

w06-Lec

# Bitwise Operations

Assembled for 204111  
by Kittipitch Kuptavanich

## Bitwise Operations

- เราสามารถนำ Boolean Operator มาใช้กับ **bit vector** ซึ่งก็คือตัวเลข 0 หรือ 1 จำนวน  $w$  ตัว

1000101.....011100101



ความยาว =  $w$

- ให้  $a = [a_{w-1}, a_{w-2}, \dots, a_0]$   
 $b = [b_{w-1}, b_{w-2}, \dots, b_0]$
- ผลลัพธ์ของ  $a$  &  $b$  จะได้เป็น bit vector ที่มีความยาว  $w$  โดยที่ โดยในหลักที่  $i$  จะมีค่าเท่ากับ  $a_i$  &  $b_i$  ( $0 \leq i < w$ )

## Operations of Boolean Algebra

$\sim$	
0	1
1	0

NOT

&	0	1
0	0	0
1	0	1

AND

	0	1
0	0	1
1	1	1

OR

^	0	1
0	0	1
1	1	0

XOR

- Operation  $\sim$  corresponds to the logical operation NOT, denoted by the symbol  $\neg$
- Operation  $\&$  corresponds to the logical operation AND, denoted by the symbol  $\wedge$
- Operation  $|$  corresponds to the logical operation OR, denoted by the symbol  $\vee$
- Operation  $\wedge$  corresponds to the logical operation Exclusive-Or (XOR), denoted by the symbol  $\oplus$  ( $p \wedge q$  มีค่าเป็น True ก็ต่อเมื่อ  $p$  และ  $q$  มีตัวใดตัวหนึ่งเป็น True แต่ไม่ใช่ทั้งสองตัว ซึ่งก็คือ  $0 \wedge 1$  หรือ  $1 \wedge 0$ )

## Bitwise Operations [2]

### ตัวอย่าง

- ให้  $a = [0110]$  และ  $b = [1100]$
- ในการทำ bitwise operation  $a \& b$ ,  $a | b$ ,  $a \wedge b$ , และ  $\sim b$  จะได้ผลลัพธ์ดังนี้

	0110	0110	0110				
&	1100		1100	^	1100	~	1100
	0100		1110		1010		0011

ทำ Operation & ในแต่ละหลักแยกกัน

## Practice 1: Bitwise Operations

- เติมตารางให้สมบูรณ์ เพื่อแสดงค่าที่ได้จากการทำ **Bitwise Operation** โดยมีค่าของ bit vector  $a$  และ  $b$  ดังกำหนด

Operation	Result
$a$	[01101001]
$b$	[01010101]
$\sim a$	_____
$\sim b$	_____
$a \& b$	_____
$a   b$	_____
$a \wedge b$	_____

## Logical Operations in C

- **Logical Operator in C** ประกอบด้วย `||`, `&&` และ `!` แทน Or, And และ Not.
- **Logical operator** พิจารณาให้ค่าใด ๆ ที่ไม่ใช่ 0 เป็น **True** และ 0 เป็น **False** โดยจะมีการ **evaluate** เป็น 1 หรือ 0 เท่านั้น เช่น

Expression	Result
<code>!0x41</code> ← Non-zero = True	<code>0x00</code> ← Evaluates to 0 (False)
<code>!0x00</code>	<code>0x01</code>
<code>!!0x41</code>	<code>0x01</code>
<code>0x69 &amp;&amp; 0x55</code>	<code>0x01</code>
<code>0x69    0x55</code>	<code>0x01</code>

## Bit-Level Operations in C

- ในภาษา C เราสามารถทำ **bitwise Boolean operation** ได้ เช่นเดียวกัน โดยใช้ operator `|`, `&`, `~` และ `^`
- โดย **operation** เหล่านี้สามารถใช้ได้กับ **data type** จำนวนเต็ม (**integral data type**) ทุกชนิดในภาษา C ตัวอย่างเช่น

C expression	Binary expression	Binary result	Hexadecimal result
<code>~0x41</code>	<code>~[0100 0001]</code>	[1011 1110]	0xBE
<code>~0x00</code>	<code>~[0000 0000]</code>	[1111 1111]	0xFF
<code>0x69 &amp; 0x55</code>	<code>[0110 1001] &amp; [0101 0101]</code>	[0100 0001]	0x41
<code>0x69   0x55</code>	<code>[0110 1001]   [0101 0101]</code>	[0111 1101]	0x7D

