

07 Instrucciones condicionales



Arquitectura de Computadoras y Ensambladores 1
M.Sc. Luis Fernando Espino Barrios
2024

Complemento a 1

- Se obtiene invirtiendo los bits de la representación binaria de un número.

2 Bits	Valor sin signo	Valor con signo (comp a 1)
00	0	0
01	1	1
10	2	-1
11	3	-0

3 Bits	Valor sin signo	Valor con signo (comp a 1)
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-3
101	5	-2
110	6	-1
111	7	-0

Complemento a 2

- Se calcula el complemento a 1 y luego se suma 1 en parte menos significativa

2 Bits	Valor sin signo	Valor con signo (comp a 2)
00	0	0
01	1	1
10	2	-2
11	3	-1

3 Bits	Valor sin signo	Valor con signo (comp a 2)
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-4
101	5	-3
110	6	-2
111	7	-1

Table 3.1: Flag bits NZCV in PSTATE.

Name		Logical instruction	Arithmetic instruction
N	(Negative)	No meaning	Bit 31 of the result is set. Indicates a negative number in signed operations.
Z	(Zero)	Result is all zeroes	Result of operation was zero
C	(Carry)	After Shift operation, '1' was left in carry flag	Result was greater than 64 bits
V	(oVerflow)	No meaning	The signed two's complement result requires more than 64 bits. Indicates a possible corruption of the result.

Name	Effect	Description
cmp	$Rn - \text{Operand2}$	Compare and set PSTATE flags.
cmn	$Rn + \text{Operand2}$	Compare negative and set PSTATE flags.
tst	$Rn \wedge \text{Operand2}$	Test bits and set PSTATE flags.
teq	$Rn \oplus \text{Operand2}$	Test equivalence and set PSTATE flags.

Table 3.2: AArch64 condition modifiers.

Condition code	Meaning	Condition flags	Binary encoding
EQ	Equal	$Z = 1$	0000
NE	Not Equal	$Z = 0$	0001
HI	Unsigned Higher	$(C = 1) \wedge (Z = 0)$	1000
HS	Unsigned Higher or Same	$C = 1$	0010
LS	Unsigned Lower or Same	$(C = 0) \vee (Z = 1)$	1001
LO	Unsigned Lower	$C = 0$	0011
GT	Signed Greater Than	$(Z = 0) \wedge (N = V)$	1100
GE	Signed Greater Than or Equal	$N = V$	1010
LE	Signed Less Than or Equal	$(Z = 1) \vee (N \neq V)$	1101
LT	Signed Less Than	$N \neq V$	1011
CS	Unsigned Overflow (Carry Set)	$C = 1$	0010
CC	No Unsigned Overflow (Carry Clear)	$C = 0$	0011
VS	Signed Overflow	$V = 1$	0110
VC	No Signed Overflow	$V = 0$	0111
MI	Minus, Negative	$N = 1$	0100
PL	Plus, Positive or Zero	$N = 0$	0101
AL	Always Executed	<i>Any</i>	1110
NV	Never Executed	<i>Any</i>	1111

Instrucciones condicionales de 2 operandos

Instrucción	Efecto	Descripción
<code>csel Rd, Rn, Rm, <cond></code>	if <cond> Rd=Rn else Rd=Rm	Selección condicional
<code>csinc Rd, Rn, Rm, <cond></code>	if <cond> Rd=Rn else Rd=Rm+1	Incremento de selección condicional
<code>csinv Rd, Rn, Rm, <cond></code>	if <cond> Rd=Rn else Rd= \neg Rm	Inversión de selección condicional
<code>csneg Rd, Rn, Rm, <cond></code>	if <cond> Rd=Rn else Rd=1+ \neg Rm	Negación de selección condicional

```
.global _start

.data
out: .ascii " - - - \n"

.text
_start:
    ldr x1, =out           // load output

    mov w3, 2              // first operand
    mov w4, 3              // second operand

    mov w5, 1              // true
    mov x6, 0              // false

    cmp w3, w4

    csel w7, w5, w6, eq     // if cmp.eq 1 else 0
    add w7, w7, 48
    strb w7, [x1]          // store 0
```

```
csel w7, w5, w6, ne      // if cmp.ne 1 else 0
add w7, w7, 48
strb w7, [x1, 2]         // store 1
```

```
csel w7, w5, w6, gt      // if cmp.gt 1 else 0
add w7, w7, 48
strb w7, [x1, 4]         // store 0
```

```
csel w7, w5, w6, lt      // if cmp.lt 1 else 0
add w7, w7, 48
strb w7, [x1, 6]         // store 1
```

```
mov x0, 1
mov x2, 8
mov x8, 64
svc 0
```

```
mov x0, 0
mov x8, 93
svc 0
```



