursor.
- PgDn Desplaza hacia abajo la ventana sobre la que está el cursor.
- Home Desplaza la ventana del cursor al extremo superior de su contenido.
- End Desplaza la ventana en la que está el cursor al extremo inferior de su contenido.
- ↑↓ Mueve el cursor una línea en la dirección de la fecha.

6.3. LOS MENÚS

En cada uno de los menús se despliega una ventana de opciones disponibles; sobre la ventana desplegada se puede realizar la selección deseada. Los menús disponibles son:

- **File**. Permite abrir un archivo, salir provisionalmente al DOS y salir del programa.
- **View**. Incluye las opciones para seleccionar el modo en el que se presenta el programa en la pantalla principal, visualizar la ventana de registros y visualizar la pantalla de salida.
- **Search**. Permite realizar búsquedas en el código fuente de la pantalla principal.
- **Run.** Incluye las opciones para ejecutar el programa desde su inicio, inicializar el programa, ejecutar las líneas restantes y borrar todos los puntos de parada.
- **Watch**. Permite añadir líneas de observación de expresiones y puntos de parada condicionales.
- **Options**. Permite seleccionar diversos aspectos de presentación y funcionamiento del programa CodeView.
- **Language**. Permite seleccionar el lenguaje de programación del código fuente.
- **Calls**. Muestra las diferentes llamadas a subrutinas que se han realizado en el programa.
- **Help**. Incluye los distintos capítulos sobre los que el sistema proporciona ayuda.

6.4. LAS EXPRESIONES

6.4.1. FORMATOS NUMÉRICOS EN LOS COMANDOS

En algunos comandos se pueden utilizar los siguientes formatos en la representación de datos:

- d ó i Entero decimal con signo.
- u Entero decimal sin signo.
- o Entero octal sin signo.
- x ó X Entero hexadecimal.
- f Real en coma flotante.
- e ó E Real en notación científica.
- g ó G Real (lo más compacto entre e/E ó g/G).
- c Carácter ASCII correspondiente.
- s Cadena finalizada con el carácter nulo.

6.4.2. TIPOS DE DATOS

En algunos comandos se puede seleccionar el modo de introducción o de presentación de los datos, de acuerdo con los siguientes tipos:

- B Bytes en formato hexadecimal.
- A Caracteres ASCII.
- I Números enteros con signo.
- U Números enteros sin signo.
- W Números de 2 bytes en hexadecimal.
- D Números de 4 bytes en hexadecimal.
- S Números de 4 bytes en coma flotante.
- L Números de 8 bytes en coma flotante.
- T Números de 10 bytes en coma flotante.

6.4.3. NÚMEROS DE LÍNEA

Para hacer referencia a alguna de las **líneas del programa en código fuente**, se indicará el número de línea precedida por un punto:

.númerolínea

6.4.4. DIRECCIONES DE MEMORIA

En la línea de comandos las referencias a una **dirección** deben hacerse con el formato:

[*segmento*:]*offset*

de modo que si no se indica *segmento*, se toma como segmento por defecto el contenido del registro DS (segmento de datos). El término *offset* se considera expresado en hexadecimal y puede indicarse de forma numérica o mediante una expresión válida (etiquetas o identificadores).

6.4.5. RANGOS DE DIRECCIONES DE MEMORIA

Los **rangos** de direcciones pueden expresarse de cualquiera de los modos siguientes:

dirección_inicial dirección_final
dirección_inicial L *número*

donde *dirección_inicial* y *dirección_final* representan direcciones o expresiones válidas para una dirección; *número* indica el número de posiciones a las que se hace referencia desde la *posición inicial* (el offset).

