# C Programming for Engineers

## Bit Manipulation

ICEN 360– Spring 2017
Prof. Dola Saha

# Bitwise Operation

➢ Computers represent all data internally as sequences of bits.

➢ Each bit can assume the value 0 or the value 1.

➢ The bitwise operators are used to manipulate the bits of integral operands both signed and unsigned.

➢ Unsigned integers are normally used with the bitwise operators.

➢ Bitwise manipulations are machine dependent.

# Bitwise Operator

| Operator | | Description |
|---|---|---|
| & | bitwise AND | Compares its two operands bit by bit. The bits in the result are set to 1 if the corresponding bits in the two operands are *both* 1. |
| \| | bitwise inclusive OR | Compares its two operands bit by bit. The bits in the result are set to 1 if *at least one* of the corresponding bits in the two operands is 1. |
| ^ | bitwise exclusive OR (also known as bitwise XOR) | Compares its two operands bit by bit. The bits in the result are set to 1 if the corresponding bits in the two operands are different. |
| << | left shift | Shifts the bits of the first operand left by the number of bits specified by the second operand; fill from the right with 0 bits. |
| >> | right shift | Shifts the bits of the first operand right by the number of bits specified by the second operand; the method of filling from the left is machine dependent when the left operand is negative. |
| ~ | complement | All 0 bits are set to 1 and all 1 bits are set to 0. |

# Bitwise Operation Example

| | |
|---|---|
| n=13 | 00001101 |
| ~n | 11110010 |
| n << 1 | Lost 0 → 00001101 → 00011010 ← 0 Inserted |
| n << 3 | Lost three 0s → 00001101 → 01101000 ← Inserted three 0's |
| n >> 3 | Inserted three 0s → 00001101 → 00000001 ← 101 Lost |

# Bitwise Operation Example

| Bit 1 | Bit 2 | Bit 1 & Bit 2 |
|-------|-------|---------------|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| Bit 1 | Bit 2 | Bit 1 \| Bit 2 |
|-------|-------|---------------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| Bit 1 | Bit 2 | Bit 1 ^ Bit 2 |
|-------|-------|---------------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

n    00001101
m    01010101
n&m   00000101

n    00001101
m    01010101
n|m   01011101

n    00001101
m    01010101
n^m   01011000

# Display Bits Example (1)

```c
1   // Fig. 10.7: fig10_07.c
2   // Displaying an unsigned int in bits
3   #include <stdio.h>
4
5   void displayBits(unsigned int value); // prototype
6
7   int main(void)
8   {
9      unsigned int x; // variable to hold user input
10
11     printf("%s", "Enter a nonnegative int: ");
12     scanf("%u", &x);
13
14     displayBits(x);
15  }
16
```

# Display Bits Example (2)

```
17    // display bits of an unsigned int value
18    void displayBits(unsigned int value)
19    {
20        // define displayMask and left shift 31 bits
21        unsigned int displayMask = 1 << 31;
22
23        printf("%10u = ", value);
24
25        // loop through bits
26        for (unsigned int c = 1; c <= 32; ++c) {
27            putchar(value & displayMask ? '1' : '0');
28            value <<= 1; // shift value left by 1
29
30            if (c % 8 == 0) { // output space after 8 bits
31                putchar(' ');
32            }
33        }
34
35        putchar('\n');
36    }
```

```
Enter a nonnegative int: 65000
     65000 = 00000000 00000000 11111101 11101000
```

# Bitwise Operation Example Code (1)

```c
 1   // Fig. 10.9: fig10_09.c
 2   // Using the bitwise AND, bitwise inclusive OR, bitwise
 3   // exclusive OR and bitwise complement operators
 4   #include <stdio.h>
 5
 6   void displayBits(unsigned int value); // prototype
 7
 8   int main(void)
 9   {
10      // demonstrate bitwise AND (&)
11      unsigned int number1 = 65535;
12      unsigned int mask = 1;
13      puts("The result of combining the following");
14      displayBits(number1);
15      displayBits(mask);
16      puts("using the bitwise AND operator & is");
17      displayBits(number1 & mask);
18
```

