

FLAGS (Revision)

- As mentioned earlier, flags are located in the most significant 4 bits (N Z C V) in CPSR.
- Instructions can be made to set flags by appending an "S" to the mnemonic, e.g. using EORS instead of EOR.

N Flag

- Checks for a negative result.
- Negative if most significant bit is 1.
- In both of the following cases, the N Flag is set

FFFFFFFF 7B000000 + FFFFFFFE +30000000 FFFFFFFD AB000000 Negative?

Programmer has to decide whether to use the N flag

V Flag

- The V flag indicates a sign overflow
- V flag is calculated by doing an exclusive OR of the carry bit going into bit 31 of the result with the carry bit coming out of bit 31.
- In the second example in previous slide, the V flag will be set and the programmer will ignore the N flag.

Z flag

- Z flag tell us the result is All Zeroes
- All 32 bits must be 0.

C Flag

- The C flag is set if
- result of an addition is greater than or equal to 2³²
- Result of a subtraction is positive (Why???)
- Result of an inline barrel shifter operation in a move or logical instruction.

COMPARISON INSTRUCTIONS

- 4 instructions do nothing except set the condition codes or test for a particular bit
- CMP Compare. CMP subtracts a register or an immediate value from a register value and updates the condition codes. You can use CMP to quickly check the contents of a register for a particular value, such as at the beginning or end of a loop.
- E.g.

CMP r8, #0 ;r8==0

BEQ routine; yes, then go to my routine

CMN

- CMN Compare negative. CMN adds a register or an immediate value to another register and updates the condition codes.
- It is the inverse of CMP
- If you typed CMP r0, #-20, the assembler will generate CMN r0, #0x14 (note 0x14 = 20)

Test Instructions

- TST Test. TST logically ANDs an arithmetic value with a register value and updates the condition codes without affecting the V flag.
- TEQ Test equivalence. TEQ logically exclusive Ors an arithmetic value with a register value and updates the condition codes without affecting the V flag. You can use TEQ to determine if two values are the same. E.g.
 TEQ r9, r4, LSL #3

Boolean Operations

Instruction Comment

AND Logically ANDs two operands

ORR Logically ORs two operands

EOR Exclusive OR of two operands

MVN Move Negative – logically NOT's all

BIC Bit Clear - clears selected bits in a

register

bits

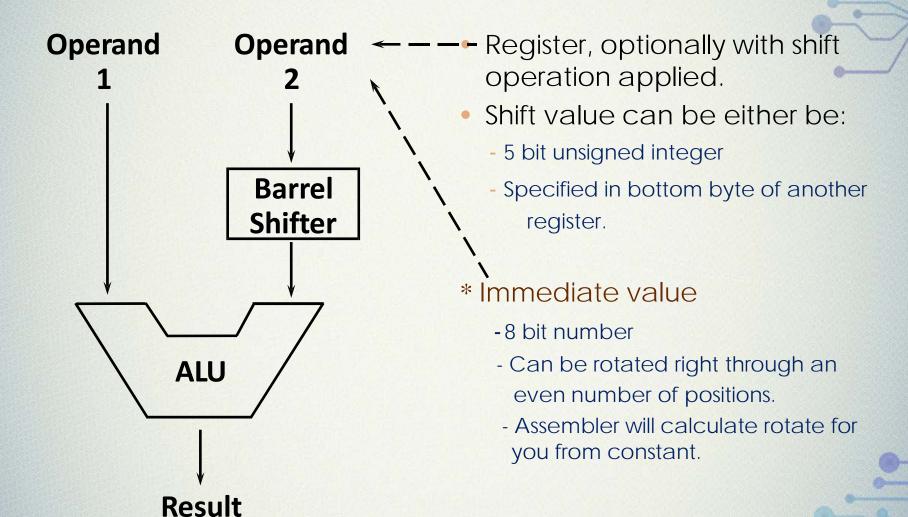
Examples

- A very fast way to load the two's complement of -1
 (0xFFFFFFFF) into a register is to logically invert zero, e.g.
 MVN r5, #0; r5 = -1 in two's complement
- AND r1, r2, r3; r1 = r2 AND r3
- ORR r1, r2, r3 ; r1 = r2 OR r3
- EOR r1, r2, r3; r1 = r2 EOR r3

