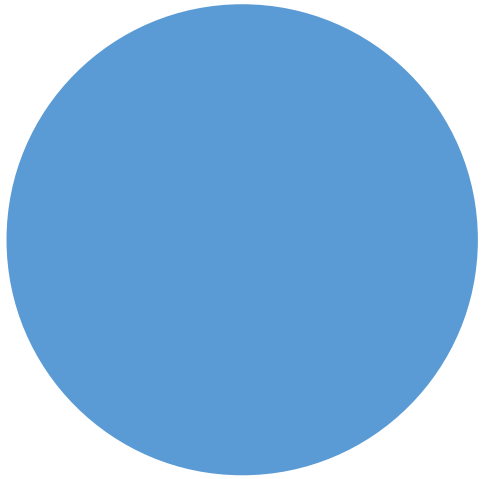
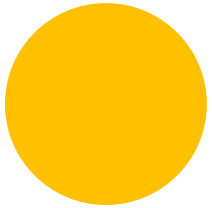


CSARCH Lecture Series: Unsigned Integer Representation

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

Copyright Notice

Overview

Reflect on the following questions:

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

```
int main()
{
    unsigned int var, var1;
    var = 20;
    var1 = 129;
}
```

Overview

- This sub-module introduces the concept of unsigned integer representation and the concept of using zero-extension to extend the bit representation
- The objective are as follows:
  - ✓Describe the process of performing unsigned integer representation
  - ✓Describe the process of performing zero-extension

Integer

- All data are stored in the computer system (memory, secondary storage, etc.) in binary format.
- Integer is a whole number without the fractional part.
- Integer can be classified either as:
  - unsigned integer (positive only)
  - signed integer (positive and negative)

Unsigned Integer

- In unsigned integer, every bit is significant and has a digit value
- Unsigned integer uses the positional number system
  - Example: what is the fewest bits to represent unsigned integer representation of decimal 5?
  - Answer: 3 bits → 101_2

Unsigned Integer

- Given an *n*-bit binary, the range of the value that can represented for unsigned integer is from 0 to 2^n-1
- For example, for an 8-bit binary, the range is from 00000000_2 to 11111111_2 (decimal: 0 to 255)

Zero-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
- Zero-extension is used to preserve the numeric value when representing an unsigned integer using more bits
- To zero-extend means to add 0 to the most significant side
- Decimal 5 is represented as 101_2. To store it to a byte-size integer data type, it will be zero-extended as 00000101_2

Try

label	Address	Memory data (binary)
var1	0008	
var	0000	

```
int main()
{
    unsigned int var, var1;
    var = 20;
    var1 = 129;
}
```

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

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

Try

label	Address	Memory data (binary)
var1	0008	0000 0000 0000 0000 0000 0000 1000 0001
var	0000	0000 0000 0000 0000 0000 0000 0001 0100

```
int main()
{
    unsigned int var, var1;
    var = 20;
    var1 = 129;
}
```

label	address	Memory data (hex)
var1	0008	0000 0081
var	0000	0000 0014

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

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

range is 0 to 2^{32}-1
(binary: 0000 0000 0000 0000 0000 0000 0000 0000 to 1111 1111 1111 1111 1111 1111 1111 1111)
(decimal: 0 to 4,294,967,295) (hexadecimal: 0000 0000 to FFFF 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) Unsigned integer representation	Decimal equivalent
0000 1111	
1111 0000	

To recall …

- What have we learned:
  - ✓Describe the process of performing unsigned integer representation
  - ✓Describe the process of performing zero-extension