

# Introduction à l'assembleur ARM: arithmétique et conditions



# Instructions arithmétiques et logiques

- Les opérations mathématiques et logiques ont la forme

```
INSTRUCTION Rd, Rs, Op1
```

- Où
  - Rd est le registre de destination
  - Rs est un registre source
  - Op1 est une opérande de type 1
- Le format de l'instruction ADD, par exemple, est:

```
ADD Rd, Rs, Op1 ; Rd ← Rs + Op1
```

```
ADD R0, R0, #1 ; R0 ← R0 + 1  
ADD R0, R0, R1 ; R0 ← R0 + R1  
ADD R0, R0, R1, LSL #1 ; R0 ← R0 + (R1 * 2)
```

# Exemples

- Soustraction

```
SUB R0, R0, #1      ; R0 ← R0 - 1  
SUB R0, R0, R1     ; R0 ← R0 - R1
```

- Décalage

```
LSL R0, R0, #1      ; R0 ← R0 * 2  
ASR R0, R0, #2     ; R0 ← R0 / 4 (préserve le signe)
```

- “Et” logique

```
AND R0, R0, #1      ; R0 ← R0 ET 1  
AND R0, R0, R1     ; R0 ← R0 ET R1
```

- Prendre le négatif

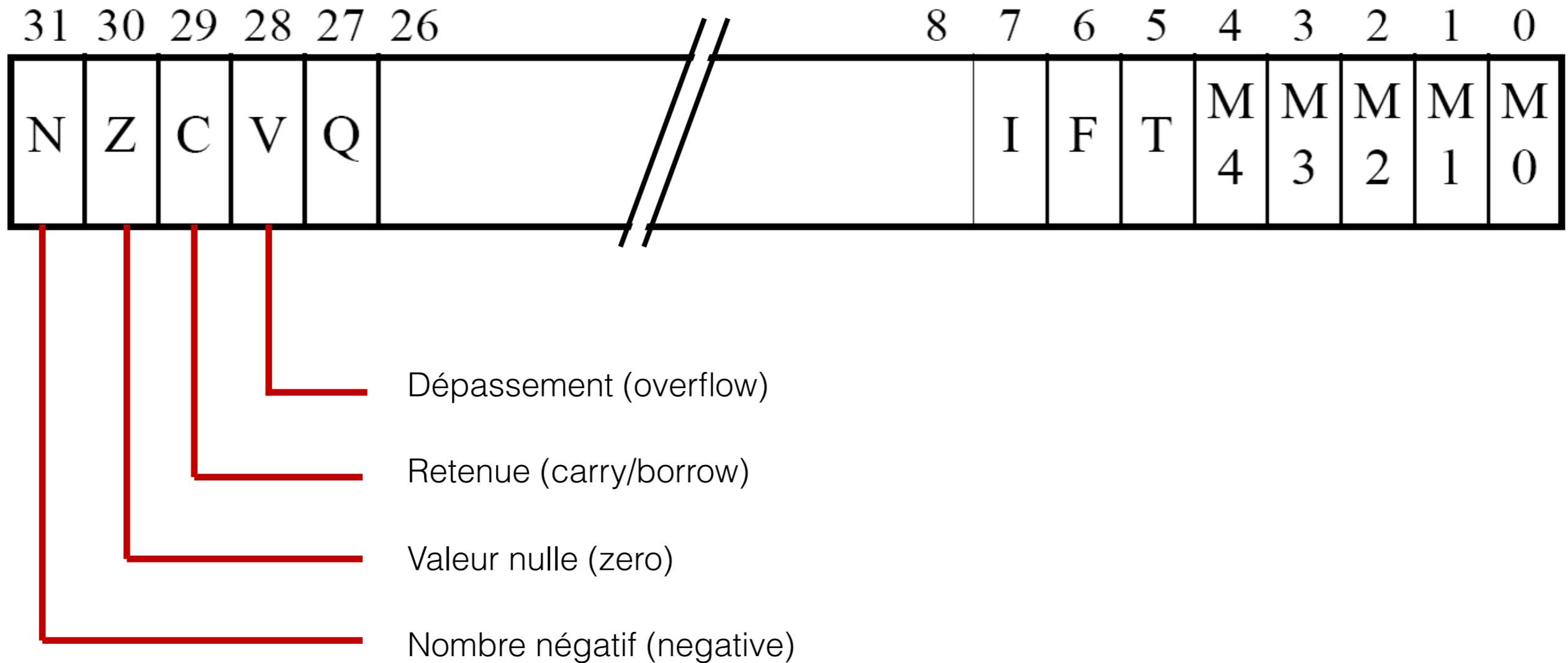
```
RSB R0, R0, #0      ; R0 ← 0 - R0, donc R0 = -R0
```

