ARM

ARM prevede 16 registri da R0 A R15

- R0 a R 12 uso generale

-R13 stack pointer

-R14 Subroutine Link Register ripristina PC alla istruzione successiva
alla chiamata subroutine,quando ha terminato la sua esecuzione
-R15 Program Counter

-CPSR Curent Program Status register owero memorizza lo stato corrente del processore attraverso dei flag

cpsr	000001d3	NZCVI SVC

Caricamneto indirizzamento immediato nel registro:

MOV R0 ,#1

r0	00000001
r1	00000000
r2	00000000

N	Z	C	V	I
Negato	Zero	Carry	Overflow	Saturazione

Esercizi somma e moltiplicazione

r0	2
r1	4
r2	6
r3	6
r4	0
r5	3
r6	4
r7	0
r8	12
r9	0
r10	0
r11	0
r12	0
sp	0
lr	0
pc	0
cpsr	467 NZCVI SVC

```
1  .global _start
2  _start:
3
4  MOV R0,#2
5  MOV R1,#4
6  ADD R3,R1,R0
7
8  MOV R5,# 3
9  MOV R6,# 4
10 MUL R8,R5,R6
```

Caricamneto registro immediato

Somma R3 <- R1+R0

r0	2
r1	4
r2	0
r3	6
r4	0
r5	3
r6	0
r7	0
r8	18
r9	0
r10	0
r11	0
r12	0
sp	0
lr	0
pc	24
cpsr	467 NZCVI SVC
spsr	0 NZCVI ?

Compile and Load (F5) Language: ARMv7

```
1  .global _start
2  _start:
3
4  MOV R0,#2
5  MOV R1,#4
6  ADD R3,R1,R0
7
8  MOV R5,# 3
9  MUL R8,R5,R3
```

Caricamneto registro immediato

Somma R3 = R1+R0

Motiplico R8= R3*R5

Sottrazione

MOV R0 ,#7
MOV R1 , #4

SUB R3,R0,R1
// R3= R0-R1

IN ARM la sottrazione conta l'ordine ripsetto somma e moltiplicazione,quando un numero é negativo o molto alto non ci capisce.

fffffffe ----> -2

Per risolvere questo problema si usa un registro speciale CPSR

CPSR ha dei flag specifici che sono :
N Z C V I

Con l'istruzione SUBS rileva se il numeo é negativo

ESEMPIO

MOV R0 ,#0xffffffff ---> -1
MOV R1 , #1

SUB R3,R0,R1 -> fffffffe ->-2

Moltiplicazione

MOV R0 ,#7
MOV R1 , #4

MUL R3,R0,R1 // R3= R0*R1

Divisione

MOV R0 ,#7
MOV R1 , #4

MUL R3,R0,R1 // R3= R0/R1

Somma

MOV R0 ,#7
MOV R1 , #4

ADD R3,R0,R1 // R3= R0+R1

Quando si fanno le somme il registro potrebbe riportare un riporto perché il numero é troppo grande per essere memorizzato.

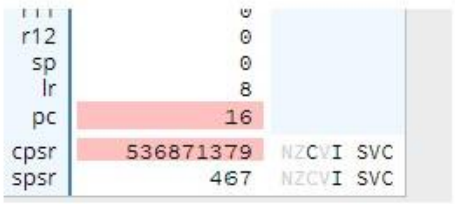
Con l'istruzione ADDS rileva il riporto

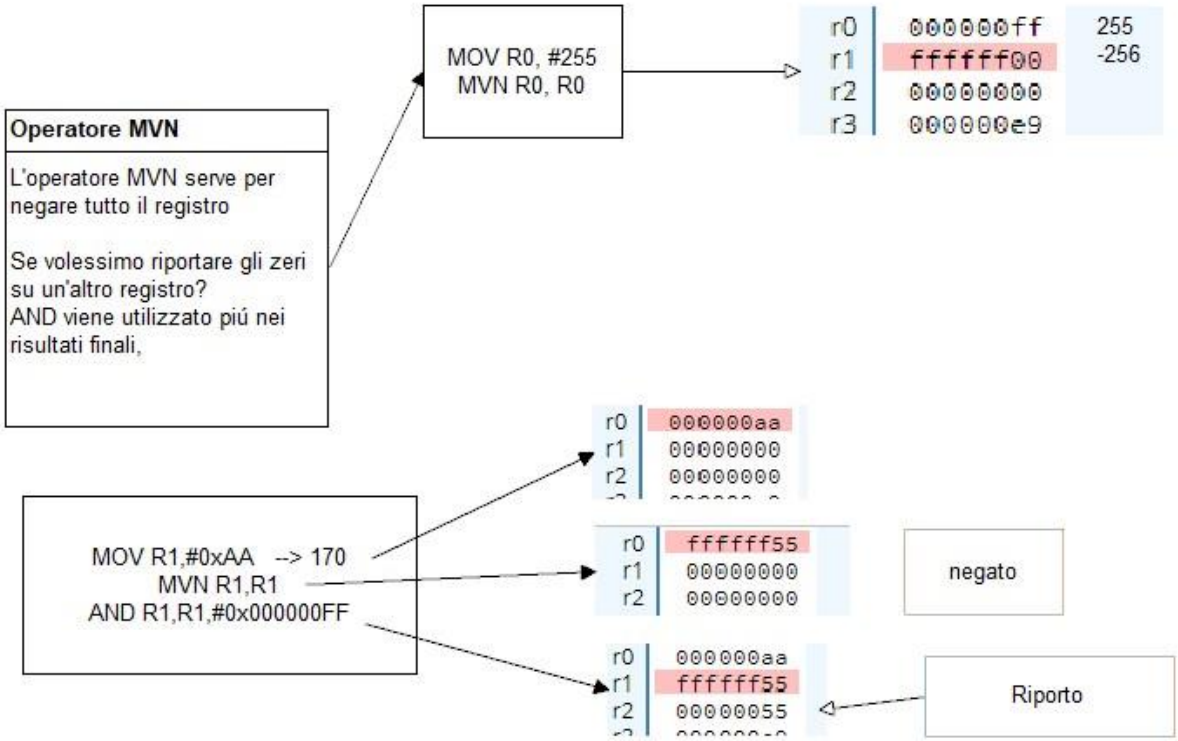
L'operazione ADC aggiunge alla query un risultato