luisespino@raspberrypi: ~/nfs



```
luisespino@raspberrypi:~/nfs $ as 08_csel.s -o 08_csel.o
```

```
luisespino@raspberrypi:~/nfs $ ld 08_csel.o -o 08_csel
```

```
luisespino@raspberrypi:~/nfs $ ./08_csel
```

```
0-1-0-1
```

```
luisespino@raspberrypi:~/nfs $
```

Instrucciones condicionales de 1 operando

Instrucción	Efecto	Descripción
cinc Rd, Rn, <cond>	if <cond> Rd=Rn+1 else Rd=Rn	Incremento condicional
cinv Rd, Rn, <cond>	if <cond> Rd= \neg Rn else Rd=Rn	Inversión condicional
cneg Rd, Rn, <cond>	if <cond> Rd=1+ \neg Rn else Rd=Rn	Negación condicional

Instrucciones condicionales tipo set

Instrucción	Efecto	Descripción
cset Rd, <cond>	if <cond> Rd=1 else Rd=0	Set condicional
csetm Rd, <cond>	if <cond> Rd=0xFF.. else Rd=0x00..	Set Mask condicional

```
.global _start

.data
out: .ascii " - - - \n"

.text
_start:
    ldr x1, =out                // load output

    mov w3, 'a'                 // first operand
    mov w4, 'a'                 // second operand

    cmp w3, w4

    cset w7, eq                 // if cmp.eq 1 else 0
    add w7, w7, 48
    strb w7, [x1]               // store 1
```



```
cset w7, ne           // if cmp.ne 1 else 0
add w7, w7, 48
strb w7, [x1, 2]      // store 0
```

```
cset w7, ge           // if cmp.gt 1 else 0
add w7, w7, 48
strb w7, [x1, 4]      // store 1
```

```
cset w7, le           // if cmp.lt 1 else 0
add w7, w7, 48
strb w7, [x1, 6]      // store 1
```

```
mov x0, 1
mov x2, 8
mov x8, 64
svc 0
```

```
mov x0, 0
mov x8, 93
svc 0
```



luisespino@raspberrypi: ~/nfs



```
luisespino@raspberrypi:~/nfs $ as 09_cset.s -o 09_cset.o
```

```
luisespino@raspberrypi:~/nfs $ ld 09_cset.o -o 09_cset
```

```
luisespino@raspberrypi:~/nfs $ ./09_cset
```

```
1-0-1-1
```

```
luisespino@raspberrypi:~/nfs $
```

Ejercicio

- Escriba un segmento de programa donde el registro x0 tenga un valor negativo, calcule mediante instrucciones condicionales el valor absoluto de x0 dejando el resultado en el mismo registro.

mov x0, -25

cmp x0, 0

cneg x0, x0, mi

Bibliografía

Arm Limited. (2024). Arm Architecture Reference Manual: for A-profile architecture.

Patterson. D. & Hennesy, J. (2017). Computer Organization and Design: ARM Edition. Elsevier.

Pyeatt, L. & Ughetta, W. (2020). ARM 64-bit Assembly Language. Elsevier.

Smith, S. (2020). Programming with 64-bit ARM Assembly Language. Apress.