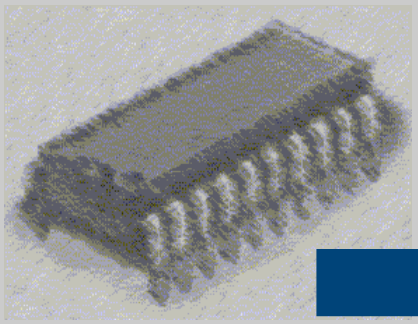
Soluciones a los problemas impares

Tema 4. Lenguaje máquina y lenguaje ensamblador



Estructura de Computadores

I. T. Informática de Gestión / Sistemas

Curso 2008-2009

Base teórica

El repertorio de instrucciones indica todas las operaciones que el computador es capaz de ejecutar. Cada instrucción es una colección de ceros y unos que indica qué hacer, con que operandos y dónde dejar el resultado. A las instrucciones expresadas mediante ceros y unos se les llama instrucciones máquina, y al conjunto de todas las instrucciones máquina del computador se le conoce con el nombre de lenguaje máquina.

Como trabajar con ceros y unos es tedioso y propenso a cometer errores, aunque se agrupen en dígitos hexadecimales, surgen las instrucciones en ensamblador y el lenguaje ensamblador. En este lenguaje, cada instrucción ensamblador se corresponde con una instrucción máquina, pero permite determinar de manera más clara de que instrucción se trata, con que operandos trabaja y dónde deja el resultado. La siguiente tabla muestra un ejemplo entre el código máquina de la instrucción y el código en ensamblador.

Instrucción	Transferir a la posición de memoria 5333h el dato contenido en el registro AX
Lenguaje máquina	1010 0011 0011 0011 0101 0011 (A33353h)
Lenguaje ensamblador	MOV [5333], AX

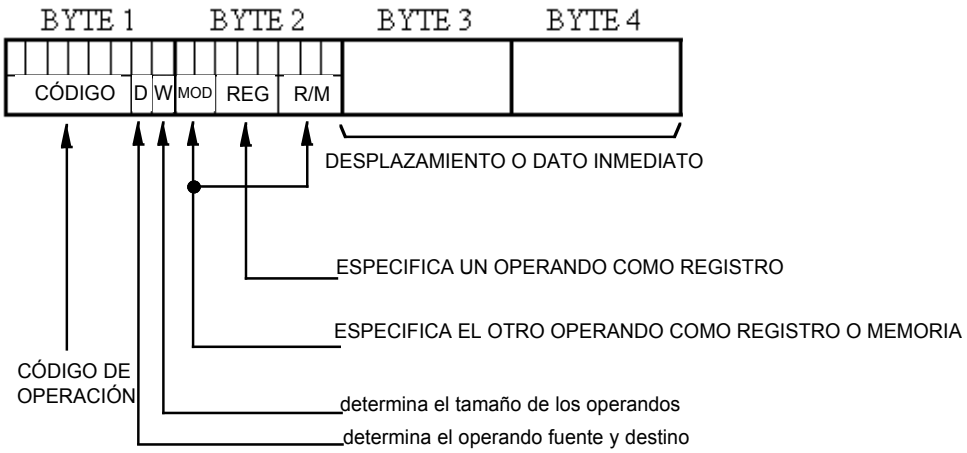
Tabla 1: ejemplo de instrucciones máquina y ensamblador.

Los juegos de instrucciones pueden contener muchas instrucciones con muchos modos de direccionamiento (repertorios CISC) o ser pocas instrucciones con pocos modos de direccionamiento (repertorios RISC). Lo que si tienen en común es que deben contener toda la información para ser ejecutadas:

- Código de operación.
- Operandos y lugar en el que dejar los resultados.
- Dirección de la instrucción siguiente.

Formato de instrucción registro-registro, registro-memoria

Las instrucciones que tienen como operandos a dos registros o a dos registros y una posición de memoria tienen el formato siguiente en el i8086:



Además se tiene las tablas de codificación siguientes:

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

Tabla codificación del operando REG

MOD = 11			CÁLCULO DE LA DIRECCIÓN FÍSICA			
R/M	W=0	W=1	R/M	**MOD = 00**	**MOD = 01**	**MOD =10**
000	AL	AX	000	[BX]+[SI]	[BX]+[SI] + Desplaz.8	[BX]+[SI] + Desplaz.16
001	CL	CX	001	[BX]+[DI]	[BX]+[DI] + Desplaz.8	[BX]+[DI] + Desplaz.16
010	DL	DX	010	[BP]+[SI]	[BP]+[SI] + Desplaz.8	[BP]+[SI] + Desplaz.16
011	BL	BX	011	[BP]+[DI]	[BP]+[DI] + Desplaz.8	[BP]+[DI] + Desplaz.16
100	AH	SP	100	[SI]	[SI] + Desplaz.8	[SI] + Desplaz.16
101	CH	BP	101	[DI]	[DI] + Desplaz.8	[DI] + Desplaz.16
110	DH	SI	110	Dirección directa	[BP] + Desplaz.8	[BP] + Desplaz.16
111	BH	DI	111	[BX]	[BX] + Desplaz.8	[BX] + Desplaz.16

Tabla de codificación para el operando R/M en función del modo de direccionamiento MOD

Modos de direccionamiento

Indican el lugar en el que se encuentran los operandos o en los que se debe dejar el resultado. Los modos de direccionamiento pueden ser:

- **Inmediato**. El operando viene expresado en el código máquina de la instrucción.
- **Directo a registro**. El operando se encuentra en el registro indicado por el código máquina de la instrucción.
- **Directo a memoria**. El operando se encuentra en la dirección de memoria contenida en la instrucción
- **Relativo**. La dirección de memoria en la que se encuentra el operando se calcula sumando un desplazamiento indicado en la instrucción al contenido de un registro también indicado en la instrucción. Dependiendo del registro que se emplee se denomina de diferentes maneras: relativo a registro base, a registro índice, a contador de programa o a pila.
- **Indirecto**. La dirección de memoria indicada en la instrucción es a su vez la dirección de memoria en la que se encuentra el operando. La

dirección puede expresarse mediante direccionamiento directo a memoria o mediante direccionamiento relativo.

