REPERTORIO DE INSTRUCCIONES 17

17.1- TIPO DE DATOS

El Pentium soporta todos los tipos de datos que manejan los lenguajes evolucionados, aunque hay cuatro fundamentales:

1º) *byte* u octeto de ocho bits.

2º) *Palabra,* compuesta por dos *bytes*.

3º) *Doble palabra* de cuatro *bytes*.

4º) *Cuádruple palabra* de ocho *bytes*.

Cuando la memoria está organizada con tamaño byte, tanto las palabras como las dobles palabras, ocupan dos o cuatro posiciones sucesivas, respectivamente. Sin embargo, la dirección de inicio de palabras y dobles palabras puede ser cualquiera y no se precisa (como sucede en otros procesadores) que estén *alineadas,* ocupando direcciones pares y múltiplos de cuatro.

Esta característica mejora la flexibilidad en la ubicación de datos y libera al programador de vigilar el direccionado, aunque se recomienda alinear dichos tipos de datos para que su acceso sea más rápido. Como se aprecia en la figura 17.1, los bytes de más peso ocupan las direcciones más altas en la memoria.

Además de los tres tipos de datos básicos, el Pentium admite los tipos que se muestran en la figura 17.2 y que se pueden clasificar, en función de las instrucciones que las manejan, de la siguiente forma:

1º) Enteros y ordinales

Son los números con y sin signo, respectivamente, que son reconocidos por todas las instrucciones aritméticas. Tienen tamaño byte, palabra y doble palabra.

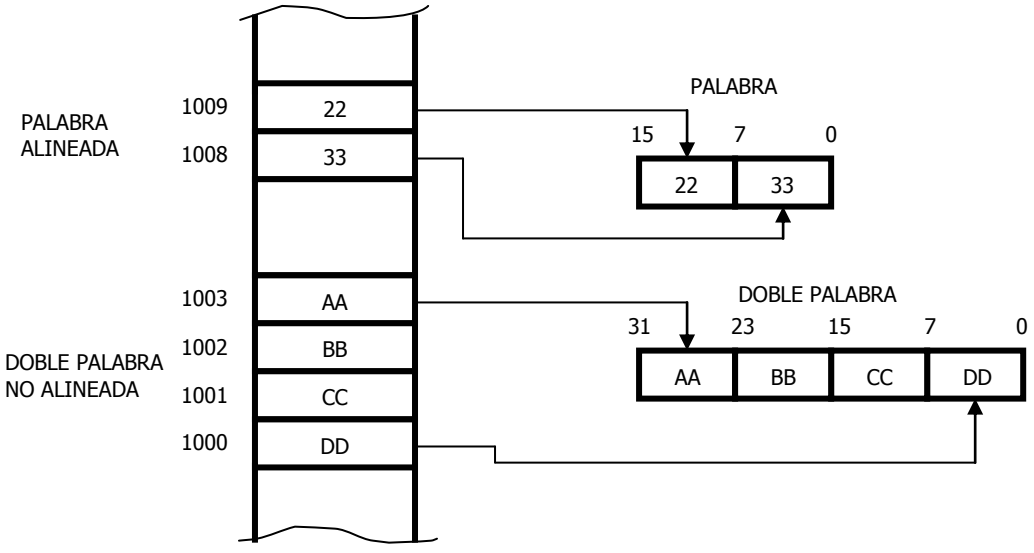


Figura 17.1. Distribución de palabras y doble palabras en una memoria organizada con tamaño byte.

2º) Números BCD (Decimal Codificado en Binario)

Es la representación de los números decimales mediante cuatro bits. Este tipo de datos lo soportan las instrucciones de ajuste, que complementan a las instrucciones aritméticas.

Los datos BCD pueden estar desempaquetados o empaquetados. En el primer caso, en cada byte hay un dígito BCD en los cuatro bits de menos peso, mientras que en los empaquetados se representan dos dígitos BCD en un byte.

3º) Cadenas

Hay instrucciones especializadas para manipular cadenas de bits, de bytes, de palabras y de dobles palabras. Las cadenas pueden alcanzar un tamaño de 4
GB.

4º) Campo de bits

Este formato permite a ciertas instrucciones manejar ciertos bits particulares en un campo, cuyo tamaño máximo puede alcanzar los 32 bits.

Las instrucciones capaces de trabajar con campos de bits y cadenas son muy interesantes en aplicaciones gráficas y de comunicaciones.

5º) Punteros de direcciones

Este tipo de datos está referenciado en los modos de direccionamiento de los operandos y en las instrucciones de salto.

El *puntero corto* o cercano, dispone de un desplazamiento de 32 bits que permite un salto dentro del segmento que se trabaja.

El *puntero largo* o lejano, además del desplazamiento, contiene el valor de un selector para uno de los seis registros de segmento. Se puede realizar un salto a otro segmento. El tamaño de este puntero es de 48 bits.

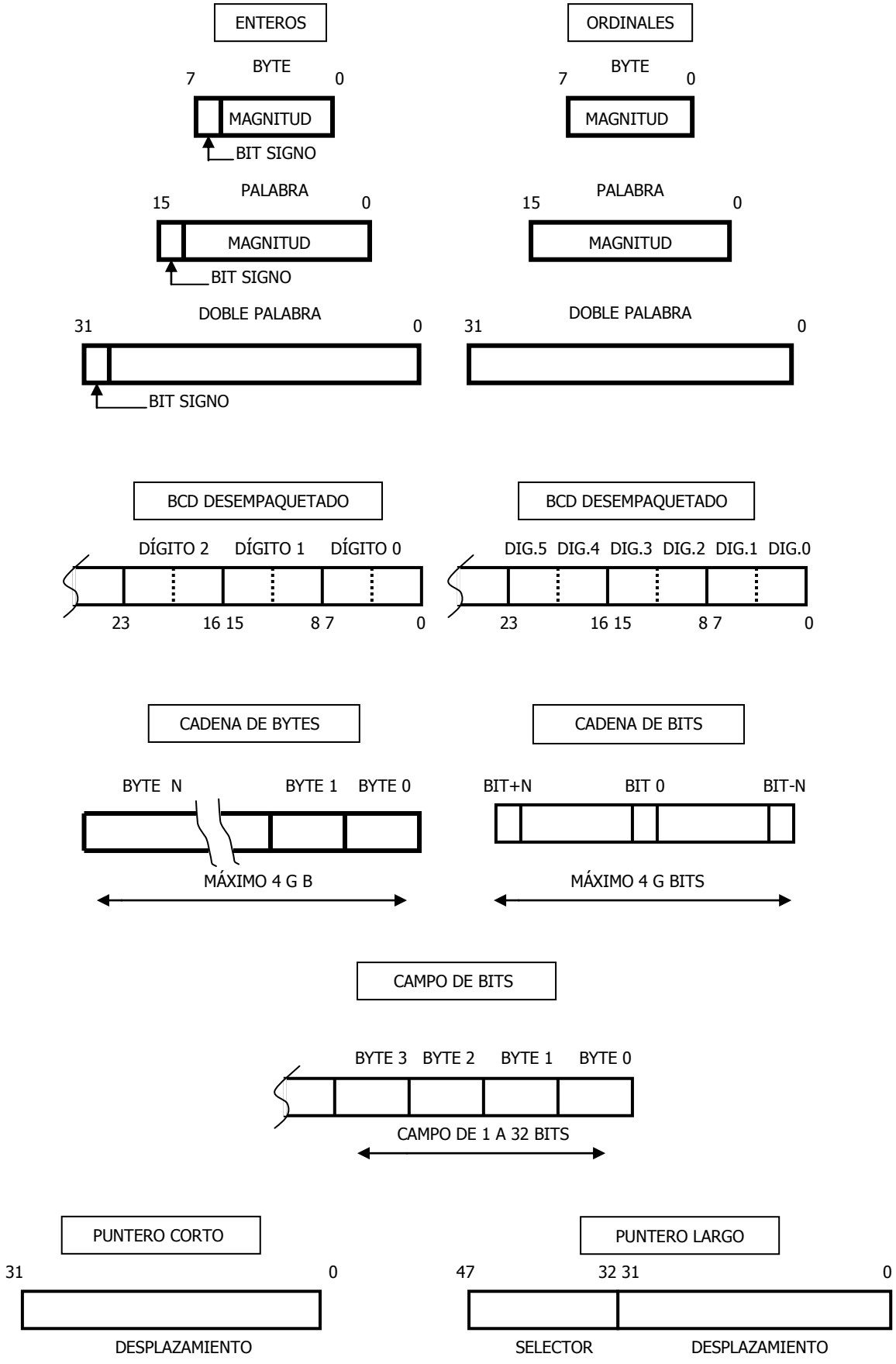


Figura 17.2. Tipos de datos que utiliza el Pentium.

17.1.1-TIPOS DE DATOS DEL COPROCESADOR MATEMÁTICO

El coprocesador matemático soporta siete tipos de datos diferentes, aunque internamente guarda toda la información en el formato de coma flotante y precisión extendida, es decir, en una representación que ocupa 80 bits, igual a la del tamaño de sus registros internos.

Como la memoria de datos tiene almacenados los siete tipos de datos en cualquier formato, son las instrucciones las que se encargan de hacer la conversión en los dos sentidos. Si una instrucción toma un dato de la memoria, lo convierte en formato de coma flotante con precisión extendida de 80 bits, al introducirlo dentro del coprocesador y viceversa.

Los siete tipos de datos se clasifican en tres grandes grupos:

1º. Enteros
En este grupo hay tres tipos: largo, corto y palabra, de 64, 32 y 16 bits, respectivamente.

2º. Decimal empaquetado
Guarda 18 dígitos BCD empaquetados, en formato de 80 bits.

3º.Coma flotante
En este apartado se distinguen los datos en simple precisión, en doble precisión y de precisión extendida, que es el formato con el que opera internamente el coprocesador.

En la figura 16.3 se muestran los sirte tipos de datos para el coprocesador matemático.

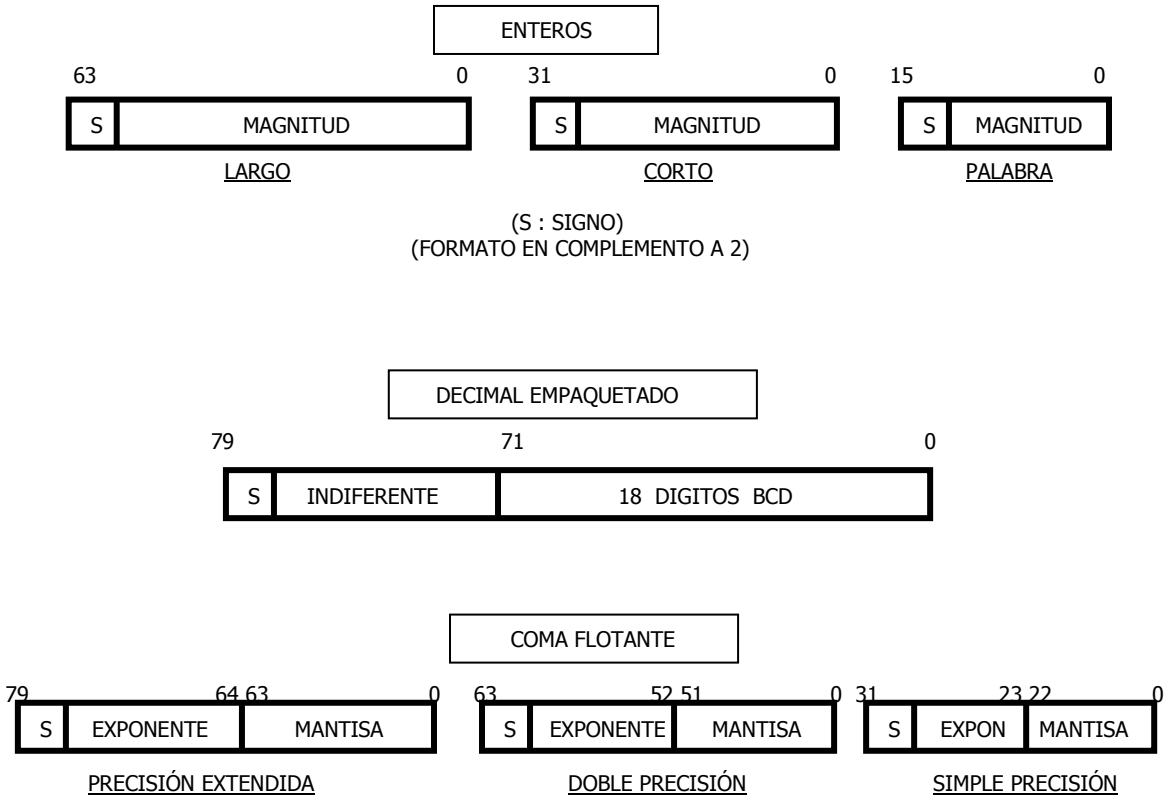


Figura 17.3. Descripción de los siete tipos de datos que admite el coprocesador matemático

17.2- MODOS DE DIRECCIONAMIENTO

Las formas que dispone el Pentium para referenciar los operandos de sus instrucciones, se pueden dividir en tres grandes categorías:

1º. Modo de direccionamiento inmediato

El operando reside en la propia instrucción.

Ejemplo

IMUL ECX, -4

Multiplica el entero contenido en el registro ECX por el operando inmediato, situado en la propia instrucción, que es -4.

2ª. Por registro

El operando reside en un registro interno del procesador.

Ejemplo

INC EBX

Incrementa el operando, que es el valor contenido en el registro EBX.

3ª.Modo de direccionado en memoria

El operando reside en la memoria y admite múltiples variantes para expresar la posición donde se encuentra.

El algoritmo fundamental que aplica el Pentium para calcular la dirección efectiva, cuando trabaja con operandos de 32 bits, es el siguiente:

Dirección = *Variable* + *Reg. Base* + *Reg. índice* X *Factor de Escala* + *Desplazamiento*

Los diferentes campos del algoritmo pueden tomar un valor de acuerdo con el registro interno de la CPU que tenga capacidad para almacenarlo, según el siguiente criterio:

Reg. Base = { EAX, EBX, ECX, EDX, ESP, EBP, ESI, EDI }

Reg. Índice = {EAX, EBX, ECX, EDX, EBP, ESI, EDI }

Factor de Escala = {1, 2, 4, 8 }

A)El primer término posible que admite el algoritmo se denomina *variable y* permite precisar un identificador o etiqueta que representa el desplazamiento del principio de una variable, como puede serlo una tabla.

B)El *registro Base* apunta al comienzo de una estructura de datos de dirección variable, por ejemplo, el comienzo de una matriz de registros, que no tiene una dirección fija en la memoria.

