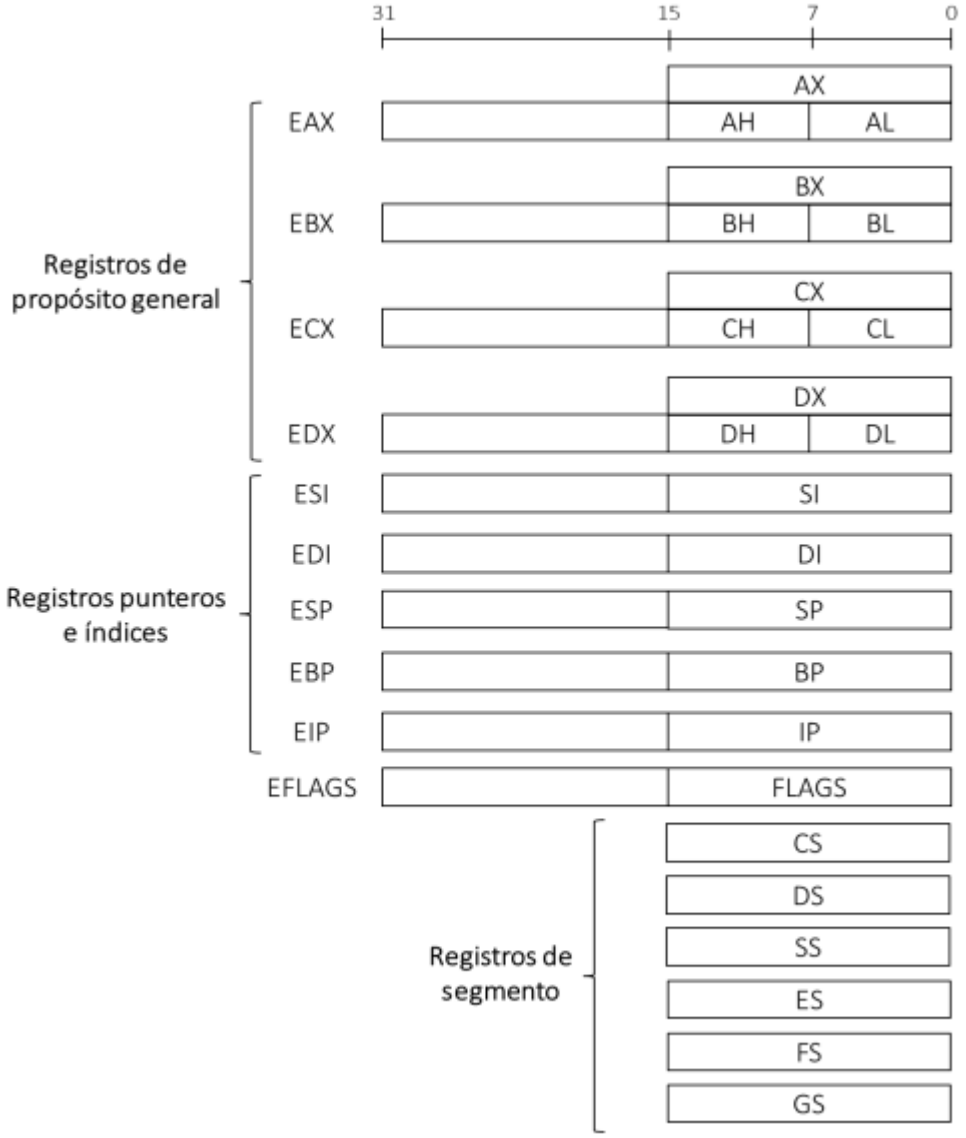
Unidad IV: Recursos del procesador

Conjunto de Registros

Registros del procesador

- Registros de propósito general: EAX, EBX, ECX, EDX.
- Registros punteros e índices: ESI, EDI, ESP, EBP, EIP.
- Registro de banderas: EFLAGS
- Registros de segmento: CS, DS, SS, ES, FS, GS.



