CSCI 6461 Computer Systems Architecture

Lecture 4

ISA: Condition Code Flags

Condition Flags

As mentioned previously, ARM maintains condition flags.

These flags are used to capture state information of the system. There are 4 flags, as follows

Flag	Name	Description
N	Negative	Instruction result is negative, i.e., bit 31 of the result is 1
Z	Zero	Instruction result is zero
C	Carry	Instruction causes a carry out
V	oVerflow	Instruction causes an overflow

Set by the ALU

- Condition flags are set by the ALU and are held in the top 4 bits of the 32-bit Current Program Status Register (CPSR/APSR)
- Subsequent instructions then execute conditionally, depending on the state of those condition flags.

Set and cleared

- Flags are set and cleared based on instructions that are specifically used for setting and clearing flags
  - eg, `TST` or `CMP`
- Instruction are told to set the flags by appending an "**S**" to the mnemonic.
- For example
  - `subs r2 r3, r7 //R2=R3-R7` and set condition flags
  - `eors` would perform an exclusive OR operation and set the flags afterward

Instructions that affect condition flags

Type	Instructions	Condition Flags
Add	ADDS, ADCS	*N*, *Z*, *C*, *V*
Subtract	SUBS, SBCS, RSBS, RSCS	*N*, *Z*, *C*, *V*
Compare	CMP, CMN	*N*, *Z*, *C*, *V*
Shifts	ASRS, LSLS, LSRS, RORS, RRXS	*N*, *Z*, *C*
Logical	ANDS, ORRS, EORS, BICS	*N*, *Z*, *C*
Test	TEQ, TST	*N*, *Z*, *C*
Move	MOVS, MVNS	*N*, *Z*, *C*
Multiply	MULS, MLAS, SMLALS, SMULLS, UMLALS, UMULLS	*N*, *Z*

When an **S** is appended to the instruction mnemonic, condition flags are changed as appropriate

Example: CMP

- `CMP` – the compare instruction
- It subtracts the 2nd source operand from the 1st and sets the condition flags based on the result
- If the numbers are equal
  - the result is zero
  - the Z flag is set
- if `op1 < op2`
  - the result is negative and N is set

NZCV duration?

How long does the NZVC flags remain after being set?

```
mov r2, #5
mov r3, #10
cmp r2, r3
NZCV = 1000
```

- The `1000` pattern will continue **through the program** until such time as some operation changes it
- It does not get "reset" after the `CMP`

NZCV duration - example

```
mov r2, #5
mov r3, #10
cmp r2, r3
sub r4, r3, r2
```

NZCV=1000

```
mov r2, #5
mov r3, #10
cmp r2, r3
subs r4, r3, r2
```

NZCV=0010

Mnemonic and Condition

The instruction mnemonic is followed by a condition mnemonic that indicates when to execute.

```
cmp r4, r5
addeq r1, r2, r3
```

- The CMP sets the Z flag if R4 and R5 are equal
- The ADDEQ executes only if the Z flag is set.

Mnemonic	Name	CondEx
EQ	Equal	Z
NE	Not equal	$\overline{Z}$
CS/HS	Carry set / unsigned higher or same	C
CC/LO	Carry clear / unsigned lower	$\overline{C}$
MI	Minus / negative	N
PL	Plus / positive or zero	$\overline{N}$
VS	Overflow / overflow set	V
VC	No overflow / overflow clear	$\overline{V}$
HI	Unsigned higher	$\overline{Z}C$
LS	Unsigned lower or same	Z OR $\overline{C}$
GE	Signed greater than or equal	$\overline{N \oplus V}$
LT	Signed less than	$N \oplus V$
GT	Signed greater than	$\overline{Z}(\overline{N \oplus V})$
LE	Signed less than or equal	Z OR $(N \oplus V)$
AL (or none)	Always / unconditional	Ignored

N flag

- This flag is useful when checking for a negative result.
- A two's complement number is considered to be negative if the most significant bit is set.
- EG -1 -2 = -3

```
mov r3, #-1
mov r4, #-2
adds r3, r4
```

N flag

The addends are positive in two's complement notation, but the sum is negative,

```
mov r3, #0x7b000000
mov r4, #0x30000000
adds r5, r4, r3
```

$$\begin{array}{r} 7B000000 \\ +\ 30000000 \\ \hline AB000000 \end{array}$$

```
NZCV = 1001
```

Since the most significant bit is now set, this forces the N bit to be set The result indicates that this positive sum cannot be represented in 32 bits, so the result effectively overflowed the precision we had available.

Z Flag

The Z flag tells us is that the result of an operation produces zero.

- All 32 bits must be zero.
- Add `#0xFFFFFFFF` and `#0x00000001`
  - The result is 0 `NZCV = 0110`
- Subtract `#0xFFFFFFFF` and `#0xFFFFFFFF`
  - The result is 0, `NZCV = 0110`

Carry Flag

The Carry flag is set if the result of an addition is $\geq 2^{32}$

```
mov r3, #0x7b000000
mov r7, #0xf0000000
adds r4, r7, r3 //value exceeds 32 bits

NZCV = 0010
```

Carry Flag

- The carry flag will **also** be set in the event of a subtraction that yields a positive number, for example:

```
mov r0, #5
mov r2, #10
subs r4, r2, r0
```

```
NZCV = 0010
```

V Flag

- The V flag indicates an **overflow** in adding when the numbers are interpreted as 2's complement (signed) numbers.
- Signed overflow:
  - if the two numbers being added have opposite signs then overflow is impossible
    - why?
  - if the two number have the same signs but the result has the opposite sign from the operands then overflow has occurred.
    - No need to look at the carry.

V flag, example

```
     1010 0001 0010 0011 0100 0101 0110 0111 (2,703,443,303 = -1,591,523,993)
+    1011 0000 0000 0000 0000 0000 0000 0000 (2,952,790,016 = -1,342,177,280)
0001 0101 0001 0010 0011 0100 0101 0110 0111 (5,656,233,319)
```

The two numbers of the same sign are added and their sign inverts. This is an **overflow** .

```
NZCV = 0011
```

```
  A1234567
+ B0000000
-----------
 151234567
```

A note on SUB and NZCV

In fact, a SUB operation can have a number of condition flag (NZCV) outputs.

Consider `subs r0, r1, r2`, and NZCV values:

- `0010`
- `0110`
- `1000`

Give some sample values for `r1` and `r2` that might yield this configuration