

# Preliminares

David Alejandro González Márquez

Departamento de Computación  
Facultad de Ciencias Exactas y Naturales  
Universidad de Buenos Aires

# Preliminares

## Ejercicio

Describe en sus palabras las funciones de las siguientes aplicaciones:

- Compilador
- Ensamblador
- Linker

# Preliminares

## Ejercicio

Describe en sus palabras las funciones de las siguientes aplicaciones:

- Compilador
- Ensamblador
- Linker

- **Compilador:** Toma código en un lenguaje de alto nivel y lo transforma a código ensamblador de alguna arquitectura.

# Preliminares

## Ejercicio

Describe en sus palabras las funciones de las siguientes aplicaciones:

- Compilador
- Ensamblador
- Linker

- **Compilador:** Toma código en un lenguaje de alto nivel y lo transforma a código ensamblador de alguna arquitectura.
- **Ensamblador:** Toma código en lenguaje ensamblador y lo traduce a código de máquina, generando un archivo objeto. Resuelve nombres, simbólicos y traduce los mnemónicos.

# Preliminares

## Ejercicio

Describe en sus palabras las funciones de las siguientes aplicaciones:

- Compilador
- Ensamblador
- Linker

- **Compilador:** Toma código en un lenguaje de alto nivel y lo transforma a código ensamblador de alguna arquitectura.
- **Ensamblador:** Toma código en lenguaje ensamblador y lo traduce a código de máquina, generando un archivo objeto. Resuelve nombres, simbólicos y traduce los mnemónicos.
- **Linker:** Toma varios archivos objeto y los transforma en un ejecutable.

## Preliminares

### Ejercicio

Muestre cómo se almacenan en memoria los siguientes datos en procesadores Big-Endian y Little-Endian:

DB 12h

DD 12345678h

DB 12h, 34h

DD 12345678h, 9ABCDEF1h

DW 1234h

DQ 123456789ABCDEF1h

DW 1234h, 5678h

DB '1234'

## Preliminares

### Ejercicio

Muestre cómo se almacenan en memoria los siguientes datos en procesadores Big-Endian y Little-Endian:

DB 12h	DD 12345678h
DB 12h, 34h	DD 12345678h, 9ABCDEF1h
DW 1234h	DQ 123456789ABCDEF1h
DW 1234h, 5678h	DB '1234'

DB, DW, DD, DQ: Pseudo-instrucciones del ensamblador, indican cómo definir datos en el archivo objeto.

**NO se ejecutan por la CPU, las interpreta el ensamblador.**

## Preliminares

### Ejercicio

Muestre cómo se almacenan en memoria los siguientes datos en procesadores Big-Endian y Little-Endian:

DB 12h	DD 12345678h
DB 12h, 34h	DD 12345678h, 9ABCDEF1h
DW 1234h	DQ 123456789ABCDEF1h
DW 1234h, 5678h	DB '1234'

DB, DW, DD, DQ: Pseudo-instrucciones del ensamblador, indican cómo definir datos en el archivo objeto.

**NO se ejecutan por la CPU, las interpreta el ensamblador.**

#### **Big Endian:**

el byte más significativo en la posición de memoria menos significativa.

#### **Little Endian:**

el byte más significativo en la posición de memoria más significativa.



# Preliminares

## Caso

DB 12h

---

DB 12h, 34h

---

DW 1234h

---

DW 1234h, 5678h

---

DD 12345678h

---

DD 12345678h, 9ABCDEF1h

---

DQ 123456789ABCDEF1h

---

DB '1234'

# Preliminares

**Caso**

**Memoria** → --|..|..|..|..|..|..|..|..|++

DB 12h

---

DB 12h, 34h

---

DW 1234h

---

DW 1234h, 5678h

---

DD 12345678h

---

DD 12345678h, 9ABCDEF1h

---

DQ 123456789ABCDEF1h

---

DB '1234'

# Preliminares

**Caso**

**Memoria** → -- | .. | .. | .. | .. | .. | .. | .. | ++

DB 12h

big endian .. | 12 | ..