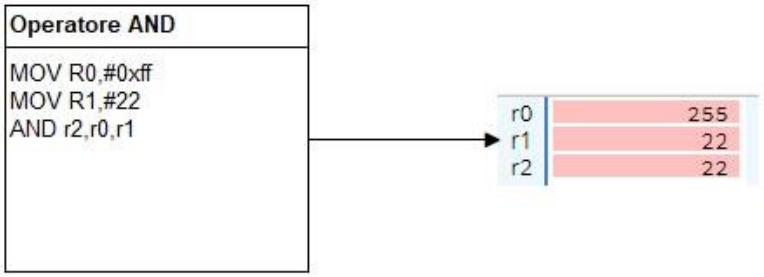
ESEMPIO

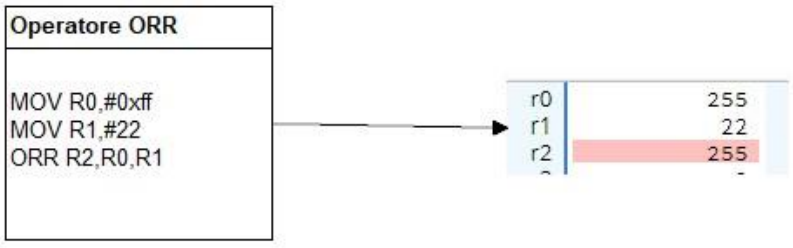
MOV R0 ,# -1
MOV R1 , # 3

ADDS R3,R0,R1 Somma il riporto
R3 = R0 + R1 + Carry(Riporto)




MOV R0, #255
MVN R0, R0
r0 000000ff 255
r1 ffffff00 -256
r2 00000000
r3 000000e9
Operatore MVN
L'operatore MVN serve per negare tutto il registro
Se volessimo riportare gli zeri su un'altro registro?
AND viene utilizzato piú nei risultati finali,
MOV R1,#0xAA --> 170
MVN R1,R1
AND R1,R1,#0x000000FF
r0 000000aa
r1 00000000
r2 00000000
r0 ffffff55
r1 00000000
r2 00000000
negato
r0 000000aa
r1 ffffff55
r2 00000055
Riporto


Operatore AND
MOV R0,#0xff
MOV R1,#22
AND r2,r0,r1
r0 255
r1 22
r2 22


Operatore ORR
MOV R0,#0xff
MOV R1,#22
ORR R2,R0,R1
r0 255
r1 22
r2 255

LSL e SLR

Gli spostamenti logici e rotazioni logiche molto utili permette di manipolare i numeri,consideriamo questi operazioni a livello bit

Ci sono due spostamento logico verso sinistra e destra:

LSL --> 1010 (10) 10100 (20)

Si sposta di sinistra di 1 posizione da dove si é inizati

0000 1010 --> 00010100

Notiamo che é esattamente il doppio, come se avessimo moltiplicato il valore x2 --> 10x2
NON É UNA COINCIDENZA!!!! modo piú rapido calcolare le moltiplicazioni

LSR --> 1010 (10) 0101 (5)

Si sposta di destra di 1 posizione da dove si é iniziati
0000 1010 --> 00000101

Notiamo che il valore é la metá del valore iniziale?Abbiamo scoperto che se ci si sposta di destra si divide /2 --> 10/2

