Convención de llamada linux x86_64

Parámetros y valores de retorno 64 bits

- **Enteros y punteros**: RDI, RSI, RDX, RCX, R8, R9
- **Flotantes**: XMM0, ... , XMM7
- **Retorno**: RAX, XMM0
- **Temporales**: RAX, R10, R11, XMM8, ..., XMM15, st2, ..., st7, k0, ..., k7
- **long doubles (temporales)**: st0, st1

No volatiles: RBX, RBP, R12, R13, R14, R15

Las funcioens llamadas si quieren modificar registros no volatiles tienen la obligación (por convención) de restaurarlos al terminar.

Los parametros que entran por registros se pasan de izquierda a derecha. Los que no alcanzan a entrar, se pasan por stack de derecha a izquierda (viendolo desde la declaración de la función).

Para llamadas a funciones de C, se necesita la pila alineada a 16 bytes (en 32 bits también)

Parámetros y valores de retorno 32 bits

- Todos los parámetros se pasan por pila (de derecha a izquierda)
- **Retorno**: EAX
- **No volatiles**: EBX, EBP, ESI, EDI

Modos de acceso a memoria

- [inmediato]
- [registro]
- [registro + registro*escala] siendo escala 1, 2, 4 u 8
- [registro + inmediato]
- [reg + reg*escala + inm]

Registros de Propósito General

Intel 64

63 ... 0
rax
rbx
rcx
rdx
rsi
rdi
rbp
rsp
r8
r9
r10
r11
r12
r13
r14
r15

Nombres para acceder a los bits del registro en las posiciones				
63-0 (64 bits)	31-0 (32 bits)	15-0 (16 bits)	15-8 (8 bits)	7-0 (8 bits)
rax	eax	ax	ah	al
rbx	ebx	bx	bh	bl
rcx	ecx	cx	ch	cl
rdx	edx	dx	dh	dl
rsi	esi	si		sil
rdi	edi	di		dil
rbp	ebp	bp		bpl
rsp	esp	sp		spl
r8	r8d	r8w		r8b
r9	r9d	r9w		r9b
r10	r10d	r10w		r10b
r11	r11d	r11w		r11b
r12	r12d	r12w		r12b
r13	r13d	r13w		r13b
r14	r14d	r14w		r14b
r15	r15d	r15w		r15b

Alineación de structs

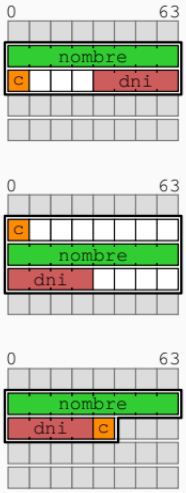
- Cada variable debe estar alineada a una pocisión multiplo de su tamaño.

- El tamaño de la estructura debe estar alineado al tamaño del atributo más grande
- En ambos casos se agrega padding para rellenar (se puede sacar con **___attribute___((*__packed__*))**)

```
struct alumno {
 char* nombre;     → 8     ⇒ 0
 char comision;    → 1     ⇒ 8
 int dni;          → 4     ⇒ 12
};                         ⇒ 16

struct alumno2 {
 char comision;    → 1     ⇒ 0
 char* nombre;     → 8     ⇒ 8
 int dni;          → 4     ⇒ 16
};                         ⇒ 24

struct alumno3 {
 char* nombre;     → 8     ⇒ 0
 int dni;          → 4     ⇒ 8
 char comision;    → 1     ⇒ 12
} __attribute__((packed));  ⇒ 13
```



Interacción con C

- Las funciones exportadas se deben declarar en la sección .text con ***global func***
- Las funciones de C llamadas desde ASM se deben declararen .text con ***extern func***

Secciones del código

- **.data**: variables globales inicializadas (DB: define byte, DW: word, DD: double word, DQ: quad word)
- **.rodata**: constantes globales inicializadas (DB, DW, DD, DQ)
- **.bss**: variables globales no inicializadas (RESB, RESW, RESD, RESQ) (reserve)
- **.text**: codigo

Dentro de *.text* la etiqueta _start sería el equivalente a la función *main*

Para ensamblar un mismo valor repetido: *"etiqueta"* **times** *"numero"* DB/BW/DD/DQ *"hexa/entero/binario/octal"*

En general las instrucciones son registro-registro; registro-memoria; registro-inmediato; memoria-registro; memoria-inmediato

