TEMA 1

1.− Qué es el ensamblador:

El sistema alfanumérico para escribir código máquina mediante expresiones abreviadas (mnemotécnicos).

La compilación es más complicada porque incluye la conversión de operaciones matemáticas complejas, comandos de lenguaje natural o tipos de comandos complejos.

Cada ordenador tiene su propio lenguaje ensamblador, exclusivo de su CPU; un lenguaje de alto nivel (LAN) puede ser compilado en distintas máquinas.

2.− Para qué se usa:

Porque hay aplicaciones o programas que deben tratar directamente con los registros de la máquina, la memoria, dispositivos de E/S, etc.

Un programa ensamblador "bien hecho" produce un ejecutable más rápido y corto.

El proceso de traducción se realiza en dos pasos:

* *Primero:* se recorre el programa fuente; por cada instrucción implementa el contador según el código de la instrucción. Comprueba si tiene o no etiqueta, y si la tiene coloca su símbolo y su dirección en la tabla de símbolos. Después compara el símbolo del código de operación con una tabla de símbolos posibles; si es válido sustituye el código real y si no emite un mensaje real; a continuación comprueba la síntasis.

* *Segundo*: recorre las instrucciones del módulo fuente reemplazando los símbolos por sus direcciones reales tomadas de la tabla.

3.− Herramientas de programación:

Editor: programa con el que construímos los módulos fuente.

Ensamblador: convierte el código fuente en el código objeto.

Módulo LST: código máquina asociado a cada instrucción; la posición en la que se encuentra y una tabla en la que se indica la dirección que corresponde a cada nombre simbólico.

Módulo OBJ: módulo objeto.

Módulo CRF: listado de referencias cruzadas; contiene información referente a cada símbolo y las sentencias donde se hace referencia al mismo.

Montador: realiza dos tareas: combina varios módulos objetos realizando las conexiones necesarias entre ellos, y convierte los módulos objeto en ejecutable.

4.− Sintaxis de un módulo fuente ensamblador:

* *Instrucciones*: son representaciones simbólicas del juego de instrucciones de la CPU.

[etiqueta] nombre_instruccion [operando(1)][comentario]

La instrucción se especifica en una sola línea y los campos se separan entre sí por blancos o tabuladores.

* *Etiqueta:* identificador simbólico que se da a la instrucción. Puede tener hasta 31 caracteres; el primero no numérico; indiferente usar mayúsculas o minúsculas. El ensamblador interpreta las etiquetas como direcciones de memoria. El último carácter es ":".

* *Nombre_instrucción:* de dos a seis letras y una instrucción se transformará en una única instrucción de código máquina.

* *Operando(s)*: especifican los datos que serán tratados por la instrucción. Puede haber 0, 1 ó 2 operando. Si tenemos dos, el primero el "destino" y el segundo "fuente". Se separán por una coma. Existen tres tipos: inmediatos, registro y memoria; y además se pueden modificar los operandos de memoria con los prefijos de segmento.

* *Comentario:* cualquier cosa que comience por ";".

* *Directivas(pseudoinstrucciones)*: son partes del fichero fuente que indican al ensamblador cómo interpretar instrucciones o datos; sólo se utilizan en tiempo de ensamblaje; no se traducen a código máquina.

[nombre] nombre_directiva [operandos] [comentario]

***** El nombre no termina en ":" e indica nombres de procedimientos y variables con las directivas apropiadas.

5.− Datos del programa:

El programa ensamblador traducirá todos los datos tratados a números binarios, pero por comodidad podemos expresar los datos en binario, decimal, hexadecimal, caracteres.

Los números son de cinco tipos:

* Binarios: sufijo "b" 1001b

* Decimales: sufijo "d" 9846d

* Hexadecimales: sufijo "h"; no puede empezar por letra 32Ah

* Octales: sufijo "o" o "q"

* Caracteres o strings: se deben escribir entre ' ' o " ".

Las directivas más utilizadas para definir datos son:

* DB: define bytes.

* DW: define word.

* DD: define dobles palabras.

* DQ: define cuádruples palabras.

* DT: define grupos de 10 bytes.

Un operador es un modificador que se utiliza en el campo de operandos para especificar un operando completo. Existen cinco tipos:

* Aritméticos: +, −, *, /, mod, shl, shr.

* Lógicos: and, or, xor y not.

* Relacionales: FFFF(cierto), 0000(falso), EQ (equivalente).

* Retorno de valores:

SEG: devuelve el valor del segmento.

