Procesadores de 32 bits

Modelo de ejecución SIMD en Procesadores IA-32

Procesadores de Señales Digitales

- Manejan un dato específico: Señales Digitalizadas
- Su criterio de diseño tiene esta impronta
- ¿Cuáles son las diferencias?
  - Los procesadores comunes Manipulan datos
    - Operaciones comunes A->B, If (A< 20) then {…..}
    - Aplicaciones: Querys a Bases de Datos, Procesamiento de texto, manejo operadores booleanos, etc.
  - Los procesadores de Señales son máquinas de cálculo repetitivo y básico
    - Operaciones comunes A + B = C, o bien A * B = C
    - Aplicaciones: Procesamiento de audio y video (Cada vez mas frecuentemente en tiempo real)
    - Un tipo muy común de cálculo tiene la forma:

$$y[n] = a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + a_3 x[n-3] + a_4 x[n-4] + \cdots$$

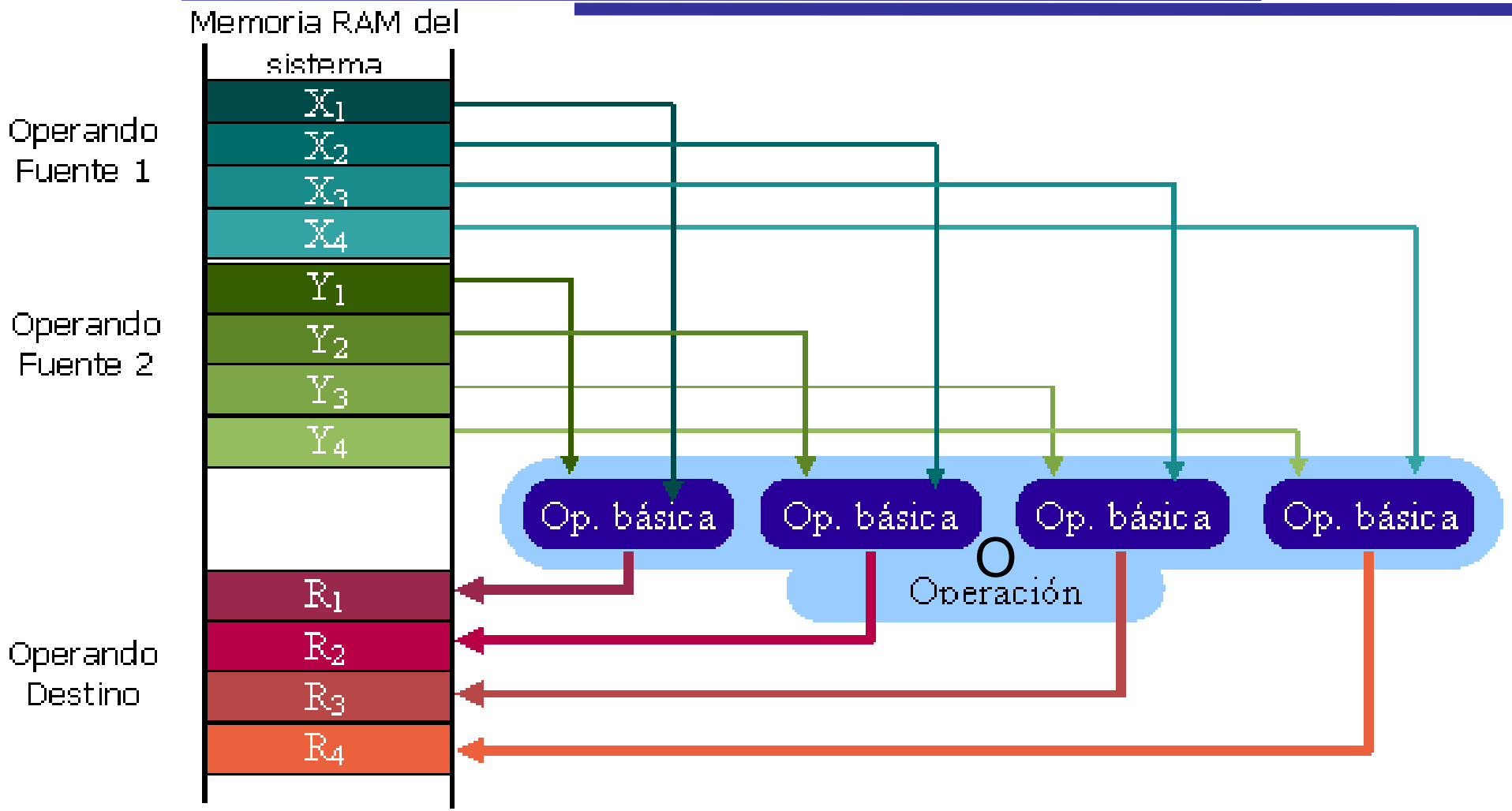
    - En consecuencia, necesitan una arquitectura muy específica para optimizar su funcionamiento

Single Instruction Multiple Data SIMD

- Permite efectuar varias operaciones aritméticas de cálculo en una sola instrucción
- Aplicaciones:
  - multimedia (audio, gráficos y video),
  - comunicaciones.
- Base conceptual del modelo SIMD
  - La mayor parte del trabajo en los algoritmos de procesamiento de señales e imágenes, consiste en ejecutar siempre la misma operación sobre una extensa lista de datos relativamente pequeños (en general de 8 o 16 bits).

Autor: Ing. Alejandro Furfaro

Single Instruction Multiple Data SIMD

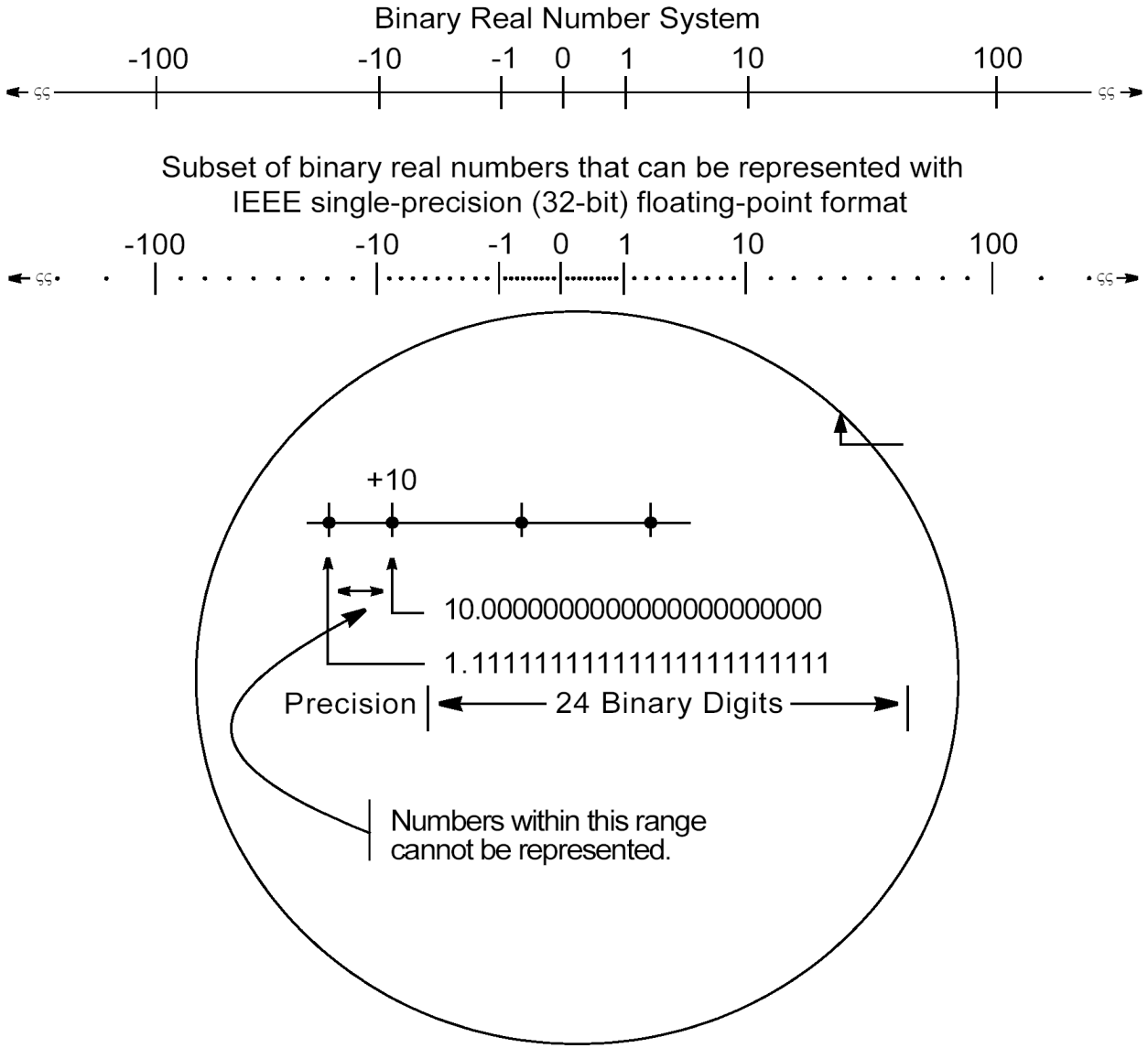


R1 = X1 O Y1 R2 = X2 O Y2 R3 = X3 O Y3 R4 = X4 O Y4

Autor: Ing. Alejandro Furfaro

Tipos de datos: Números reales

- El rango de los números reales comprende desde -∞ hasta +∞.
- Los registros de un procesador tienen resolución finita.
- Por lo tanto un computador solo puede representar un sub conjunto de $\mathscr{R}$.
- Además, no es solo un tema de magnitud sino de resolución.



Autor: Ing. Alejandro Furfaro

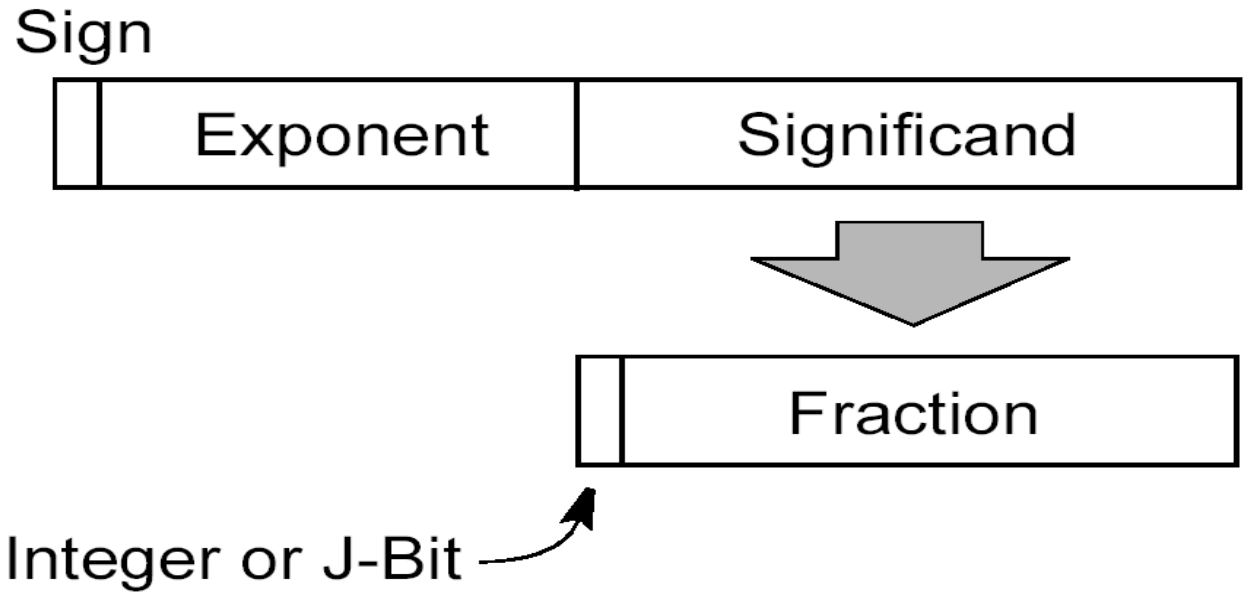
Tipos de datos: Números reales

- Para el caso de los números reales se trabaja en notación científica.
  - Ej: para el número 725,832, la representación es $7.25832\ E_{10}^{2}$, o lo que es igual $7.25832 \cdot 10^{2}$
- El tipo de dato empleado para representar números en notación científica en el mundo binario de los microprocesadores, es la representación en punto flotante.

Autor: Ing. Alejandro Furfaro

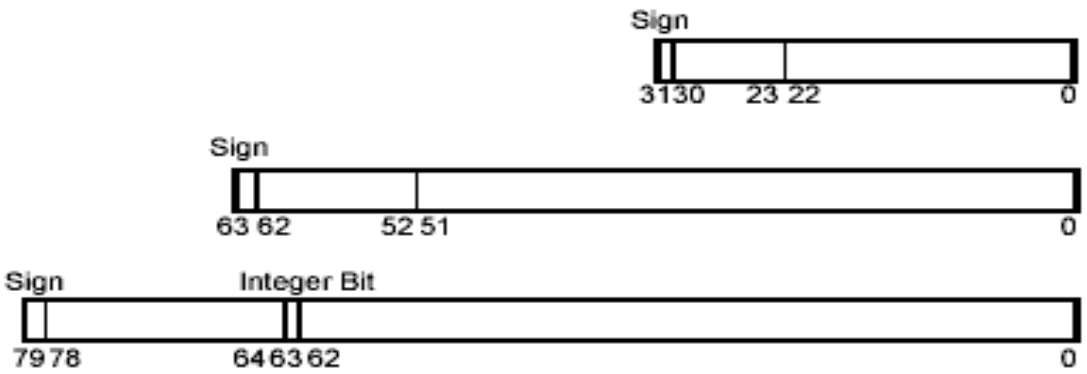
Tipos de datos: Números reales

- Representación de números en punto flotante
  - Un bit de signo,
  - Un valor fraccionario denominado mantisa o significando,
    - Cuenta con una parte entera de 1 bit (J-bit), y el resto de la parte fraccionaria. Sin embrago por lo general el J-bit no se representa sino que por el contrario, es un valor implícito en la mantisa.
  - Y un valor de exponente al cual se debe elevar la base numérica de representación para obtener el valor real.



Tipos de datos: Números reales

- Una de las principales ventajas de operar en punto flotante es que se eliminan los problemas relacionados con las escalas.
- El Standard IEEE 754 para punto flotante binario es el mas ampliamente utilizado. En este Standard se especifican los formatos para 32 bits, 64 bits, y 80-bits.
- En 2008 se introdujeron un formato de 16 bits y el de 80 fue reemplazado por uno de 128 bits (IEEE 754-2008)



Single Precision Floating Point

$$(-1)^s * f * 2^{e-127}$$

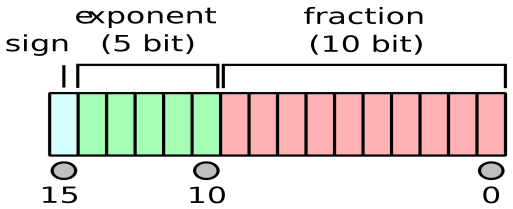
Double Precision Floating Point

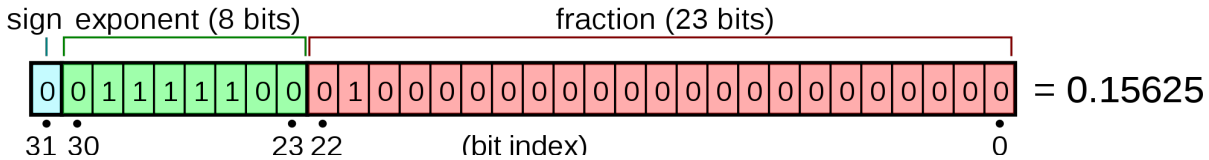
$$(-1)^s * f * 2^{e-1023}$$

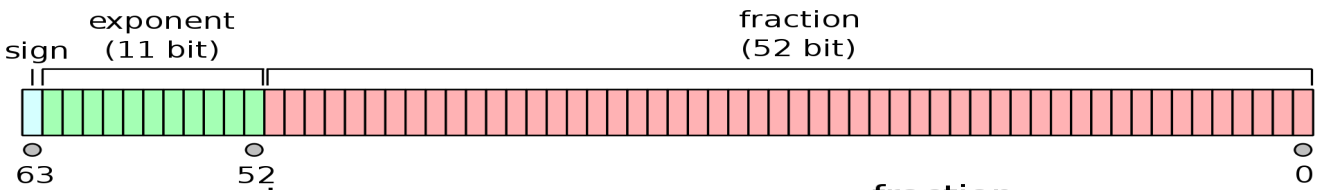
Double Extended Precision Floating Point