# Bitwise Operation Example Code (2)

```
19      // demonstrate bitwise inclusive OR (|)
20      number1 = 15;
21      unsigned int setBits = 241;
22      puts("\nThe result of combining the following");
23      displayBits(number1);
24      displayBits(setBits);
25      puts("using the bitwise inclusive OR operator | is");
26      displayBits(number1 | setBits);
27
28      // demonstrate bitwise exclusive OR (^)
29      number1 = 139;
30      unsigned int number2 = 199;
31      puts("\nThe result of combining the following");
32      displayBits(number1);
33      displayBits(number2);
34      puts("using the bitwise exclusive OR operator ^ is");
35      displayBits(number1 ^ number2);
36
```

# Bitwise Operation Example Code (3)

```
37      // demonstrate bitwise complement (~)
38      number1 = 21845;
39      puts("\nThe one's complement of");
40      displayBits(number1);
41      puts("is");
42      displayBits(~number1);
43   }
44
```

# Bitwise Operation Example Code (4)

```c
45   // display bits of an unsigned int value
46   void displayBits(unsigned int value)
47   {
48       // declare displayMask and left shift 31 bits
49       unsigned int displayMask = 1 << 31;
50
51       printf("%10u = ", value);
52
53       // loop through bits
54       for (unsigned int c = 1; c <= 32; ++c) {
55           putchar(value & displayMask ? '1' : '0');
56           value <<= 1; // shift value left by 1
57
58           if (c % 8 == 0) { // output a space after 8 bits
59               putchar(' ');
60           }
61       }
62
63       putchar('\n');
64   }
```