- **Implícito**. Alguno o todos los operandos de la instrucción no se muestran explícitamente en la instrucción sino que trabaja con unos fijos.

EL i80x86/808

El i8086 es un microprocesador de 16 bits que está internamente formado por dos procesadores la EU (Execution Unit) y la BIU (Bus Interfaces Unit). La EU es la unidad prioritaria. La BIU esla encargada de la búsqueda de instrucciones en memoria, de la lectura y escritura de memoria y de los puertos. Un esquema del i8086 se muestra en la siguiente figura.

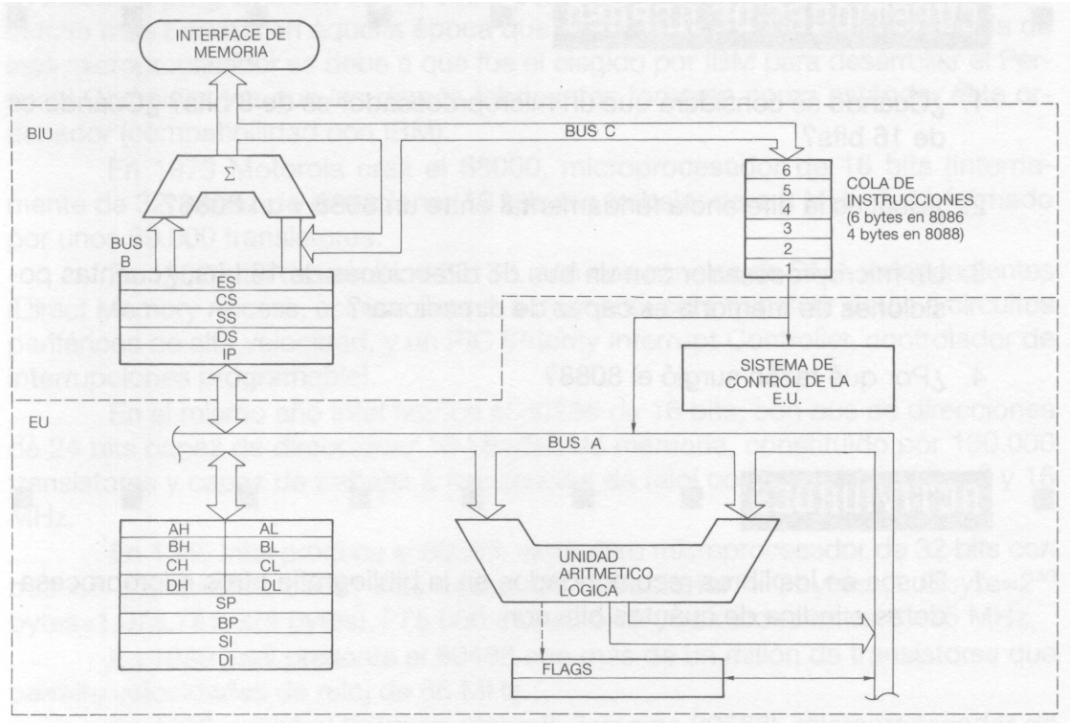


Figura 1: estructura de bloques del i8086.

Aunque el i8086 es de 16 bits, puede acceder a 1MB de memoria gracias a la segmentación, operación consistente en dividir el mega en trozos de 64KB e indicar en que trozo y en que dirección de los 64KB se encuentra.

Para llevar a cabo la segmentación existen los llamados registros de segmento:

- CS: segmento de código.
- SS: segmento de pila.
- DS: segmento de datos.
- ES: segmento extra.

La manera de obtener la dirección física de memoria es:

Dirección física = Registro de segmento x 10h + Dirección física

Finalmente en el i8086 los datos mayores de un byte se almacena en el formato little endian, es decir, el byte más bajo ocupa la dirección más baja de memoria y el byte más significativo la más alta (para datos de 16 bits).

Repertorio de instrucciones del i8086

A continuación se describen brevemente algunas de las instrucciones del i8086, de las directivas y de las funciones de interrupción más empleadas en clase y en el laboratorio.

Instrucciones del i8086

Instrucciones de trasferencia de datos

- **Nombre:** **MOV**
- **Formato:** **MOV destino, origen**
- **Descripción:**
  Transfiere un byte o una palabra desde el operando origen al operando destino.

- **Nombre:** **PUSH**
- **Formato:** **PUSH origen**
- **Descripción:**
  Decrementa el puntero de pila (SP) en 2 y luego transfiere la palabra que se ha especificado en el operando origen a lo alto de la pila.

- **Nombre:** **POP**
- **Formato:** **POP destino**
- **Descripción:**

  Transfiere un byte o una palabra desde la cima de la pila al operando destino y luego incrementa la pila en 2.

Instrucciones de bifurcación.

- **Nombre:** **CALL**
- **Formato:** **CALL destino**
- **Descripción:**

  Bifurca a un procedimiento, salvando antes la dirección de la instrucción siguiente en la pila para poder volver a dicha instrucción una vez ejecutado el procedimiento.

  El procedimiento puede estar dentro del mismo segmento (llamada NEAR) o en otro segmento (llamada FAR).

