Práctica de Organización del Computador II

SIMD

Organización del Computador II
DC - UBA

Segundo Cuatrimestre 2022

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

Registros y tipos de datos

- 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).

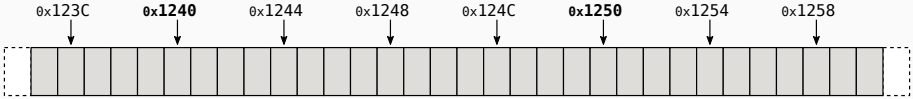
Operaciones Load/Store

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

Operaciones Load/Store

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

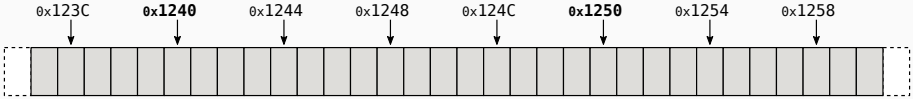
Ejemplo:



Operaciones Load/Store

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:

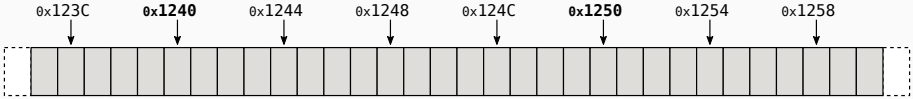


`MOVD [0x123C], xmm0` ✓ `[0x123C]` ← `xmm0(32:0)`

Operaciones Load/Store

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:

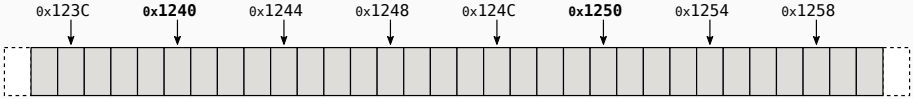



```
MOVD [0x123C], xmm0        ✓   [0x123C] ← xmm0(32:0)
MOVQ xmm0, [0x1245]        ✓   xmm0(64:0) ← [0x1245]
```

Operaciones Load/Store

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:

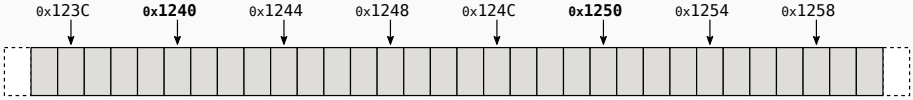


`MOVD [0x123C], xmm0`	✓	`[0x123C]` ← `xmm0(32:0)`
`MOVQ xmm0, [0x1245]`	✓	`xmm0(64:0)` ← `[0x1245]`
`MOVDQA xmm0, [0x1245]`	×	Error dirección no alineada.

Operaciones Load/Store

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:

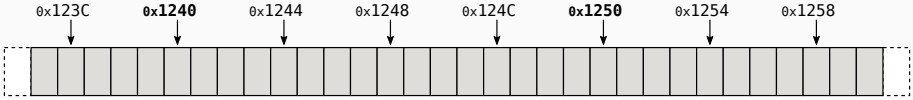


`MOVD [0x123C], xmm0`	✓	`[0x123C]` ← `xmm0(32:0)`
`MOVQ xmm0, [0x1245]`	✓	`xmm0(64:0)` ← `[0x1245]`
`MOVDQA xmm0, [0x1245]`	×	Error dirección no alineada.
`MOVDQA [0x1250], xmm0`	✓	`[0x1250]` ← `xmm0(128:0)`

Operaciones Load/Store

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(32:0)`
`MOVQ xmm0, [0x1245]`	✓	`xmm0(64:0)` ← `[0x1245]`
`MOVDQA xmm0, [0x1245]`	×	Error dirección no alineada.
`MOVDQA [0x1250], xmm0`	✓	`[0x1250]` ← `xmm0(128:0)`
`MOVSS xmm0, [0x1248]`	✓	`xmm0(32:0)` ← `[0x1248]` ; sobre punto flotante

Operaciones Load/Store

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(32:0)`
`MOVQ xmm0, [0x1245]`	✓	`xmm0(64:0)` ← `[0x1245]`
`MOVDQA xmm0, [0x1245]`	×	Error dirección no alineada.
`MOVDQA [0x1250], xmm0`	✓	`[0x1250]` ← `xmm0(128:0)`
`MOVSS xmm0, [0x1248]`	✓	`xmm0(32:0)` ← `[0x1248]` ; sobre punto flotante
`MOVUPS [0x1258], xmm0`	✓	`[0x1258]` ← `xmm0(128:0)` ; sobre punto flotante

Operaciones Load/Store

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 word to dqword

Operaciones Load/Store

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 word to dqword

Operaciones Load/Store

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 word to dqword

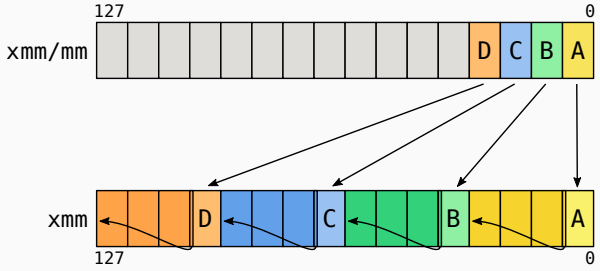
Ejemplos:

Operaciones Load/Store

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 word to dqword

Ejemplos:

`PMOVSXBD xmm0, xmm0` ✓

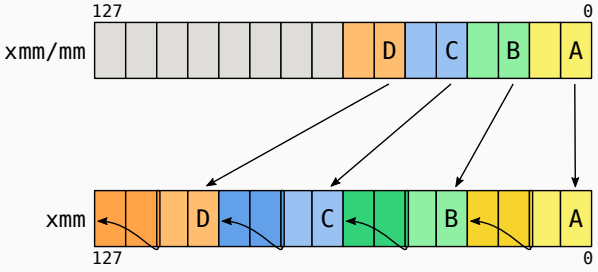


Operaciones Load/Store

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 word to dqword

Ejemplos:

```
PMOVSXBD xmm0, xmm0      ✓
PMOVZXWD xmm0, [data]    ✓
```

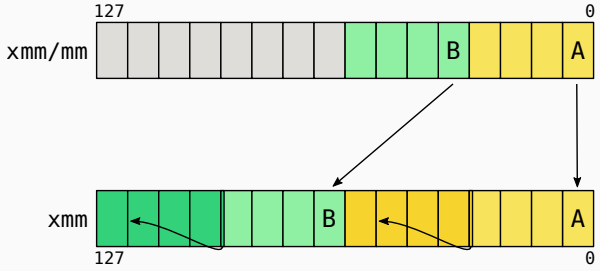


Operaciones Load/Store

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 word to dqword

Ejemplos:

```
PMOVSXBD xmm0, xmm0      ✓
PMOVZXWD xmm0, [data]    ✓
PMOVZXDQ xmm0, xmm1      ✓
```



Operaciones Load/Store

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 word to dqword

Ejemplos:

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

Operaciones Load/Store

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 word to dqword

Ejemplos:

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

Operaciones Aritméticas

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

Operaciones Aritméticas

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

Ejemplos:

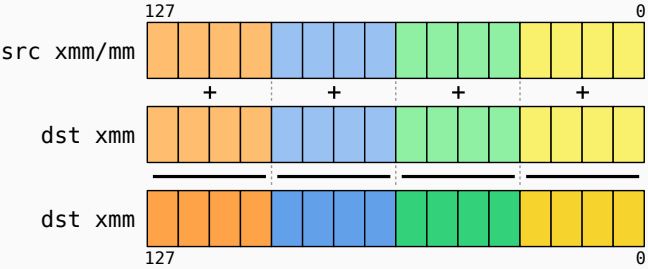
Operaciones Aritméticas

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

Ejemplos:

```
PADDD xmm0, xmm1
```

✓

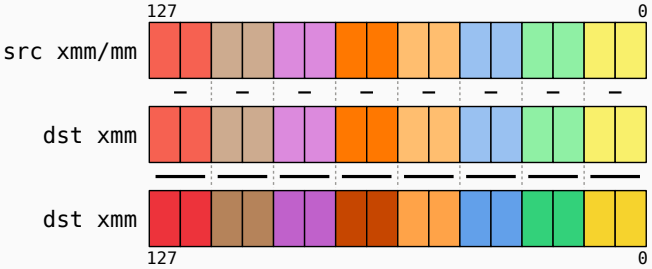


Operaciones Aritméticas

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

Ejemplos:

```
PADDD xmm0, xmm1        ✓
PSUBW xmm0, [data]      ✓
```

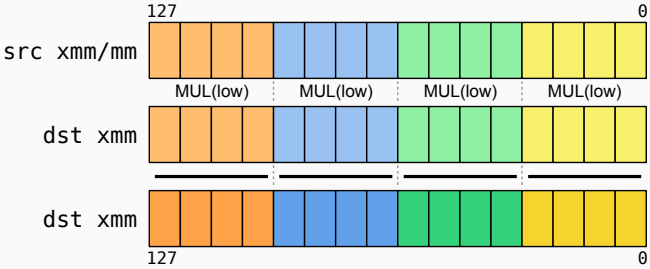


Operaciones Aritméticas

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

Ejemplos:

```
PADDD xmm0, xmm1        ✓
PSUBW xmm0, [data]      ✓
PMULLD xmm0, xmm1       ✓
```

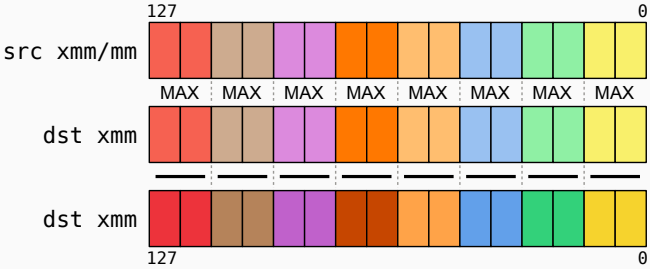


Operaciones Aritméticas

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

Ejemplos:

```
PADDD xmm0, xmm1          ✓
PSUBW xmm0, [data]        ✓
PMULLD xmm0, xmm1         ✓
PMAXSW xmm0, [data]       ✓
```



Operaciones Aritméticas

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

Ejemplos:

```
PADDD xmm0, xmm1        ✓
PSUBW xmm0, [data]      ✓
PMULLD xmm0, xmm1       ✓
PMAXSW xmm0, [data]     ✓
PMINSB [data], xmm0     ×   Modo de direccionamiento invalido.
```

Operaciones Aritméticas

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

Operaciones Aritméticas

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

Ejemplos:

