COL216 Lecture 3

Rishabh Dhiman

13 January 2022

1 Format for DP Instructions

1.1 Shift Operations on Registers

Shift type:

- LSL – logical shift left
- LSR – logical shift right
- ASR – arithmetic shift right
- ROR – rotate right

Shift amount can be,

- 5 bit unsigned constant for shift amount,
- 4 bit register.

Rm, LSL #4 – const type 0 Rm Rm, LSL Rs – Rs 0 type 1 Rm

1.2 Rotate operation on Constant

rot	lmm
4	8

lmm is an 8 bit constant (0 to 255) which is zero extended to 32 bits, and rotated right by 2 $\times$ rot bits.

- operand2 = #400 $\rightarrow$ lmm = 100, rot = 15, corresponds to right shift by $32 - 2 \times 15$,
- operand2 = #800 $\rightarrow$ lmm = 50, rot = 14, correspnods to right shift by $32 - 2 \times 14$.

1.3 Multiply

Format for mul r1, r2, r3

	F	l	opc		Rn	Rd	Rs	1001	Rm
4	2	1	4	1	4	4	4	4	4

1.4 Multiply Accumulate

Format for mla r1, r2, r3, r4; – r1 ← r2 × r3 + r4.

	00	0	opc		Rn	Rd	Rs	1001	Rm

1.5 Multiplying by a constant

mul r1, r2, #10 → mov r3, #10; mul r1, r2, r3;

Multiplying by a power of 2 is equivalent to a logical left shift.

2 Format for DT Instructions

Rd ← Memory [Rn + offset]

	F	opc		Rn	Rd	offset
4	2	6	1	4	4	12

F = 01.

The 12-bit offset can be

- 12 bit unsigned constant,
- 4 bit register number, 8 shift specification

2.1 Load

ldr r4, [r5, #32], opc = 25

2.2 Indexing an Array

```
mul r4, r5, #4
add r2, r2, r4
ldr r6, [r2, #0]

add r2, r2, r5, LSL #4
ldr r6, [r2]
```

Further reduction (fill it in):

```
ldr r6, [r2, r5, LSL #4]
```

2.3 Opcode field in DT Instructions

6 opcode bits specify,

- I (immediate): constant or register with shift
- P (pre/post indexing): pre or post indexing
- U (up/down): whether to add or subtract offset

- B (byte): byte or word transfer
- W (write back): whether to write back address into base register (Rn) or not.
- L (load/store): load from memory or store from memory

Post-indexing only makes sense with write-back.
Load half-word is a completely unrelated operation.

- `ldr r4, [r5, -r6]`
- `str r4, [r5, r6, LSL #2]`
- `ldrb r4, [r5, #32]!`
- `strb r4, [r5, #-32]`
- `ldr r4, [r5], r6`
- `ldr r4, [r5], r6, LSL #2`

2.3.1 Using auto-incrememnt/decrement

When pointer is at first location vs when it points to the (possibly empty) element before address location.

- get/pop: post-increment, pre-increment
- put/push: pre-decrement, post-decrement