6.5. LOS COMANDOS

6.5.1. COMANDOS DE EJECUCIÓN

- **Trace** (F8): T [*número*]. Ejecuta la línea o instrucción actual entrando en las rutinas, procedimientos o interrupciones. *número* indica cuantas líneas o instrucciones se ejecutarán.
- **Program step** (F10): P [*número*]. Ejecuta la línea o instrucción actual saltando sobre las rutinas, procedimientos o interrupciones. *número* indica cuantas líneas o instrucciones se ejecutarán.
- **Go** (F5): G [*dirección*]. Ejecuta el programa restante hasta el final o hasta el próximo *punto de parada*. *Dirección* es un símbolo, un número de línea o una dirección en el que parará la ejecución.
- **Execute**: E. Ejecuta el programa restante en modo lento hasta el final o hasta el próximo *punto de parada*
- **Restart**: L. Reinicializa el programa para su ejecución desde el principio.

6.5.2. COMANDOS DE EXAMEN DE DATOS Y EXPRESIONES

- **Display expression**: ? *expresión* [,*formato*]. Evalúa el término *expresión* y muestra el resultado en el *formato* expresado opcionalmente.
- **Examine symbol**: X. Muestra los símbolos o etiquetas utilizados y su dirección.
- **Dump**: D[*tipo*][*dirección*|*rango*]. Muestra los datos contenidos en la posición (*dirección*) o posiciones de memoria indicadas (*rango*), expresadas en el *tipo* indicado.
- **Compare memory**: C *rango dirección*. Compara los bytes en la memoria indicada por la expresión *rango* con las correspondientes posiciones a partir de *dirección* (*rango* debe expresar posición inicial y número de posiciones). Como resultado aparecen las posiciones que contienen valores diferentes.
- **Search memory**: S *rango lista*. Busca en las posiciones de memoria especificadas en *rango* los valores indicados en *lista* y muestra las posiciones en las que se encuentran.
- **Port input**: I *puerto*. Lee y muestra el byte leído en la dirección de E/S dada por *puerto*.
- **Register**: R. Muestra el contenido de todos los registros, la siguiente línea de código que se va a ejecutar, la correspondiente instrucción en ensamblador, la dirección y el código de operación que contiene.
- **8087**: 7. Muestra en la ventana de diálogo el contenido de los registros del chip 8087 o del emulador instalado.

6.5.3. COMANDOS DE PUNTOS DE PARADA

- **Break set**: BP [*dirección* [*número*] "*comando*"]]. Pone un punto de parada en la línea actual o en la posición de memoria o línea representada por *dirección*; *número* indica las veces que ha de pasar el programa por ese punto para que se produzca la parada. Pueden incluirse uno o varios comandos separados por *punto y coma*, que se ejecutarán tras producirse la parada.
- **Breakpoin clear**: BC [*lista*]. Borra uno o varios puntos de parada. El término *lista* representa a una relación de números de los puntos de parada (* equivale a todos).
- **Breakpoint disable**: BD [*lista*]. Desactiva temporalmente los puntos de parada incluidos en la relación que representa *lista*.
- **Breakpoint enable**: BE [*lista*]. Activa los puntos de parada incluidos en la relación que representa *lista*.
- **Breakpoint list**: BL. Muestra la lista numerada de los puntos de parada existentes en el programa e informa del estado (*enable/disable*) de cada uno de ellos.

6.5.4. COMANDOS DE VISUALIZACIÓN

- **Watch expression**: W? *expresión* [,*formato*]. Muestra el valor del término *expresión*, que representa a una variable o a una combinación válida de variables y operadores en el *formato* que de forma opcional puede seleccionarse.
- **Watch memory**: W[*tipo*] *dirección*|*rango*. Muestra el valor contenido en la posición o posiciones de memoria determinadas por el término *dirección*|*rango*, de acuerdo con el *tipo* indicado opcionalmente (byte, ascii, word, etc.).
- **Watchpoint**: WP? *expresión*. Realiza un *punto de parada* cuando el término *expresión* es evaluado como cierto (distinto de cero), siempre que se trate de una expresión válida.
- **Tracepoint (expresión)**: TP? *expresión*. Realiza un *punto de parada* cuando cambia de *expresión*.
- **Tracepoint (memoria)**: TP [*tipo*] *dirección*|*rango*. Realiza un *punto de parada* cuando cambia el valor contenido en la posición o posiciones de memoria determinadas por el término *dirección*|*rango*.
- **Watch delete**: Y *número*. Borra el punto de parada señalado por el término *número* y establecido con los comandos Watch, Watchpoint o Tracepoint (Y* borra todos).
- **Watch list**: W. Muestra la lista numerada de los puntos de parada establecidos con los comandos Watchpoint y Tracepoint.