---

DB 12h, 34h

---

DW 1234h

---

DW 1234h, 5678h

---

DD 12345678h

---

DD 12345678h, 9ABCDEF1h

---

DQ 123456789ABCDEF1h

---

DB '1234'

# Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
<hr/>		
DB 12h, 34h		
<hr/>		
DW 1234h		
<hr/>		
DW 1234h, 5678h		
<hr/>		
DD 12345678h		
<hr/>		
DD 12345678h, 9ABCDEF1h		
<hr/>		
DQ 123456789ABCDEF1h		
<hr/>		
DB '1234'		

# Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
DB 12h, 34h	big endian	..   12   34   ..
DW 1234h		
DW 1234h, 5678h		
DD 12345678h		
DD 12345678h, 9ABCDEF1h		
DQ 123456789ABCDEF1h		
DB '1234'		

# Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
DB 12h, 34h	big endian	..   12   34   ..
	little endian	..   12   34   ..
DW 1234h		
DW 1234h, 5678h		
DD 12345678h		
DD 12345678h, 9ABCDEF1h		
DQ 123456789ABCDEF1h		
DB '1234'		

# Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
DB 12h, 34h	big endian	..   12   34   ..
	little endian	..   12   34   ..
DW 1234h	big endian	..   12   34   ..
DW 1234h, 5678h		
DD 12345678h		
DD 12345678h, 9ABCDEF1h		
DQ 123456789ABCDEF1h		
DB '1234'		

# Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
DB 12h, 34h	big endian	..   12   34   ..
	little endian	..   12   34   ..
DW 1234h	big endian	..   12   34   ..
	little endian	..   34   12   ..
DW 1234h, 5678h		
DD 12345678h		
DD 12345678h, 9ABCDEF1h		
DQ 123456789ABCDEF1h		
DB '1234'		



# Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
DB 12h, 34h	big endian	..   12   34   ..
	little endian	..   12   34   ..
DW 1234h	big endian	..   12   34   ..
	little endian	..   34   12   ..
DW 1234h, 5678h	big endian	..   12   34   56   78   ..
DD 12345678h		
DD 12345678h, 9ABCDEF1h		
DQ 123456789ABCDEF1h		
DB '1234'		

# Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
DB 12h, 34h	big endian	..   12   34   ..
	little endian	..   12   34   ..
DW 1234h	big endian	..   12   34   ..
	little endian	..   34   12   ..
DW 1234h, 5678h	big endian	..   12   34   56   78   ..
	little endian	..   34   12   78   56   ..
DD 12345678h		
DD 12345678h, 9ABCDEF1h		
DQ 123456789ABCDEF1h		
DB '1234'		

# Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
DB 12h, 34h	big endian	..   12   34   ..
	little endian	..   12   34   ..
DW 1234h	big endian	..   12   34   ..
	little endian	..   34   12   ..
DW 1234h, 5678h	big endian	..   12   34   56   78   ..
	little endian	..   34   12   78   56   ..
DD 12345678h	big endian	..   12   34   56   78   ..
	little endian	..   78   56   34   12   ..
DD 12345678h, 9ABCDEF1h		
DQ 123456789ABCDEF1h		
DB '1234'		

# Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
DB 12h, 34h	big endian	..   12   34   ..
	little endian	..   12   34   ..
DW 1234h	big endian	..   12   34   ..
	little endian	..   34   12   ..
DW 1234h, 5678h	big endian	..   12   34   56   78   ..
	little endian	..   34   12   78   56   ..
DD 12345678h	big endian	..   12   34   56   78   ..
	little endian	..   78   56   34   12   ..
DD 12345678h, 9ABCDEF1h	big endian	..   12   34   56   78   9A   BC   DE   F1   ..
	little endian	..   78   56   34   12   F1   DE   BC   9A   ..
DQ 123456789ABCDEF1h		
DB '1234'		

# Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
DB 12h, 34h	big endian	..   12   34   ..
	little endian	..   12   34   ..
DW 1234h	big endian	..   12   34   ..
	little endian	..   34   12   ..
DW 1234h, 5678h	big endian	..   12   34   56   78   ..
	little endian	..   34   12   78   56   ..
DD 12345678h	big endian	..   12   34   56   78   ..
	little endian	..   78   56   34   12   ..
DD 12345678h, 9ABCDEF1h	big endian	..   12   34   56   78   9A   BC   DE   F1   ..
	little endian	..   78   56   34   12   F1   DE   BC   9A   ..
DQ 123456789ABCDEF1h	big endian	..   12   34   56   78   9A   BC   DE   F1   ..
	little endian	..   F1   DE   BC   9A   78   56   34   12   ..
DB '1234'		

# Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
DB 12h, 34h	big endian	..   12   34   ..
	little endian	..   12   34   ..
DW 1234h	big endian	..   12   34   ..
	little endian	..   34   12   ..
DW 1234h, 5678h	big endian	..   12   34   56   78   ..
	little endian	..   34   12   78   56   ..
DD 12345678h	big endian	..   12   34   56   78   ..
	little endian	..   78   56   34   12   ..
DD 12345678h, 9ABCDEF1h	big endian	..   12   34   56   78   9A   BC   DE   F1   ..
	little endian	..   78   56   34   12   F1   DE   BC   9A   ..
DQ 123456789ABCDEF1h	big endian	..   12   34   56   78   9A   BC   DE   F1   ..
	little endian	..   F1   DE   BC   9A   78   56   34   12   ..
DB '1234'	big endian	..   31   32   33   34   ..

## Preliminares

Caso	Memoria →	--   ..   ..   ..   ..   ..   ..   ..   ++
DB 12h	big endian	..   12   ..
	little endian	..   12   ..
DB 12h, 34h	big endian	..   12   34   ..
	little endian	..   12   34   ..
DW 1234h	big endian	..   12   34   ..
	little endian	..   34   12   ..
DW 1234h, 5678h	big endian	..   12   34   56   78   ..
	little endian	..   34   12   78   56   ..
DD 12345678h	big endian	..   12   34   56   78   ..
	little endian	..   78   56   34   12   ..
DD 12345678h, 9ABCDEF1h	big endian	..   12   34   56   78   9A   BC   DE   F1   ..
	little endian	..   78   56   34   12   F1   DE   BC   9A   ..
DQ 123456789ABCDEF1h	big endian	..   12   34   56   78   9A   BC   DE   F1   ..
	little endian	..   F1   DE   BC   9A   78   56   34   12   ..
DB '1234'	big endian	..   31   32   33   34   ..
	little endian	..   31   32   33   34   ..

## Preliminares

### Ejercicio

¿Cuál es el rango de representación de los números enteros sin signo con 8, 16 y 32 bits de precisión? ¿Cuál es el rango de representación de los números enteros en complemento a dos con 8, 16 y 32 bits de precisión?



## Preliminares

### Ejercicio

¿Cuál es el rango de representación de los números enteros sin signo con 8, 16 y 32 bits de precisión? ¿Cuál es el rango de representación de los números enteros en complemento a dos con 8, 16 y 32 bits de precisión?

	Sin signo	Con signo
8	0 a 255	-128 a 127

## Preliminares

### Ejercicio

¿Cuál es el rango de representación de los números enteros sin signo con 8, 16 y 32 bits de precisión? ¿Cuál es el rango de representación de los números enteros en complemento a dos con 8, 16 y 32 bits de precisión?

	Sin signo	Con signo
8	0 a 255	−128 a 127
16	0 a 65535	−32768 a 32767
32	0 a 4294967295	−2147483648 a 2147483647

## Preliminares

### Ejercicio

¿Cuál es el rango de representación de los números enteros sin signo con 8, 16 y 32 bits de precisión? ¿Cuál es el rango de representación de los números enteros en complemento a dos con 8, 16 y 32 bits de precisión?

Sin signo	0	a	$2^n - 1$
Con signo	$-2^{n-1}$	a	$2^{n-1} - 1$

	Sin signo	Con signo
8	0 a 255	-128 a 127
16	0 a 65535	-32768 a 32767
32	0 a 4294967295	-2147483648 a 2147483647

## Preliminares

### Ejercicio

Expresa los números -123 y 123 en notación complemento a dos con 8 bits de precisión y realice la suma de estos dos números bit a bit.

## Preliminares

### Ejercicio

Expresa los números -123 y 123 en notación complemento a dos con 8 bits de precisión y realice la suma de estos dos números bit a bit. Luego, expresa los números 133 y 123 en notación binaria con 8 bits de precisión (notación sin signo), y realice la suma de estos dos números bit a bit.