# Bitwise Operation Example Code Output

```
The result of combining the following
     65535 = 00000000 00000000 11111111 11111111
         1 = 00000000 00000000 00000000 00000001
using the bitwise AND operator & is
         1 = 00000000 00000000 00000000 00000001

The result of combining the following
        15 = 00000000 00000000 00000000 00001111
       241 = 00000000 00000000 00000000 11110001
using the bitwise inclusive OR operator | is
       255 = 00000000 00000000 00000000 11111111

The result of combining the following
       139 = 00000000 00000000 00000000 10001011
       199 = 00000000 00000000 00000000 11000111
using the bitwise exclusive OR operator ^ is
        76 = 00000000 00000000 00000000 01001100

The one's complement of
     21845 = 00000000 00000000 01010101 01010101
is
4294945450 = 11111111 11111111 10101010 10101010
```
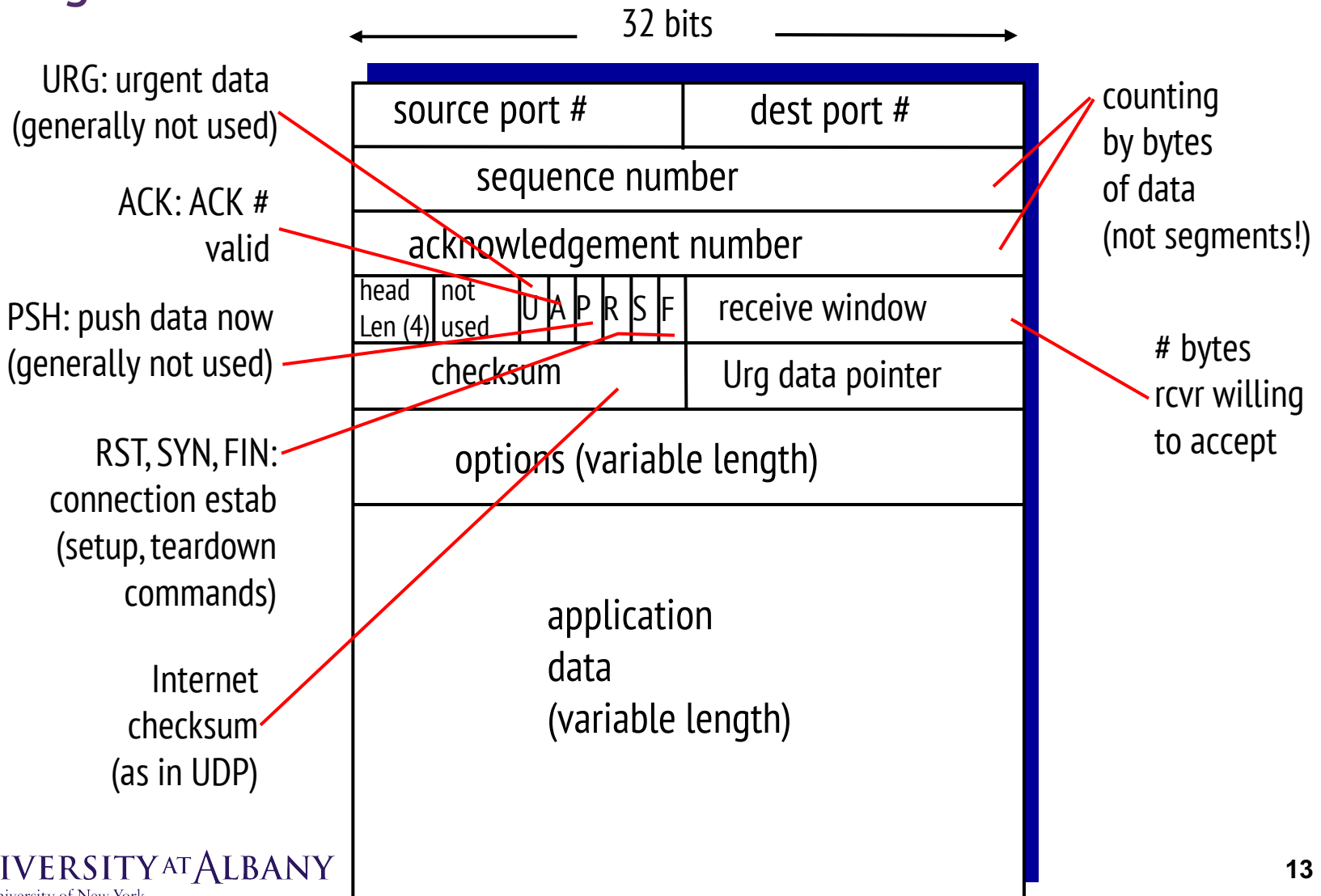
# Bitwise Operation Application:

## TCP segment structure



32 bits

URG: urgent data (generally not used)

ACK: ACK # valid

PSH: push data now (generally not used)

RST, SYN, FIN: connection estab (setup, teardown commands)

Internet checksum (as in UDP)

| source port # | dest port # |
|---|---|
| sequence number | |
| acknowledgement number | |
| head Len (4) | not used | U A P R S F | receive window |
| checksum | Urg data pointer |
| options (variable length) | |
| application data (variable length) | |

counting by bytes of data (not segments!)

# bytes rcvr willing to accept

UNIVERSITY AT ALBANY
State University of New York

# Multiply and Divide by Bitwise Operation

➢ Left Shift

  ▪ Multiply

➢ Right Shift

  ▪ Divide

UNIVERSITY AT ALBANY
State University of New York

# Revisiting Classwork Assignment

➢ Write a program to generate data for N students. Use structure to create numeric ID and points (max 100) as 2 separate members of the structure. Randomly generate data for N students. Display both the ID and the points of the student who has received highest point. Write three separate functions to complete the program:

1. `generateStudentData(),` input is array of students
2. `printStudentInfo(),` input is a single student
3. `getTopStudent(),` input is array of students and output is a single student