COMSM1302
Overview of Computer Architecture

Lecture 13

Advanced Math Operations

In the previous lecture

- General introduction to ARM architecture.
- Conditional code flags and conditional execution.
- Data processing instructions
  - Arithmetic and logical operations.
  - Comparisons (no results - just set condition codes)
  - Data movement between registers.
- Branching instructions.

In this lecture

Multiplication and Division → Shifting operations → Fixed Point Math → Square Root

- At the end of this lecture:
  - Solve problems that require multiplication and division.
  - Use shifting operations to do efficient calculations.
  - Use fixed point math to do accurate calculations.

Advance Math Operations

Multiplication and Division → Shifting operations → Fixed Point Math → Square Root

Multiplication Instructions

- Multiply
  - MUL{cond}{S} Rd, Rm, Rs
  - Rd = Rm * Rs

$\underset{32}{Rd} = \underset{32}{Rm} * \underset{32}{Rs} = 64$

~~32~~ 32 32

$R_0 = 0x2$ ×

$R_1 = 0x\text{FFFF FFFF}$

$= \cancel{1}\,\underbrace{\text{FFFF FFFE}}_{32}$

- Multiply Accumulate
  - Does addition for free
  - MLA{cond}{S} Rd, Rm, Rs, Rn
  - Rd = (Rm * Rs) + Rn

UMULL

32, 32 → 32 × 32

SMULL

Division Instructions

- Unsigned integer division
  - UDIV{cond} Rd, Rm, Rs ; Rd = Rm / Rs

9 2 Rd = 4

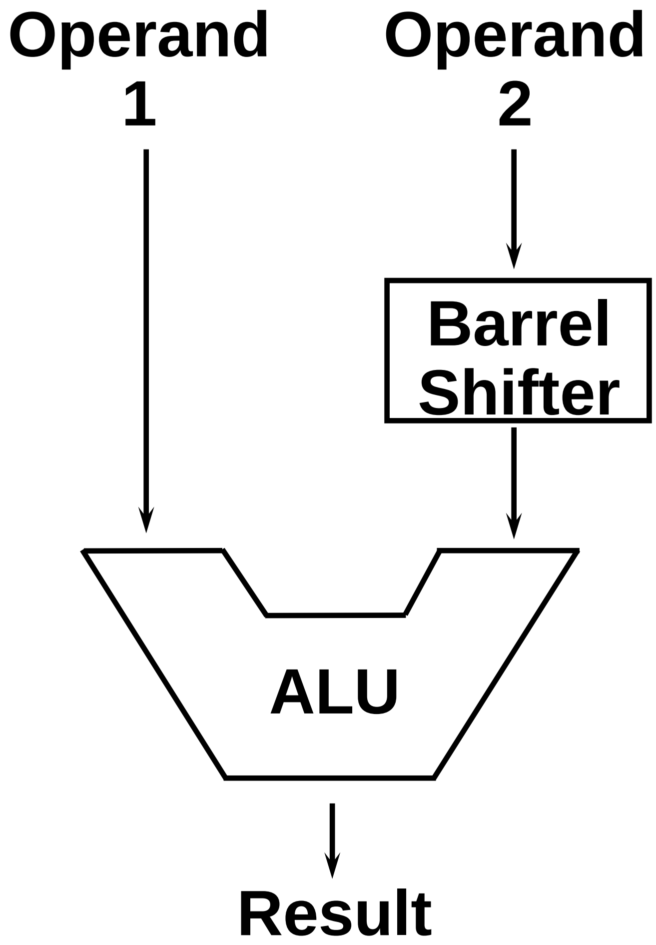
$\boxed{4.5}$

- Signed integer division
  - SDIV{cond} Rd, Rm, Rs ; Rd = Rm / Rs

Advance Math Operations

Multiplication and Division → **Shifting operations** → Fixed Point Math → Square Root

The Barrel Shifter



Barrel Shifter – Logical Left Shift

- Shifts left by the specified amount (multiplies by powers of two) e.g.
  - LSL{cond}{S} Rd, Rm, Rs
  - LSL{cond}{S} Rd, Rm, #sh
  - `MOV r1,r0,LSL #1`
  - `LSL r1,r0,#1`

$R_0 = 0000\ 0011 \rightarrow 3$

$0000\ 0110 \rightarrow 6$

R_1

$R_1 = R_0 * 2$

R_3: $0000\ 1100$

12

2

$R_3 = R_0 \times 2^2$ 4

9.

