

Práctica de Organización del Computador II

SIMD

Organización del Computador II
DC - UBA

Primer Cuatrimestre 2023

- Vamos a resolver algoritmos utilizando **instrucciones vectoriales**.
- Debemos **conocer** las instrucciones que tenemos disponibles.
- y las **técnicas** para pensar algoritmos desde la operatoria vectoriales.

- Registros:

XMM0 a XMM15 de 128 bits (16 bytes)

- Tipos de datos:

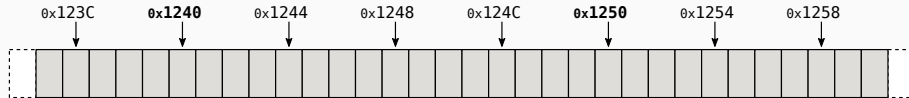
Enteros: 8, 16, 32, 64 y 128.

Float: 32 (Float) y 64 (Double).

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

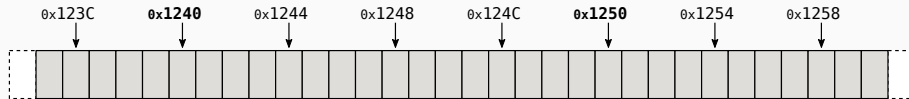
MOVD	MOVQ	Move Doubleword/Quadword
MOVSS	MOVSD	Moves a 32bits Single FP/64bits Double FP
MOVDQA	MOVDQU	Moves aligned/unaligned double quadword
MOVAPS	MOVUPS	Moves 4 aligned/unaligned 32bit singles
MOVAPD	MOVUPD	Moves 2 aligned/unaligned 64bit doubles

Ejemplo:



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

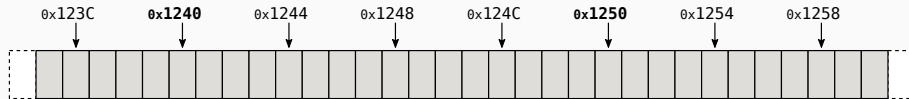
Ejemplo:



MOVD [0x123C], xmm0 ✓ [0x123C] ← xmm0(31:0)

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

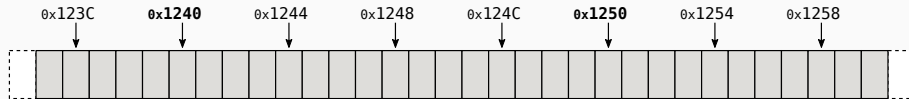
Ejemplo:



MOVD [0x123C], xmm0 ✓ [0x123C] ← xmm0(31:0)
MOVQ xmm0, [0x1245] ✓ xmm0(63:0) ← [0x1245]

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

Ejemplo:



MOVD [0x123C], xmm0	✓	[0x123C] ← xmm0(31:0)
MOVQ xmm0, [0x1245]	✓	xmm0(63:0) ← [0x1245]
MOVDQA xmm0, [0x1245]	×	Error dirección no alineada.

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

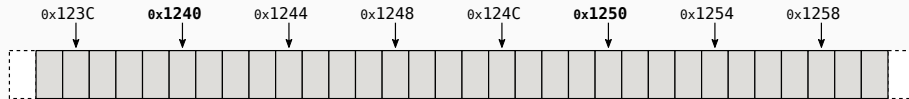
Ejemplo:



MOVD [0x123C], xmm0	✓	[0x123C] ← xmm0(31:0)
MOVQ xmm0, [0x1245]	✓	xmm0(63:0) ← [0x1245]
MOVDQA xmm0, [0x1245]	✗	Error dirección no alineada.
MOVDQA [0x1250], xmm0	✓	[0x1250] ← xmm0(127:0)

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

Ejemplo:



MOVD [0x123C], xmm0	✓	[0x123C] ← xmm0(31:0)
MOVQ xmm0, [0x1245]	✓	xmm0(63:0) ← [0x1245]
MOVDQA xmm0, [0x1245]	✗	Error dirección no alineada.
MOVDQA [0x1250], xmm0	✓	[0x1250] ← xmm0(127:0)
MOVSS xmm0, [0x1248]	✓	xmm0(31:0) ← [0x1248] ; sobre punto flotante

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

Ejemplo:



MOVD [0x123C], xmm0	✓	[0x123C] ← xmm0(31:0)
MOVQ xmm0, [0x1245]	✓	xmm0(63:0) ← [0x1245]
MOVDQA xmm0, [0x1245]	✗	Error dirección no alineada.
MOVDQA [0x1250], xmm0	✓	[0x1250] ← xmm0(127:0)
MOVSS xmm0, [0x1248]	✓	xmm0(31:0) ← [0x1248] ; sobre punto flotante
MOVUPS [0x1258], xmm0	✓	[0x1258] ← xmm0(127:0) ; sobre punto flotante

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

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

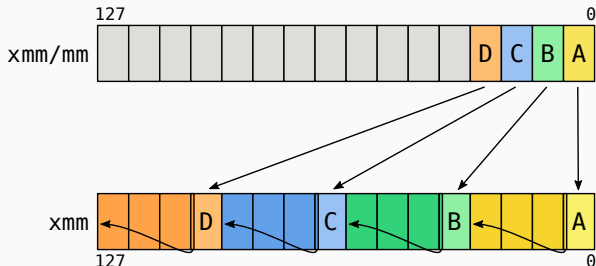
PMOVSXBW	PMOVZXBW	packed sign/zero extension byte to word
PMOVSXBQ	PMOVZXBQ	packed sign/zero extension byte to dword
PMOVSXWD	PMOVZXWD	packed sign/zero extension word to dword
PMOVSXWQ	PMOVZXWQ	packed sign/zero extension word to qword
PMOVXBDQ	PMOVZBDQ	packed sign/zero extension dword to qword

Ejemplos:

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

Ejemplos:

PMOVSXBD xmm0, xmm0

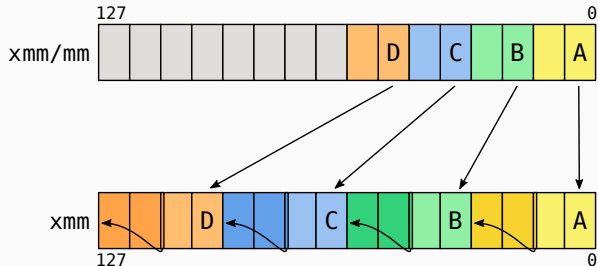


PMOVSXBW	PMOVZXBW	packed sign/zero extension byte to word
PMOVSXBD	PMOVZXBD	packed sign/zero extension byte to dword
PMOVSXBQ	PMOVZXBQ	packed sign/zero extension byte to qword
PMOVSXWD	PMOVZXWD	packed sign/zero extension word to dword
PMOVSXWQ	PMOVZXWQ	packed sign/zero extension word to qword
PMOVXSDQ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

Ejemplos:

PMOVSXBD xmm0, xmm0 ✓

PMOVZXWD xmm0, [data] ✓



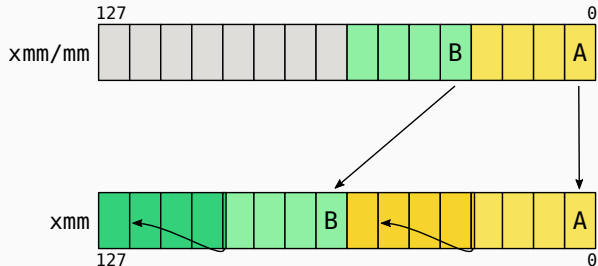
PMOVSXBW	PMOVZXBW	packed sign/zero extension byte to word
PMOVSXBD	PMOVZXBD	packed sign/zero extension byte to dword
PMOVSXBQ	PMOVZXBQ	packed sign/zero extension byte to qword
PMOVSXWD	PMOVZXWD	packed sign/zero extension word to dword
PMOVSXWQ	PMOVZXWQ	packed sign/zero extension word to qword
PMOVSXDQ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

Ejemplos:

PMOVSXBD xmm0, xmm0 ✓

PMOVZXWD xmm0, [data] ✓

PMOVZXDQ xmm0, xmm1 ✓



PMOVSX ^B W	PMOVZX ^B W	packed sign/zero extension byte to word
PMOVSX ^B D	PMOVZX ^B D	packed sign/zero extension byte to dword
PMOVSX ^B Q	PMOVZX ^B Q	packed sign/zero extension byte to qword
PMOVSX ^W D	PMOVZX ^W D	packed sign/zero extension word to dword
PMOVSX ^W Q	PMOVZX ^W Q	packed sign/zero extension word to qword
PMOVSX ^D Q	PMOVZX ^D Q	packed sign/zero extension dword to qword

Ejemplos:

PMOVSXBD xmm0, xmm0	✓
PMOVZXWD xmm0, [data]	✓
PMOVZXDQ xmm0, xmm1	✓
PMOVZXQD xmm0, xmm0	✗ Instrucción invalida.