C)El *registro Índice* mueve el apuntador dentro de las estructuras de datos para acceder a los diferentes campos de que constan. El *factor de escala* tiene en cuenta el tamaño del elemento (octeto, palabra, etc.).

D)Finalmente, el *desplazamiento* de valor constante se usa para seleccionar un dato de dirección conocida, o bien, la base de una estructura de datos que responde a una dirección fija.

Este completo y potente sistema de direccionamiento, proporciona al Pentium un acceso cómodo a gran variedad de datos en la memoria, que son tan frecuentes en los lenguajes de alto nivel, como el C, o el FORTRAN.

Aunque se ha indicado que todos los registros generales pueden actuar como Base y como Índice, hay que matizar, que, como sucedía en el 8086, cada uno está dedicado preferentemente a ciertas misiones. Por ejemplo, el registro EAX es el más solicitado para la implementación de instrucciones de tipo aritmético. Los registros ESI y EDI se emplean comúnmente como registros Índice. También es habitual encontrarse con instrucciones que seleccionan implícitamente uno de los registros, como es el caso de las instrucciones con repetición de bucles (LOOP), en el que como registro contador se usa ECX. No obstante, el Pentium admite una eficaz flexibilidad de empleo de los registros.

Ejemplo:

ADD EAX, TABLA[EBP + ESP4 +8]

Se deposita en EAX el resultado de la suma del contenido de EAX con el dato de 32 bits ubicado en la posición de memoria de TABLA, a la que se suma EBP, ESI*4 y 8.

El algoritmo empleado para calcular la dirección efectiva cuando se trabaja con operandos de 16 bits, es el siguiente:

DIRECCION = VARIABLE + REG. BASE + REG. INDICE + DESPLAZAMIENTO

El registro Base puede ser *BX* o *BP* y el registro Índice el *SI* o el *DI.*

Con este modo de direccionamiento se trabaja sobre segmentos de 64 KB de tamaño máximo.

El manejo de las puertas de E/S se realiza mediante instrucciones especializadas, dedicadas a esta finalidad. En el Pentium se dedica un tamaño de 64 KB no segmentado al espacio de E/S.

Las instrucciones que introducen o sacan datos por las puertas de E/S, disponen de un operando que especifica la posición de éstas en el espacio de E/S, bien directamente, si el valor de la puerta es inferior a *255,* o bien colocando en DX su valor, cualquiera que sea el mismo.

17.3-CLASIFICACIÓN Y CARACTERÍSTICAS GENERALES DEL REPERTORIO DE INSTRUCCIONES

En este apartado se describen de forma muy sucinta todas las instrucciones del Pentium, clasificándolas según la función que realizan, para dar una idea de todas las existentes y su misión principal.

Para resaltar la aportación del Pentium, se separan las instrucciones que se han incorporado en este procesador de las convencionales conocidas en modelos precedentes. Junto con las nuevas instrucciones, otras ventajas inherentes al Pentium son:

La posibilidad de tratar operandos de 32 bits, también implica una mayor rapidez de ejecución y también los nuevos tipos de datos y posibilidad de combinar varios de ellos en ciertas instrucciones.

El siguiente capítulo está destinado a proporcionar un resumen detallado de todas las instrucciones del Pentium, ordenadas alfabéticamente e incluyendo todas las características de interés para el programador que las va a aplicar.

17.3.1-INSTRUCCIONES PRIVILEGIADAS

Cuando trabaja el Pentium en Modo Protegido se debe asegurar la protección de la memoria y de las tareas para hacer inaccesible el paso de una LDT a otra, así como la modificación no regulada de los registros del sistema (GDTR, LDTR, TR, IDTR, etc).

La mayoría de las instrucciones del Pentium se pueden ejecutar en cualquier nivel de privilegio, pero hay unas pocas que pueden comprometer la integridad del sistema cuyo uso está restringido. Se clasifican en dos grandes grupos:

INSTRUCCIONES PROTEGIDAS

Se pueden ejecutar únicamente desde niveles de privilegio que sean jerárquicamente superiores al campo de Nivel de privilegio de E/S (IOPL) del registro EFLAGS.

INSTRUCCIONES PRIVILEGIADAS

Sólo pueden ejecutarse en el máximo nivel de privilegio, o sea, con segmentos que tengan CPL = 0.

Dentro de las instrucciones privilegiadas, que sólo se usan en el 5.0. y nunca en las aplicaciones, se distinguen cuatro grupos:

1º)Cualquier instrucción que pueda modificar el campo IOPL, como IRET, POPF y las relacionadas con la conmutación de tarea (TS).

2º)Instrucciones que afectan a los registros que hacen referencia a tablas que controlan el sistema en Modo Protegido.

3º) Instrucciones que afecten a la MSW, que es la palabra baja del CRO.

4º) Instrucción HLT.

Dentro del segundo grupo, se comentan algunas peculiaridades de las instrucciones que lo componen.

LGDT – LIDT

Sirven para cargar los registros GDTR e IDTR, respectivamente. Tanto GDTR como IDTR, sólo se deben cargar en la inicialización del sistema.

SGDT – SIDT

Almacenan los contenidos de GDTR e IDTR, respectivamente.

Las cuatro instrucciones de carga y almacenamiento de los registros GDTR e IDTR, hacen uso de operandos de memoria de ocho bytes de longitud.

LLDT- LTR

Cargan a LDT y a IR, respectivamente. El operando que manejan es el selector de un descriptor apropiado.

LDTR y IR deben cargarse de forma explícita durante la inicialización del sistema.

SLDT - STR

Almacenan el valor de LDTR y TR, respectivamente y pueden ejecutarse en cualquier nivel de privilegio.

Las tres instrucciones que afectan a MSW, son:

LMSW

Carga en la MSW el valor del operando. Instrucción privilegiada.

SMSW

Almacena el valor de MSW, pudiéndose ejecutar desde cualquier nivel de privilegio.

CLTS

Borra o pone a 0 el señalizador TS de tarea conmutada. Instrucción privilegiada.

Finalmente, la instrucción *HLT* también es privilegiada, pues pasa al procesador a un estado de parada bloqueando la ejecución de instrucciones. Del estado de parada se sale mediante un RESET o una interrupción.

17.3.2- INSTRUCCIONES PROTEGIDAS

Desde las aplicaciones hay que acceder a las puertas de E/S para utilizar los periféricos, pero el acceso ha de estar bajo control del sistema.

En Modo Protegido, el sistema controla el campo de dos bits IOPL del registro EFLAGS, que determina a partir de qué nivel de privilegio se puede acceder a las E/S. Será necesario que el CPL del segmento que intenta usar E/S sea igual o mayor que IOPL para poder ejecutar instrucciones de E/S.

Recuérdese que cada tarea disponía de otro mecanismo para restringir el uso de las E/S. Se trataba del mapa de bits de permiso de E/S, que ocupaba una zona del segmento TSS y que, poniendo a 1 ó a 0 los bits del mapa, prohibían o permitían el acceso *;* cada una de las puertas de E/S.

Mediante el control del campo IOPL (con la instrucción privilegiada POPF) el sistema puede establecer dinámicamente a cada tarea, un control riguroso a las E/S. Así, cuando el sistema pasa a realizar una tarea y coloca el campo IOPL 1, sólo se podrán ejecutar instrucciones de E/S desde segmentos de código que tengan un CPL cuyo valor sea 0 ó 1.

Las **instrucciones protegidas** son:

IN, OUT, INS y OUTS

Corresponden a instrucciones de E/S de operandos y cadenas, respectivamente.

SEI y CLI

Permiso y prohibición de las interrupciones enmascarables.

17.4- INSTRUCCIONES ARITMÉTICAS

AAA

Después de realizarse una suma, ajusta el último byte del resultado al formato BCD.

AAD

Ajusta el dividendo que va a participar en una operación de división para que el resultado, después de efectuada la división, lo proporcione en formato cociente-resto.

AAM

Ajusta el resultado de una operación de dos números BCD a dicho formato.

AAS

Tras realizar una resta, ajusta el último byte del resultado a BCD.

DAA

Tras sumar dos números BCD, dejando el resultado en AL, transforma dicho resultado en dos dígitos BCD.

DAS

Tras efectuar una resta de dos números BCD, dejando el resultado en AL, transforma dicho resultado en dos dígitos BCD.

ADD

Realiza la suma de dos operandos

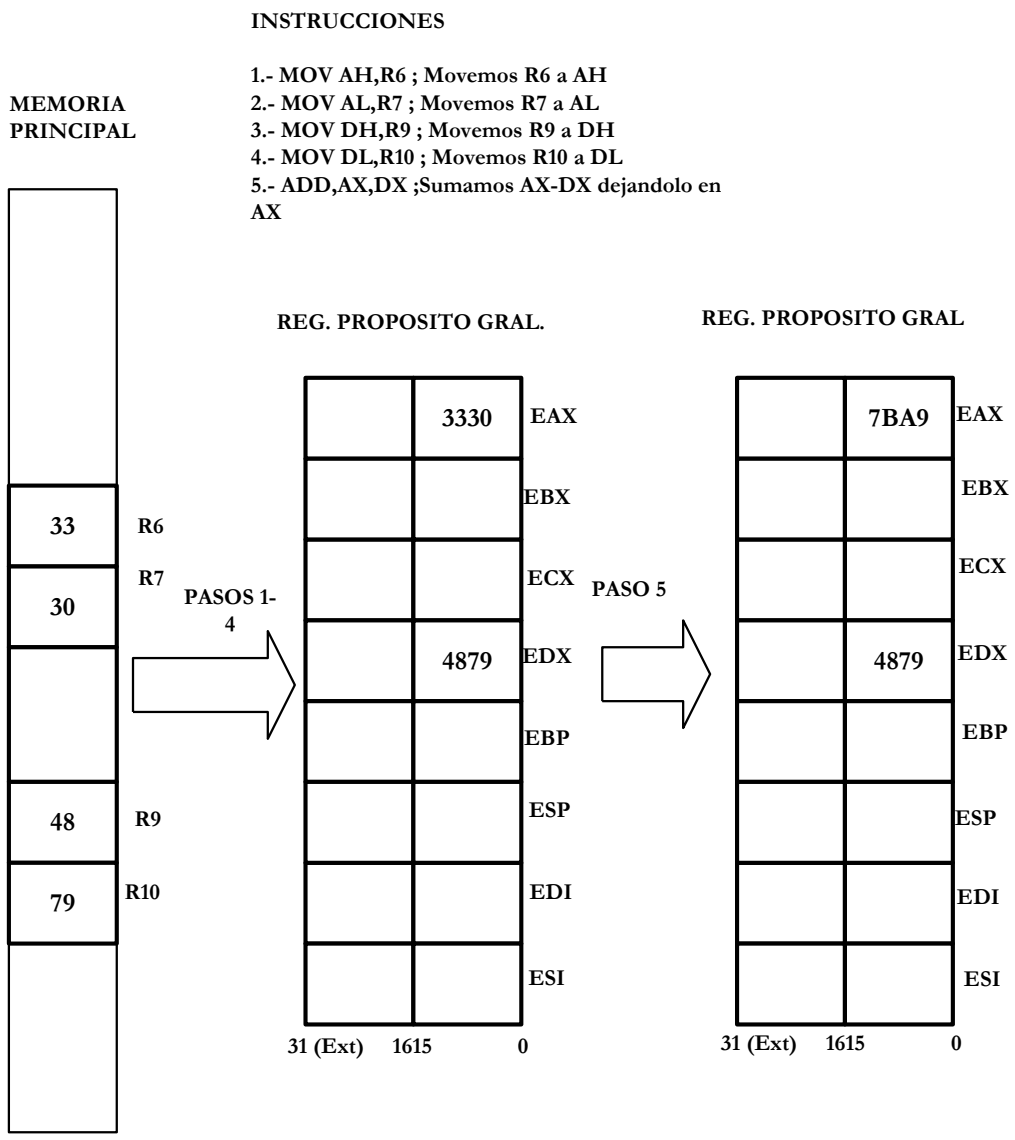


Figura 17.4. Resultado de la operación suma.

ADC

Suma dos operandos y el acarreo.

SUB

Efectúa la resta de dos operandos:

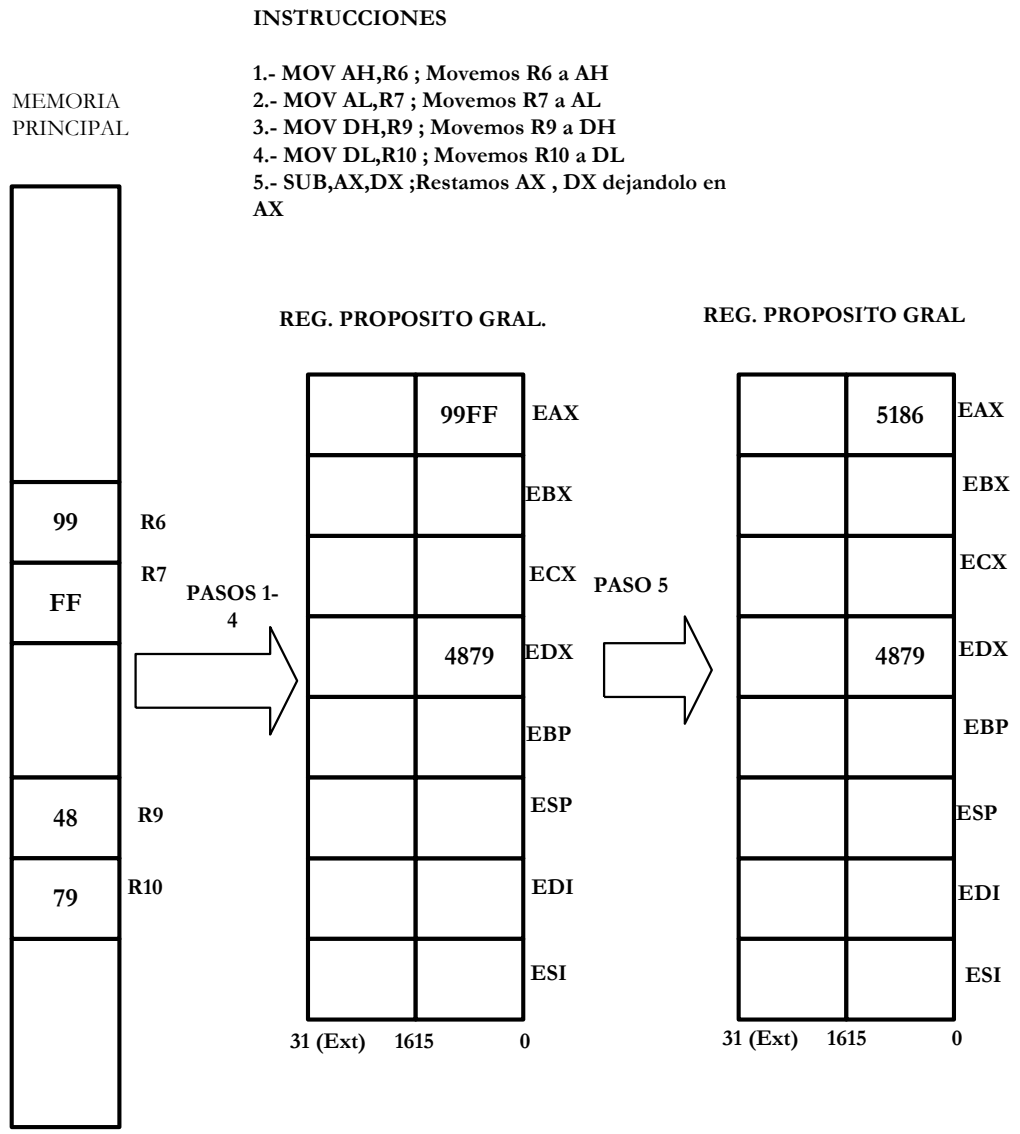


Figura 17.5. Resultado de la operación resta.

