Architettura Degli Elaboratori Mod 2
Filippo Bergamasco

Instruction mnemonic	Operation
ADD Rd, Rn, Rm	Rd := Rn + Rm (non setta APSR)

ADD Rd, Rn, Rm, LSL #x
ADD Rd, Rn, Rm, LSR #x
ADD Rd, Rn, Rm, ASR #x
	Rd := Rn + (Rm << #x) // shift logico
	Rd := Rn + (Rm >> #x) // shift logico
	Rd := Rn + (Rm >> #x) // shift aritmetico
	Nota: #x puo' essere sostituito da un registro

ADD Rd, Rn, #x	Rd := Rn + #x //con #x immediato a 12 bit Nota: non e' possibile usare XZR o WZR al posto di Rd su operazioni con immediati.

Altre istruzioni aritmetico logiche:

SUB (sottrazione), AND (and logico), EOR (xor), ORR (or logico),
MUL (moltiplicazione), LSL (shift logico a sinistra),
LSR (shift logico a destra).
Tutte le operazioni logico/aritmetiche se usate col suffisso "s" settano i bit del registro APSR (status codes)

Instruction mnemonic	Operation
MOV Rd, Rn	Rd := Rn // pseudo istruzione
MOV Rd, #x	Rd := #x // con #x immediato a 12bit)
MOVZ Rd, #x LSL #k	Carica un immediato #x a 16 bit shiftato di #k = 0,16,32,48 bits a sinistra in Rd. Setta tutti i bit rimanenti a zero.
MOVK Rd, #x LSL #k	Carica un immediato #x a 16 bit shiftato di #k = 0,16,32,48 bits a sinistra in Rd. Lascia inalterati tutti gli altri bit
ADR Xd, <label>	Carica l'indirizzo di <label> nel registro Xd Nota: <label> entro +- 1Mb dal PC
SXTB Rd, Rn	Estende di segno il primo byte di Rn e salva il risultato in Rd
SXTH Rd, Rn	Estende di segno i primi 2 byte di Rn e salva il risultato in Rd
SXTW Rd, Rn	Estende di segno i primi 4 byte di Rn e salva il risultato in Rd
UXTB Rd, Rn	Estende con zeri il primo byte di Rn e salva il risultato in Rd
UXTH Rd, Rn	Estende con zeri i primi 2 byte di Rn e salva il risultato in Rd
UXTW Rd, Rn	Estende con zeri i primi 4 byte di Rn e salva il risultato in Rd

SVC #0	Supervisor call
B <label>	Salta a <label>
BL <label>	Branch and link <label>
RET	B LR
BR Xn	Salta all'indirizzo contenuto in Xn
BLR Xn	Branch and link all'indirizzo contenuto in Xn
CBZ Rm, <label>	Salta se Rm==0
CBNZ Rm, <label>	Salta se Rm!=0
CMP Rn, Rm	Compara Rn e Rm e setta gli status codes di APSR
B.<COND> <label>	Branch condizionato (vedi tab.)
CSEL Rd,Rn,Rm,<COND>	if <COND> then Rd=Rn else Rd=Rm
CSINC Rd,Rn,Rm,<COND>	if <COND> then Rd=Rn else Rd=Rm+1

NOTA:
Rd, Rn, Rm: Qualsiasi registro a 64bit (x0..x31) o 32bit (w0..w31)
(non mescolabili).
Xd: Qualsiasi registro a 64bit (x0..x31)

<COND>	Significato
EQ	Equal (Z==1)
NE	Not equal (Z==0)
HS	Unsigned higher or same (C==1)
LO	Unsigned lower (C==0)
MI	Minus (negative) (N==1)
PL	Plus (positive or zero) (N==0)
VS	Overflow set (V==1)
VC	Overflow clear (V==0)
HI	Unsigned higher (C==1 && Z==0)
LS	Unsigned lower or same !(C==1 && Z==0)
GE	Signed greater than or equal (N==V)
LT	Signed less than (N!=V)
GT	Signed greater than (Z==0 && N==V)
LE	Signed less than or equal !(Z==0 && N==V)
AL	Always

Load/Store

Instruction mnemonic	Operation
ldr xd, <addr>	carica 8 byte da <addr> a xd
ldr wd, <addr>	carica 4 byte da <addr> a wd
ldrb wd, <addr>	carica 1 byte da <addr> a wd (zero extend)
ldrsb wd, <addr>	carica 1 byte da <addr> a wd (sign extend)
ldrh wd, <addr>	carica 2 byte da <addr> a wd (zero extend)
ldrsh wd, <addr>	carica 2 byte da <addr> a wd (sign extend)
strb wd, <addr>	memorizza il byte meno signficativo di w0 in <addr>
strh wd, <addr>	memorizza 2 byte meno signficativi di w0 in <addr>
strw wd, <addr>	memorizza 4 byte meno signficativi di w0 in <addr>

Addressing modes:

1. Base register: <addr> := [Xd]
   Mem[Xd]

2. Offset: <addr> := [Xd, #n]
   Mem[Xd + #n]

3. Pre-indexed: <addr> := [Xd, #n]!
   Xd = Xd + #n
   Mem[Xd]

4. Post-indexed: <addr> := [Xd], #n
   Mem[Xd]
   Xd = Xd + #n