PMOVSX ^B W	PMOVZX ^B W	packed sign/zero extension byte to word
PMOVSX ^B D	PMOVZX ^B D	packed sign/zero extension byte to dword
PMOVSX ^B Q	PMOVZX ^B Q	packed sign/zero extension byte to qword
PMOVSX ^W D	PMOVZX ^W D	packed sign/zero extension word to dword
PMOVSX ^W Q	PMOVZX ^W Q	packed sign/zero extension word to qword
PMOVSX ^D Q	PMOVZX ^D Q	packed sign/zero extension dword to qword

Ejemplos:

PMOVSXBD xmm0, xmm0	✓	
PMOVZXWD xmm0, [data]	✓	
PMOVZXDQ xmm0, xmm1	✓	
PMOVZXQD xmm0, xmm0	×	Instrucción invalida.
PMOVSXBD [data], xmm0	×	Modo de direccionamiento invalido.

PADDB	PADDW	PADDQ	PADDQ	Add Integer
PSUBB	PSUBW	PSUBD	PSUBQ	Sub Integer
PMULHW	PMULHW			Mul Integer Word
PMULHD	PMULHD			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

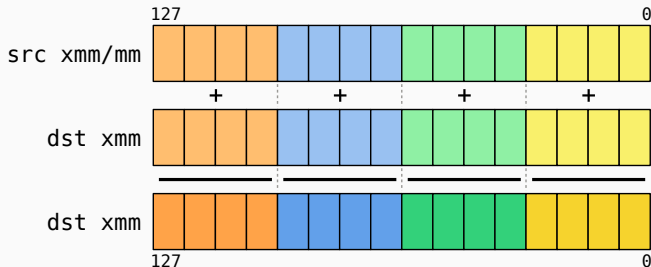
PADDB	PADDW	PADDQ	PADDQ	Add Integer
PSUBB	PSUBW	PSUBD	PSUBQ	Sub Integer
PMULHW	PMULW			Mul Integer Word
PMULHD	PMULD			Mul Integer Dword
PMINSB	PMASB	PMINUB	PMASB	Max and Min Integer
PMINSW	PMASW	PMINUW	PMASW	Max and Min Integer
PMINSD	PMASD	PMINUD	PMASD	Max and Min Integer

Ejemplos:

PADDB	PADDW	PADDQ	PADDQ	Add Integer
PSUBB	PSUBW	PSUBD	PSUBQ	Sub Integer
PMULHW	PMULLW			Mul Integer Word
PMULHD	PMULDQ			Mul Integer Dword
PMINSB	PMASB	PMINUB	PMASUB	Max and Min Integer
PMINSW	PMASW	PMINUW	PMASUW	Max and Min Integer
PMINSD	PMASD	PMINUD	PMASUD	Max and Min Integer

Ejemplos:

PADDQ xmm0, xmm1

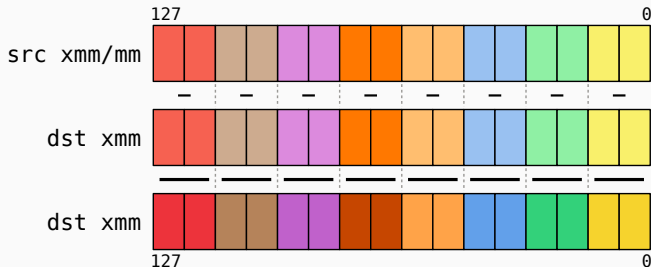


PADDB	PADDW	PADDQ	PADDQ	Add Integer
PSUBB	PSUBW	PSUBD	PSUBQ	Sub Integer
PMULHW	PMULLW			Mul Integer Word
PMULHD	PMULDQ			Mul Integer Dword
PMINSB	PMASB	PMINUB	PMASUB	Max and Min Integer
PMINSW	PMASW	PMINUW	PMASUW	Max and Min Integer
PMINSD	PMASD	PMINUD	PMASUD	Max and Min Integer

Ejemplos:

PADDQ xmm0, xmm1 ✓

PSUBW xmm0, [data] ✓



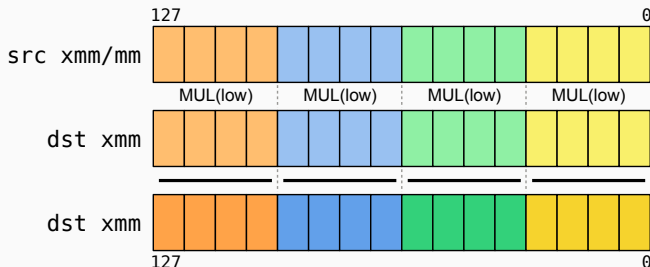
PADDB	PADDW	PADDD	PADDQ	Add Integer
PSUBB	PSUBW	PSUBD	PSUBQ	Sub Integer
PMULHW	PMULLW			Mul Integer Word
PMULHD	PMULLD			Mul Integer Dword
PMINSB	PMASB	PMINUB	PMASUB	Max and Min Integer
PMINSW	PMASW	PMINUW	PMASUW	Max and Min Integer
PMINSD	PMASD	PMINUD	PMASUD	Max and Min Integer

Ejemplos:

PADDD xmm0, xmm1 ✓

PSUBW xmm0, [data] ✓

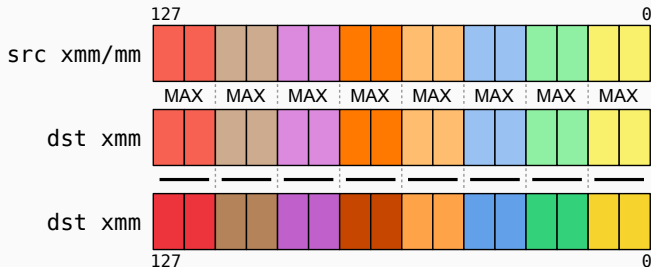
PMULLD xmm0, xmm1 ✓



PADDB	PADDW	PADDQ	PADDQ	Add Integer
PSUBB	PSUBW	PSUBD	PSUBQ	Sub Integer
PMULHW	PMULLW			Mul Integer Word
PMULHD	PMULLD			Mul Integer Dword
PMINSB	PMASB	PMINUB	PMASUB	Max and Min Integer
PMINSW	PMASW	PMINUW	PMASUW	Max and Min Integer
PMINSD	PMASD	PMINUD	PMASUD	Max and Min Integer

Ejemplos:

PADDQ xmm0, xmm1 ✓
 PSUBW xmm0, [data] ✓
 PMULLD xmm0, xmm1 ✓
 PMASW xmm0, [data] ✓



PADDB	PADDW	PADDQ	PADDQ	Add Integer
PSUBB	PSUBW	PSUBD	PSUBQ	Sub Integer
PMULHW	PMULLW			Mul Integer Word
PMULHD	PMULLD			Mul Integer Dword
PMINSB	PMASB	PMINUB	PMASUB	Max and Min Integer
PMINSW	PMASW	PMINUW	PMASUW	Max and Min Integer
PMINSD	PMASD	PMINUD	PMASUD	Max and Min Integer

Ejemplos:

PADDQ xmm0, xmm1	✓	
PSUBW xmm0, [data]	✓	
PMULLD xmm0, xmm1	✓	
PMASW xmm0, [data]	✓	
PMINSB [data], xmm0	×	Modo de direccionamiento invalido.