# Problème à résoudre

- But: comparer deux nombres placés dans R1 et R2
  - Si  $R1 > R2$ , mettre R3 dans R0
  - Sinon, mettre R4 dans R0
- Comment faire?
- Nous allons avoir besoin de trois mécanismes:
  1. *Une instruction pour comparer R1 et R2*
  2. *Un endroit pour stocker le résultat de la comparaison*
  3. *Des instructions conditionnelles, activées seulement si la comparaison répond à certains critères*

# 2. Endroit pour stocker

- Un registre de statut décrit l'état du processeur



# CPSR: détection de conditions

- N: Détection de signe négatif
  - 1 si résultat  $< 0$ , 0 autrement
- Z: Détection de zéro
  - 1 si résultat = 0, 0 autrement
  - Souvent utilisé pour détecter les égalités
- C: Détection de retenue («carry») ou d'emprunt («borrow»)
  - 1 si l'opération a impliqué une retenue, 0 autrement
  - Ex. retenue d'addition de nombres positifs
- V: Détection de débordements (overflow)
  - 1 si l'opération a impliqué un débordement, 0 autrement
  - Ex. débordement signé lors d'une addition

# Retenue («carry») avec nombres *non-signés*

- Nombres *non-signés* sur 4 bits
- $10 + 8 = ?$
- $0b1010 + 0b1000 = 0b1\ 0010$ 
  - Nous avons besoin d'un 5e bit pour que le résultat soit valide, c'est le bit «carry»!

# Retenue («carry») avec nombres *signés*

- Nombres *signés* (complément-2) sur 4 bits
- $-2 + 2 = ?$ 
  - $0b1110 + 0b0010 = 0b1\ 0000$ 
    - Le résultat est valide sur 4 bits, mais un 5e bit est activé: il y a «carry»!
- Est-ce qu'il y a débordement?
  - Non car les deux bits de signes étaient différents initialement.



# 1. Instruction pour comparer

- L'instruction CMP compare deux nombres, et met à jour les drapeaux de l'ALU (le CPSR)

```
CMP R1, R2 ; calcule R1 - R2, change les drapeaux
```

# 1. Instruction(s) pour comparer

- Les instructions arithmétiques et logiques changent les drapeaux de l'ALU, lorsque l'option "S" est rajoutée après le nom de l'instruction

```
INSTRUCTIONS Rd, Rs, Op1 ; exécute l'instruction,  
                        ; et met à jour les drapeaux
```

- Exemple:

```
SUBS R0, R1, R2 ; R0 ← R1 - R2  
                ; et met à jour les drapeaux
```

Quels seront les drapeaux N et Z du CPSR?

# Démonstration (comparaisons)

# 3. Instructions conditionnelles

- L'instruction

```
MOVcc Rd, Op1
```

met l'opérande de type 1 *Op1* dans le registre *Rd*,  
*si la condition cc est vraie*

- Exemple:

```
MOVEQ R3, R1      ; R3 ← R1 seulement si le drapeau Z est 1  
ADDNE R2, R2, R1  ; R2 ← R2 + R1 seulement si le drapeau Z est 0
```

# Instructions conditionnelles

Code assembleur:

```
MOVEQ R3, R1 ; R3 ← R1 seulement si le drapeau Z est 1
```

Équivalent, en C, à:

```
if (Z == 1) {  
    R3 = R1;  
}
```

Code assembleur:

```
ADDNE R2, R2, R1 ; R2 ← R2 + R1 seulement si le drapeau Z est 0
```

Équivalent, en C, à:

```
if (Z == 0) {  
    R2 = R2 + R1;  
}
```

# Codes de condition (CC)

- Plusieurs instructions s'exécutent si une condition est rencontrée.
- Toutes les conditions sont évaluées à partir des drapeaux de L'ALU et assument que ceux-ci ont été déterminés auparavant.
  - Par exemple, la condition EQ (equal) assume qu'une soustraction ou comparaison a été faite avant: si le résultat de l'opération est 0, le drapeau Z sera à 1 et la condition EQ sera rencontrée.
- Les drapeaux N (Negative), Z (Zero), C (Carry) et V (Overflow) servent à évaluer toutes les conditions.
- Les drapeaux et les conditions à évaluer changent si les nombres comparés sont signés ou s'ils ne le sont pas.