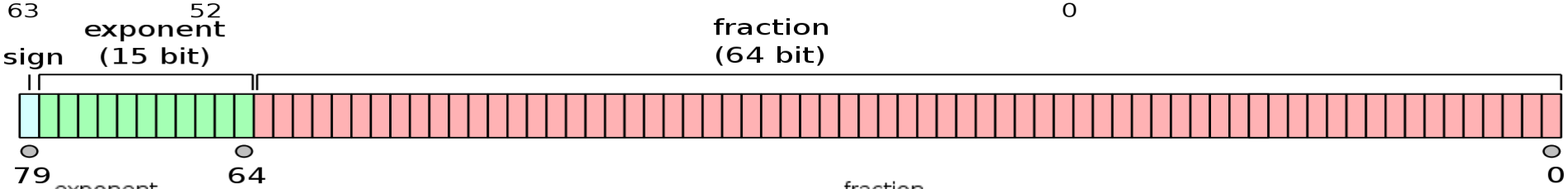
$$(-1)_8^s * f * 2^{e-16383}$$

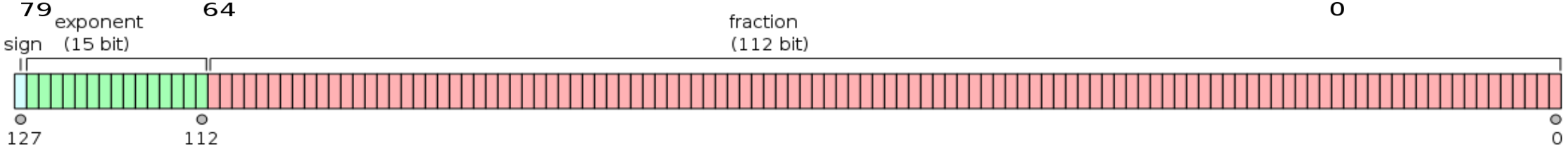
IEEE 754 y 754-2008. Formatos











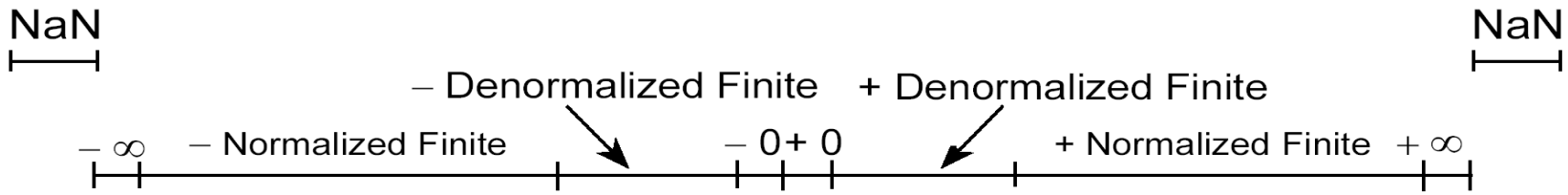
Autor: Ing. Alejandro Furfaro

Codificación de números reales

- Ceros signados
- Números finitos de-normalizados
- Números finitos normalizados
- Infinitos signados
- NaNs (Not a Number)
- Números Indefinidos

Autor: Ing. Alejandro Furfaro

Codificación de números reales



Real Number and NaN Encodings For 32-Bit Floating-Point Format

S	E	Sig[1]			S	E	Sig[1]
1	0	0.000...	– 0	+ 0	0	0	0.000...
1	0	0.XXX...[2]	– Denormalized Finite	+Denormalized Finite	0	0	0.XXX...[2]
1	1...254	1.XXX...	– Normalized Finite	+Normalized Finite	0	1...254	1.XXX...
1	255	1.000...	– ∞	+ ∞	0	255	1.000...
X[3]	255	1.0XX...[2]	SNaN	SNaN	X[3]	255	1.0XX...[2]
X[3]	255	1.1XX...	QNaN	QNaN	X[3]	255	1.1XX...

NOTES:

1. Integer bit of fraction implied for single-precision floating-point format.
2. Fraction must be non-zero.
3. Sign bit ignored.

Autor: Ing. Alejandro Furfaro

Codificación de números reales

- Ceros signados
  - Una operación puede dar +0 o -0 en función del bit de signo.
  - En ambos casos el valor es el mismo.
  - El signo de un resultado cero depende de la operación en sí y del modo de redondeo utilizado.
  - Los ceros signados ayudan a interpretar el intervalo aritmético en el que se ubicaría el resultado si la precisión aritmética fuese mayor.
  - Indica la dirección desde la cual ocurrió el redondeo a cero, o el signo de un infinito que fue invertido.

Autor: Ing. Alejandro Furfaro

Codificación de números reales

- Números finitos normalizados
  - El rango de éstos números se compone de todos los valores finitos distintos de cero codificables en formato de números reales entre 0 e ∞.
  - En el formato de punto flotante simple precisión estos números se componen de todos aquellos cuyos exponentes desplazados van de 1 a 254, (no desplazados van de -126 a 127).
  - Cuando se aproximan a cero, estos números no pueden seguir expresándose en este formato, ya que el rango del exponente no puede compensar el desplazamiento a izquierda del punto decimal.
  - Cuando se llega a un exponente cero en un número normalizado, se pasa al rango de-normalizado.

Codificación de números reales

- Números finitos de-normalizados
  - En general las operaciones entre números normalizados arrojan como resultado otro número normalizado.
  - Si hay underflow se pasa a trabajar en formato de-normalizado.
  - Ej:

Operation	Sign	Exponent*	Significand
True Result	0	–129	1.01011100000...00
Denormalize	0	–128	0.10101110000...00
Denormalize	0	–127	0.01010111000...00
Denormalize	0	–126	0.00101011100...00
Denormal Result	0	–126	0.00101011100...00

* Expressed as an unbiased, decimal number.

Bits de pérdida de precisión

Codificación de números reales

- Infinitos signados
  - +∞ y -∞, representan los máximos números reales positivo y negativo representables en formato de unto flotante
  - La mantisa siempre es 1.000....00, y el máximo exponente desplazado representable (p. Ej. 255 para precisión simple)
- NaNs
  - Not a Number.
  - No son parte del rango de números reales.
  - QNaN: Quiet NaN tiene el bit mas significativo fraccional seteado. Pueden propagarse por posteriores operaciones sin generar una excepción.
  - SNaN: Signaled NAN. Tiene en cero el bit fraccional mas significativo. Resulta de una operación inválida de punto flotante.

Autor: Ing. Alejandro Furfaro

Redondeo

- Consideremos el siguiente número binario

  1.0110 0000 1011 0100 1101 0111 E2 1101

  – La parte fraccional tiene 24 bits.

  – ¿Cómo lo representamos en punto flotante simple precisión si tenemos 24 bits para la parte decimal?.

  – Hay que redondearlo en 23 bits => hay que decidir que valor darle a los últimos tres bits:

    - Alternativa 1

      1.0110 0000 1011 0100 1101 011 E2 1101

    - Alternativa 2

      1.0110 0000 1011 0100 1101 100 E2 1101

  – Cualquiera sea el criterio de redondeo, se pierde precisión

Tipos de datos: Números reales

Class		Sign	Biased Exponent	Significand	
				Integer[1]	Fraction
Positive	+∞	0	11..11	1	00..00
	+Normals	0 . . 0	11..10 . . 00..01	1 . . 1	11..11 . . 00..00
	+Denormals	0 . . 0	00..00 . . 00..00	0 . . 0	11.11 . . 00..01
	+Zero	0	00..00	0	00..00
Negative	–Zero	1	00..00	0	00..00
	–Denormals	1 . . 1	00..00 . . 00..00	0 . . 0	00..01 . . 11..11
	–Normals	1 . . 1	00..01 . . 11..10	1 . . 1	00..00 . . 11..11
	–∞	1	11..11	1	00..00
NaNs	SNaN	X	11..11	1	0X..XX[2]
	QNaN	X	11..11	1	1X..XX
	QNaN Floating-Point Indefinite	1	11..11	1	10..00
	Single-Precision: Double-Precision: Double Extended-Precision:		← 8 Bits → ← 11 Bits → ← 15 Bits →		← 23 Bits → ← 52 Bits → ← 63 Bits →

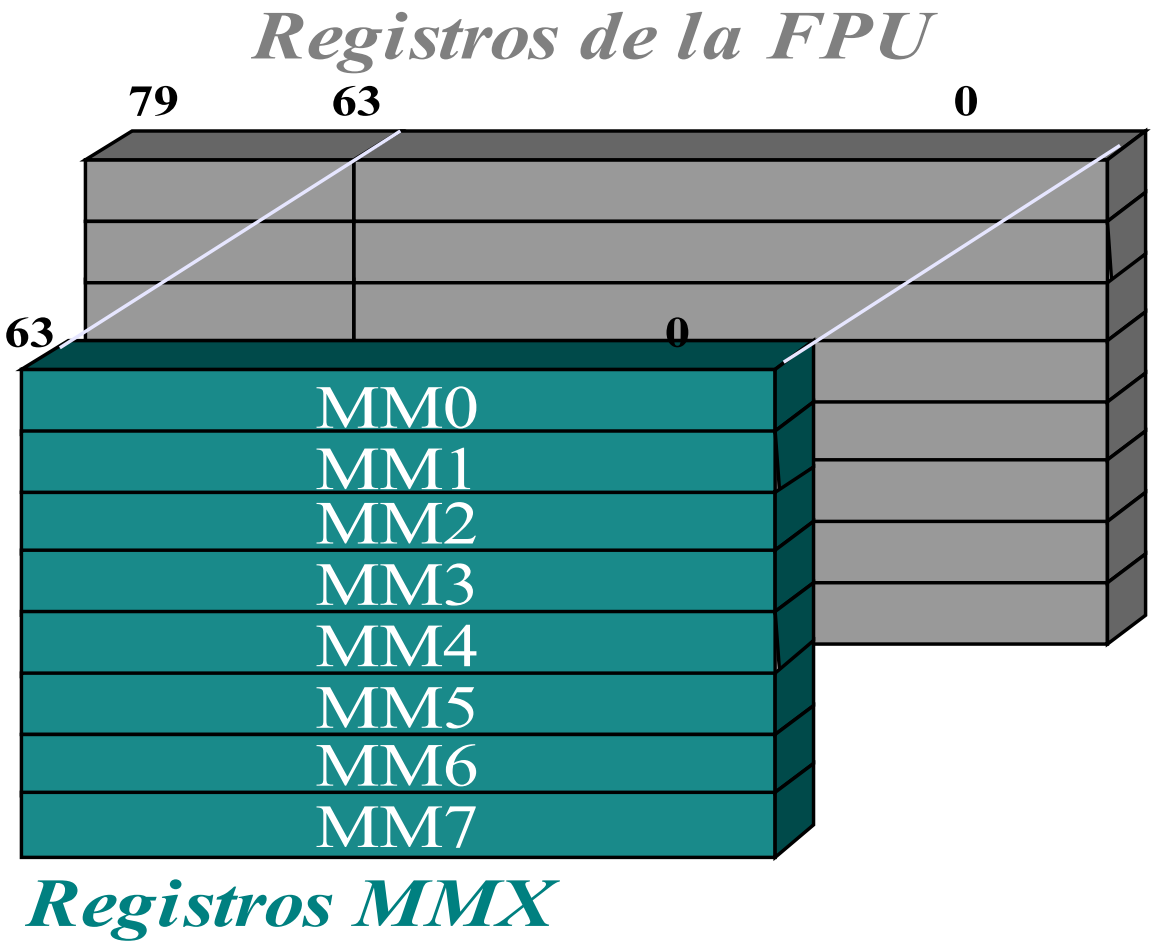
1. El bit entero está implícito y no se almacena para formatos single-precision y double-precision.
2. La fracción para codificación de SNaN debe ser distinta de cero, con el bit mas significativo en 0.

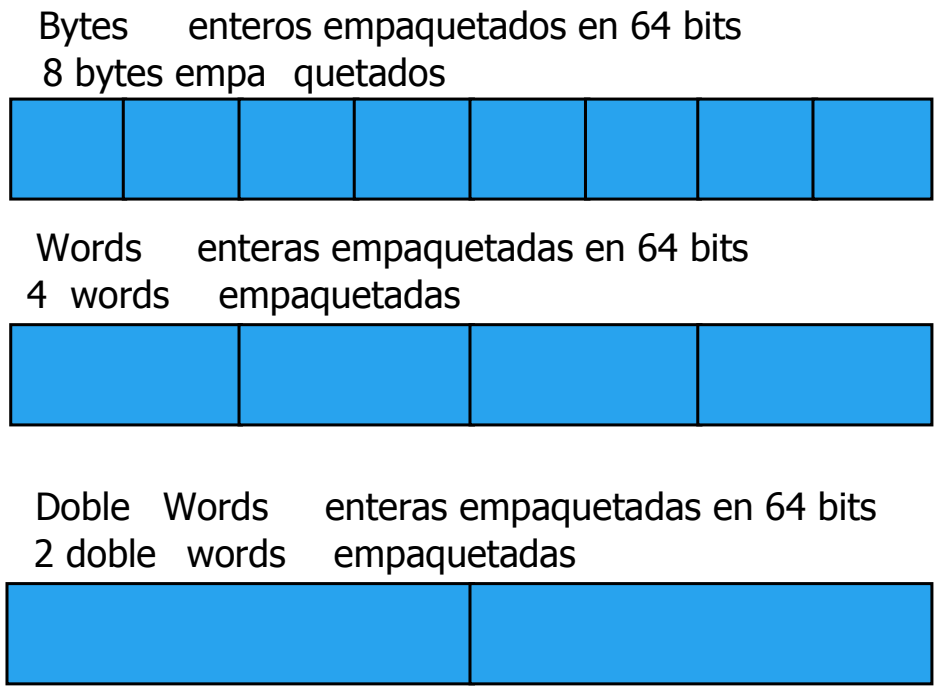
Procesamiento Digital de Señales

Modelo SIMD en procesadores IA-32

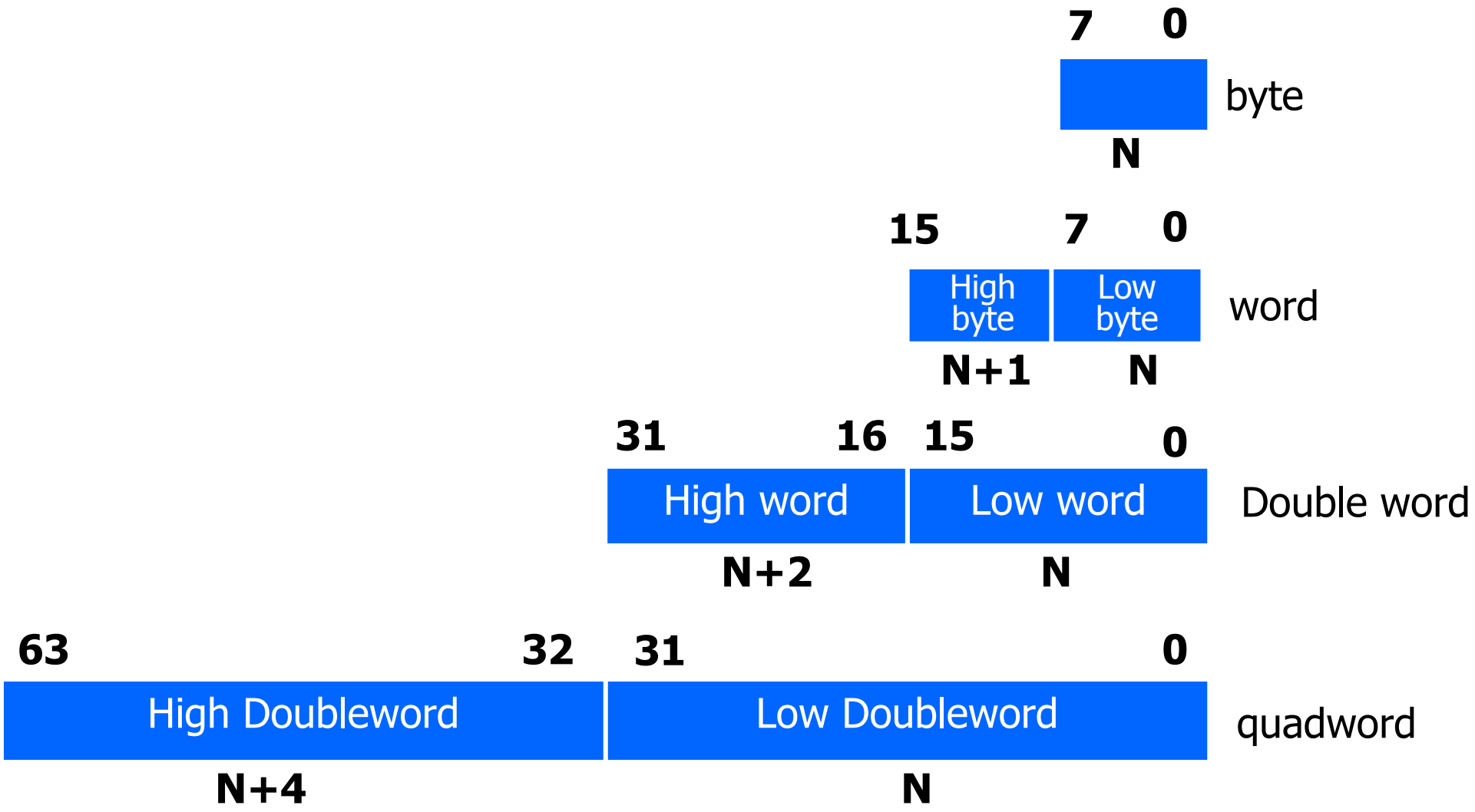
Primer implementación SIMD: Tecnología MMX

El procesador Pentium MMX introdujo en la arquitectura IA-32 un set de recursos para el tratamiento de señales.