- **Nombre:** **RET**
- **Formato:** **RET**
- **Descripción:**

  Retorna de un procedimiento a la dirección salvada en la pila. Dependiendo de si se vuelve de un procedimiento NEAR o de un procedimiento FAR el retorno se hace de forma diferente. En el primer caso, se quita de la cima de la pila una palabra que corresponde a la dirección de retorno. En el caso de un procedimiento FAR se quitan dos palabras, la primera se corresponde con el desplazamiento y las segunda al segmento de la dirección de retorno.

- **Nombre:** **INT**
- **Formato:** **INT tipo_interrupción**

- **Descripción:**

  INT activa el procedimiento de interrupción especificado por el operando. La dirección del vector de interrupción se calcula multiplicando por 4 el operando, que es un valor entre 0 y 255.

  El vector de interrupción se compone de dos palabras: la primera palabra es el desplazamiento y la segunda el segmento.

- **Nombre:** **IRET**
- **Formato:** **IRET**
- **Descripción:**

  Devuelve el control a la dirección de retorno salvada en la pila y restaura los flags. Se emplea para finalizar un procedimiento de interrupción.

- **Nombre:** **LOOP**

- **Formato:** **LOOP desplazamiento**
- **Descripción:**

  Si CX es diferente de cero, entonces IP = IP + desplazamiento. Si CX es cero entonces ejecuta la instrucción siguiente. El desplazamiento debe estar comprendido entre -128 y 127.

  Mediante esta instrucción es posible implementar bucles. También son factibles los bucles anidados pero debemos hacer uso de la pila.

- **Nombre:** **JMP**
- **Formato:** **JMP dirección.**
- **Descripción:**

  Realiza un salto incondicional. La bifurcación puede ser dentro del mismo segmento, en cuyo caso IP se sustituye por el valor del desplazamiento. Si la bifurcación es a otro segmento se sustituyen los valores correspondientes a CS y a IP.

  La bifurcación puede ser especificando una etiqueta o la dirección.

- **Nombre:** **J XXX**
- **Formato:** **J {condición de salto} dirección**
- **Descripción:**

  Salta a la dirección especificada en el operando si es cierta la condición. En caso de no satisfacerse la condición se ejecuta la instrucción siguiente. (* comparación sin signo)

  Las condiciones de salto son:

Instrucción	Bifurca si la condición es:
JA*	superior
JAE*	superior o igual
JB*	inferior
JBE*	inferior o igual
JC	acarreo
JCXZ	CX = 0
JE	igual
JG	mayor
JGE	mayor o igual
JL	menor
JLE	menor o igual
JNA*	no superior
JNAE*	no superior ni igual
JNB*	no inferior
JNBE*	no inferior no igual
JNC	no acarreo
JNE	no igual
JNG	no mayor
JNGE	no mayor ni igual
JNL	no menor

Instrucción	Bifurca si la condición es:
JNLE	no menor o igual
JNO	no desbordamiento
JNP	no paridad
JNS	no signo(positivo)
JNZ	no cero
JO	overflow
JP	paridad
JPE	paridad par
JPO	paridad impar
JS	signo (negativo)
JZ	Cero
** son números sin signo*	

Instrucciones aritméticas.

- **Nombre:** **ADD**
- **Formato:** **ADD destino, origen**
- **Descripción:**
  Suma los dos operandos y el resultado lo deja en el operando destino. Los operandos deben ser del mismo tipo.

- **Nombre:** **ADC**
- **Formato:** **ADC destino, origen**
- **Descripción:**
  Suma los dos operandos. Suma uno si estás activado el flag de acarreo. El resultado se almacena en el operando destino. Además los operandos deben ser del mismo tipo.

- **Nombre:** **SUB**
- **Formato:** **SUB destino, origen**
- **Descripción:**
  Resta el operando origen del operando destino. El resultado se almacena en el operando destino y además, ambos operandos deben ser del mismo tipo.

- **Nombre:** **SBB**
- **Formato:** **SBB destino, origen**
- **Descripción:**
  Resta el operando origen del operando destino. Resta uno si el flag de acarreo está activo. Los operandos deben ser del mismo tipo. El resultado se almacena en el operando destino.

- **Nombre:** **MUL**
- **Formato:** **MUL origen**
- **Descripción:**

Multiplica, sin considerar el signo el acumulador (AL o AX) por el operando origen. Si el operando origen es de tipo byte el resultado se almacena en AX. Si es de tipo palabra se almacena en DX (palabra superior) y AX (palabra inferior).

- **Nombre:** **IMUL**
- **Formato:** **IMUL origen**
- **Descripción:**

  Multiplica, considerando el signo, el acumulador AL o AX por el operando origen. Si el operando fuente es un byte se almacena el resultado en AX. Si se trata de una palabra, se almacena en DX (palabra superior) AX (palabra inferior).

- **Nombre:** **DIV**
- **Formato:** **DIV origen**
- **Descripción:**

  Divide, sin considerar el signo, el acumulador AL o AX y su extensión (AH o DX) por el operando origen. El resultado se almacena en AL o AX, según el operando sea de un byte o de una palabra. El resto se almacena en la extensión del acumulador AH o DX.

- **Nombre:** **IDIV**
- **Formato:** **IDIV origen**
- **Descripción:**

  Divide, considerando el signo, el acumulador AL o AX y su extensión (AH o DX) por el operando origen. El resultado se almacena en AL o AX, según el operando sea de un byte o de una palabra. El resto se almacena en la extensión del acumulador AH o DX.

- **Nombre:** **INC**
- **Formato:** **INC destino**
- **Descripción:**

  Suma una unidad al operando destino. El operando puede ser de tipo byte o palabra.
- **Nombre:** **DEC**
- **Formato:** **DEC destino**
- **Descripción:**

  Resta una unidad al operando destino. El operando puede ser de tipo byte o palabra.