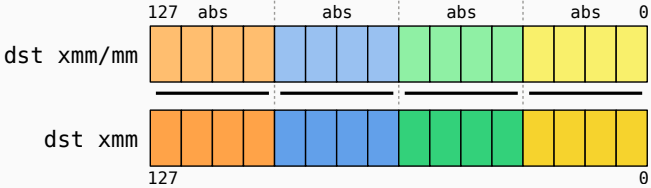
Operaciones Aritméticas

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

Ejemplos:

```
PABSD xmm0, xmm0
```

✓

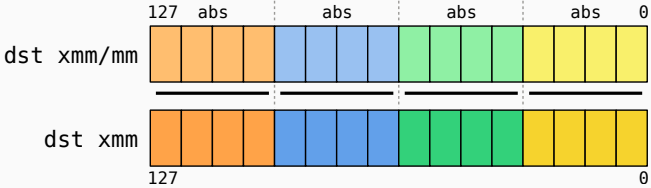


Operaciones Aritméticas

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]      ✓
```



Operaciones Aritméticas

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.
```

Operaciones Aritméticas: P.F. (Ref)

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

Operaciones Aritméticas: P.F. (Ref)

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

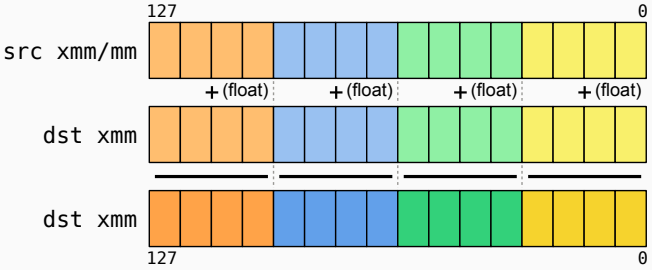
Ejemplos:

Operaciones Aritméticas: P.F. (Ref)

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

Ejemplos:

```
ADDPS xmm0, [data]
```
✓

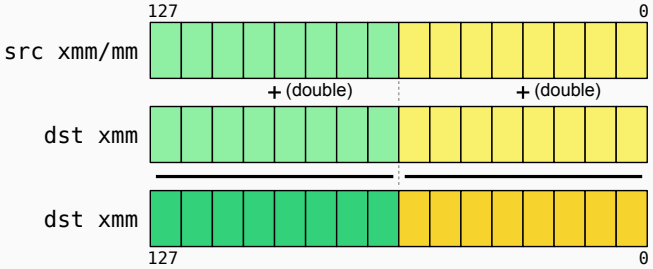


Operaciones Aritméticas: P.F. (Ref)

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

Ejemplos:

```
ADDPS xmm0, [data]    ✓
ADDPD xmm0, [data]    ✓
```

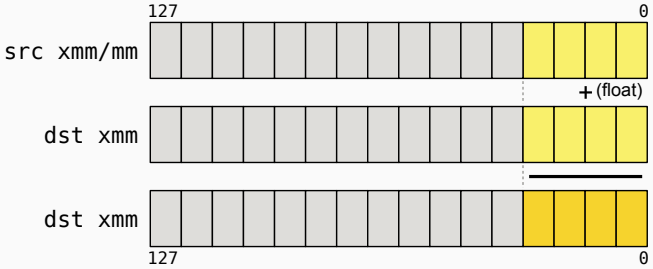


Operaciones Aritméticas: P.F. (Ref)

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

Ejemplos:

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

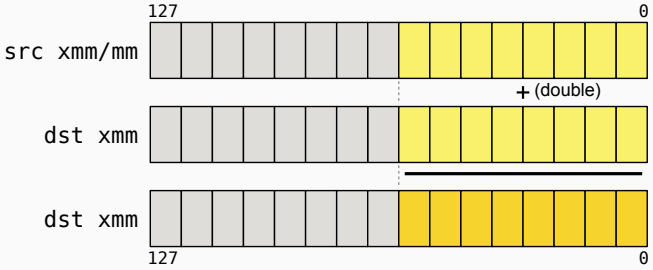


Operaciones Aritméticas: P.F. (Ref)

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

Ejemplos:

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



Operaciones Aritméticas: P.F. (Ref)

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

Ejemplos:

```
ADDPS xmm0, [data]      ✓
ADDPD xmm0, [data]      ✓
ADDSS xmm0, [data]      ✓
ADDSD xmm0, [data]      ✓
MINSD [data], xmm0      ×   Modo de direccionamiento invalido.
```

Operaciones Aritméticas: SQRT (Ref.)

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

Operaciones Aritméticas: SQRT (Ref.)

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

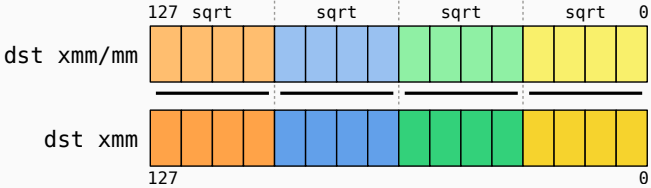
Ejemplos:

Operaciones Aritméticas: SQRT (Ref.)

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

Ejemplos:

`SQRTPS xmm0, [data]` ✓

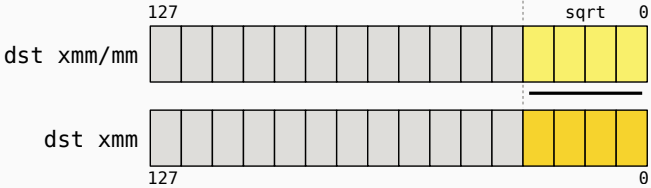


Operaciones Aritméticas: SQRT (Ref.)