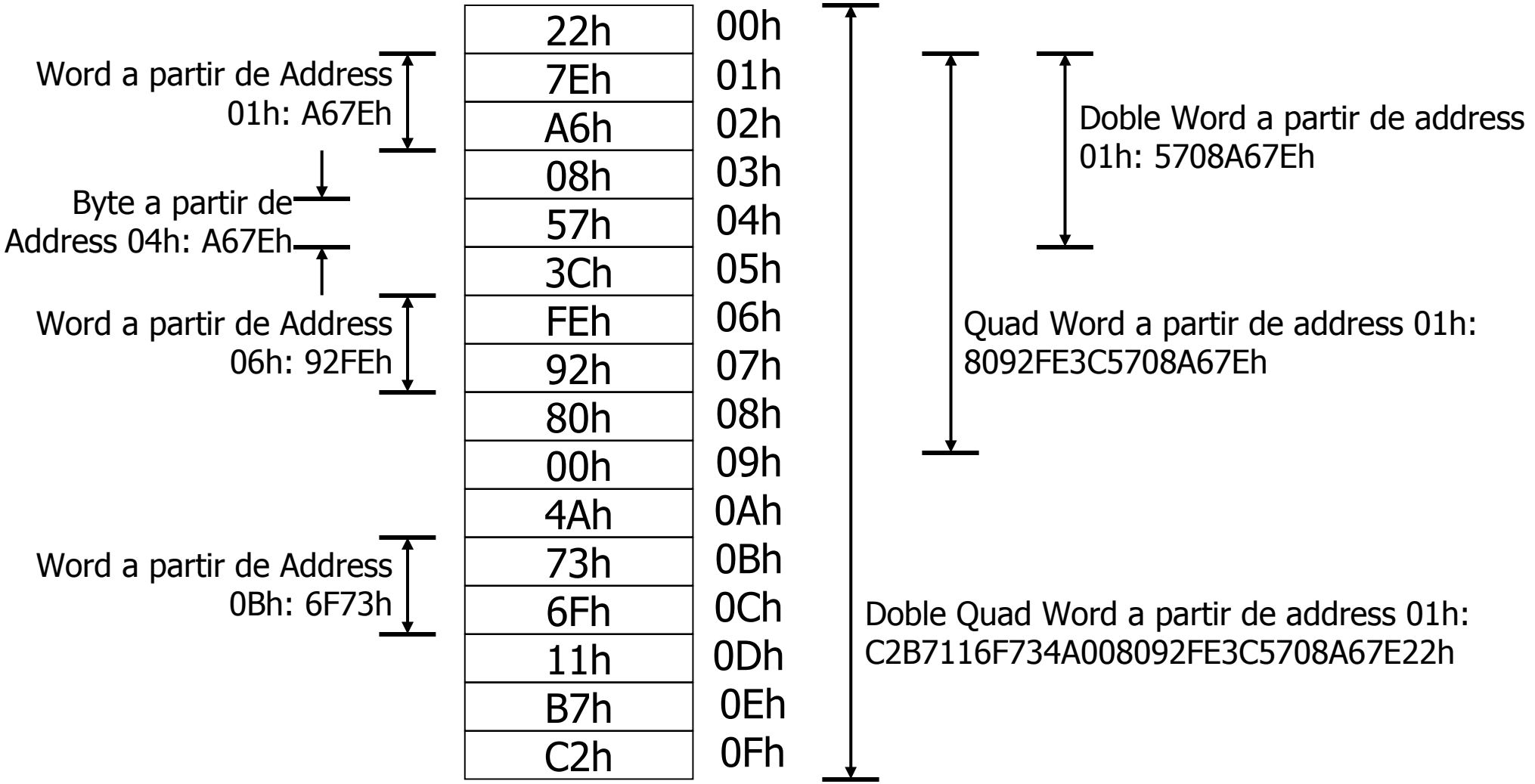




MMX : Tipos de datos



Datos en Memoria



Tipos de datos enteros

- Enteros no signados
  - 0 – 255 bytes enteros no signados
  - 0 – 65535 words enteras no signadas
  - 0 – (2^{32}–1) double words enteras no signadas
  - 0 – (2^{64}–1) quad words enteras no signadas
- Enteros signados
  - -128 – 127 bytes enteros signados
  - -32768 – 32767 words enteras signadas
  - -2^{31} – (2^{31}–1) double words enteras signadas
  - -2^{63} – (2^{63}–1) quad words enteras signadas

Autor: Ing. Alejandro Furfaro

Tipos de datos enteros

Class		Two's Complement Encoding	
		Sign	
Positive	Largest	0	11..11
	.	.	.
	.	.	.
	Smallest	0	00..01
Zero		0	00..00
Negative	Smallest	1	11..11
	.	.	.
	.	.	.
	Largest	1	00..00
Integer indefinite		1	00..00
		Signed Byte Integer:	← 7 bits →
		Signed Word Integer:	← 15 bits →
		Signed Doubleword Integer:	← 31 bits →
		Signed Quadword Integer:	← 63 bits →

Autor: Ing. Alejandro Furfaro

Algoritmos DSP

- Para implementar hardware apto para DSP, los diseñadores estudiaron las características de los algoritmos de procesamiento de audio, imágenes 2D y 3D, compresión de audio y video, text to speech, filtrado digital, etc.
- Características comunes
  - tipos de datos de poco tamaño,
    - pixeles: 8 o 16 bits
    - muestras de audio: 8 o 16 bits
  - patrones de acceso secuencial a memoria,
    - buffers circulares
  - operaciones simples y recurrentes sobre los datos de entrada.
    - Acumulaciones de productos

Aritmética convencional. Algoritmos de calculo

- Acumulación de resultados sobre un registro
  - Se llega a un punto en el que el rango del resultado excede la capacidad de bits del registro de acumulación.
  - En este punto los procesadores convencionales indican la situación mediante un flag de overflow, y en el operando destino se almacena el valor de desborde.
  - La aplicación chequea dentro del lazo de cálculo este flag y de acuerdo a su estado decide si sigue la acumulación.
  - El costo de esta práctica de programación es tiempo de ejecución para analizar como tratar cada resultado parcial.

Aritmética DSP. Algoritmos de calculo

- Aritmética de desborde (Wraparound)
  - Operatoria clásica basada en afectar un flag durante la operación que genera el desborde e invertir tiempo de procesamiento en el loop para evaluar el estado de ese flag
- Aritmética Saturada
  - Al producirse una condición fuera de rango el operando destino mantiene el máximo o mínimo valor del rango (dependiendo si la condición se ha producido por exceso o por defecto).
  - Aritmética Saturada Signada

Tipo de dato	Límite inferior		Límite superior	
	Hexadecimal	Decimal	Hexadecimal	Decimal
byte signado	80	-128	7F	127
word signada	8000	-32768	7FFF	32767

  - Aritmética Saturada No Signada

Tipo de dato	Límite inferior		Límite superior	
	Hexadecimal	Decimal	Hexadecimal	Decimal
byte no signado	00	0	FF	255
word no signada	0000	0	FFFF	65535

Autor: Ing. Alejandro Furfaro

Aritmética saturada vs. aritmética de desborde

En algunas aplicaciones la aritmética saturada provee soluciones más eficaces que la aritmética de desborde. Por ejemplo, al procesar video cuando un nivel de negro satura, no tiene sentido seguir procesando la variable ya que no produce mas efecto sobre la salida. En cambio si utilizamos la habitual lógica de desborde el valor remanente en el registro de resultado al saturar el negro nos lleva de vuelta al blanco, ya que afecta el bit de overflow pero el número almacenado en el operando del resultado es mucho menor al máximo del rango.

Por ejemplo, supongamos el siguiente caso: Dos operandos de 64 bits con bytes empaquetados. Los valores en todos los casos están en hexadecimal.

	byte —	7	6	5	4	3	2	1	0
	operando 1	4D	23	9F	C0	11	4A	29	0B
	operando 2	32	FF	1A	0D	3F	AF	B0	36
Suma empaquetada	Aritmética de desborde	7F	22	B9	CD	50	F9	D9	41
	Aritmética Saturada	7F	FF	B9	CD	50	F9	D9	41

Instrucciones MMX: Transferencia de datos

- MOVD: Move Double Word

```
MOVD mm,r/m32   Move double word from r/m32 to mm.
MOVD r/m32,mm   Move double word from mm to r/m32.
MOVD xmm,r/m32  Move double word from r/m32 to xmm.
MOVD r/m32,xmm  Move double word from xmm to r/m32.
r = Reg. de propósito general (EAX, EBX, EBP, etc.)
m32 = Dirección de memoria para un dato de 32 bits
mm = Registro MMX de 64 bits (MM0 a MM7)
xmm = Registro XMM de 128 bits
```


Autor: Ing. Alejandro Furfaro

Instrucciones MMX: Transferencia de Datos

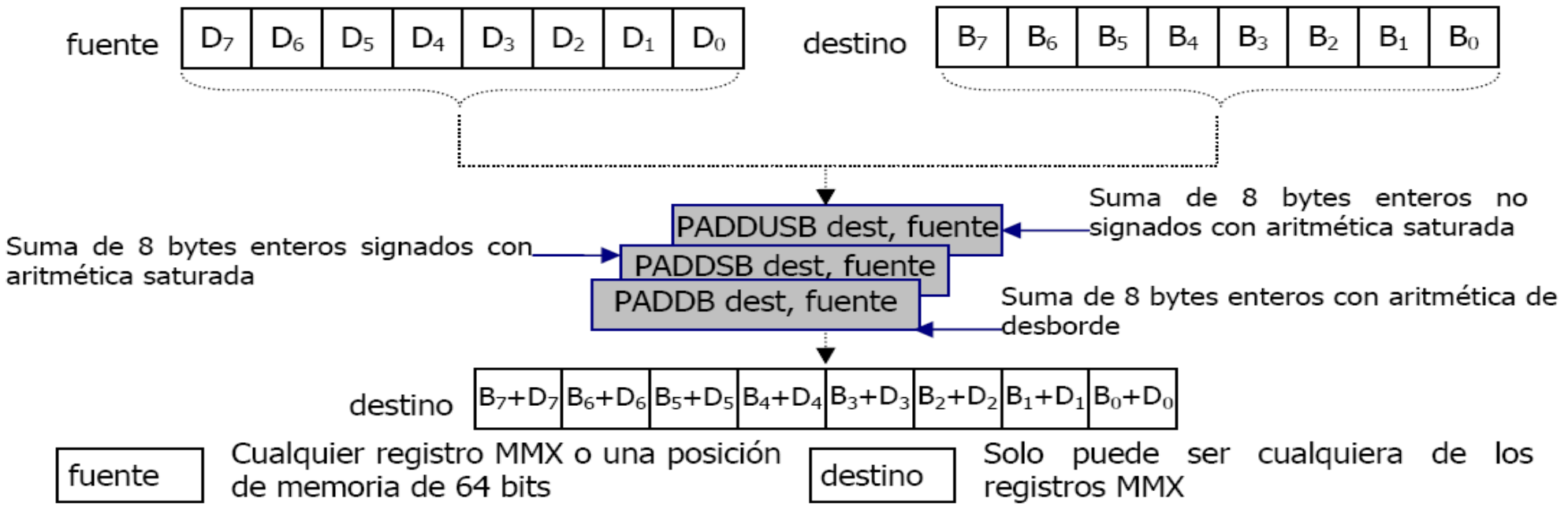
- MOVQ: Move Quad Word

```
MOVQ mm,mm/m64 Move quadword from mm/m64 to mm.
MOVQ mm/m64,mm  Move quadword from mm to mm/m64.
MOVQ xmm1,xmm2/m64 Move quadword from xmm2/mem64 to xmm1.
MOVQ xmm2/m64,xmm1Move quadword from xmm1 to xmm2/mem64.
m64 = Dirección de memoria para un dato de 64 bits
mm  = Registro MMX de 64 bits (MM0 a MM7)
xmm = Registro XMM de 128 bits
```

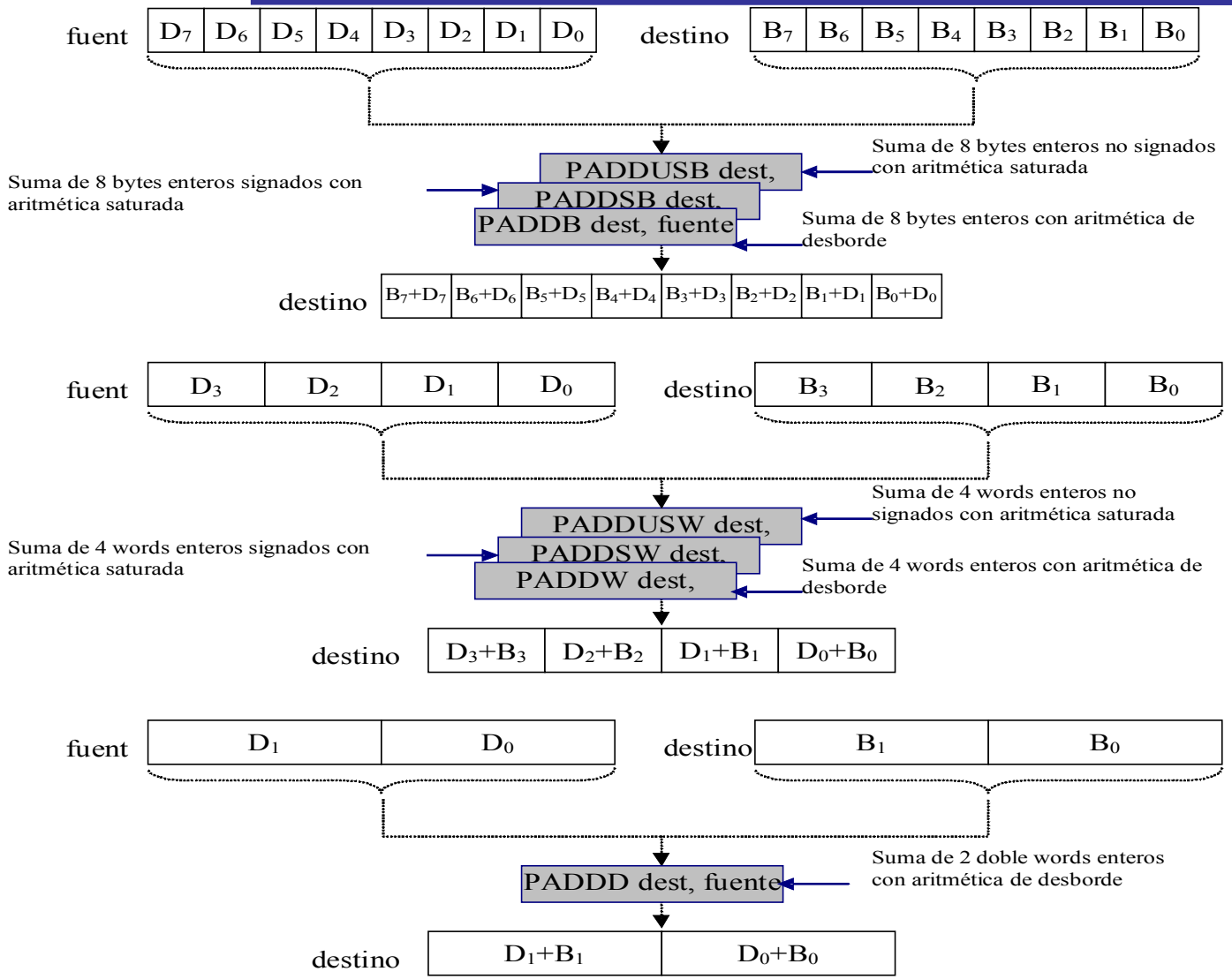

Autor: Ing. Alejandro Furfaro

Instrucciones MMX: Aritméticas

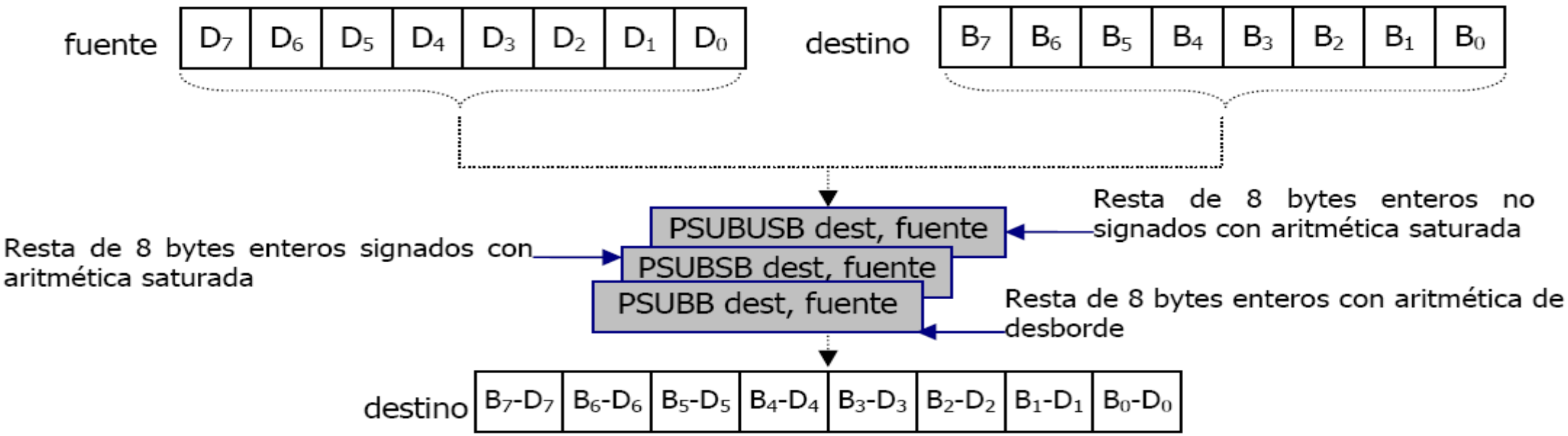
Aritmética →	Desborde			Saturación Signada		Saturación No Signada	
Operación ↓	**byte**	**word**	**doble word**	**byte**	**word**	**byte**	**word**
Suma	PADDB	PADDW	PADDD	PADDSB	PADDSW	PADDUSB	PADDUSW
Resta	PSUBB	PSUBW	PSUBD	PSUBSB	PSUBSW	PSUBUSB	PSUBUSW
Multiplicación		PMULLW PMULHW					
Multiplicación / Acumulación	PMADDWD						