## Preliminares

### Ejercicio

Expresa los números -123 y 123 en notación complemento a dos con 8 bits de precisión y realice la suma de estos dos números bit a bit. Luego, expresa los números 133 y 123 en notación binaria con 8 bits de precisión (notación sin signo), y realice la suma de estos dos números bit a bit.

$$\begin{array}{r} 123 = 01111011 \\ -123 = 10000101 \\ \hline 100000000 \end{array}$$

$$\begin{array}{r} 123 = 01111011 \\ 133 = 10000101 \\ \hline 100000000 \end{array}$$

## Preliminares

### Ejercicio

Expresa los números -123 y 123 en notación complemento a dos con 8 bits de precisión y realice la suma de estos dos números bit a bit. Luego, expresa los números 133 y 123 en notación binaria con 8 bits de precisión (notación sin signo), y realice la suma de estos dos números bit a bit. ¿Qué conclusión puede sacar al observar el resultados de ambas operaciones?

$$\begin{array}{r} 123 = 01111011 \\ -123 = 10000101 \\ \hline 10000000 \end{array}$$

$$\begin{array}{r} 123 = 01111011 \\ 133 = 10000101 \\ \hline 10000000 \end{array}$$

Esa es la razón por la cual no hay dos ADD/SUB, sino uno solo tanto para números con signo como sin signo.

**Es responsabilidad del programador saber con qué tipo de números se está operando, y prestar atención a los flags correctos.**

## Preliminares

### Ejercicio

Explique qué indican y cuándo se setean los flags de paridad (PF), de cero (ZF) y de signo (SF). Explique las diferencias entre el flag de carry (CF) y el flag de overflow (OF).



## Preliminares

### Ejercicio

Explique qué indican y cuándo se setean los flags de paridad (PF), de cero (ZF) y de signo (SF). Explique las diferencias entre el flag de carry (CF) y el flag de overflow (OF).

**Importante:**

Los flags se setean dependiendo de la operación. La interpretación depende del programador.

# Preliminares

## Ejercicio

Explique qué indican y cuándo se setean los flags de paridad (PF), de cero (ZF) y de signo (SF). Explique las diferencias entre el flag de carry (CF) y el flag de overflow (OF).

**Importante:**

Los flags se setean dependiendo de la operación. La interpretación depende del programador.

CF = 1	Bit más significativo en la suma. En la resta si hay <i>borrow</i> .
CF = 0	cualquier otro caso

---

# Preliminares

## Ejercicio

Explique qué indican y cuándo se setean los flags de paridad (PF), de cero (ZF) y de signo (SF). Explique las diferencias entre el flag de carry (CF) y el flag de overflow (OF).

**Importante:**

Los flags se setean dependiendo de la operación. La interpretación depende del programador.

CF = 1	Bit más significativo en la suma. En la resta si hay <i>borrow</i> .
CF = 0	cualquier otro caso
OF = 1	Si hay overflow (el resultado esta fuera de la representación)
OF = 0	cualquier otro caso

# Preliminares

## Ejercicio

Explique qué indican y cuándo se setean los flags de paridad (PF), de cero (ZF) y de signo (SF). Explique las diferencias entre el flag de carry (CF) y el flag de overflow (OF).

**Importante:**

Los flags se setean dependiendo de la operación. La interpretación depende del programador.

CF = 1	Bit más significativo en la suma. En la resta si hay <i>borrow</i> .
CF = 0	cualquier otro caso
OF = 1	Si hay overflow (el resultado esta fuera de la representación)
OF = 0	cualquier otro caso
PF = 1	Si el byte menos significativo tiene un número par de 1s
PF = 0	cualquier otro caso

# Preliminares

## Ejercicio

Explique qué indican y cuándo se setean los flags de paridad (PF), de cero (ZF) y de signo (SF). Explique las diferencias entre el flag de carry (CF) y el flag de overflow (OF).

**Importante:**

Los flags se setean dependiendo de la operación. La interpretación depende del programador.

CF = 1	Bit más significativo en la suma. En la resta si hay <i>borrow</i> .
CF = 0	cualquier otro caso
OF = 1	Si hay overflow (el resultado esta fuera de la representación)
OF = 0	cualquier otro caso
PF = 1	Si el byte menos significativo tiene un número par de 1s
PF = 0	cualquier otro caso
SF = 1	Si el bit más significativo es 1
SF = 0	cualquier otro caso

# Preliminares

## Ejercicio

Explique qué indican y cuándo se setean los flags de paridad (PF), de cero (ZF) y de signo (SF). Explique las diferencias entre el flag de carry (CF) y el flag de overflow (OF).