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]   ✓
```



Operaciones Aritméticas: SQRT (Ref.)

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.
```

Operaciones Aritméticas: Saturación (Ref)

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

Operaciones Aritméticas: Saturación (Ref)

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

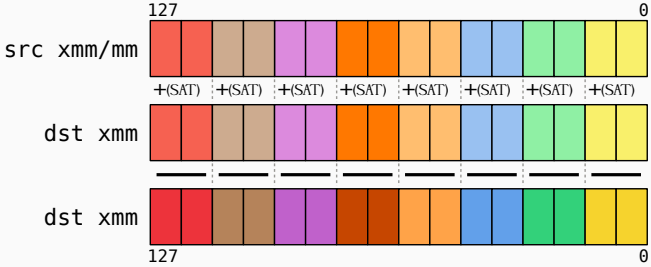
Ejemplos:

Operaciones Aritméticas: Saturación (Ref)

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

Ejemplos:

`PADDSW xmm0, xmm0` ✓

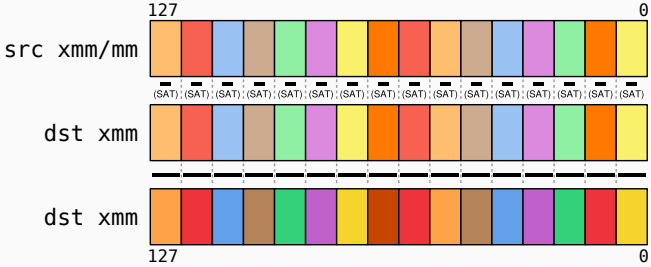


Operaciones Aritméticas: Saturación (Ref)

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

Ejemplos:

```
PADDSW xmm0, xmm0          ✓
PSUBUSB xmm0, [data]       ✓
```



Operaciones Aritméticas: Saturación (Ref)

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

Ejemplos:

```
PADDSW xmm0, xmm0          ✓
PSUBUSB xmm0, [data]       ✓
PSUBSW [data], xmm0        ×   Modo de direccionamiento invalido.
```

Ejemplo

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)
```

Incrementar Brillo


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

section .text

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

Incrementar Brillo


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

section .text

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

Incrementar Brillo


```
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 |
```

Incrementar Brillo


```
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 |
    .ciclo:
        movdqu xmm0, [rdi]  ; xmm0 = |  d15 | ... |  d0 |
```

Incrementar Brillo


```
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 |
    .ciclo:
        movdqu xmm0, [rdi]  ; xmm0 = |  d15 | ... |  d0 |
        paddusb xmm0, xmm8  ; xmm0 = |d15+10| ... |d0+10|
```

Incrementar Brillo


```
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 |
    .ciclo:
        movdqu xmm0, [rdi]  ; xmm0 = |  d15 | ... |  d0 |
        paddusb xmm0, xmm8  ; xmm0 = |d15+10| ... |d0+10|
        movdqu [rdi], xmm0
```

Incrementar Brillo

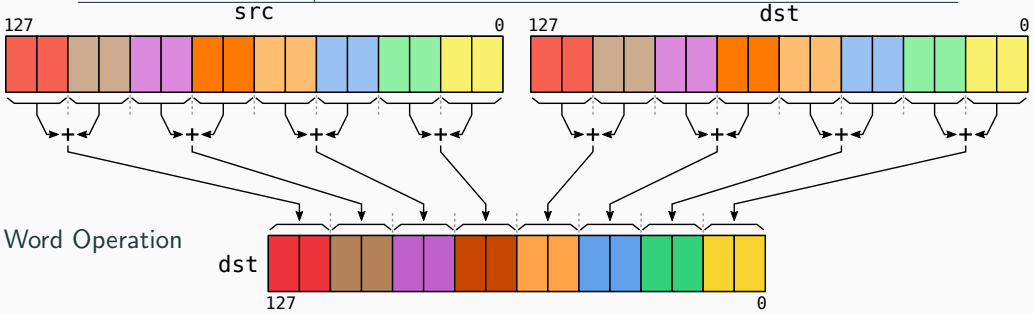

```
section .rodata
diez: times 16 db 10

section .text

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

Operaciones Aritméticas horizontales

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

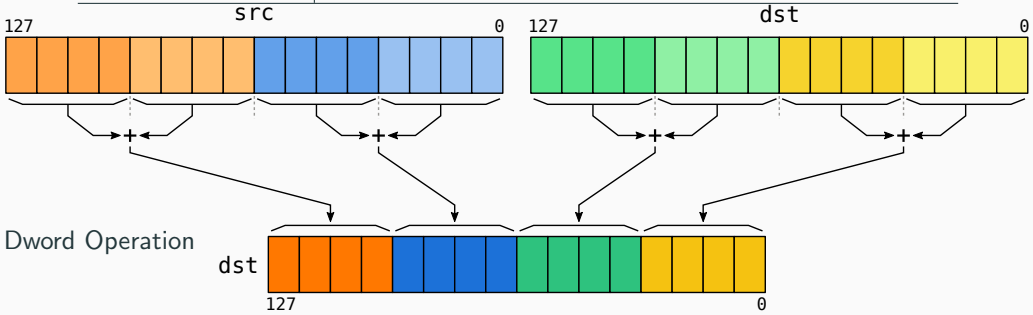
Operaciones Aritméticas horizontales

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



Operaciones Aritméticas horizontales

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



Operaciones Lógicas

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.

Operaciones Lógicas

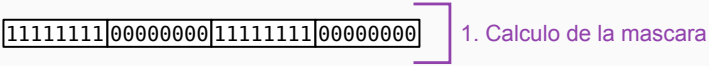
PAND	PANDN	POR	PXOR	Operaciones lógicas para enteros.
ANDPS	ANDNPS	ORPS	XORPS	Operaciones lógicas para *float*.
ANDPD	ANDNPD	ORPD	XORPD	Operaciones lógicas para *double*.

