CME 112- Programming Languages II

Week 12
Bitwise Operators

Assist. Prof. Dr. Caner Özcan

I'm human, so I'm free. if I'm free, I must be responsible.
SARTRE

Binary Number System

- Binary number system uses 0 or 1 for each digit.
- For computer systems everything is coded in binary.

$$(d_4 d_3 d_2 d_1 d_0)_2 = (d_0 . 2^0) + (d_1 . 2^1) + (d_2 . 2^2) + (d_3 . 2^3) + (d_4 . 2^4)$$

$$(10011)_2 = (1 . 2^0) + (1 . 2^1) + (0 . 2^2) + (0 . 2^3) + (1 . 2^4) = 19$$

Hexadecimal Number System

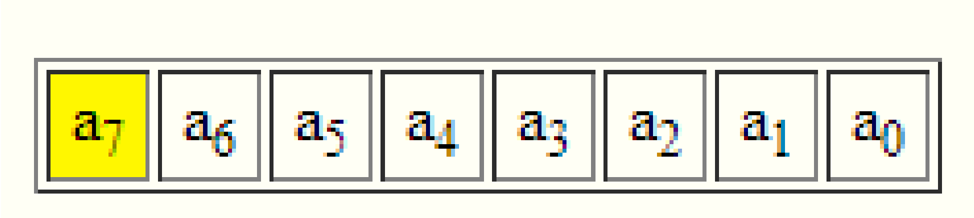
- Hexadecimal number system has 16 different symbol.

Decimal	:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

$(3FC)_{16} = (3 . 16^2) + (F . 16^1) + (C . 16^0) = 768 + 240 + 12 = 1020$
$(1FA9)_{16} = (1 . 16^3) + (F . 16^2) + (A . 16^1) + (9 . 16^0) = 4096 + 3840 + 160 + 9 = 8105$
$(75)_{16} = (7 . 16^1) + (5 . 16^0) = 112 + 5 = 117$

Signed Numbers in Binary System

- Variables in C can be signed or unsigned.
- Think of a 8 bits (1 byte) number.



- If the number is negative then highest level bit (7th bit in this sample) is considered as sign bit.
- If the sign bit is 1 then number is negative, otherwise number is positive.

Signed Numbers in Binary System

- Decimal equivalent of a signed binary number can be found with:

$(a_7a_6a_5a_4a_3a_2a_1a_0)_2 = (a_7 . -2^7) + (a_6 . 2^6) + ... + (a_1 . 2^1) + (a_0 . 2^0)$

- $(1011\ 1011)_2$ = -69 (If the number is signed)
  $(1011\ 1011)_2$ = 187 (If the number is unsigned)

- $(1100\ 1101)_2$ = -51 (If the number is signed)
  $(1100\ 1101)_2$ = 205 (If the number is unsigned)

- $(0110\ 1101)_2$ = 109 (If the number is signed se)
  $(0110\ 1101)_2$ = 109 (If the number is unsigned)

Bitwise Operators

- Operations on bits at individual levels can be carried out using Bitwise operations in C.
- Bits come together to form a byte which is the lowest form of data that can be accessed in digital hardware.
- The whole representation of a number is considered while applying a bitwise operator.
- Each bit can have the value 0 or the value 1.

Bitwise Operators

Sembol	Operator
&	Bitwise AND
|	Bitwise Inclusive OR
^	Bitwise Exclusive OR
<<	Sola kaydır
>>	Sağa kaydır
~	Bire tümleyen

Bitwise AND (&)

- The bitwise AND operator is a single ampersand: &.
- It is just a representation of AND and does its work on bits and not on bytes, chars, integers, etc.
- So basically a binary AND does the logical AND of the bits in each position of a number in its binary form.
- 11001110 & 10011000 = 10001000
- 5 & 3 = 1 (101 & 011 = 001)

Bitwise OR (|)

- Bitwise OR works in the same way as bitwise AND.
- Its result is a 1 if one of the either bits is 1 and zero only when both bits are 0.
- Its symbol is '|' which can be called a pipe.
- 11001110 | 10011000 = 11011110
- 5 | 3 = 7 (101 | 011 = 111)

Bitwise Exclusive OR (^)

- The Bitwise EX-OR performs a logical EX-OR function or in simple term adds the two bits discarding the carry.
- Thus result is zero only when we have 2 zeroes or 2 ones to perform on.
- Sometimes EX-OR might just be used to toggle the bits between 1 and 0.
- Thus: i = i ^ 1 when used in a loop toggles its values between 1 and 0.
- 5 ^ 3 = 6 (101 ^ 011 = 110)

Bitwise Operators

bit a	bit b	a & b (a AND b)	a | b (a OR b)	a ^ b (a XOR b)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Right Shift (> >)

- The symbol of right shift operator is >>.
- For its operation, it requires two operands.
- It shifts each bit in its left operand to the right. The number following the operator decides the number of places the bits are shifted (i.e. the right operand).
- Thus by doing **number >> 3** all the bits will be shifted to the right by three places and so on.
- Blank spaces generated on the left most bits are filled up by zeroes
- Right shift can be used to divide a bit pattern by 2 as shown:

  10 >> 1 = 5 (1010) >> 1 = (0101)

Right Shift (> >)

- If the number is signed, then sign extension is done in right shift operation.
- Sign extension puts the highest bit's value of the number into the blank spaces on the left most bits generated.

```
10000000000000001000000110000000
11110000000000000001000000110000
```

- In this sample, as the original number's highest bit is 1, new genarated bits are also 1 after right shift.

