Convención de llamada linux x86_64

Parámetros y valores de retorno 64 bits

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

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

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

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

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

Parámetros y valores de retorno 32 bits

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

Modos de acceso a memoria

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

Registros de Propósito General

Intel 64

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

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

Alineación de structs

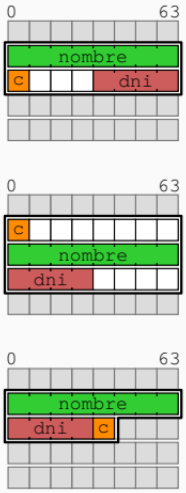
- Cada variable debe estar alineada a una pocisión multiplo de su tamaño.

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

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

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

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



Interacción con C

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

Secciones del código

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

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

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

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

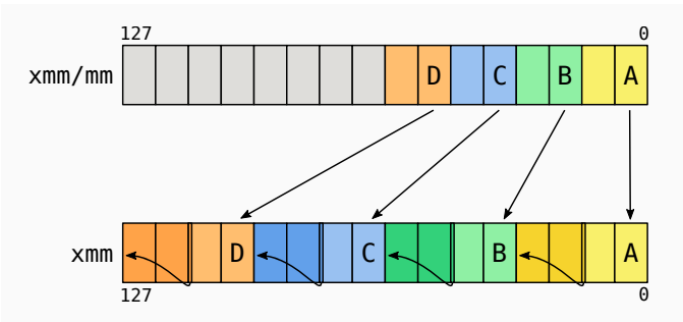
SIMD

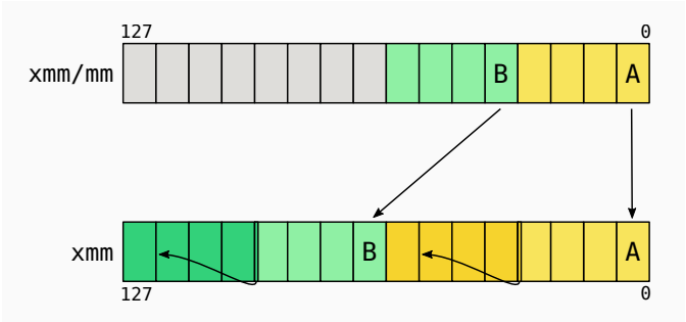
MOV r-m m-r

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

Packed MOV r-r r-m

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

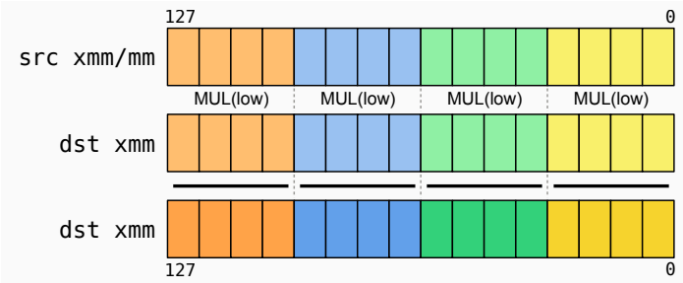




Packed operaciones aritmeticas r-r r-m

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

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



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

Packed operaciones fp r-r r-m

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

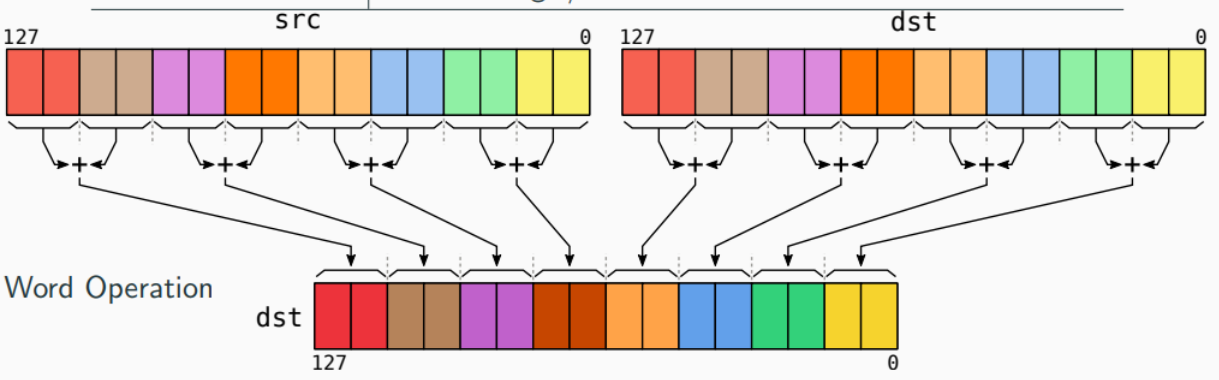
SQRTSS	SQRTPS	Square root of Scalar/Packed Single FP values
SQRTSD	SQRTPD	Square root of Scalar/Packed Double FP values

Packed operaciones saturadas con enteros r-r r-m

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

Packed operaciones horizontales r-r r-m

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



Packed operaciones lógicas y shifts r-r r-m

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

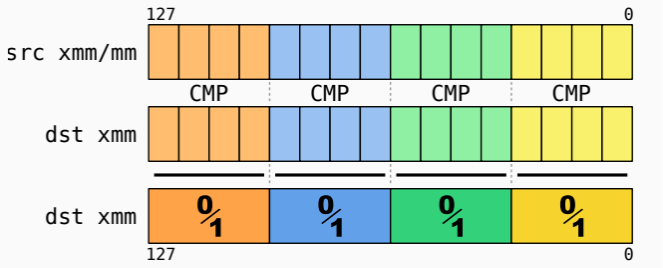
- Actuan lógicamente sobre todo el registro, sin importa el tamaño del operando.
- La distinción entre PS y PD se debe a meta información para el procesador.

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

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

Packed compare enteros y flotantes r-r r-m

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



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

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

Desenpaquetado

Notar que hay para tomar los lows y highs

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