Contador de Programa (PC)

- El registro más importante es el Contador de programa (**PC**), que apunta a la siguiente instrucción que se buscará para su ejecución.

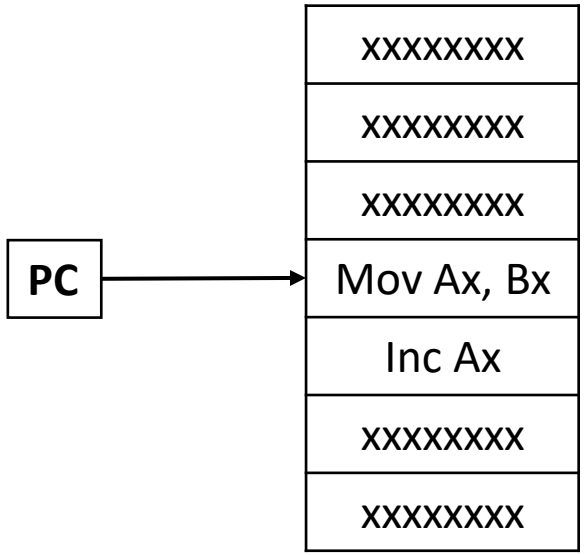


Figura 2: El contador de programa apunta a la siguiente instrucción a ejecutar. Una vez que el procesador trae esa instrucción, el PC se incrementa.

Registros de propósito general

- Todos son de 32 bits.
- Cada uno de estos registros contiene un registro de 16 bits en su parte menos significativa., el cual a su vez se divide en 2 registros de 8 bits.
- Son considerados de propósito general, pero tienen algunas funciones específicas.
- EAX: El registro de aritmética principal.
- EBX: Para almacenar punteros a memoria.
- ECX: Bucles
- EDX: Se utiliza en multiplicación y división.

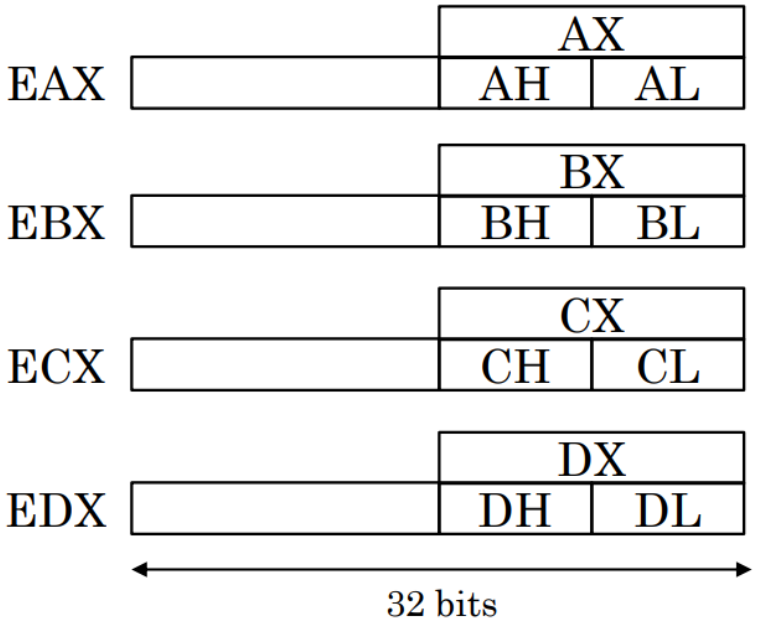


Figura 3: Representación de como los registros de 32 bits se parten en registros de 16 bits y 8 bits.

Registros punteros e índices

- Todos son de 32 bits.
- Contienen un registro de 16 bits en su parte menos significativa.
- ESI y EDI se utilizan para almacenar punteros a memoria, especialmente para operaciones con cadenas.
- EBP: Manipulación de la pila.
- ESP: Apuntador a la pila.
- EIP: Apuntador de instrucción.

