Arquitectura IA-32

Antecedentes

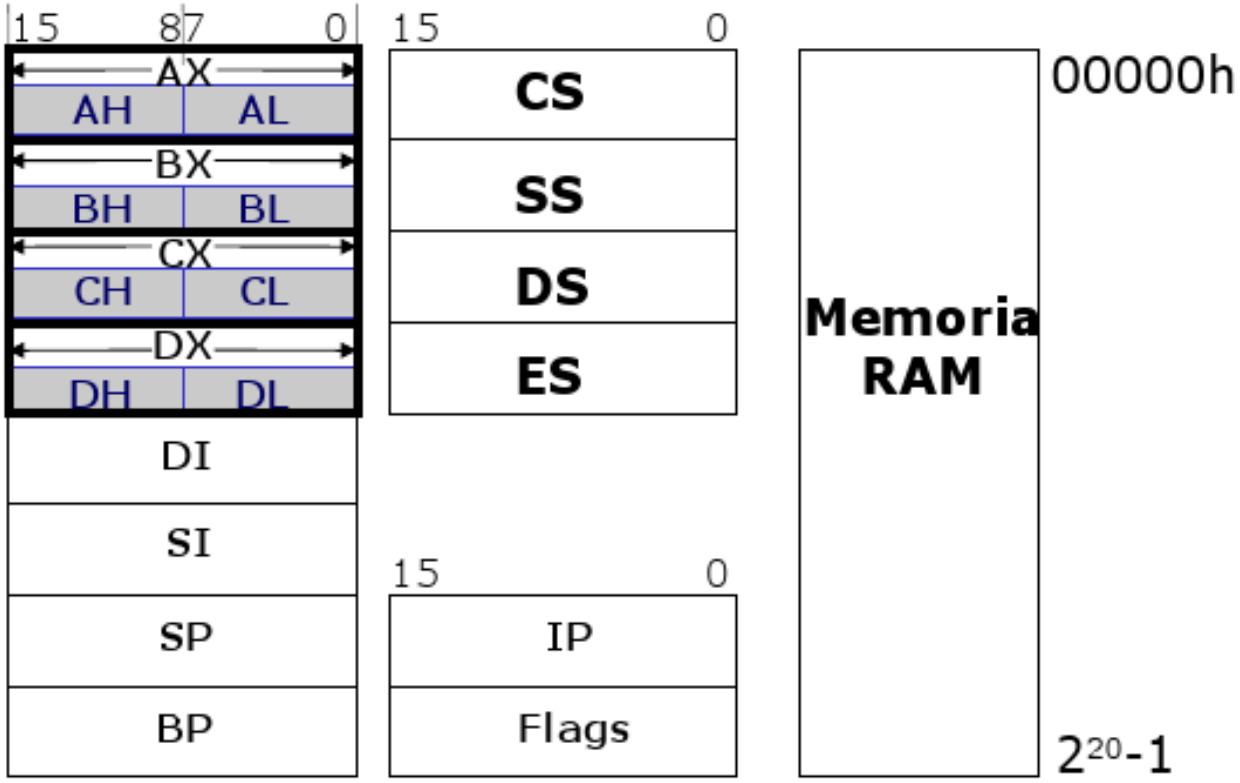
- Año 1978: Se lanza la familia iaPx86
  - Procesador 8086: 16 bits
  - 8 meses después: Procesador 8088. Ídem 8086 pero bus externo = 8 bits
    - Compromiso público de compatibilidad ascendente.
    - Procesador base de la primer IBM PC
- Año 1982: Procesador 80286
  - Arquitectura de 16 bits
  - Capacidad de multitasking
  - Sin suficientes recursos de hardware para hostear un UNIX clásico
- Año 1985: Procesador 80386. Presentación de la Arquitectura IA-32
- Hasta el presente mas de 15 modelos de procesadores compatibles y 7 microarquitecturas diferentes.

Modos de Operación

- Modo Real
- Modo Protegido
- Modo Virtual 8086
- Modo Mantenimiento del Sistema (SMM)
- Modo IA-32e (Intel 64 Architecture)
  - Modo Compatibilidad (sub modo del modo IA-32e).
  - Modo 64 bits (sub modo del modo IA-32e)

Modo Real:
Entorno de ejecución en 16 bits

Cualquier procesador IA-32 arranca en este modo de operación.
Presenta de manera exacta el entorno de ejecución de un 8086 (compatibilidad).



No obstante, el Modelo de Registros Generales de 32 bits está disponible en Modo Real

Administración de memoria - Generalidades

- Existen dos formas de organizar la memoria de un computador
  - En páginas
  - En segmentos

Administración de la memoria - Paginación

- Las páginas tienen tamaño fijo. Ej: 4 Kbytes
- Son contiguas (no se solapan ni son disjuntas)
- Ventajas.
  - Administración simple de la memoria (tamaño fijo)
- Desventajas
  - Si requiero memoria para una variable de 1 byte el sistema me asigna una página de 4 K completa
  - Si se requiere reubicar código el algoritmo, en principio, es complejo.

