Microprocessor Systems

Flags

The overflow flag

- In two's complement any hexadecimal number which has an 8 or greater for the most significant digit is negative since the 'sign bit' or m.s.b. is 1.
- So if we add two very big numbers so that the sum is greater than $(2^{31} - 1)$ then that number will be negative if we are working in 2's complement.
- The overflow flag can be used to identify when a 2's complement result goes 'out of range'.

Example

```
  0x59  68 2F 00
+ 0x41  90 AB 00
  0x9A F8 DA 00
```

- If we are working in two's complement the sum is a negative number and is clearly incorrect.
- If the instruction ADDS is used the negative flag would be set but the carry flag would be cleared because the sum is still a 32 bit result.

In Binary:

```
  0101 1001 0110 1000 0010 1111 0000 0000
+ 0100 0001 1001 0000 1010 1011 0000 0000
  1001 10...        ← result (top 6 bits)
```

The overflow flag, V, is set because there is an overflow into the 'sign bit'.

Adding Negative Numbers

- Using two's complement we can do sums such as x + (-y) for example if we add $1{,}500{,}000{,}000_{10}$ to $-1{,}100{,}000{,}000_{10}$
- First find the 2's complement of $-1{,}100{,}000{,}000_{10}$

Positive number	0x4190AB00
Invert	0xBE6F54FF
Add 1	0xBE6F5500

- and now add this to 0x59682F00

Adding Negative Numbers

```
   0x  59 68 2F 00
+  0x  BE 6F 55 00
  ------------------
  0x1 17 D7 84 00
```

- Because we are working in 2's complement we can ignore the carry 1 and the answer in the lowest 32 bits is 0x17D78400 or $400{,}000{,}000_{10}$ which is correct.
- The carry flag is set but the overflow and negative flags are cleared.

Adding Negative Numbers Cont'd

- Consider this in Binary:

```
  0101 1001 0110 1000 0010 1111 0000 0000
+ 1011 1110 0110 1111 0101 0101 0000 0000
  0001 01…    ← result (top 6 bits)
1 1111 00…    ← carry from previous column
```

- The sign bit of the result is 0 indicating a positive result. A carry out occurs so C is set but the 32 bit result is correct so V is cleared.

A negative result

- What happens if we add two negative numbers or if we add a smaller positive number to a bigger negative number - the result should be negative.
- E.g.
  - if we add 1,100,000,00010 to -1,500,000,00010
  - The 2's complement of -1,500,000,00010 is 0xA697D100 and add this to 0x4190AB00.

```
    0x41 90 AB 00
+   0xA6 97 D1 00
    -------------
    0xE8 28 7C 00
```

A negative result

$$\begin{array}{rr} & \text{0x41 90 AB 00} \\ + & \underline{\text{0xA6 97 D1 00}} \\ & \text{0xE8 28 7C 00} \end{array}$$

- The result is negative as expected since the sign bit is 1 but is it correct?
  - Apply 2's complement to the result to find it's positive value
  - invert 0xE8287C00 to get 0x17D783FF and add 1 to find: 0x17D78400
  - which is $400{,}000{,}000_{10}$ as expected.

2’s Complement Subtraction

- Subtraction, such as x - y, is implemented in a microprocessor by finding the two’s complement of (-y) and then performing the addition, x + (-y).
- So if we subtract a big positive number from a small negative number so that the difference is less than -2^{31} then that number will be positive if we are working in 2’s complement.

A negative overflow?

0xBE6F5500 – 0x59682F00 becomes

0xBE6F5500 + 0xA697D100

```
   0x  BE  6F 55 00
 + 0x  A6  97 D1 00
   0x1 65  07 26  00
```

- Ignoring the carry 1 the 32 bit result 0x65072600 is +ive (= $1{,}694{,}967{,}296_{10}$) and clearly incorrect.
- Again this would set the overflow flag.

Flags Summary

- The zero flag, Z, is set when the result (not including any carry out) is 0x00000000.
- The negative flag, N, is set when the most significant bit of the 32 bit result is 1.
- The carry flag, C, is set when the result (taken as an unsigned integer) is greater than (2^{32} - 1).
- The overflow flag, V, is set when the result (taken as a two's compliment number) is greater than (2^{31} - 1) or less than -2^{31} *(for two numbers with the same signs)*