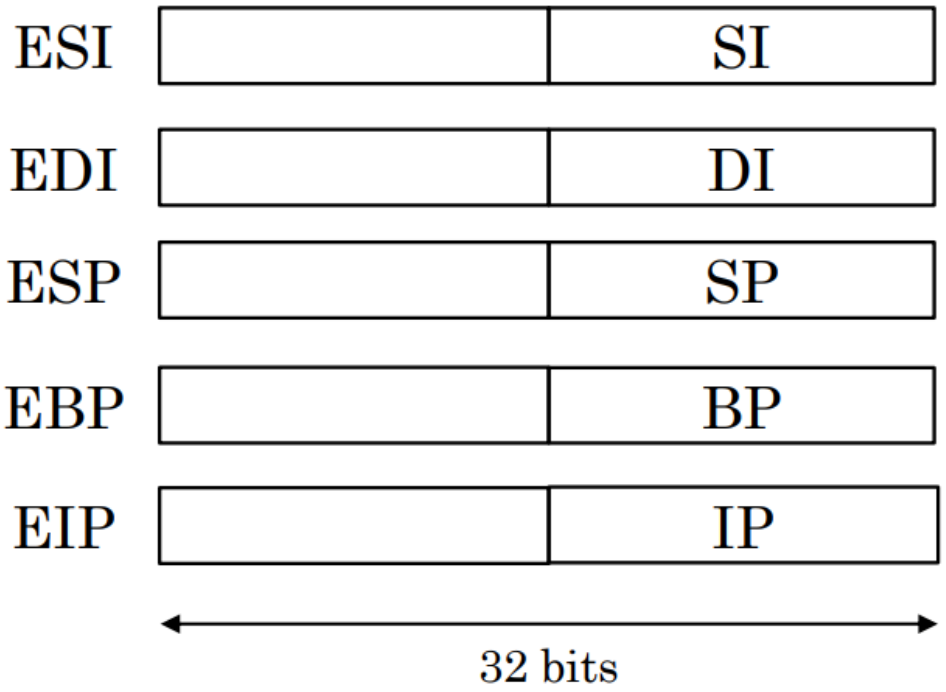


Figura 4: Los registros punteros e índices. Cada uno puede ser utilizado como un registro de 32 bits o de 16 bits.

Registros de segmento

- Todos son de 16 bits.
- No se pueden separar.
- CS: Segmento de código.
- DS: Segmento de datos.
- SS: Segmento de pila
- ES, FS y GS: Segmentos extra.