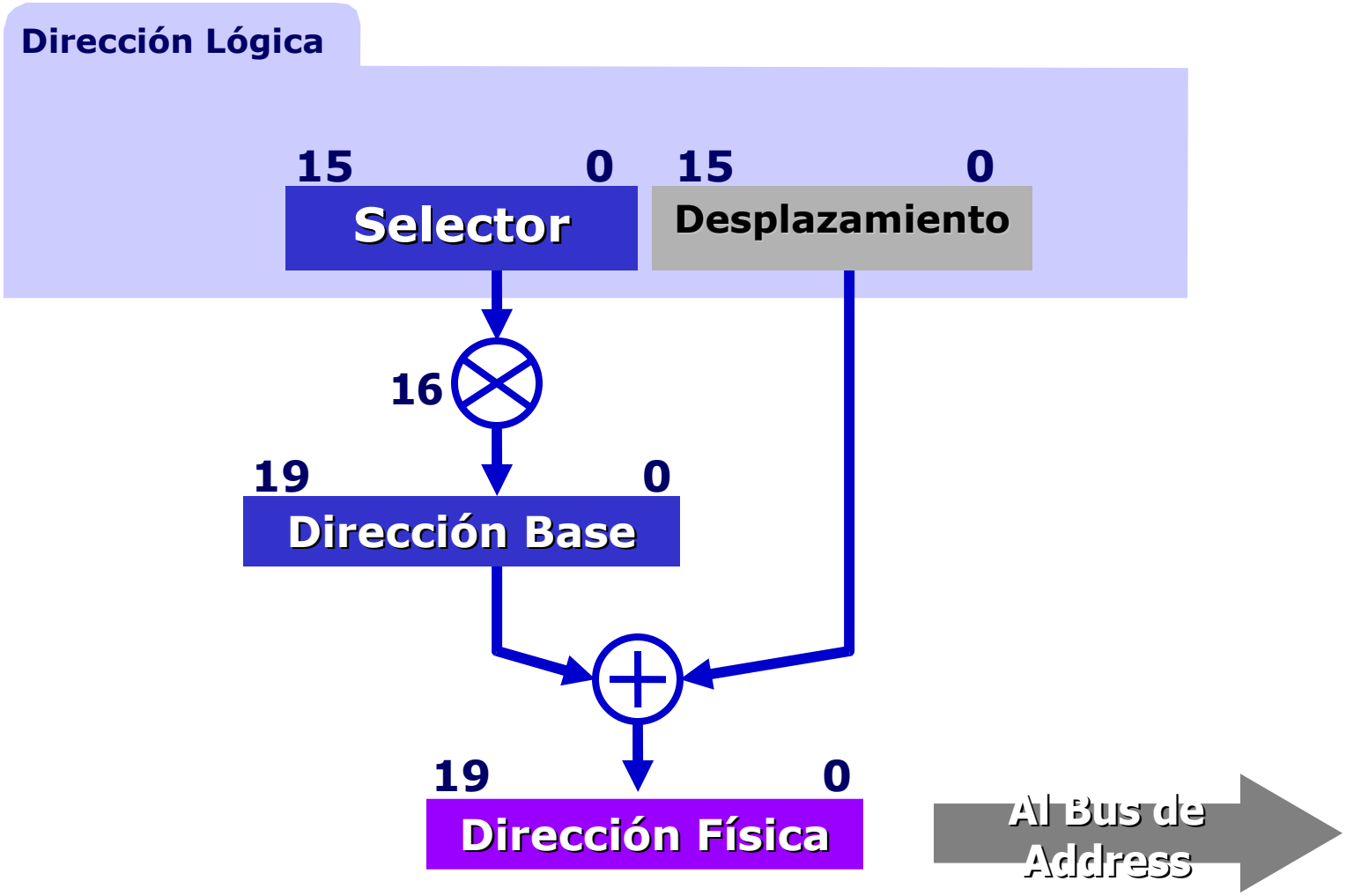
Administración de la memoria - Segmentación

- Los segmentos son de tamaño variable
- No necesariamente son contiguos (se pueden solapar o estar disjuntos)
- Se cuenta con un registro de desplazamiento que haciendo referencia al principio del segmento (dirección base del segmento) permite desplazarse a lo largo del mismo
  - Un Contador de Programa a través de un segmento de código.
  - Un Stack Pointer a través de un Segmento de pila
- Ventajas.
  - Manteniendo fijo el valor del segmento el movimiento a través del mismo se realiza solo con el registro de desplazamiento.
  - Reubicación en memoria simple de código, pilas y bloques de datos.
  - Si se necesitan pocos bytes para una variable pido al sistema un segmento de por ejemplo 16 bytes. Se desperdicia mucho menos memoria
- Desventajas
  - Es engorroso administrar memoria con bloques de tamaño variable