- **Nombre:** **NEG**
- **Formato:** **NEG destino**
- **Descripción:**

  Cambia de signo mediante el complemento a 2 del operando destino. Deja el resultado en el operando destino. El operando puede ser de tipo byte o palabra.

Instrucciones de comparación.

- **Nombre:** **CMP**
- **Formato:** **CMP destino, origen**
- **Descripción:**

  Resta el operando origen del operando destino pero no devuelve el resultado. Los operandos son del tipo byte o palabra. Se alteran los flag dependiendo del resultado.

Instrucciones lógicas.

- **Nombre:** **AND**
- **Formato:** **AND destino, origen**
- **Descripción:**

  Es una operación Y lógica a nivel de bits. El resultado se almacena en destino.

A	B	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

- **Nombre:** **OR**
- **Formato:** **OR destino, origen**
- **Descripción:**

  Es una operación O lógica a nivel de bits. El resultado se almacena en destino.

A	B	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

- **Nombre:** **OR**
- **Formato:** **XOR destino, origen**
- **Descripción:**

  Es una operación O lógica EXCLUSIVO a nivel de bits. El resultado se almacena en destino.

A	B	a OR b
0	0	0
0	1	1
1	0	1
1	1	0

- **Nombre:** **NOT**
- **Formato:** **NOT destino**
- **Descripción:**

  Realiza el complemento a 1.

A	NOT a
0	1
1	0

Instrucciones de desplazamiento.

- **Nombre:** **SAR**
- **Formato:** **SAR destino, contador**
- **Descripción:**

  Desplaza a la derecha los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **SAL**
- **Formato:** **SAL destino, contador**
- **Descripción:**

  Desplaza a la izquierda los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **SHR**
- **Formato:** **SHR destino, contador**
- **Descripción:**
  Desplaza a la derecha de manera lógica los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **SHL**
- **Formato:** **SHL destino, contador**
- **Descripción:**
  Desplaza a la izquierda de forma lógica los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **ROR**
- **Formato:** **ROR destino, contador**
- **Descripción:**
  Rota a la derecha los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **ROL**
- **Formato:** **ROL destino, contador**
- **Descripción:**
  Rota a la izquierda los bits del operando destino el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.
- **Nombre:** **RCR**
- **Formato:** **RCR destino, contador**
- **Descripción:**
  Rota a la derecha los bits del operando destino y el flag de acarreo el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

- **Nombre:** **RCL**
- **Formato:** **RCL destino, contador**
- **Descripción:**
  Rota a la izquierda los bits del operando destino y el flag de acarreo CF el número de bits especificado en contador. Si el número de bits a desplazar es 1 se puede especificar directamente. Si es mayor que uno **se debe cargar el valor en CL**.

Instrucciones de bit.

- **Nombre:** **TEST**
- **Formato:** **TEST destino, fuente**
- **Descripción:**
  Es igual que la operación AND (y lógico) pero no se guarda el resultado en destino. Se realiza a nivel de bit y modifica los flag de estado

- **Nombre:** **CLI**
- **Formato:** **CLI**
- **Descripción:**
  Desactiva las interrupciones. Las interrupciones no enmascarables no se pueden inhibir.

- **Nombre:** **STI**
- **Formato:** **STI**
- **Descripción:**
  Permite las interrupciones.

- **Nombre:** **CLC**
- **Formato:** **CLC**
- **Descripción:**
  Desactiva el flag de acarreo.

- **Nombre:** **STC**
- **Formato:** **STC**
- **Descripción:**
  Pone a 1 el flag de acarreo

Instrucciones de entrada /salida.

- **Nombre:** **IN**
- **Formato:** **IN acumulador, puerta**
- **Descripción:**
  Transfiere un byte o una palabra de una puerta de entrada del procesador al registro AL o AX. El número de la puerta se puede especificar de las formas siguientes:

  - Un valor variable almacenado en el registro DX
  - Un valor fijo entre 0 y 255.

- **Nombre:** **OUT**
- **Formato:** **OUT acumulador, puerta**
- **Descripción:**

  Transfiere un byte o una palabra A una puerta de entrada del procesador desde el registro AL o AX. El número de la puerta se puede especificar de las formas siguientes:

  - Un valor variable almacenado en el registro DX
  - Un valor fijo entre 0 y 255.

Instrucciones varias.

- **Nombre:** **WAIT**
- **Formato:** **WAIT**
- **Descripción:**

  Permite la comunicación con otros coprocesadores. Pone al procesador en un estado de espera hasta que se activa la línea TEST. La instrucción WAIT comprueba la línea TEST cada cinco intervalos de reloj.

- **Nombre:** **HLT**
- **Formato:** **HLT**
- **Descripción:**

  Para el procesador. Volverá a trabajar si:
  - Se hace RESET en el ordenador.
  - Se recibe una interrupción no enmascarable.
  - Se recibe una interrupción enmascarable siempre que se permitan las interrupciones, es decir, que previamente no las hayamos deshabilitado con CLI.

- **Nombre:** **NOP**
- **Formato:** **NOP**
- **Descripción:**

El procesador no hace nada y pasa a ejecutar la instrucción siguiente.

DIRECTIVAS.

Directivas de datos.

- **Nombre:** **EQU**

- **Formato:** **nombre EQU expresión**
- **Descripción:**
  Asigna a nombre el valor de una expresión, que puede ser:
  - ❍ Una constante numérica.
  - ❍ Otro nombre.
  - ❍ Cualquier operación con números y nombres que de un resultado numérico.
  - ❍ Una referencia a una dirección empleando cualquier tipo de direccionamiento.