OFFSET: devuelve el desplazamiento.

TYPE <variable>: nos devuelve el tipo de operando.

SIZE <variable>: devuelve el número de bytes reservados para dicha variable.

* Atributos: modifican los atributos de la expresión cualificada.

6.− Estructura del microprocesador:

6.1.− Estructura del 8086 o 8088

Constan de dos unidades interconectadas en el mismo chip de inicio. Una es BIU y la otra EU (unidad ejecución).

La BIU es la encargada de buscar instrucciones y acceder a datos del exterior.

La EU es la encargada de ejecutar las instrucciones realizando las operaciones necesarias para ello.

Cuando la BIU localiza en memoria un objeto de código máquina, lo coloca en una línea de espera llamada 'cola de instrucciones'. En el 8086 la cola tiene una longitud de seis octetos y al código en memoria se accede de dos en dos octetos. En el 8088 tiene cuatro octetos y se accede de octeto en octeto.

6.2.− Juegos de registros:

* De la unidad de ejecución:

Propósito general: registros de 16 bits que pueden subdividirse y direccionarse separadamente. Nos quedarían 8 registros de 8 bits cada uno:

AX: es el acumulador.

BX: base

CX: contador

DX: datos.

Otros 4 registros punteros y de índice: no pueden subdividirse.

SP: puntero de pila.

BP: puntero base.

SI, DI: indican los registros índice y destino.

* De indicadores: registros de 16 bits, contiene varios bits de estado del procesador.

ZF: indicador de 0.

SF: indicador de signo. Se pone a 1 si el signo es "−".

PF: indicador de paridad.

CF: indicador de acarreo.

AF: indicador auxiliar para aritmética decimal.

DF: indicador de dirección. Controla la dirección en las operaciones con strings, incrementado o decrementando los registros índices.

IF: indicador de interrupción. Indica si están permitidas o no las interrupciones de dispositivos externos.

TF: indicador de traza. Controla la operación modo paso a paso.

* Registro de la BIU: contiene 4 registros de segmento.

CS: segmento de código.

DS: segmento de datos.

SS: segmento de pila.

ES: segmento extra.

Un registro de instrucciones (IP) contiene la dirección lógica de la instrucción a ejecutar. Es decir, el desplazamiento dentro del segmento de código cuya dirección de comienzo está en CS.

6.3.− Uso de los registros

AX: se utiliza implícitamente en la * y la ":"; explícitamente en la E/S de palabras. También se puede usar como registro general.

AL: se usa igual que el AX pero al trabajar con bytes.

AH: se usa como destino implícito en "*" y ":" a nivel de bytes.

CX: controla bucles, iteración de movimiento de cadenas, desplazamientos y rotaciones.

DX: almacena datos de 16 bits y puede considerarse como una extensión de AX en "*" y ":" de 16 bits.

También se usa en E/S indirecta (dirección del puerto).

BX: registro base para los direccionamientos.

SI,DI, BP: partes de los modos de direccionamiento.

IP y SP: control del flujo del programa.

6.4.− Consecución de una dirección real en memoria:

El espacio de direccionamiento es 1 Mb lo que supondría tener direcciones de 20 bits. El esquema de segmentación nos permite acceder correctamente a 1 Mb completo. Funciona de la siguiente forma:

Dirección de comienzo de segmento y desplazamiento: conformaran cualquier dirección del 8086. La dirección de comienzo se almacena en uno de los cuatro registros de segmento (CS, SS, DS, ES). El desplazamiento se puede componer de varias partes.

Decalage: es un número fijo.

Una base: almacenada en registro base.

Un índice: almacenado en un registro índice.

DIRECCIÓN REAL = 10h * (@inicial del segmeto) + DESPLAZAMIENTO

El uso de diferentes registros de segmento significa que hay áreas de trabajo separadas del programa, la pila, los datos. Cada área de trabajo tiene 84 Kb como máximo. Como tenemos 4 registros de segmentación, el área de trabajo podrá llegar a 4 * 64 = 256 Kb.El programador puede determinar la posición de estos segmentos al principio del programa o mientras se ejecuta cambiando dinámicamente la dirección de estos segmentos.

7.− Instrucciones básicas del µ8086 o µ8088:

7.1.− De transferencias de datos:

- **MOV destino, fuente**: transfiere un byte o una palabra desde el operando fuente al operando destino. El destino puede ser un registro o un elemento de memoria. El operando fuente puede ser un registro, un elemento de memoria o un valor inmediato. Ambos operandos deben ser del mismo tipo.