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

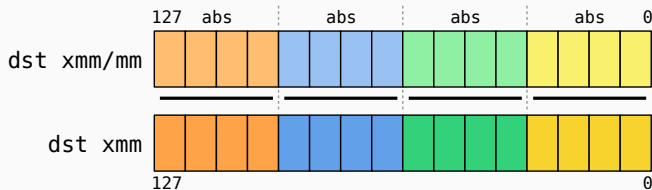
PABSB	Absolute for 8 bit Integers
PABSW	Absolute for 16 bit Integers
PABSD	Absolute for 32 bit Integers

Ejemplos:

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

Ejemplos:

PABSD xmm0, xmm0

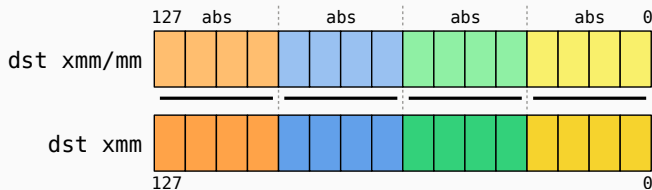


PABSB	Absolute for 8 bit Integers
PABSW	Absolute for 16 bit Integers
PABSD	Absolute for 32 bit Integers

Ejemplos:

PABSD xmm0, xmm0 ✓

PABSD xmm0, [data] ✓



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

Ejemplos:

PABSD xmm0, xmm0



PABSD xmm0, [data]



PABSD [data], xmm0



Modo de direccionamiento invalido.

ADD ^{PS}	ADD ^{SS}	ADD ^{PD}	ADD ^{SD}	Addition of FP values
SUB ^{PS}	SUB ^{SS}	SUB ^{PD}	SUB ^{SD}	Subtraction of FP values
MUL ^{PS}	MUL ^{SS}	MUL ^{PD}	MUL ^{SD}	Multiply of FP values
DIV ^{PS}	DIV ^{SS}	DIV ^{PD}	DIV ^{SD}	Division of FP values
MAX ^{PS}	MAX ^{SS}	MIN ^{PS}	MIN ^{SS}	Max and Min of Single FP values
MAX ^{PD}	MAX ^{SD}	MIN ^{PD}	MIN ^{SD}	Max and Min of Double FP values

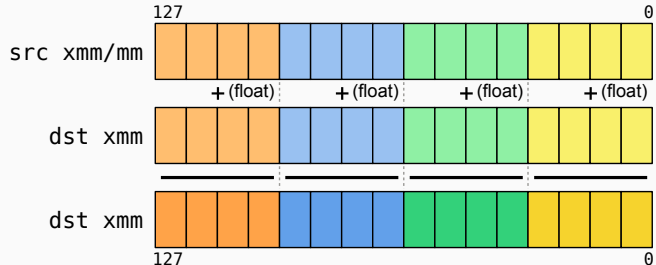
ADD ^{PS}	ADD ^{SS}	ADD ^{PD}	ADD ^{SD}	Addition of FP values
SUB ^{PS}	SUB ^{SS}	SUB ^{PD}	SUB ^{SD}	Subtraction of FP values
MUL ^{PS}	MUL ^{SS}	MUL ^{PD}	MUL ^{SD}	Multiply of FP values
DIV ^{PS}	DIV ^{SS}	DIV ^{PD}	DIV ^{SD}	Division of FP values
MAX ^{PS}	MAX ^{SS}	MIN ^{PS}	MIN ^{SS}	Max and Min of Single FP values
MAX ^{PD}	MAX ^{SD}	MIN ^{PD}	MIN ^{SD}	Max and Min of Double FP values

Ejemplos:

ADD ^{PS}	ADD ^{SS}	ADD ^{PD}	ADD ^{SD}	Addition of FP values
SUB ^{PS}	SUB ^{SS}	SUB ^{PD}	SUB ^{SD}	Subtraction of FP values
MUL ^{PS}	MUL ^{SS}	MUL ^{PD}	MUL ^{SD}	Multiply of FP values
DIV ^{PS}	DIV ^{SS}	DIV ^{PD}	DIV ^{SD}	Division of FP values
MAX ^{PS}	MAX ^{SS}	MIN ^{PS}	MIN ^{SS}	Max and Min of Single FP values
MAX ^{PD}	MAX ^{SD}	MIN ^{PD}	MIN ^{SD}	Max and Min of Double FP values

Ejemplos:

ADDPS xmm0, [data] ✓

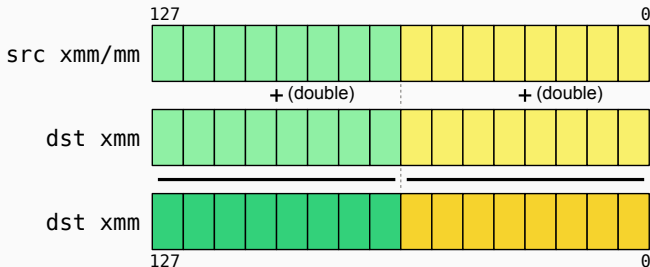


ADDPS	ADDSS	ADDPD	ADDSD	Addition of FP values
SUBPS	SUBSS	SUBPD	SUBSD	Subtraction of FP values
MULPS	MULSS	MULPD	MULSD	Multiply of FP values
DIVPS	DIVSS	DIVPD	DIVSD	Division of FP values
MAXPS	MAXSS	MINPS	MINSS	Max and Min of Single FP values
MAXPD	MAXSD	MINPD	MINSD	Max and Min of Double FP values

Ejemplos:

ADDPS xmm0, [data] ✓

ADDPD xmm0, [data] ✓



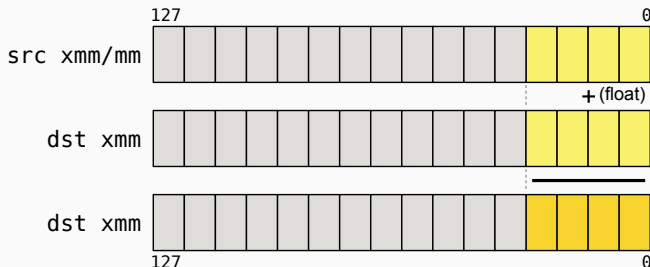
ADD ^{PS}	ADD ^{SS}	ADD ^{PD}	ADD ^{SD}	Addition of FP values
SUB ^{PS}	SUB ^{SS}	SUB ^{PD}	SUB ^{SD}	Subtraction of FP values
MUL ^{PS}	MUL ^{SS}	MUL ^{PD}	MUL ^{SD}	Multiply of FP values
DIV ^{PS}	DIV ^{SS}	DIV ^{PD}	DIV ^{SD}	Division of FP values
MAX ^{PS}	MAX ^{SS}	MIN ^{PS}	MIN ^{SS}	Max and Min of Single FP values
MAX ^{PD}	MAX ^{SD}	MIN ^{PD}	MIN ^{SD}	Max and Min of Double FP values

Ejemplos:

ADDPS xmm0, [data] ✓

ADDPD xmm0, [data] ✓

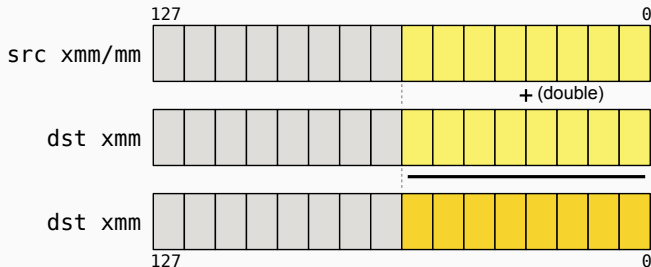
ADDSS xmm0, [data] ✓



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	Division 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

Ejemplos:

ADDPS xmm0, [data] ✓
 ADDPD xmm0, [data] ✓
 ADDSS xmm0, [data] ✓
 ADDSD xmm0, [data] ✓



ADD ^{PS}	ADD ^{SS}	ADD ^{PD}	ADD ^{SD}	Addition of FP values
SUB ^{PS}	SUB ^{SS}	SUB ^{PD}	SUB ^{SD}	Subtraction of FP values
MUL ^{PS}	MUL ^{SS}	MUL ^{PD}	MUL ^{SD}	Multiply of FP values
DIV ^{PS}	DIV ^{SS}	DIV ^{PD}	DIV ^{SD}	Division of FP values
MAX ^{PS}	MAX ^{SS}	MIN ^{PS}	MIN ^{SS}	Max and Min of Single FP values
MAX ^{PD}	MAX ^{SD}	MIN ^{PD}	MIN ^{SD}	Max and Min of Double FP values

Ejemplos:

ADDPS xmm0, [data] ✓

ADDPD xmm0, [data] ✓

ADDSS xmm0, [data] ✓

ADDSD xmm0, [data] ✓

MINSD [data], xmm0 ✗ Modo de direccionamiento invalido.