- 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.

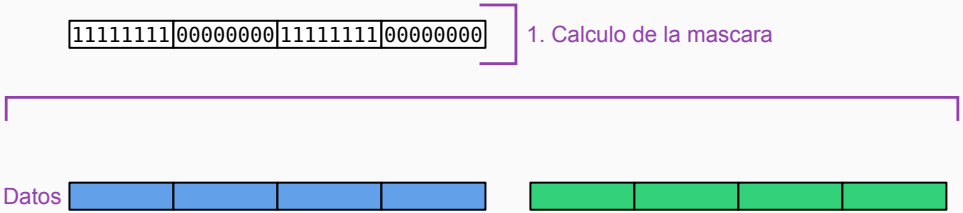
PSLLW	PSLLD	PSLLQ	PSLLDQ*
PSRLW	PSRLD	PSRLQ	PSRLDQ*
PSRAW	PSRAD		

- 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.

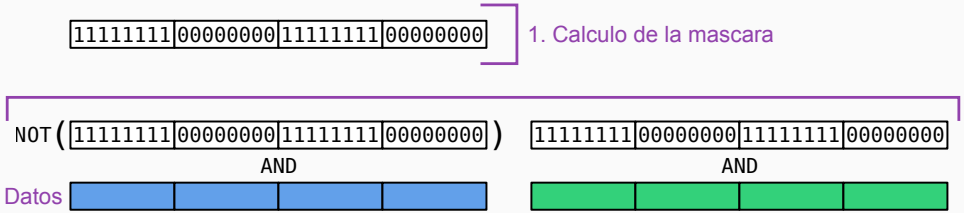
Técnica: Operatoria con mascaras



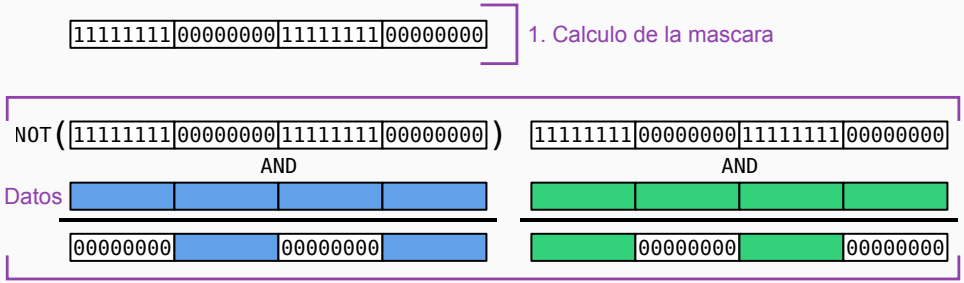
Técnica: Operatoria con mascaras



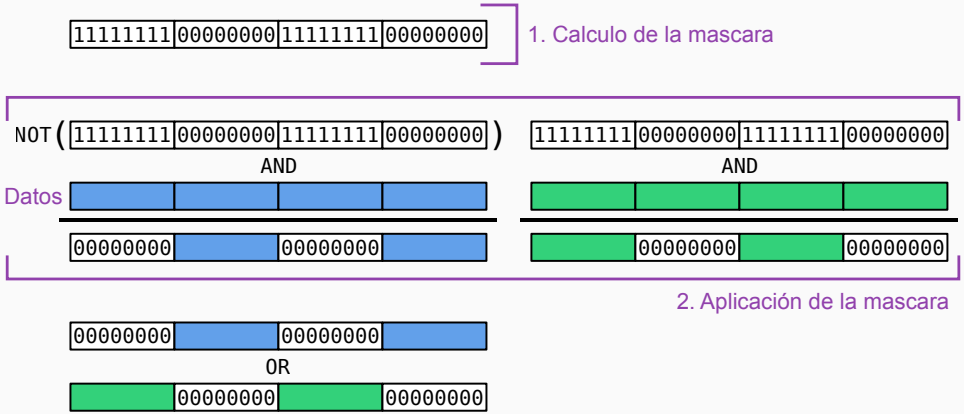
Técnica: Operatoria con mascaras



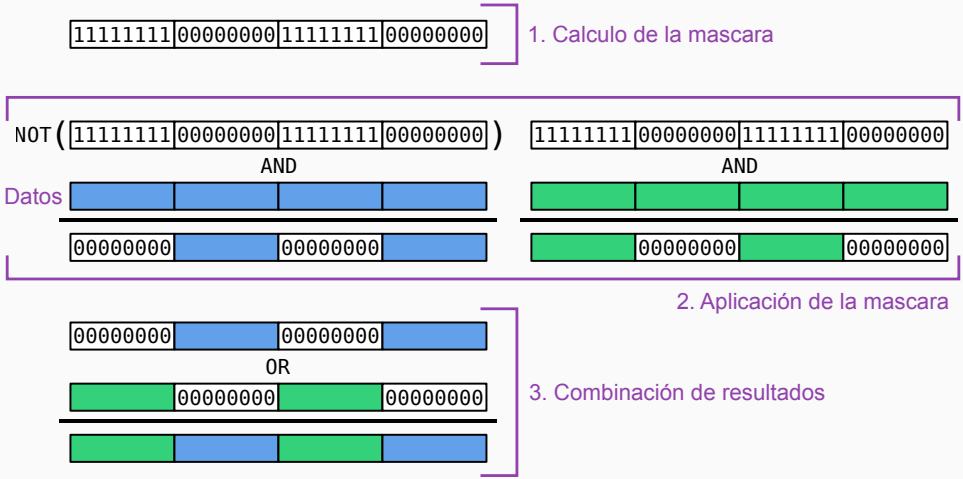
Técnica: Operatoria con mascaras



Técnica: Operatoria con mascaras



Técnica: Operatoria con mascaras



Operaciones de comparación

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

Operaciones de comparación

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

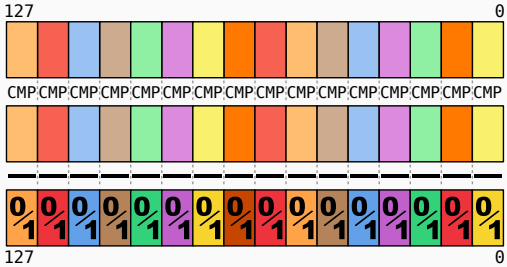
Ejemplos:

Operaciones de comparación

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

Ejemplos:

`PCMPEQB xmm0, [data]` ✓

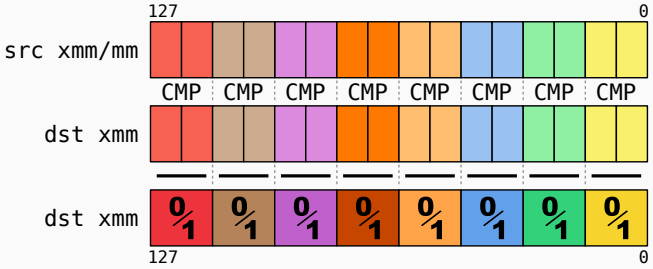


Operaciones de comparación

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]    ✓
```

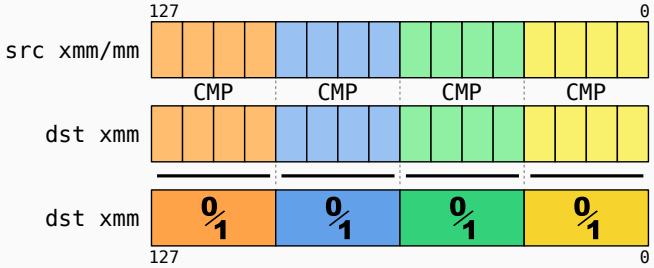


Operaciones de comparación

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]   ✓
```

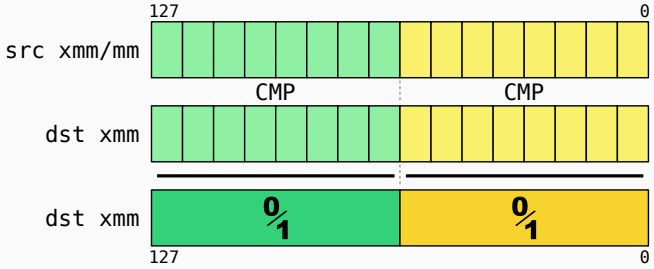


Operaciones de comparación

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]   ✓
```



Operaciones de comparación

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.
```

Operaciones de comparación

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	$A, B = Ordered$

Operaciones de comparación

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	$A \leqslant B$
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	$A, B = Ordered$

Operaciones de comparación

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	$A \leqslant B$
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	$A, B = Ordered$

Operaciones de comparación

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	$A \leqslant B$
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	$A, B = Ordered$

Operaciones de comparación

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	$A \leqslant B$
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	$A, B = Ordered$

Ejemplo (Ref.)

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)
```

Suma pares


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

Suma pares


```
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 |
```

Suma pares


```
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:
        pmovsxwd xmm0, [rdi] ; (ejemplo) xmm0 = | 00001233 | 00007314 | 00003011 | FFFF9311 |
```

Suma pares


```
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:
        pmovsxwd xmm0, [rdi] ; (ejemplo) xmm0 = | 00001233 | 00007314 | 00003011 | FFFF9311 |
        pabsd xmm1, xmm0     ; (ejemplo) xmm1 = | 00001233 | 00007314 | 00003011 | 00006CEE |
```

Suma pares


```
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:
        pmovsxwd 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 |
```

Suma pares


```
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:
        pmovsxwd 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 |
```

Suma pares


```
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:
        pmovsxwd 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 |
```

Suma pares


```
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:
        pmovsxwd 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 |
        paddd  xmm8, xmm1    ;           xmm8 = |   SUM3   |   SUM2   |   SUM1   |   SUM0   |
        add rdi, 8
    loop .ciclo
```

Suma pares

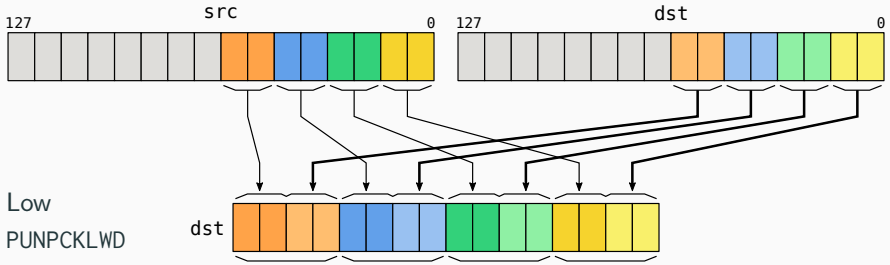