SBB

Resta el minuendo del sustraendo más el acarreo o llevada**.**

DEC

Decrementa el operando.

INC

Incrementa el operando.

MUL

Multiplica el operando, sin signo, por AL, AX ó EAX, según el tamaño del operando, dejando el resultado en AX, EAX ó EDX:EAX, respectivamente.

IMUL

Multiplicación de operandos con signo.

DIV

Divide a AL, AX o EAX por el operando especificado en la instrucción, dejando el resultado en dos registros, en forma de Resto:Cociente.

IDIV

División con signo, semejante a la anterior.

CBW/CWD

Convierten AL y AX a doble tamaño, respectivamente.

NEG

Realiza el complemento a dos del operando.

CMP

Compara dos operandos, reflejando el resultado únicamente en los señalizadores.

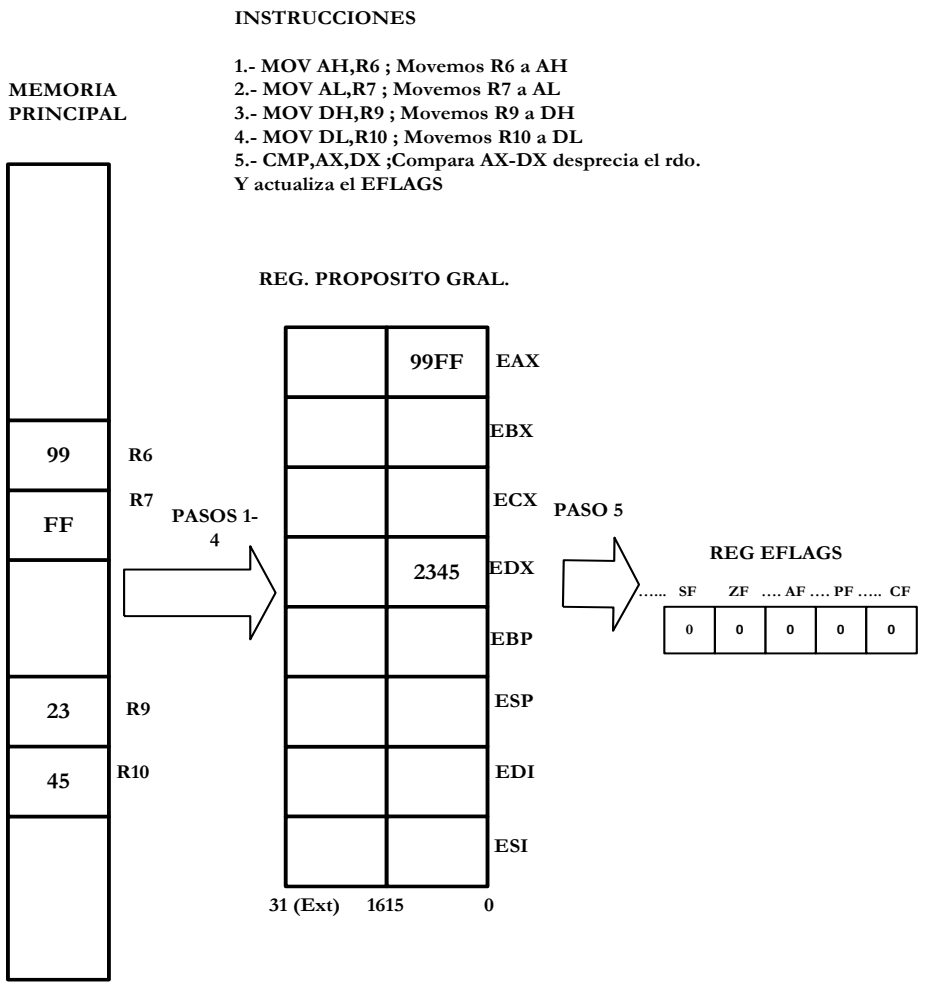


Figura 17.6. Resultado de comparar dos posiciones de memoria.

17.4.1- NUEVAS INSTRUCCIONES ARITMÉTICAS DEL PENTIUM.

CWDE

Convierte una palabra a doble palabra.

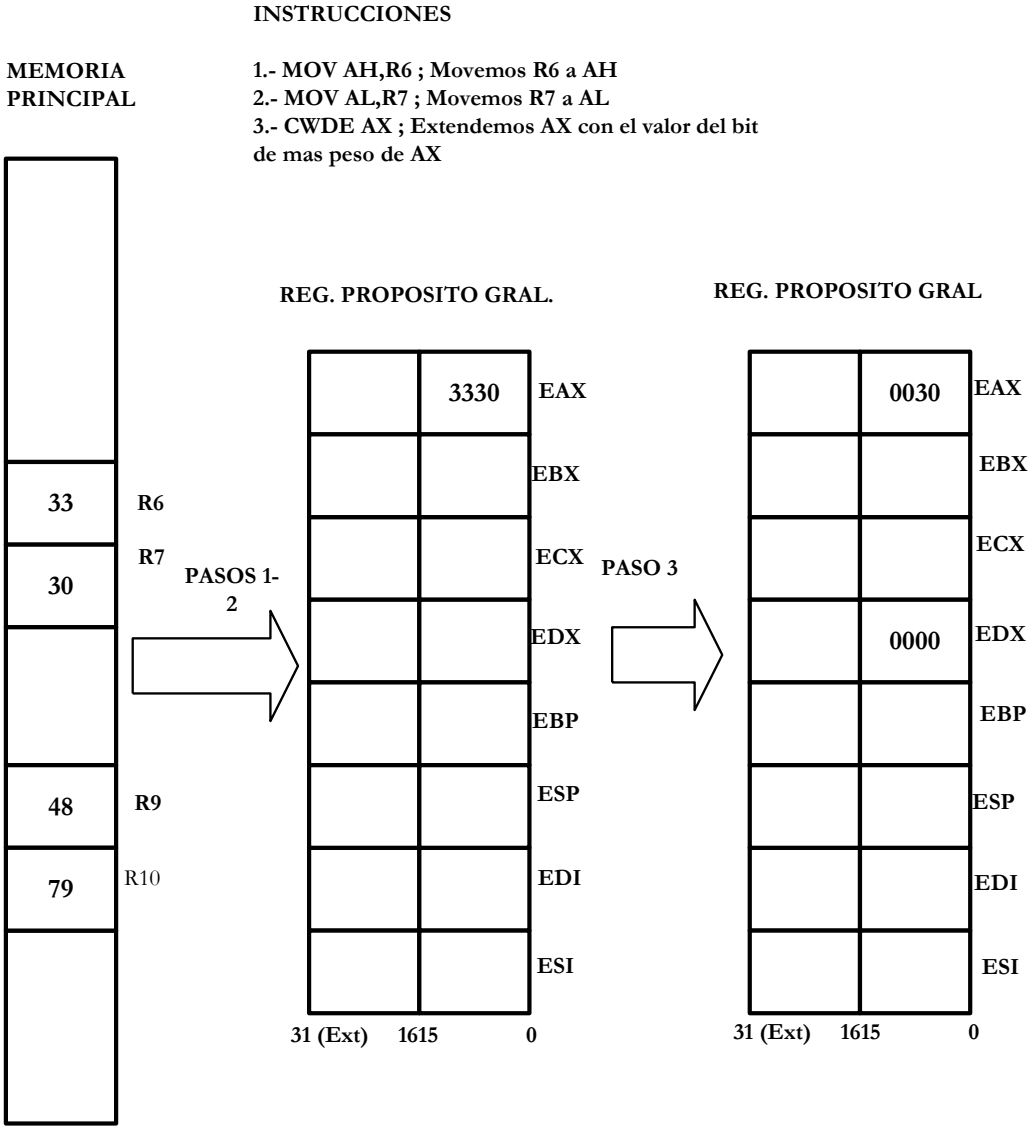


Figura 17.7. Convierte una palabra en doble palabra

CDQ

Convierte una doble palabra en cuádruple.

17.5- INSTRUCCIONES LÓGICAS

AND-OR-XOR-NOT

Realizan las operaciones lógicas AND, OR, EOR y NOT, respectivamente.

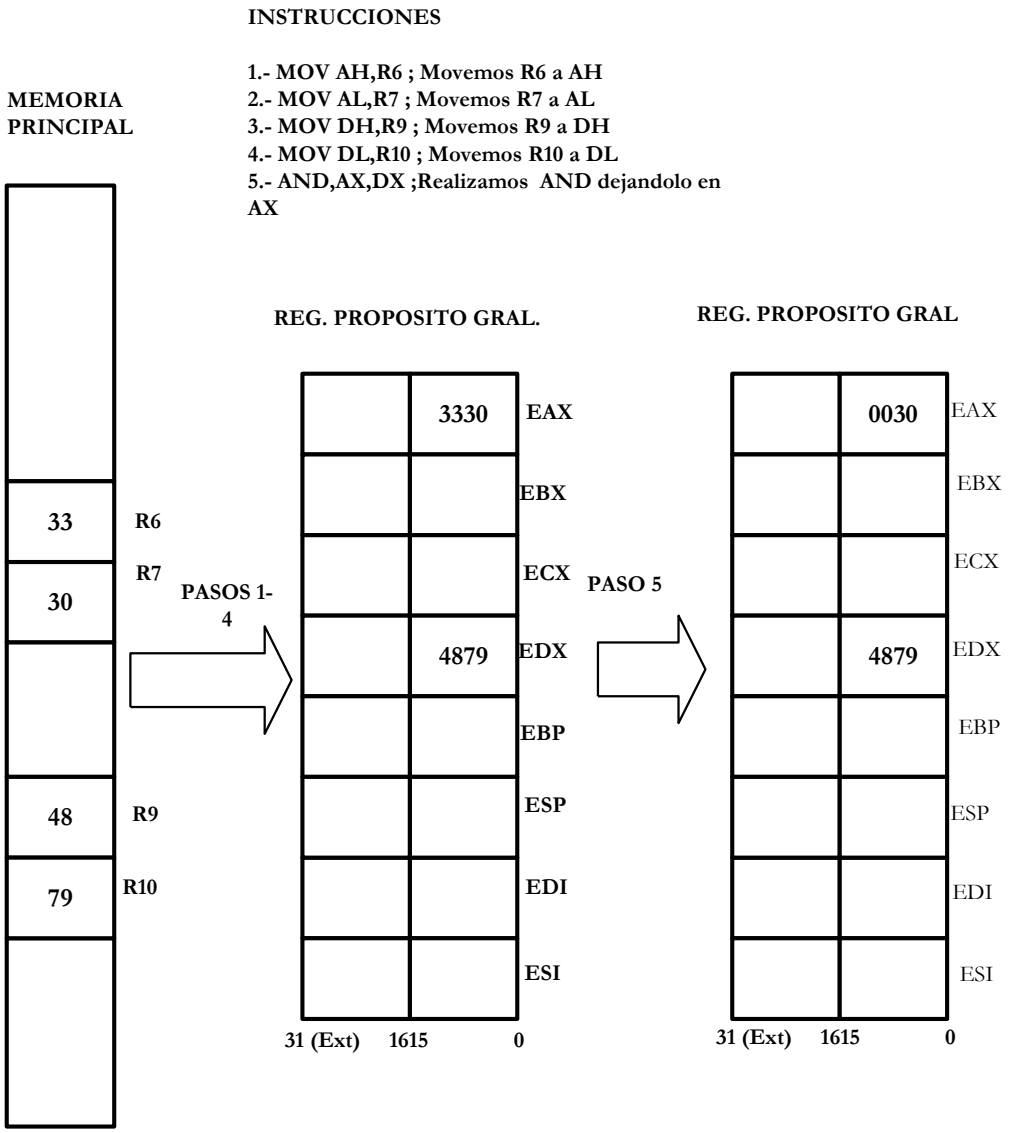


Figura 17.8. Resultado de la operación AND

ROL/ROR

Rotación a la izquierda o derecha de los bits del primer operando, tantas veces como lo indique CL u otro operando.