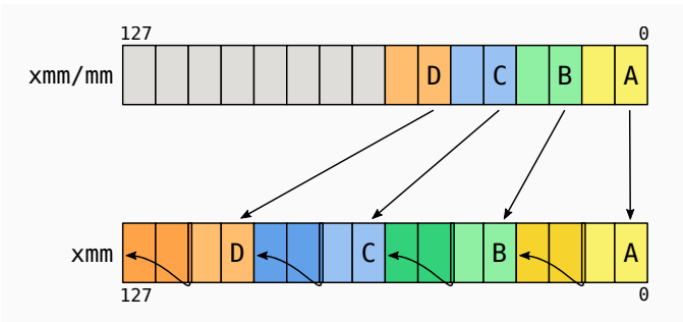
SIMD

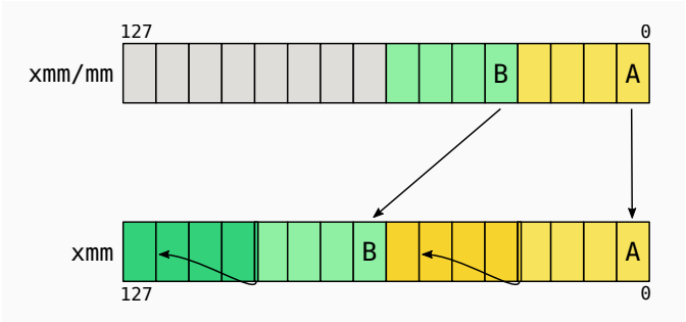
MOV xmm-m m-r

MOVD	MOVQ	Move Doubleword/Quadword
MOVSS	MOVSD	Moves a 32bits Single FP/64bits Double FP
MOVDQA	MOVDQU	Moves aligned/unaligned double quadword
MOVAPS	MOVUPS	Moves 4 aligned/unaligned 32bit singles
MOVAPD	MOVUPD	Moves 2 aligned/unaligned 64bit doubles

Packed MOV xmm-xmm xmm-m

PMOVSXBW	PMOVZXBW	packed sign/zero extension byte to word
PMOVSXBD	PMOVZXBD	packed sign/zero extension byte to dword
PMOVSXBQ	PMOVZXBQ	packed sign/zero extension byte to qword
PMOVSXWD	PMOVZXWD	packed sign/zero extension word to dword
PMOVSXWQ	PMOVZXWQ	packed sign/zero extension word to qword
PMOVSXDQ	PMOVZXDQ	packed sign/zero extension dword to qword

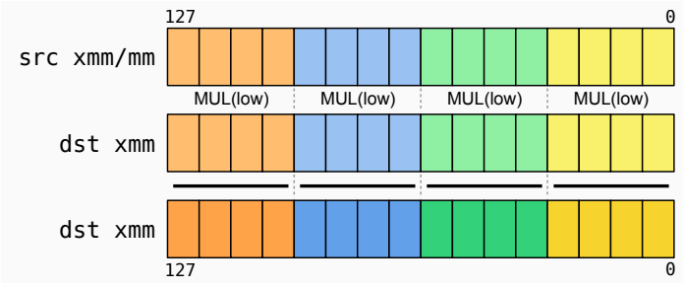




Packed operaciones aritmeticas xmm-xmm xmm-m

PADDB	PADDW	PADDD	PADDQ	Add Integer
PSUBB	PSUBW	PSUBD	PSUBQ	Sub Integer
PMULHW	PMULLW			Mul Integer Word
PMULHD	PMULLD			Mul Integer Dword
PMINSB	PMAXSB	PMINUB	PMAXUB	Max and Min Integer
PMINSW	PMAXSW	PMINUW	PMAXUW	Max and Min Integer
PMINSD	PMAXSD	PMINUD	PMAXUD	Max and Min Integer

Notar que pmul tiene low y high, con low se guarda el resultado de la parte baja o alta (al multiplicar en el peor caso se necesita el doble de bits)



PABSB	Absolute for 8 bit Integers
PABSW	Absolute for 16 bit Integers
PABSD	Absolute for 32 bit Integers

Packed operaciones fp xmm-xmm xmm-m

ADDPS	ADDSS	ADDPD	ADDSD	Addition of FP values
SUBPS	SUBSS	SUBPD	SUBSD	Subtraction of FP values
MULPS	MULSS	MULPD	MULSD	Multiply of FP values
DIVPS	DIVSS	DIVPD	DIVSD	Divition of FP values
MAXPS	MAXSS	MINPS	MINSS	Max and Min of Single FP values
MAXPD	MAXSD	MINPD	MINSD	Max and Min of Double FP values

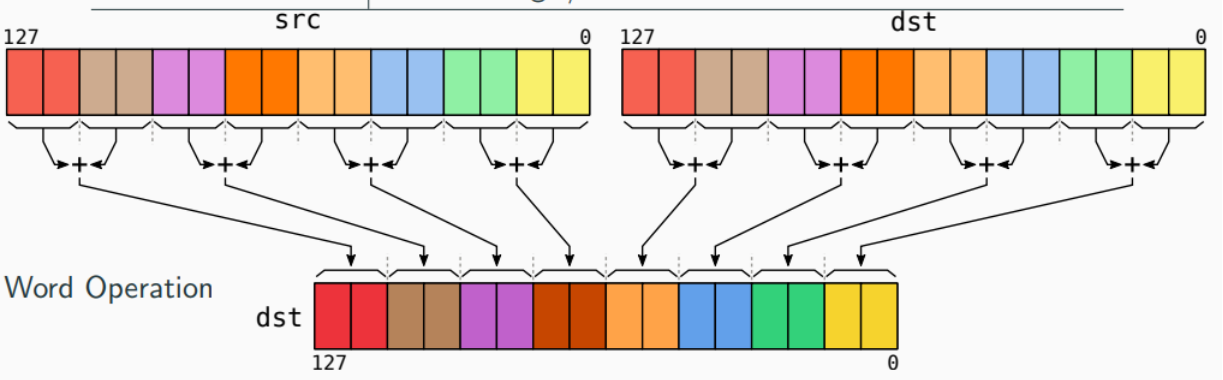
SQRTSS	SQRTPS	Square root of Scalar/Packed Single FP values
SQRTSD	SQRTPD	Square root of Scalar/Packed Double FP values