Bit Clear Example

- BIC r1, r2, r3; r1 = r2 AND NOT r3
- The Bit Clear instruction can be used to clear selected bits in a register. For each bit in the second operand, a 1 clears the corresponding bit in the first operand and a 0 leaves it unchanged.
- BIC r2, r3, #0xFF0000000; clears the upper byte of register r3 and moves the result to register r2. Same use as AND r2, r3, #0x00FFFFFF

Using the Barrel Shifter: The Second Operand



Barrel Shifter - Left Shift

- Shifts left by the specified amount (multiplies by powers of two) e.g.
- LSL #5 => multiply by 32

Logical Shift Left (LSL)

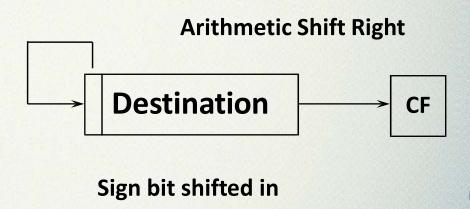


Barrel Shifter - Right Shifts

Logical Shift Right (LSR)
Shifts right by the
specified amount
(divides by powers of
two) e.g. LSR #5 = divide by 32



Arithmetic Shift Right (ASR)
Shifts right (divides by powers of two) and preserves the sign bit, for 2's complement operations. e.g. ASR #5 = divide by 32



Barrel Shifter - Rotations

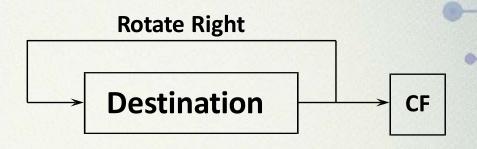
Rotate Right (ROR)

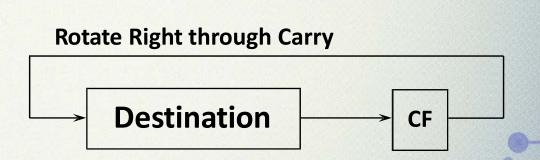
Similar to an ASR but the bits wrap around as they leave the LSB and appear as the MSB.

e.g. ROR #5

Note the last bit rotated is also used as the Carry Out.

Rotate Right Extended (RRX)
This operation uses the CPSR
C flag as a 33rd bit.





Summary Of Shifts And Rotates

Shift/Rotate	Operation	Use
LSL	Logical shift left by n bits	Multiplication by 2 ⁿ
LSR	Logical shift right by n bits	Unsigned division by 2 ⁿ
ASR	Arithmetic shift right by n bits	Signed division by 2 ⁿ
ROR	Rotate right by n bits	32-bit rotate
RRX	Rotate right extended by one bit	33-bit rotate, 33 rd bit is Carry flag

Shift and rotate examples

- MOV r4, r6, LSL #4; r4 = r6 << 4 bits
- MOV r4, r6, LSL r3; r4 = r6 << # specified in r3
- MOV r4, r6, ROR #12; r4 = r6 rotated right 12 bits
- All shift operations take one clock cycle to execute, except register-specified shifts which take another clock cycle.
- Shifts and rotates are a very fast way to perform certain multiplications and divisions.

Example

 The shift and logical operations can also be used to move data from one byte to another. Suppose we need to move the uppermost byte from register r2 and put it at the bottom of register r3. The contents of register r3 are shifted left by 8 bits first.

MOV r0, r2, LSR #24 ; extract top byte from r2 into r0

ORR r3, r0, r3, LSL #8; shift up r3 and insert r0

Workings

- Let r2 contains 0xD0ABCDEF
- MOV r0, r2, LSR #24 will result in r0 containing
 0x000000D0 or 0xD0 (the upper most byte of r2)
- Assume r3 contains 0xABCD
- After LSL #8, r3 becomes 0xABCD00
- After the OR operation, r3 becomes 0xABCDD0!

Summary

- Know your flags, N, V, Z, C
- Comparison instructions CMP, CMN
- Boolean Operations
- Shift and Rotates