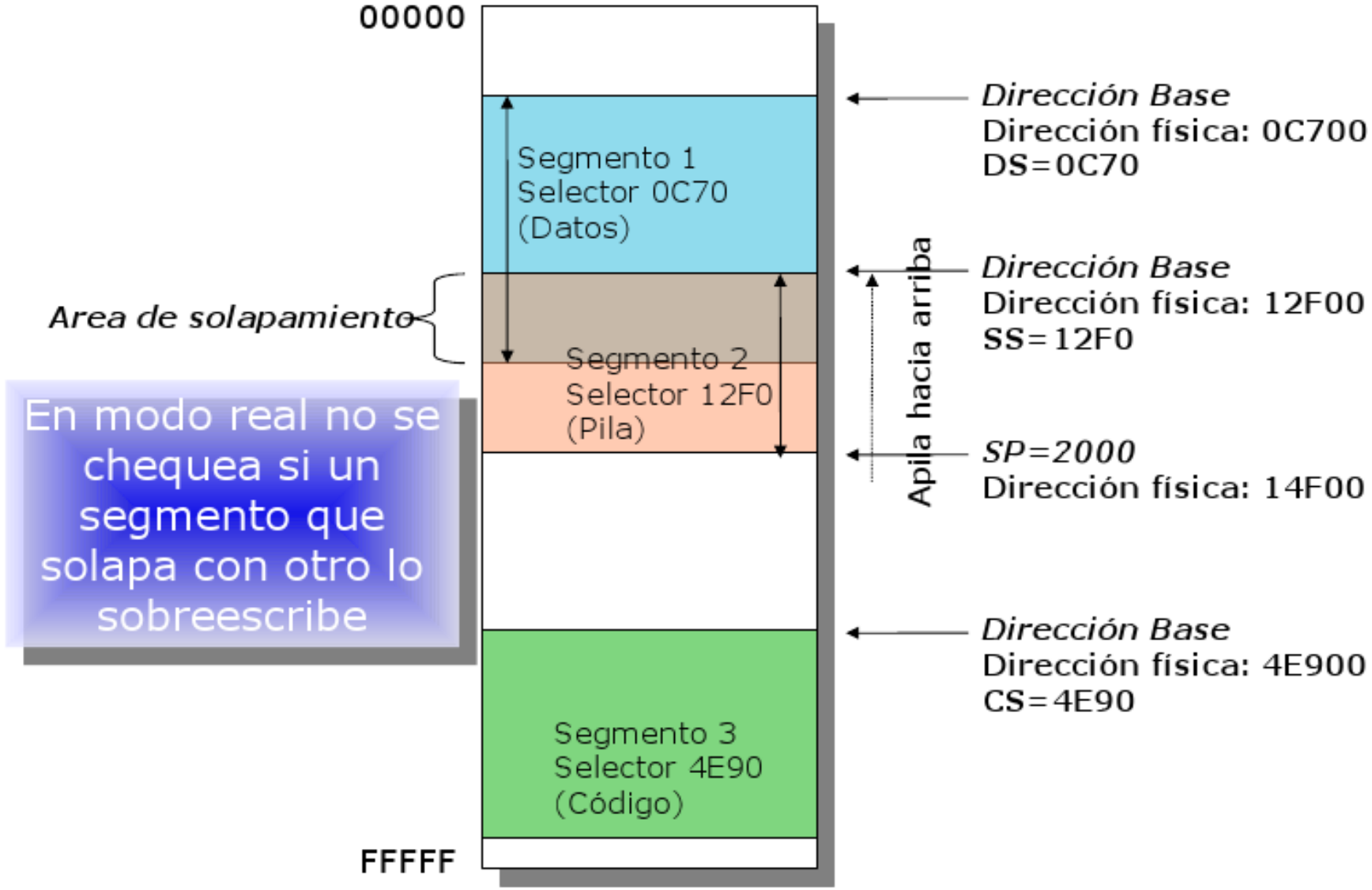
Administración de Memoria en x86

- Por diversos motivos que en su momento tuvieron sentido, Intel definió organizar el espacio de direccionamiento de la Familia iAPx86 en segmentos
- El compromiso de compatibilidad ató a los siguientes procesadores a mantener este esquema.
- Inicialmente para los procesadores 8086 y 8088 se definieron 4 registros de segmento para almacenar hasta 4 selectores de segmento.
- Al trabajar con registros de 16 bits los segmentos tienen a lo sumo 64K de tamaño.

Direccionamiento en Modo Real



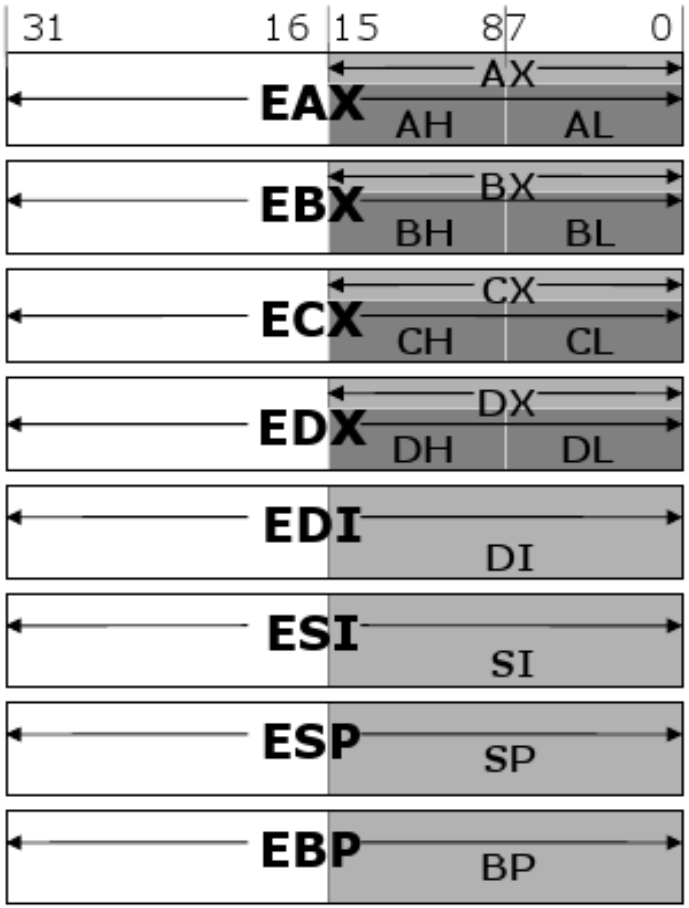
Ejemplo de Segmentación en x86



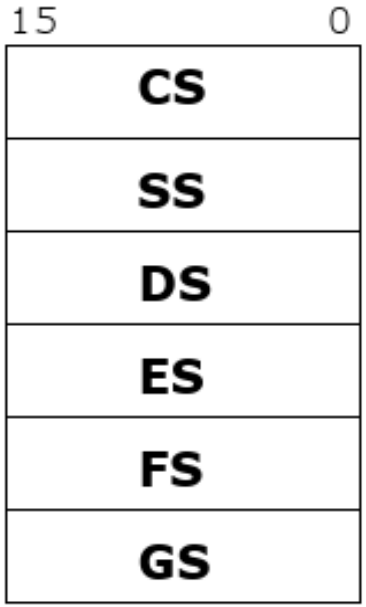
En modo real no se chequea si un segmento que solapa con otro lo sobreescribe

Entorno Básico de ejecución en 32 bits

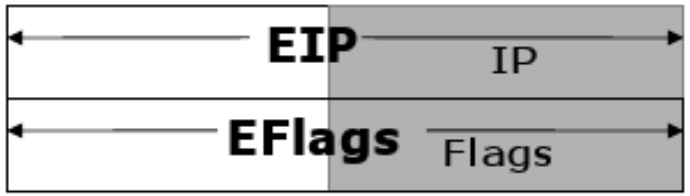
Registros de Propósito General



Registros de Segmento



Registros de Control y Estado



Reglas de Selección de Segmentos

Tipo de referencia	Registro utilizado	Segmento utilizado	Regla por default
Instrucciones	CS	Segmento de Código	En todos los instrucción Fetch
Pila	SS	Segmento de Pila	Todos los push y pop, y cualquier referencia a memoria que use ESP o EBP como registros Base
Datos Locales	DS	Segmento de Datos	Todas las referencias a memoria excepto las correspondientes a la pila y a Segmentos destino en operaciones de cadenas (strings)
Cadenas Destino	ES	Segmento de Datos apuntado con el ES	Segmentos de dato destino en operaciones de Cadenas de datos (strings)

Modo Protegido

- Modo de trabajo no default, pero natural para este tipo de procesadores
- Se setea por software desde Modo Real
- Implementa una alta capacidad de direccionamiento de memoria
  - 4 Gbytes hasta el procesador Pentium y Pentium MMX
  - 64 Gbytes en los procesadores Pentium Pro en adelante (activando por software extensiones de memoria)
- Posee un atributo llamado Virtual 8086 que permite ejecutar un programa desarrollado para un procesador 8086/8088 como una tarea en modo protegido (al principio se lo consideraba un modo de funcionamiento en si mismo).
- Este modo es el eje de la 2da. parte del presente curso, ya que es la base sobre la que se sustenta cualquier sistema operativo moderno.

Modo Mantenimiento del Sistema (SMM)

- Modo de trabajo para realizar operaciones especiales como Manejo de energía o seguridad.
- Introducido por los procesadores 80386SL y 80486SL.
- Se ingresa por medio de:
  - Señal de hardware Pin SMI#
  - Mediante un mensaje SMI recibido desde el APIC (Advanced Programmable Interrupt Controller)
- Cuando ingresa a este estado el procesador salva el contexto de la tarea en ejecución y pasa a ejecutar un bloque de software específico para este modo.
- Cuando termina la ejecución del código SMM, retorna retomando la tarea interrumpida en el punto exacto en que la abandonó.