Packed operaciones saturadas con enteros xmm-xmm xmm-m

PADDSB	PADDSW	Add Int saturation
PADDUSB	PADDUSW	Add Int unsigned saturation
PSUBSB	PSUBSW	Sub Int saturation
PSUBUSB	PSUBUSW	Sub Int unsigned saturation

Packed operaciones horizontales xmm-xmm xmm-m

PHADDW	PHADDD	Horizontal addition of unsigned 16bit/32bit integers
PHADDSW		Horizontal saturated addition of 16bit integers
PHSUBW	PHSUBD	Horizontal subtraction of unsigned 16bit/32bit integers
PHSUBSW		Horizontal saturated subtraction of 16bit words
HADDPS	HADDPD	Packed Single/Double FP Horizontal Add
HSUBPS	HSUBPD	Packed Single/Double FP Horizontal Subtract



Packed operaciones lógicas y shifts xmm-xmm xmm-m

PAND	PANDN	POR	PXOR	Operaciones lógicas para enteros.
ANDPS	ANDNPS	ORPS	XORPS	Operaciones lógicas para *float*.
ANDPD	ANDNPD	ORPD	XORPD	Operaciones lógicas para *double*.

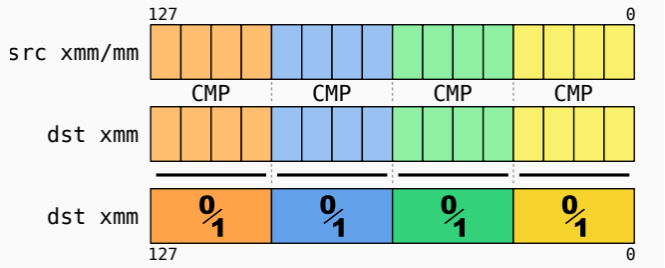
- Actuan lógicamente sobre todo el registro, sin importa el tamaño del operando.
- La distinción entre PS y PD se debe a meta información para el procesador.

PSLLW	PSLLD	PSLLQ	PSLLDQ*
PSRLW	PSRLD	PSRLQ	PSRLDQ*
PSRAW	PSRAD		

- Todos los *shifts* operan de forma lógica como aritmética, tanto a derecha como izquierda.
- Se limitan a realizar la operación sobre cada uno de los datos dentro del registro según su tamaño.
- * En las operaciones indicas, el parámetro es la cantidad de bytes del desplazamiento.

Packed compare enteros y flotantes xmm-xmm xmm-m

PCMPEQB	PCMPEQW	PCMPEQD	PCMPEQQ	Compare Packed Data for Equal
PCMPGTB	PCMPGTW	PCMPGTD	PCMPGTQ	Compare Packed Signed Int for Greater Than



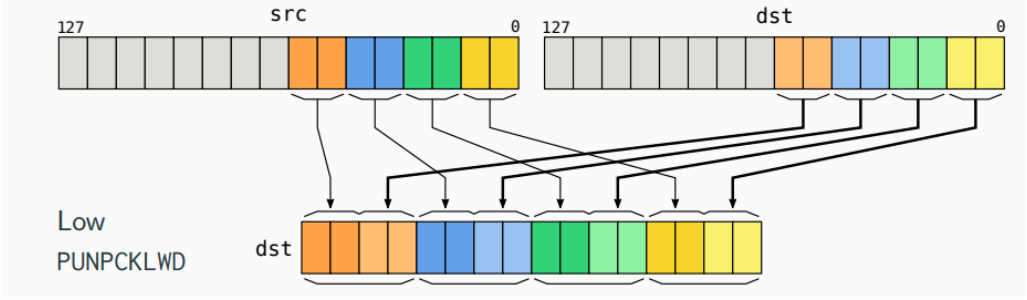
CMPxxPD	Compare Packed Double-Precision Floating-Point Values
CMPxxPS	Compare Packed Single-Precision Floating-Point Values
CMPxxSD	Compare Scalar Double-Precision Floating-Point Values
CMPxxSS	Compare Scalar Single-Precision Floating-Point Values
COMISD	Compare Scalar Ordered Double-Precision Floating-Point Values and Set EFLAGS
COMISS	Compare Scalar Ordered Single-Precision Floating-Point Values and Set EFLAGS

	Acción	xx	CMPxxyy A, B
0	Igual	EQ	$A = B$
1	Menor	LT	$A < B$
2	Menor o Igual	LE	$A \leqslant B$
3	No Orden	UNORD	$A, B = unordered$
4	Distinto	NEQ	$A \neq B$
5	No Menor	NLT	$not(A < B)$
6	No Meno o Igual	NLE	$not(A \leqslant B)$
7	Orden	ORD	$A, B = Ordered$

Desenpaquetado

Notar que hay para tomar los lows y highs

PUNPCKLBW	PUNPCKHBW	Unpacks 8 enteros de 8 bits en words
PUNPCKLWD	PUNPCKHWD	Unpacks 4 enteros de 16 bits en dwords
PUNPCKLDQ	PUNPCKHDQ	Unpacks 2 enteros de 32 bits en qwords
PUNPCKLQDQ	PUNPCKHQDQ	Unpacks 1 entero de 64 bits en 128 bits
UNPCKLPS	UNPCKHPS	Unpacks Single FP
UNPCKLPD	UNPCKHPD	Unpacks Double FP