เลขฐาน 16 ในภาษา C ขึ้นต้นด้วย **0x**

## Practice 2: Logical Operations

- กำหนด  $x$  และ  $y$  มีค่า **byte value** เท่ากับ `0x66` และ `0x39` ตามลำดับ ให้เติม **byte value** ที่เป็นผลลัพธ์จาก **bitwise operation** และ **logical operation** ในตารางด้านล่างให้สมบูรณ์

Expression	Value	Expression	Value
<code>x &amp; y</code>	_____	<code>x &amp;&amp; y</code>	_____
<code>x   y</code>	_____	<code>x    y</code>	_____
<code>~x   ~y</code>	_____	<code>!x    !y</code>	_____
<code>x &amp; !y</code>	_____	<code>x &amp;&amp; ~y</code>	_____

## Shift Operations in C

- ในภาษา C ยังมี shift operation ที่ทำหน้าที่ เลื่อน bit vector ไปทางซ้าย หรือขวา โดยใช้เครื่องหมาย << หรือ >>
- ตัวอย่างเช่น  $x \ll 4$  หมายถึงการเลื่อน bit ใน  $x$  ไปทางซ้าย 4 ตำแหน่ง
- โดยปกติแล้วเมื่อมีการทำ left shift (<<) ตำแหน่งในทางด้านขวาของ bit vector ที่ว่างลง จะถูกแทนด้วยเลขศูนย์

$11001011 \ll 2$  // shift ซ้ายไป 2 ตำแหน่ง  
  
 $00101100$

## Shift Operations in C [2]

- ในกรณีการทำ right shift (>>) จะต่างออกไป โดยปกติแล้ว เครื่องคอมพิวเตอร์ จะสามารถทำ right shift ได้ 2 แบบ คือ
  - Logical Right Shift: ตำแหน่งที่ว่างลงทางซ้ายจะถูกแทนด้วย 0
  - Arithmetic Right Shift: ตำแหน่งที่ว่างลงทางซ้ายจะถูกแทนด้วยค่าของ bit ซ้ายสุดก่อนทำการ shift

Operation	Values	
Argument x	[01100011]	[10010101]
$x \ll 4$	[00110000]	[01010000]
$x \gg 4$ (logical)	[00000110]	[00001001]
$x \gg 4$ (arithmetic)	[00000110]	[1111001]

**Note:**  
bit ที่ถูกขีดเส้นใต้ คือ bit ที่ถูกใส่แทนลงไปในพื้นที่ว่างลง ในการทำ left และ right shift operation

## Shift Operations in C [2]

- ในกรณีการทำ right shift (>>) จะต่างออกไป โดยปกติแล้ว เครื่องคอมพิวเตอร์ จะสามารถทำ right shift ได้ 2 แบบ คือ
  - Logical Right Shift: ตำแหน่งที่ว่างลงทางซ้ายจะถูกแทนด้วย 0

Operation	Values	
Argument x	[01100011]	[10010101]
$x \ll 4$	[00110000]	[01010000]
$x \gg 4$ (logical)	[00000110]	[00001001]

**Note:**  
bit ที่ถูกขีดเส้นใต้ คือ bit ที่ถูกใส่แทนลงไปในพื้นที่ว่างลง ในการทำ left และ right shift operation

## Practice 3: Shift Operations

- ให้เปลี่ยนเลขฐาน 16 ที่กำหนดให้ใน column ซ้ายสุดเป็นเลขฐาน 2 แล้วทำ shift operation ตามที่ระบุ จากนั้นให้เปลี่ยนผลลัพธ์ที่ได้ จากเลขฐาน 2 กลับเป็นเลขฐาน 16

x	x << 3		(Logical) x >> 2		(Arithmetic) x >> 2	
	Binary	Hex	Binary	Hex	Binary	Hex
0xC3	_____	_____	_____	_____	_____	_____
0x75	_____	_____	_____	_____	_____	_____
0x87	_____	_____	_____	_____	_____	_____
0x66	_____	_____	_____	_____	_____	_____

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15