Instrucciones Aritméticas MMX: Suma empaquetada



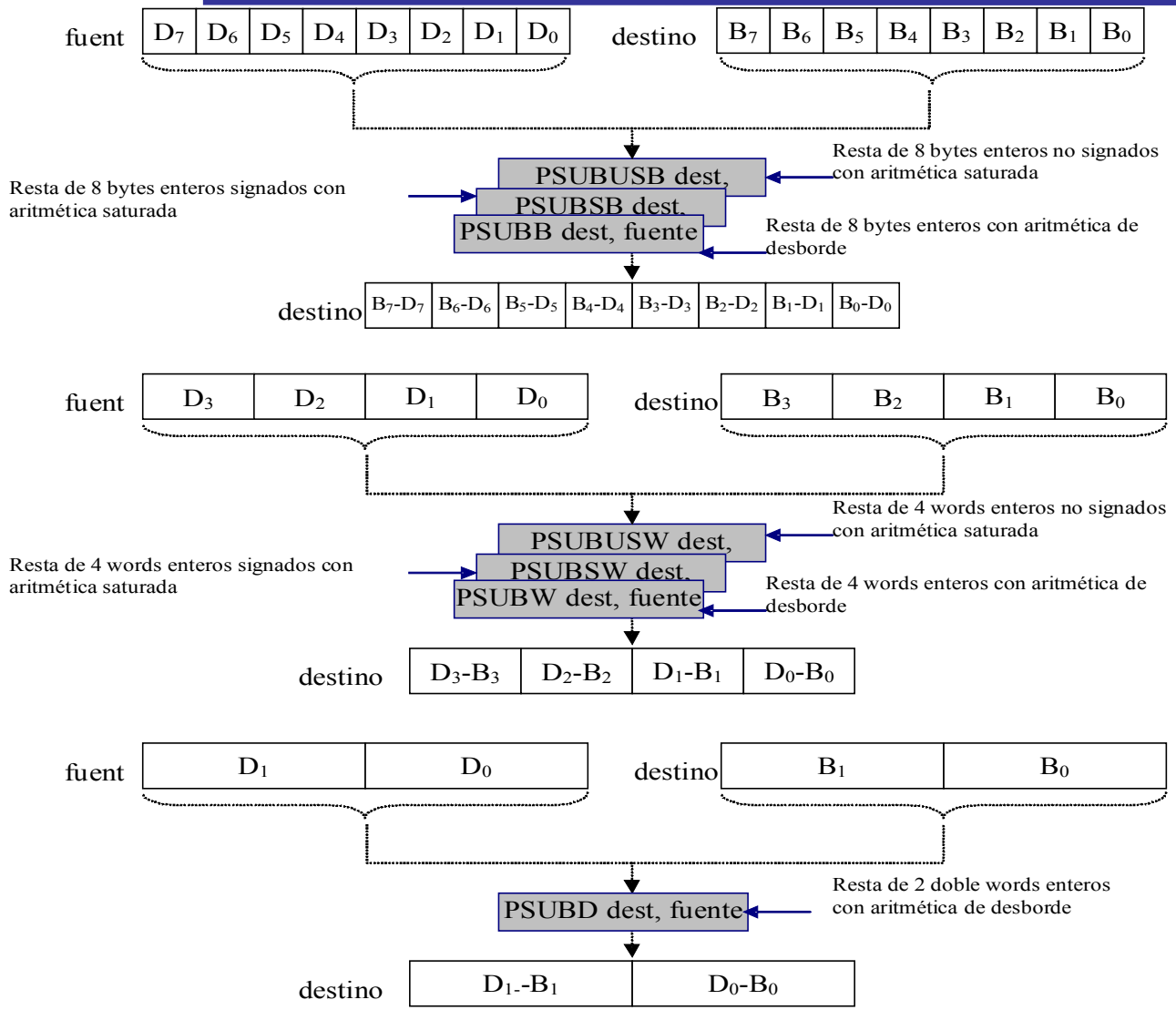
Instrucciones Aritméticas MMX: Resta empaquetada



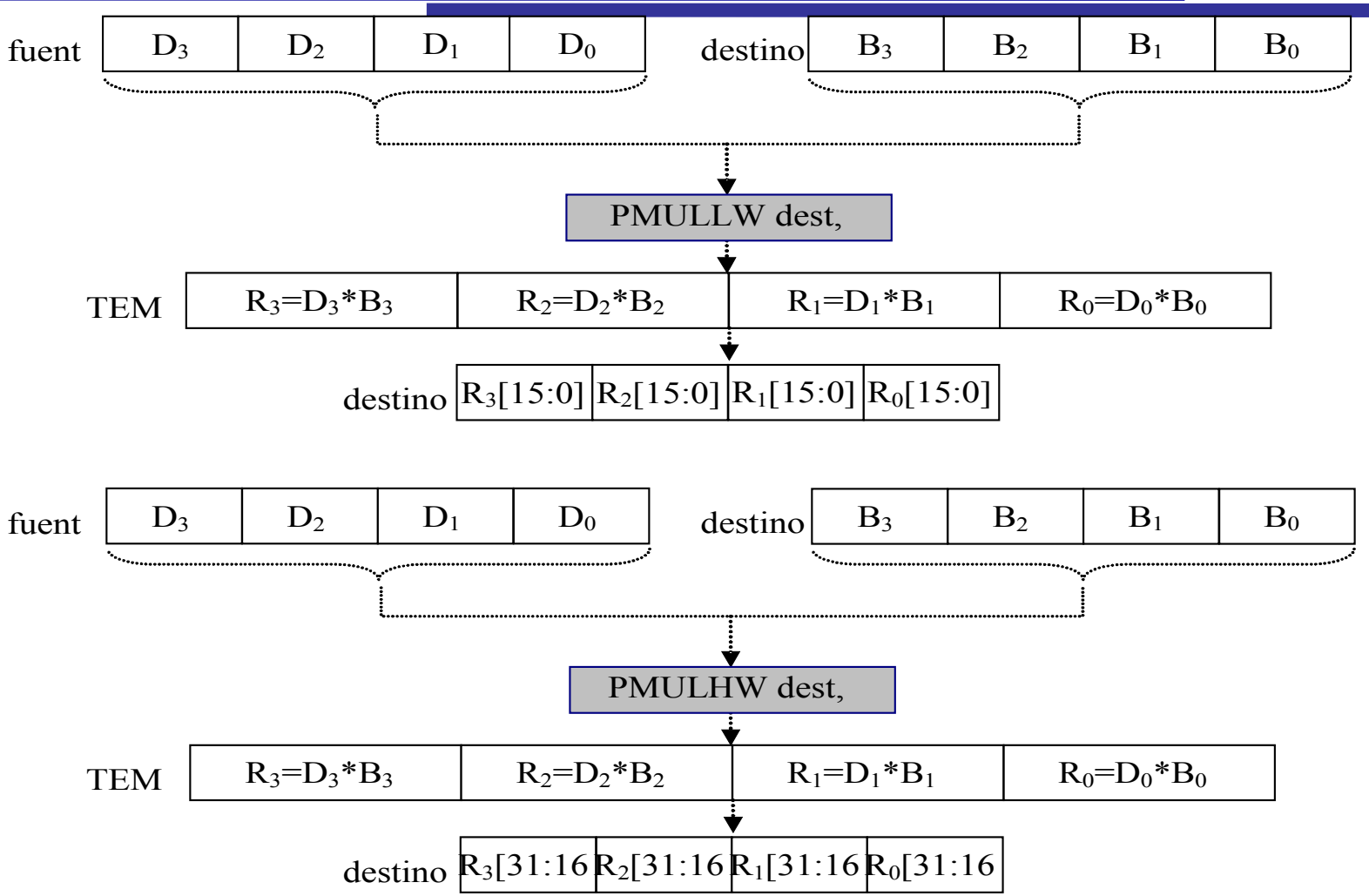
fuente	Cualquier registro MMX o una posición de memoria de 64 bits	destino	Solo puede ser cualquiera de los registros MMX

Autor: Ing. Alejandro Furfaro

Instrucciones Aritméticas MMX: Resta empaquetada



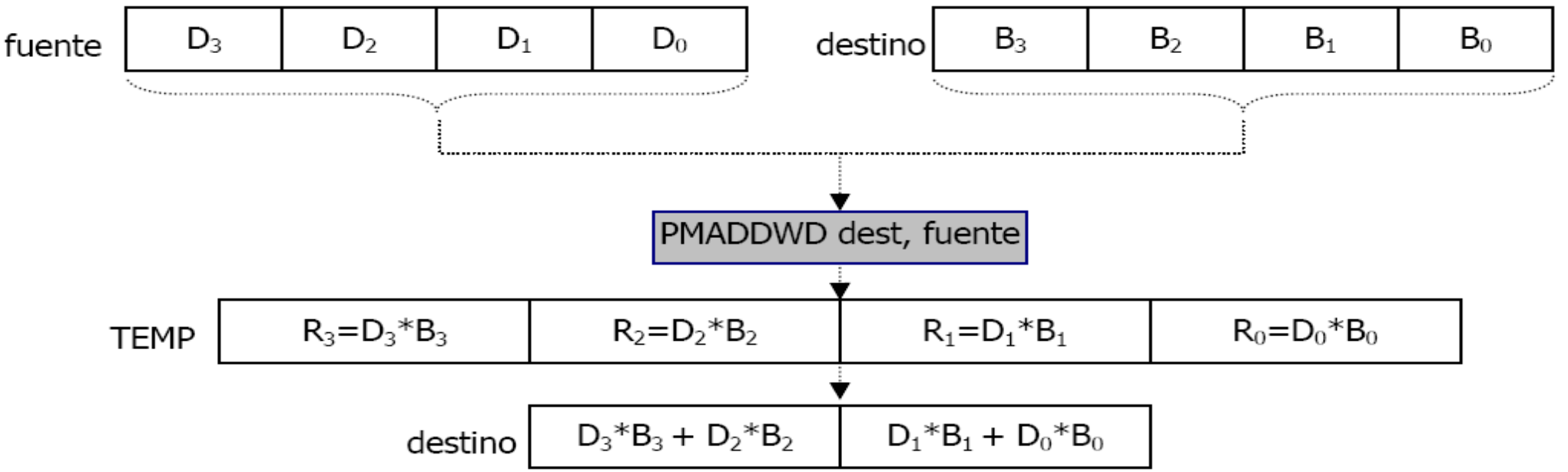
Instrucciones Aritméticas MMX: Multiplicación empaquetada



fuente — Cualquier registro MMX o una posición de memoria de 64 bits

destino — Solo puede ser cualquiera de los registros MMX

Instrucciones Aritméticas MMX: Suma de productos



fuente	Cualquier registro MMX o una posición de memoria de 64 bits
destino	Solo puede ser cualquiera de los registros MMX

Autor: Ing. Alejandro Furfaro

Instrucciones MMX

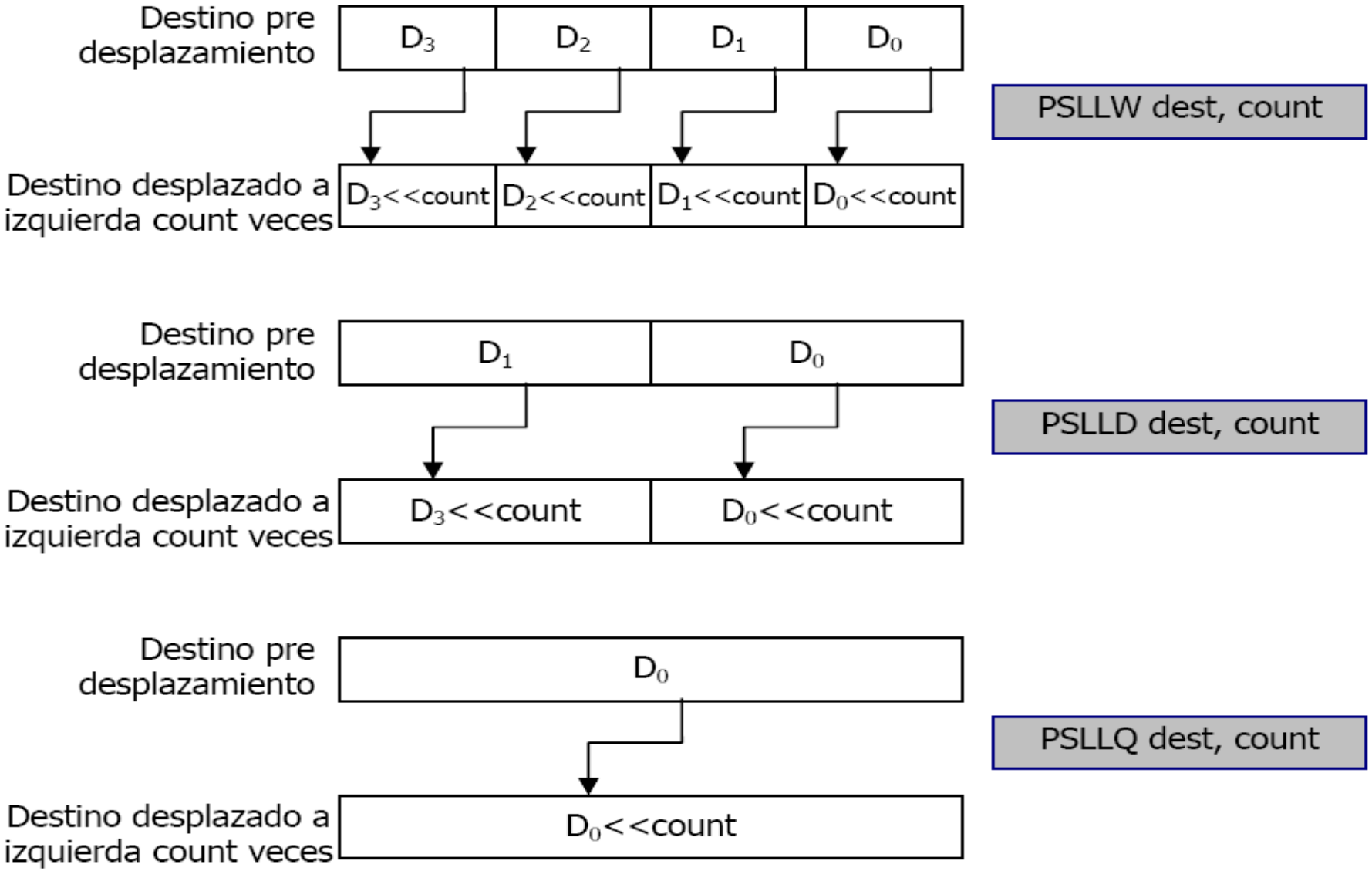
- Lógicas
  - Tratan a los contenidos de los registros MMX como datos puros de 64 bits.
  - No tienen en cuenta ningún tipo de empaquetamiento de modo que resultan mas similares a las instrucciones correspondientes al modelo de registros de propósito general.
  - Son cuatro instrucciones PAND, PNAND, POR, y PXOR.
  - La P inicial las identifica como instrucciones específicas para este tipo de registros.
  - El operando fuente pueden ser cualquier registro MMX o cualquier dirección de memoria de 64 bits de ancho.
  - El operando destino solo puede ser un registro MMX.