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

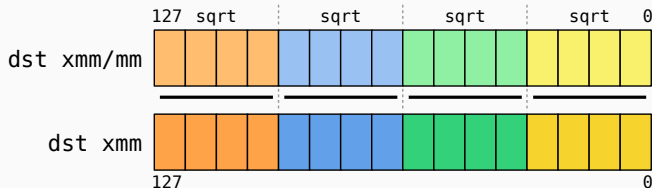
SQRTSS	SQRTPS	Square root of Scalar/Packed Single FP values
SQRTSD	SQRTPD	Square root of Scalar/Packed Double FP values

Ejemplos:

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

Ejemplos:

SQRTPS xmm0, [data]

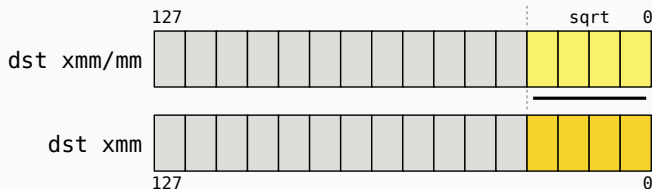


SQRTSS	SQRTPS	Square root of Scalar/Packed Single FP values
SQRTSD	SQRTPD	Square root of Scalar/Packed Double FP values

Ejemplos:

SQRTPS xmm0, [data] ✓

SQRTSS xmm0, [data] ✓



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

Ejemplos:

SQRTPS xmm0, [data] ✓

SQRTSS xmm0, [data] ✓

SQRTPD [data], xmm0 ✗ Modo de direccionamiento invalido.

PADD ^{SB}	PADD ^{SW}	Add Int saturation
PADD ^{USB}	PADD ^{USW}	Add Int unsigned saturation
PSUB ^{SB}	PSUB ^{SW}	Sub Int saturation
PSUB ^{USB}	PSUB ^{USW}	Sub Int unsigned saturation

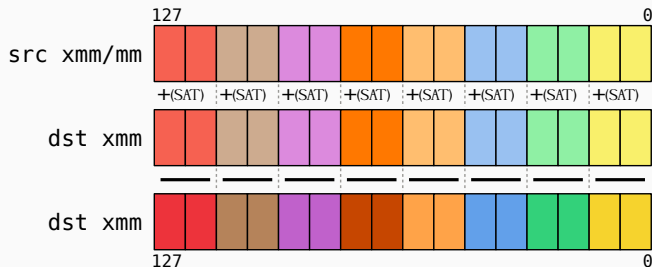
PADD ^{SB}	PADD ^{SW}	Add Int saturation
PADD ^{USB}	PADD ^{USW}	Add Int unsigned saturation
PSUB ^{SB}	PSUB ^{SW}	Sub Int saturation
PSUB ^{USB}	PSUB ^{USW}	Sub Int unsigned saturation

Ejemplos:

PADD ^{SB}	PADD ^{SW}	Add Int saturation
PADD ^{USB}	PADD ^{USW}	Add Int unsigned saturation
PSUB ^{SB}	PSUB ^{SW}	Sub Int saturation
PSUB ^{USB}	PSUB ^{USW}	Sub Int unsigned saturation

Ejemplos:

PADD^{SW} xmm0, xmm0



PADD ^{SB}	PADD ^{SW}	Add Int saturation
PADD ^{USB}	PADD ^{USW}	Add Int unsigned saturation
PSUB ^{SB}	PSUB ^{SW}	Sub Int saturation
PSUB ^{USB}	PSUB ^{USW}	Sub Int unsigned saturation

Ejemplos:

PADD^{SW} xmm0, xmm0



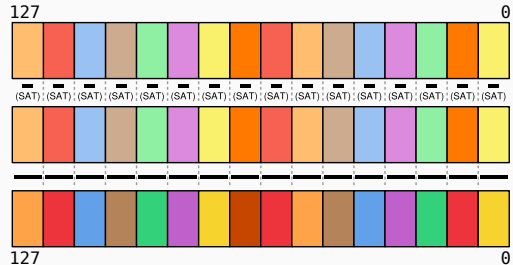
PSUB^{USB} xmm0, [data]



src xmm/mm

dst xmm

dst xmm



PADD ^{SB}	PADD ^{SW}	Add Int saturation
PADD ^{USB}	PADD ^{USW}	Add Int unsigned saturation
PSUB ^{SB}	PSUB ^{SW}	Sub Int saturation
PSUB ^{USB}	PSUB ^{USW}	Sub Int unsigned saturation

Ejemplos:

PADD^{SW} xmm0, xmm0



PSUB^{USB} xmm0, [data]



PSUB^{SW} [data], xmm0



Modo de direccionamiento invalido.

Incrementar Brillo

Dado una imagen 32x32 pixeles de un byte en escala de grises. Incrementar el brillo de la misma en 10 unidades.

```
void incrementarBrillo10(uint8_t *imagen)
```

```
section .rodata
diez: times 16 db 10

section .text

incrementarBrillo10: ; rdi = imagen
    push rbp
    mov rbp, rsp
```

```
section .rodata
diez: times 16 db 10

section .text

incrementarBrillo10: ; rdi = imagen
    push rbp
    mov rbp, rsp
    mov rcx, (32*32 >> 4)
```

```
section .rodata
diez: times 16 db 10

section .text

incrementarBrillo10: ; rdi = imagen
    push rbp
    mov rbp, rsp
    mov rcx, (32*32 >> 4)
    movdqu xmm8, [diez] ; xmm0 = | 10 | ... | 10 |
```

```
section .rodata
diez: times 16 db 10

section .text

incrementarBrillo10: ; rdi = imagen
    push rbp
    mov rbp, rsp
    mov rcx, (32*32 >> 4)
    movdqu xmm8, [diez]      ; xmm8 = | 10 | ... | 10 |
.ciclo:
    movdqu xmm0, [rdi]      ; xmm0 = | d15 | ... | d0 |
```

```
section .rodata
diez: times 16 db 10

section .text

incrementarBrillo10: ; rdi = imagen
    push rbp
    mov rbp, rsp
    mov rcx, (32*32 >> 4)
    movdqu xmm8, [diez] ; xmm8 = | 10 | ... | 10 |
.ciclo:
    movdqu xmm0, [rdi] ; xmm0 = | d15 | ... | d0 |
    paddusb xmm0, xmm8 ; xmm0 = |d15+10| ... |d0+10|
```

```
section .rodata
diez: times 16 db 10

section .text

incrementarBrillo10: ; rdi = imagen
    push rbp
    mov rbp, rsp
    mov rcx, (32*32 >> 4)
    movdqu xmm8, [diez]      ; xmm8 = | 10 | ... | 10 |
    .ciclo:
        movdqu xmm0, [rdi]   ; xmm0 = | d15 | ... | d0 |
        paddusb xmm0, xmm8   ; xmm0 = |d15+10| ... |d0+10|
        movdqu [rdi], xmm0
```