Shuffles xmm-xmm xmm-m128

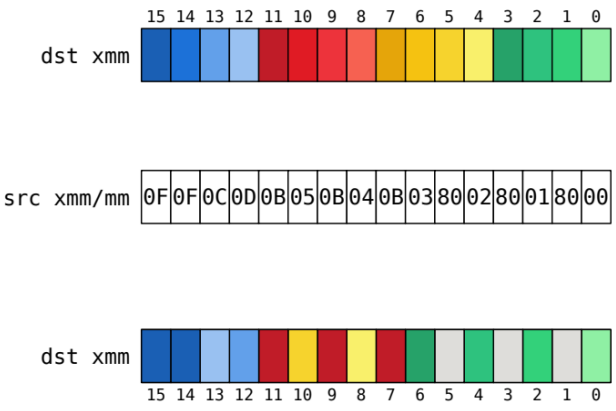
Las instrucciones de *Shuffle* permiten **reordenar** datos en registros.
Sus parámetros serán el **registro a reordenar** y una **máscara** que indicará cómo hacerlo.

- PSHUFB - Shuffle Packed Bytes
- PSHUFHW - Shuffles high 16bit values
- PSHUFLW - Shuffles low 16bit values
- PSHUFD - Shuffle Packed Doublewords
- SHUFPS - Shuffle Packed Single FP Values
- SHUFPD - Shuffle Packed Double FP Values

Las máscaras negativas dejan en 0 el paquete

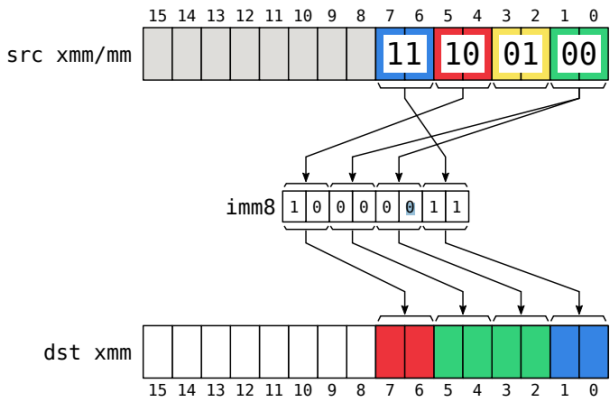
packed shuffle bytes xmm-xmm-imm8 xmm-m128-imm8

```
PSHUFB dst, src
```



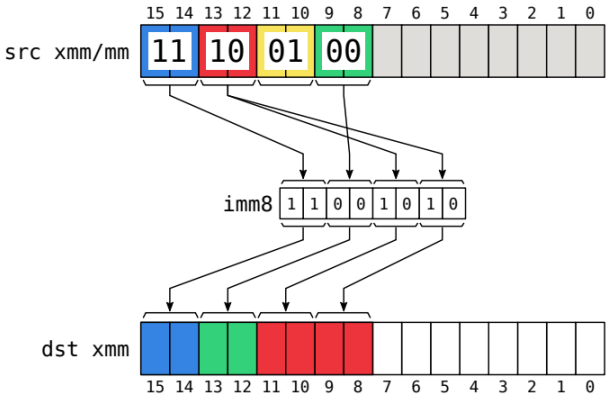
packed shuffle low words xmm-xmm-imm8 xmm-m128-imm8

·PSHUFLW dst, src , imm8



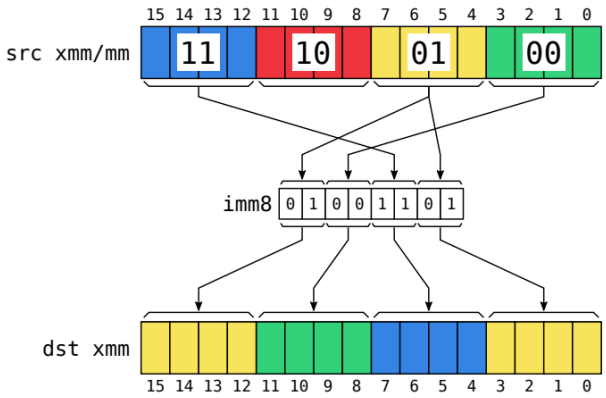
packed shuffle high words xmm-xmm-imm8 xmm-m128-imm8

·PSHUFHW dst, src , imm8



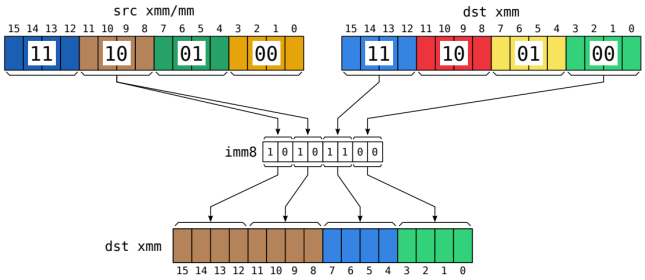
packed shuffle double words xmm-xmm-imm8 xmm-m128-imm8

·PSHUFD dst, src , imm8



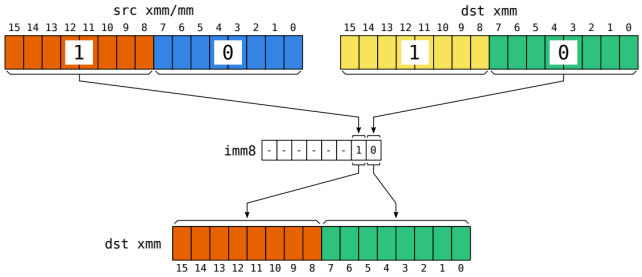
shuffle singles floating points xmm-xmm-imm8 xmm-m128-imm8

Ejemplo - SHUFPS dst, src , imm8



shuffle doubles floating points xmm-xmm-imm8 xmm-m128-imm8

Ejemplo - SHUFPD dst, src , imm8



Insert y extract

Insert/Extract

Las instrucciones de *Insert* y *Extract*, permiten como su nombre lo indica, **insertar** y **extraer** valores dentro de un registro.