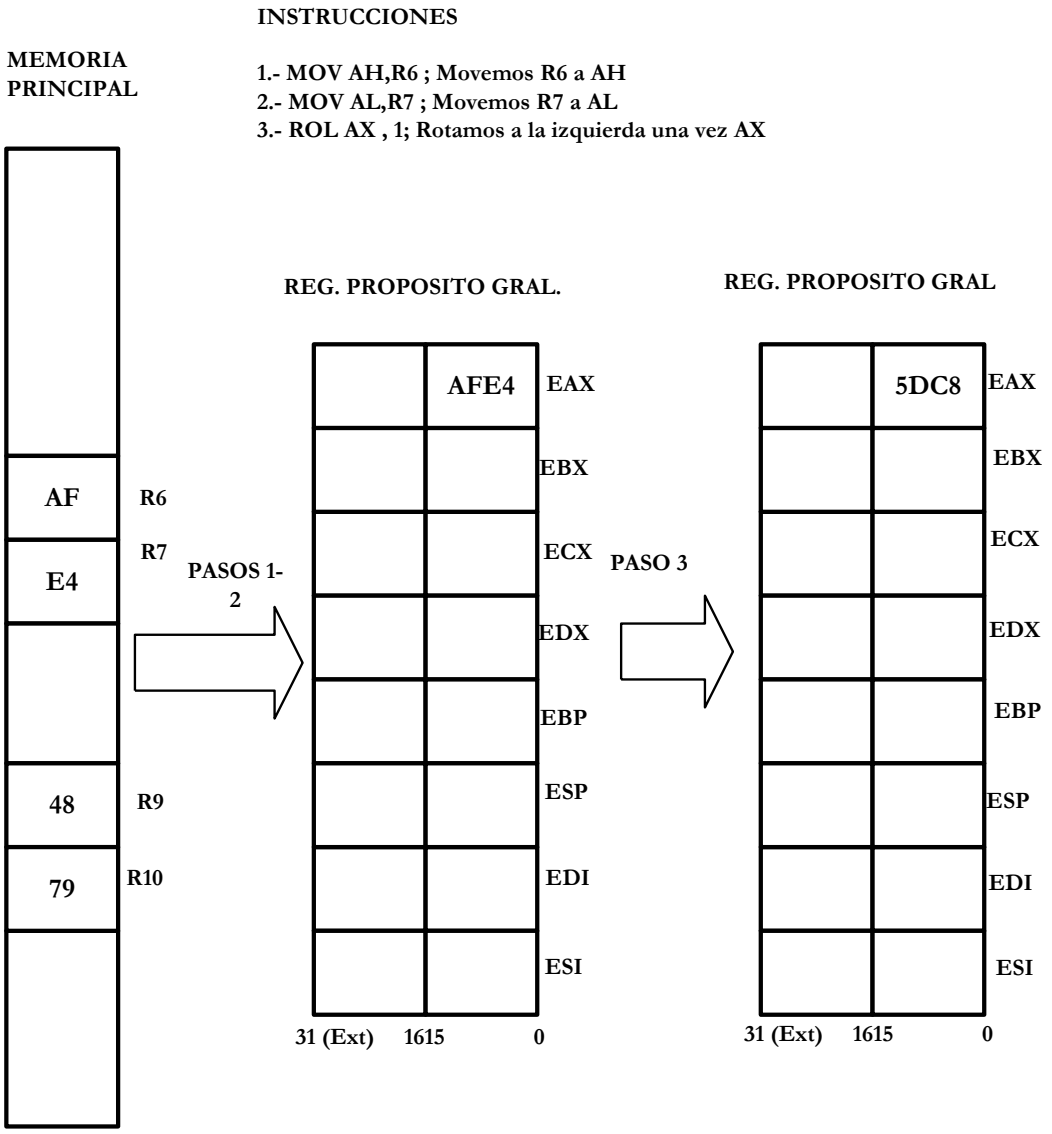


Figura 17.9. Resultado de rotar a la izquierda el operando

RCL/RCR

Rotación a izquierda o derecha, respectivamente, de los bits del primer operando, junto con el acarreo, el número de veces especificado por CL.

TEST

Realiza la operación lógica AND de los operandos, sin resultado. Sólo afecta a los señalizadores.

SAL/SAR

Desplazamiento aritmético a izquierda o derecha.

SHL/SHR

Desplazamiento lógico (sin preservar el valor del bit de signo) a izquierda o derecha.

17.5.1- Nuevas instrucciones lógicas del Pentium

SHLD

Desplaza el contenido del primer y segundo operando, conjuntamente, a la izquierda, el número de veces especificado en el tercer, operando.

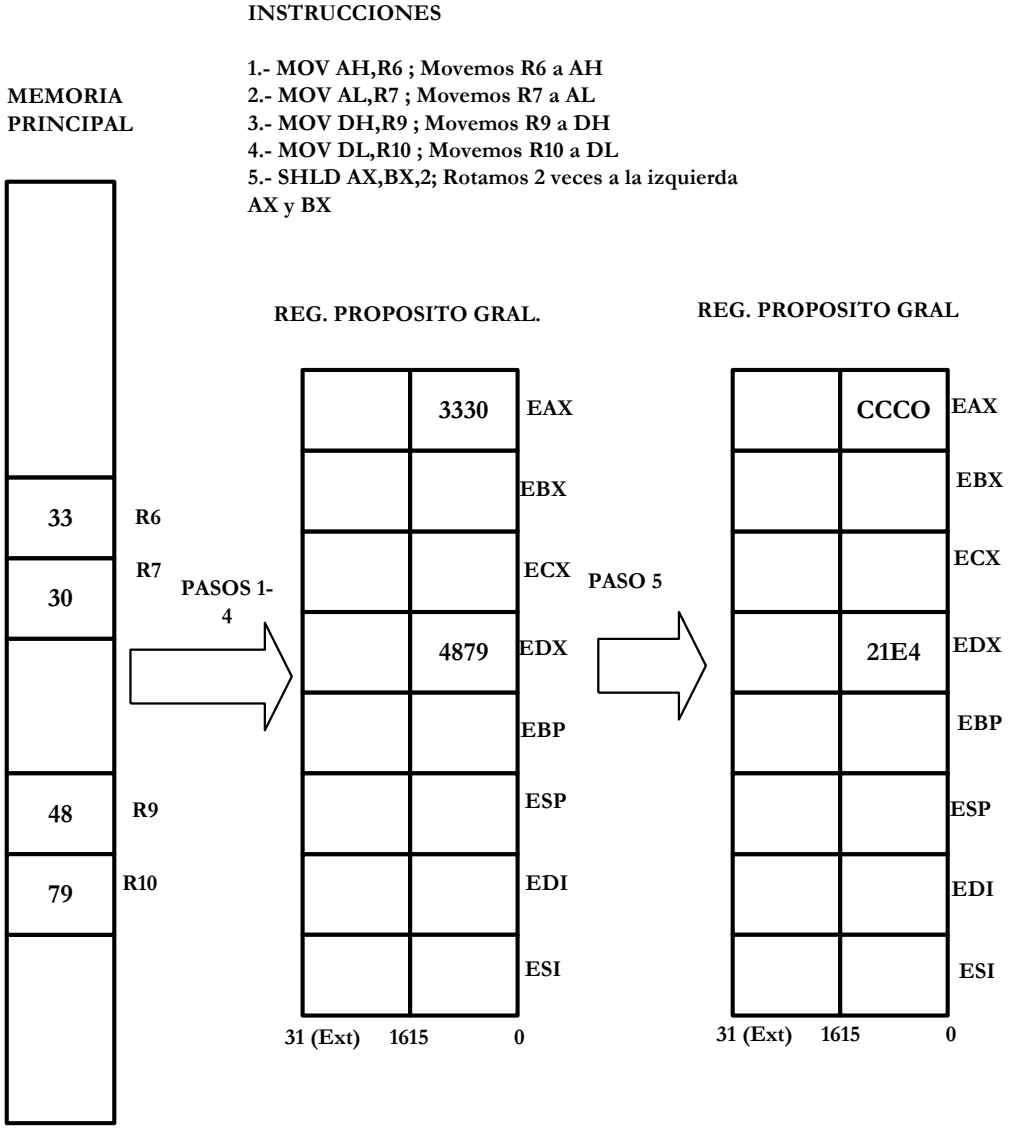


Figura 17.10. Resultado de la operación de rotar dos veces a la izquierda dos operandos distintos

SHRD

Igual formato y comportamiento que SHRD, pero realizando el desplazamiento a la derecha.

17.6- INSTRUCCIONES DE CADENAS

CMPS - CMPSB - CMPSW

Sirven para comparar cadenas, restando el contenido del elemento apuntado por ES:EDI, con el de DS:ESI, afectando sólo a los señalizadores.

INSTRUCCIONES

1.- CMPS ;Compara ESI de DS con EDI de CS.
Desprecia el rdo. Y actualiza el EFLAGS

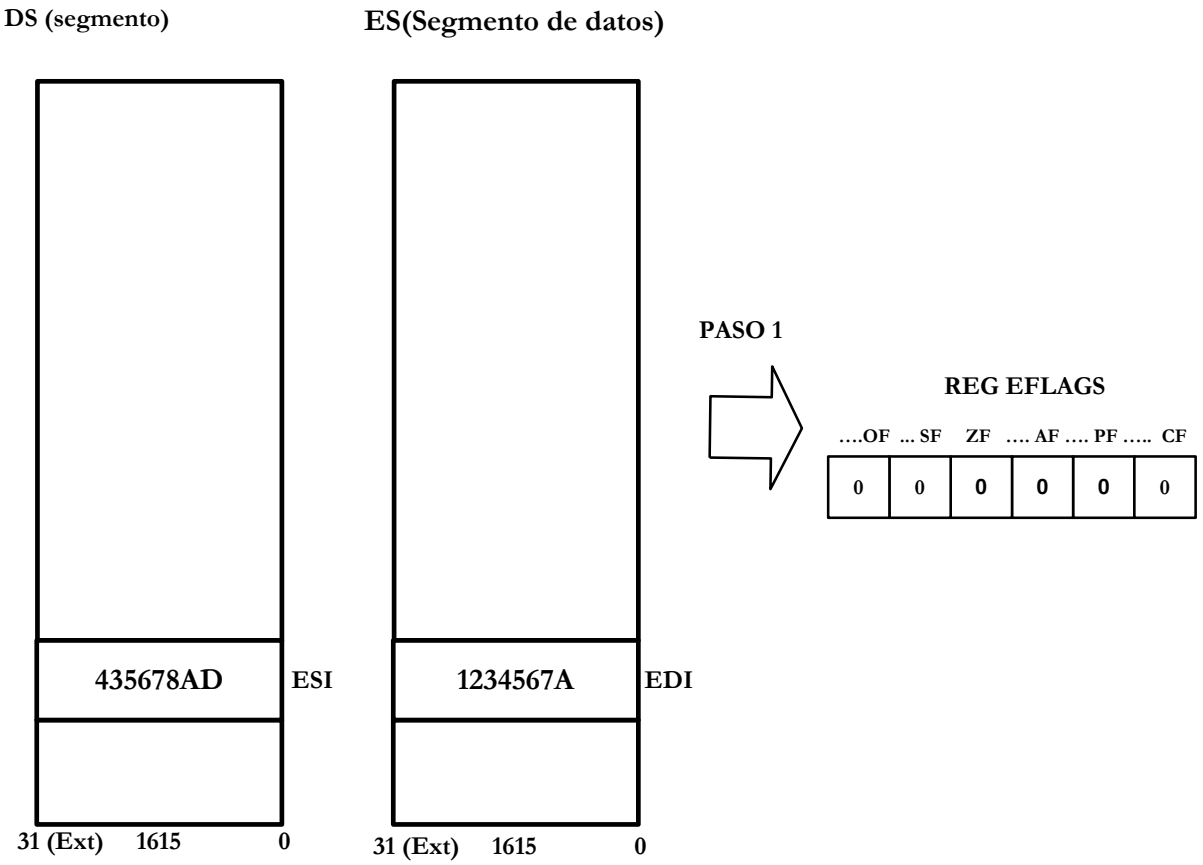


Figura 17.11. Resultado de comparar el contenido de dos segmentos y la representación de la actualización de EFLAGS

MOVS

Mueve el contenido de la dirección apuntada por DS:ESI a la referenciada por el puntero ES:EDI, incrementando el valor de los registros índice, según el tamaño en bytes de los operandos.

LODS -ODSB - LODSW

Transfiere al Acumulador el elemento apuntado por DS:ESI.

STOS - STOSB - STOSW

Transfiere el contenido del Acumulador a la dirección apuntada por ES:EDI.

SCAS - SCASB - SCASW

Resta el contenido direccionado por ES:EDI al Acumulador, afectando exclusivamente a los señalizadores.

INS - INSB - INSW

Carga en la dirección apuntada por ES:EDI el valor de la puerta de E/S apuntada por DX.

OUTS - OUTSB - OUTSW

Saca por la puerta apuntada por DX, uno o dos bytes a partir de la dirección señalada por ES:EDI.

REP

Es un prefijo que se antepone a ciertas instrucciones de cadenas. Repite la ejecución de la instrucción a la que antecede tantas veces como indica el valor del contenido de ECX.

REPE - REPZ

Se repite la instrucción a la que antecede hasta que ECX valga 0, o bien, el señalizador Z = 0.

REPNE - REPNZ

Es igual que la anterior, excepto que termina la repetición cuando Z = 1, o bien, ECX = 0.

XLAT-XLATB

Depositan en AL el contenido de la dirección obtenida al sumar EBX + AL. Sirve principalmente para la creación de ficheros

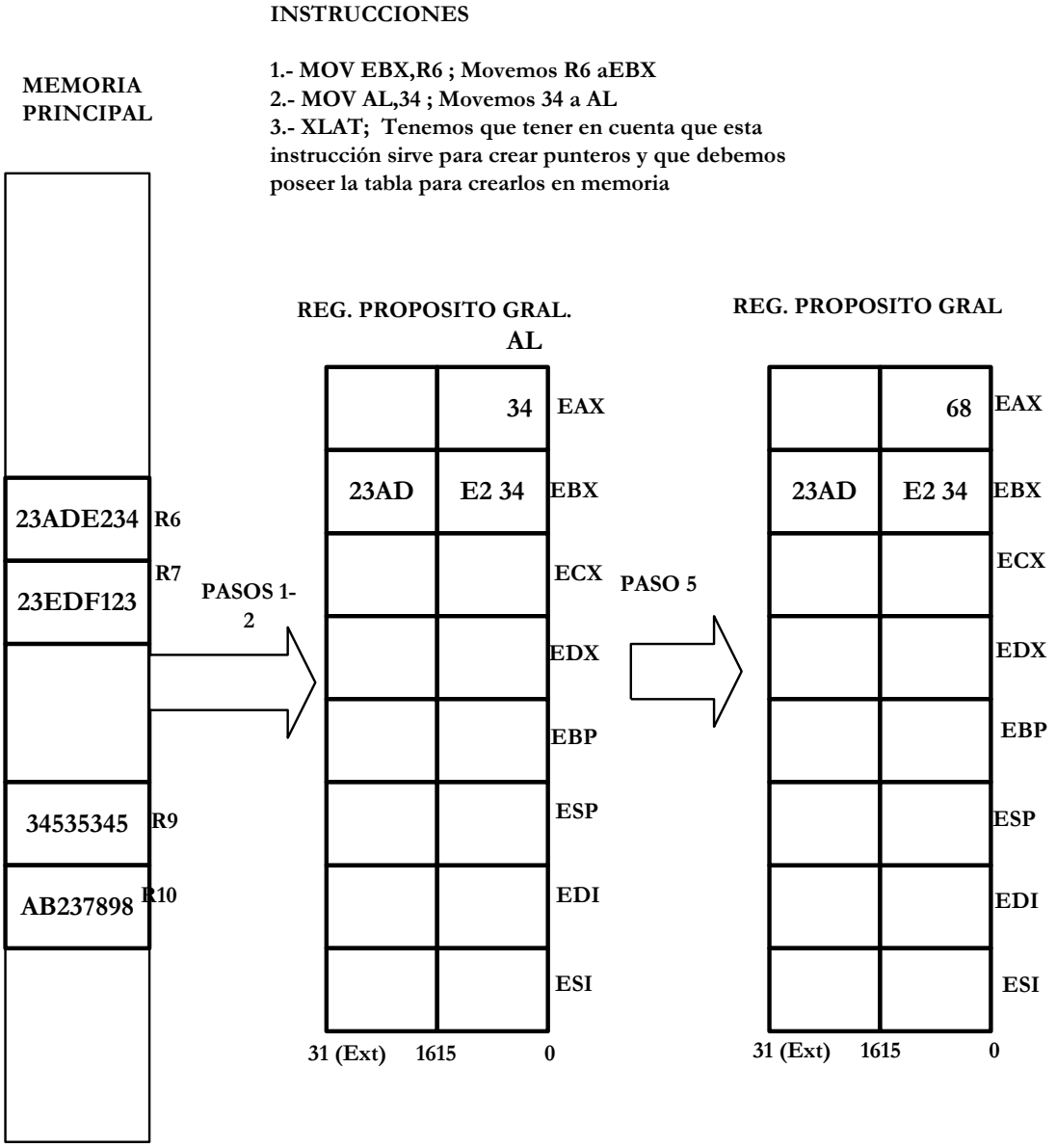


Figura 17.12.Representación de la creación de punteros en memoria mediante XLAT.