# Codes de condition

Table A8-1 Condition codes

cond	Mnemonic extension	Meaning (integer)	Meaning (floating-point) <sup>a</sup>	Condition flags
0000	EQ	Equal	Equal	Z == 1
0001	NE	Not equal	Not equal, or unordered	Z == 0
0010	CS <sup>b</sup>	Carry set	Greater than, equal, or unordered	C == 1
0011	CC <sup>c</sup>	Carry clear	Less than	C == 0
0100	MI	Minus, negative	Less than	N == 1
0101	PL	Plus, positive or zero	Greater than, equal, or unordered	N == 0
0110	VS	Overflow	Unordered	V == 1
0111	VC	No overflow	Not unordered	V == 0
1000	HI	Unsigned higher	Greater than, or unordered	C == 1 and Z == 0
1001	LS	Unsigned lower or same	Less than or equal	C == 0 or Z == 1
1010	GE	Signed greater than or equal	Greater than or equal	N == V
1011	LT	Signed less than	Less than, or unordered	N != V
1100	GT	Signed greater than	Greater than	Z == 0 and N == V
1101	LE	Signed less than or equal	Less than, equal, or unordered	Z == 1 or N != V
1110	None (AL) <sup>d</sup>	Always (unconditional)	Always (unconditional)	Any

a. Unordered means at least one NaN operand.

b. HS (unsigned higher or same) is a synonym for CS.

c. LO (unsigned lower) is a synonym for CC.

d. AL is an optional mnemonic extension for always, except in IT instructions. For details see *IT* on page A8-104.

# Dans le simulateur...

The image shows a debugger simulator interface with several panels:

- Simulation**: A top bar with buttons for "Arrêter", "Réinitialiser", and navigation arrows.
- Registre Généraux (User)**: A table of registers R0-R15 with their current values in hexadecimal.
- Code**: A list of assembly instructions with line numbers 1-18. The current instruction is highlighted in red.
- État courant**: A section for CPSR and SPSR flags, with a yellow box highlighting the CPSR flags.
- Mémoire**: A memory dump showing addresses and their corresponding hex values.
- Instructions pour l'activation des breakpoints**: A list of keyboard shortcuts for setting breakpoints.

**Registre Généraux (User)**

Nom	Valeur
R0	00000001
R1	fffffff
R2	00000000
R3	00000000
R4	00000000
R5	00000000
R6	00000000
R7	00000000
R8	00000000
R9	00000000
R10 (s1)	00000000
R11 (fp)	00000000
R12 (ip)	00000000
R13 (sp)	00000000
R14 (lr)	00000000
R15 (pc)	00000094

**Code**

```
1 SECTION INTVEC
2
3 B main
4
5
6 SECTION CODE
7
8 main
9
10 MOV R0, #1
11 MOV R1, #-1
12
13 CMP R0, R1
14 B main
15
16
17 SECTION DATA
18
```

**État courant**

	CPSR	SPSR
Negatif (N)	Faux	
Zero (Z)	Faux	
Emprunt (C)	Vrai	
Dépassement (V)	Vrai	
Ignore IRQ	Faux	
Ignore FIQ	Faux	

**Mémoire**

addr	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0x00000000	1E	00	00	EA	--	--	--	--	--	--	--	--	--	--	--	--
0x00000010	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000020	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000030	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000040	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000050	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000060	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000070	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000080	01	00	A0	E3	00	10	E0	E3	01	00	50	E1	FB	FF	FF	EA
0x00000090	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x000000a0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x000000b0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x000000c0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x000000d0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x000000e0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x000000f0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000100	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000110	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000120	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
0x00000130	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

**Instructions pour l'activation des breakpoints**

- Écriture (W) : Ctrl/Cmd + Clic
- Lecture (R) : Shift + Clic
- Lecture et Écriture (RW) : Ctrl/Cmd + Shift + Clic
- Exécution (E) : Alt + Clic