### Importante:

Los flags se setean dependiendo de la operación. La interpretación depende del programador.

CF = 1	Bit más significativo en la suma. En la resta si hay <i>borrow</i> .
CF = 0	cualquier otro caso
OF = 1	Si hay overflow (el resultado esta fuera de la representación)
OF = 0	cualquier otro caso
PF = 1	Si el byte menos significativo tiene un número par de 1s
PF = 0	cualquier otro caso
SF = 1	Si el bit más significativo es 1
SF = 0	cualquier otro caso
ZF = 1	Si el resultado es cero
ZF = 0	cualquier otro caso

# Preliminares

## Ejercicio

Indique cuáles son las condiciones para que se activen las siguientes instrucciones de salto: JA, JAE, JB, JBE, JE, JG, JGE, JL, JLE y JZ.

# Preliminares

## Ejercicio

Indique cuáles son las condiciones para que se activen las siguientes instrucciones de salto: JA, JAE, JB, JBE, JE, JG, JGE, JL, JLE y JZ.

Inst.	Condición		Interpretación
JA	CF=0 and ZF=0	Above	Mayor (sin signo)
JAE	CF=0	Above or Equal	Mayor o Igual (sin signo)
JB	CF=1	Below	Menor (sin signo)
JBE	CF=1 or ZF=1	Below or Equal	Menor o Igual (sin signo)



# Preliminares

## Ejercicio

Indique cuáles son las condiciones para que se activen las siguientes instrucciones de salto: JA, JAE, JB, JBE, JE, JG, JGE, JL, JLE y JZ.

Inst.	Condición		Interpretación
JA	CF=0 and ZF=0	Above	Mayor (sin signo)
JAE	CF=0	Above or Equal	Mayor o Igual (sin signo)
JB	CF=1	Below	Menor (sin signo)
JBE	CF=1 or ZF=1	Below or Equal	Menor o Igual (sin signo)
JE	ZF=1	Equal	Igual
JG	ZF=0 and SF=OF	Greater (signed)	Mayor (con Signo)
JGE	SF=OF	Greater or Equal (signed)	Mayor o Igual (con Signo)
JL	SF != OF	Less (signed)	Menor (con Signo)
JLE	ZF=1 or SF != OF	Less or Equal (signed)	Menor o Igual (con Signo)
JZ	ZF=1	Zero	Cero

# Instrucciones y Registros

Operaciones

ADD, SUB, MOV, SHL, JMP ...

# Instrucciones y Registros

Operaciones

ADD, SUB, MOV, SHL, JMP ... (ver manual)

# Instrucciones y Registros

## Operaciones

ADD, SUB, MOV, SHL, JMP ... (ver manual)

## Registros

**8 bits:** AL BL CL DL DIL SIL BPL SPL R8B ... R15B

**16 bits:** AX BX CX DX DI SI BP SP R8W ... R15W

**32 bits:** EAX EBX ECX EDX EDI ESI EBP ESP R8D ... R15D

**64 bits:** RAX RBX RCX RDX RSI RDI RBP RSP R8 ... R15

# Instrucciones y Registros

## Operaciones

ADD, SUB, MOV, SHL, JMP ... (ver manual)

## Registros

**8 bits:** AL BL CL DL DIL SIL BPL SPL R8B ... R15B

**16 bits:** AX BX CX DX DI SI BP SP R8W ... R15W

**32 bits:** EAX EBX ECX EDX EDI ESI EBP ESP R8D ... R15D

**64 bits:** RAX RBX RCX RDX RSI RDI RBP RSP R8 ... R15

**128 bits:** XMM0, ... , XMM15

# Instrucciones y Registros

## Operaciones

ADD, SUB, MOV, SHL, JMP ... (ver manual)

## Registros

**8 bits:** AL BL CL DL DIL SIL BPL SPL R8B ... R15B

**16 bits:** AX BX CX DX DI SI BP SP R8W ... R15W

**32 bits:** EAX EBX ECX EDX EDI ESI EBP ESP R8D ... R15D

**64 bits:** RAX RBX RCX RDX RSI RDI RBP RSP R8 ... R15

**128 bits:** XMM0, ... , XMM15

## Direccionamiento

$$\left[ \frac{\text{Base}}{\text{RAX}} + \left( \frac{\text{Index}}{\text{RAX}} * \frac{\text{Scale}}{1} \right) + \frac{\text{Displacement}}{\text{Cte. 32 bits}} \right]$$