Autor: Ing. Alejandro Furfaro

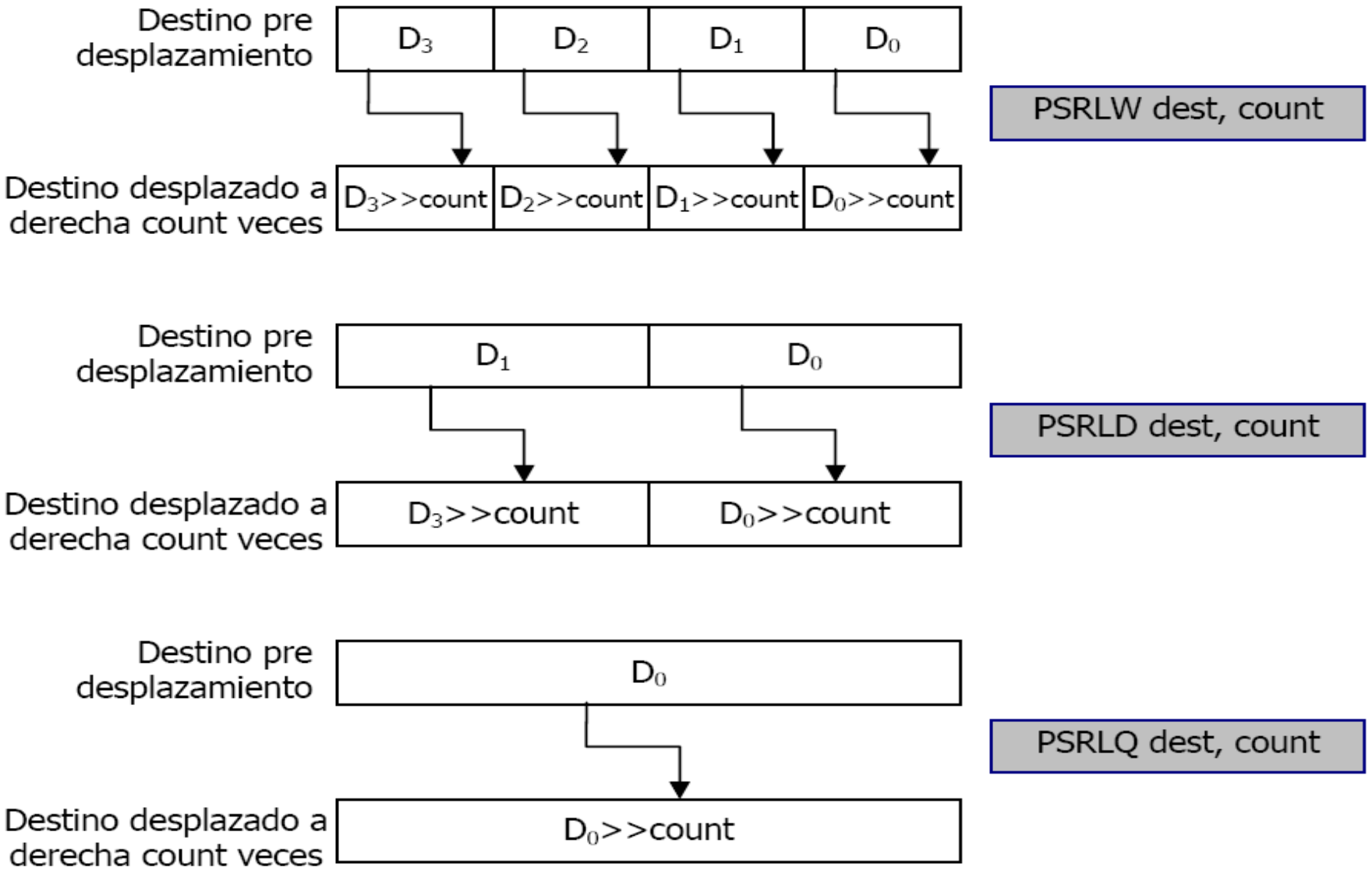
Instrucciones MMX: Desplazamiento

- Desplaza a derecha o izquierda en la cantidad de bits que se indica cada operando empaquetado dentro del registro o dirección de memoria que se indica.
  - Las instrucciones PSLLW, PSLLD, y PSLLQ desplazan en forma lógica a izquierda (Packed Shift Left Logical) los respectivos operandos empaquetados (Words, Doble words, o Quad Words).
  - Las instrucciones PSRLW, PSRLD, y PSRLQ realizan lo propio pero desplazando a la derecha (Packed Shift Right Logical).
  - Las instrucciones PSRAW y PSRAD realizan un desplazamiento a derecha aritmético (conservan el signo de cada operando empaquetado). Solo se dispone de instrucciones de Word y doble words empaquetadas.

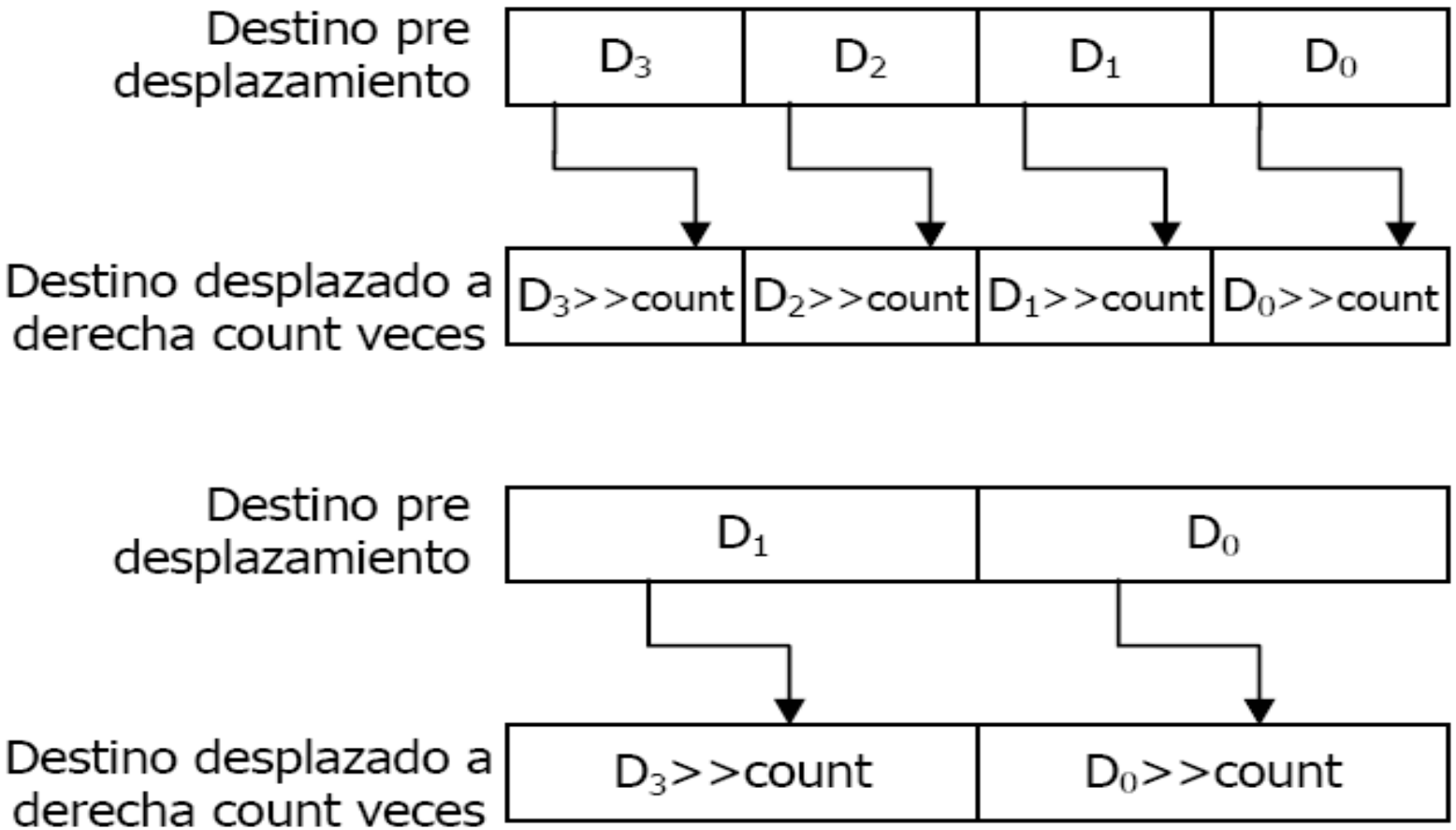
Instrucciones MMX: Desplazamiento



Instrucciones MMX: Desplazamiento



Instrucciones MMX: Desplazamiento



Instrucciones MMX: Comparación

•A diferencia de las instrucciones de comparación convencionales, estas instrucciones no afectan el contenido del registro EFLAGS.

•Lo que hacen en cambio es setear una máscara de bits en el operando destino, que pueden utilizarse como MOV condicionales sin saltos condicionales que demanden tiempo de verificación.

•Hay dos tipos de comparaciones posibles: por igual (PCMPEQ) o por mayor (PCMPGT).

•PCMPEQB, PCMPEQW, PCMPEQD y PCMPGTB, PCMPGTW, PCMPGTD comparan los correspondientes elementos signados (bytes, words, o double words) entre los operandos fuente y destino por igual o mayor que, respectivamente.

Autor: Ing. Alejandro Furfaro

Instrucciones MMX: Comparación

- En el caso de que la condición sea EQ (Equal) cada dato empaquetado del operando destino que cumpla la condición se pone en FF, FFFF, o FFFFFFFF, según el tipo de dato empaquetado que se evalúe. Si la condición no se cumple se pone a 0.
- Son instrucciones útiles para operaciones tan simples como inicializar en 1's un determinado registro, como por ejemplo:

```
PCMPEQB   mm0,mm0 ;mm0 =FF FF FF FF FF FF FF FF.
```

- También para operaciones algo mas sofisticadas como generación de máscaras de modo de procesar bits de manera mas eficiente.

Autor: Ing. Alejandro Furfaro

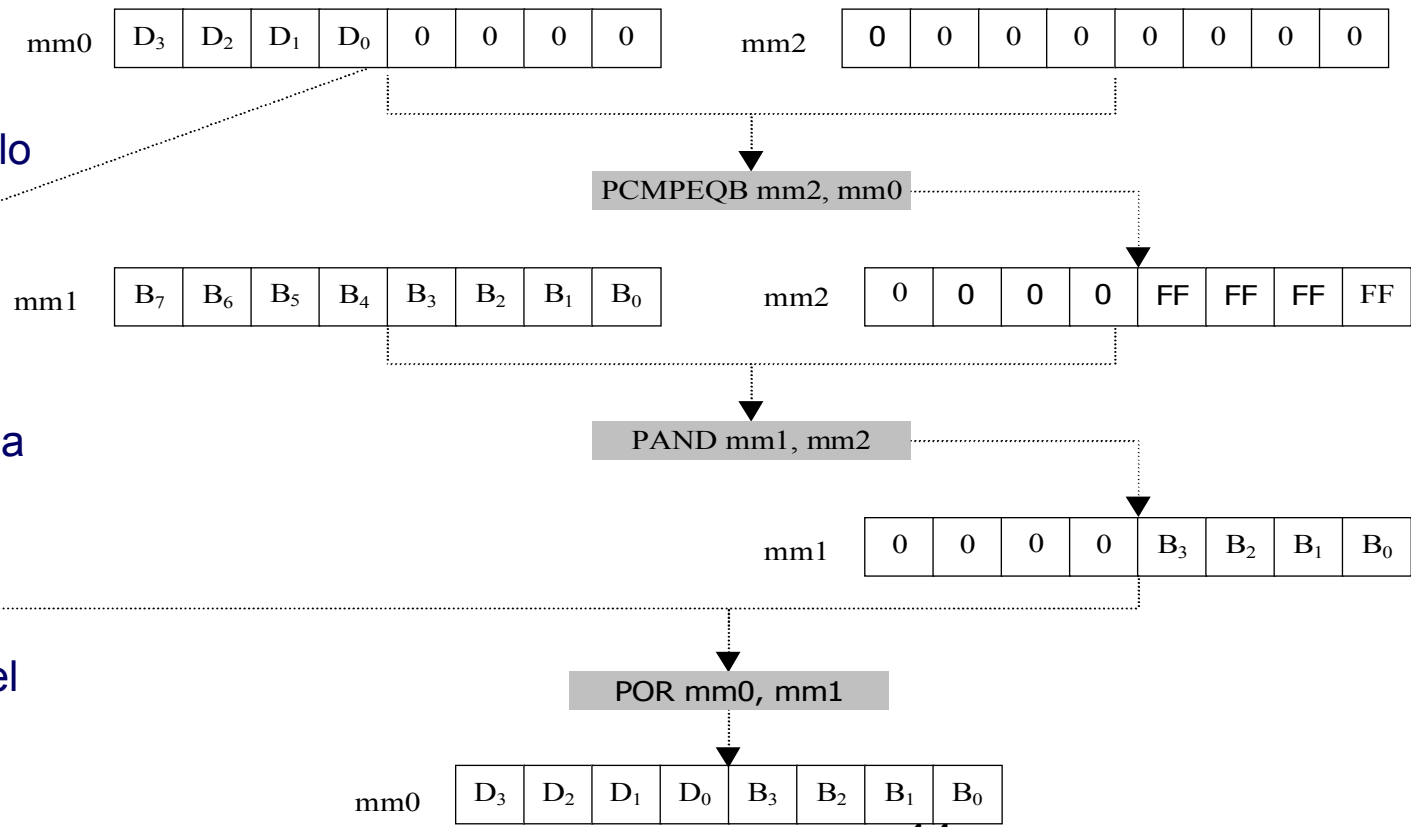
Instrucciones MMX: Ejemplo con Comparación

- Procesamiento de sprites o iconos en gráficos 2D.
- Un sprite es un mapa de bits en el que se dibuja una figura pequeña, de modo tal que los bits que no se utilizan deben quedar “transparentes”, es decir, quedan del color del fondo.
- Supongamos un sprite de 8 pixels de ancho por otros 8 pixels de alto.
  - Para la primer línea de 8 pixels, el dibujo del sprite corresponde a los primeros cuatro mientras que los cuatro siguientes deben ser transparentes, es decir, deben ir del color del fondo.

Autor: Ing. Alejandro Furfaro

Instrucciones MMX: Ejemplo con Comparación

- mm0 carga la primer línea del sprite, (8 bytes empaquetados).
- mm2 trabaja como auxiliar, está pre seteado en cero.
- PCMPEQB entre ambos registros, generará una máscara con los bytes que contendrán información del sprite en cero y los que deben resultar transparentes en FF.
- El resultado se procesa con la información de los 8 pixels del fondo de la imagen (pre almacenado en el registro mm1) a través de un operador AND. Al resultado le queden solo los cuatro pixels del fondo que se deben mostrar.
- Nos queda imprimir la línea del sprite sobre este fondo, del cual hemos eliminado los cuatro pixels que serán sobre impresos por la línea del sprite.
- Tratando a este resultado mediante una operación OR con los 8 pixels originales del sprite se compone el dibujo buscado sobre el fondo existente.