**État courant**

0x00000000 Go

Suivre PC

Cycle courant : 4

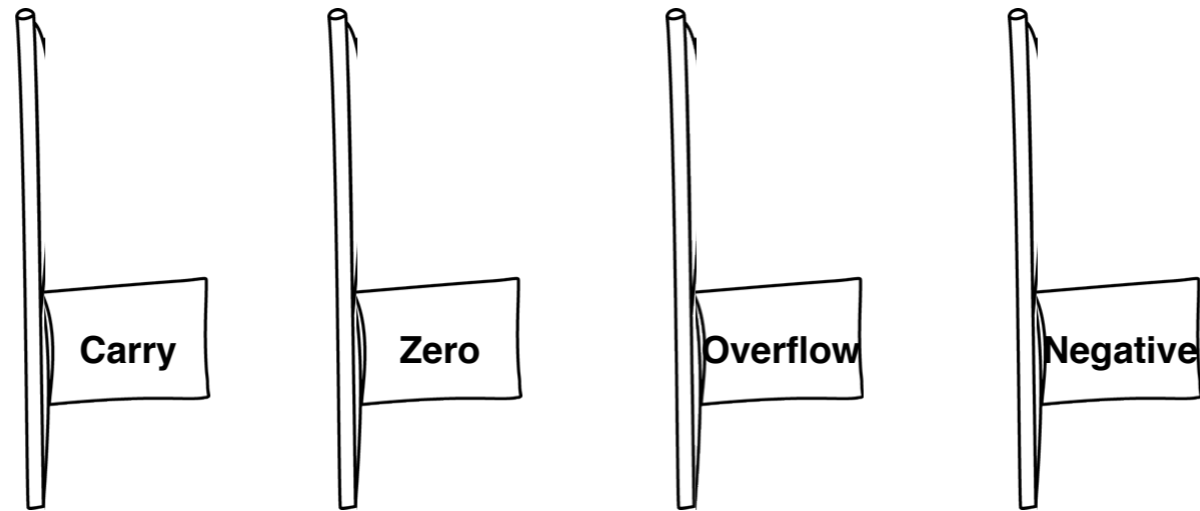
Charger un fichier Télécharger

Drapeaux dans le CPSR



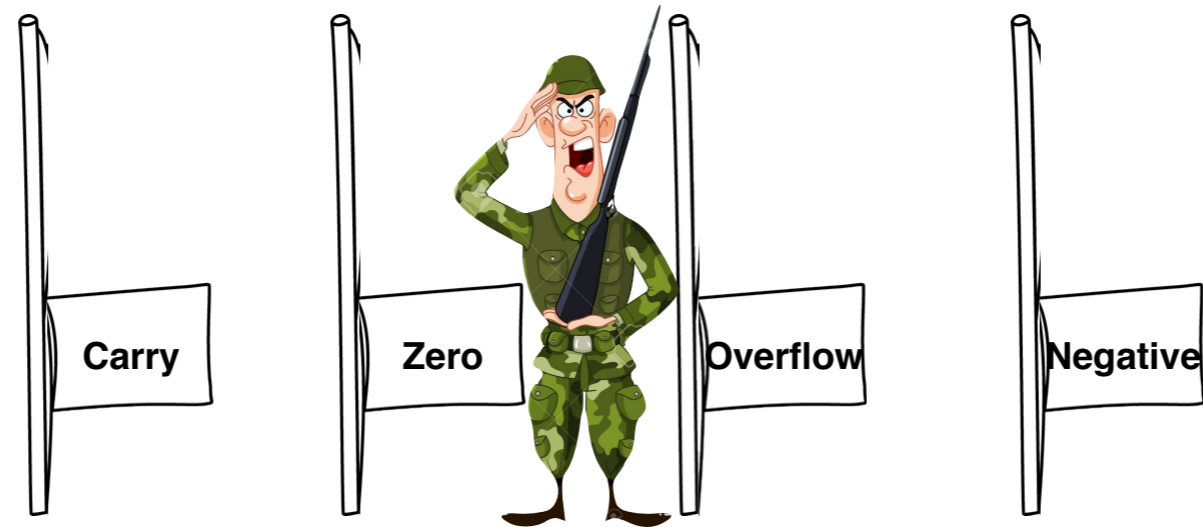
# Dans la série «analogies visuelles déconcertantes™»

```
SUBS  R0, R1, R2      ; R0 ← R1 - R2  
                        ; et met à jour les drapeaux
```