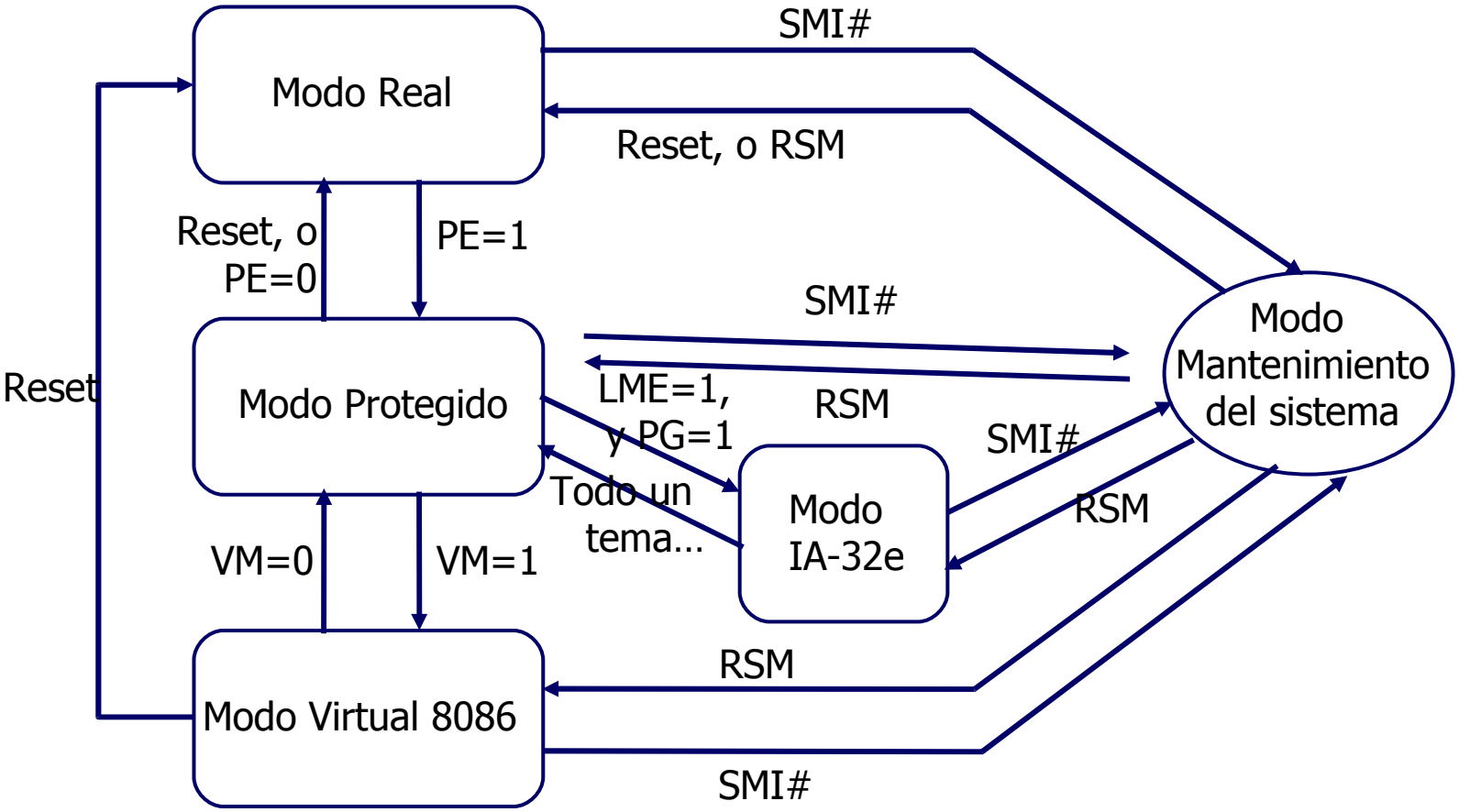
Modo IA-32e - Modo Compatibilidad

- Permite ejecutar cualquier programa desarrollado en una arquitectura IA-32 sin necesidad de recompilar.
- Apto para escribir código de sistemas operativos de 64 bits que permitan migrar el código binario existente sin re compilar
- En el modo compatibilidad el modelo de arquitectura de programación es el mismo que el IA-32
  - Los mismos registros
  - El mismo espacio de direccionamiento de memoria
- No compatible con
  - Tareas que ejecutan con atributo Virtual 8086
  - Tareas manejadas por hardware.

