ARM Instructions

- General features
  - All instructions are encoded as 32-bit words
  - Only load and store instructions can access memory
  - Arithmetic and logic instructions use register operands
- Load and Store instructions
  - LDR & STR read and write 32-bit memory words
  - LDRB & LDRH read 8-bit or 16-bit values into low part of register
    - High bits within register are zero filled
  - LDRSB & LDRSH read 8-bit or 16-bit values into low part of register
    - High bits within register are filled with copies of sign bit

ARM Instructions

- Other Store instructions
  - STRB stores the low byte of a register into memory
  - STRH stores the low 16 bits of a register into memory
    - Must use half-word aligned address (i.e., multiple of 2)

ARM Instructions

- Block Transfer Instructions
  - Transfer a block of consecutive memory words to or from a subset of general purpose registers
    - List of registers must appear in increasing order

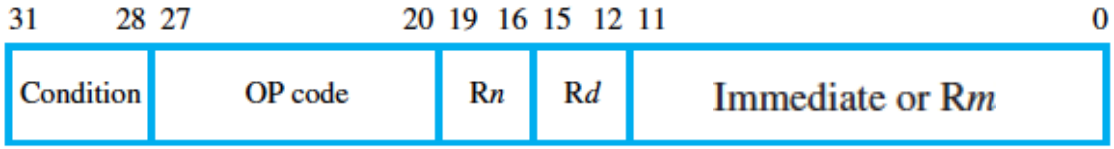
Example: base register R10 contains 1000 initially

```
LDMIA   R10!,{R0, R2, R5, R7}
```

Loads contents of locations 1000,1004,1008 and 1012 into registers R0, R2, R5 and R7. Suffix IA means “increment after” (i.e., post increment) So R10 is incremented by 4 after each word transfer and final value 1016 is written to R10 due to the “!” writeback indicator.

ARM Instructions

- Arithmetic instructions
  - Use registers or immediate operands
  - Assembly syntax is OP Rd,Rn,Rm
  - Machine code format:



Addition and Subtraction

The instruction

```
ADD   R0, R2, R4
```

performs the operation

$$R0 \leftarrow [R2] + [R4]$$

The instruction

```
SUB   R0, R6, R5
```

performs the operation

$$R0 \leftarrow [R6] - [R5]$$

ARM Instructions

The second source operand can be specified in the immediate mode. Thus,

```
ADD     R0, R3, #17
```

performs the operation

$$R0 \leftarrow [R3] + 17$$

The immediate operand is an 8-bit value contained in bits b_{7-0} of the encoded machine instruction. It is an unsigned number in the range 0 to 255. The assembly language allows negative values to be used as immediate operands. If the instruction

```
ADD     R0, R3, #−17
```

is used in a program, the assembler replaces it with the instruction

```
SUB     R0, R3, #17
```

ARM Instructions

- Shifting of second source register operand
  - Register second operand can be shifted before use
    - LSL (logical shift left)
    - LSR (logical shift right)
    - ASR (arithmetic shift right)
    - ROR (rotate right)
  - Specified after the register name:

    ADD R0,R2,R4,LSL #4
    - R4 is shifted left 4 bits (R4*16), then added to R2 & sum is placed into R0.

ARM Instructions

- Shifting of second source immediate operand
  - Contained in low byte of machine instruction (0 – 255)
  - Expanded into a limited number of 32-bit values
    - Programmer specifies value
    - Assembler zero-extends 8-bit value to 32 bits & rotates an even number of bits to generate the value
    - The shift amount and 8-bit value are encoded in the low-order 12 bits of the machine instruction

ARM Instructions

- Multiple-Word Operands
  - Carry flag, C, is used to perform multiple precision addition & subtraction
  - ADC (add with carry) & SBC (subtract with carry)

    Example - to add the 64-bit number in the register pair R2,R3 to the 64-bit number in the register pair R4,R5:

```
ADDS    R7,R3,R5    ;add low parts
ADC     R6,R4,R2    ;add high parts & include carry
```

    The S suffix causes the ADD to set the condition flags

ARM Instructions

- Multiplication
  - Two basic forms of multiply instruction
  - MUL R0,R1,R2 ; R0 ← [R1] × [R2]
    - Only the low 32 bits of 64-bit product are retained
  - MLA R0,R1,R2,R3 ; R0 ← ([R1] × [R2]) + [R3]
    - Called multiply and accumulate
    - Often used in signal-processing applications
  - Other versions of MUL and MLA are available
    - Generate 64-bit results
    - Handle signed and unsigned operands

ARM Instructions

- No provision for shifting or rotating operands in multiplication
- No divide instructions
  - Division must be done in software