9.0

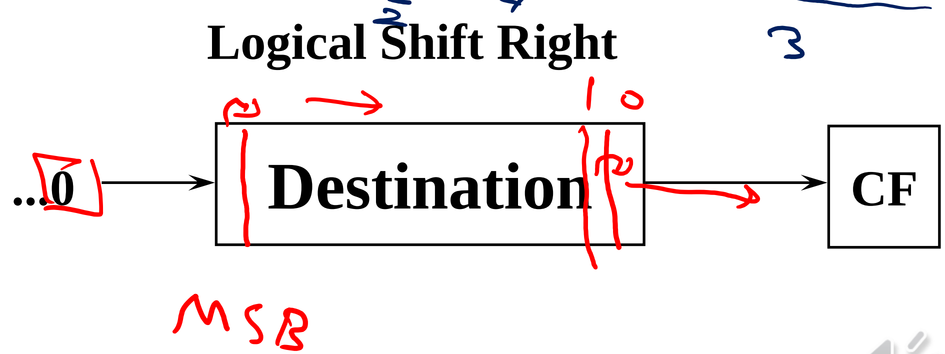
Logical Shift Left (LSL)

Barrel Shifter – Logical Right Shift

- Shifts right by the specified amount (divides by powers of two) e.g.
  - LSR{cond}{S} Rd, Rm, Rs
  - LSR{cond}{S} Rd, Rm, #sh
  - `MOV r1,r0,LSR #1`
  - `LSR r1,r0,#1`

LSR#1

LSR#2



Barrel Shifter – Arithmetic Right Shift

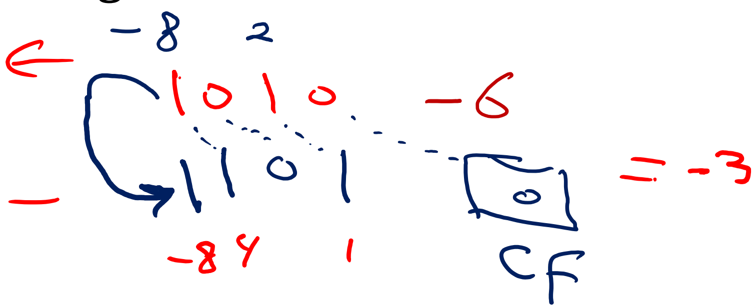
- Shifts right (divides by powers of two) and preserves the sign bit, for 2's complement operations.
  - ASR{cond}{S} Rd, Rm, Rs
  - ASR{cond}{S} Rd, Rm, #sh
  - `MOV r1,r0,ASR #1`
  - `ASR r1,r0,#1`

Arithmetic Shift Right

Sign bit shifted in

R_0 ←

R_1 —



$R_1 = \frac{R_0}{2}$

Barrel Shifter – Shift Operations

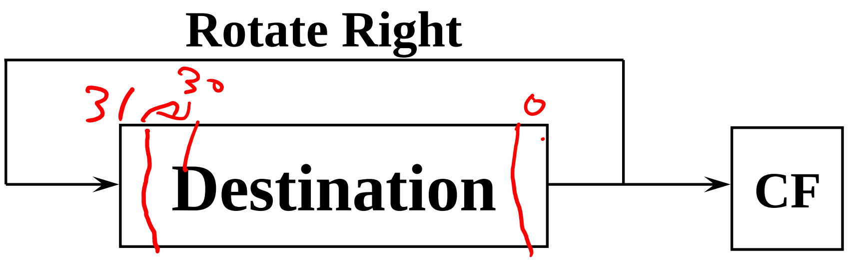
- Logical Shift Left: LSL
- Logical Shift Right: LSR
- Arithmetic Shift Right: ASR

- Why do not we have Arithmetic Shift Left (ARL) ?

Barrel Shifter - Rotations

Rotate Right (ROR)

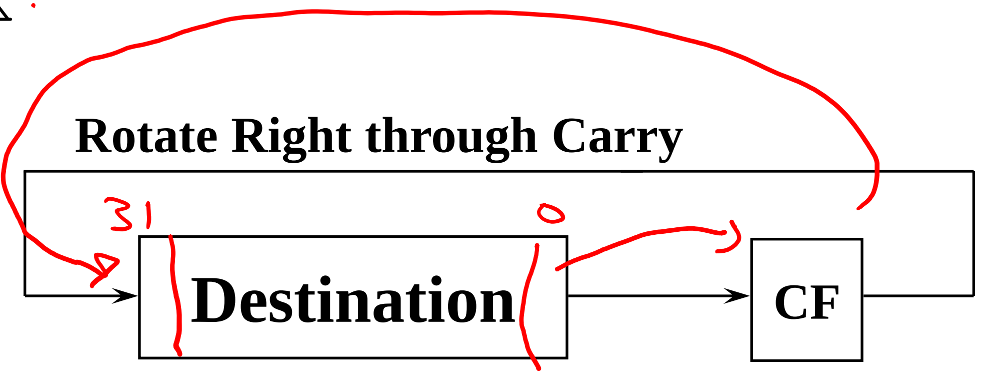
- Similar to an ASR but the bits wrap around as they leave the LSB and appear as the MSB.
  - ROR{cond}{S} Rd, Rm, Rs
  - ROR{cond}{S} Rd, Rm, #sh
  - `MOV r1,r0,ROR #1`
  - `ROR r1,r0,#1`