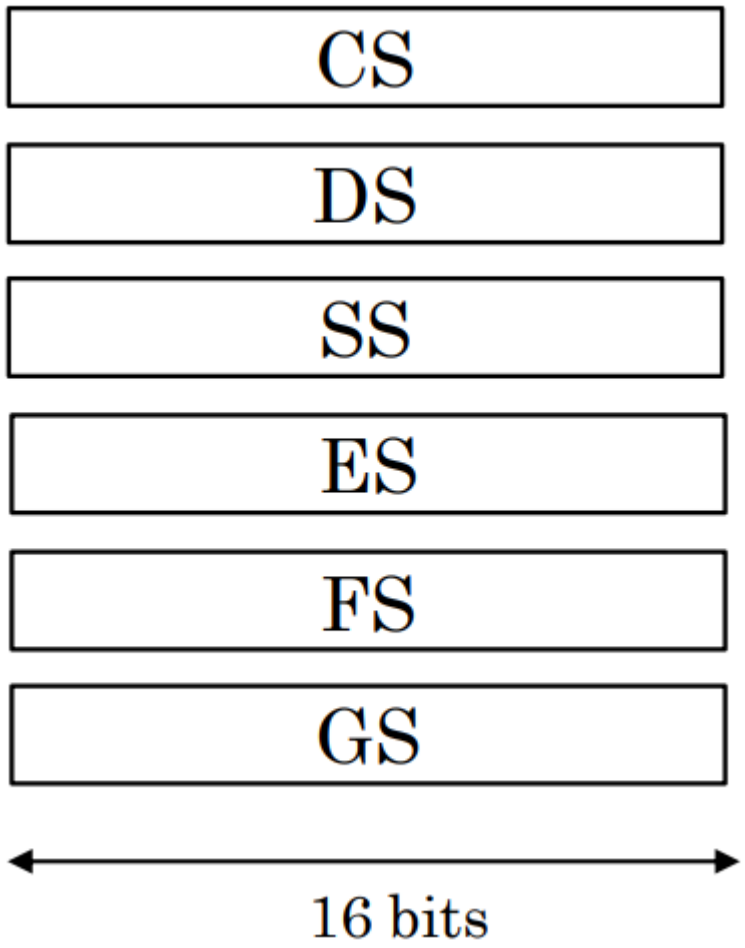


Figura 5: Registros de segmento de 16 bits.

Registro de banderas

- Registro de 32 bits.
- Bits 18 al 31 están reservados.
- Bits 0 al 11 bits de banderas.
- Indica la condición actual del procesador.
- Contiene información de la última operación aritmética.

EFLAGS

31-18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reservados	VM	RF		NT	IO PL	IO PL	O	D	I	T	S	Z		A		P		C

Registro de banderas

- C: Acarreo
- P: Paridad
- A: Acarreo auxiliar
- Z: Cero
- S: Signo
- O: Sobreflujo
- T: Bandera trampa
- I: Habilitar interrupciones
- D: Bandera de dirección

Modos de Direccionamiento

Modos de Direccionamiento

- Especifican la forma de interpretar la información contenida en el campo operando de la instrucción.

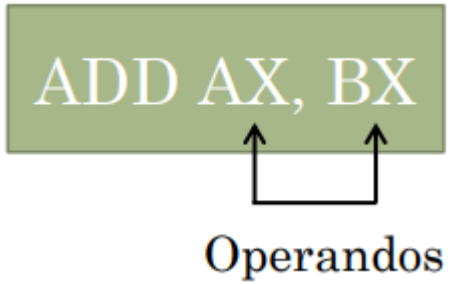


Figura 6: Instrucción ADD AX, BX.
AX y BX son nuestros operandos.

Modos de direccionamiento

Inmediato
Registro
Desplazamiento
Base
Base con desplazamiento
Base con índice y desplazamiento
Índice escalado y desplazamiento
Base con índice escalado y desplazamiento

Direccionamiento inmediato

- El operando está incluido en la instrucción.

```
MOV EAX, 0x12345678
MOV ECX, 0xF2
MOV SI, 0x35B2
MOV BH, 0x2C
```

Tabla con contenido en los registros

Registro	Contenido
EAX	12345678
EBX	xxxx2Cxx
ECX	000000F2
EDX	
ESI	xxxx35B2
EDI	
ESP	
EBP	

Restricciones

- No se permite asignar un valor inmediato a un registro de segmento.

INVALIDO:

MOV DS, 8B45

Direccionamiento de Registro

- El operando está en un registro.

```
MOV EBP, EAX
MOV BX, BP
MOV ESP, EAX
MOV DL, AH
```

Tabla con contenido en los registros

Registro	Contenido
EAX	E7027193
EBX	XXXX7193
ECX	
EDX	XXXXXX71
ESI	
EDI	
ESP	E7027193
EBP	E7027193

Restricciones

1. Los registros deben ser del mismo tamaño.

INVALIDO:

MOV EAX, BX

Restricciones

2. El programador no debe modificar el registro de segmento de código.

INVALIDO:

MOV CS, DX

Restricciones

3. No se transfiere de un registro de segmento a otro registro de segment.

INVALIDO:

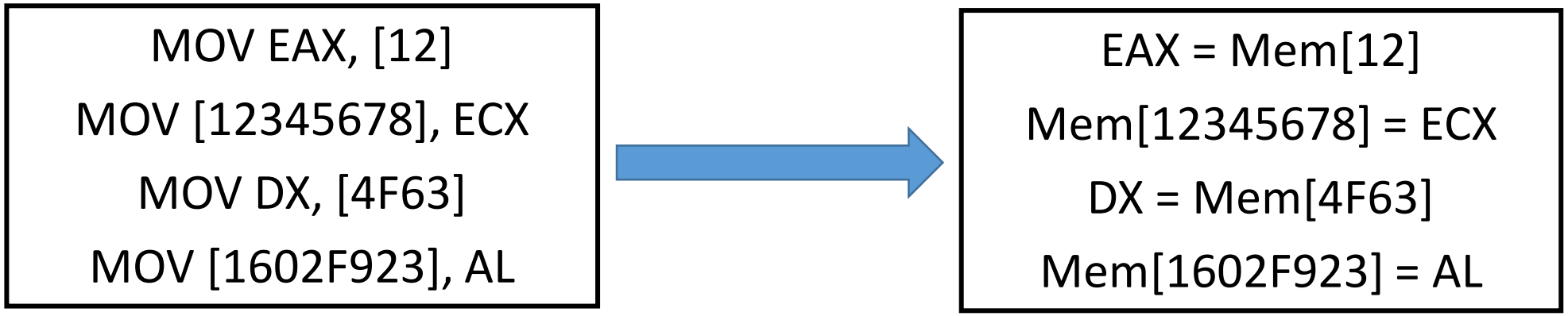
```
MOV DS, SS
```

VALIDO:

```
MOV AX, SS
MOV DS, AX
```

Direccionamiento Desplazamiento

- El operando está en memoria. La dirección de memoria del operando está incluida en la instrucción.



Direccionamiento Desplazamiento

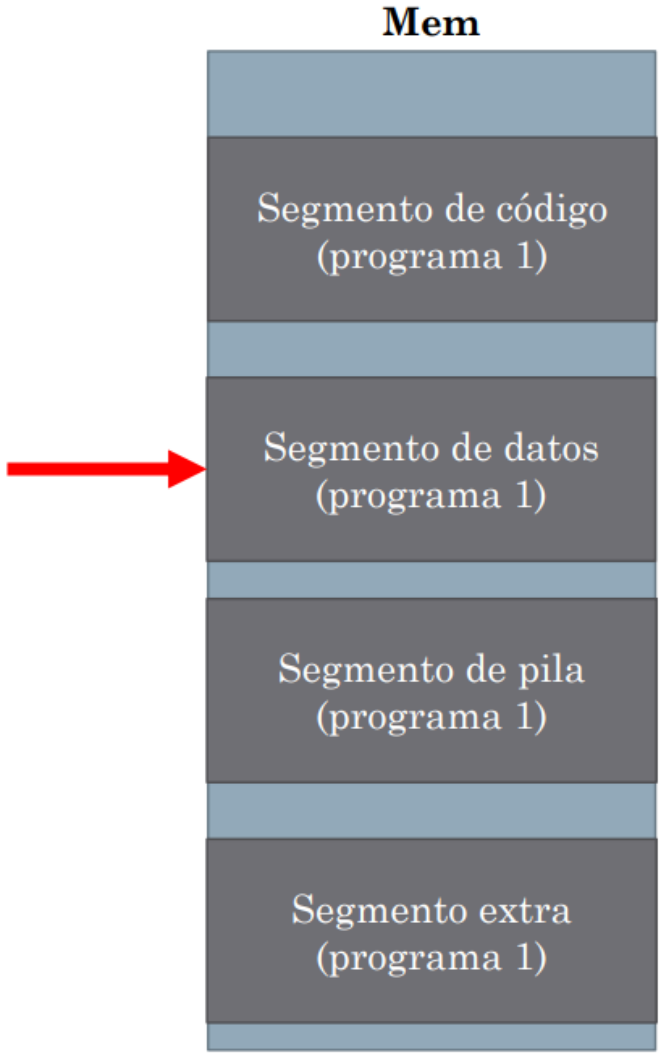


Figura 7: Representación de la memoria de un programa. La memoria se divide en 4 segmentos, Segmento de código, Segmento de datos, Segmento de pila, Segmento extra.