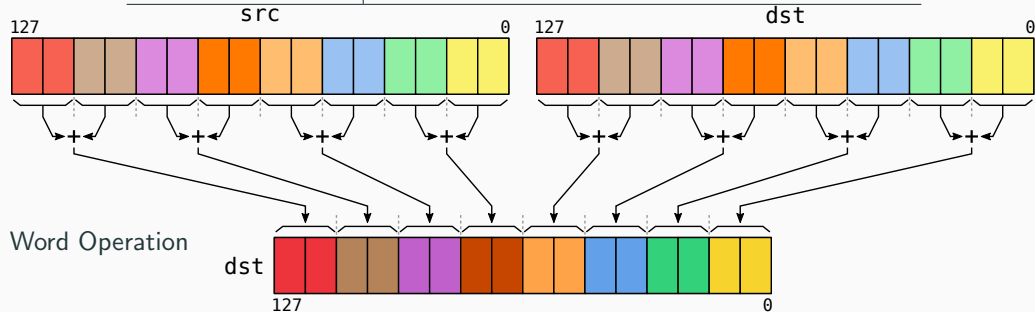
```
section .rodata
diez: times 16 db 10

section .text

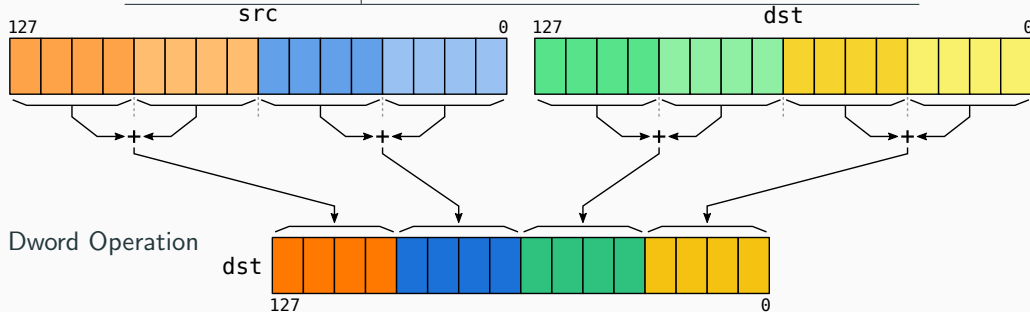
incrementarBrillo10: ; rdi = imagen
    push rbp
    mov rbp, rsp
    mov rcx, (32*32 >> 4)
    movdqu xmm8, [diez]      ; xmm8 = | 10 | ... | 10 |
    .ciclo:
        movdqu xmm0, [rdi]   ; xmm0 = | d15 | ... | d0 |
        paddusb xmm0, xmm8   ; xmm0 = |d15+10| ... |d0+10|
        movdqu [rdi], xmm0
        add rdi, 16
    loop .ciclo
    pop rbp
    ret
```


PHADDW	PHADD	Horizontal addition of 16bit/32bit integers
PHADDSW		Horizontal saturated addition of 16bit integers
PHSUBW	PHSUB	Horizontal subtraction of 16bit/32bit integers
PHSUBSW		Horizontal saturated subtraction of 16bit words
HADDPS	HADDP	Packed Single/Double FP Horizontal Add
HSUBPS	HSUBP	Packed Single/Double FP Horizontal Subtract

PHADDW	PHADD	Horizontal addition of 16bit/32bit integers
PHADDSW		Horizontal saturated addition of 16bit integers
PHSUBW	PHSUBD	Horizontal subtraction of 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



PHADDW	PHADD	Horizontal addition of 16bit/32bit integers
PHADDSW		Horizontal saturated addition of 16bit integers
PHSUBW	PHSUBD	Horizontal subtraction of 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



PAND	PANDN	POR	PXOR	Operaciones lógicas para enteros.
AND ^{PS}	ANDN ^{PS}	OR ^{PS}	XOR ^{PS}	Operaciones lógicas para <i>float</i> .
AND ^{PD}	ANDN ^{PD}	OR ^{PD}	XOR ^{PD}	Operaciones lógicas para <i>double</i> .

- Actúan lógicamente sobre todo el registro, sin importar el tamaño del operando.
- La distinción entre ^{PS} y ^{PD} se debe a meta información para el procesador.

PAND	PANDN	POR	PXOR	Operaciones lógicas para enteros.
AND $\textcolor{teal}{PS}$	ANDN $\textcolor{teal}{PS}$	OR $\textcolor{teal}{PS}$	XOR $\textcolor{teal}{PS}$	Operaciones lógicas para <i>float</i> .
AND $\textcolor{teal}{PD}$	ANDN $\textcolor{teal}{PD}$	OR $\textcolor{teal}{PD}$	XOR $\textcolor{teal}{PD}$	Operaciones lógicas para <i>double</i> .

- Actúan lógicamente sobre todo el registro, sin importar el tamaño del operando.
- La distinción entre $\textcolor{teal}{PS}$ y $\textcolor{teal}{PD}$ se debe a meta información para el procesador.

PSLL $\textcolor{teal}{W}$	PSLL $\textcolor{teal}{D}$	PSLL $\textcolor{teal}{Q}$	PSLLDQ*
PSRL $\textcolor{teal}{W}$	PSRL $\textcolor{teal}{D}$	PSRL $\textcolor{teal}{Q}$	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 indicadas, el parámetro es la cantidad de bytes del desplazamiento.

11111111	00000000	11111111	00000000
----------	----------	----------	----------

1. Calculo de la mascara

11111111 00000000 11111111 00000000

1. Calculo de la mascara

Datos



11111111 00000000 11111111 00000000

1. Calculo de la mascara

NOT (11111111 00000000 11111111 00000000)

11111111 00000000 11111111 00000000

AND

AND

Datos



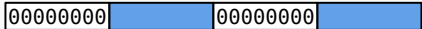
11111111 00000000 11111111 00000000

1. Calculo de la mascara

NOT (11111111 00000000 11111111 00000000)

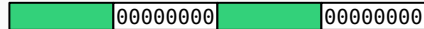
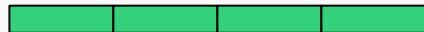
AND

Datos



11111111 00000000 11111111 00000000

AND



2. Aplicación de la mascara

11111111 00000000 11111111 00000000

1. Calculo de la mascara

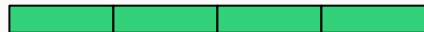
NOT (11111111 00000000 11111111 00000000)

11111111 00000000 11111111 00000000

AND

AND

Datos



00000000 [blue block] 00000000 [blue block]

[green block] 00000000 [green block] 00000000

2. Aplicación de la mascara

00000000 [blue block] 00000000 [blue block]

OR

[green block] 00000000 [green block] 00000000

11111111 00000000 11111111 00000000

1. Calculo de la mascara

NOT (11111111 00000000 11111111 00000000)

11111111 00000000 11111111 00000000

AND

AND

Datos



00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

2. Aplicación de la mascara

00000000 00000000 00000000 00000000

OR

00000000 00000000 00000000 00000000

3. Combinación de resultados

00000000 00000000 00000000 00000000

PCMP EQ B	PCMP EQ W	PCMP EQ D	PCMP EQ Q	Compare Packed Data for Equal
PCMP GT B	PCMP GT W	PCMP GT D	PCMP GT Q	Compare Packed Signed Int for Greater Than

PCMP EQ B	PCMP EQ W	PCMP EQ D	PCMP EQ Q	Compare Packed Data for Equal
PCMP GT B	PCMP GT W	PCMP GT D	PCMP GT Q	Compare Packed Signed Int for Greater Than

Ejemplos:

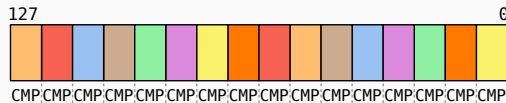
PCMPEQB	PCMPEQW	PCMPEQD	PCMPEQQ	Compare Packed Data for Equal
PCMPGTB	PCMPGTW	PCMPGTD	PCMPGTQ	Compare Packed Signed Int for Greater Than

Ejemplos:

PCMPEQB xmm0, [data]



src xmm/mm



dst xmm



dst xmm



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

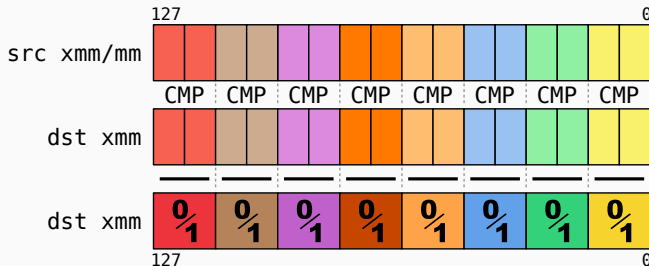
Ejemplos:

PCMPEQB xmm0, [data]

✓

PCMPEQW xmm0, [data]

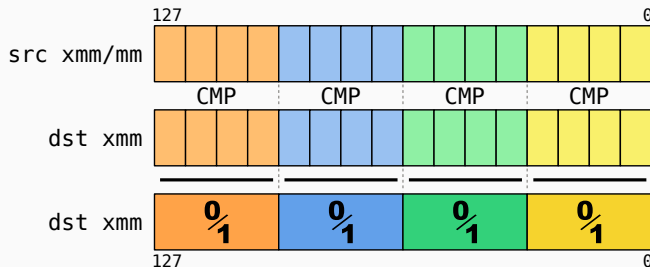
✓



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

Ejemplos:

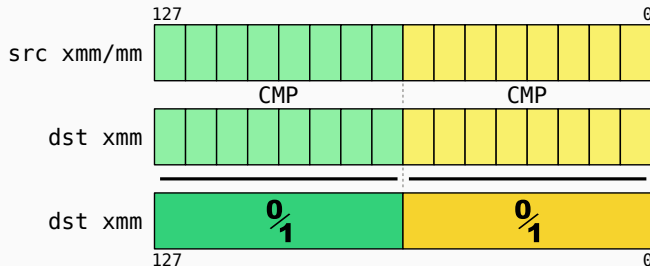
PCMPEQB xmm0, [data] ✓
 PCMPEQW xmm0, [data] ✓
 PCMPEQD xmm0, [data] ✓



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

Ejemplos:

PCMPEQB xmm0, [data] ✓
 PCMPEQW xmm0, [data] ✓
 PCMPEQD xmm0, [data] ✓
 PCMPEQQ xmm0, [data] ✓



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

Ejemplos:

PCMPEQB xmm0, [data]	✓	
PCMPEQW xmm0, [data]	✓	
PCMPEQD xmm0, [data]	✓	
PCMPEQQ xmm0, [data]	✓	
PCMPGTQ [data], xmm0	✗	Modo de direccionamiento invalido.

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	$A \leq B$
3	No Orden	UNORD	$A, B = \text{unordered}$
4	Distinto	NEQ	$A \neq B$
5	No Menor	NLT	$\text{not}(A < B)$
6	No Menor o Igual	NLE	$\text{not}(A \leq B)$
7	Orden	ORD	$A, B = \text{Ordered}$

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

Ejemplos:

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

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

Ejemplos:

CMPEQPD xmm0, [data] ✓

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

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

Ejemplos:

CMPEQPD xmm0, [data] ✓
 CMPLEPD xmm0, [data] ✓

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

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

Ejemplos:

CMPEQPD xmm0, [data] ✓

CMPLEPD xmm0, [data] ✓

CMPORDPD xmm0, [data] ✓ ; (Nan)

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

Suma pares

Dado un vector de 128 enteros con signo de 16 bits. Sumar todos los valores pares y retornar el resultado de la suma en 32 bits.

```
int32_t sumarPares(int16_t *v)
```



```
sumarpares: ; rdi = int16_t *v  
    push rbp  
    mov rbp, rsp
```

```
sumarpares: ; rdi = int16_t *v
    push rbp
    mov rbp, rsp
    mov rcx, (128 >> 2) ; rcx = 128 / 4
    pxor xmm8, xmm8 ; xmm8 = | 0 | 0 | 0 | 0 |
```

```
sumarpares: ; rdi = int16_t *v
    push rbp
    mov rbp, rsp
    mov rcx, (128 >> 2) ; rcx = 128 / 4
    pxor xmm8, xmm8 ; xmm8 = | 0 | 0 | 0 | 0 |
.ciclo:
    pmovsxd xmm0, [rdi] ; (ejemplo) xmm0 = | 00001233 | 00007314 | 00003011 | FFFF9311 |
```

```
sumarpares: ; rdi = int16_t *v
    push rbp
    mov rbp, rsp
    mov rcx, (128 >> 2) ; rcx = 128 / 4
    pxor xmm8, xmm8 ; xmm8 = | 0 | 0 | 0 | 0 |
.ciclo:
    pmovsxd xmm0, [rdi] ; (ejemplo) xmm0 = | 00001233 | 00007314 | 00003011 | FFFF9311 |
    pabsd xmm1, xmm0 ; (ejemplo) xmm1 = | 00001233 | 00007314 | 00003011 | 00006CEE |
```

```
sumarpares: ; rdi = int16_t *v
    push rbp
    mov rbp, rsp
    mov rcx, (128 >> 2) ; rcx = 128 / 4
    pxor xmm8, xmm8 ; xmm8 = | 0 | 0 | 0 | 0 |
.ciclo:
    pmovsxd xmm0, [rdi] ; (ejemplo) xmm0 = | 00001233 | 00007314 | 00003011 | FFFF9311 |
    pabsd xmm1, xmm0 ; (ejemplo) xmm1 = | 00001233 | 00007314 | 00003011 | 00006CEE |
    pslld xmm1, 31 ; (ejemplo) xmm1 = | 80000000 | 00000000 | 80000000 | 00000000 |
```

```
sumarpares: ; rdi = int16_t *v
    push rbp
    mov rbp, rsp
    mov rcx, (128 >> 2) ; rcx = 128 / 4
    pxor xmm8, xmm8 ; xmm8 = | 0 | 0 | 0 | 0 |
.ciclo:
    pmovsxd xmm0, [rdi] ; (ejemplo) xmm0 = | 00001233 | 00007314 | 00003011 | FFFF9311 |
    pabsd xmm1, xmm0 ; (ejemplo) xmm1 = | 00001233 | 00007314 | 00003011 | 00006CEE |
    pslld xmm1, 31 ; (ejemplo) xmm1 = | 80000000 | 00000000 | 80000000 | 00000000 |
    psrad xmm1, 31 ; (ejemplo) xmm1 = | FFFFFFFF | 00000000 | FFFFFFFF | 00000000 |
```

```
sumarpares: ; rdi = int16_t *v
    push rbp
    mov rbp, rsp
    mov rcx, (128 >> 2) ; rcx = 128 / 4
    pxor xmm8, xmm8 ; xmm8 = | 0 | 0 | 0 | 0 |
.ciclo:
    pmovsxd xmm0, [rdi] ; (ejemplo) xmm0 = | 00001233 | 00007314 | 00003011 | FFFF9311 |
    pabsd xmm1, xmm0 ; (ejemplo) xmm1 = | 00001233 | 00007314 | 00003011 | 00006CEE |
    pslld xmm1, 31 ; (ejemplo) xmm1 = | 80000000 | 00000000 | 80000000 | 00000000 |
    psrad xmm1, 31 ; (ejemplo) xmm1 = | FFFFFFFF | 00000000 | FFFFFFFF | 00000000 |
    pandn xmm1, xmm0 ; (ejemplo) xmm1 = | 00000000 | 00007314 | 00000000 | FFFF9311 |
```

```
sumarpares: ; rdi = int16_t *v
    push rbp
    mov rbp, rsp
    mov rcx, (128 >> 2) ; rcx = 128 / 4
    pxor xmm8, xmm8 ; xmm8 = | 0 | 0 | 0 | 0 |
    .ciclo:
        pmovsxd xmm0, [rdi] ; (ejemplo) xmm0 = | 00001233 | 00007314 | 00003011 | FFFF9311 |
        pabsd xmm1, xmm0 ; (ejemplo) xmm1 = | 00001233 | 00007314 | 00003011 | 00006CEE |
        psllq xmm1, 31 ; (ejemplo) xmm1 = | 80000000 | 00000000 | 80000000 | 00000000 |
        psrad xmm1, 31 ; (ejemplo) xmm1 = | FFFFFFFF | 00000000 | FFFFFFFF | 00000000 |
        pandn xmm1, xmm0 ; (ejemplo) xmm1 = | 00000000 | 00007314 | 00000000 | FFFF9311 |
        paddq xmm8, xmm1 ; xmm8 = | SUM3 | SUM2 | SUM1 | SUM0 |
        add rdi, 8
    loop .ciclo
```