No se pueden mover datos entre dos elementos de memoria.

MOV AX, PEPE Juan y Pepe son dos elementos de mem. luego no lo podemos

MOV JUAN, AX hacer directamente, así que utilizamos el auxiliar {AX}.

No se puede mover un valor inmediato a un registro de segmento:

MOV ES, 113 ES: registro de memoria

MOV AX,113 113: valor inmediato.

MOV ES, AX

No se puede utilizar CS como operando destino.

- **XCHG destino, fuente**: intercambia los contenidos de las palabras fuente y destino.
- **PUSH fuente**: introduce el dato en la pila. No se puede utilizar CS como operando fuente. Equivale a decrementar el puntero de la pila SP en 2(equivale a una palabra) y luego transferir la palabra del operando fuente a lo alto de la pila.

Queremos que la variable PEPE entre en la pila:

- PUSH (PEPE)
- SUB SP, 2

MOVE AX,PEPE

MOVE {SP}, AX

- **POP destino:** Extrae un dato de la pila y lo lleva al destino.

POP AX

MOV AX, {SP}

ADD SP,2

Queremos sacar de la pila la variable PEPE:

POP PEPE

MOVE AX,{SP}

ADD SP, 2

MOVE PEPE,AX

- **IN acumulador, puerto:** lee el contenido del puerto de E/S y lleva el dato al acumulador.
- **OUT puerto, acumulador:** enviará el dato al puerto de E/S
- **LEA registro, fuente:** carga la dirección efectiva de un dato en un registro. El operando fuente debe ser un operando de memoria y el destino es un registro de 16 bits.

LEA AX, XX {SI}

Si XX = 1234h

SI = 0006h

AX será igual a 123Ah

- **LDS registro, fuente:**transfiere un puntero de 32 bits (dirección completa compuesta por desplazamiento y segmento) correspondiente al segundo operando (debe ser un operando de memoria de doble palabra). El registro de segmento es el DS.
- **LES registro, fuente:**igual al anterior pero utiliza el segmento ES.

7.2.− De aritmética binaria a entera:

Las operaciones en aritmética binaria a entera permiten a la CPU realizar cálculos con números enteros positivos y negativos con una representación en complemento a 2.

- **NEG operando:** cambia el signo del operando. Equivaldría al NOT del número y le sumaría 1.
- **ADD destino, fuente:** destino = destino + fuente.
- **ADC destino, fuente:** destino = destino + fuente + carry (acarreo).
- **SUB destino, fuente:** destino = destino − fuente.
- **SBB destino, fuente:** destino = destino − (fuente + acarreo).
- **MUL operando:** multiplica sin considerar el signo. Multiplica el acumulador {AL} o {AX}por el operando fuente. Si el operando fuente es de tipo byte, el resultado se almacena en AX y si es de tipo palabra el resultado se almacena en AX la parte inferior y en DX la palabra superior.

Si tipo fuente = byte:

AX = AL * fuente (multiplicación sin signo)

Si tipo fuente = palabra:

DX, AX = AX * fuente (multiplicación sin signo)

Si mitad superior (CF: acarreo) del resultado = 0

En CC CF = 1

- **IMUL operando:** multiplica considerando el signo.
- **DIV operando:** divide sin considerar el signo, un número contenido en el acumulador entre el operando fuente. El cociente se almacena en el acumulador. El resto se almacena en la extensión del acumulador. Si la extensión de AX será DX (que ocurrirá cuando sea de tipo palabra), la operación y la extensión de AL será AH.

AX

AX AL

DX

- **IDIV operando:** igual que el DIV pero considerando el signo.
- **CBW:** pasa de byte a palabra el contenido del acumulador.
- **CWD:** pasa de palabra a doble palabra el contenido del acumulador.
- **INC destino:** incrementa el destino.
- **DEC destino:** decrementa el destino.

7.3.− Operaciones lógicas:

Se usan para realizar operaciones a nivel de bits.

- **NOT operando:** cambia los bits 1 por 0 y viceversa y devuelve el resultado en el mismo operando.

AL = F2h AL 1111 0010

NOT AL; NOT AL 0000 1101 = Odh

- **OR destino, fuente:** operación *o lógico inclusivo*. El resultado se almacena en destino.

AX = FEDC h = 1111 1110 1101 1100

BX = 1234 h = 0001 0010 0011 0100

OR AX, BX 1111 1110 1111 1100 = FEFC h

- **AND destino, fuente:** la operación *Y lógica* entre 2 operandos, el resultado se deja en destino.

