CSARCH Lecture Series: Signed Integer Representation (2's complement)

Sensei RL Uy
College of Computer Studies
De La Salle University
Manila, Philippines

'2110

Copyright Notice

Overview

Reflect on the following questions:

- How are data stored in the memory?
- Given the code below, how are signed integer data stored in the memory?

```
int main()
{
    int var, var1;
    var = 25;
    var1 = -25;
}
```

Overview

- This sub-module introduces the concept of representing signed integer using 2’s complement representation and the concept of using sign extension to extend the bit representation
- The objective is as follows:
  - ✓Describe the process of performing signed integer representation using 2’s complement
  - ✓Describe the process of performing sign-extension

Signed Integer

- In signed integer, the most significant bit is used to represent positive (0) or negative(1)
- Given an *n*-bit binary, if 1 bit is used to represent sign bit then the remaining *n*-1 bit is used to represent the magnitude

Signed Integer

- There are several representations used to represent signed integer:
  - sign-and-magnitude
  - 1's complement (also known as *n*-1's complement)
  - 2's complement (also known as *n*'s complement)

- The standard of representing signed integer is the 2's complement representation which will be the focus of this sub-module

2’ complement

- For positive integer, the 2’s complement is the same as unsigned integer (i.e., positional notation representation)

- For negative integer, perform 2’s complement on the “positive” representation of the negative integer
- 2’s complement is done by:
    - starting from the least significant bit, keep copying the bit while it is 0 until you encounter the first “1” bit. Copy that first “1” bit as well
    - perform 1’s complement (i.e., flip or complement) on the rest of the bit

2's complement

- What is the 2's complement of -15?
  - (Step 1 – positive representation): 01111_2
  - (Step 2a – encounter the first "1" bit, copy): 1_2
  - (Step 2b – perform 1's complement on the rest of the bit): 10001_2
  - Answer: 10001_2

- What is the 2's complement of -12?
  - (Step 1 – positive representation): 01100_2
  - (Step 2a – while zero copy the bit until you encounter the first "1", copy also): 100_2
  - (Step 2b – perform 1's complement on the rest of the bit): 10100_2
  - Answer: 10100_2

2’s complement

- What is the 2’s complement of +15?
  - positive representation: 01111_2
  - Answer: 01111_2

- What is the 2’s complement of +12?
  - positive representation: 01100_2
  - Answer: 01100_2

2’s complement

- 01011 is the 2’s complement representation of which decimal number?

Since the sign bit is 0, it means positive. Use the positional notation to get the magnitude

Answer: 11

- 101010 is the 2’s complement representation of which decimal number?

Since the sign bit is 1, it means negative. Get the 2’s complement of the representation and use the positional notation to get the magnitude

2’s complement `010110`

Answer: -22

2's complement

- Reflection: what is the 8-bit 2's complement of +0 and -0

- What is the 2's complement representation of +0?

  000000000

- What is the 2's complement representation of -0?

  000000000

- Given an *n*-bit binary, the range of the value that can represented for signed integer is from $-(2^{n-1})$ to $+(2^{n-1})-1$

- For example, for an 8-bit binary, the range is from -128 to + 127
  - Positive: 00000000_2 to 01111111_2 (decimal 0 to 127)
  - Negative: 10000000_2 to 11111111_2 (decimal -128 to -1)

Sign-Extension

- Variants of integer data type are usually in the power of 2 (i.e., 8-bit, 16-bit, 32-bit, 64-bit, etc.)
- For most programming languages, the *int* type is 32-bit
- Sign-extension is used to preserve the numeric value when representing a signed integer using more bits
- To sign-extend means to copy the sign bit in the most significant side
- Decimal +5 is represented as 0101_2. To store it to a byte-size integer data type, it will be sign-extended as 00000101_2
- Decimal -5 is represented as 1011_2. To store it to a byte-size integer data type, it will be sign-extended as 11111011_2

Try

label	Address	Memory data (binary)
var1	0008	
var	0000	

```
int main()
{
    int var, var1;
    var = 25;
    var1 = -25;
}
```

Assume that int is 32-bit, how will var and var1 be represented in the memory?

What is the smallest negative and the largest positive value that can be represented in the memory?

Try

label	Address	Memory data (binary)
var1	0008	1111 1111 1111 1111 1111 1111 1110 0111
var	0000	0000 0000 0000 0000 0000 0000 0001 1001

label	address	Memory data (hex)
var1	0008	FFFF FFE7
var	0000	0000 0019

```
int main()
{
    int var, var1;
    var = 25;
    var1 = -25;
}
```

Assume that int is 32-bit, how will var and var1 be represented in the memory?

What is the smallest negative and the largest positive value that can be represented in the memory?

smallest negative is 1000 0000 0000 0000 0000 0000 0000 0000 (-2^{31} or -2,147,483,648 or 0x8000 0000)
largest positive is 0111 1111 1111 1111 1111 1111 1111 1111 ($+2^{31}-1$ or +2,147,483,647 or 0x7FFF FFFF)

Hexadecimal is used as short-hand writing for binary (imagine writing 32 or 64 bits). From this point on, we will use hexadecimal. But, to emphasize, data are represented in the memory as binary.

Try

(8-bit) Signed integer representation	Decimal equivalent
0001 0001	
1111 0000	

To recall …

- What have we learned:
  - ✓Describe the process of performing signed integer representation using 2's complement
  - ✓Describe the process of performing sign-extension