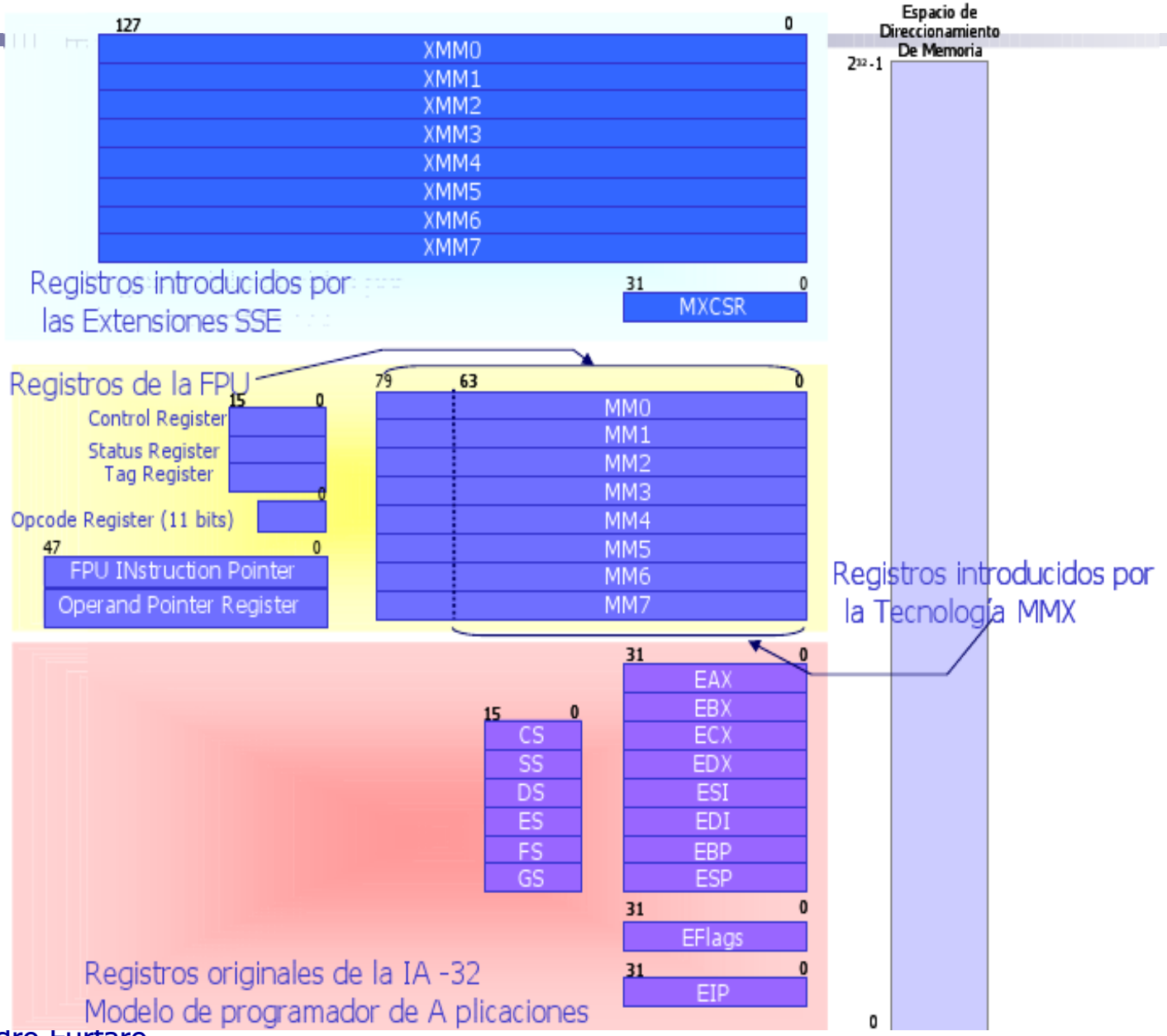
Modo IA-32e - Modo 64 bits

- Se referencia normalmente como modo 64 bits.
- Las aplicaciones acceden a un espacio lineal de direcciones de 64 bits
- Los registros de propósito general se extienden a 64 bits.
- Se agregan nuevos registros SIMD de 128 bits pasando de 8 registros a 16.
- Por default el tamaño de operando es 32 bits
- El prefijo REX permite trabajar con operandos de 64 bits extendiendo las instrucciones existentes a esta capacidad de operandos

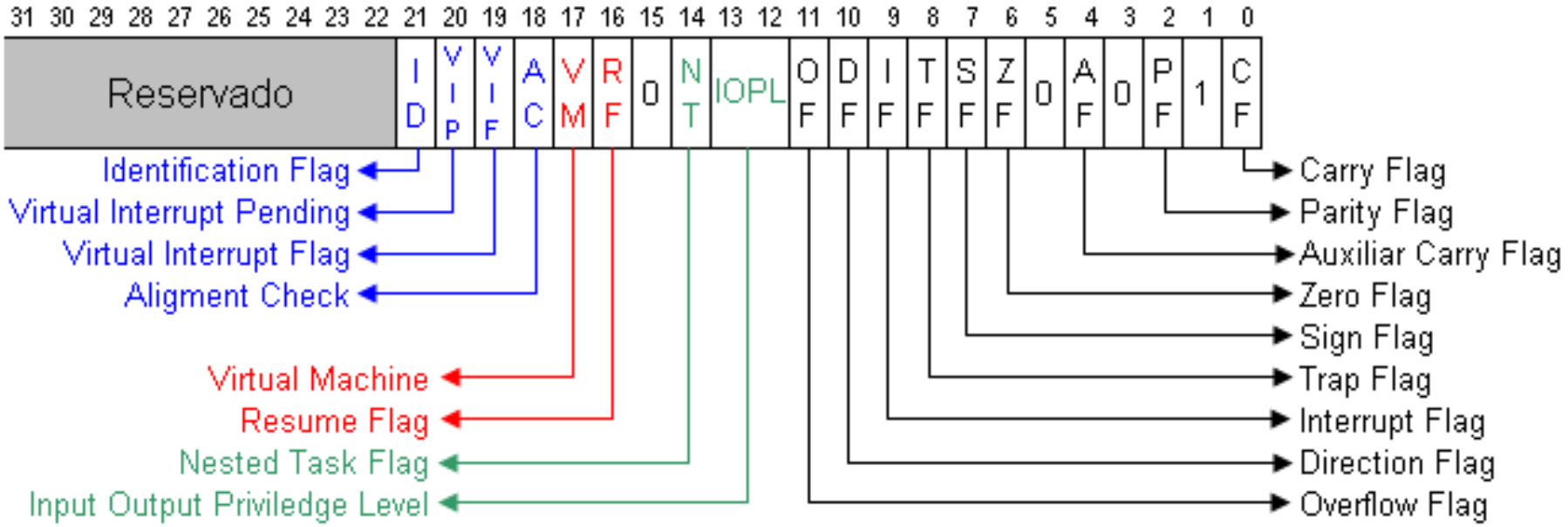
Modos de Funcionamiento



Entorno básico de ejecución IA-32



Entorno Básico de Ejecución en 32 bits

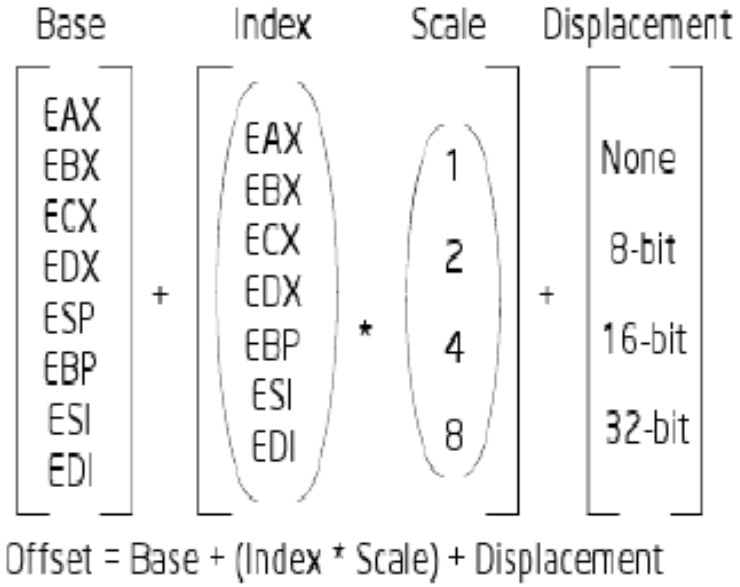


Los Flags originales del 8088 son los de color negro
El 80286 agregó los de color verde
El 386 agregó el rojo
Los Procesadores Pentium agregaron paulatinamente los flags en color azul