17.7- INSTRUCCIONES DE TRANSFERENCIA DE CONTROL

JMP

Realiza un salto a la dirección que se indica en el descriptor referenciado.

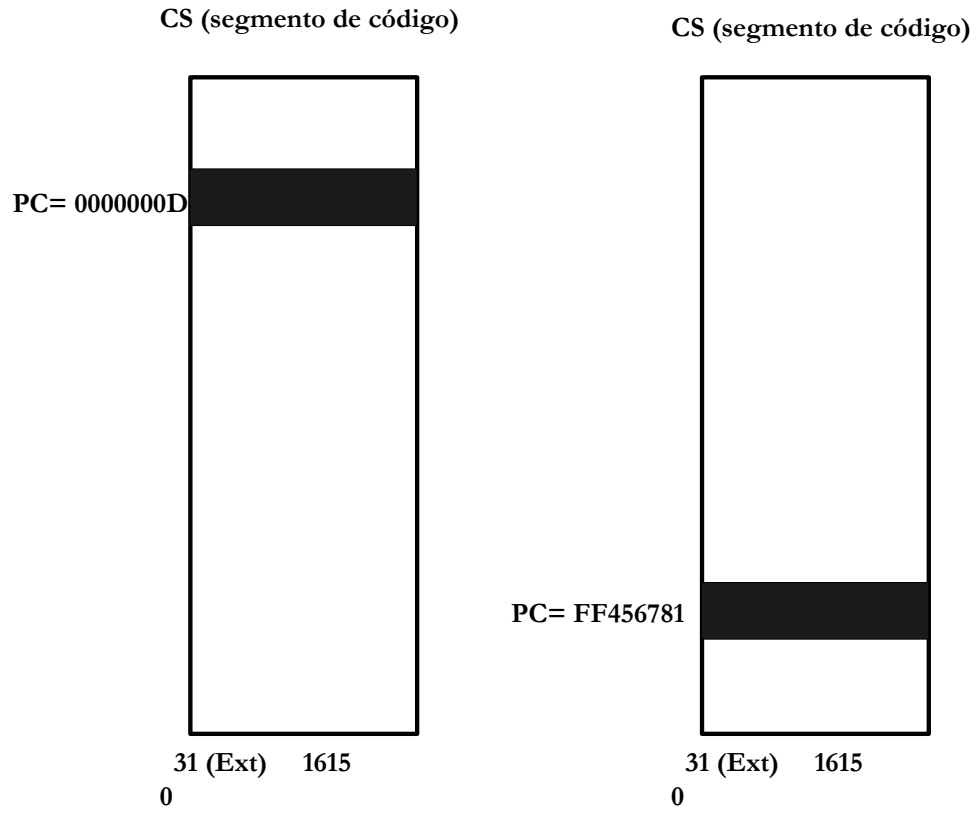


Figura 17.13.Representación de un salto incondicional dentro de un segmento de código.

LOOP

Provoca un salto a una etiqueta corta (situada entre -128 y + 127 posiciones).

LOOPZ - LOOPE - LOOPNZ - LOOPNE

Salto corto mientras ECX no sea 0 y Z =0 ó 1, según sean las dos primeras o las dos últimas.

CALL

Llamada o salto a una rutina.

RET

Retorno de una rutina.

17.7.1- INSTRUCCIONES DE TRANSFERENCIA DE CONTROL ESPECIALES

*J*cc son instrucciones de salto condicional en las que CC representa el código de condición y son las siguientes.

JA

Salta si el primer operando es mayor que el Segundo. Sin signo

JAE

Salta si el primer operando es menor que el Segundo. Sin signo

JB

Salta si el primer operando es menor igual que el Segundo. Sin signo

JBE

Salta si el primer operando es mayor igual que el Segundo. Sin signo

JE

Salta si el bit Z del EFLAGS es igual a 1

JO

Salta si el bit de OVERFLOW del EFLAGS es igual a 1.

JNO

Salta si el bit de OVERFLOW del EFLAGS es igual a 0.

JCXZ

Salta si el registro de CX es distinto a 0.

JS

Salta si el bit S del EFLAGS es igual a 1.

JNS

Salta si el bit S del EFLAGS es igual a 0.

JNA

Salta si el primer operando es mayor que el Segundo. Con signo

JNAE

Salta si el primer operando es menor que el Segundo. Con signo

JNB

Salta si el primer operando es menor igual que el Segundo. Con signo

JNBE

Salta si el primer operando es mayor igual que el Segundo. Con signo

JC

Salta si el bit de CARRY del EFLAGS es igual a 1

INSTRUCCIONES

MEMORIA PRINCIPAL

1.- CMP AX,DX;Comparamos los contenidos de AX-DX previamente introducidos y actualizamos EFLAGS.

4.- JC FF45679E; Miramos si el Flag de Carry está a uno y si lo cumple saltamos al contenido de la instruccion y lo dejamos en PC

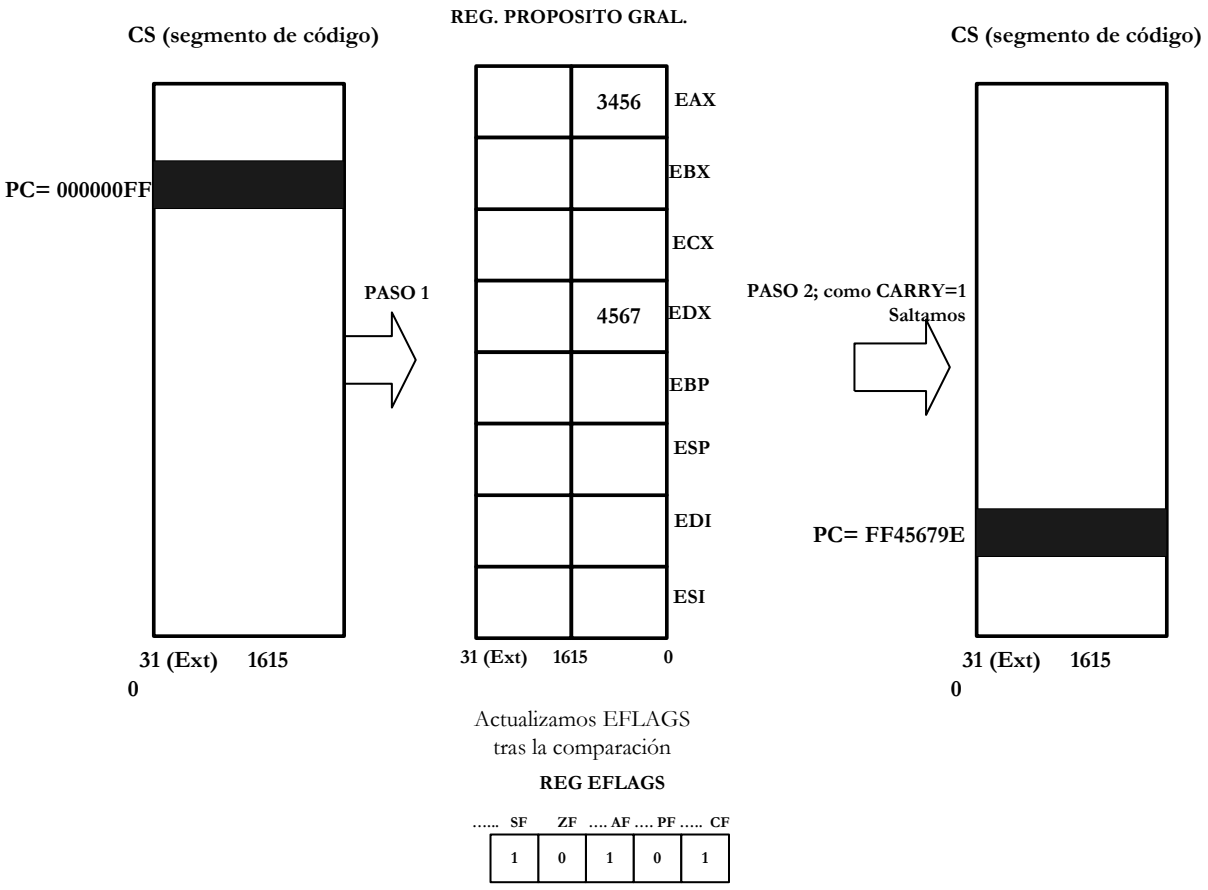


Figura 17.14.Representación de un salto condicional (CARRY=1?) dentro de un segmento de código.

JNC

Salta si el bit de CARRY del EFLAGS es igual a 1

JP

Salta si la paridad es impar o no-paridad

JNP

Salta si la paridad es par o hay paridad

17.8- INSTRUCCIONES DE TRANSFERENCIA DE DATOS

IN - OUT

Entrada y salida de información desde las puertas de E/S.

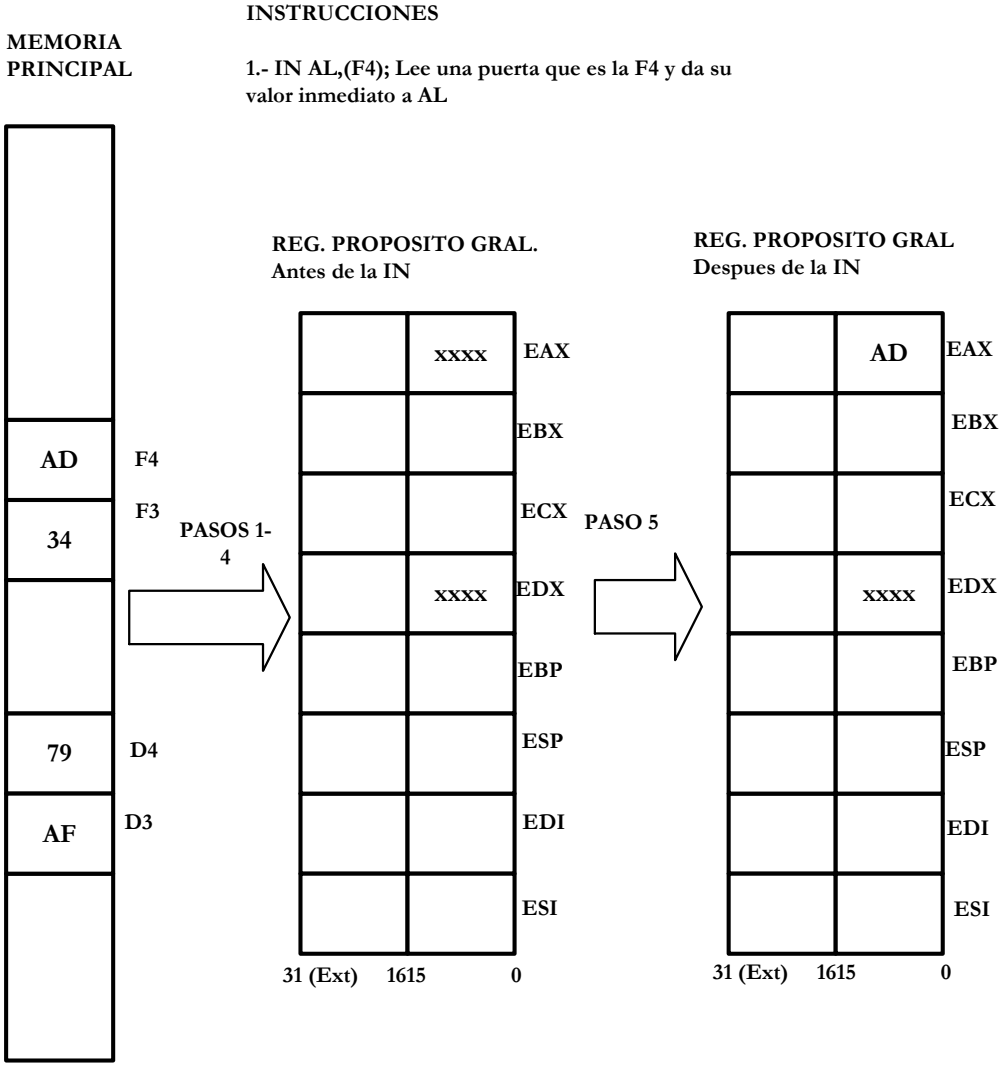


Figura 17.15. Introducimos un valor de una posición de memoria en AL mediante la instrucción IN

POP

Se transfiere desde la cima de la pila una información a un registro.

POPA

Transfiere desde la pila ocho palabras que se cargan en los ocho registros generales.

POPF

Transfiere desde la cima de la pila una palabra al registro de señalizadores.

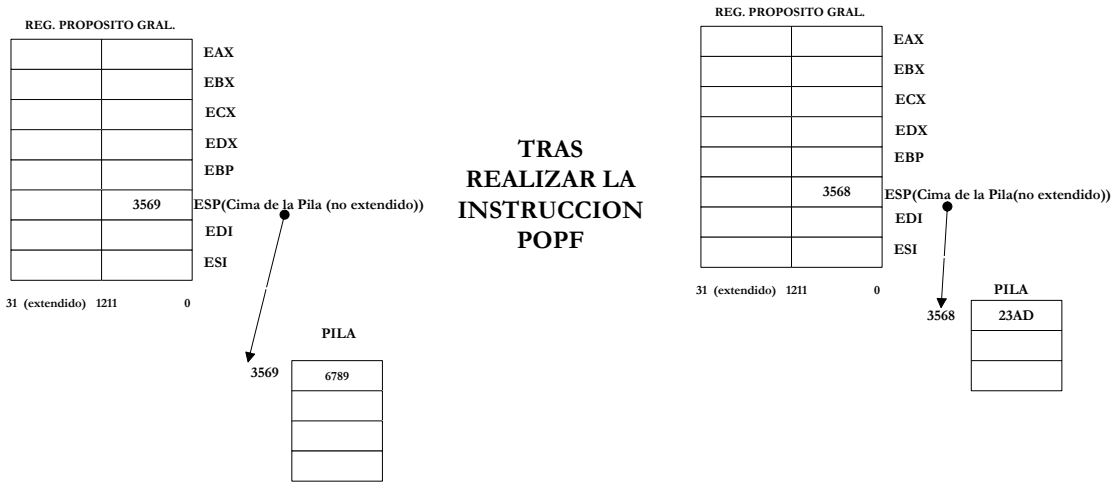


Figura 17.16. Representación del antes y el después de realizar la instrucción POPF.