Direccionamiento Desplazamiento

MOV AL, [05]

AL = 1B

Contenidos de una memoria hipotética

Dirección	Contenido
0	12
1	34
2	56
3	78
4	90
5	1B
6	26
7	F5
8	98
9	00
A	59

Inicio del segmento

Direccionamiento Desplazamiento

ECX = 59B62011

MOV [07], CL

Se modifica la dirección a la que apunta [07] con el byte menos significativo de ECX.

Contenidos de una memoria hipotética

Dirección	Contenido
0	12
1	34
2	56
3	78
4	90
5	1B
6	26
7	11
8	98
9	00
A	59

Inicio del segmento

Direccionamiento Desplazamiento

MOV AX, [05]

¿AX = 261B

Ó

AX = 1B26?

Al ejecutar la instrucción MOV AX, [05] se deben traer 16 bits de memoria pues AX es de 16 bits. ¿Cómo se acomodan los datos traídos desde memoria?

Contenidos de una memoria hipotética

Dirección	Contenido
0	12
1	34
2	56
3	78
4	90
5	1B
6	26
7	11
8	98
9	00
A	59

Inicio del segmento

Endianness

- Se refiere al orden en que se almacenan los bytes que componen un dato.
- Se divide en 2 categorías:
  - Little Endian
  - Big Endian

Little endian

- El byte **menos** significativo del dato se almacena en la posición **menos** significativa.

EBX = 03509BF6

MOV [A503], BX

Segmento de datos

Dirección	Contenido
A503	F6
A504	9B

Al ejecutar la instrucción MOV [A503], BX se guarda en la dirección menos significativa el byte menos significativo de BX, y en la dirección más significativa el byte más significativo de BX.

Big endian

- El byte **más** significativo del dato se almacena en la posición **menos** significativa.

EBX = 03509BF6

MOV [A503], BX

Segmento de datos

Dirección	Contenido
A503	9B
A504	F6

Al ejecutar la instrucción MOV [A503], BX se guarda en la dirección menos significativa el byte más significativo de BX, y en la dirección más significativa el byte menos significativo de BX.

Comparación Endianness

EBX = 03509BF6

MOV [A503], EBX

Little Endian

Segmento de datos

A503	F6
A504	9B
A505	50
A506	03

Big Endian

Segmento de datos

A503	03
A504	50
A505	9B
A506	F6

Direccionamiento Desplazamiento

MOV AX, [05]

¿AX = 261B

Ó

AX = 1B26?

Al ejecutar la instrucción MOV AX, [05] se deben traer 16 bits de memoria pues AX es de 16 bits. El procesador i386 es Little Endian por lo que AX = 261B

Contenidos de una memoria hipotética

Dirección	Contenido
0	12
1	34
2	56
3	78
4	90
5	1B
6	26
7	11
8	98
9	00
A	59

Inicio del segmento

MOV [E05], 4F

¿Mem[E05] = 4F

Ó

Mem[E05] = 004F

Ó

Mem[05] = 0000004F?

MOV byte[E05], 4F

MOV word[E05], 4F

MOV dword[E05], 4F

Restricciones

No se transfiere de una dirección de memoria a otra dirección de memoria.

INVALIDO:

MOV [1234], [5678]

VALIDO:

MOV EAX, [5678]
MOV [1234], EAX

Direccionamiento Base

- El operando está en memoria. La dirección de memoria del operando está en un registro.

