CMPT-150

Introduction to Computer Design

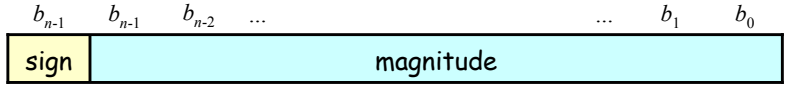
SFU, Harbour Centre, Spring 2007

Lecture 11: Feb. 13, 2007

- Signed Arithmetic and signed number representations
  - Sign-magnitude
  - 1's complement
  - 2's complement
- Overflow

Binary Representations for Signed Numbers

- How to deal with negative numbers in binary representation?
- First solution: Sign-Magnitude



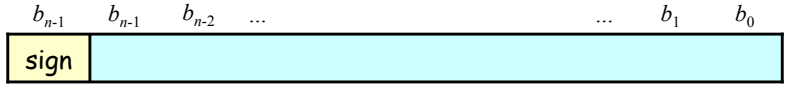
  - The magnitude part indicates the absolute value of the number
  - The sign bit indicates if the number is negative (1) or positive (0)
- Examples: Assume n=4. **0110→6** **1110→-6**
- Range: $-2^{n-1}-1 \le N \le 2^{n-1}$
- Advantage: Intuitive
- Disadvantages:
  - Double representation for zero: **000…0 = 0** **100…0 = -0**
  - Complicates arithmetic
  - Complex logic

Representation using 1's Complement

- The 1's complement of an n-bit binary number N: $(2^n - 1) - N$

  since $2^n - 1 = \underbrace{111\ldots1}_{n}$, it is the bitwise-NOT of N

0	1	1	0	1	0	1	1
1	0	0	1	0	1	0	0



- Again the most significant bit (b_{n-1}) indicates the sign.
- b_{n-1}=0 implies the coded number is $b_{n-2}\ldots b_0$
- b_{n-1}=1 implies the coded number is the 1's complement of $b_{n-1}\ldots b_0$
- Examples: Assume n=4. **0110→6** **1110→-0001=-1**
- How to code a negative number? Take its complement!

  Example: How to code -3? **3 = 0011** ➔ the complement is **1100**

Arithmetic in 1's Complement

- Advantage: Easy to add and subtract
- Addition: $M + N$, $M \ge 0$, $N \ge 0$
  - M,N: as usual
  - M,$-N$, $M \ge N$: $M+(2^n-1)-N$. The result should be $M-N$, so we need to subtract 2^n and add 1. Example: 5+(-2)
  - M,$-N$, $M < N$: $M+(2^n-1)-N$. The result is $-(N-M)$, which is coded as $(2^n-1)-(N-M)$ - exactly what we got. Example: 2+(-5)
  - $-M$,$-N$: $(2^n-1)-M+(2^n-1)-N$. The result should be $(2^n-1)-(M+N)$, so we need to subtract 2^n and add 1. Example: (-5)+(-2)

➔ Addition is done as usual. In case of a carry from the last bits, ignore it (= subtract 2^n) and add one to the result.

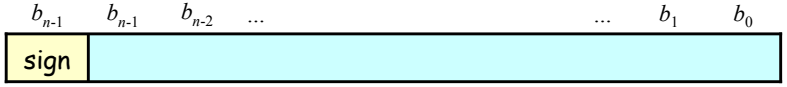
- Subtraction by adding the opposite (=complement): **5-7=5+(-7)**

Representation using 2's Complement

- The 2's complement of an n-bit binary number N: $2^n - N$ if $N \neq 0$, otherwise 0.

➔ 2's complement = 1's complement + 1

- 2's complement of N can be obtained by:
  - Copy all the least significant 0's.
  - Copy the first 1.
  - Complement the remaining bits.
- Example: Write the 2's complement of 1101100.
- Representing signed numbers using 2's complement:



- b_{n-1}=0 implies the coded number is $b_{n-2}...b_0$
- b_{n-1}=1 implies the coded number is the 2's complement of $b_{n-1}...b_0$

Example: Signed Integer Representation

- Assume n=3.

Number	Sign -Mag.	1's Comp.	2's Comp.
+3	011	011	011
+2	010	010	010
+1	001	001	001
+0	000	000	000
–0	100	111	—
–1	101	110	111
–2	110	101	110
–3	111	100	101
–4	—	—	100

- 2's complement has only one representation for zero ➔ can represent one more integer (2^{n-1}).

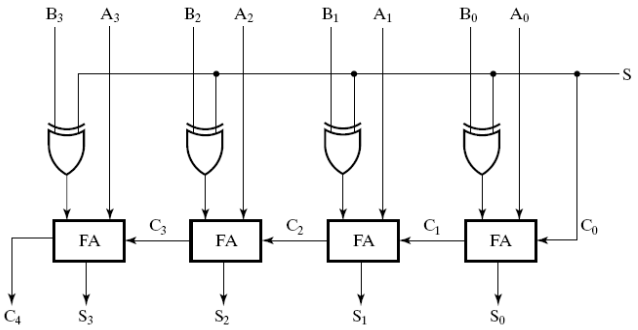
Arithmetic in 2's Complement

- Advantage: Easy to add and subtract
- Addition: $M + N$, $M \geq 0$, $N \geq 0$
  - M,N: as usual
  - M,$–N$, $M \geq N$: $M+2^n–N$. The result should be $M–N$, so we need to subtract 2^n. Example: 5+(-2)
  - M,$–N$, $M < N$: $M+2^n–N$. The result is $–(N–M)$, which is coded as $2^n–(N–M)$ - exactly what we got. Example: 2+(-5)
  - $–M$,$–N$: $(2^n–M)+(2^n–N)$. The result should be $2^n–(M+N)$, so we need to subtract 2^n. Example: (-5)+(-2)

➔ Addition is done as usual. In case of a carry from the last bits, ignore it (= subtract 2^n).

- Subtraction by adding the opposite (=complement): 5-7=5+(-7)

2's Complement Adder/Subtractor



- $A=A_3A_2A_1A_0$ $B=B_3B_2B_1B_0$
- If S=0 computes A+B, if S=1 computes A+(–B)=A–B
- Because of the ease of adding/subtracting signed integers in 2's complement most machines use this representation.

Overflow

- Overflow occurs if $n+1$ bits are required to represent the result from some operation on two n-bit numbers.
- Example: Assume 2's complement representation with $n=4$, then 0101+0011 causes an overflow.
- For addition (and subtraction) overflow can occur only when adding two positive or two negative numbers.
- Assuming the result should fit into an n-bit word, an overflow cannot be resolve. Example:

```
unsigned char a, b=200, c=100;
a = b + c; /* overflow */
```

- The responsibility of the machine is to inform the user/programmer that an overflow has occurred.

Overflow Detection in 2's Complement

- The range of 2's complement representation: $-2^{n-1} \le N \le 2^{n-1} - 1$
- An overflow occurs when:
  - The sum of two positive numbers is $\ge 2^{n-1}$; or
  - The sum of two negative numbers is less than -2^{n-1}.
- Examples: $n = 8$.

Carries:	0 1	Carries:	1 0
+ 70	0 1000110	- 70	1 0111010
+ 80	0 1010000	- 80	1 0110000
+150	1 0010110	-150	0 1101010

- An overflow occurs if the last two carries are different ➔ can be detected using a XOR gate.