PUSH

Transfiere un registro a la pila.

PUSHA

Transfiere a la cima de la pila ocho registros generales de 16 bits (AX, BX, CX, DX, SP, BP, SI y DI).

PUSHF

Transfiere a la pila el registro de los señalizadores.

MOV

Transfiere al primer operando el valor del segundo.

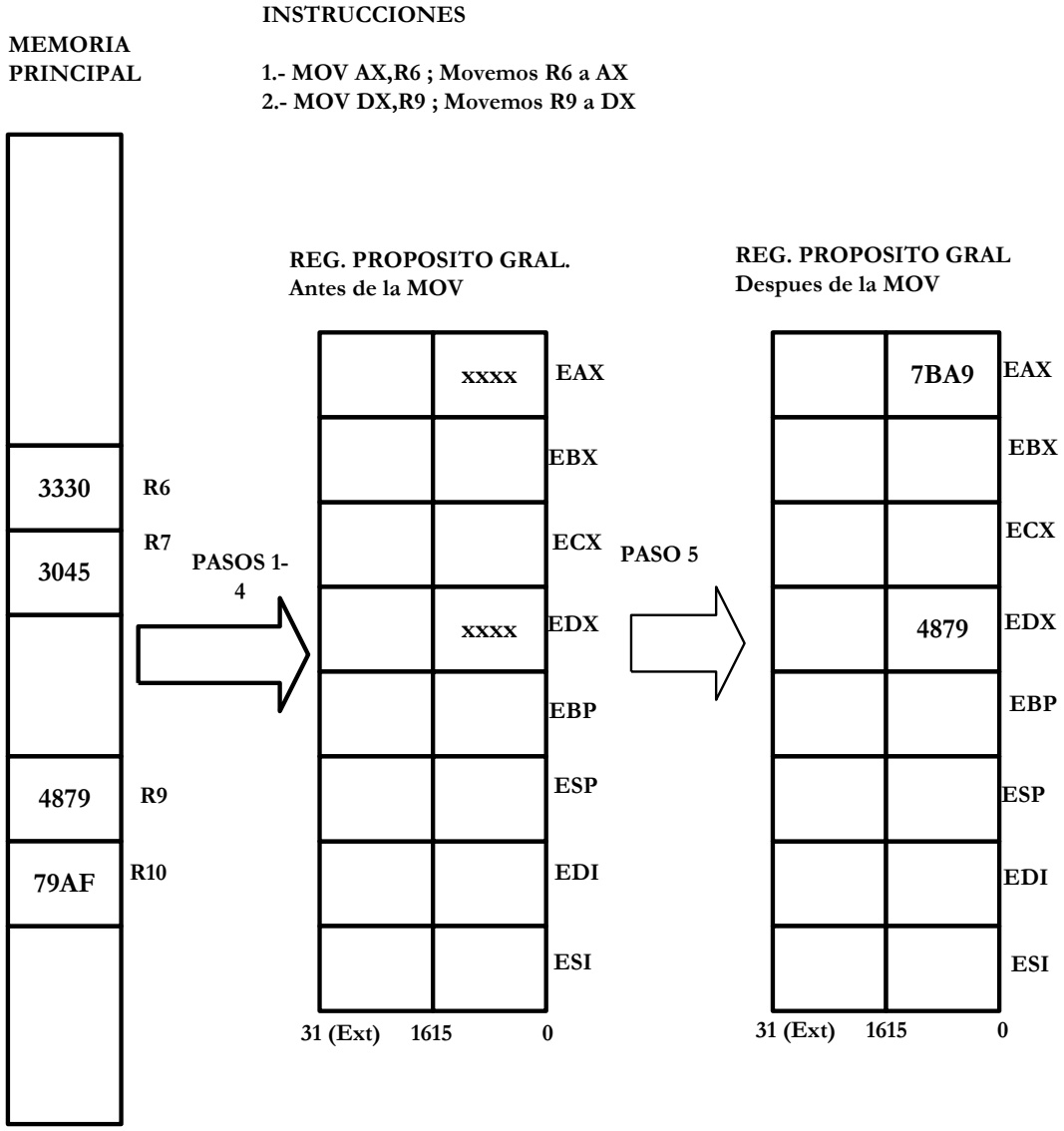


Figura 17.17. Representación gráfica de movimiento de operandos mediante MOV.

XCHG

Intercambia el contenido de los operandos.

LEA

Transfiere a un registro el valor de la dirección que corresponde al segundo operando.

17.8.1- NUEVAS INSTRUCCIONES DE TRANSFERENCIA DE DATOS

El Pentium admite nuevas instrucciones de transferencia para operandos de 32 bits, como *POPAD, POPFD, PUSHFD* y *PUSHAD*. También está la instrucción *MOVSX,* que extiende el signo de un byte ó una palabra a 16 ó 32 bits, respectivamente.

17.9- INSTRUCCIONES DE CONTROL DE LOS SEÑALIZADORES

CLC - STC

Ponen a 0 ó a 1 el señalizador de acarreo, respectivamente.

Reservado **CF (Antes CLC)**

00……..0	ID	VIF	VIP	AC	VM	RF	0	NT	IO	PL	OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	x
31	**21**																					**0**

REALIZAMOS LA INSTRUCCIÓN

……. CLC; …….

Reservado **CF(Despues CLC)**

00……..0	ID	VIF	VIP	AC	VM	RF	0	NT	IO	PL	OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	0
31	**21**																					**0**

Figura 17.18. Representación de cómo quedaría EFLAGS tras realizarse la operación CLC

CLD - STD

Ponen a 0 ó a 1 al señalizador D, respectivamente.

CMC

Complementa el valor del señalizador C.

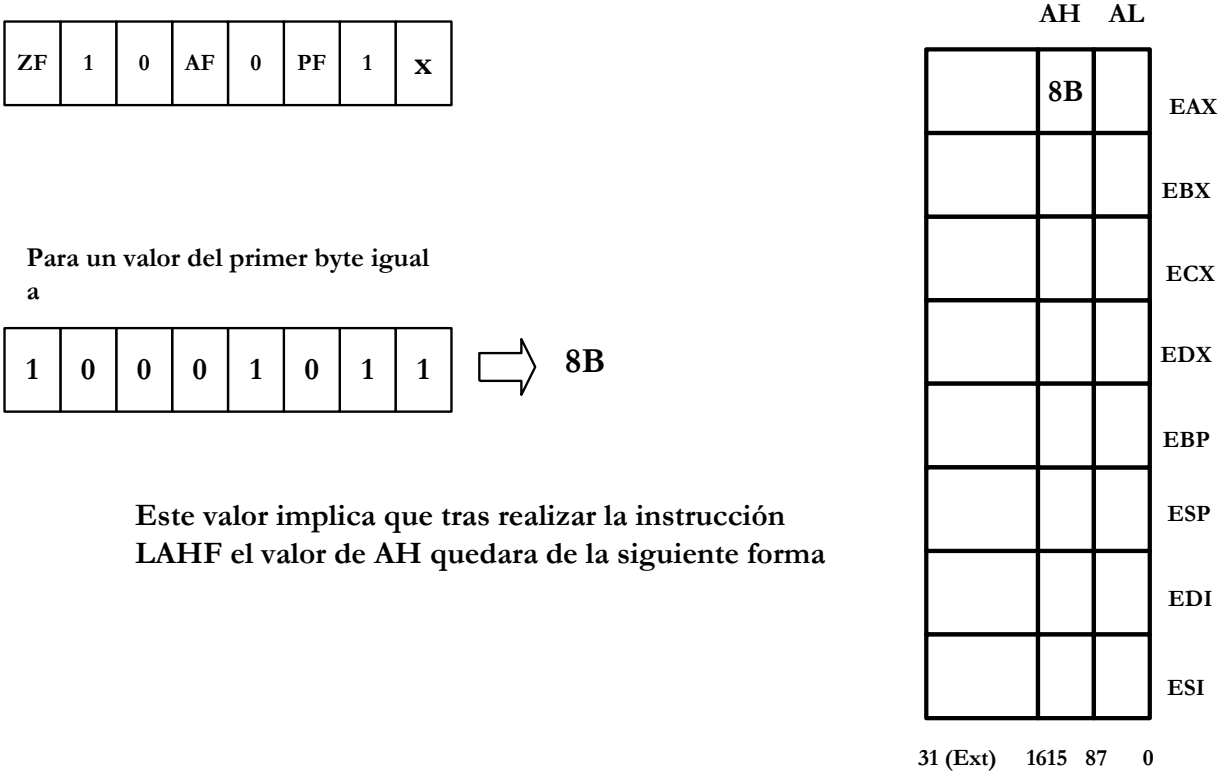
CLI

Pone a 0 el señalizador 1. Inhabilita interrupciones enmascarables.

LAHF

Transfiere al registro AH el byte de menos peso del registro de señalizadores.

Estamos ante el primer byte del registro EFLAGS que va a ser transferido a AH (registro de proposito general)



Para un valor del primer byte igual a

1	0	0	0	1	0	1	1

⇨ **8B**

Este valor implica que tras realizar la instrucción LAHF el valor de AH quedara de la siguiente forma

Figura 17.19. Carga del registro AH con el byte de menos peso de EFLAGS

SAHF

Transfiere el contenido de AH al byte de menos peso del registro de señalizadores.

17.10- INSTRUCCIONES DE ASIGNACIÓN CONDICIONAL

Es un grupo de instrucciones nuevas en el Pentium.

Si se cumple la condición implícita en la instrucción, se pone a 1 el operando y, en caso contrario, se pone a 0. Las instrucciones son las siguientes:

SETA

Pone a uno si el primer operando es mayor que el Segundo. Sin signo

SETAE

Pone a uno si el primer operando es menor que el Segundo. Sin signo

SETB

Pone a uno si el primer operando es menor igual que el Segundo. Sin signo

SETBE

Pone a uno si el primer operando es mayor igual que el Segundo. Sin signo

SETC

Pone a uno si el bit de CARRY del EFLAGS es igual a 1

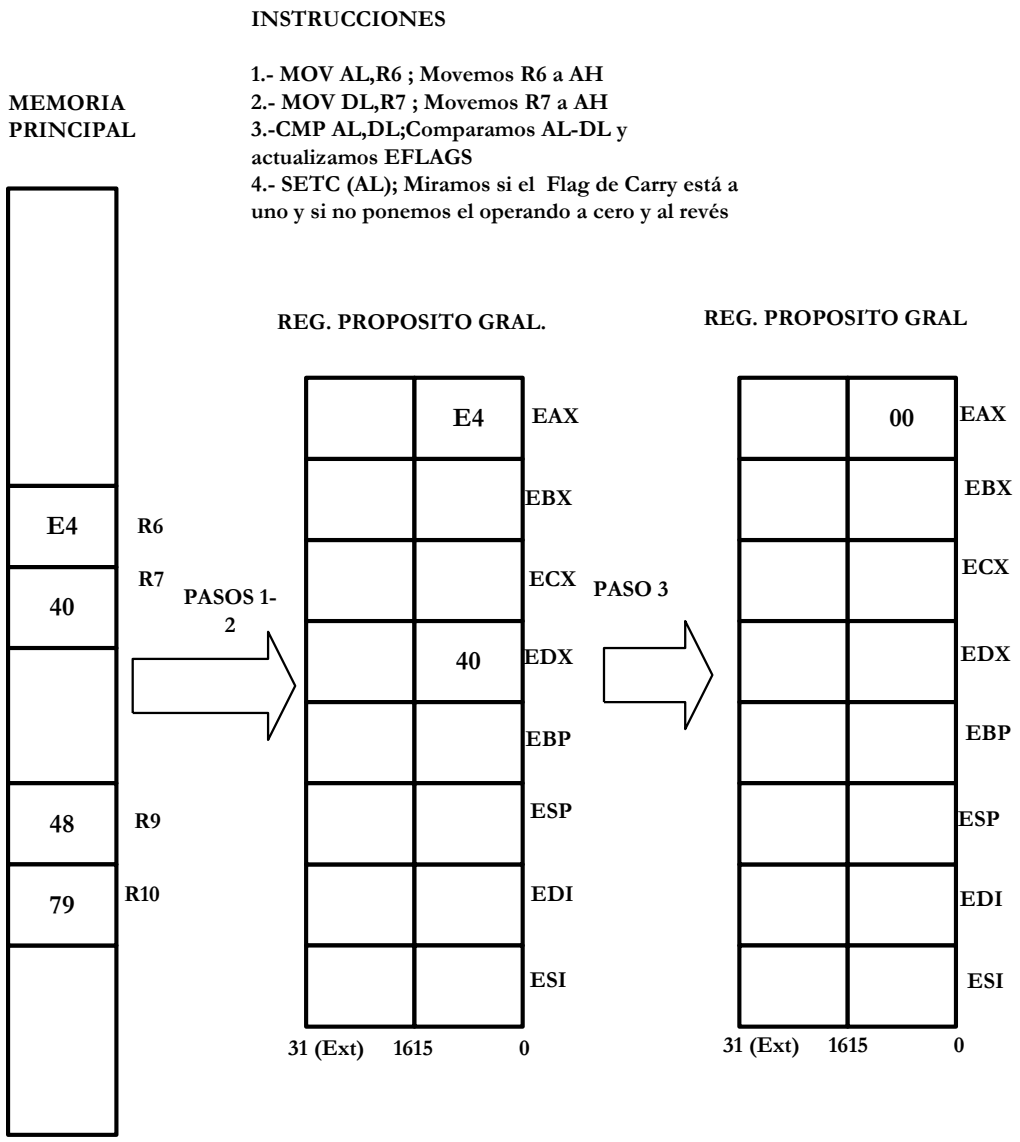


Figura 17.20. La condición no se cumple por lo que el registro se pone a cero

SETNC

Pone a uno si el bit de CARRY del EFLAGS es igual a 1

SETE

Pone a uno si el bit Z del EFLAGS es igual a 1

SETO

Pone a uno si el bit de OVERFLOW del EFLAGS es igual a 1.

SETNO

Pone a uno si el bit de OVERFLOW del EFLAGS es igual a 0.

SETCXZ

Pone a uno si el registro de CX es distinto a 0.

SETS

Pone a uno si el bit S del EFLAGS es igual a 1.