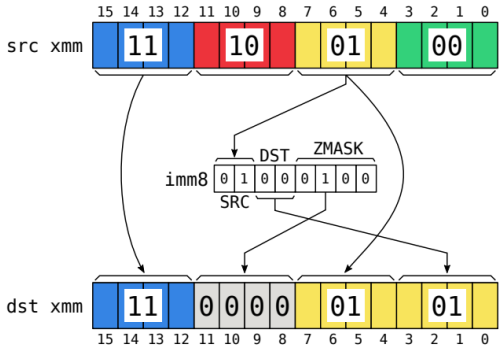
- INSERTPS - Insert Packed Single FP Value
- EXTRACTPS - Extract Packed Single FP Value
- PINSRB - Insert Byte
- PINSRW - Insert Word
- PINSRD - Insert Dword
- PINSRQ - Insert Qword
- PEXTRB - Extract Byte
- PEXTRW - Extract Word
- PEXTRD - Extract Dword
- PEXTRQ - Extract Qword

Insert Packed Single Precision floating point xmm-xmm-i, r-m32-imm8

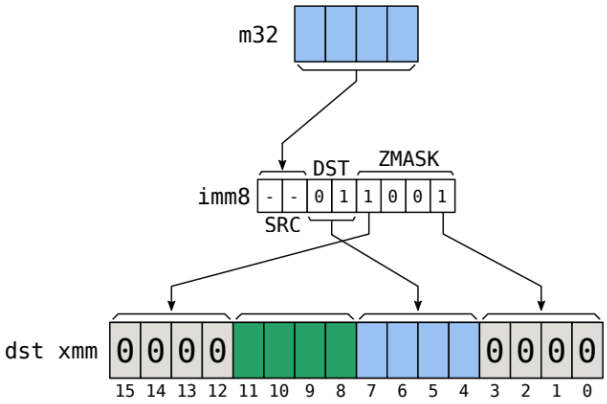
El inmediato me indica en “dst” el lugar en donde se copiará el punto flotante, “src” cuál elemento del xmm src se copia (si es memoria de 32 bits, se ignora ya que solo hay una cosa para copiar) y el “zmask” indica que bloques poner

en 0. Los bloques que no se escriben se mantienen como estaban.

Ejemplo - INSERTPS dst, src , imm8



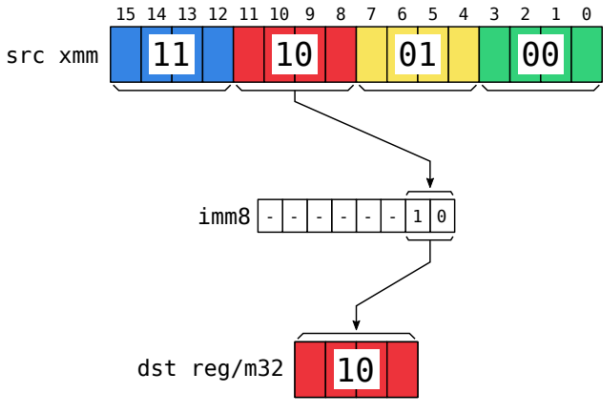
· INSERTPS dst, src , imm8



Extract Packed Single Precision floating point r32/r64/m32-xmm-imm8

Si recibe un registro de 64 bits, pone en 0 los demas bits

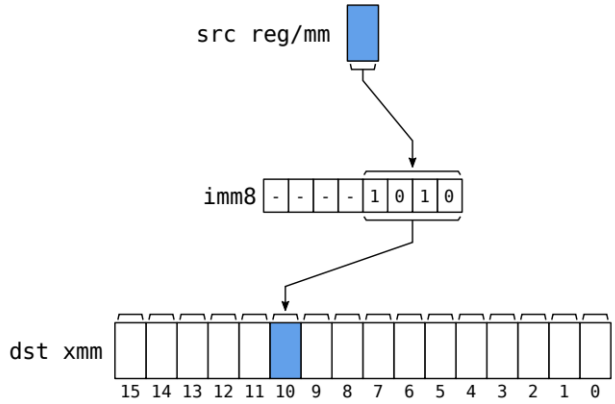
EXTRACTPS dst, src , imm8



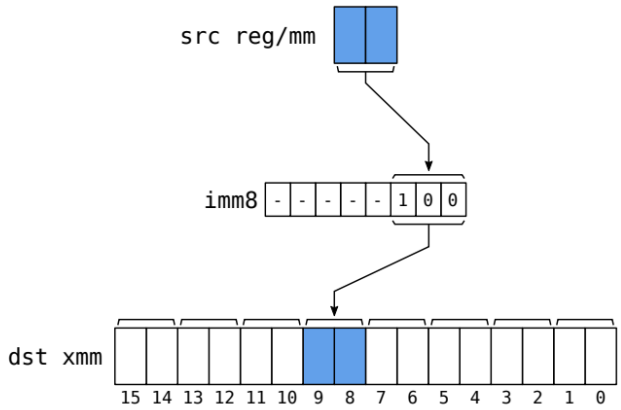
Packed Insert Byte/Word/Dword/Qword xmm-r/m8/16/32/64-imm8