...	...	2
R15	R15	4
	(NO RSP)	8

# Manuales

- Intel 64 and IA-32 Architectures  
Software Developer's Manual  
**Volume 1: Basic Architecture**
- Intel 64 and IA-32 Architectures  
Software Developer's Manual  
**Volume 2 (2A, 2B & 2C): Instruction Set Reference, A-Z**
- Intel 64 and IA-32 Architectures  
Software Developer's Manual  
**Volume 3 (3A, 3B & 3C): System Programming Guide**

## Instrucciones - Generalidades

- Las instrucciones en general toman **uno o dos parámetros**



## Instrucciones - Generalidades

- Las instrucciones en general toman **uno o dos parámetros**
- Instrucciones de un parámetro, ejemplos:
  - `inc [RAX]` ; incrementa en 1 el valor en la posición de memoria apuntada por el registro RAX
  - `dec CX` ; decrementa en 1 el valor de registro CX

## Instrucciones - Generalidades

- Las instrucciones en general toman **uno o dos parámetros**
- Instrucciones de un parámetro, ejemplos:
  - `inc [RAX]` ; incrementa en 1 el valor en la posición de memoria apuntada por el registro RAX
  - `dec CX` ; decrementa en 1 el valor de registro CX
- Instrucciones de dos parámetros, ejemplos:
  - `mov [RDX], RDX` ; mueve el valor en el registro RDX a la memoria apuntada por RDX
  - `add EAX, 10` ; suma 10 al registro EAX

## Instrucciones - Generalidades

- Las instrucciones en general toman **uno o dos parámetros**
- Instrucciones de un parámetro, ejemplos:
  - `inc [RAX]` ; incrementa en 1 el valor en la posición de  
                  ; memoria apuntada por el registro RAX
  - `dec CX`       ; decrementa en 1 el valor de registro CX
- Instrucciones de dos parámetros, ejemplos:
  - `mov [RDX], RDX` ; mueve el valor en el registro RDX  
                      ; a la memoria apuntada por RDX
  - `add EAX, 10`     ; suma 10 al registro EAX
- **Solo uno de los parámetros puede ser una referencia a memoria.**

## Instrucciones - Ejemplo Manual

# Instrucciones - Ejemplo Manual

## ADD—Add

Opcode	Instruction	Op/ En	64-bit Mode	Compat/ Leg Mode	Description
04 <i>ib</i>	ADD AL, <i>imm8</i>	I	Valid	Valid	Add <i>imm8</i> to AL.
05 <i>iw</i>	ADD AX, <i>imm16</i>	I	Valid	Valid	Add <i>imm16</i> to AX.
05 <i>id</i>	ADD EAX, <i>imm32</i>	I	Valid	Valid	Add <i>imm32</i> to EAX.
REX.W + 05 <i>id</i>	ADD RAX, <i>imm32</i>	I	Valid	N.E.	Add <i>imm32</i> sign-extended to 64-bits to RAX.
80 /0 <i>ib</i>	ADD r/m8, <i>imm8</i>	MI	Valid	Valid	Add <i>imm8</i> to r/m8.
REX + 80 /0 <i>ib</i>	ADD r/m8*, <i>imm8</i>	MI	Valid	N.E.	Add sign-extended <i>imm8</i> to r/m64.
81 /0 <i>iw</i>	ADD r/m16, <i>imm16</i>	MI	Valid	Valid	Add <i>imm16</i> to r/m16.
81 /0 <i>id</i>	ADD r/m32, <i>imm32</i>	MI	Valid	Valid	Add <i>imm32</i> to r/m32.
REX.W + 81 /0 <i>id</i>	ADD r/m64, <i>imm32</i>	MI	Valid	N.E.	Add <i>imm32</i> sign-extended to 64-bits to r/m64.
83 /0 <i>ib</i>	ADD r/m16, <i>imm8</i>	MI	Valid	Valid	Add sign-extended <i>imm8</i> to r/m16.
83 /0 <i>ib</i>	ADD r/m32, <i>imm8</i>	MI	Valid	Valid	Add sign-extended <i>imm8</i> to r/m32.
REX.W + 83 /0 <i>ib</i>	ADD r/m64, <i>imm8</i>	MI	Valid	N.E.	Add sign-extended <i>imm8</i> to r/m64.
00 /r	ADD r/m8, r8	MR	Valid	Valid	Add r8 to r/m8.
REX + 00 /r	ADD r/m8*, r8*	MR	Valid	N.E.	Add r8 to r/m8.
01 /r	ADD r/m16, r16	MR	Valid	Valid	Add r16 to r/m16.
01 /r	ADD r/m32, r32	MR	Valid	Valid	Add r32 to r/m32.
REX.W + 01 /r	ADD r/m64, r64	MR	Valid	N.E.	Add r64 to r/m64.
02 /r	ADD r8, r/m8	RM	Valid	Valid	Add r/m8 to r8.
REX + 02 /r	ADD r8*, r/m8*	RM	Valid	N.E.	Add r/m8 to r8.
03 /r	ADD r16, r/m16	RM	Valid	Valid	Add r/m16 to r16.
03 /r	ADD r32, r/m32	RM	Valid	Valid	Add r/m32 to r32.
REX.W + 03 /r	ADD r64, r/m64	RM	Valid	N.E.	Add r/m64 to r64.