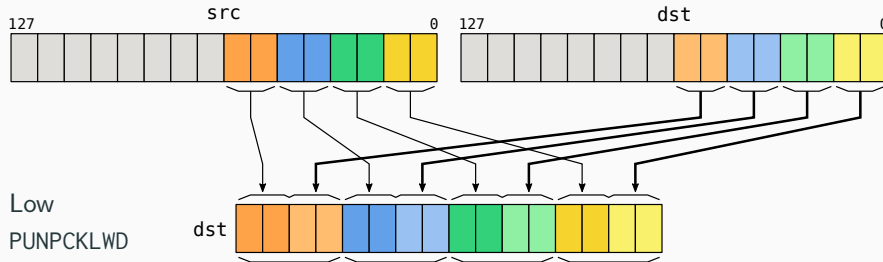


```
sumarpares: ; rdi = int16_t *v
    push rbp
    mov rbp, rsp
    mov rcx, (128 >> 2) ; rcx = 128 / 4
    pxor xmm8, xmm8 ; xmm8 = | 0 | 0 | 0 | 0 |
    .ciclo:
        pmovsxdq xmm0, [rdi] ; (ejemplo) xmm0 = | 00001233 | 00007314 | 00003011 | FFFF9311 |
        pabsd xmm1, xmm0 ; (ejemplo) xmm1 = | 00001233 | 00007314 | 00003011 | 00006CEE |
        psllq xmm1, 31 ; (ejemplo) xmm1 = | 80000000 | 00000000 | 80000000 | 00000000 |
        psrad xmm1, 31 ; (ejemplo) xmm1 = | FFFFFFFF | 00000000 | FFFFFFFF | 00000000 |
        pandn xmm1, xmm0 ; (ejemplo) xmm1 = | 00000000 | 00007314 | 00000000 | FFFF9311 |
        paddq xmm8, xmm1 ; xmm8 = | SUM3 | SUM2 | SUM1 | SUM0 |
        add rdi, 8
    loop .ciclo
    phaddq xmm8, xmm8 ; xmm8 = | ... | ... | SUM3+SUM2 | SUM1+SUM0 |
    phaddq xmm8, xmm8 ; xmm8 = | ... | ... | ... | SUM3+SUM2+SUM1+SUM0 |
```

```
sumarpares: ; rdi = int16_t *v
    push rbp
    mov rbp, rsp
    mov rcx, (128 >> 2) ; rcx = 128 / 4
    pxor xmm8, xmm8 ; xmm8 = | 0 | 0 | 0 | 0 |
    .ciclo:
        pmovsxdq xmm0, [rdi] ; (ejemplo) xmm0 = | 00001233 | 00007314 | 00003011 | FFFF9311 |
        pabsd xmm1, xmm0 ; (ejemplo) xmm1 = | 00001233 | 00007314 | 00003011 | 00006CEE |
        psllq xmm1, 31 ; (ejemplo) xmm1 = | 80000000 | 00000000 | 80000000 | 00000000 |
        psrad xmm1, 31 ; (ejemplo) xmm1 = | FFFFFFFF | 00000000 | FFFFFFFF | 00000000 |
        pandn xmm1, xmm0 ; (ejemplo) xmm1 = | 00000000 | 00007314 | 00000000 | FFFF9311 |
        paddq xmm8, xmm1 ; xmm8 = | SUM3 | SUM2 | SUM1 | SUM0 |
        add rdi, 8
    loop .ciclo
    phaddq xmm8, xmm8 ; xmm8 = | ... | ... | SUM3+SUM2 | SUM1+SUM0 |
    phaddq xmm8, xmm8 ; xmm8 = | ... | ... | SUM3+SUM2+SUM1+SUM0 |
    movd eax, xmm8 ; eax = SUM3+SUM2+SUM1+SUM0
    pop rbp
    ret
```

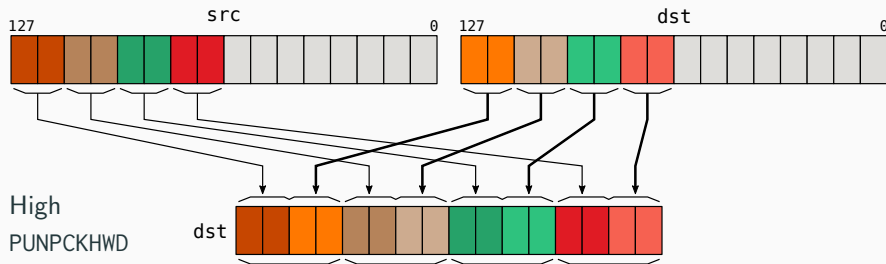
Operaciones de desempaquetado (Unpack)

PUNPCK ^L BW	PUNPCK ^H BW	Unpacks 8 enteros de 8 bits en words
PUNPCK ^L WD	PUNPCK ^H WD	Unpacks 4 enteros de 16 bits en dwords
PUNPCK ^L DQ	PUNPCK ^H DQ	Unpacks 2 enteros de 32 bits en qwords
PUNPCK ^L QDQ	PUNPCK ^H QDQ	Unpacks 1 entero de 64 bits en 128 bits
UNPCK ^L PS	UNPCK ^H PS	Unpacks Single FP
UNPCK ^L PD	UNPCK ^H PD	Unpacks Double FP



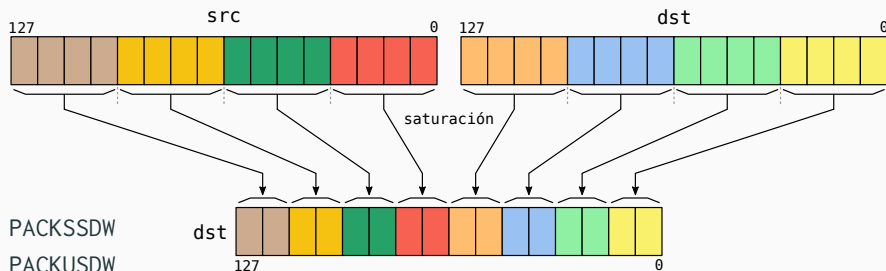
Operaciones de desempaquetado (Unpack)

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



Operaciones de desempaquetado (Unpack)

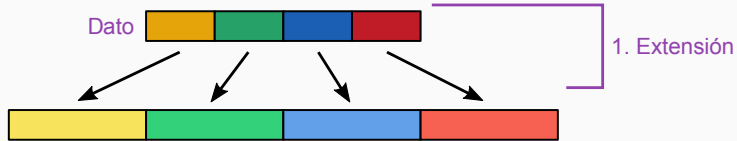
PACKSSDW	Packs 32 bits (signado) a 16 bits (signado) usando saturation
PACKUSDW	Packs 32 bits (signado) a 16 bits (sin signo) usando saturation
PACKSSWB	Packs 16 bits (signado) a 8 bits (signado) usando saturation
PACKUSWB	Packs 16 bits (signado) a 8 bits (sin signo) usando saturation