Se inserta en el registro xmm un byte/word/dword/qword del registro o memoria del src en la pocisión especificada por el inmediato

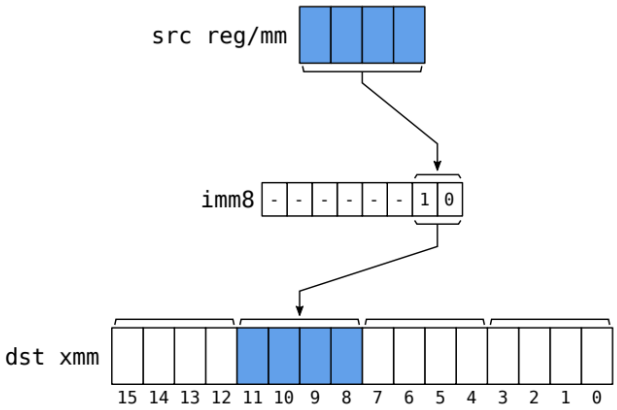
PINSRB dst, src , imm8



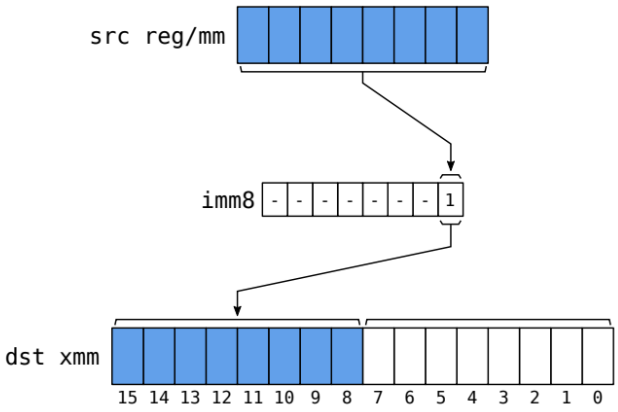
PINSRW dst, src , imm8



PINSRD dst, src , imm8



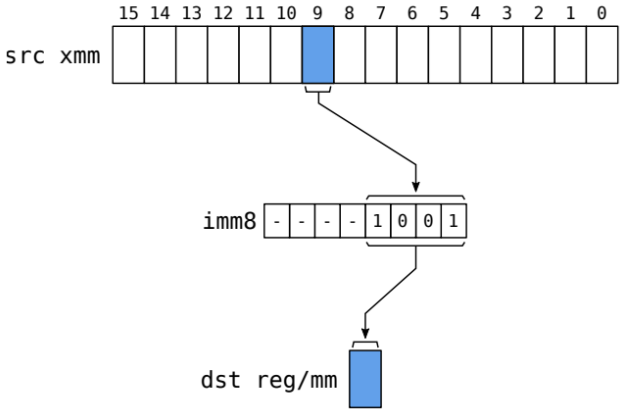
PINSRQ dst, src , imm8



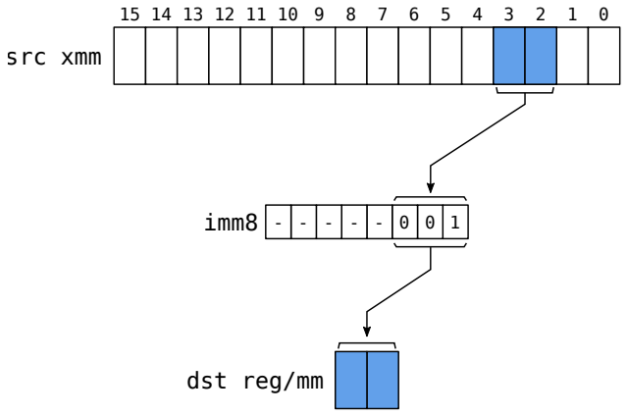
Packed Extract Byte/Word/Dword/Qword r/m8/16/32/64-xmm-imm8

Toma el byte/word/dword/qwrod del bloque especificado por el inmediato del registro xmm y lo pone en el registro o momeria. Para los registros, los bits restantes los pone en 0.

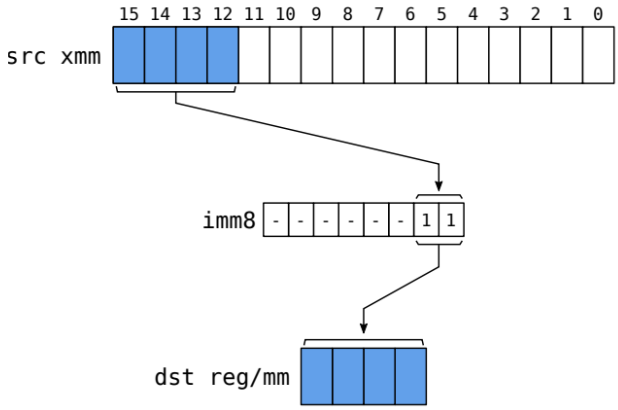
PEXTRB dst, src , imm8



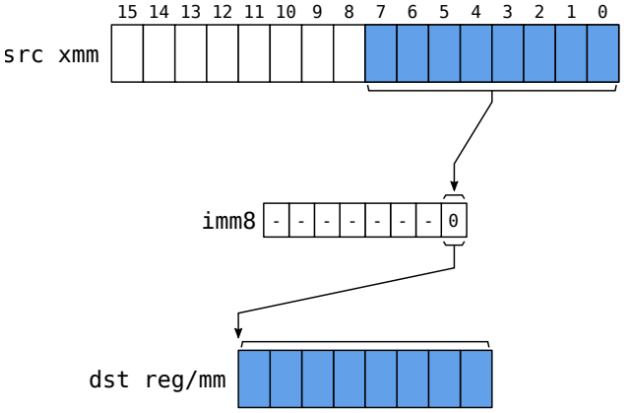
PEXTRW dst, src , imm8



PEXTRD dst, src , imm8



PEXTRQ dst, src , imm8



Blend (mezclar registros)