- Registros que operan sobre el segmento de datos: EAX, EBX, ECX, EDX, ESI, EDI.

- Registros que operan sobre el segmento de pila: ESP y EBP.

Direccionamiento Base

```
MOV EAX, [EBX]
MOV [EAX], EDX
MOV BX, [EBP]
MOV [ESP], AL
```

```
EAX = Mem[EBX]
Mem[EAX] = ECX
BX = Mem[EBP]
Mem[ESP] = AL
```

Direccionamiento Base con desplazamiento

- El operando está en memoria. La dirección de memoria del operando está en un registro más un desplazamiento.

```
MOV EAX, [EBX + E7027193]
MOV byte [EAX + 56D], F5
MOV [ESP – 5], AH
MOV EDX, [EBP + 9037AB]
```

```
EAX = Mem[EBX + E7027193]
byte Mem[EAX + 56D] = F5
Mem[ESP – 5] = AH
EDX = Mem[EBP + 9037AB]
```

Direccionamiento Base con índice

- El operando está en memoria. La dirección de memoria del operando está en un **registro base** (primer registro) más un **registro índice** (segundo registro).
- El **registro base** establece el segmento a usar.
- Los registros EAX, EBX, ECX, EDX, ESI, EDI, EBP pueden ser utilizados como bases o índices.
- El registro BP solo puede ser utilizado como base.

Direccionamiento Base con índice

```
MOV EAX, [ECX + EBP]
MOV word [EDI + ESI], F5
MOV [ESP + ECX], AH
MOV EDX, [EBP + EBX]
MOV EAX, [EBX + EBX]
```

```
EAX = Mem[ECX + EBP]
word Mem[EDI + ESI] = F5
Mem[ESP + ECX] = AH
EDX = Mem[EBP + EBX]
EAX = Mem[EBX + EBX]
```

Direccionamiento Base con índice y desplazamiento

MOV EAX, [ECX + EBP + 2F19]

MOV word [EDI + ESI + 9037B738], F5

MOV [ESP][ECX][4B024], AH

MOV EDX, [EBP + EBX - E02719]

EAX = Mem[ECX + EBP + 2F19]

word Mem[EDI + ESI + 9037B738] = F5

Mem[ESP + ECX + 4B024] = AH

EDX = Mem[EBP + EBX - E02719]

Direccionamiento Índice escalado [y desplazamiento]

- El operando está en memoria. La dirección de memoria del operando está en un registro índice escalado.
- Sintaxis:
  - [desplazamiento][índice*n]
- Donde n puede ser 1, 2, 4 u 8.

Direccionamiento Índice escalado

```
MOV EAX, [ECX*4]
MOV dword[EDI*8], 1B65
MOV [EBP*2], AH
```

```
EAX = Mem[ECX*4]
Mem[EDI*8] = 00001B65
Mem[EBP*2] = AH
```

Con desplazamiento

MOV EAX, [5][ECX*4]

MOV word[EDI*8 + 1F3], 1B

EAX = Mem[ECX*4 + 5]

Mem[EDI*8 + 1F3] = 001B

Direccionamiento Base con índice escalado [y desplazamiento]

- El operando está en memoria. La dirección de memoria del operando está en una base más un índice escalado.
- Sintaxis:
    - [desplazamiento][base][index*n]

Direccionamiento Base con índice escalado

```
MOV EAX, [EBP + ECX*4]
MOV [ECX + EBP*4], AX
MOV [EDX*2 + EBP], CH
```

```
EAX = Mem[EBP + ECX*4]
Mem[ECX + EBP*4] = AX
Mem[EDX*2 + EBP] = CH
```

Con desplazamiento

MOV [38][ECX + EBP*4], AX

MOV [2A][EDX*2 + EBP], CH

Mem[ECX + EBP*4 + 38] = AX

Mem[EDX*2 + EBP + 2A] = CH