SETNS

Pone a uno si el bit S del EFLAGS es igual a 0.

SETNA

Pone a uno si el primer operando es mayor que el Segundo. Con signo

SETNAE

Pone a uno si el primer operando es menor que el Segundo. Con signo

SETNB

Pone a uno si el primer operando es menor igual que el Segundo. Con signo

SETNBE

Pone a uno si el primer operando es mayor igual que el Segundo. Con signo

SETP

Pone a uno si la paridad es impar o no paridad

SETNP

Pone a uno si la paridad es par o hay paridad

17.11- INSTRUCCIONES DE BIT

Todas estas instrucciones son nuevas.

BT

Asigna al señalizador CF el valor del bit del primer operando, quedando especificada su posición por el segundo operando.

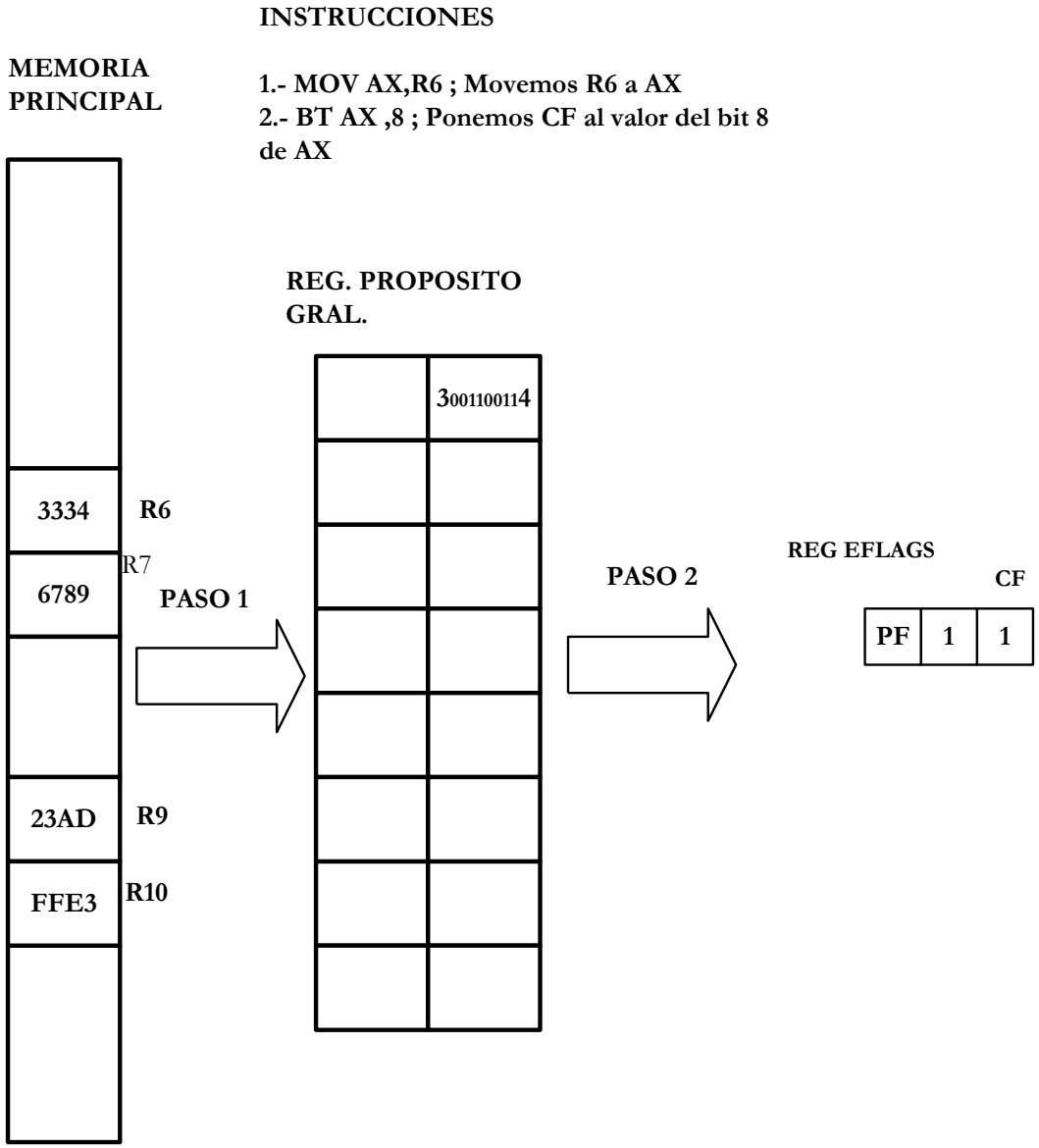


Figura 17.21. El octavo bit de AX es igual a UNO por lo que ponemos CF de EFLAGS a UNO.

BTC

Realiza la misma operación que BT, pero también complementa el bit especificado en La instrucción.

BTR

Igual que BT, pero pone a 0 el bit especificado en la instrucción.

BTS

Igual que la anterior, pero pone a 1 el bit especificado en la instrucción.

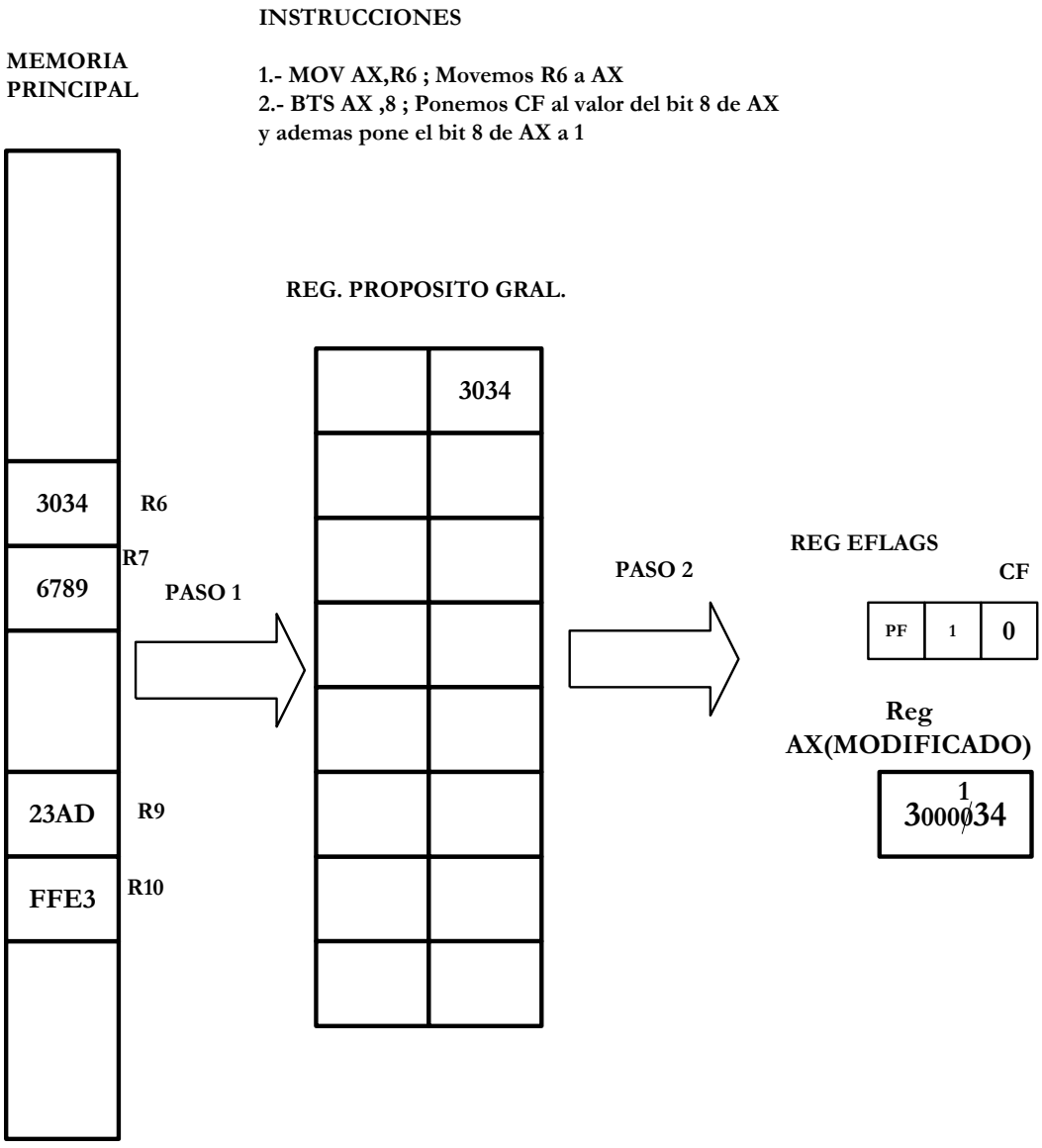


Figura 17.22. El octavo bit de AX es igual a CERO por lo que ponemos CF de EFLAGS a CERO y además complementamos ese bit.

BSF

Si todos los bits, recorridos de menor a mayor peso, del primer operando, especificados por el segundo, son ceros, Z = 1. Si se encuentra algún bit a 1, Z = 0 guardando en el primer operando la posición del primer bit a 1 que se haya encontrado.

BSR

Igual que la anterior instrucción, pero la exploración se realiza desde el bit de más peso al de menos peso.

17.12-INSTRUCCIONES DE ALTO NIVEL

BOUND

Comprueba que un operando está comprendido entre dos límites, y, en caso contrario, genera la excepción *5*.

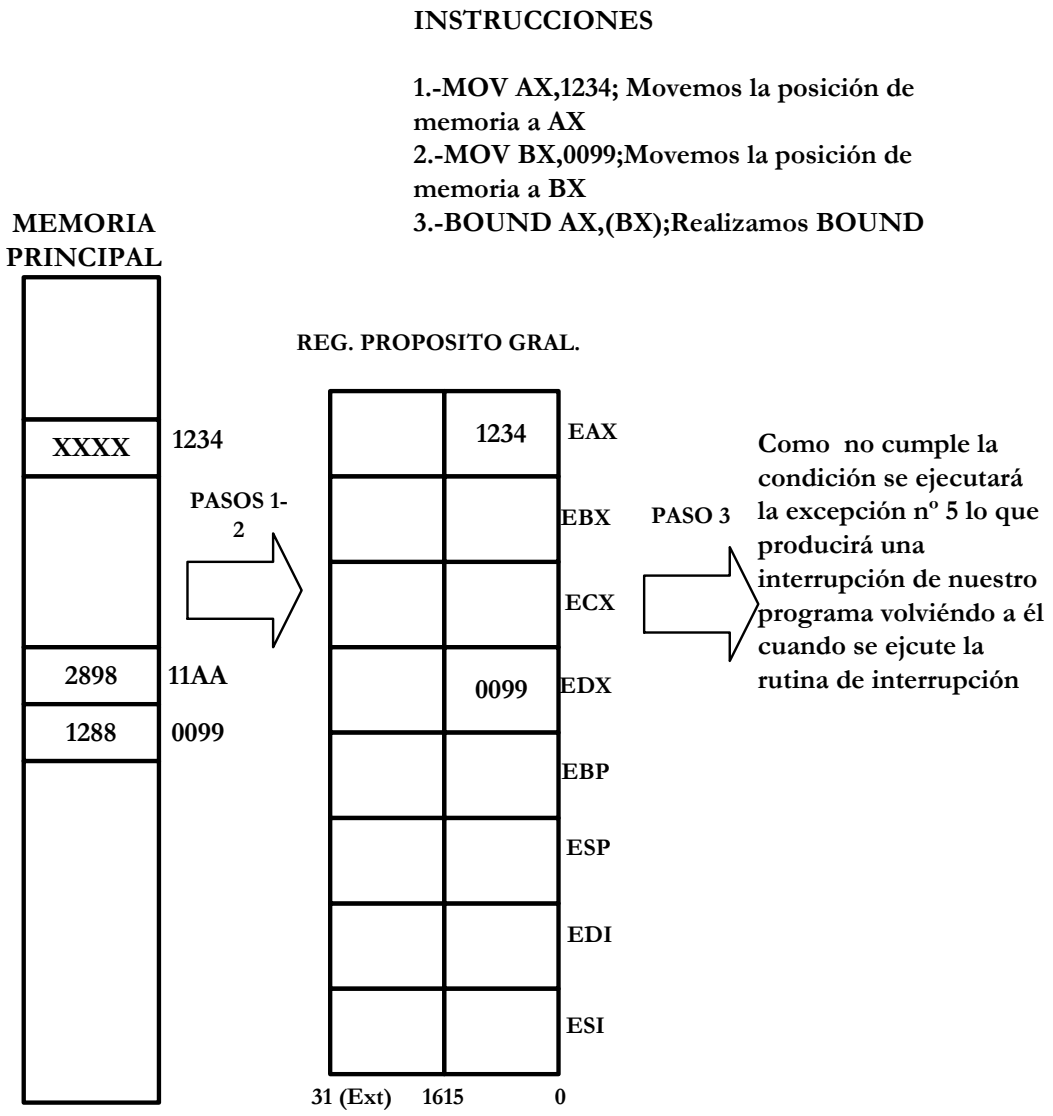


Figura 17.23. BOUND

ENTER

Sirve para crear en la nueva pila que se usa, tras una llamada a un procedimiento, un espacio reservado para el paso de parámetros. El primer operando expresa el número de bytes que se reservan en la nueva pila. El segundo operando expresa el grado de anidamiento.

LEA VE

Elimina el espacio reservado en la pila del procedimiento saliente, asignando como nueva pila la del procedimiento anterior en anidamiento.

17.13- INSTRUCCIONES ESPECIALES

NOP

No efectúa operación alguna.

LOCK

Es un prefijo de algunas instrucciones que, al ser ejecutadas, activan la patilla de salida LOCK# del procesador, impidiendo la cesión del *bus* hasta la finalización de la instrucción con LOCK.

17.14- INSTRUCCIONES MULTISEGMENTO

Implican la utilización de más de un segmento por procedimiento.

CALL

Llamada a una rutina o procedimiento.

RET

Retorno de un procedimiento.

INT

Llamada a un programa de manejo de una interrupción.

INTO

Llamada a la entrada 4 de la tabla de interrupciones, si OF = 1.

IRET

Retorno de un programa de interrupción.