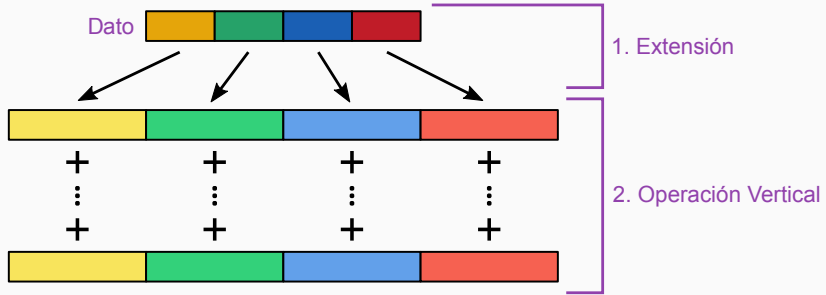


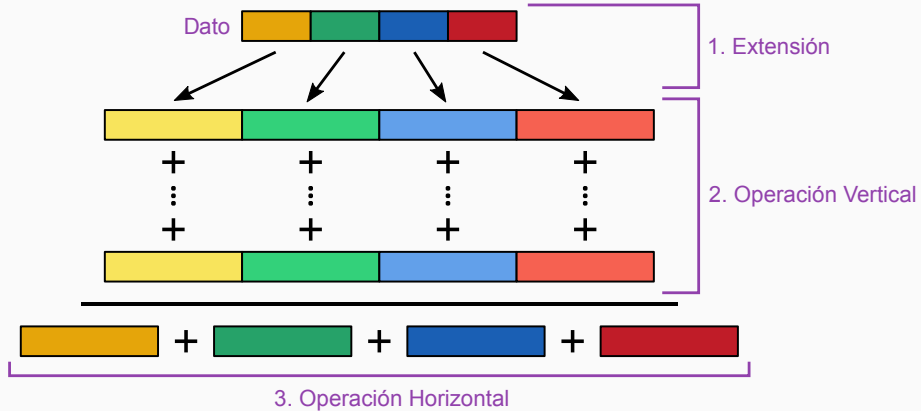
Dato

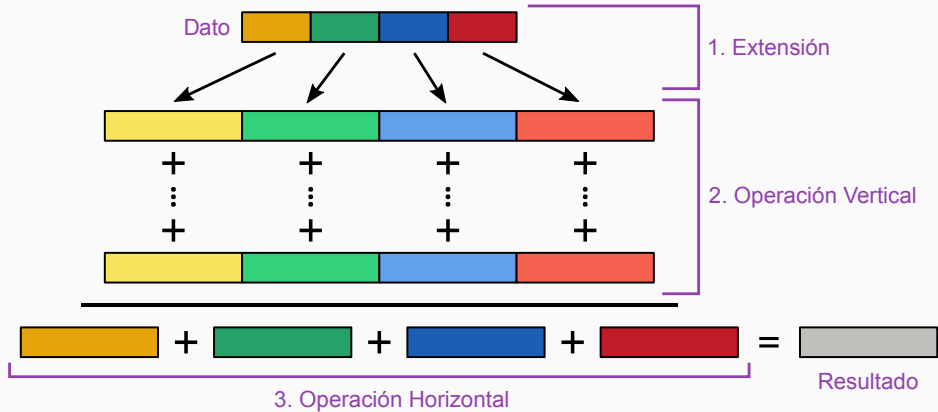


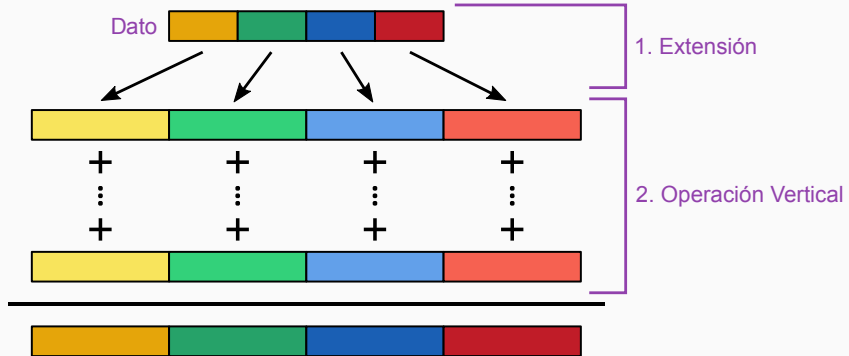
Dato 

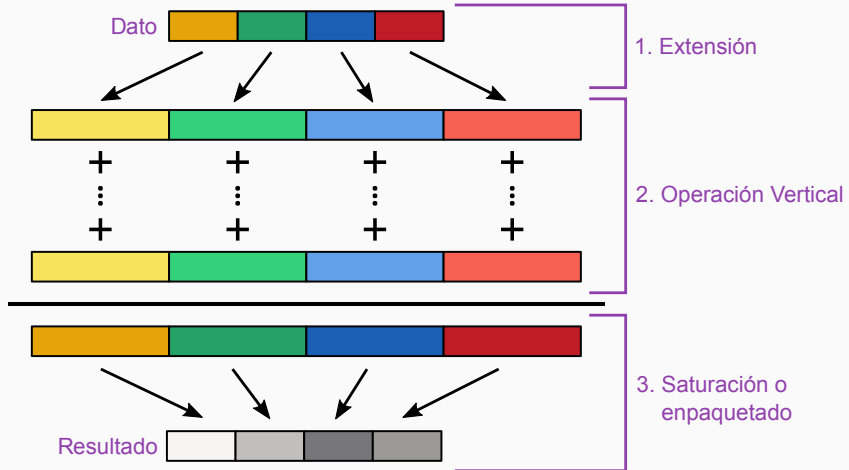












Multiplicar vectores

Dado dos vectores de 128 enteros con signo de 16 bits. Multiplicar cada uno de ellos entre si y almacenar el resultado en un vector de enteros de 32 bits.

```
void mulvec(int16_t *v1, int16_t *v2, int32_t *resultado)
```

```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado  
push rbp  
mov rbp, rsp  
mov rcx, (128 >> 3) ; rcx = 128 / 8
```

```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado
push rbp
mov rbp, rsp
mov rcx, (128 >> 3) ; rcx = 128 / 8
.ciclo:
    movdqa xmm0, [rdi] ; xmm0 = | a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
    movdqa xmm1, [rsi] ; xmm1 = | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
```



```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado
push rbp
mov rbp, rsp
mov rcx, (128 >> 3) ; rcx = 128 / 8
.ciclo:
    movdqa xmm0, [rdi] ; xmm0 = | a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
    movdqa xmm1, [rsi] ; xmm1 = | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
    movdqa xmm2, xmm0 ; xmm2 = | a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
    pmulhw xmm2, xmm1 ; xmm2 = | hi(a7*b7)      ...      hi(a0*b0) |
    pmullw xmm0, xmm1 ; xmm0 = | low(a7*b7)      ...      low(a0*b0) |
```

```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado
push rbp
mov rbp, rsp
mov rcx, (128 >> 3) ; rcx = 128 / 8
.ciclo:
    movdqa xmm0, [rdi] ; xmm0 = | a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
    movdqa xmm1, [rsi] ; xmm1 = | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
    movdqa xmm2, xmm0 ; xmm2 = | a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
    pmulhw xmm2, xmm1 ; xmm2 = | hi(a7*b7)      ...      hi(a0*b0) |
    pmullw xmm0, xmm1 ; xmm0 = | low(a7*b7)      ...      low(a0*b0) |
    movdqa xmm1, xmm0 ; xmm1 = | low(a7*b7)      ...      low(a0*b0) |
    punpcklwd xmm0, xmm2 ; xmm0 = | hi:low(a3*b3)  ...  hi:low(a0*b0) |
    punpckhwd xmm1, xmm2 ; xmm1 = | hi:low(a7*b7)  ...  hi:low(a4*b4) |
```

```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado
push rbp
mov rbp, rsp
mov rcx, (128 >> 3) ; rcx = 128 / 8
.ciclo:
    movdqa xmm0, [rdi] ; xmm0 = | a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
    movdqa xmm1, [rsi] ; xmm1 = | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
    movdqa xmm2, xmm0 ; xmm2 = | a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
    pmulhw xmm2, xmm1 ; xmm2 = | hi(a7*b7) | ... | hi(a0*b0) |
    pmullw xmm0, xmm1 ; xmm0 = | low(a7*b7) | ... | low(a0*b0) |
    movdqa xmm1, xmm0 ; xmm1 = | low(a7*b7) | ... | low(a0*b0) |
    punpcklwd xmm0, xmm2 ; xmm0 = | hi:low(a3*b3) | ... | hi:low(a0*b0) |
    punpckhwd xmm1, xmm2 ; xmm1 = | hi:low(a7*b7) | ... | hi:low(a4*b4) |
    movdqa [rdx], xmm0
    movdqa [rdx+16], xmm1
```

```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado
push rbp
mov rbp, rsp
mov rcx, (128 >> 3) ; rcx = 128 / 8
.ciclo:
    movdqa xmm0, [rdi] ; xmm0 = | a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
    movdqa xmm1, [rsi] ; xmm1 = | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
    movdqa xmm2, xmm0 ; xmm2 = | a7 | a6 | a5 | a4 | a3 | a2 | a1 | a0 |
    pmulhw xmm2, xmm1 ; xmm2 = | hi(a7*b7) | ... | hi(a0*b0) |
    pmullw xmm0, xmm1 ; xmm0 = | low(a7*b7) | ... | low(a0*b0) |
    movdqa xmm1, xmm0 ; xmm1 = | low(a7*b7) | ... | low(a0*b0) |
    punpcklwd xmm0, xmm2 ; xmm0 = | hi:low(a3*b3) | ... | hi:low(a0*b0) |
    punpckhwd xmm1, xmm2 ; xmm1 = | hi:low(a7*b7) | ... | hi:low(a4*b4) |
    movdqa [rdx], xmm0
    movdqa [rdx+16], xmm1
    add rdx, 32
    add rdi, 16
    add rsi, 16
loop .ciclo
pop rbp
ret
```

¿Preguntas?