Registro EFLAGS

Modos de Direccionamiento

- Implícito
- Inmediato
- Registro
- Desplazamiento
- Base
- Base + desplazamiento
- Indexado
- Indexado + desplazamiento
- Base + Indexado
- Base + Indexado + desplazamiento



Ejemplos de Modo de Direccionamiento

- Implícito

```
clc
```

- Inmediato

```
mov eax,0x20
mov eax,20h
xor ecx,0xFF0F
```

- Registro

```
inc edx
sub eax,edx
```

Ejemplos de Modo de Direccionamiento

- Desplazamiento

```
add [2C00h],ecx
dec [0x7C00]
```

- Base

```
mov  edx,i    ; edx=desplazamiento de i en el
              ; segmento de datos
inc  [edx]    ; incrementamos i ;)
```

Ejemplos de modos de direccionamiento

- Base + Desplazamiento
  - Búsqueda de parámetros en el stack frame

```
push ebp                ;resguardamos ebp
mov  ebp,esp            ;lo apuntamos a la pila
mov  eax,[ebp+12] ;extrae el parámetro que
   ;está 12 bytes hacia el fondo de la pila.
```

  - Base + Desplazamiento (elementos de tamaño no standard)

```
mov  ecx, size_tabla
mov  ebx, tabla   ;Base de la tabla en ebx
mas:
   and  [ebx + tamaño_registro],0xFE
   add  ebx,tamaño_registro
   loop mas
```

Ejemplos de Modos de direccionamiento

- Indexado

```
   mov  ecx, size_tabla
   mov  esi, tabla
mas:
   and  [esi],0xFE
   inc  esi
   loop   mas
```

- Indexado con escala

```
   mov  ecx, size_tabla
   mov  esi, tabla   ;elementos de 4 bytes
mas:
   and  [esi*4],0xFE;and a un byte cada 4
   inc  esi          ;siguiente elemento
   loop   mas
```

Ejemplos de Modos de direccionamiento

- Indexado + desplazamiento. Otra forma del ejemplo anterior

```
    mov  ecx, size_tabla
    xor  esi, esi
mas:
    and  [esi*4 + tabla],0xFE
    inc  esi
    loop   mas
```

Ejemplos de Modos de direccionamiento

- Base + Índice con escala

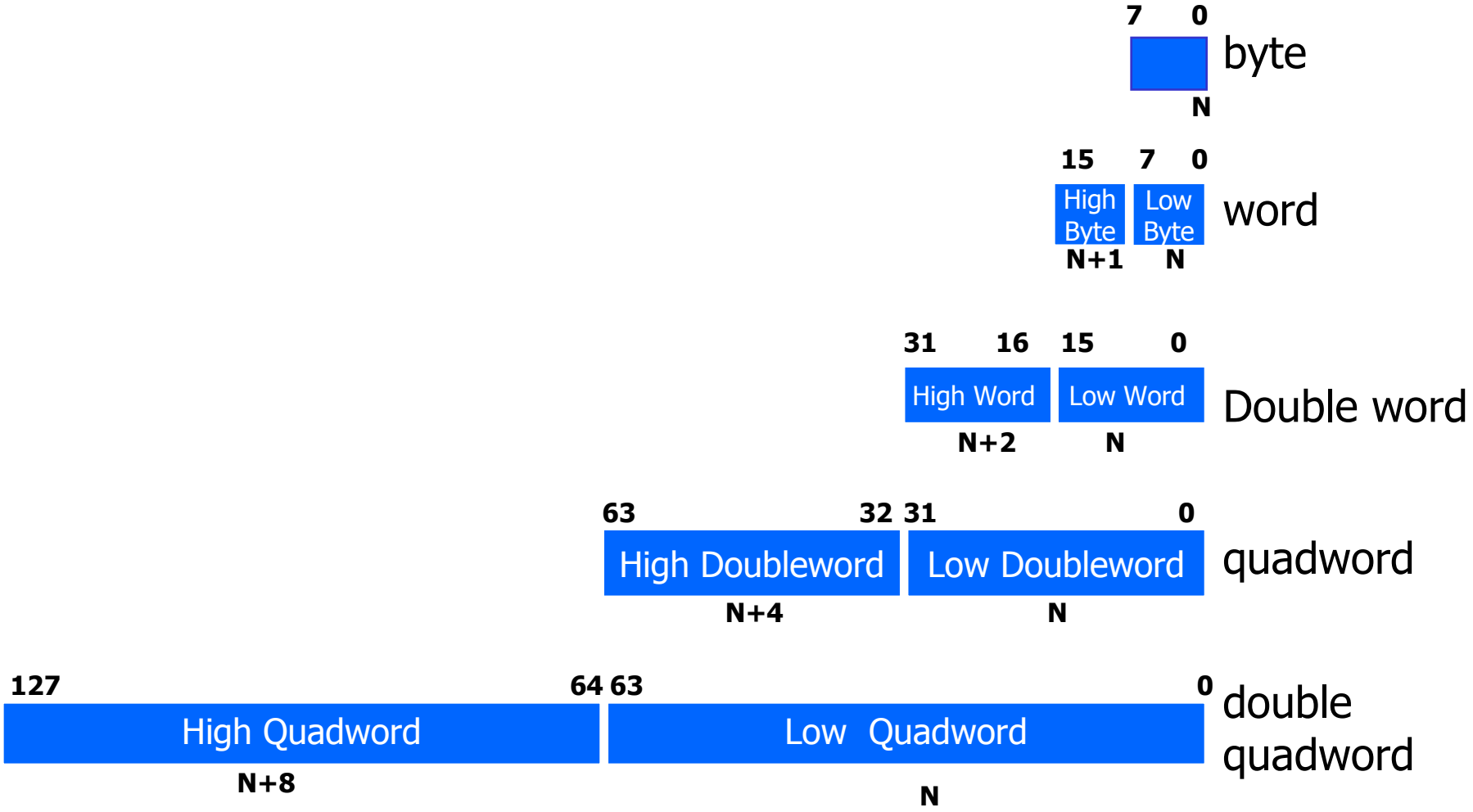
```
   mov  ebx,0xB8000; Base buffer video.
col:
   mov  ecx, size_row ;ecx = bytes por fila
   xor  edi, edi
row:
   mov  [ebx, edi*2],70h ; Video Inverso
   inc  edi  ;siguiente elemento de 2 bytes
   loop row  ;si CX = 0 completó fila
   add  ebx, size_row*2  ;siguiente fila
   cmp  ebx, 0x000B9000; fin del buffer?
   jle  col
```