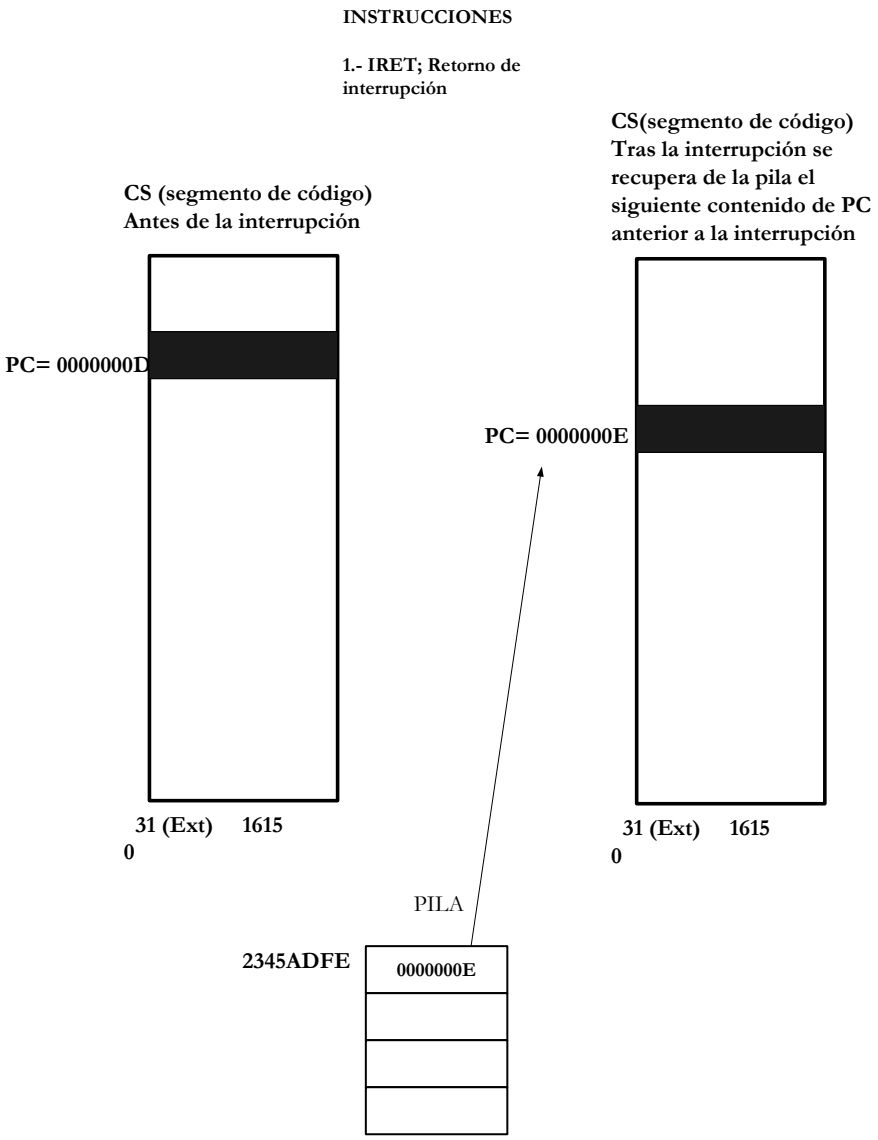


Figura 17.24 Recuperamos de la pila el valor de PC para continuar nuestro programa, después actualizaremos ESI

LDS

Carga simultáneamente el registro DS y otro general con seis bytes de memoria.

LES

Igual que la anterior, pero cargando el registro ES.

JMP

Salto a un segmento de código.

17.14.1- NUEVAS INSTRUCCIONES MULTISEGMENTO

LFS - LGS - LSS

Igual que LDS y LES, pero afectando a los registros FS, GS y SS.

MOV- POP- PUSH

Permiten realizar transferencias de registros de segmentos a otros registros o a la pila.

17.15- INSTRUCCIONES DEL SISTEMA OPERATIVO

ARPL

Comprueba el RPL del primer operando con el del segundo. Si es menor, Z = 1 e iguala los RPL al valor del segundo. (Explicado en el Tema 12 (Puertas de Llamada))

CLTS

Pone a 0 el bit TS de la MSW situada en CR0.

TS(bit de conmutacion de tareas)

PE	CD	NW	RESERVADO	AM	__	WP	RESERVADO	NE	ET	1	EM	MP	PE

31 0

REALIZAMOS LA INSTRUCCIÓN

……. CLTS; …….

TS(bit de conmutacion de tareas)

PE	CD	NW	RESERVADO	AM	__	WP	RESERVADO	NE	ET	0	EM	MP	PE

31 0

Figura 17.25. Representación de EFLAGS tras la realización de la operación CLTS.

HLT

Detiene la ejecución de un programa.

LAR

Carga en el primer operando los derechos de acceso del descriptor al que hace referencia el segundo operando.

LGDT

Carga el GDTR desde una posición de memoria.

LIDT

Carga el IDTR desde una posición de memoria.

LLDT

Carga el registro LDTR.

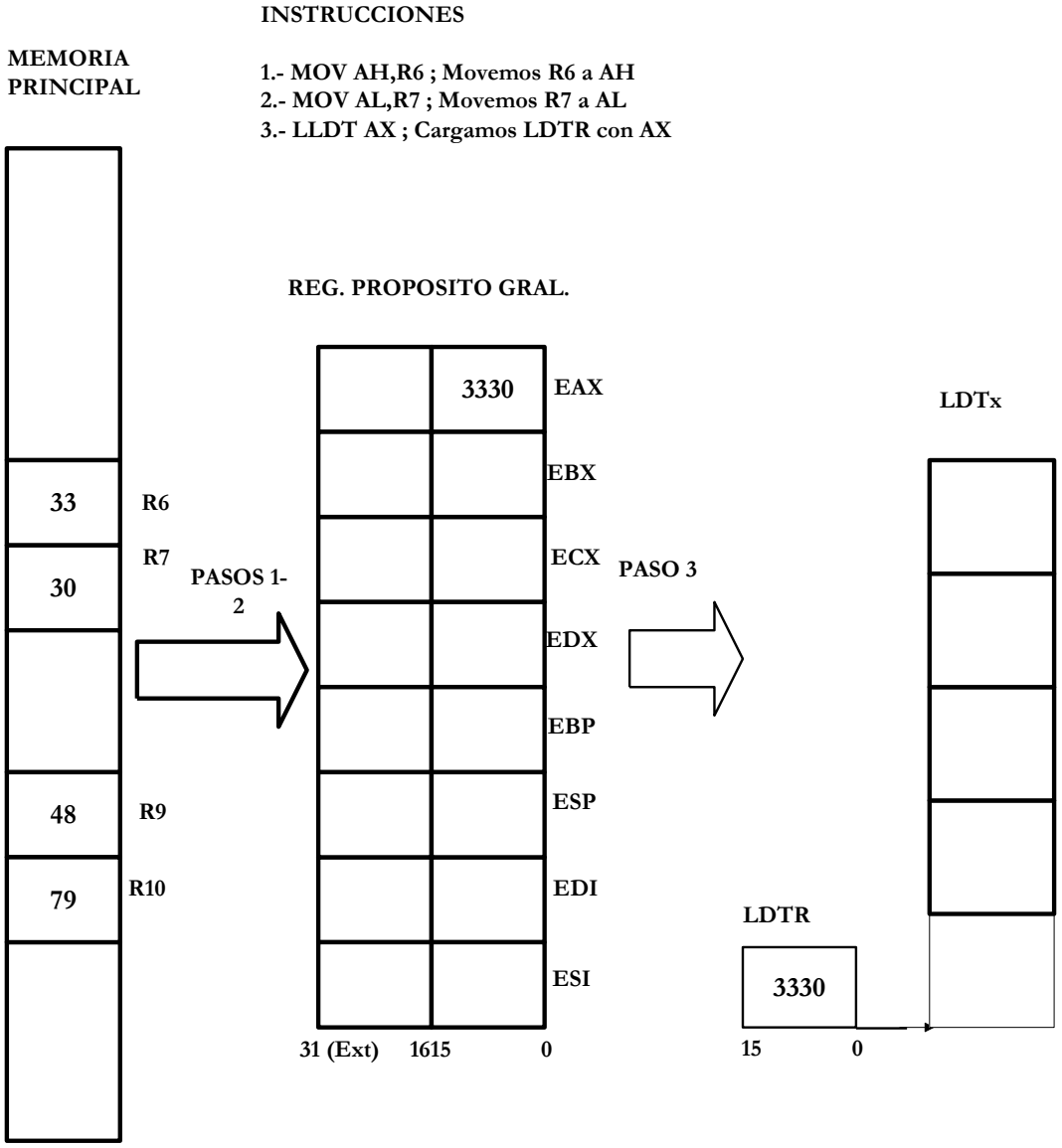


Figura 17.26. Carga de LLDT desde un registro de memoria previamente cargado en AX.

LMSW

Carga a MSW.

LSL

Carga el primer operando con el límite del segmento especificado por el segundo operando.

LTR

Carga el registro TR.

SGDT - SIDT - SLDT- SMSW - STR

Almacenan en la memoria a los registros GDTR, IDTR, LDTR, MSW y TR.

VERR

Verifica si puede ser leído el segmento definido por el selector que se proporciona en el operando de la instrucción. En caso afirmativo, Z = 1.

VERW

Verifica si puede ser escrito un segmento.

17.16- INSTRUCCIONES PARA EL COPROCESADOR

ESC

Precede a las instrucciones que debe procesar el coprocesador matemático. Indica al Pentium que lo que sigue a este prefijo lo envíe al coprocesador.

WAIT

Se detiene el Pentium hasta que la patilla BUSY# se desactive. Así el procesador espera al coprocesador.

17.17.NUEVAS INSTRUCCIONES DEL MICROPROCESADOR PENTIUM

CMPXCHG8B reg, mem64 (Compare and Exchange 8 Bytes)

Compara el valor de 64 bits ubicado en EDX:EAX con un valor de 64 bits situado en memoria. Si son iguales, el valor en memoria se reemplaza por el contenido de ECX:EBX y el indicador ZF se pone a uno. En caso contrario, el valor en memoria se carga en EDX:EAX y el indicador ZF se pone a cero.

CPUID (CPU Identification)

Le informa al software acerca del modelo de microprocesador en que está ejecutando. Un valor cargado en EAX antes de ejecutar esta instrucción indica qué información deberá retornar CPUID. Si EAX = 0, se cargará en dicho registro el máximo valor de EAX que se podrá utilizar en CPUID (para el Pentium este valor es 1). Además, en la salida aparece la cadena de identificación del fabricante contenido en EBX, ECX y EDX. EBX contiene los primeros cuatro caracteres, EDX los siguientes cuatro, y ECX los últimos cuatro. Para los procesadores Intel la cadena es "GenuineIntel".

Luego de la ejecución de CPUID con EAX = 1, EAX[3:0] contiene la identificación de la revisión del microprocesador, EAX[7:4] contiene el modelo (el primer modelo está indicado como 0001b) y EAX[11:8] contiene la familia (5 para el Pentium). EAX[31:12], EBX y ECX están reservados. El procesador pone el registro de características en EDX a 1BFh, indicando las características que soporta el Pentium. Un bit puesto a uno indica que esa característica está soportada. La instrucción no afecta los indicadores.

RDMSR (Read from Model-Specific Register)

El valor en ECX especifica uno de los registros de 64 bits específicos del modelo del procesador. El contenido de ese registro se carga en EDX:EAX. EDX se carga con los 32 bits más significativos, mientras que EAX se carga con los 32 bits menos significativos.

RDTSC (Read from Time Stamp Counter)

Copia el contenido del contador de tiempo (TSC) en EDX:EAX (el Pentium mantiene un contador de 64 bits que se incrementa por cada ciclo de reloj). Cuando el nivel de privilegio actual es cero el estado del bit TSD en el registro de control CR4 no afecta la operación de esta instrucción. En los anillos 1, 2 ó 3, el TSC se puede leer sólo si el bit TSD de CR4 vale cero.

INSTRUCCIONES

1.- RDTSC;Carga el contenido del contador de tiempo, ese contador se autoincrementa por el Pentium

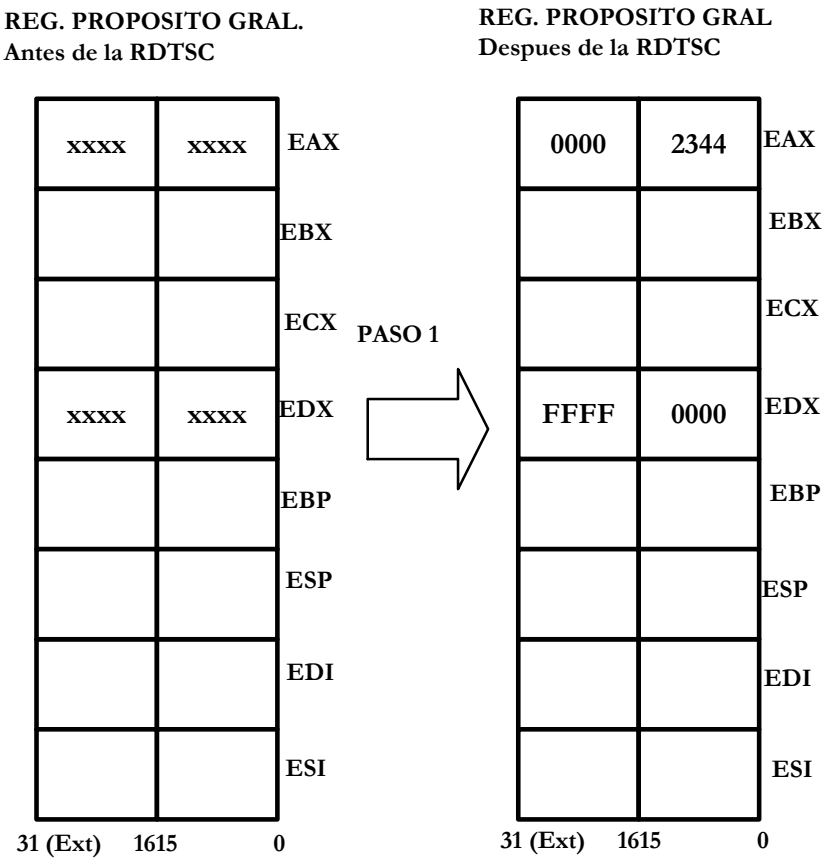


Figura 17.27. Carga del contador del Pentium en los registros de propósito general EAD:EDX

RSM (Resume from System Management Mode)

El estado del procesador se restaura utilizando la copia que se creó al entrar al modo de manejo del sistema (SMM). Sin embargo, los contenidos de los registros específicos del modelo no se afectan. El procesador sale del SMM y retorna el control a la aplicación o sistema operativo interrumpido. Si el procesador detecta alguna información inválida, entra en el estado de apagado (shutdown).

WRMSR (Write to Model-Specific Register)

El valor en ECX especifica uno de los registros de 64 bits específicos del modelo del procesador. El contenido de EDX:EAX se carga en ese registro. EDX debe contener los 32 bits más significativos, mientras que EAX debe contener los 32 bits menos significativos.