Registros (Assembler)

\*Los tipos de registro pueden ser de 16 bits o de 32 bits.

\*Existen distintos tipos de registros que veremos a continuación:

🡪***Registros de propósito general***:

* Pueden almacenar datos o instrucciones (en el caso de las instrucciones, el registro contiene un desplazamiento que apunta a una dirección de memoria).

🡺 **AX**: acumulador de 16 bits. 🡺 **EAX**: acumulador de 32 bits.

🡺 **BX**: base de 16 bits. 🡺 **EBX**: base de 32 bits.

🡺 **CX**: contador de 16 bits. 🡺 **ECX**: contador de 32 bits.

🡺 **DX**: reg. de datos de 16 bits. 🡺 **EDX**: reg. de datos de 32 bits.

🡺 **SP**: puntero de pila (16 bits). 🡺 **ESP**: puntero de pila (32 bits).

🡺 **BP**: base de pila (16 bits). 🡺 **EBP**: base de pila (32 bits).

🡺 **SI**: índice fuente (16 bits). 🡺 **ESI**: índice fuente (32 bits).

🡺 **DI**: índice destino (16 bits). 🡺 **EDI**: índice destino (32 bits).

🡺 **AL, BL, CL o DL**: byte de menor peso del registro indicado.

🡺 **AH, BH, CH o DH**: byte de mayor peso del registro indicado.

* El registro puntero de pila (SP o ESP) disminuye en 1 cuando un elemento es añadido a la pila para escribir el elemento en la cima de la pila. Cuando un elemento es quitado de la pila, entonces incrementa en 1.
* El registro base de pila (BP o EBP) es utilizado para acceder a estructuras de datos pasadas en la pila. Apunta a la base de la pila, pero cuando existen rutinas toma el papel del puntero de pila para que este no pierda su valor.

🡪***Registro puntero de instrucciones***:

* Almacena la dirección de la próxima instrucción a ejecutar.
* Es afectada por el control de transferencias de las instrucciones, las interrupciones y las excepciones.

🡺 **IP**: puntero de instrucciones (16 bits). 🡺 **EIP**: puntero de instrucciones (32 bits).

* En memoria segmentada, la dirección de la instrucción en curso puede hallarse sumando el desplazamiento (2 bytes de menos peso del EIP, que es el IP) a la base donde comienza el segmento de código (CS).

🡪 ***Registros de estado o señalizadores***:

* También es conocido como FLAGS o EFLAGS y abarca casi todas las operaciones controladas por la unidad aritmético-lógica (ALU) y algunos restantes que actúan como señalizadores del sistema.

🡺 **CF**: señalizador de acarreo en el MSB (1 si hay acarreo, 0 si no).

🡺 **PF**: señalizador de paridad impar (1 si hay bits impares, 0 si no lo hay).

🡺 **AF**: señalizador de acarreo auxiliar o intermedio (1 si hay acarreo en el bit tres, 0 si no lo hay).

🡺 **ZF**: señalizador de cero (1 si todos los bits son ceros, 0 si no).

🡺 **SF**: señalizador de signo (1 si el MSB es 1, 0 si es 0).

🡺 **TF**: excepción al terminar la ejecución de la instrucción (1 provoca una excepción, 0 no).

🡺 **IF**: señalizador de habilitación de interrupciones (1 atiende las interrupciones no mascarables y permite reconocer las peticiones de interrupción mascarables que provoca la INTR, 0 no atiende nada).

🡺 **DF**: señalizador de dirección de exploración de strings (1 decremento automático de ESI y EDI que direccionan la cadena, 0 incremento automático de ESI y EDI).

🡺 **OF**: señalizador de overflow (1 si hay overflow, 0 si no lo hay).

🡺 **IOPL**: señalizador de privilegio de entradas y salidas y también el nivel de privilegio para alterar a IF al cargar el registro a EFLAGS o FLAGS (11 pueden acceder todos, 10 nivel 2, 01 nivel 1, 00 solamente puede acceder el sistema operativo). Se emplea en modo protegido con 2 bits.

🡺 **NT**: tarea anidada (1 cuando luego de realizar una tarea hay que volver a la misma, 0 cuando esto no sucede).

🡺 **RF**: señalizador de reanudación (1 ignora el fallo y continua la ejecución de instrucciones, 0 no lo hace).

