w09-Lec2

Bitwise Operations

Assembled for 204111

by Kittipitch Kuptavanich

Operations of Boolean Algebra

~	
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 ~ corresponds to the logical operation *NOT*, denoted by the symbol ¬**
- **Operation & corresponds to the logical operation *AND*, denoted by the symbol ∧**
- **Operation | corresponds to the logical operation *OR*, denoted by the symbol ∨**
- **Operation ^ corresponds to the logical operation Exclusive-Or (*XOR*), denoted by the symbol ⊕ (p ^ q มีค่าเป็น True ก็ต่อเมื่อ p และ q มีตัวใดตัวหนึ่งเป็น True แต่ไม่ใช่ทั้งสองตัว ซึ่งก็คือ 0 ^ 1 หรือ 1 ^ 0)**

Bitwise Operations

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

$$\underbrace{1000101\ldots\ldots011100101}_{\text{ความยาว} = w}$$

- ให้ $a = [a_{w-1}, a_{w-2}, \ldots, a_0]$

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

Bitwise Operations [2]

ตัวอย่าง

- ให้ a = [0110] และ b = [1100]
- ในการทำ **bitwise operation** a & b, a | b, a ^ b, และ ~b

จะได้ผลลัพธ์ดังนี้

$$\begin{array}{r} 0110 \\ \& \ 1100 \\ \hline 0100 \end{array} \qquad \begin{array}{r} 0110 \\ | \ 1100 \\ \hline 1110 \end{array} \qquad \begin{array}{r} 0110 \\ \wedge \ 1100 \\ \hline 1010 \end{array} \qquad \begin{array}{r} \sim \ 1100 \\ \hline 0011 \end{array}$$

ทำ Operation & ในแต่ละหลักแยกกัน

Practice 1: Bitwise Operations

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

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

Python bitwise ~ Operator

- Python ใช้การแทนค่าตัวเลขแบบ two's complement
  - จำนวนเต็มบวกใด ๆ มีค่า MSB เป็น 0
  - จำนวนเต็มลบใด ๆ มีค่า MSB เป็น 1
- พิจารณาค่า 3 (11) ในภาษา Python
  - การแทนค่าจึงอยู่ในรูป [011] เพิ่ม MSB = 0
- ดังนั้น ~3 ใน Python จึงมีค่า [100]
  - ตีความแบบ two's complement ได้เป็น -4

```
>>> ~3
-4
```


Python bitwise ~ Operator [2]

- เช่นเดียวกันในกรณีจำนวนลบ เช่น -35
  - เขียนแบบ two's complement
    - 35 = [0100011] ดังนั้น
    - -35 = [1011101] กรณีนี้ไม่ต้องเพิ่ม MSB
  - ~-35 จึงมีค่า [0100010] = 34

```
>>> ~-35
34
>>> int('0100010', 2)
34
```

- ค่า ~x จะมีค่าเท่ากับ -x – 1 ทั้งจำนวน + - และ 0

> - สังเกตการใช้ฟังก์ชัน int() เพื่อแปลงเลขฐาน 2 เป็น integer
> - เลข 2 เป็น optional parameter (default คือ ฐาน 10)

Bit-Level Operations in Python

- ใน Python เราสามารถทำ bitwise Boolean operation ได้เช่นเดียวกัน โดยใช้ operator |, &, ~ และ ^
- โดย operation เหล่านี้สามารถใช้ได้กับ data type จำนวนเต็ม ตัวอย่างเช่น

Python expression	Binary expression	Binary result	2's Complement Value
~0x41	~[0100 0001]	[1011 1110]	_____
~0x00	~[0000 0000]	[1111 1111]	_____
0x69 & 0x55	[0110 1001] & [0101 0101]	[0100 0001]	_____
0x69 | 0x55	[0110 1001] | [0101 0101]	[0111 1101]	_____

> เลขฐาน 16 ใน Python ขึ้นต้นด้วย 0x

Shift Operations in Python

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

```
11001011 << 2          // shift ซ้ายไป 2 ตำแหน่ง
```

```
00101100
```


Shift Operations in Python [2]

- ในกรณีการทำ right shift (>>) จะต่างออกไปโดยจะเป็นการทำ Arithmetic Right Shift
  - Arithmetic Right Shift: ตำแหน่งที่ว่างลงทางซ้ายจะถูกแทนด้วยค่าของ bit ซ้ายสุดก่อนทำการ shift เพื่อคงเครื่องหมาย + หรือ - ไว้
- ค่าที่ได้จากการทำ left shift (x << *n*) จะเท่ากับ x * _____
- ค่าที่ได้จากการทำ right shift (x >> *n*) จะเท่ากับ x // _____

Operation	Values	
Argument x	[01100011]	[10010101]
x << 4	[0011*0000*]	[0101*0000*]
x >> 4 (arithmetic)	[*0000*0110]	[*1111*1001]

Practice 3: Shift Operations

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

x		x << 3		(Arithmetic) x >> 2	
Hex	Binary	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