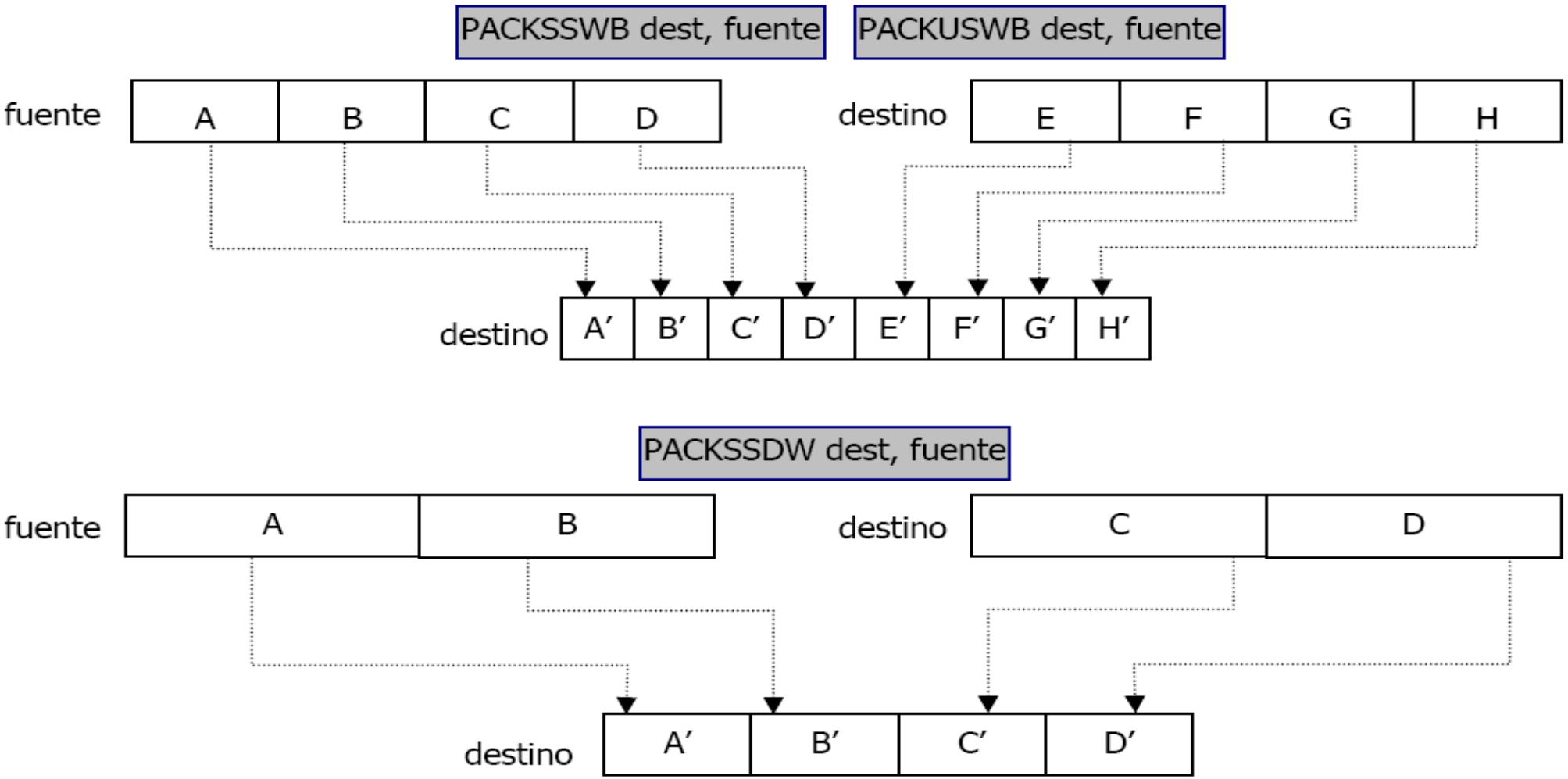


Autor: Ing. Alejandro Furfaro

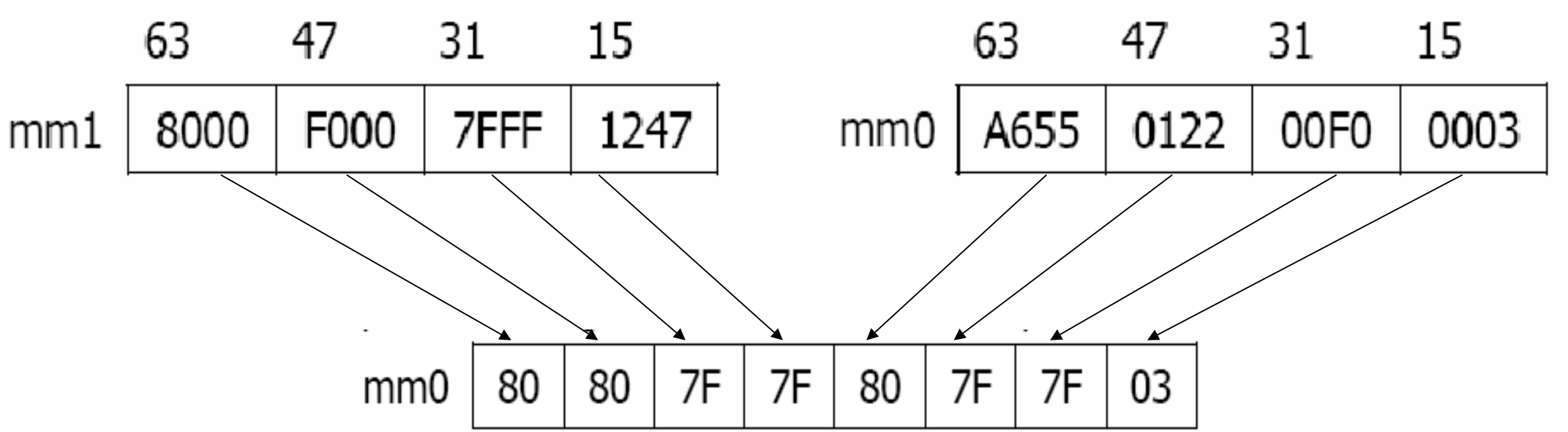
Instrucciones MMX: Conversión

- Se trata de un conjunto de instrucciones auxiliares para convertir datos empaquetados a formatos de datos empaquetados de menor tamaño.
  - PACKSSWB, se utiliza para convertir words signadas empaquetadas en bytes signados empaquetados.
  - PACKSSDW, se utiliza para convertir doble words signadas empaquetadas en words signadas empaquetadas.
  - PACKUSWB, se utiliza para convertir words no signadas en bytes no signados
- Los datos se arman a partir de dos registros MMX o de un registro MMX (operando destino) y una dirección de memoria de 64 bits (operando origen), y el resultado se almacena en el operando destino.

Instrucciones MMX: Conversión



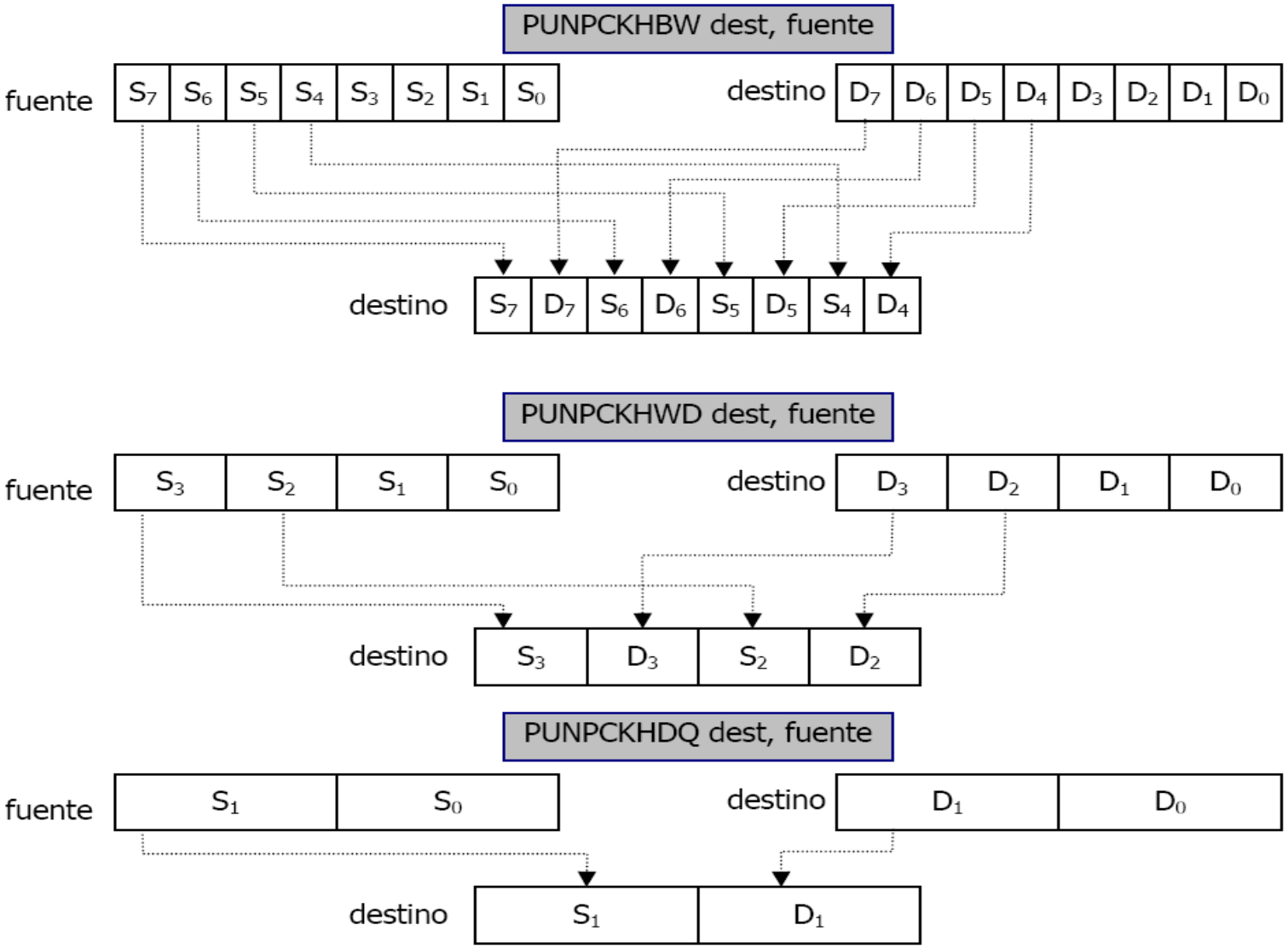
Instrucciones MMX: Conversión



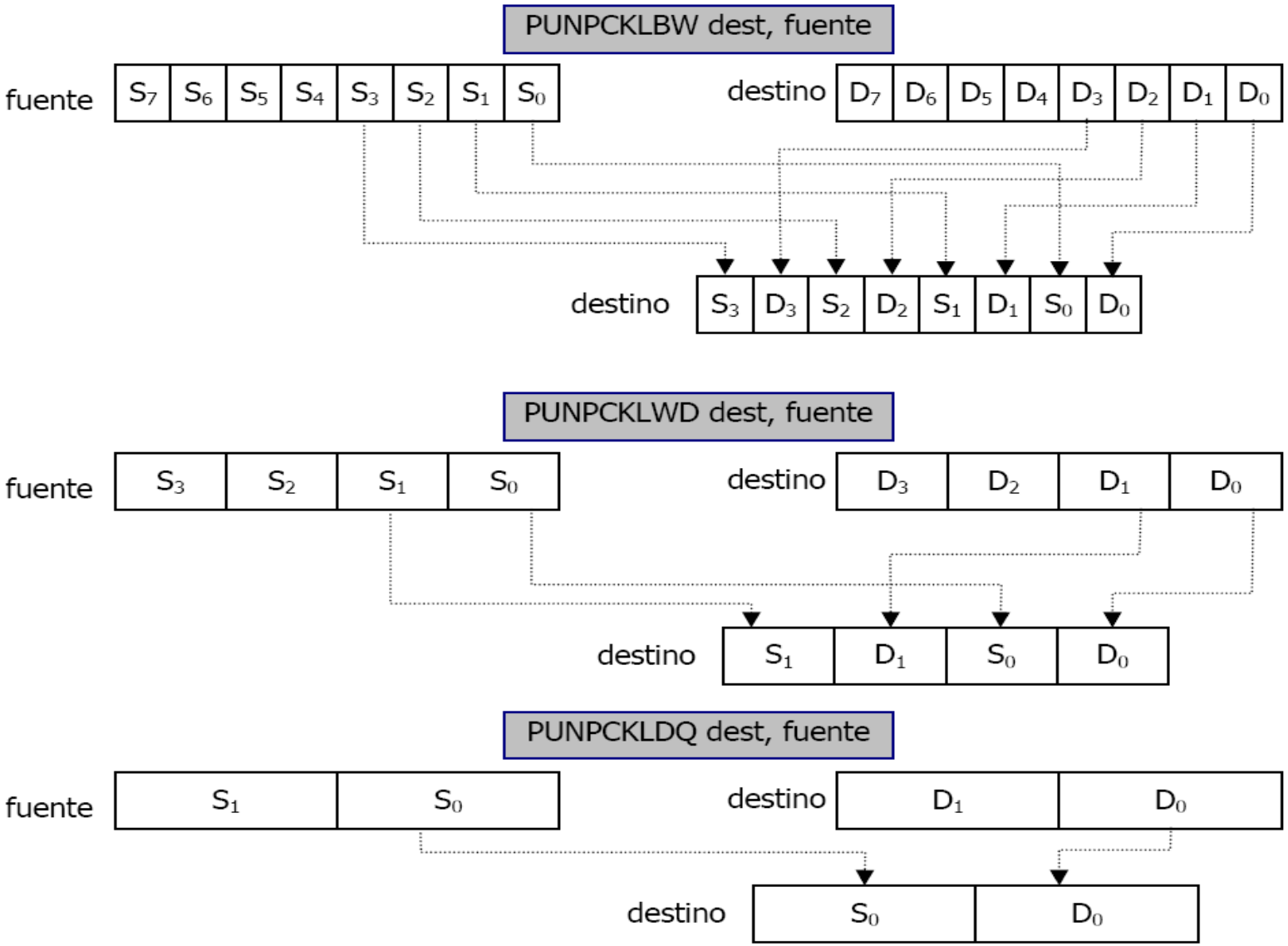
Instrucciones MMX: Desempaquetado

- PUNPCKHBW, PUNPCKHWD, y PUNPCKHDQ, desempaquetan (UNPCK), la parte alta de los operandos fuente y destino (32 bits en cada caso) almacenando los 64 bits del resultado en el operando destino. PUNPCKHBW desempaqueta los bytes empaquetados en la mitad alta de cada operando en el operando destino

Instrucciones MMX: Desempaquetado



Instrucciones MMX: Desempaquetado

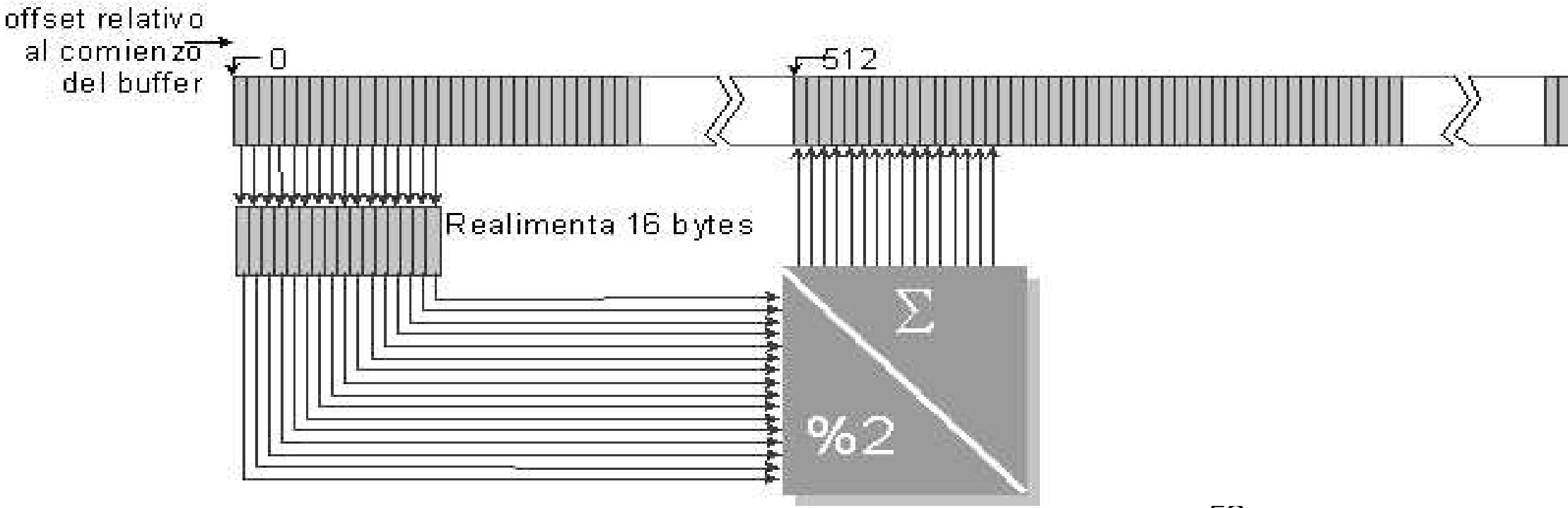


Instrucciones MMX: EMMS

- Esta instrucción se usa para vaciar el contenido de los registros MMX.
- Se debe ejecutar al final de cada rutina que utilice estos recursos antes de invocar alguna instrucción de la FPU

Instrucciones MMX: Ejemplo 2 Eco

- El algoritmo está pensado con de modo tal de comenzar a realimentar las muestras hacia la entrada con una demora equivalente al tiempo de 512 muestras de modo que el efecto resulte apreciable.