```
sumarpares: ; rdi = int16_t *v
    push rbp
    mov rbp,rsp
    mov rcx, (128 >> 2)  ; rcx = 128 / 4
    pxor xmm8, xmm8      ; xmm8 = | 0 | 0 | 0 | 0 |
    .ciclo:
        pmovsxwd xmm0, [rdi] ; (ejemplo) xmm0 = | 00001233 | 00007314 | 00003011 | FFFF9311 |
        pabsd xmm1, xmm0     ; (ejemplo) 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 |
        paddd  xmm8, xmm1    ;           xmm8 = |   SUM3   |   SUM2   |   SUM1   |   SUM0   |
        add rdi, 8
    loop .ciclo
    phaddd xmm8, xmm8  ; xmm8 = | ... | ... | SUM3+SUM2 |      SUM1+SUM0      |
    phaddd xmm8, xmm8  ; xmm8 = | ... | ... |    ...    | SUM3+SUM2+SUM1+SUM0 |
```

Suma pares


```
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:
        pmovsxwd 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 |
        paddd  xmm8, xmm1    ;           xmm8 = |   SUM3   |   SUM2   |   SUM1   |   SUM0   |
        add rdi, 8
    loop .ciclo
    phaddd xmm8, xmm8  ; xmm8 = | ... | ... | SUM3+SUM2 |      SUM1+SUM0      |
    phaddd xmm8, xmm8  ; xmm8 = | ... | ... |    ...    | SUM3+SUM2+SUM1+SUM0 |
    movd eax, xmm8     ; eax = SUM3+SUM2+SUM1+SUM0
    pop rbp
    ret
```

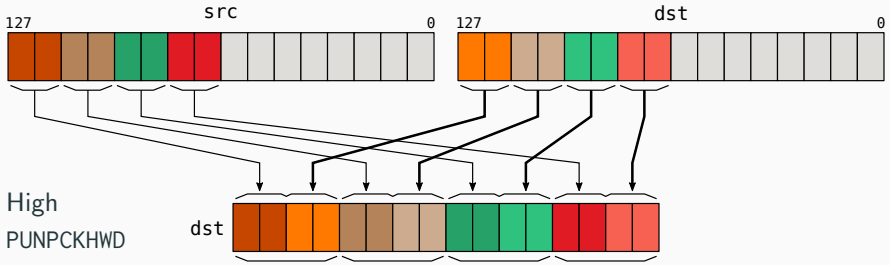
Operaciones de desempaquetado (Unpack)

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



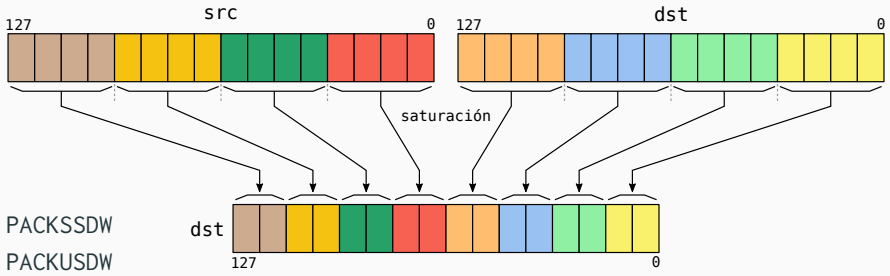
Operaciones de desempaquetado (Unpack)

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)

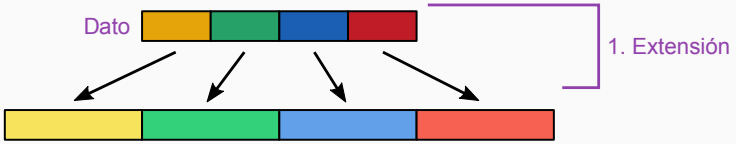
PACKSSDW	Packs 32 bits (signado) a 16 bits (signado) usando saturation
PACKUSDW	Packs 32 bits (signado) a 16 bits (sin signo) usando saturation
PACKSSWB	Packs 16 bits (signado) a 8 bits (signado) usando saturation
PACKUSWB	Packs 16 bits (signado) a 8 bits (sin signo) usando saturation



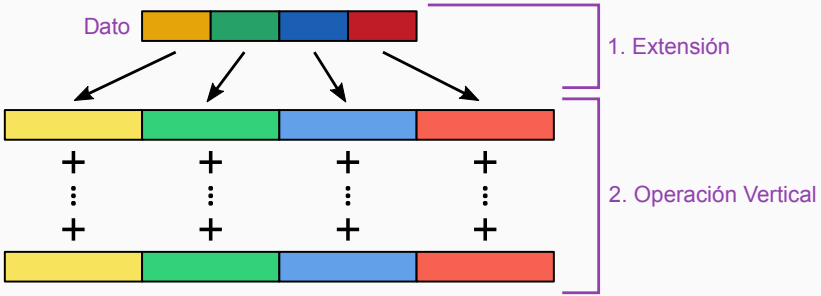
Técnica: Operatoria de desempaquetado y empaquetado

Técnica: Operatoria de desempaquetado y empaquetado

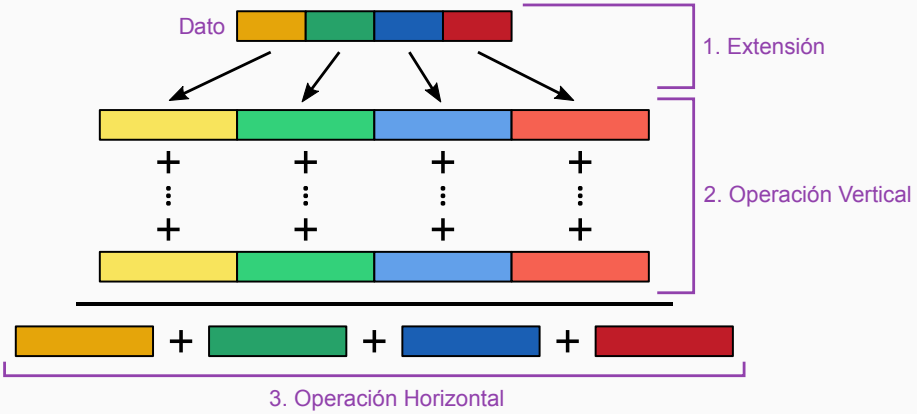
Técnica: Operatoria de desempaquetado y empaquetado



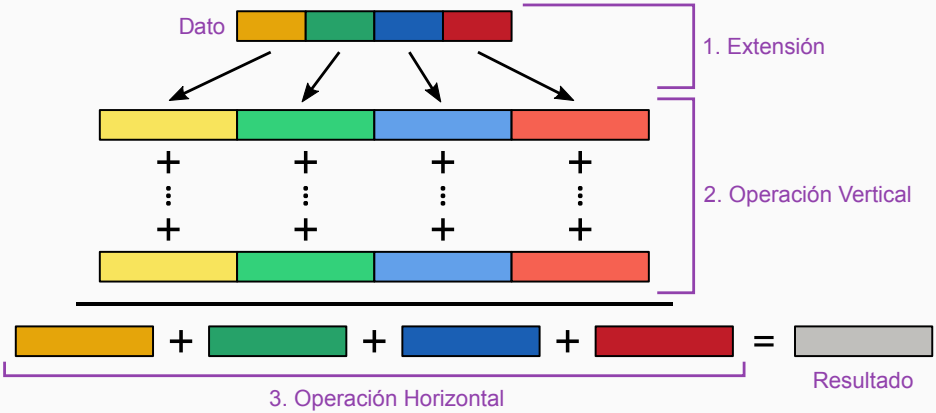
Técnica: Operatoria de desempaquetado y empaquetado



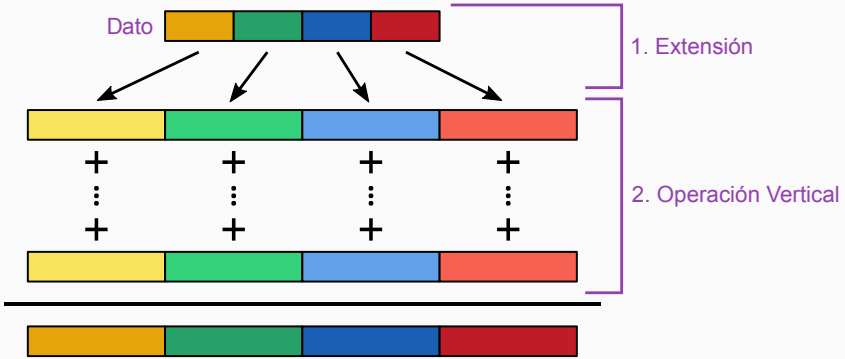
Técnica: Operatoria de desempaquetado y empaquetado



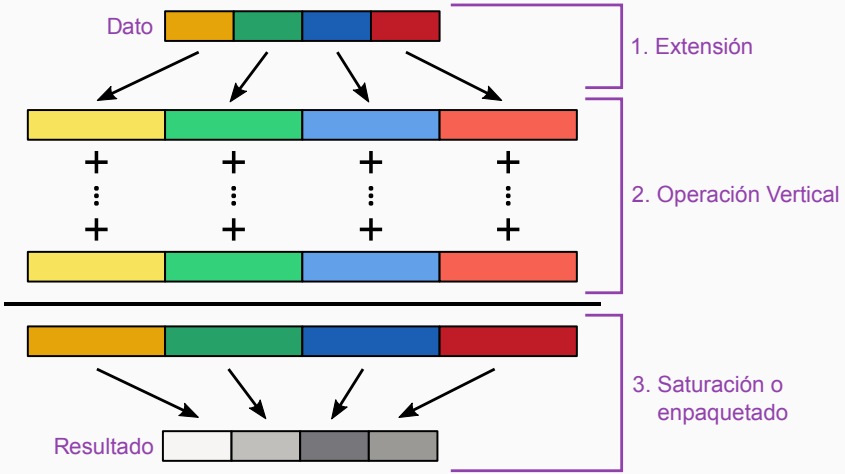
Técnica: Operatoria de desempaquetado y empaquetado



Técnica: Operatoria de desempaquetado y empaquetado



Técnica: Operatoria de desempaquetado y empaquetado



Ejemplo

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)
```

