COS10004 Computer Systems

Lecture 5.2 Number Systems: Signed numbers

CRICOS provider 00111D

Dr Chris McCarthy

Number representation

- Computers are digital systems:
  - data stored in the form of 0s and 1s
- Computers commonly perform arithmetic operations on numbers:
  - integers (signed and unsigned) and real numbers
- How we represent numbers with bits profoundly impacts how:
  - Bits are used
  - arithmetic operations are implemented.

Signed numbers

- So far we've been concentrating on usigned (positive only) numbers
- How can we represent signed numbers in a computer ?
- There are a number of schemes for representing signed numbers in binary format.
  - **sign-magnitude representation**
  - **twos-complement representation.**

Sign Magnitude

- Use the most significant bit to represent the sign:
  - 0 is positive
  - 1 is negative
  - Eg. sign magnitude in 8 bits (Big Endian):

0	0	0	0	0	1	1	0	6_{10}
1	0	0	0	0	1	1	0	-6_{10}

Sign-Magnitude Steps

Find the sign magnitude representation of 70_{10}

Step 1: find binary representation using 8 bits

$$70_{10} = 01000110_2$$

Step 2: if the number is a negative number flip left most bit

01000110 (no flipping, since it is +ve)

So: $70_{10} = 01000110_2$ (in 8-bit sign/magnitude form)

SIGN-MAGNITUDE TRADEOFFS

> **Trade-offs:**

+ Simple and intuitive
+ easy to implement
- Reduced value range (we lose a bit !)
- How to represent 0 ?

2's Complement

- We can solve these issues using 2's complement representation
- We can think of 2's complement as shifting the range of possible values so that "0" is in the middle of the range.
  - Value range: $[-2^{N-1}, 2^{N-1} - 1]$, where N is the number of bits
- To do this requires a series of steps which you will need to remember

2's Complement Representation

Find the 2's complement representation of -6_{10}

Step1: find binary representation in 8 bits

$$6_{10} = 00000110_2$$

Step 2: Complement the entire positive number, and then add one

00000110	(complemented)->	11111001
(add one)	->	+ 1
		11111010

So: $-6_{10} = 11111010_2$ *(in 2's complement form, using any of above methods)*

2's Complement Representation

Find the Two's Complement of 72_{10}

Step 1: Find the 8 bit binary representation of the positive value.

$72_{10} = 01001000_2$

Step 2: Since number is positive do nothing.

So: $72_{10} = 01001000_2$ *(in 2's complement form, using any of above methods)*

2's Complement Tradeoffs

\+ Memory efficient (i.e, it retains the full representational capacity of the bits)

\+ Retains property that most significant bit still indicates sign

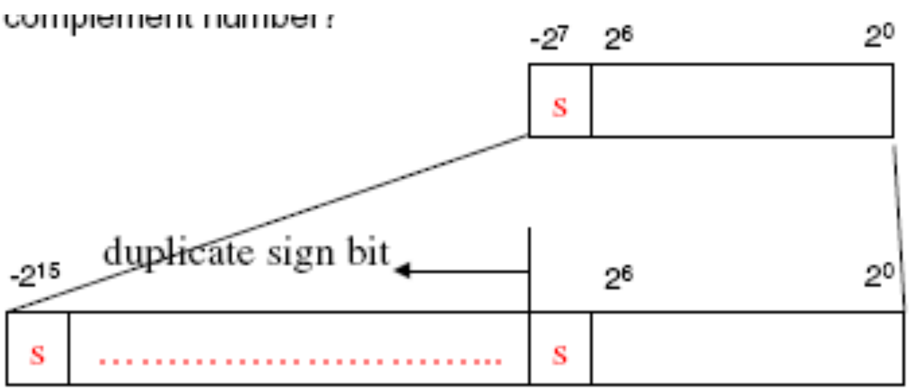
\+ zero represented unambiguously

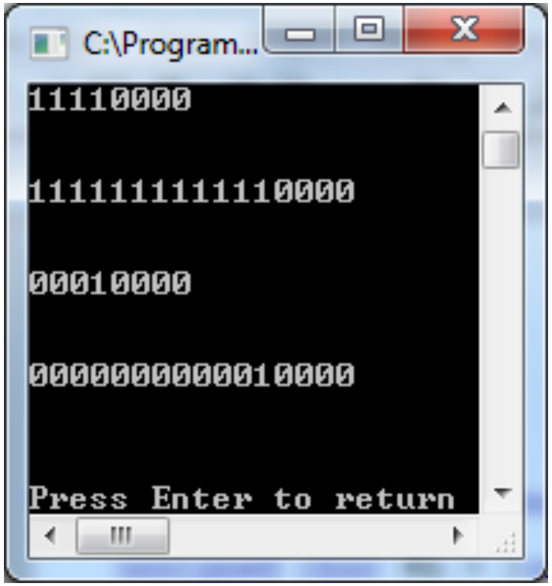
But:

- Slightly more complex transformation (but easily achieved using ALU)

Sign-Extension

- Sometimes we want to represent a value within a larger word size (eg., an 8, 16 or 32 bit word)
- We can extend any signed binary number by repeating the sign bit up to the size needed





EXAMPLE: SIGN EXTENSION (8 TO 16 BIT)

signext.c

- e.g. -16 (stored as 2's compliment):

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	1	1	1	0	0	0	0

becomes:

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0

+16

0	0	0	1	0	0	0	0

becomes:

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

SUMMARY

- Number representation a fundamental design choice of computer systems
- Signed numbers can be represented in different ways:
  - Sign magnitude
  - 2's complement
- We can extend either to larger register sizes using sign extension
- Next Lecture: representing real numbers !