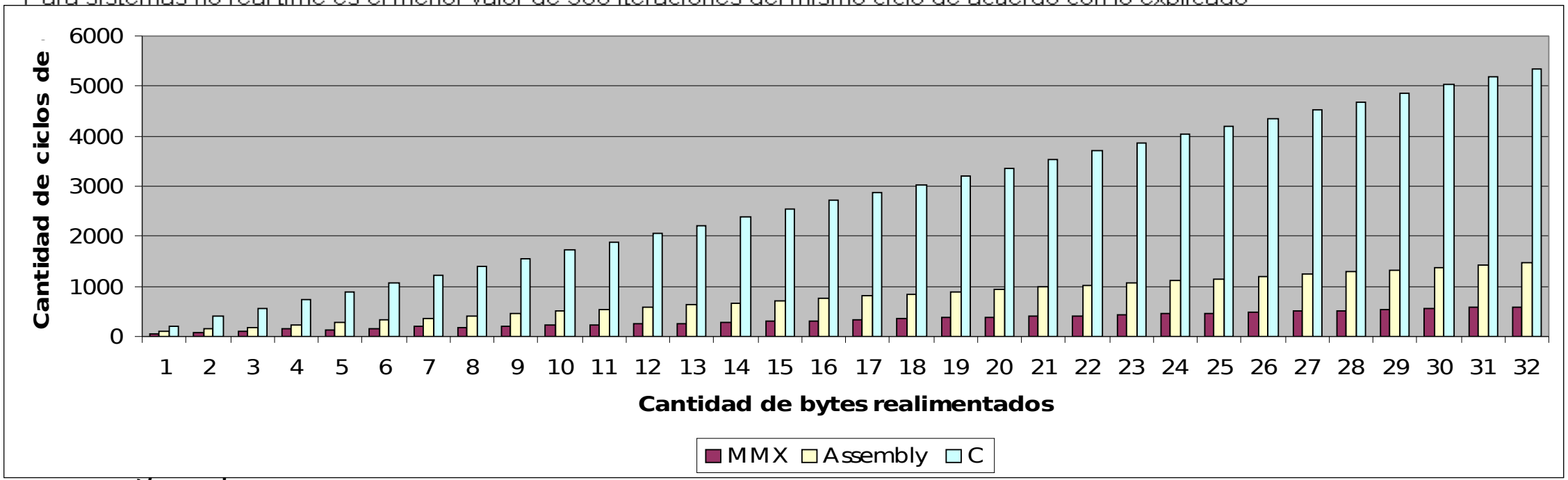


Instrucciones MMX: Ejemplo 2 Eco

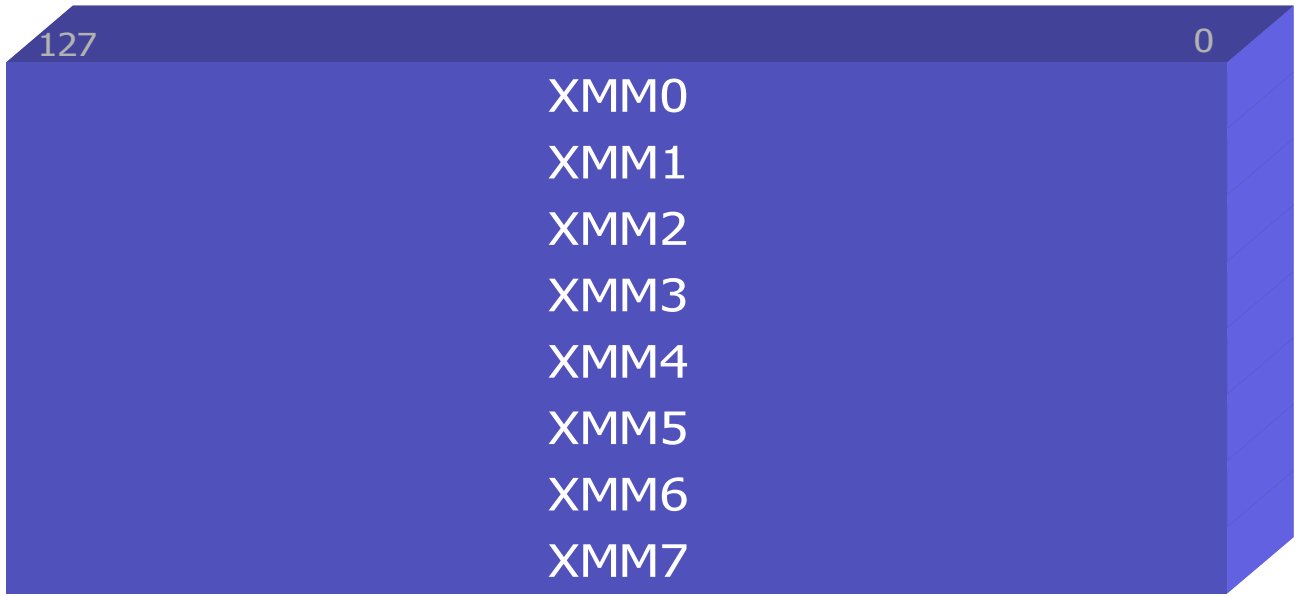
Cantidad de bytes Realimentados		16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256
Cantidad de clocks por Algoritmo *	MMX	54	69	101	146	130	144	192	178	198	219	235	246	266	282	302	314
	Assembly	91	144	187	231	276	320	364	407	451	496	540	584	627	671	716	760
		59,3%	47,9%	54,0%	63,2%	47,1%	45,0%	52,7%	43,7%	43,9%	44,2%	43,5%	42,1%	42,4%	42,0%	42,2%	41,3%
	C	211	395	559	727	893	1063	1230	1391	1557	1717	1884	2055	2217	2381	2536	2716
		25,6%	17,5%	18,1%	20,1%	14,6%	13,5%	15,6%	12,8%	12,7%	12,8%	12,5%	12,0%	12,0%	11,8%	11,9%	11,6%

Cantidad de bytes Realimentados		272	288	304	320	336	352	368	384	400	416	432	448	464	480	496	512
Cantidad de clocks por Algoritmo *	MMX	334	349	372	382	404	416	439	450	470	486	510	519	542	554	578	586
	Assembly	804	847	891	936	980	1024	1067	1111	1156	1200	1244	1287	1331	1376	1420	1464
		41,5%	41,2%	41,8%	40,8%	41,2%	40,6%	41,1%	40,5%	40,7%	40,5%	41,0%	40,3%	40,7%	40,3%	40,7%	40,0%
	C	2878	3035	3213	3358	3541	3712	3856	4034	4188	4336	4536	4671	4849	5035	5192	5350
		11,6%	11,5%	11,6%	11,4%	11,4%	11,2%	11,4%	11,2%	11,2%	11,2%	11,2%	11,1%	11,2%	11,0%	11,1%	11,0%

* Para sistemas no real time es el menor valor de 500 iteraciones del mismo ciclo de acuerdo con lo explicado



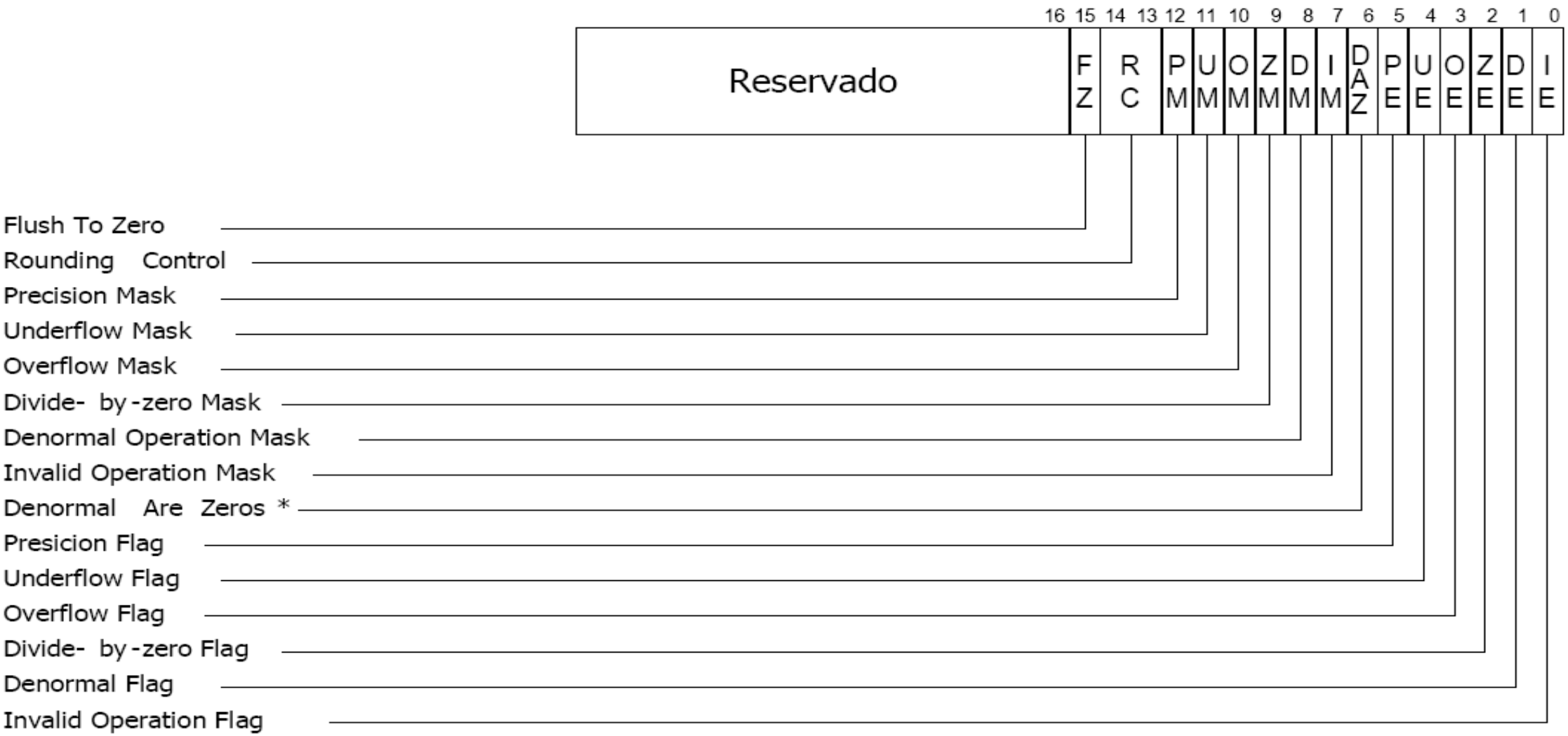
Streaming SIMD Extensions SSE



- Se incorpora a partir del procesador Pentium III
- Mejoras adicionales al modelo SIMD del Pentium MMX, y Pentium II
  - Tipos de datos de 128-bit empaquetando cuatro números de punto flotante simple precisión
  - Soporte para aritmética sobre operandos enteros de 64-bit.
  - Instrucciones de conversión entre tipos de datos nuevos y existentes.
  - Extiende el soporte para cacheabilidad, prebúsqueda y operaciones de ordenamiento de memoria.

Autor: Ing. Alejandro Furfaro

Registro de Estado y Control MXCSR



* El flag denormal-are-zeros (DAZ) es introducido por los procesadores Pentium 4 y Xeon.

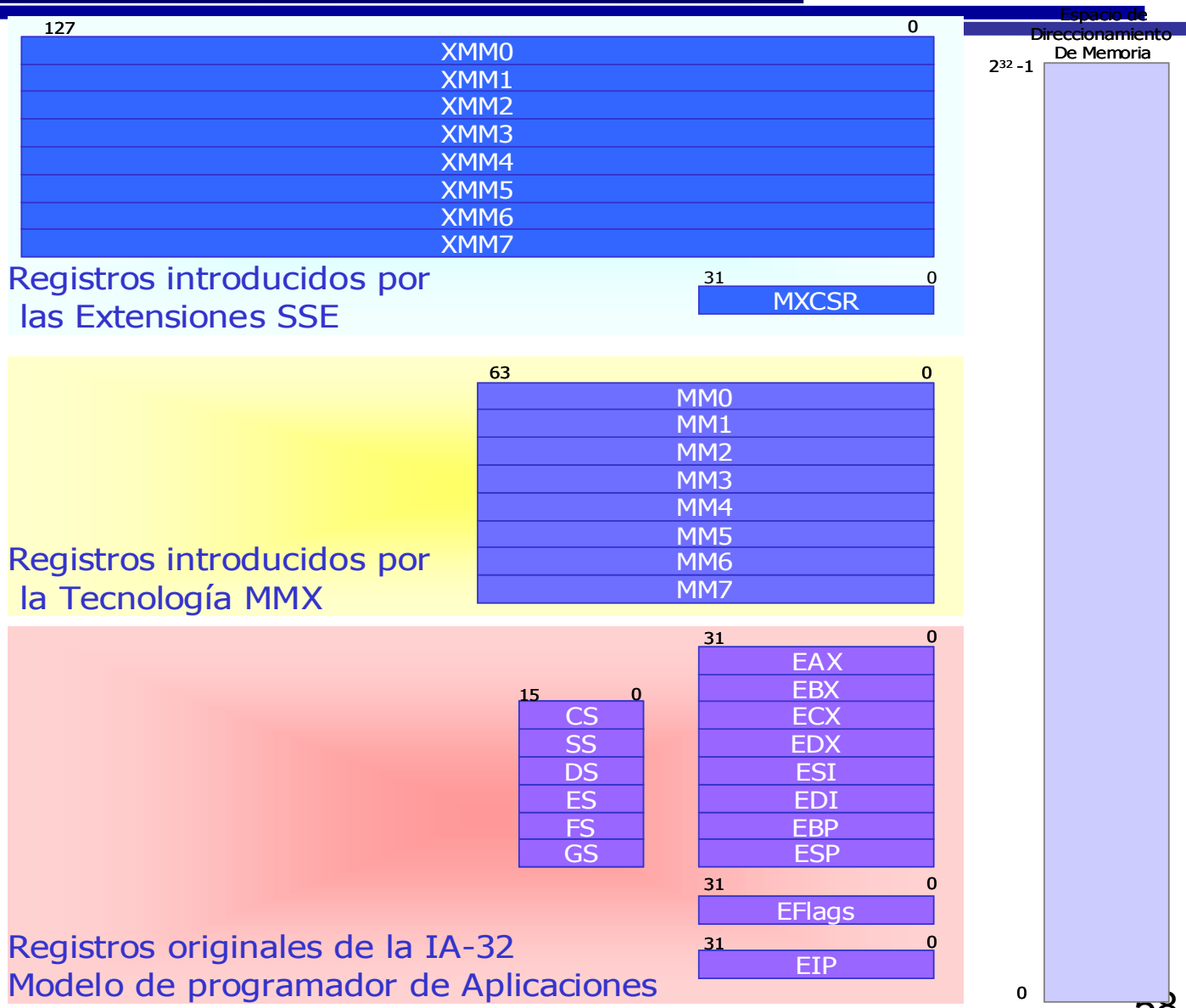
Redondeo

- El bit 15 (FZ) del registro MXCSR habilita el modo "flush-to-zero", que controla la respuesta a la condición de underflow de una instrucción SIMD de punto flotante.
- Cuando se enmascara la excepción de underflow excepción y se habilita el modo "flush-to-zero", el procesador realiza las siguientes operaciones cuando detecta una condición de underflow en punto flotante:
  - Retorna cero con el signo del resultado correcto
  - Pone en '1' los flags de las excepciones de precisión y de underflow

Redondeo

- Si no se enmascara la excepción de underflow, se ignora el bit "flush-to-zero". El modo "flush-to-zero" no es compatible con el Standard IEEE 754. La respuesta indicada por el IEEE a un underflow es entregar el resultado denormalizado. El modo "flush-to-zero" se provee en principio por razones de performance. Al costo de alguna pérdida de precisión, puede obtenerse una ejecución mas rápida para aplicaciones en donde la condición de underflow es común y es tolerable el redondeo a cero cuando se produce el underflow.
- El bit "flush-to-zero" se limpia durante el arranque o en el reset del procesador, y se deshabilita el modo "flush-to-zero".

SSE: Modelo completo de programación



Instrucciones SSE

- 4 Subgrupos:
  - Instrucciones SIMD single-precision floating-point que operan sobre los registros XMM.
  - Instrucciones de manejo de estado que operan sobre el registro MXSCR
  - Instrucciones SIMD con enteros de 64-bit que operan sobre los registros MMX.
  - Instrucciones de Cacheability control, prefetch, y de ordenamiento de instrucciones.

Instrucciones SSE SIMD Single-Precision Floating-Point

- Operan sobre valores de punto flotante single-precision, empaquetados o escalares
  - Instrucciones SSE de Transferencia de Datos
  - Instrucciones SSE de Aritmética Empaquetada
  - Instrucciones SSE de Comparación
  - Instrucciones SSE Lógicas
  - Instrucciones SSE de Shuffle y Desempaquetado
  - Instrucciones SSE de Conversión

Instrucciones de transferencia SSE

- **MOVAPS** Mueve cuatro valores alineados empaquetados de punto flotante single-precision entre registros XMM o entre un registro XMM y memoria
- **MOVUPS** Mueve cuatro valores no alineados empaquetados de punto flotante single-precision entre registros XMM o entre un registro XMM y memoria
- **MOVHPS** Mueve dos valores empaquetados de punto flotante single-precision floating-point hacia y desde la quadword alta de un registro XMM y memoria
- **MOVHLPS** Mueve dos valores empaquetados de punto flotante single-precision desde la quadword alta de un registro XMM a la quadword baja de otro registro XMM

Instrucciones de transferencia SSE

- **MOVLPS** Mueve dos valores de punto flotante empaquetados single-precision hacia o desde la quadword baja de un registro XMM y memoria
- **MOVLHPS** Mueve dos valores empaquetados de punto flotante single-precision desde la quadword baja de un registro XMM a la quadword alta de otro registro XMM
- **MOVMSKPS** Extrae la máscara de signo de cuatro valores empaquetados de punto flotante single-precision
- **MOVSS** Mueve un valor escalar de punto flotante single-precision entre registros XMM o entre un registro XMM y memoria

Autor: Ing. Alejandro Furfaro

Instrucciones SSE de Aritmética empaquetada

- **ADDPS** Suma valores de punto flotante empaquetados single-precision
- **ADDSS** Suma valores escalares de punto flotante single-precision
- **SUBPS** Resta valores de punto flotante empaquetados single-precision
- **SUBSS** Resta valores escalares de punto flotante single-precision
- **MULPS** Multiplica valores de punto flotante empaquetados single-precision
- **MULSS** Multiplica valores escalares de punto flotante single-precision

Instrucciones SSE de Aritmética empaquetada

- **DIVPS Divide valores de punto flotante empaquetados single-precision**
- **DIVSS Divide valores escalares de punto flotante single-precision**
- **RCPPS Computa el inverso de valores de punto flotante empaquetado single-precision**
- **RCPSS Computa el inverso de un valor escalar de punto flotante single-precision**
- **SQRTPS** Computa la raíz cuadrada de valores de punto flotante empaquetados single-precision
- **SQRTSS** Computa la raíz cuadrada de valores escalares de punto flotante single-precision

Instrucciones SSE de Aritmética empaquetada

- **RSQRTPS** Computa la inversa de la raíz cuadrada de valores empaquetados de punto flotante single-precision
- **RSQRTSS** Computa la inversa de la raíz cuadrada de valores escalares de punto flotante single-precision
- **MAXPS** Retorna el máximo de valores empaquetados de punto flotante single-precision
- **MAXSS** Retorna el máximo de un valor escalar de punto flotante single-precision
- **MINPS** Retorna el mínimo de valores empaquetados de punto flotante single-precision
- **MINSS** Retorna el mínimo de un valor escalar de punto flotante single-precision

Instrucciones SSE de Comparación

- **CMPPS** Compara valores empaquetados de punto flotante single-precision
- **CMPSS** Compara escalares de punto flotante single-precision
- **COMISS** Realiza una comparación ordenada de valores escalares de punto flotante single-precision y modifica flags en el registro EFLAGS
- **UCOMISS** Realiza comparación no ordenada de escalares de punto flotante single-precision y modifica los flags en el registro EFLAGS

