Introduction

- In later labs we will be using C code to write instructions
- In this section we want to take a look at the low level features of the ARM processor
  - ↳ for this we need a basic idea of Assembly language

* knowing the low level features of a processor allows us to make informed decisions on which processors to use

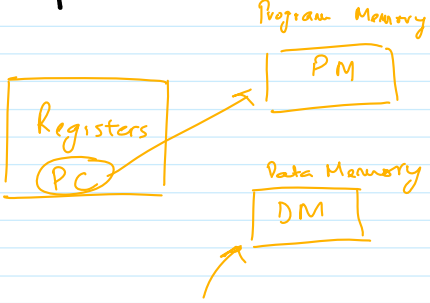
- We shall talk broadly about the various categories of ARM instructions
  - ↳ our focus will be on the data processing instructions

ARM Instruction Set

- ARM instructions can be categorized into 3 groups

① Data processing Instructions (this section)
- ↳ operate on values in registers
- ↳ carry out various arithmetic and logic operations on data

② Data Transfer Instructions
- ↳ move values between registers and memory



③ Control Flow Instructions
- ↳ change the value of the program counter (PC)

PC - Program Counter (review)

- PC is a register that will be pointing to the address of the next instruction in the program memory.

- Whenever a new instruction is brought or fetched from memory, it will be fetched from the address which is stored in the PC

Arithmetic Instructions

* Multiply 2 32-bit nos
  → result = 64 bit no.

```
ADD     r0, r1, r2      ; r0 = r1 + r2
ADC     r0, r1, r2      ; r0 = r1 + r2 + C      (C: carry bit)
SUB     r0, r1, r2      ; r0 = r1 - r2
SBC     r0, r1, r2      ; r0 = r1 - r2 + C - 1
RSB     r0, r1, r2      ; r0 = r2 - r1          (reverse subtract)
RSC     r0, r1, r2      ; r0 = r2 - r1 + C - 1
```

- All operations are viewed as either unsigned or 2's compliment signals.

Bit-wise logical instructions

```
AND     r0, r1, r2      ; r0 = r1 AND r2
ORR     r0, r1, r2      ; r0 = r1 OR r2
EOR     r0, r1, r2      ; r0 = r1 XOR r2
BIC     r0, r1, r2      ; r0 = r1 AND NOT r2
```

Register-register Move operations

```
MOV     r0, r1          ; r0 = r1
MVN     r0, r1          ; r0 = NOT r2
```

MVN: move negated

Comparison Instruction

```
CMP     r1, r2          ; set cc on (r1 - r2)
CMN     r1, r2          ; set cc on (r1 + r2)
```

```
CMN     r1, r2          ; set cc on (r1+r2)
TST     r1, r2          ; set cc on (r1 and r2)
TEQ     r1, r2          ; set cc on (r1 XOR r2)
```

- These instructions affect the condition codes (N, Z, C, V) in the current program status register (CPSR)
  - ↳ These instructions do not result in any register (r0)
  - ↳ They only set the condition flags so you can use the results later

Specifying Immediate Operands

```
ADD     r1, r2, #2      ; r1 = r2 + 2
SUB     r3, r3, #1      ; r3 = r3 - 1
AND     r6, r4, #&0f    ; r6 = r4[3:0]
```

- Notations:
  - → # indicates immediate value
  - → & indicates hexidecimal notation
- Allowed immediate values:
  - ↳ 0 to 255 (8 bits) rotated by any number of bit positions that is a multiple of 2.