Blend

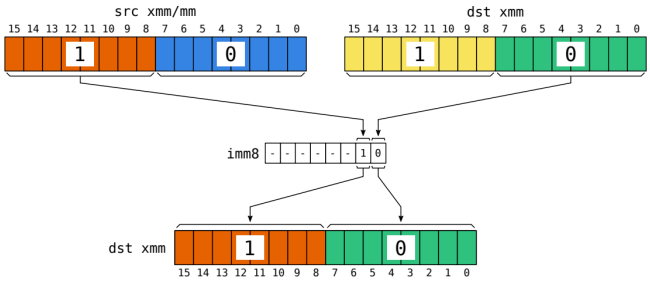
Las instrucciones de *Blend* permiten **mezclar** registros dependiendo del valor de sus datos.
Usando tanto inmediatos como otros registros.

- BLENDPS - Blend Packed Single FP Values
- BLENDPD - Blend Packed Double FP Values
- BLENDVPS - Variable Blend Packed Single FP Values
- BLENDVPD - Variable Blend Packed Double FP Values
- PBLENDW - Blend Packed Words
- PBLENDVB - Variable Blend Packed Bytes

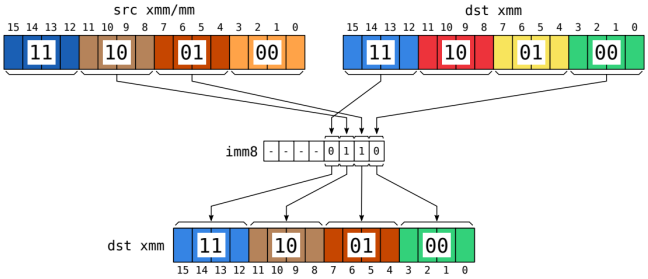
Blend Packed SP y DP xmm-xmm/m128-imm8

El inmediato sirve para ir eligiendo de que registro elegir. Cada bit representa un bloque en el destino, si vale 0 se elige del "dst", si es 1 se elige de "src".

Ejemplo-BLENDPD dst, src , imm8



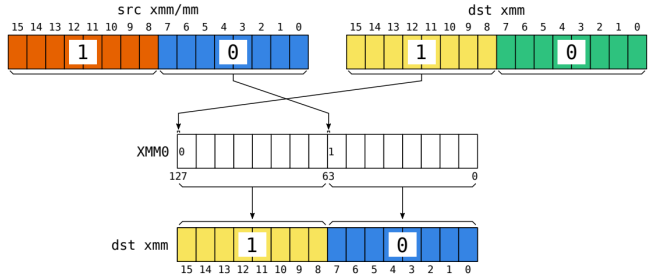
Ejemplo-BLENDPS dst, src , imm8



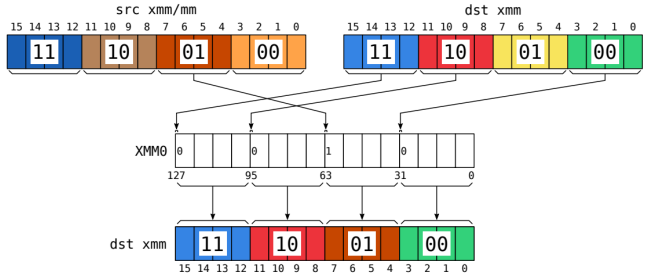
Variable Blend Packed SP y DP xmm-xmm/m128-

Misma idea que el anterior pero se usa el registro xmm0 en vez de un inmediato, pero se fija en si el numero es positivo o negativo (el bit más significativo de cada bloque)

Ejemplo-BLENDVPD dst, src



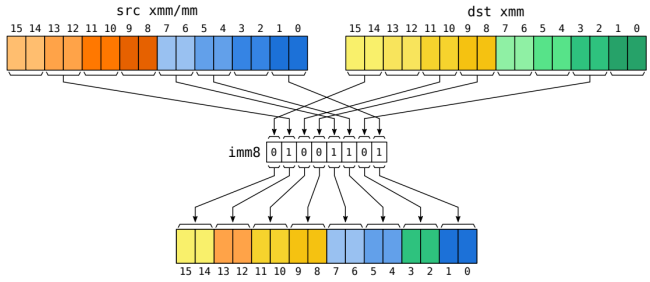
Ejemplo-BLENDVPS dst, src



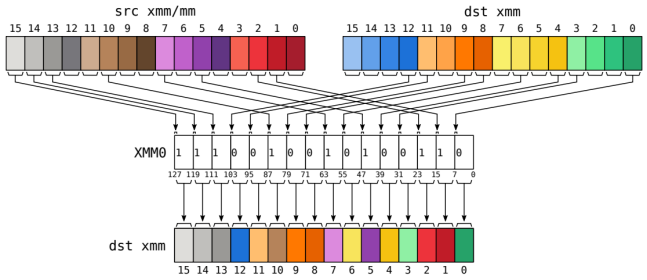
Blend Packed Words/bytes xmm-xmm/m128-imm8