Ejemplos de Modos de direccionamiento

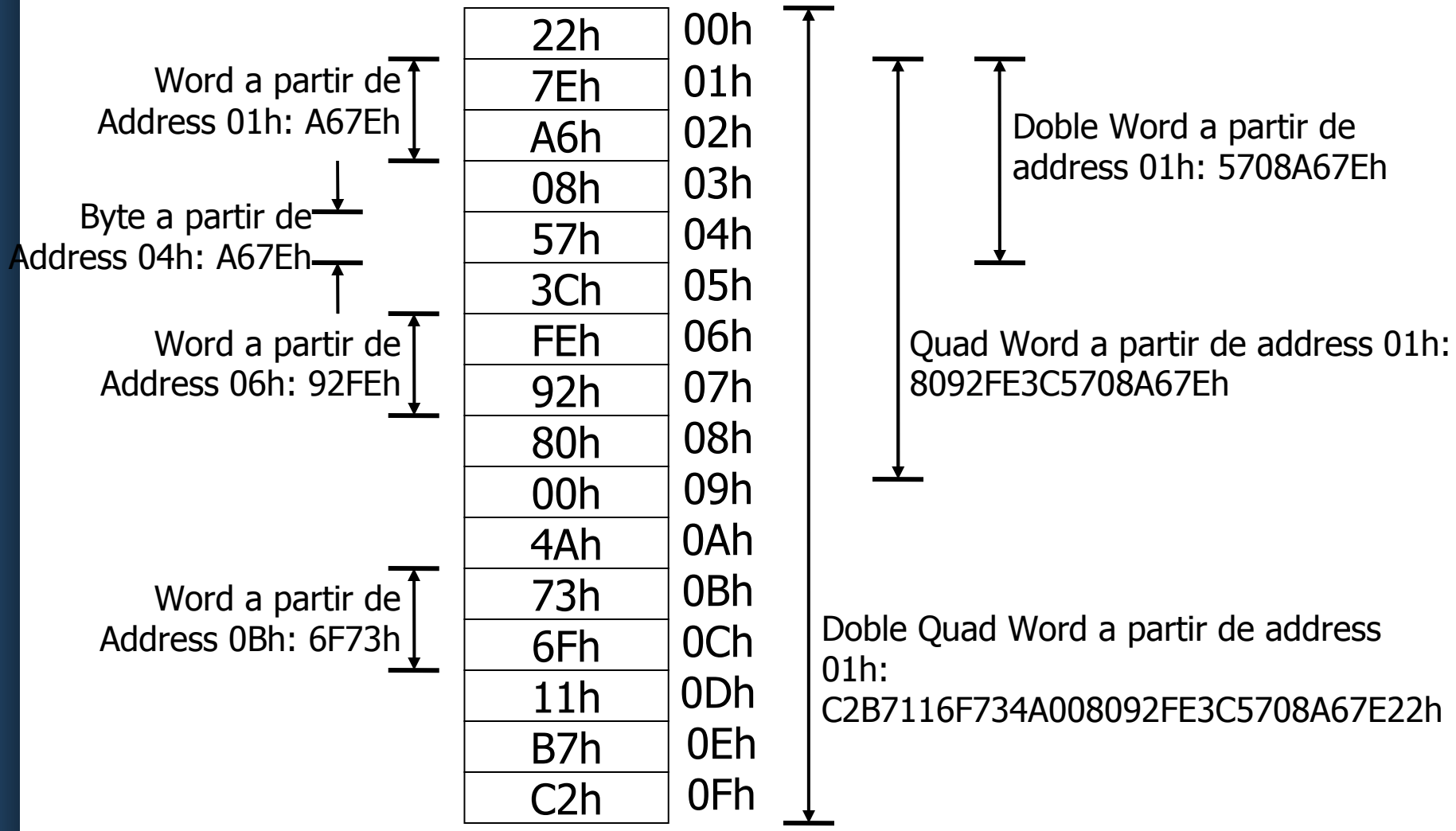
- Base + Índice + Desplazamiento

```
    xor  ebx,ebx        ;ebx = 0
col:
    xor  edi,edi        ;edi =0
    mov  ecx,size_row ;ecx = bytes por fila
row:
    mov  [ebx + edi*2 + 0xB8000],70h; Inverso
    inc  edi  ;siguiente elemento de 2 bytes
    loop row  ;si CX = 0 completó fila
    add  ebx,size_row*2 ;siguiente fila
    cmp  ebx,0x1000   ;fin del buffer?
    jle  col
```