🡺 **VM**: modo virtual 86 (1 pasa de modo protegido a modo virtual 86, 0 no lo hace).

🡺 **AC**: bit de chequeo de alineamiento (1 produce una excepción para alinear una palabra, 0 no lo hace).

🡺 **VIP**: interrupción mascarable virtual pendiente (1 interrupción pendiente, 0 no la hay).

🡺 **VIF**: interrupción virtual (equivalente a IF pero trabajando en modo virtual 8086).

🡺 **ID**: bit de identificación (1 soporta la instrucción CPUID para identificar la CPU, 0 no la soporta).

🡪 ***Registros de segmento***:

* Existen 3 tipos de segmentos: segmentos de pila, de código y de datos.

🡺 CS: segmento de código (16 bits).

🡺 SS: segmento de pila (16 bits).

🡺 DS: segmento de datos (16 bits). 🡺 ES, FS, GS: segmentos extra de datos (32 bits).

* El segmento de código (CS) contiene el segmento de código que se está ejecutando en CPU. El desplazamiento que debe añadírsele reside en el IP o EIP.
* El segmento de pila (SS) contiene el valor del selector del segmento de pila en curso. El desplazamiento que debe añadirse a la base del SS para determinar la cima, donde se cargan y descargan los datos reside en el SP o ESP.
* El segmento de datos (DS) contiene el valor del selector del segmento de datos. El desplazamiento que debe añadirse viene especificado en el modo de direccionamiento usado en la instrucción para expresar operandos y el resultado.

Memoria/Segmentación en modo real

\*No posee tipo alguno de protección ni posibilidad de manejo de memoria virtual.

\*Un segmento queda definido por:

🡪 Base o dirección de comienzo (20 bits).

🡪 Desplazamiento de 16 bits (admite un tamaño máximo 64KBytes y la capacidad máxima de MP de 1 Mbyte).

\*Todo segmento (datos, pila o código) está especificado por una dirección lógica, compuesta de 2 campos de 16 bits cada uno:

🡪 Selector: hace referencia a la base del segmento, que se deduce a partir del valor contenido en el registro de segmento apropiado. Como el 8086 maneja una MP de 1 Mbyte se añaden 4 ceros a los 16 bits del registro de segmento.

🡪 Desplazamiento: en el caso del CS se almacena en el IP; en el caso del SS se almacena en SP; en el caso del DS lo expresa el modo de direccionamiento de los operandos o el resultado de la instrucción en curso.

INDICE

15 3 2 1 0

(REGISTRO SEGMENTO)

DESPLAZAMIENTO

TI

SELECTOR

31 0

RPL

|  |  |  |
| --- | --- | --- |
|  | BASE | DESPLAZAMIENTO |
| CÓDIGO | **CS x 16** | **IP** |
| PILA | **SS x 16** | **SP** |
| DATOS | **DS x 16**  **ES x 16**  **FS x 16**  **GS x 16** | **DESPLAZAMIENTO** |

\*En este modo se encuentran 8 registros de propósito general (AX, BX, CX, DX, SP, BP, SI y DI) de 16 y también de 32 bits; también se encuentran los 6 registros de segmento.

\*Solo puede utilizarse el método de segmentación, no paginación.

\*Las instrucciones específicas de protección son inútiles.

Memoria/Segmentación en modo protegido

\*Un segmento queda caracterizado por tres parámetros fundamentales, que forman un conjunto denominado descriptor de segmento con una longitud total de 64 bits:

🡪 Base: formada por 32 bits, que representan la longitud de la dirección de memoria física.

🡪Límite: formado por 20 bits que determinan el tamaño exacto del segmento usado por el programador.

🡪 Atributos o derechos de acceso: tienen un tamaño de 12 bits que proporcionan características relevantes del segmento (Tipo, nivel de privilegio, indicadores sobre aspectos relacionados con la gestión de memoria virtual, etc).

\*Para obtener el valor del descriptor de segmento se utiliza el registro de segmento para acceder a unas tablas que residen en MP.



\*El objetivo de este modo es proteger a las distintas tareas, cuando se opera en modo multitarea, de los posibles accesos incorrectos o inválidos.

Espacio virtual

\*Vamos a utilizar la dirección lógica de la misma forma que fue explicada en el modo real.