Barrel Shifter - Rotations

Rotate Right Extended (RRX) by one bit.

- This operation uses the CPSR C flag as a 33rd bit.
  - RRX{cond}{S} Rd, Rm
  - `MOV r1,r0,RRX`
  - `RRX r1,r0`



Using a Shifted Register

- Multiplications by a constant equal to a ((power of 2) ± 1) can be done in one cycle.

- Example: r0 = r1 * 5

  = r1 + (r1 * 4)

2^2 LSL #2

ADD r0, r1, r1, LSL #2

R_1 + $4 \times R_1$

$R_0 = R_1 + (R_1 \times 4)$

Using a Shifted Register -2

- Example: r2 = r3 * 105
  = r3 * 15 * 7
  = r3 * (16 - 1) * (8 - 1)

$r_2 \cdot 7$

$r_2 (8-1)$

$8 r_2 - r_2$

$R_2 = 16 r_3 - r_3 = 15 r_3$

2^4 2^3

$r_3 \times 16$

```
RSB r2, r3, r3, LSL #4    ; r2 = r3 * 15
RSB r2, r2, r2, LSL #3    ; r2 = r2 * 7
```

$8 r_2$ $2^3 = 8$

$r_2 = 8 r_2 - r_2$

$r_2 = 7 (r_2)$

$15 \cdot r_3 \Rightarrow r_2 = 7 \cdot 15 r_3 \Rightarrow r_2 = 105 \times r_3$

Advance Math Operations

Multiplication and Division → Shifting operations → **Fixed Point Math** → Square Root

Fixed Point Math

- We will use first 8 bits for the fraction and the last 24 bits for the integer part. @ 24.8 (I . F)
- What is r0 value that represent the value 1.5

Fixed point

Choose the location of the point carefully, considering

- What **range** do you need?
  - from *<smallest number>* to *<largest number>*
- What **precision** do you need?
  - *What is the required distance between successive numbers?*

2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	
8	4	2	1	0.5	0.25	0.125	0.0625	**Base 10**
0	0	0	0	1	0	1	1	**0.6875**
1	0	0	0	1	0	0	0	**8.5**

2^{23} ... 2^2 2^0 2^{-1} 2^{-2} 2^{-3} ... 2^{-8} → 1/256

1/2 1/4 1/8

R0 | 0 0 0 0 0 0 0 0 0 1 | 1 0 0 0 0 0 0 0 |

31 30 ... 9 8 7 6 5 4 3 2 1 0

$\underbrace{}_{24\ I}$ $\underbrace{}_{8\ F}$

... 0 0 0 180

MOV R0, #0x180

R0 = 1.5

Multiplication with Fixed Point Math

- Multiplication of two fixed point registers cause lose in integer precision.
  - @24.8 * @24.8 = @16.16 → ~~24.16~~

$R_0 \times R_1 = R_2$ R_2 @16.16

$0.2 \times 0.2 = 0.04$

2×1.5

0x200 × 0x180 = 0x30000 → LSR #8

0x00300 = 3

I F

Division with Fixed Point Math

- Division of two fixed point registers cause loss in fractional precision.
  - @16.16 / @24.8 = @16.8

@24.8 / @24.8 = 32

0.2 / 0.2 = 1

1.5 → 180

0x180

3 / 2

0x300 / 0x200 = 0x1 / 0x30000/0x200 =

24.8 → LSL #8 → @16.16

Advance Math Operations

Multiplication and Division → Shifting operations → Fixed Point Math → Square Root

Square Root Algorithm -1/3

- Newton-Raphson's Method (Numerical).
- $y_{n+1} = \left(y_n + \frac{w}{y_n}\right) * {}^1/_2$

input

$\rightarrow n \rightarrow \infty$ $y_n = \sqrt{w}$

n	w	y_n	$\frac{w}{y_n}$	$\left(y_n + \frac{w}{y_n}\right)$	$\left(y_{n_0} + \frac{w}{y_{n_0}}\right) * {}^1/_2$
0	4	y_0 4	1	5	2.5
1	4	y_1 2.5			
2	4	y_2			