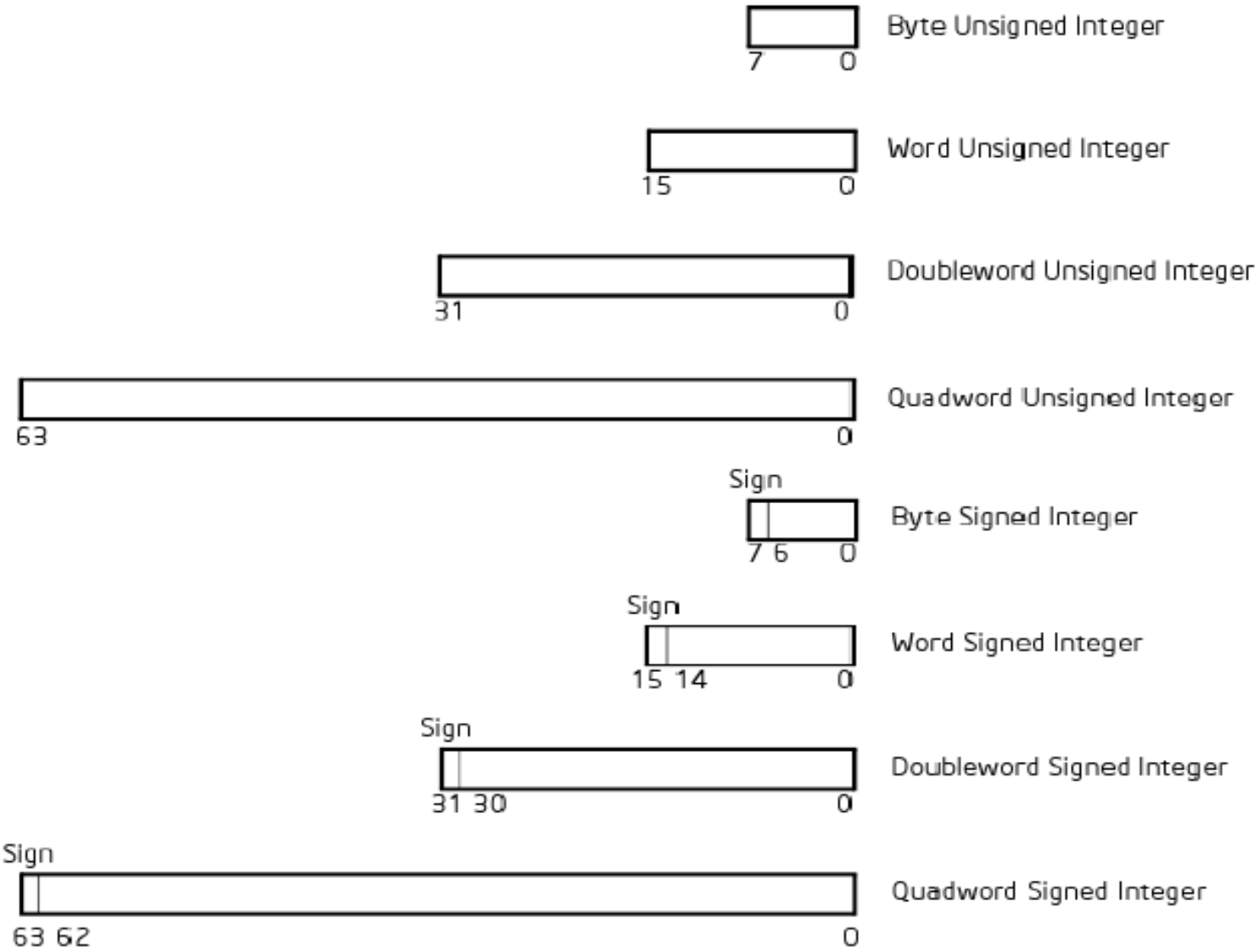
Tipos de datos fundamentales



Datos en Memoria



Números Enteros



Set de Instrucciones

- No puede haber procesador en el mundo dueño de un set de instrucciones mas “CISC”.
- Lo mas conveniente es explicar su funcionamiento a medida que se utilizan.
- Los grupos de instrucciones mas relevantes para el alcance de nuestro curso son:
  - General purpose
  - x87 FPU
  - x87 FPU and SIMD state management
  - Intel MMX technology
  - SSE extensions
  - SSE2 extensions
  - SSE3 extensions

Llamadas a subrutinas

- Se realizan mediante la instrucción CALL
- Cuando finaliza la subrutina se ejecuta RET para volver al flujo de programa desde el que se realizó la llamada vía CALL.
- El stack es un espacio lineal de direcciones.
- Las operaciones son con words o dwords dependiendo del tipo de segmento definido (16 o 32 bits)
- El stack debe estar alineado a word o dword según el tipo de segmento.

Programando con las instrucciones de Propósito General

- Data transfer
- Binary arithmetic
- Decimal arithmetic
- Logical
- Shift y rotate
- Bit y byte
- Control transfer
- String
- I/O
- Flag control
- Segment register
- Misceláneas

Estas instrucciones se enumeran sucintamente en las siguientes transparencias. Queda como tarea la lectura de los detalles de implementación en los Vol 2A y 2B de los manuales del procesador

Movimiento de Datos

- Movimientos (MOV)
- Movimiento Condicional (CMOV)
- Exchange
  - XCHG
  - BSWAP
  - CMPXCHG
  - CMPXCHG8B
- Manipulación de la pila
  - PUSH y POP
  - PUSHA y POPA
  - ENTER y LEAVE
- Conversión de tipos (CBW, CWD, CWDE, CD

Aritmética Binaria

- Suma y Resta. (ADD, ADC, SUB, SBB)
- Incremento y decremento (INC, DEC)
- Comparación (CMP, NEG)
- Multiplicación y División (MUL, DIV, IMUL, IDIV)

Aritmética Decimal

- Ajustes decimales para nros. BCD empaquetados
  - DAA
  - DAS
- Ajustes decimales para nros. BCD Desempaquetados
  - AAA
  - AAS
  - AAM
  - AAD

Lógicas

- AND
- OR
- XOR

Shift y Rotate

- SHR
- SAR
- SHL/SAL
- SHRD
- SHLD
- ROL
- ROR
- RAL
- RAR

Bits y Bytes

- BT, BTS, BTR, BTC
- BSF, BSR
- TEST
- SET

Transferencias

- Incondicionales
  - JMP, CALL, RET, INT, IRET
- Condicionales
  - Jcc (cc = condición)
  - LOOP
  - JECXZ
  - JCXZ
  - INTO

Strings

- MOVS
- SCA
- STOS
- LODS
- CMPS

} Sufijo
BWoD
De acuerdo con el tipo de dato involucrado

- Prefijos de repetición
  - REP
  - REPZ, REPE
  - REPNZ, REPNE

Entrada / Salida

- IN
- OUT
- INS
- OUTS

} Sufijo
BWoD
De acuerdo con el tipo de dato involucrado

Control de Flags

- Carry
  - STC, CLC, CMC
- Direction Flag
  - CLD, STD
- Registro EFLAGS
  - LAHF, SAHF, PUSHF, POPF, PUSHFD, POPFD
- Flag de Interrupciones
  - CLI, STI

Segment Register

- MOV y POP
  - Cuando el operando destino es ES, DS, SS, FS, o GS.
  - Nunca CS
- JMP y CALL far.
- RET far
- INT, IRET
- BOUND
- Carga de punteros far
  - LDS, LSS, LES, LGS, LFS

Misceláneas

- LEA
- XLAT, XLATB
- CPUID
- NOP
- UD2

Referencias

Intel® 64 and IA-32 Architecture Software Developer's Manual Volume 1: Basic Architecture.

Intel® 64 and IA-32 Architecture Software Developer's Manual Volume 2A: Instruction Set Reference A-M

Intel® 64 and IA-32 Architecture Software Developer's Manual Volume 2B:

Instruction Set reference N-Z