🡪 Selector (14 bits): Desplazamiento (32 bits)

\*Los 14 bits de mayor peso del registro de segmento determinan el selector de segmento en memoria virtual y sus 2 últimos bits indican el nivel de privilegio del segmento que solicita usar ese selector.

\*La MMU convierte la dirección virtual de 46 bits en una dirección física de 32 bits. Esta recoge la dirección virtual y la introduce en la Unidad de Segmentación. Esta última posee la tabla de segmentos, que determina y registra a los segmentos que están en memoria principal y la posición que estos ocupan. Sigue el siguiente mecanismo:

4 GB MAX.

MEMORIA

VIRTUAL

MEMORIA

FÍSICA

MECANISMO DE

DIRECCIONAMIENTO

(MMU)

DIRECCIÓN FÍSICA

DIRECCIÓN VIRTUAL

AUSENCIA

46

32

64 TB MAX

SISTEMA OPERATIVO

Espacio lineal

\*La dirección lineal coincide con la dirección física correspondiente a la MP, si la Unidad de Paginación no está activada.

\*Basándonos en cómo es organizado un segmento en modo protegido:

BASE

(31-24)

LÍMITE

(19-16)

BASE

(23-16)

LÍMITE (15-0)

BASE (15-0)

31 24 23 20 19 16 15 7 0

DIRECCIÓN n

DIRECCIÓN n+4

G D/B 0 AVL

23 20

P S A

0 7

DPL

TIPO

ATRIBUTOS

+4

+0

\*Bits que conforman el espacio de los atributos:

* **Bit de presencia (P):** indica si el segmento al que referencia el descriptor se halla en MP (1) o no (0).
* **Nivel de privilegio (DPL):** indica el nivel de privilegio del segmento referenciado por el descriptor. Consta de 2 bits ya que el valor varía entre 0 y 3 (binarios).
* **Tipo de segmento (S):** si su valor es 1 el tipo de segmento es normal (pila, datos, código); si su valor es 0 se refiere a un segmento del sistema, que referencia a un recurso especial del sistema.
* **Tipo:** es un campo de 3 bits que determina, en el caso de un segmento normal, si es de pila, de código o de datos. También determina el acceso permitido: lectura, escritura o pila.
* **Accedido (A):** se pone en 1 cada vez que el procesador accede al segmento.
* **Granularidad (G):** si se encuentra en 0, entonces el segmento estará expresado en bytes (tamaño máximo 1 MB); si se encuentra en uno estará paginado en páginas de 4 KB (tamaño máximo 4 GB).
* **Defecto/grande (D/B):** en los segmentos de código se utiliza el bit D y en los de datos el bit B. Si D/B = 1 entonces podrán tomarse direcciones efectivas y operandos de 32 bits; en el caso contrario solo de 16 bits.
* **Disponible (AVL):** bit a disposición del usuario para diferenciar a segmentos específicos.

\*El segmento de código tiene 3 tipos: (1) solo ejecutable, (2) ejecutable y leíble y (3) ajustable o conforming.

\*El segmento de datos tiene 3 tipos: (1) se puede leer y escribir, (2) solo puede leerse y (3) es un segmento de pila.

E

1 C R

0 ED W

DATOS

CÓDIGO

TIPO

* Ejecutable (E): si esta en 1 determina un segmento de código; en caso contrario a los segmentos de datos que no pueden ejecutarse.

E

1 C R

0 ED W

DATOS

CÓDIGO

TIPO

E

1 C R

0 ED W

DATOS

CÓDIGO

TIPO

\*En caso de que E=1, los otros 2 bits de este campo cumplen lo siguiente:

* Ajustable o Conforming (C): si su valor es 0 no cambia el nivel de privilegio; si su valor es 1 su nivel de privilegio pasa a ser el mismo que el del segmento que lo había pedido (pueden ejecutarse y compartirse programas con diferentes niveles de privilegio).
* Leíble (R): si su valor es 1 el segmento de código puede leerse.

\*En caso de que E=0, los otros 2 bits de este campo cumplen lo siguiente:

* Expansión decreciente (ED): si su valor es 0 el segmento de datos es normal; si su valor es 1 se trata de un segmento de pila.
* Escribible (W): si su valor es 1 el segmento de datos puede leerse y escribirse; si su valor es 0 solo puede leerse.