$= y_1$

Square Root Algorithm -2/3

- Newton-Raphson's Method (Numerical).
- $y_{n+1} = \left(y_n + \frac{w}{y_n}\right) * {}^1/_2$

n	w	y_n	$\frac{w}{y_n}$	$\left(y_n + \frac{w}{y_n}\right)$	$\left(y_n + \frac{w}{y_n}\right) * {}^1/_2$
0	4	4	1	5	2.5
1	4	2.5	1.6	4.1	2.05
2	4	2.05			

$= y_2$ $n+1$ (1)

Square Root Algorithm -3/3

- Newton-Raphson's Method (Numerical).
- $y_{n+1} = \left(y_n + \frac{w}{y_n}\right) * {}^1/_2$

$\sqrt{4} = 2$

n	w	y_n	$\frac{w}{y_n}$	$\left(y_n + \frac{w}{y_n}\right)$	$\left(y_n + \frac{w}{y_n}\right) * {}^1/_2$
0	4	4	1	5	2.5
1	4	2.5	1.6	4.1	2.05
2	4	2.05	1.951	4.001	$\boxed{2.0006}$

y_2

y_3

$n+1$

0.0006

Square Root Code -1/4

```
MOV r0, #4
MOV r1, r0
_LOOP:
UDIV r2, r0, r1

ADD r3, r1, r2
LSR r4, r3 #.I

MOV r1, r4
B _Loop
```

n	w	y_n	$\frac{w}{y_n}$	$\left(y_n + \frac{w}{y_n}\right)$	$\left(y_n + \frac{w}{y_n}\right) * {}^{1}/_{2}$
	R0	R1	R2	R3	R4
0	4	4	1	5	2.5
1	4	2.5	1.6	4.1	2.05
2	4	2.05	1.951	4.001	2.0006

Square Root Code -2/4

```
MOV r0,#4
MOV r1,r0
_loop:
UDIV r2,r0,r1
ADD r3,r1,r2
MOV r4,r3, lsr #1
MOV r1,r4
B _loop
```

SUB r5, R1, r4
CMP r5, 0.001
BLT _end

_end: b _end

n	w	y_n (R1)	$\frac{w}{y_n}$	$\left(y_n + \frac{w}{y_n}\right)$	$\left(y_n + \frac{w}{y_n}\right) * {}^1/_2$ (R4)
0	4	4	1	5	2.5
1	4	2.5	1.6	4.1	2.05
2	4	2.05	1.951	4.001	2.0006

Square Root Code -3/4

```
MOV  r0,#4
MOV  r1,r0
_loop:
UDIV r2,r0,r1
ADD  r3,r1,r2
MOV  r4,r3, lsr #1
SUB  r5,r1,r4
CMP  r5,0.001
BLT  _end
MOV  r1,r4
B  _loop
_end: b _end
```

MOV r0, r0, LSL #16

MOV r1, r1, LSL #8

R_0 @16.16, R_1 @24.8 → R_2 @ 16.8

@ 24.8

@24.8 → ASR #8 → @32

@24.8

@24.8

n	w	y_n	$\frac{w}{y_n}$	$\left(y_n + \frac{w}{y_n}\right)$	$\left(y_n + \frac{w}{y_n}\right) * {}^1/_2$
0	4	4	1	5	2.5
1	4	2.5	1.6	4.1	2.05
2	4	2.05	1.951	4.001	2.0006

Square Root Code -4/4

```
MOV r0,#4
MOV r1,r0
MOV r0,r0,lsl #16 @16.16
MOV r1,r1,lsl #8 @24.8
_loop:
UDIV r2,r0,r1 @16.8
ADD r3,r1,r2 @24.8
MOV r4,r3, lsr #1 @24.8
SUB r5,r1,r4 @24.8
CMP r5,#1 @24.8
BLT _end
MOV r1,r4 @24.8
B _loop
_end: b _end
```

n	w	y_n	$\frac{w}{y_n}$	$\left(y_n + \frac{w}{y_n}\right)$	$\left(y_n + \frac{w}{y_n}\right) * {}^{1}/_{2}$
0	4	4	1	5	2.5
1	4	2.5	1.6	4.1	2.05
2	4	2.05	1.951	4.001	2.0006

Summary

1. Multiplication and division instructions.
2. Shifting operations.
3. Fixed point math.
4. Example: Newton and Raphson's method