Note Title 31-07-2012

July 31.

Where are we?

- ORR, AND, EOR
- ADD, SUB, MOV, MVN
- LDR, STR (50%)
- B, BL

Instruction Set - II

Logical: ORR, AND

EOR

$$\begin{array}{r} 1011 \\ \oplus\ 1001 \\ \hline 0010 \end{array}$$

A (BIC) $B = A \wedge \overline{B}$

BIC

$$\begin{array}{r} 11\underbrace{010}11 \\ 00111\,00 \\ \hline 11000\,11 \end{array}$$

LSL $R_1, R_2, \#2$

$R_1 = R_2 \times 4$

LSR

1011 >> 2

= 0010

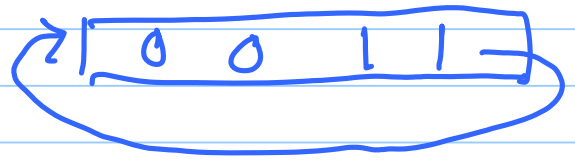
MSB LSB

ASR

1011 [A>>] 2 (-5)

= 11 10 (-2)

ROR



ROR 1 1011 ROR 1 1100

ROL (not required)

Arithmetic Instructions - II.

ADD, SUB ✓

MOV

MVN R_1, R_2 $\quad R_1 = \sim R_2$

Short-cut : 1

Only if 2nd operand is a register

→ This is allowed.

R_2

R_1

$R_3 = R_1 + R_2 \times 4$

ADD R_3, R_1, R_2 [LSL / LSR / ASR / ROR] #2

Short-cut 2

```
if (a > b)
    d = e + f
```

```
3   CMP   R0, R1
    BLE   .exit
    ADD   R2, R3, R4
```

```
2   CMP     R0, R1
    ADDGT   R2, R3, R4
```

Inner workings of cmp

Special register: CPSR

N	Z	C	F

N ← Negative Z → Zero

C ← Carry F → Overflow.

			N	Z	C	F
	(5)	(3)				
CMP	R_1, R_2		0	0	0	0
	(4)	(4)	0	1	0	0
	(3)	(5)	1	0	0	0

LE (N == 1 || Z == 1)

GT (N == 0 && Z == 0)

GE (N == 0)

EQ (Z == 1)

NE (Z == 0)

Short-cut 3

```
c = a + b
if (c > 0)
    d = e + f
```

```
4   ADD   R3, R1, R2
    CMP   R3, #0
    BLE   .exit
    ADD   R4, R5, R6
```

```
2   ADDS  R3, R1, R2
    ADDGT R4, R5, R6
```

ADDEQS

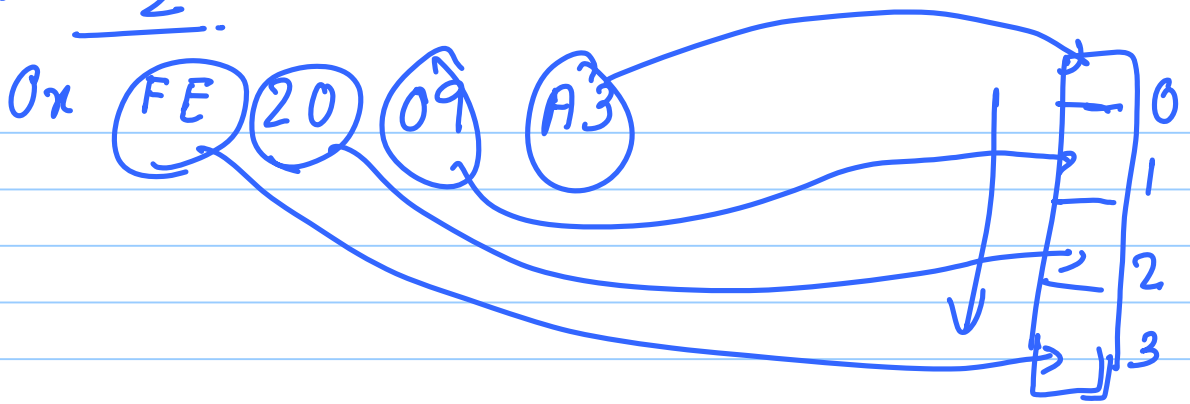
CMP is logically the same as SUBS

Any other arithmetic/logical instruction suffixed with (S) says something about the result through the CPSR.

	(N)	(Z)	(F)	(C)
RSB R_3, R_1, R_2 $R_3 = R_2 - R_1$	res < 0	res == 0	overflow result does not fit	bit falls of from the left

SBC (subtract with carry) [Look up]
ADC (add with carry)

Load/Store - 2.



Little Endian: lowest position saves LSB
(ARM, Intel x86)

Big Endian: lowest position saves MSB
(IBM Power, HP, Java)

LDR STR

(char)

LDRB (load byte)