6.5.5. COMANDOS DE ANÁLISIS DEL CÓDIGO

- **Set mode**: S[+|-|&]. Cambia el modo de presentación del código en la ventana principal entre el modo fuente (+), modo ensamblado (-) o modo mezclado (&).
- **Unassemble**: U [*dirección*|*rango*]. Muestra el contenido y el código en ensamblador correspondiente a la *dirección* o al *rango* de direcciones indicado.
- **View**: V [*expresión*]. Hace que aparezca en la pantalla principal la línea en la que aparece la *expresión* que se indica o la línea de código señalada (.*línea*).
- **Current location**: .. Un punto en la línea de comandos hace que aparezca en el centro de la pantalla la próxima línea que se va a ejecutar.

6.5.6. COMANDOS DE MODIFICACIÓN DE CÓDIGO O DATOS

- **Assemble**: A [*dirección*]. Permite ensamblar una instrucciones e introducir los códigos de operación correspondientes a partir de la *dirección* que se señala.
- **Enter**: E[*tipo*] *dirección* [*lista*]. Carga a partir de la posición de memoria indicada por *dirección*, los datos que se incluyen en el término *lista*. El término *tipo* indica el tipo de dato que será introducido (byte, word, ascii, etc.).
- **Fill memory**: F *rango lista*. Rellena las posiciones de memoria especificadas por *rango* con los valores incluidos en *lista*.
- **Move memory**: M *rango dirección*. Copia el contenido del bloque de memoria especificado por *rango* a otro bloque del mismo tamaño que comienza en *dirección*.
- **Port output**: O *puerto dato*. Envía *dato* al puerto de entrada salida cuya dirección es *puerto*.
- **Register**: R [*registro* [[=] *expresión*]]. Muestra el valor contenido en el *registro* indicado y permite cambiar ese valor. Pueden utilizarse los siguientes nombres de registros: AX BX CX DX CS DS SS ES SP BP SI DI IP F. Los bits del registro de estado (F) pueden modificarse individualmente entre cada uno de los dos estados posibles (RF *SET*|*CLEAR*):

Tabla 6.1. Indicadores

NOMBRE	SET	CLEAR
Overflow	**OV**	**NV**
Dirección	**DN**	**UP**
Interrupción	**EI**	**DI**
Signo	**NG**	**PL**
Cero	**ZR**	**NZ**
Acarreo auxiliar	**AC**	**NA**
Paridad	**PE**	**PO**
Acarreo	**CY**	**NC**

6.5.7. COMANDOS DE CONTROL DEL SISTEMA

- **Help**: H. Presenta la primera pantalla de ayuda.
- **Quit**: Q. Abandona el programa CodeView.
- **Radix**: N [*número*]. Cambia la base de numeración en la que se expresan los números para entrar argumentos y visualizar el valor de las expresiones por defecto. Los valores posibles de *número* son 8 (octal), 10 (decimal) y 16 (hexadecimal).
- **Redraw**: @. Redibuja la pantalla de CodeView cuando aparecen interferencias en esta causadas por funcionamientos anómalos.
- **Screen exchange** (F4): \. Presenta la ventana de salida de los programas.
- **Search**: /*expresión*. Sitúa el cursor sobre la línea en la que aparece *expresión.*
- **Shell escape**: ![*comando*]. Permite salir temporalmente al sistema operativo D.O.S. (se regresará tecleando EXIT). Si se incluye *comando* será un comando del D.O.S. que ejecutará fuera del entorno del programa CodeView de forma temporal.
- **Tab set**: #*número*. Establece el *número* de caracteres a los que equivale el código del tabulador.
- **Option**: O[*opcion*[+|-]]. Activa (+) o desactiva (-) una de las opciones (F, B, C, 3) del menú Options.