Nombre: **DB**
- **Formato:** **nombre DB expresión [, ...]**
- **Descripción:**
  Reserva memoria para una variable de 8 bits, nombre es opcional y únicamente identifica el primer byte.
- **Nombre:** **DD**
- **Formato:** **nombre DD expresión [, ...]**
- **Descripción:**
  Reserva memoria para una variable de tipo doble palabra, es decir 4 bytes ó 32 bits, nombre es opcional y únicamente identifica el primer byte.

- **Nombre:** **DQ**
- **Formato:** **nombre DQ expresión [, ...]**
- **Descripción:**
  Reserva memoria para una variable de tipo cuádruple palabra, es decir 64 bits, nombre es opcional y únicamente identifica el primer byte.

- **Nombre:** **DT**
- **Formato:** **nombre DT expresión [, ...]**
- **Descripción:**
  Reserva memoria para una variable de tipo diez bytes de memoria, es decir 80 bits, nombre es opcional y únicamente identifica el primer byte.

- **Nombre:** **DW**
- **Formato:** **nombre DW expresión [, ...]**
- **Descripción:**
  Reserva memoria para una variable de tipo palabra, es decir 2 bytes ó 16 bits, nombre es opcional y únicamente identifica el primer byte.

- **Nombre:** **PUBLIC**
- **Formato:** **PUBLIC símbolo [, ...]**
- **Descripción:**
  Permite que los símbolos especificados sean accesibles por otros módulos.

- **Nombre:** **EXTRN**
- **Formato:** **EXTRN nombre:tipo [, ...]**
- **Descripción:**
  Sirve para poder hacer referencia a símbolos definidos en otros módulos.

- **Nombre:** **END**
- **Formato:** **END [expresión]**
- **Descripción:**
  Indica el final del programa fuente. La expresión es una etiqueta que sirve para identificar el comienzo del programa fuente.

- **Nombre:** **PROC**
- **Formato:** **nombre PROC [atributo]**
- **Descripción:**

Indica el comienzo del procedimiento nombre. Un procedimiento es un bloque de instrucciones.

- **Nombre:** **ENDP**
- **Formato:** **nombre ENDP**
- **Descripción:**
  Indica el final del procedimiento.

- **Nombre:** **STRUCT**
- **Formato:** **nombre STRUCT**
- **Descripción:**
  Define una plantilla de campos a nivel de bytes. Una vez definida la estructura, su nombre se puede usar para reservar e inicializar memoria.

Funciones de la interrupción 21h.

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

Función	Descripción
AH = 1h	**Entrada desde el teclado**. Esta función espera a que se teclee un carácter por teclado. Escribe el carácter en pantalla y devuelve el código ASCII en el registro AL
	Modifica **AL** con el código ASCII del carácter leído.
AH = 2h	**Salida a la pantalla**. Muestra un carácter en pantalla. Se debe guardar en **DL** el código ASCII del carácter que se desea sacar por pantalla.
	Modifica **DL** se le asigna el código ASCII del carácter que se desea sacar por pantalla.
	Modifica **AL** devuelve un código de error.
AH = 8h	**Entrada desde el teclado sin reproducir el carácter por la pantalla**. Lee un carácter por pantalla pero no lo muestra por pantalla.
	Modifica **AL** con el código ASCII del carácter leído.
AH = 9h	**Muestra cadena**. Muestra por pantalla la cadena a la que apunta la pareja de registros DS:DX. El final de la cadena se debe marcar con el carácter $.
AH = 0Ah	**Lee cadena**. Lee una cadena desde el teclado.
AH = 4Ch	**Sale al DOS.** Devuelve el control al DOS, igual que la interrupción 20h. Devuelve en **AL** el código de retorno al DOS que se desee.
	Modifica **AL** con el valor que se desea devolver al DOS para ser usado con el IF ERRORLEVEL.

1. Se tiene la siguiente instrucción del i80x86: MOV AL, Numero:
   - Se sabe que el contador de programa IP vale 6.
   - La instrucción ocupa 3 bytes.
   - Numero hace referencia a la posición de memoria 000Ah
   - El registro CS vale 51D7h
   - El registro DS vale 51C7h

   Se pide:

   a) Calcular cuál será la dirección física de memoria de la siguiente instrucción.

   b) b) Calcular la dirección física de memoria de la que se lee el dato.

---

2. Realiza el mismo ejercicio que en el caso anterior pero suponiendo que IP vale 4 y Numero hace referencia a la posición 031Dh

---

3. Se tiene la siguiente instrucción XOR AX, AX:

- IP = 15h.
- CS = 51D7h
- DS = 51C7h
- La instrucción ocupa 2 bytes

Se pide:

a) Calcular cuál será la dirección física de memoria para la instrucción siguiente.

b) Calcular la dirección física de memoria de la que se lee el dato.

---

4. Realiza el mismo ejercicio que en el caso anterior pero suponiendo que IP vale 7 y Numero hace referencia a la posición 031Dh.

---

5. Sea la instrucción MUL BX
   - IP = 4:
   - La instrucción ocupa 2 bytes
   - CS = 5180h
   - DS =2345h

   Se pide:

   a) Calcular cuál será la dirección física de memoria para la instrucción siguiente.
   b) Calcular la dirección física de memoria en la que se guarda el resultado.

---

6. Sea la instrucción DIV BX
   - IP = 4:
   - La instrucción ocupa 2 bytes
   - CS = 5765h
   - DS =2AA5h

   Se pide:

   c) Calcular cuál será la dirección física de memoria para la instrucción siguiente.
   d) Calcular la dirección física de memoria en la que se guarda el resultado.

---

7. Se tiene la instrucción en ensamblador ADD AL, [BX]:

- El contador de programa IP vale 9
- El registro BX tiene el valor 1500h
- El registro CS es igual a 8000h
- El registro DS vale 9250h
- La instrucción ocupa 2 bytes

Se pide:

a) Calcular cuál será la dirección física de memoria de la siguiente instrucción.

b) Calcular la dirección física de memoria de la que se obtiene el operando.