Left Shift (<<)

- The symbol of left shift operator is <<.
- It shifts each bit in its left operand to the left. It works opposite to that of right shift operator.
- Blank spaces which is generated on the right most bits are filled up by zeroes
- Left shift can be used to multiply an integer in multiples of 2 as in:

5 << 1 = 10   (101) << 1 = (1010)

Unary Operator ~ One's Complement

- The one's complement (~) or the bitwise complement gets us the complement of a given number.
- Thus we get the bits inverted, for every bit 1 the result is bit 0 and conversely for every bit 0 we have a bit 1.
- ~ 5 = 2 (~ 101 = 010)

Usage of Bitwise Operators

- It is better to know how bitwise operations take place while we write programs.
- OR operator is the union of bits of two numbers having the value 1.

```
  10101010101010101010101010101010
  01010101010101010101010101010101
| --------------------------------
  11111111111111111111111111111111
```

Usage of Bitwise Operators

- AND operator is intersection of bits of two numbers having the value 1.

```
   10101010101010101010101010101010
   01010101010101010101010101010101
&  --------------------------------
   00000000000000000000000000000000
```

- In this sample there is no bits both have 1. So the intersection of all bits are 0.

Usage of Bitwise Operators

- OR operator can be used to make a number's bits 1.

Before : 00000000111111110000000011111111
Bits to be 1 : 00010000000000000000100000000000
After : 00010000111111110001000011111111

- AND operator can be used to check if a bit is 1 or not.

00000111010110111100110100010101
00000000000000010000000000000000→ Maske

Example: Keyboard Codes

- When the data which shows the states of keys information read from memory, the meaning of every bit is :

Bit	State
0	Right shift pressed/not
1	Left shift pressed/not
2	Ctrl pressed/not
3	Alt pressed/not
4	Scroll on/off
5	Num Lock on/off
6	Caps Lock On/off

Example: Keyboard Codes

- For checking whether numlock is on or off, we need to check bit number 5 of the key information data x.
- For this purpose we can perform binary AND operation with x and 32 operands.
- For example, if the key information data is 01101011, then we can use (00100000=32) to check is bit number 5 is 1 or 0.

  01101011 &

  00100000 → Mask
- As the bit number 5 is 1 in key information data the result is 32, otherwise result would be 0.

Example: Ipv4 Address

- IPv4 adresses are stored in network packages in 32 bit form.
- Each 8 bits correspond to a segment of ip number which is separated by point.
- For example: 192.168.1.2 is 0xc0a80102 in hexadecimal format.
- Lets write a program that reads 32 bits IPv4 adress and writes each segment separated with points.

Example: Ipv4 Address

- For this we need to take each 8 bits from 32 bit IPv4 adress using & bitwise operator with a suitable mask.
- For example if we want to take lowest 8 bits we have to use a mask 0x000000ff which will preserve the lowest 8 bits of the data.

Example: Ipv4 Address

- If the preserved bits is not the lowest 8 we have to right shift the obtained number to the lowest 8 bit.

Value : 11000000101010000000000100000010 c0a80102 3232235778
Mask : 11111111000000000000000000000000 ff000000 4278190080
Result : 11000000000000000000000000000000 c0000000 3221225472

- The result we get here is 3221225472 and not 192 as we expected.
- The reason is that the obtained number is not in the lowest 8 bit. We need to shift the number 24 times to the right. (>> 24)

Value : 11000000101010000000000100000010 c0a80102 3232235778
Mask : 11111111000000000000000000000000 ff000000 4278190080
Result : 00000000000000000000000011000000 c0000000 192

Example: Ipv4 Address


```c
#include <stdio.h>
int main(void)
{
    unsigned int ipAdres = 0xc0a80102;
    unsigned maske =0xff000000;
    int segment1,segment2,segment3,segment4;
    int i, bit=32;
    unsigned tmp;
    for(i=1;i<=4;i++)
    {
        tmp = ipAdres & maske;
        if(i!=4){
            maske = maske >> 8;
            tmp = tmp >> (bit-i*8);
            printf("%d.",tmp);
        }
        else printf("%d",tmp);
    }

    getchar();
    return 0;
}
```