Misma idea de mezclar según el bit del bloque representado por el inmediato. Si es 1 se toma la word en dicha posición en “src” y 0 en “dst”

Ejemplo-PBLENDW dst, src , imm8



Ejemplo-PBLENDVB dst, src



Conversiones float-double/int-float y truncado

Conversiones

Instrucciones entre enteros y punto flotante

- CVTSI2SS - Dword Integer to Scalar Single FP → `CVTSI2SS xmm, r/m32`
- CVTSS2SI - Scalar Single FP to Dword Integer → `CVTSS2SI r32, xmm/m32`
- CVTSI2SD - Dword Integer to Scalar Double FP → `CVTSI2SD xmm, r/m64`
- CVTSD2SI - Scalar Double FP to Dword Integer → `CVTSD2SI r64, xmm/m64`
- CVTDQ2PS - Packed Dword Integers to Packed Single FP (4X) → `CVTDQ2PS xmm1, xmm2/m128`
- CVTPS2DQ - Packed Single FP to Packed Dword Integers (4X) → `CVTPS2DQ xmm1, xmm2/m128`
- CVTDQ2PD - Packed Dword Integers to Packed Double FP (2X) → `CVTDQ2PD xmm1, xmm2/m64`
- CVTPD2DQ - Packed Double FP to Packed Dword Integers (2X) → `CVTPD2DQ xmm1, xmm2/m128`

Conversiones

Instrucciones de redondeo

- ROUNDSS - Round Scalar Single FP to Integer → `ROUNDSS xmm1, xmm2/m32, imm8`
- ROUNDSD - Round Scalar Double FP to Integer → `ROUNDSD xmm1, xmm2/m64, imm8`
- ROUNDPS - Round Packed Single FP to Integer (4X) → `ROUNDPS xmm1, xmm2/m128, imm8`
- ROUNDPD - Round Packed Double FP to Integer (2X) → `ROUNDPD xmm1, xmm2/m128, imm8`

El parámetro inmediato indica el tipo de redondeo.

Instrucciones de truncado

- CVTTSS2SI - Truncation Scalar Single FP to Dword Integer (1X) → `CVTTSS2SI r32, xmm/m32`
- CVTTSD2SI - Truncation Scalar Double FP to Signed Integer (1X) → `CVTTSD2SI r32, xmm/m64`
- CVTTPS2DQ - Truncation Packed Single FP to Packed Dword Int. (4X) → `CVTTPS2DQ xmm1, xmm2/m128`

Conversiones

Las instrucciones de conversión son de la forma: CVTxx2yy

Donde xx e yy pueden valer:

ps - Packed Single FP	pd - Packed Double FP	pi - Packed Integer
ss - Scalar Single FP	sd - Scalar Double FP	si - Scalar Integer
		dq - Packed Dword

Instrucciones solo de punto flotante

- CVTSD2SS - Scalar Double FP to Scalar Single FP (1X) → `CVTSD2SS xmm1, xmm2/m64`
- CVTSS2SD - Scalar Single FP to Scalar Double FP (1X) → `CVTSS2SD xmm1, xmm2/m32`
- CVTPD2PS - Packed Double FP to Packed Single FP (2X) → `CVTPD2PS xmm1, xmm2/m128`
- CVTPS2PD - Packed Single FP to Packed Double FP (2X) → `CVTPS2PD xmm1, xmm2/m64`

Comandos utiles de gdb

comando	descripcion
b < archivo >:< linea >	pone un break en el archivo y linea especificada
info breaks	lista todos los breakpoints
delete n	elimina el breakpoint n
r	corre el programa
n	ejecuta la siguiente linea
c	continua la ejecución
gdb –args < archivo > arg1 arg2 arg3	ejecuta el archivo pasandole argumentos
info registers < registro >	devuelve el valor del registro, podemos pasarle tambien los eflags
info locals	lista las variables del stack frame actual
info args	lista los argumentos del stack frame actual
backtrace	imprime el backtrace del stack entero
backtrace n	idem pero solo los n ultimos stack frames
backtrace -n	idem pero los primeros n stack frames
backtrace full	imprime el backtrace del stack entero con sus variables locales
up / up n	se mueve al stack frame superior (o n stack frames arriba) (para imprimir variables locales de otro stack frame si se quiciera)
down / down n	se mueve al stack frame inferior (o n stack frames abajo)
x < addr >	imprime el valor de memoria especificado por la dirección addr
x/nfu < addr >	idem pero puedo especificar la cantidad n de bloques en unidad “u” a imprimir; el formato f; la unidad u que puede ser b (byte), h (halfword, 2 bytes), w (word 4 bytes), g (giant words, 8 bytes). mas info
print < expresion >	imprime el valor de la expresión
print /f < expresion >	idem con el formato especificado
display < expresion >	habilita la impresión automática por cada paso del debuger
display/f < expresion >	idem con el formato especificado
undisplay < expresion >	deshabilita la impresión automática de la expresión

formatos

- o: octal

- x: hexadecimal
- d: decimal
- u: unsigned decimal
- t: binary
- f: floating point
- a: address
- c: char
- s: string
- i: instruction