---

8. Se tiene la instrucción en ensamblador ADC AL, [BX]:

- El contador de programa IP vale 9
- El registro BX tiene el valor AB00h
- El registro CS es igual a 8008h
- El registro DS vale 9229h
- La instrucción ocupa 2 bytes

Se pide:

a) Calcular cuál será la dirección física de memoria de la siguiente instrucción.

b) Calcular la dirección física de memoria de la que se obtiene el operando.

---

9. Se tiene la instrucción en ensamblador ADD AL, [BX][SI]:

   - El contador de programa IP vale 8
   - El registro BX tiene el valor 1800h
   - El registro SI vale FFFFh
   - El registro CS es igual a 9000h
   - El registro DS vale 9250h
   - La instrucción ocupa 2 bytes

   Se pide:

   a) Calcular cuál será la dirección física de memoria de la siguiente instrucción.

   b) Calcular la dirección física de memoria de la que se obtiene el operando:

10. Se tiene la instrucción en ensamblador SBB AL, [BX][DI]:

    - El contador de programa IP vale 8
    - El registro BX tiene el valor B800h
    - El registro DI vale 0FFFh
    - El registro CS es igual a 9CD0h
    - El registro DS vale 4570h
    - La instrucción ocupa 4 bytes

    Se pide:

    a) Calcular cuál será la dirección física de memoria de la siguiente instrucción.

    b) Calcular la dirección física de memoria de la que se obtiene el operando:

11. Se tiene la instrucción en ensamblador ADD Numero[SI], AL:

    - El contador de programa IP vale 9
    - El registro SI tiene el valor 2500h
    - El registro CS es igual a A000h
    - El registro DS vale A250h
    - La etiqueta Numero hace referencia a la posición 5000h
    - La instrucción ocupa 3 bytes

    Se pide:

    a) Calcular cuál será la dirección física de memoria de la siguiente instrucción.
    b) Calcular la dirección física de memoria en la que se deja el resultado

12. Se tiene la instrucción en ensamblador MOV Numero[SI], AL:

    - El contador de programa IP vale Ah
    - El registro SI tiene el valor A500h
    - El registro CS es igual a A500h
    - El registro DS vale A850h
    - La etiqueta Numero hace referencia a la posición A000h
    - La instrucción ocupa 5 bytes

    Se pide:

    a) Calcular cuál será la dirección física de memoria de la siguiente instrucción.
    b) Calcular la dirección física de memoria en la que se deja el resultado

13. ¿Cuál es el código máquina para la instrucción del i8086 ADD AL, [BX] sabiendo que el código de operación de la instrucción ADD es 000000?

14. ¿Cuál es el código máquina para la instrucción del i8086 ADD [BX], AL sabiendo que el código de operación de la instrucción ADD es 000000?

15. ¿Cuál es el código máquina para la instrucción SUB AL, [BX][SI] del i8086 sabiendo que el código de operación de la instrucción SUB es 001010?

16. ¿Cuál es el código máquina para la instrucción SUB [BX][SI], BL del i8086 sabiendo que el código de operación de la instrucción SUB es 001010?

17. ¿Cuál es el código máquina para la instrucción ADD Numero[SI], BL del i8086 sabiendo que el código de operación de la instrucción ADD es 000000 y que Numero referencia la posición 000Ah de memoria?

18. ¿Cuál es el código máquina para la instrucción SUB Numero[SI], CL del i8086 sabiendo que el código de operación de la instrucción SUB es 001010 y que Numero referencia la posición 1234h de memoria?

19. ¿Cuál es el código máquina para la instrucción del i8086 MOV AX, BX sabiendo que el código de operación de la instrucción ADD es 100011?

20. ¿Cuál es el código máquina para la instrucción del i8086 MOV CL, DH sabiendo que el código de operación de la instrucción MOV es 100011?

21. ¿Cuál es el código máquina para la instrucción SBB AL, [BX][SI] del i8086 sabiendo que el código de operación de la instrucción SBB es 000110?

22. ¿Cuál es el código máquina para la instrucción MOV AL, [BX][SI] del i8086 sabiendo que el código de operación de la instrucción MOV es 100011?

23. La instrucción 'MOV' es una instrucción que puede ocupar de 2 a 6 bytes dependiendo del tipo de transferencia que se realice. En la siguiente tabla se muestran 4 tipos de instrucciones MOV con la codificación asociada.

MOV	1er byte	2º byte	3er byte	4º byte	5º byte	6º byte
Inmediato a mem.	1100011w	mod 000 r/m	desp (L)	desp (H)	data	data si w=1
Registro a registro	100010dw	mod reg r/m				
Memoria a registro Registro a memoria	100010dw	mod reg r/m	desp (L) (si procede)	desp (H) (si procede)		
Inmediato a registro	1011w reg	data	data si w=1			

Indíquese qué conjunto de bytes está asociado a cada una de las instrucciones que se muestran en la siguiente tabla.

	Código
A	B8h 34h 12h
B	8Bh F7h
C	C6h 06h 0Bh 00h 66h
D	8Bh FEh
E	8Ah 16h 08h 00h

	Instrucción
1	mov SI, DI
2	mov ax,1234h
3	mov DI,SI
4	mov var2,66h
5	mov dl,var1

24. Partiendo de los datos siguientes:

Dirección	Contenido
...	...
3FAF8h	34h
3FAF9h	01h
3FAFAh	00h
3FAFBh	72h
3FAFCh	69h
3FAFDh	3Fh
3FAFEh	01h
3FAFFh	02h
3FB00h	96h
3FB01h	44h
...	...