Multiplicar vectores


```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado
push rbp
mov rbp,rsp
mov rcx, (128 >> 2) ; rcx = 128 / 8
```

Multiplicar vectores


```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado
push rbp
mov rbp,rsp
mov rcx, (128 >> 2) ; 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 |
```

Multiplicar vectores


```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado
push rbp
mov rbp,rsp
mov rcx, (128 >> 2) ; 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) |
```

Multiplicar vectores


```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado
push rbp
mov rbp,rsp
mov rcx, (128 >> 2) ; 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(0a*b0) |
    punpckhwd xmm1, xmm2  ; xmm1 = | hi:low(a7*b7)    ...    hi:low(a4*b4) |
```

Multiplicar vectores


```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado
push rbp
mov rbp,rsp
mov rcx, (128 >> 2) ; 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(0a*b0) |
    punpckhwd xmm1, xmm2  ; xmm1 = | hi:low(a7*b7)    ...    hi:low(a4*b4) |
    movdqa [rdx], xmm0
    movdqa [rdx+16], xmm1
```

Multiplicar vectores


```
mulvec: ; rdi = int16_t *v1, rsi = int16_t *v2, rdx = int32_t *resultado
push rbp
mov rbp,rsp
mov rcx, (128 >> 2) ; 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(0a*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?