

# Instrucciones Aritméticas

**Instrucciones aritméticas.** Realizan operaciones aritméticas sobre números binarios puros o en Complemento a 2. Son instrucciones de este grupo add, cmp, adc, sbc, mul.

## add

Suma sin afectar al carry

add{s}{cond} {rd}, rn, Oper2

add{cond} {rd}, rn, #imm8

La instrucción add suma en rd los valores de rn y Operand2 o imm8. En ciertas circunstancias, el ensamblador puede sustituir una instrucción por otra. Tenelo en cuenta si tienes que leer un listado desensamblado.

donde:

- **s** es opcional, si se especifica, se actualiza el registro de flags CPSR, de acuerdo al resultado.
- **cond** opcional, es un código de condición.
- **rd** Registro destino.
- **rn** es el registro que contiene el primer operando.
- **Oper2** segundo operando.
- **imm8** Es cualquier valor que va de 0-4095.

## Códigos de Condicion

Código	Sufijo	Flags	Significado
0000	eq	Z set	Igual
0001	ne	Z clear	No Igual
0011	cs	C set	Mayor o igual sin signo
0011	cc	C clear	Menor sin signo
0100	mi	N set	Negativo
0101	pl	N clear	Positivo o cero
0110	vs	V set	Overflow
0111	vc	V clear	No overflow
1000	hi	C set and Z clear	mayor sin signo
1001	ls	C clear and Z set	menor o igual sin signo
1010	ge	N equals V	Mayor o igual
1011	lt	N not equal to V	Menor que
1100	gt	Z clear AND (N equal to V)	Mayor que
1101	le	Z set OR (N not equal to V)	menor o igual que
1110	al	(ignored)	always, siempre

Ejemplo:

```

mov r1, #5      @ R1 <-- 5
mov r2, #10     @ R2 <-- 10
add r0, r1, r2  @ R0 <-- 5 + 10 sin afectar el registro CPSR
adds r3, r1, r2 @ R3 <-- 5 + 10 afectando el registro CPSR

```

La forma de ver como se modifica el registro CPSR, es cuando debugueamos con gdb

```
i r cpsr
```

### Flags que modifica:

Si se especifica **s**, estas instrucciones actualizan los indicadores **N, Z, C y V** de acuerdo con el resultado.

### adc

Suma con carry

```
adc{s}{cond} {rd}, rn, Oper2
```

La instrucción adc (suma con acarreo) suma los valores en rn y Oper2, junto con el flag de acarreo.

donde:

- **s** es opcional, si se especifica, se actualiza el registro de flags CPSR, de acuerdo al resultado.
- **cond** opcional, es un código de condición.
- **rd** Registro destino.
- **rn** es el registro que contiene el primer operando.
- **Oper2** segundo operando.

Ejemplo:

```

mov r1, #5      @ R1 <-- 5
mov r2, #10     @ R2 <-- 10
adc r0, r1, r2  @ R0 <-- 5 + 10 + carry

```

La forma de ver como se modifica el registro CPSR, es cuando debugueamos con gdb

```
i r cpsr
```

### Flags que modifica:

Si se especifica **s**, estas instrucciones actualizan los indicadores **N, Z, C y V** de acuerdo con el resultado.

### sub

Resta sin afectar al carry

`sub{s}{cond} {rd}, rn, Oper2`

`sub{cond} {rd}, rn, #imm8`

La instrucción `sub` resta en `rd = rn - Oper2`. En ciertas circunstancias, el ensamblador puede sustituir una instrucción por otra. Tenelo en cuenta si tienes que leer un listado desensamblado.

donde:

- **s** es opcional, si se especifica, se actualiza el registro de flags CPSR, de acuerdo al resultado.
- **cond** opcional, es un código de condición.
- **rd** Registro destino.
- **rn** es el registro que contiene el primer operando.
- **Oper2** segundo operando.
- **imm8** Es cualquier valor que va de 0-4095.

Ejemplo:

```
mov r1, #15      @ R1 <-- 5
mov r2, #10      @ R2 <-- 10
sub r0, r1, r2   @ R0 <-- 15 - 10 sin afectar el registro CPSR
subs r3, r1, r2  @ R3 <-- 15 - 10 sin afectar el registro CPSR
```

La forma de ver como se modifica el registro CPSR, es cuando debuguemos con `gdb`

```
i r cpsr
```

### Flags que modifica:

Si se especifica **s**, estas instrucciones actualizan los indicadores **N**, **Z**, **C** y **V** de acuerdo con el resultado.

### sbc

Restar con carry

`sbc{s}{cond} {rd}, rn, Oper2`

La instrucción `sbc` (Restar con acarreo) resta el valor de `Oper2` del valor en `rn`. Si el flag carry es cero es clara, el resultado se le resta uno.

donde:

- **s** es opcional, si se especifica, se actualiza el registro de flags CPSR, de acuerdo al resultado.
- **cond** opcional, es un código de condición.
- **rd** Registro destino.
- **rn** es el registro que contiene el primer operando.
- **Oper2** segundo operando.

Ejemplo:

```

mov r1, #15      @ R1 <-- 5
mov r2, #10      @ R2 <-- 10
sbc r3, r1, r2   @ R3 <-- 15 - 10 -1, si carry=0

```

La forma de ver como se modifica el registro CPSR, es cuando debugueamos con gdb

```
i r cpsr
```

### Flags que modifica:

Si se especifica **s**, estas instrucciones actualizan los indicadores **N, Z, C y V** de acuerdo con el resultado.

### cmp y cmn

Estas instrucciones comparan el valor en un registro con Operando2. Actualizan los **indicadores** (bits) del registro de flags: **CPSR** en base al resultado, el resultado no afecta a ningún registro.

cmp{cond} rn, Operando2

cmn{cond} rn, Operando2

donde:

- **cond** es un código de condición opcional
- **rn** registro conteniendo el primer operando
- **Oper2** segundo operando

La instrucción **cmp resta** el valor de Operando2 del valor en rn. Esto es lo mismo que una instrucción subs, excepto que el resultado se descarta.

La instrucción **cmn Compare Negative** agrega el valor de Operando2 al valor en rn. Esto es lo mismo que una instrucción adds, excepto que el resultado se descarta.

### Flags que modifica:

Estas instrucciones actualizan los flags N, Z, C y V de acuerdo con el resultado.

Ejemplo:

```

mov r1, #5      @ R1 <-- 5
mov r2, #10     @ R2 <-- 10
cmp r1, r2      @ Compara los valores de los registros seteando flags.

```

(196)

From:

<http://wiki.educabit.ar/> - **Wiki Sistemas**

Permanent link:

[http://wiki.educabit.ar/doku.php?id=arm\\_addsubcmp](http://wiki.educabit.ar/doku.php?id=arm_addsubcmp)

Last update: **2025/09/11 22:48**