DS = 6543h
SS = 3FA0h
SP = 00FAh

Se pide

a) La dirección del mapa de memoria a la que se saltaría si se ejecutase la instrucción IRET.

b) ¿Qué ocurriría a continuación si a partir de esa posición de memoria, se encontrase el siguiente fragmento de código?:

```
        jc fin
        mov ah,02h
        mov dl,"M"
        int 21h
        jmp sigue
fin:    mov ah,4ch
        int 21h
        end inicio
```

25. Sea el código:

```
                dosseg
                .model small
                .stack 100h
                .data
LeeCadena       equ 0Ah
EscribeCadena equ 09h
Fin             equ 4ch
cad             db "Esta es la cadena a copiar$"
cad1            db 5, 0, 0, 0, 0, 0, 0
                .code
B8              mov ax, @data
8E D8           mov ds, ax

8D 16 001B      lea dx, cad1
B4 0A           mov ah, LeeCadena
CD 21           int 21h

8D 16 0000      lea dx, cad
B4 09           mov ah, EscribeCadena
CD 21           int 21h

33C9            ???????
B4 4C           mov ah, Fin
CD 21           int 21h
                End
```

AX = 0000
BX = 0000
CX = 0000
DX = 0000
SP = 0100
BP = 0000
SI = 0000
DI = 0000
DS = 3F2F
ES = 3F2F
SS = 3F44
CS = 3F3F
IP = 0010

Se pide:

a) Dirección efectiva y física de las variables cad y cad1 a partir de los datos de la tabla.

b) Dirección de comienzo de la instrucción lea dx, cad1 según los datos de la tabla que corrresponden en el momento en el que se ha ejecutado mov ax, @data.

c) ¿Qué instrucción equivale al código máquina 33 C9, si se sabe que el formato de la instrucción XOR es:

Primer byte	Segundo byte
0011 00dw	mod reg r/m

d) ¿Cuál es la dirección de comienzo de la última instrucción del programa?

Solución ejercicio 1

Apartado a)

El contador de programa (IP) determina el desplazamiento de la instrucción con respecto al segmento de código.

Las instrucciones de ensamblador no pueden solaparse, con lo que si la instrucción ocupa 3 bytes, la siguiente instrucción comenzará 3 bytes después de la que se está ejecutando.

Todos estos desplazamientos hacen referencia al segmento de código.

Por tanto, la dirección efectiva será:

CS x 10h + Desplazamiento = 51D7h x 10 + (6 + 3) = 51D79h

Apartado b)

El desplazamiento vendrá dado por la dirección en la que se encuentra la variable, 000Ah

Al tratarse de un dato que se encuentra en una posición de memoria, se tendrá que hacer referencia al segmento de datos DS.

Por lo tanto, la dirección efectiva será:

DS x 10h + Desplazamiento = 51C7h x 10 + 000Ah = 51C7Ah

Solución ejercicio 3

Apartado a)

.

CS x 10h + Desplazamiento = 51D7h x 10 + (15h + 2) = 51D87h

Apartado b)

Al tratarse de un direccionamiento directo a registro, no se accede a memoria con lo que no se debe calcular una dirección efectiva de memoria

Solución ejercicio 5

Apartado a)

CS x 10h + Desplazamiento = 5180h x 10 + (4 + 2) = 51806h

Apartado b)

Al tratarse de un direccionamiento implícito, no se accede a memoria con lo que no se debe calcular una dirección efectiva de memoria

Solución ejercicio 7

Apartado a)

CS x 10h + Desplazamiento = 8000h x 10 + (9 + 2) = 8000Bh

Apartado b)

DS x 10h + Desplazamiento = 9250h x 10 + 1500h = 93A00h

Solución ejercicio 9

Apartado a)

CS x 10h + Desplazamiento = 9000h x 10 + (9 + 2) = 9000Ah

Apartado b)

DS x 10h + Desplazamiento = 9250h x 10 + (1800h + FFFFh)

= A3CFFh

Solución ejercicio 11

Apartado a)

CS x 10h + Desplazamiento = A000h x 10 + (9 + 3) = A000Ch

Apartado b)

DS x 10h + Desplazamiento = A250h x 10 + (5000h + 2500h)

= A9A00h

Solución ejercicio 13

En el campo REG se codifica el registro de la instrucción (el primero si hay dos). El registro que empleamos es el AL que es de 8 bits. Con lo que REG valdrá 000

Por último, el segundo operando es un acceso a memoria y R/M = 111

De esta forma, el código máquina de la instrucción ADD AL, [BX[será:

Primer byte								**Segundo byte**							
0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1
Código Operación						D	W	MOD		REG			R/M		

O lo que es lo mismo el 0207h

Solución ejercicio 15

Se tienen que ir rellenado los diferentes campos para el formato de la instrucción.

El campo código valdrá 001010

El bit D especifica si el registro codificado en el campo REG es el origen o el destino del resultado. Como el resultado de la suma es el registro AL, será un 1

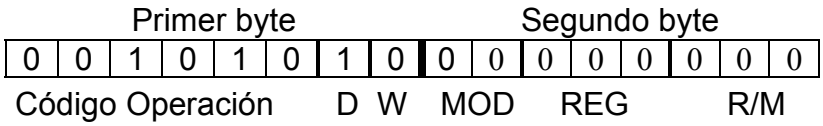
El bit W indica si los operandos son de 8 o de 16 bits. Como AL es un registro de 8 bits será un 0.

El campo MOD indica el modo de direccionamiento de la instrucción. Será un direccionamiento a memoria sin desplazamiento. Con lo que valdrá 00

En el campo REG se codifica el registro de la instrucción (el primero si hay dos). El registro que empleamos es el AL que es de 8 bits. Con lo que REG valdrá 000

Por último, el segundo operando es un acceso a memoria y R/M = 000

De esta forma, el código máquina de la instrucción SUB AL, [BX[[SI] será:



O lo que es lo mismo el 2A00h

Solución ejercicio 17

Se tienen que ir rellenado los diferentes campos para el formato de la instrucción.