Example: Binary Addition

```c
#include <stdio.h>
#include <stdlib.h>

//binary addition
int main()
{
    unsigned int x=3, y=1, sum, carry;
    sum = x ^ y;
    carry = x & y;
    while(carry!=0)
    {
        carry = carry << 1;
        x = sum;
        y = carry;
        sum = x ^ y;
        carry = x & y;
    }
    printf("%d", sum);
    getchar();
    return 0;
}
```

References

- Doç. Dr. Fahri Vatansever, “Algoritma Geliştirme ve Programlamaya Giriş”, Seçkin Yayıncılık, 12. Baskı, 2015.
- Kaan Aslan, “A’dan Z’ye C Klavuzu 8. Basım”, Pusula Yayıncılık, 2002.
- Paul J. Deitel, “C How to Program”, Harvey Deitel.
- “A book on C”, All Kelley, İra Pohl

Any

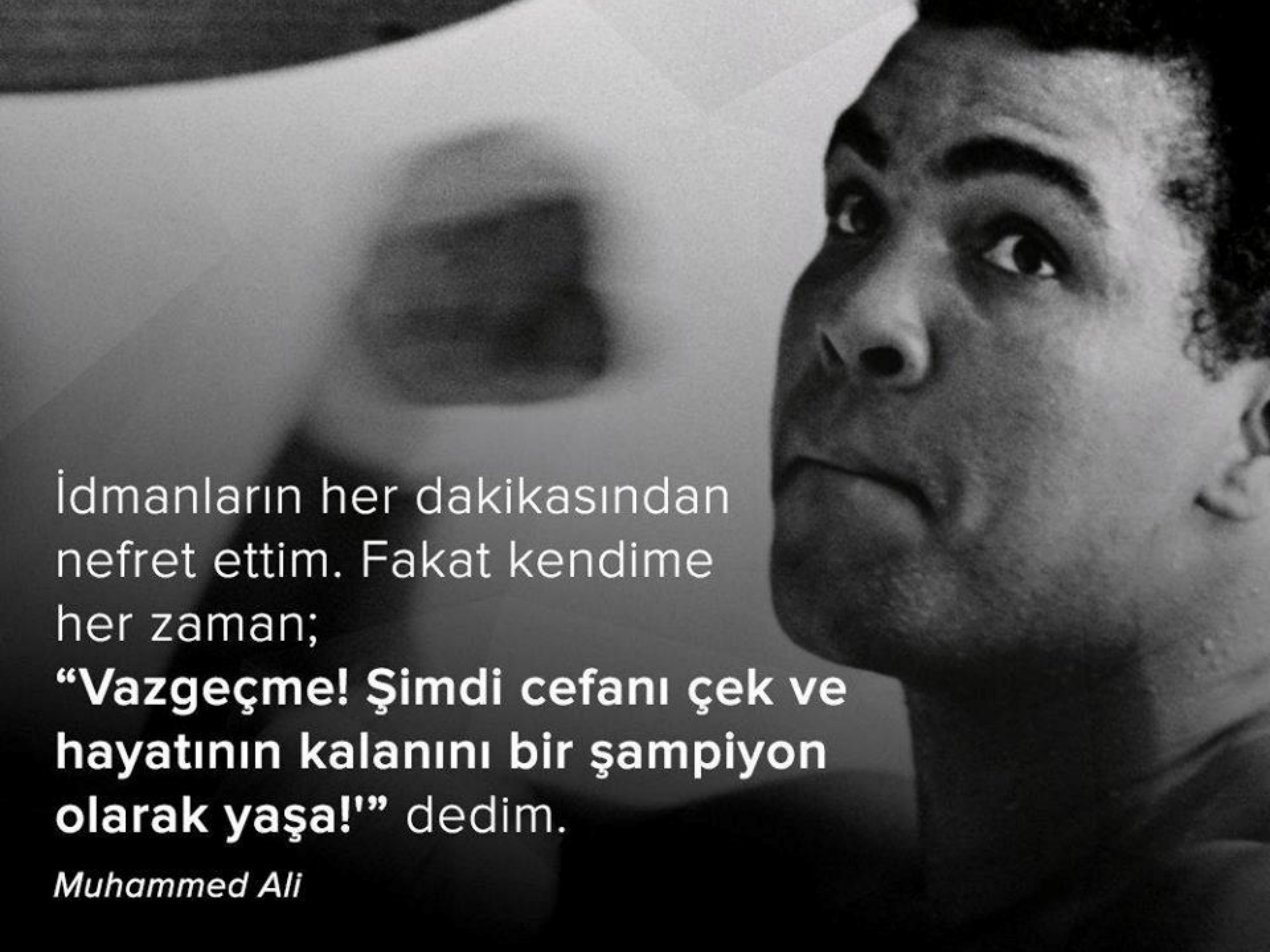
Questions

?

Thanks for listening

CANER ÖZCAN

canerozcan@karabuk.edu.tr


İdmanların her dakikasından nefret ettim. Fakat kendime her zaman;
“Vazgeçme! Şimdi cefanı çek ve hayatının kalanını bir şampiyon olarak yaşa!'” dedim.
Muhammed Ali