Addition Examples (4-bit)

Arithmetic	Signed Interpretation	Unsigned Interpretation	N	Z	C	V
1111 +	-1 +	15 +	1	0	1	0
1111	-1	15				
1 1110	-2	14				
0001 +	1 +	1 +	0	0	0	0
0010	2	2				
0 0011	3	3				
1010 +	-6+	10+	0	0	1	1
1001	-7	9				
1 0011	3	3				
0010 +	2 +	2+	1	0	0	1
0111	7	7				
0 1001	-7	9				
1111 +	-1 +	15 +	0	0	1	0
0010	2	2				
1 0001	1	1				
0000 +	0 +	0 +	0	1	0	0
0000	0	0				
0 0000	0	0				
0001 +	1 +	1 +	0	1	1	0
1111	-1	15				
1 0000	0	0				

Observe :

- Unsigned results are wrong when C=1
- Signed results are wrong when V=1

Obviously, N and Z cannot be 1 simultaneously

Can Z and V be 1 simultaneously?

Signed Addition

- For *unsigned* addition, carry is an indication of the result being wrong

- For *signed addition*,
  - Overflow occurs when (+ve) + (+ve) = (-ve) or (-ve) + (-ve) = (+ve)
  - If the sign bits (MSB) of the operands are the same, and is different from that of the result, there is an overflow
  - Overflow can never occur when you add two numbers of different signs (they tend to cancel each other)
  - Carry and overflow are unrelated
  - The hardware as well as the instruction used for performing both signed and unsigned addition are the same – only the *interpretation of the result* and/or *the way the resultant flags are used* are different

Signed Subtraction

- For *unsigned* subtraction, borrow is an indication of the result being wrong. However, note that ARM implements carry flag as not(borrow) for subtraction
- For *signed* subtraction,
  - Overflow occurs when (+ve) - (-ve) = (-ve) or (-ve) - (+ve) = (+ve)
  - If the sign bits (MSB) of the operands are different, and the result has a sign same as that of the second operand, there is an overflow
  - Overflow can never occur in subtraction when operands are of the same sign (they tend to cancel each other)
  - Carry and overflow are unrelated
  - The hardware as well as the instruction used for performing both signed and unsigned subtraction are the same – *only the interpretation of the result* and/or *the way the resultant flags are used* are different
    - For example, for checking '>', we should use GT when the flags are set by a SUBS/CMP operation on signed numbers, whereas it should be HI for unsigned numbers

Condition codes and Flags for CMP / SUBS

cond	Mnemonic	Name	CondEx
0000	EQ	Equal	Z
0001	NE	Not equal	$\bar{Z}$
0010	CS / HS	Carry set / Unsigned higher or same	C
0011	CC / LO	Carry clear / Unsigned lower	$\bar{C}$
0100	MI	Minus / Negative	N
0101	PL	Plus / Positive of zero	$\bar{N}$
0110	VS	Overflow / Overflow set	V
0111	VC	No overflow / Overflow clear	$\bar{V}$
1000	HI	Unsigned higher	$\bar{Z}C$
1001	LS	Unsigned lower or same	$Z\ OR\ \bar{C}$
1010	GE	Signed greater than or equal	$\overline{N \oplus V}$
1011	LT	Signed less than	$N \oplus V$
1100	GT	Signed greater than	$\bar{Z}(\overline{N \oplus V})$
1101	LE	Signed less than or equal	$Z\ OR\ (N \oplus V)$
1110	AL (or none)	Always / unconditional	ignored

The meanings/interpretations of condition codes above are based on the result of SUBS/CMP operation

For RISC-V, the condition codes (NZCV) are not stored for use by a future instruction, but used within the same instruction for branching. beq (Z), bne ($\bar{Z}$), blt ($N \oplus V$), bge ($\overline{N \oplus V}$), bltu ($\bar{C}$), bgeu (C)