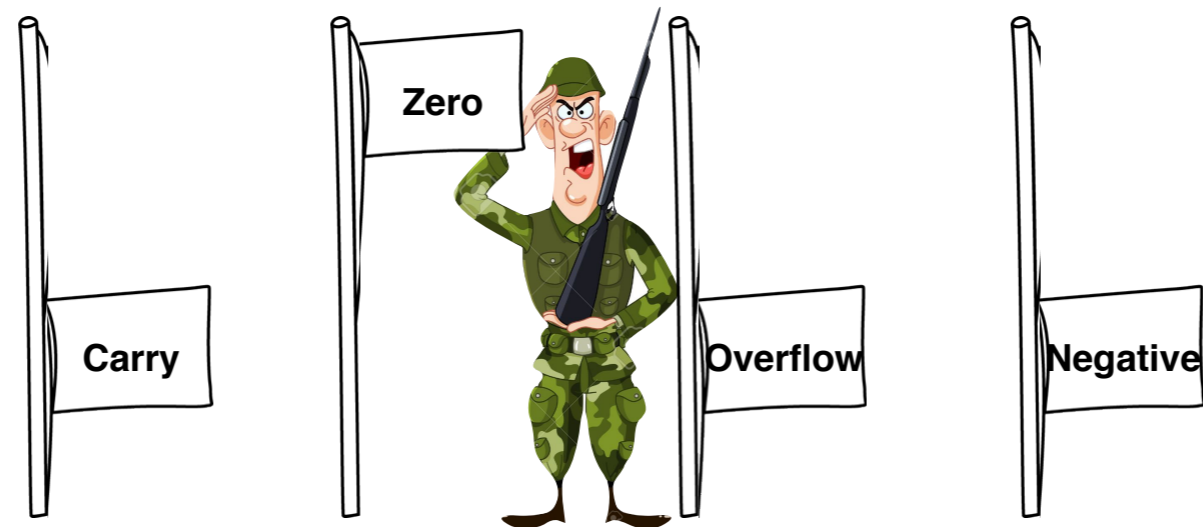
# Dans la série «analogies visuelles déconcertantes™»

```
SUBS  R0, R1, R2      ; R0 ← R1 - R2  
                        ; et met à jour les drapeaux
```



# Dans la série «analogies visuelles déconcertantes™»

```
SUBS R0, R1, R2      ; R0 ← R1 - R2  
                    ; et met à jour les drapeaux
```



# Dans la série «analogies visuelles déconcertantes™»

```
SUBS R0, R1, R2      ; R0 ← R1 - R2  
                    ; et met à jour les drapeaux
```



```
ADDEQ R2, R2, R1    ; R2 = R2 + R1 seulement si le drapeau Z est 1  
                    ; sinon, on passe à l'instruction suivante
```

# Problème à résoudre

- But: comparer deux nombres placés dans R1 et R2
  - Si  $R1 > R2$ , mettre R3 dans R0
  - Sinon ( $R1 \leq R2$ ), mettre R4 dans R0
- Comment faire?

Code	Symbole
GT	>
GE	>=
LT	<
LE	<=

```
CMP R1, R2 ; calcule R1 - R2, change les drapeaux
MOVGT R0, R3 ; si R1 > R2, R0 ← R3
MOVLE R0, R4 ; si R2 >= R1, R0 ← R4
```

# Démonstration (comparaisons #2)

# Problème à résoudre

- But:

- $R0 = \text{abs}(R1 - R2)$  ; valeur absolue

- Comment faire? Indices:

- $R0 = R1 - R2$  si  $R1 > R2$
- $R0 = R2 - R1$  sinon
- l'instruction RSB peut être utilisée pour calculer le négatif d'un registre

```
RSB R0, R0, #0 ; R0 ← -R0
```

- Solution (à 3 instructions):

```
CMP R1, R2 ; calcule R1 - R2, change les drapeaux  
SUBGT R0, R1, R2 ; si R1 > R2, R0 ← R1 - R2  
SUBLE R0, R2, R1 ; si R1 ≤ R2, R0 ← R2 - R1
```

- Solution (à 2 instructions):

```
SUBS R0, R1, R2 ; calcule R1 - R2, change les drapeaux  
RSBLE R0, R0, #0 ; si R1 ≤ R2, R0 ← -R0 (donc R0 ← R2 - R1)
```

Code	Symbole
GT	>
GE	>=
LT	<
LE	<=

# Démonstration (valeur absolue)