El campo código valdrá 000000

El bit D especifica si el registro codificado en el campo REG es el origen o el destino del resultado. Como el origen de la suma es el registro BL, D = 0

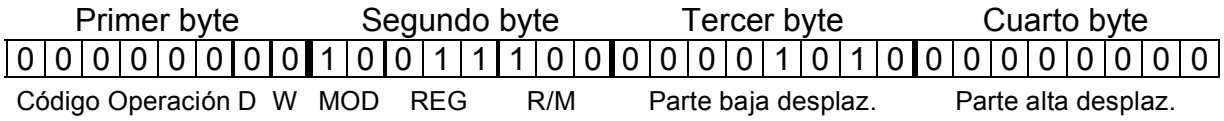
El bit W indica si los operandos son de 8 o de 16 bits. Como BL es un registro de 8 bits será un 0.

El campo MOD indica el modo de direccionamiento de la instrucción. Será un direccionamiento a memoria con desplazamiento de 16 bits. Con lo que valdrá 10

En el campo REG se codifica el registro de la instrucción (el primero si hay dos). El registro que empleamos es el BL que es de 8 bits. Con lo que REG valdrá 011

Por último, el segundo operando es un acceso a memoria y valdrá R/M 100

De esta forma, el código máquina de la instrucción ADD Numero[SI], BL será:



O lo que es lo mismo el 0094 C0A00h

Solución ejercicio 19

Se tienen que ir rellenado los diferentes campos para el formato de la instrucción.

El campo código valdrá 100011

El bit D especifica si el registro codificado en el campo REG es el origen o el destino del resultado. Como el destino de la operación es el registro AX, será un 1

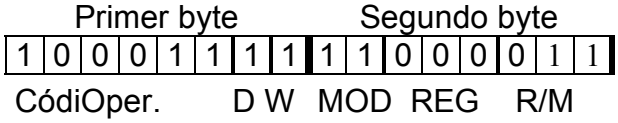
El bit W indica si los operandos son de 8 o de 16 bits. Como AX es un registro de 8 bits será un 1.

El campo MOD indica el modo de direccionamiento de la instrucción. Será un direccionamiento a registro). Con lo que valdrá 11

En el campo REG se codifica el registro de la instrucción (el primero si hay dos). El registro que empleamos es el AX que es de 16 bits. Con lo que REG valdrá 000

Por último, el segundo operando es un direccioamiento a registro con lo que R/M valdrá 011

De esta forma, el código máquina de la instrucción MOV AX, BX será:



O lo que es lo mismo el 8FC3h

Solución ejercicio 21

Se tienen que ir rellenado los diferentes campos para el formato de la instrucción.

El campo código valdrá 000110

El bit D especifica si el registro codificado en el campo REG es el origen o el destino del resultado. Como el resultado de la suma es el registro AL, será un 1

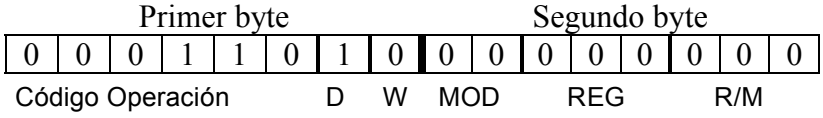
El bit W indica si los operandos son de 8 o de 16 bits. Como AL es un registro de 8 bits será un 0.

El campo MOD indica el modo de direccionamiento de la instrucción. Será un direccionamiento a memoria sin desplazamiento. Con lo que valdrá 00

En el campo REG se codifica el registro de la instrucción (el primero si hay dos). El registro que empleamos es el AL que es de 8 bits. Con lo que REG valdrá 000

Por último, el segundo operando es un acceso a memoria y valdrá R/M 000

De esta forma, el código máquina de la instrucción SUB AL, [BX[[SI] será:



O lo que es lo mismo el 1A00h

Solución ejercicio 23

Instrucción	Código
1	D
2	A
3	B
4	C
5	E

mov si,di

Registro a registro	100010dw	mod reg r/m				
	8Bh	F7h				

d=1 (campo reg = destino)
w =1 (2 bytes)
mod = 11
reg = 110 (SI)
r/m = 111 (DI)

mov ax,1234h

Inmto. A registro	1011w reg	data	data si w=1			
	B8h	34h	12h			

w=1 (2 bytes)
reg = 000 (AX)
data = 34h (parte baja)
data = 12h (parte alta)

mov di,si

Registro a registro	100010dw	mod reg r/m				
	8Bh	FEh				

d=1 (campo reg = destino)
w =1 (2 bytes)
mod = 11
reg = 111 (DI)
r/m = 110 (SI)

mov var2,66h

Inmediato a mem.	1100011w	mod 000 r/m	Desp (L)	Desp (H)	data	data si w=1
	C6h	06h	0Bh	00h	66h	

w=0 (byte)
mod=00
r/m = 110
data = 66h
desp = 000Bh

mov dl, var1

Memoria a registro	100010dw	mod reg r/m	desp (L)	desp(H)		
	8Ah	16h	08h	00h		

d=1
w=0 (byte)
mod = 00
reg = 010
r/m = 110
desp = 0008h

Solución ejercicio 25

Apartado a)

Nombre de la variable	Dirección efectiva	Dirección física
cad	0	3F2F0h
cad1	26	3F30Ah

Apartado b)

Dirección de comienzo: CS: IP+2 ⇨3F3Fh:0012h ⇨

Dirección física: 3F402h

Apartado c)

XOR CX,CX

Apartado d)

Dirección de comienzo: CS: IP+22⇨3F3Fh:0032h ⇨

Dirección física: 3F422h