ROR

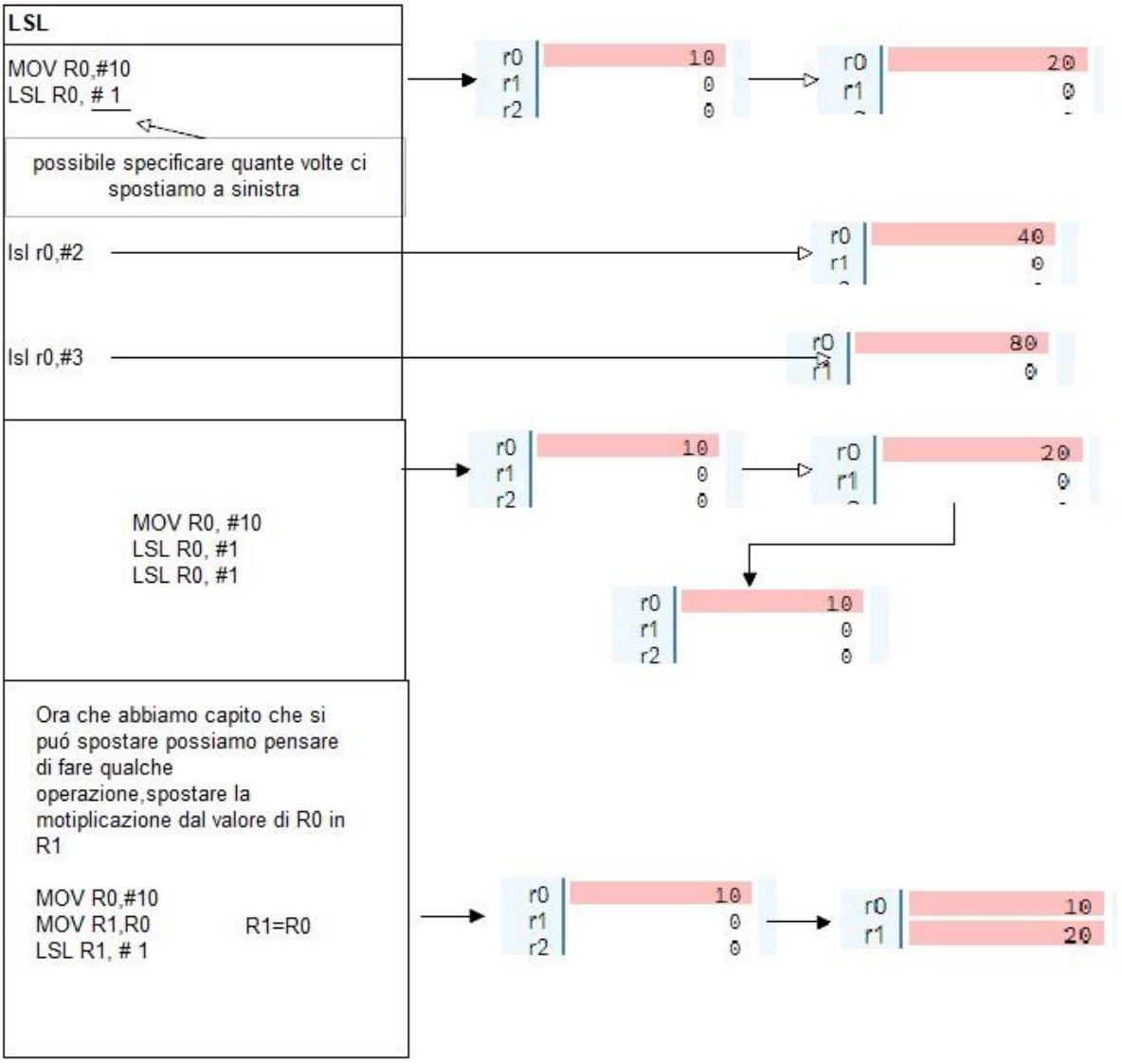
ROR 0000 0101 --> 1000 0010

Si é spostata di 1 poszione a destra ruotando

0000 0101 ---> 1000 0010

La rotazione viene utilizzata in genere per l'hash,crypto,grafica.

Esiste sola la rotazione a destra nel caso devi spostare a sinistra devi spostare di n bit


LSL
MOV R0,#10
LSL R0, # 1
possibile specificare quante volte ci spostiamo a sinistra
lsl r0,#2
lsl r0,#3
r0 10
r1 0
r2 0
r0 20
r1 0
r0 40
r1 0
r0 80
r1 0
MOV R0, #10
LSL R0, #1
LSL R0, #1
r0 10
r1 0
r2 0
r0 20
r1 0
r0 10
r1 0
r2 0
Ora che abbiamo capito che si puó spostare possiamo pensare di fare qualche operazione,spostare la motiplicazione dal valore di R0 in R1
MOV R0,#10
MOV R1,R0 R1=R0
LSL R1, # 1
r0 10
r1 0
r2 0
r0 10
r1 20