### NOTES:

\*In 64-bit mode, r/m8 can not be encoded to access the following byte registers if a REX prefix is used: AH, BH, CH, DH.

# Instrucciones - Ejemplo Manual

## Description

Adds the destination operand (first operand) and the source operand (second operand) and then stores the result in the destination operand. The destination operand can be a register or a memory location; the source operand can be an immediate, a register, or a memory location. (However, two memory operands cannot be used in one instruction.) When an immediate value is used as an operand, it is sign-extended to the length of the destination operand format.

The ADD instruction performs integer addition. It evaluates the result for both signed and unsigned integer operands and sets the OF and CF flags to indicate a carry (overflow) in the signed or unsigned result, respectively. The SF flag indicates the sign of the signed result.

. . .

# Instrucciones - Ejemplo Manual

## Description

Adds the destination operand (first operand) and the source operand (second operand) and then stores the result in the destination operand. The destination operand can be a register or a memory location; the source operand can be an immediate, a register, or a memory location. (However, two memory operands cannot be used in one instruction.) When an immediate value is used as an operand, it is sign-extended to the length of the destination operand format.

The ADD instruction performs integer addition. It evaluates the result for both signed and unsigned integer operands and sets the OF and CF flags to indicate a carry (overflow) in the signed or unsigned result, respectively. The SF flag indicates the sign of the signed result.

. . .

## Operation

DEST  $\leftarrow$  DEST + SRC;

## Flags Affected

The OF, SF, ZF, AF, CF, and PF flags are set according to the result.

## Protected Mode Exceptions

#GP(0)	If the destination is located in a non-writable segment. If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit. If the DS, ES, FS, or GS register is used to access memory and it contains a NULL segment selector.
#SS(0)	If a memory operand effective address is outside the SS segment limit.
#PF(fault-code)	If a page fault occurs.
#AC(0)	If alignment checking is enabled and an unaligned memory reference is made while the current privilege level is 3.
#UD	If the LOCK prefix is used but the destination is not a memory operand.

## Bibliografía: Fuentes y material adicional

- Convenciones de llamados a función en x86:  
[https://en.wikipedia.org/wiki/X86\\_calling\\_conventions](https://en.wikipedia.org/wiki/X86_calling_conventions)
- Notas sobre System V ABI:  
[https://wiki.osdev.org/System\\_V\\_ABI](https://wiki.osdev.org/System_V_ABI)
- Documentación de NASM:  
<https://nasm.us/doc/>
- Artículo sobre el flag -pie:  
<https://eklitzke.org/position-independent-executables>
- Documentación de System V ABI:  
[https://uclibc.org/docs/psABI-x86\\_64.pdf](https://uclibc.org/docs/psABI-x86_64.pdf)
- Manuales de Intel:  
<https://software.intel.com/en-us/articles/intel-sdm>



# ¡Gracias!

Recuerden leer los comentarios al final de este video por aclaraciones o fe de erratas.