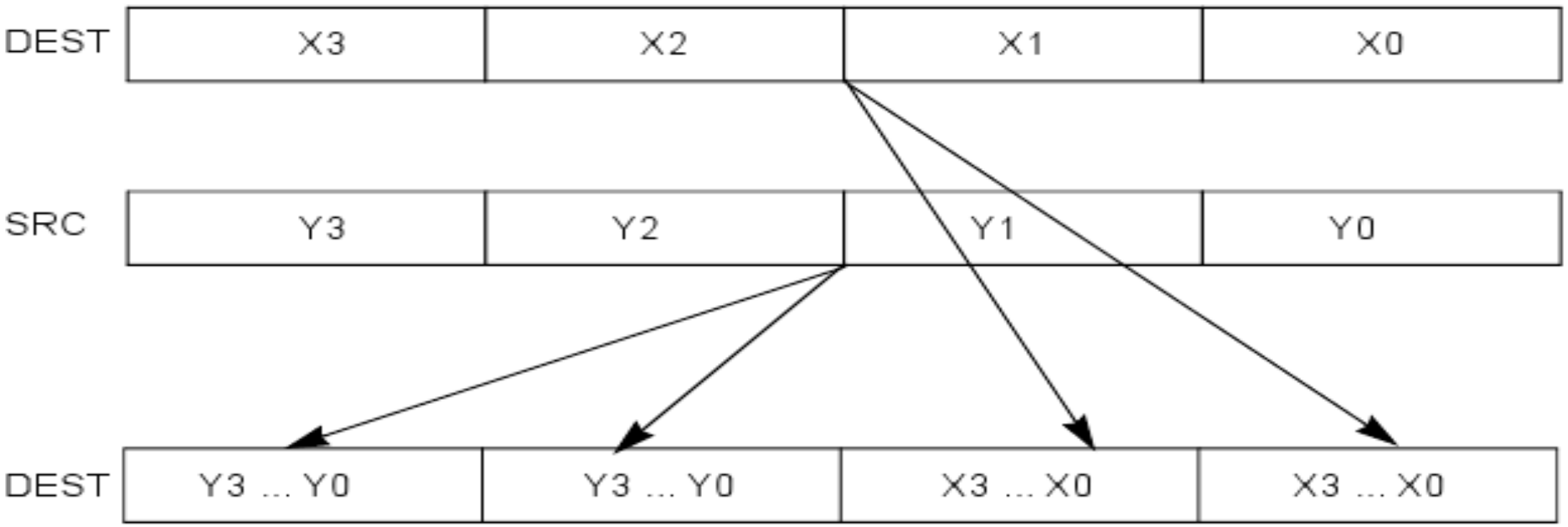
Instrucciones Lógicas SSE

Todas consideran a los operandos como un vector de bits. Se las denomina operaciones “bitwise”.

- **ANDPS** Realiza una AND lógica entre dos valores de punto flotante single-precision empaquetados
- **ANDNPS** Realiza una NAND lógica entre dos valores de punto flotante single-precision empaquetados
- **ORPS** Realiza una OR lógica entre dos valores de punto flotante single-precision empaquetados
- **XORPS** Realiza una XOR lógica entre dos valores de punto flotante single-precision empaquetados

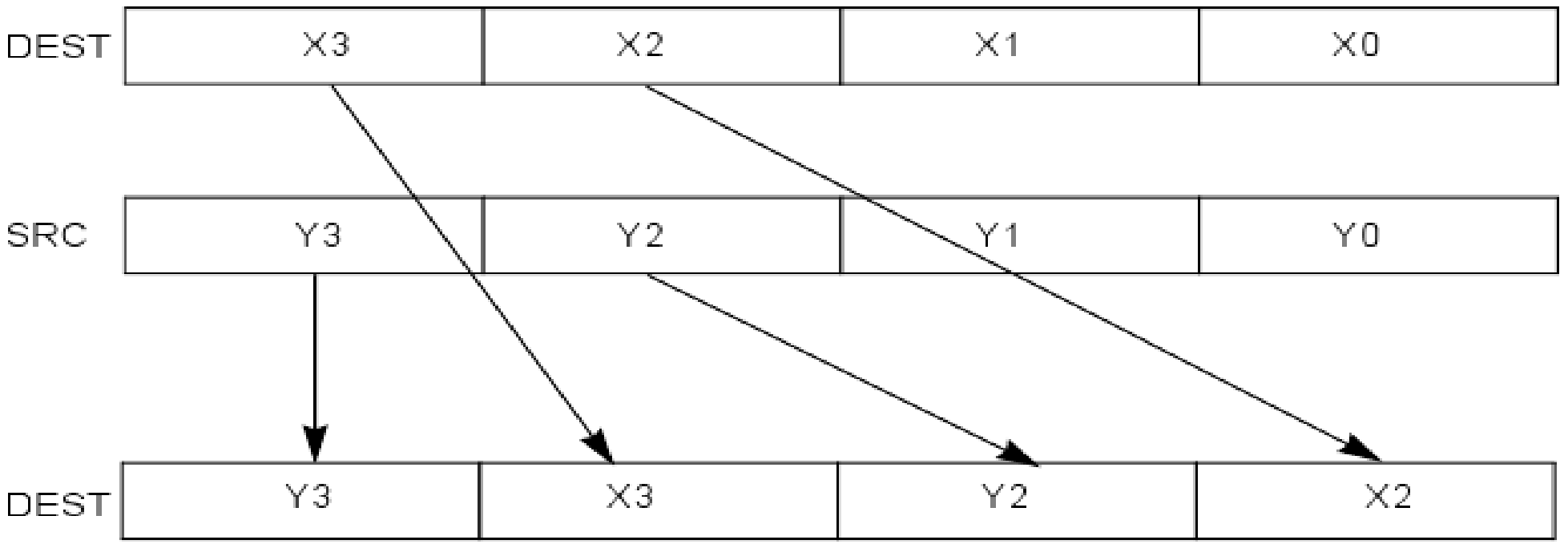
Instrucciones SSE de Shuffle y Unpack

- **SHUFPS** Cambia el orden (Shuffle) de los valores en operandos de punto flotante single-precision empaquetados



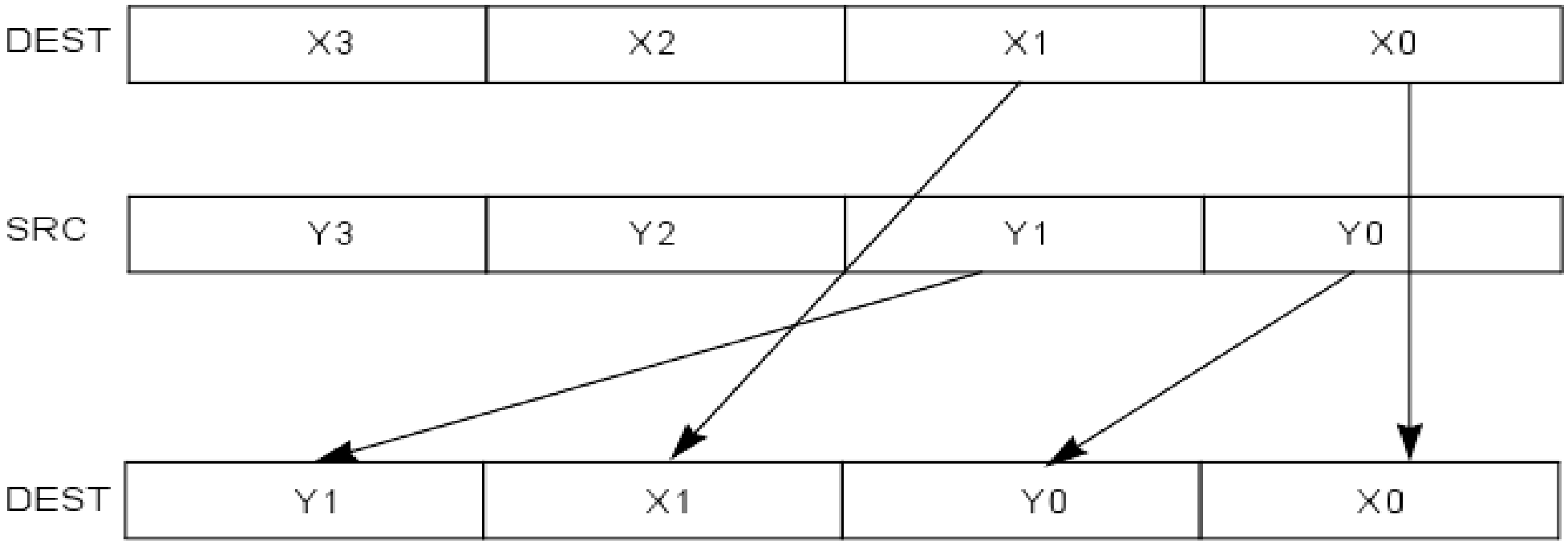
Instrucciones SSE de Shuffle y Unpack

- **UNPCKHPS** Desempaqueta y separa los dos valores de orden alto de dos operandos de punto flotante single-precision



Instrucciones SSE de Shuffle y Unpack

- **UNPCKLPS** Desempaqueta y separa los dos valores de orden bajo de dos operandos de punto flotante single-precision



Instrucciones SSE de Conversión

- **CVTPI2PS** Convierte enteros doubleword empaquetados a valores enteros single-precision floating-point empaquetados.
- **CVTSI2SS** Convierte enteros doubleword a valores escalares punto flotante single-precision.
- **CVTPS2PI** Convierte valores en punto flotante single-precision empaquetados a doubleword enteros empaquetados
- **CVTTPS2PI** Convierte con truncamiento valores en punto flotante single-precision a doubleword enteras empaquetadas
- **CVTSS2SI** Convierte un valor escalar de punto flotante single-precision a una doubleword entera.

Instrucciones SSE de Manejo de estado MXCSR

- **LDMXCSR** Carga el registro MXCSR
- **STMXCSR** Salva el estado del registro MXCSR

Instrucciones SSE para manejo de enteros SIMD de 64 bits

- **PAVGB** Calcula el promedio de bytes enteros sin signo empaquetados
- **PAVGW** Calcula el promedio de words enteras sin signo empaquetadas
- **PEXTRW** Extrae word
- **PINSRW** Inserta word
- **PMAXUB** Máximo de bytes enteros sin signo empaquetados.
- **PMAXSW** Máximo de words enteras signadas empaquetadas

Instrucciones SSE para manejo de enteros SIMD de 64 bits

- **PMINUB** Mínimo de bytes enteros sin signo empaquetados
- **PMINSW** Mínimo de words enteras signadas empaquetadas
- **PMOVMSKB** Mover byte de máscara
- **PMULHUW** Multiplica enteros empaquetados sin signo y almacena el resultado alto
- **PSADBW** Calcula la suma de diferencias absolutas
- **PSHUFW** Cambia de orden words enteras empaquetadas en registros MMX

Autor: Ing. Alejandro Furfaro

Instrucciones de Cacheability control, prefetch, y de ordenamiento de instrucciones

- **MASKMOVQ** Almacenamiento No-temporal de los bytes seleccionados de una quadword desde un registro MMX a memoria.
- **MOVNTQ** Almacenamiento No-temporal de una quadword desde un registro MMX a memoria
- **MOVNTPS** Almacenamiento No-temporal de cuatro valores de punto flotante single-precision desde un registro MMX a memoria
- **PREFETCHh** Carga 32 bytes (una línea) en el nivel caché especificada a partir de una dirección especificada, hasta un nivel especificado del cache
- **SFENCE** Serializa operaciones de Store

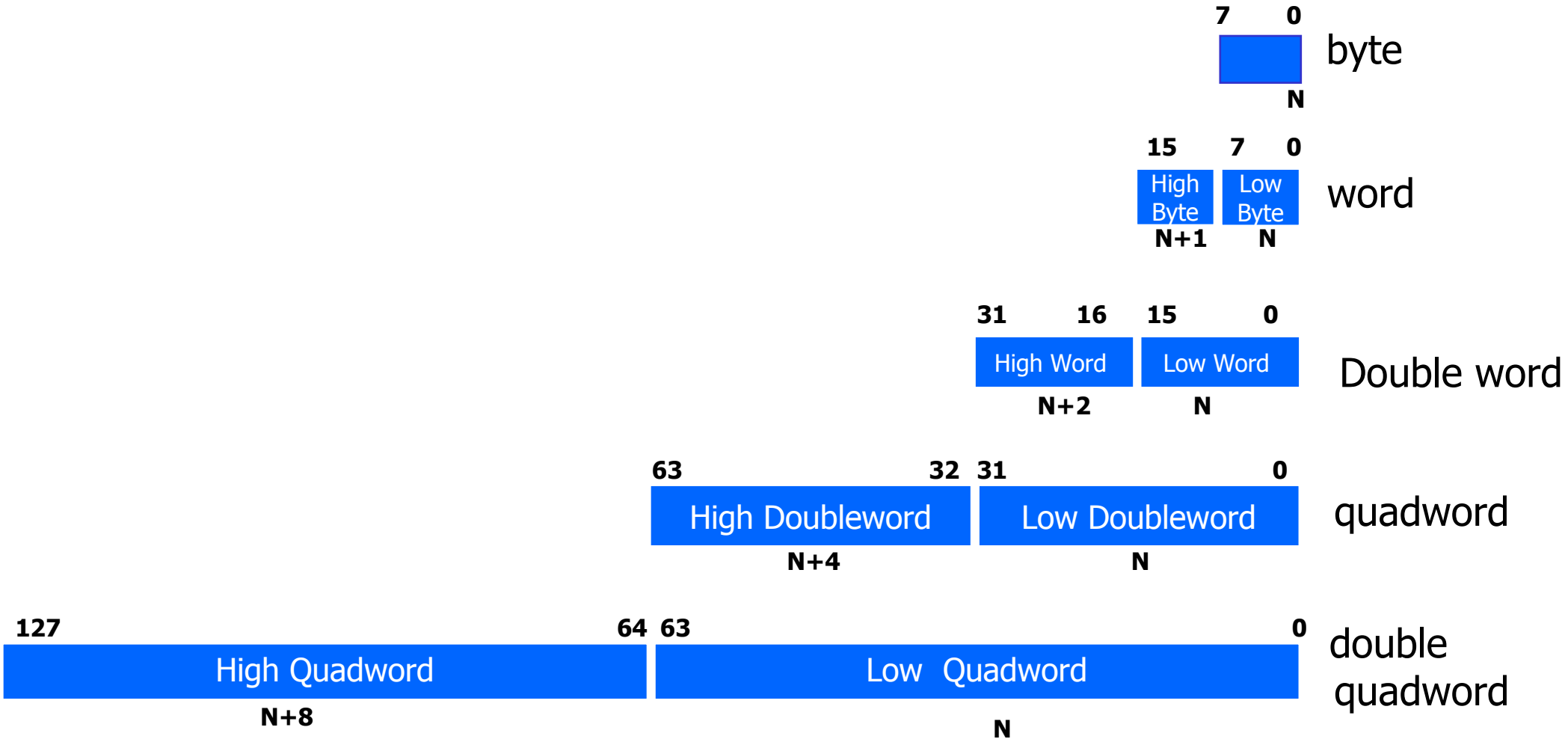
Streaming SIMD Extensions 2 (SSE2)

- Seis tipos de datos:
  - 128-bit packed double-precision floating-point (dos IEEE Standard 754 doble precisión)
  - Valores de punto flotante empaquetados en una doble quadword)
    - Bytes enteros empaquetados en 128 bits
    - Words enteras empaquetadas en 128 bits
    - doubleword enteras empaquetadas en 128 bits
    - quadword enteras empaquetadas en 128-bits
- Instrucciones de soporte para estos tipos de datos

Streaming SIMD Extensions 2 (SSE2)

- Operaciones:
  - Instrucciones sobre punto flotante doble precisión empaquetados y escalares
  - Instrucciones enteras SIMD adicionales de 64 y 128 bits
  - Versiones de 128 bits de las instrucciones SIMD enteras introducidas con la tecnología MMX
- Extensiones SSE
  - Instrucciones adicionales para control de cacheabilidad y ordenamiento de instrucciones

SSE, SSE2, SSE3: Tipos de datos



Streaming SIMD Extensions 3 (SS3)

- 13 instrucciones.
  - Diez instrucciones nuevas de soporte SIMD
  - Una para acelerar el estilo de programación de punto flotante x87 en conversiones a enteros
  - MONITOR y MWAIT aceleran la sincronización de threads.

Resultados Convolución SSE

C	ASM	SSE
1014762	227668	199388
1016344	227724	194124
1018570	227710	194656
1015728	227612	199318
1015434	227710	194572
1016167,6	227684,8	196411,6
4,46	1,00	0,86

Resultados Detección de Bordes SSE

Bordes		
C	ASM	SSE
14269038	9461102	182518
14269584	9461564	182518
14271390	9461494	182518
14271138	9461326	182532
14270690	9461746	182490
14269864	9461438	182504
14270284	9461445	182513,333
1,51	1,00	0,02

Medición de performance

- Recurso Registro tsc (MSR, a partir del procesador Pentium)
- 64 bits
- Se incrementa cada ciclo de clock
- Instrucción rdtsc

  [edx:eax] <- [tsc]
- ¿Como lo usamos en C?
  - In line Assembly

Autor: Ing. Alejandro Furfaro

Referencias

Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1

Capítulos 2, 4, 9, 10, 11, 12

Organización y Arquitectura de Computadores. 5ta. Ed. William Stallins.

Capítulos Apéndice III.