AX = FEDC h 1111 1110 1101 1100

BX = 1234 h 0001 0010 0011 0100

ADD AX, BX 0001 0010 0001 0100 = 1214 h

- **XOR destino, fuente:** la operación *o lógico exclusiva*; el resultado se deja en destino.

AX = FEDC h 1111 1110 1101 1100

BX = 1234 h 0001 0010 0011 0100

XOR AX, BX 1110 1100 1110 1000 = ECE8 h

7.4.− Operaciones de desplazamiento y rotaciones:

- **SAL destino, contador** y **SHL destino, contador:** realizan la misma operación y son lógicamente la misma instrucción. Desplaza a la izquierda los bits del operando destino, el número de bits indicado en el segundo operando. Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando. Desplazar a la izquierda una vez equivale a multiplicar por dos.

MOV CL, 2 AL = 1100 1100 b CF = 1

SAL AL, CL AL = 0011 0000 b CF = 1

- **SAR destino, contador:** desplazamiento aritmético a la derecha. Desplaza a la derecha los bits del operando destino, el número de bits especificados por el segundo operando. Los bits de la izquierda se rellenan con el bit del signo del primer operando. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando. Desplazar a la derecha una vez es como dividir por 2.

MOV CL, 2 AL = 1100 1100 b CF = 0

SAR AL, CL AL = 1111 0011 b CF = 0

- **SHR destino, contador:** desplazamiento lógico a la derecha. Desplaza a la derecha los bits del segundo operando destino, el número de bits que especifica en el 2º operando. Se rellena con 0.

MOV CL, 2 AL = 0011 0011 b CF = 0

SHR AL, CL AL = 0000 1100 b CF = 1

- **ROL destino, contador:** rotar a la izquierda los bits del operando destino el número de bits como indique el segundo operando. Si el número de bits a desplazar es uno se puede especificar directamente y si es mayor debe cargarse en CL.

MOV CL,2 AL = 1100 1100 CF= 0

ROL AL, CL AL = 1001 1001 CF = 1

AL = 0011 0011 CF= 1

- **ROR destino, contador:** rotar a la derecha los bits del operando destino el número de bits como indique el segundo operando. Si el número de bits a desplazar es uno se puede especificar directamente y si es mayor debe cargarse en CL.
- **RCL destino, contador:** rotar a la izquierda a través del acarreo. Lo de desplazar igual que lo anterior.
- **RCR destino, contador:** rota a la derecha a través de acarreo. Lo de desplazar es igual a todo lo anterior.

8.− Función de gestión de cadena:

Una cadena es una serie de bytes o palabras de hasta 64 Kb. Por defecto estas instrucciones suponen que por un lado de la cadena fuente viene del segmento de datos(DS) y desplazamiento (SI) y la cadena destino viene del segmento extra (ES) y desplazamiento (DI).

Los registros SI y DI se actualizan automáticamente después de cada operación. Si él o los operandos son de tipo byte, el incremento es a 1 y si son de tipo palabra el incremento es 2. A su vez este incremento puede ser positivo o negativo según el estado de la bandera de dirección (DF). Si DF está a 0 el incremento es positivo y si está a 1 el incremento es negativo.

- **MOVS cadenadestino, cadenafuente:** transfiere un byte o una palabra de la cadena fuente (direccionada por SI) en el segmento de datos a la cadena destino (direccionada por DI) en el segmento extra. Actualiza SI y DI para que apunten al siguiente elemento de la cadena.

- (MOVSB) La cadena está compuesta por bytes. DS : {SI} a ES : {DI}
- (MOVSW) La cadena está compuesta por palabras. DS : {SI} a ES : {DI}

Los operandos especificados en MOVS los utiliza el ensamblador sólo para verificar el tipo (si es byte o es palabra) y para ver si se ha especificado un registro de segmento. MOVS se puede reasignar el elemento fuente, pero no el destino.

REP: repite la operación de cadena y hace que se repita un determinado número de veces que vendrá especificado en el registro contador.

- REPE: repite si igual
- REPNE: repite si no igual.
- REPZ: repite si cero.
- REPNZ: repite si no cero.

Mover 100 bytes o palabras desde FUENTE (en el segmento de datos) a DESTINO (en el ES).

CL D

LEA SI, FUENTE

LEA SI, DESTINO

MOV CX, 100

REP MOVS{DI}, {SI}

1