Shifted Register Operands

- The second source operand may be shifted either by a constant number of bit positions or by a register specific number of positions

```
ADD     r1, r2, r3, LSL #3      ; r1 = r2 + (r3 << 3)
ADD     r1, r2, r3, LSL r5      ; r1 = r2 + (r3 << r5)
```

- Various shift and rotate options

LSL → logic shift left

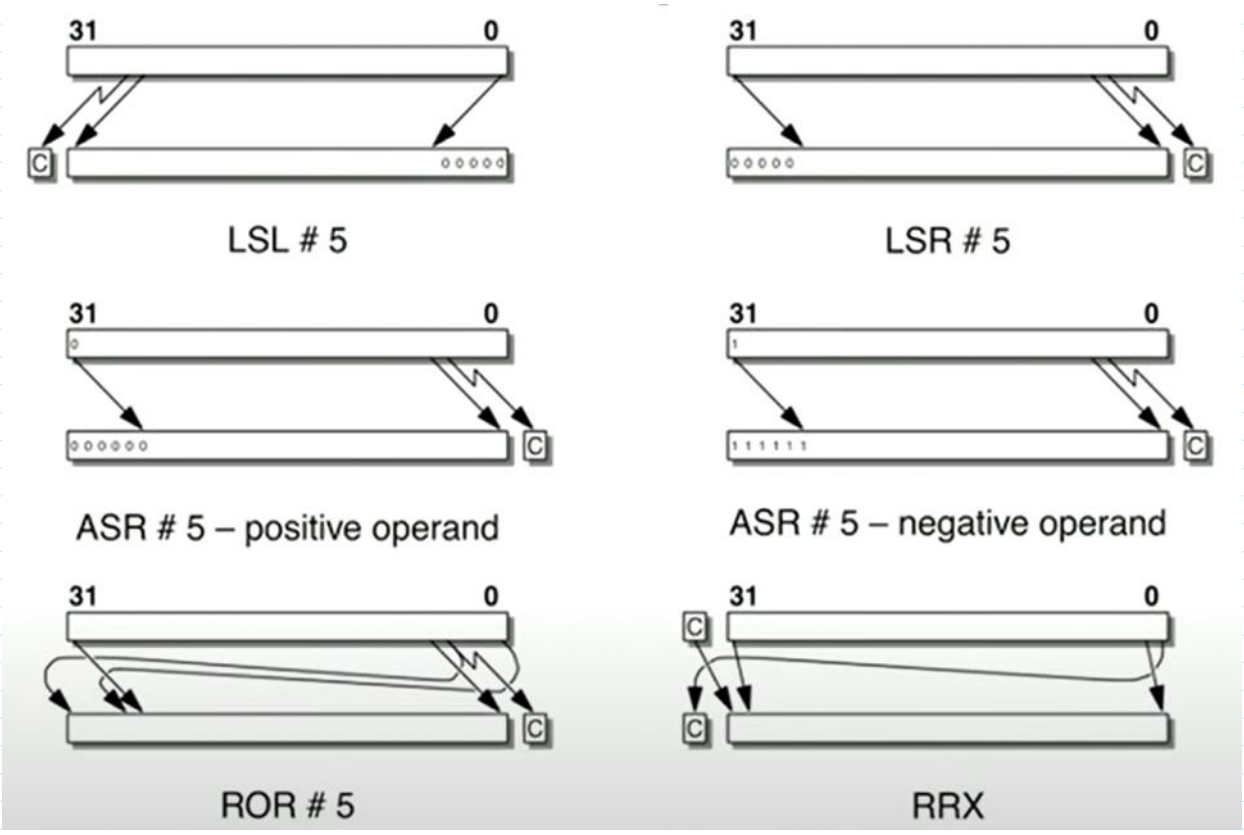
LSR → logic shift right

ROR → rotate right

RRX → rotate right extend by 1 bit

ASL → arithmetic shift left

ASR → arithmetic shift right



Multiplication Instruction

```
MUL     r1, r2, r3      ; r1 = (r2 x r3) [31:0]
```

* only the least significant 32 bits are returned
* immediate operands are not supported

Multiply-accumulate Instruction

```
MLA     r1, r2, r3, r4      ; r1 = (r2 x r3 + r4) [31:0]
```

* required in digital signal processing applications