Agenda

Bitwise operators

Bit shifting

Working with bitfields and masks

Ex: 4 & 8
Ex: 4 | 8
Ex: ~4

Bitwise operations: AND, OR, XOR, NOT

A	B	A&B

A	B	A^B

A	B	A|B

A	~A

Exercise: Compute a & b

Give your final result as a hexadecimal and decimal number

Suppose a = 10 and b = 6.

Exercise: Compute a | b

Give your final result as a hexadecimal and decimal number

Suppose a = 10 and b = 6.

Exercise: Compute a ^ b

Give your final result as a hexadecimal and decimal number

Suppose a = 10 and b = 6.

Exercise: Compute ~a

Give your final result as a hexadecimal and decimal number

Suppose a = 10.

Bitops: Code example

```
#include <stdio.h>

int main() {
  unsigned char a = 10;
  unsigned char b = 6;
  unsigned char not = ~a;

  printf("%02X & %02X = %02X\n", a, b, a & b);
  printf("%02X | %02X = %02X\n", a, b, a | b);
  printf("%02X ^ %02X = %02X\n", a, b, a ^ b);
  printf("~%02X = %02X\n", a, not);
  printf("%lu\n", sizeof(unsigned char));

  return 0;
}
```

Exercise: What are the values of result1 and result2?

```
int value1 = 0;
 printf("Enter value 1: ");
 scanf(" %d", &value1);

 int value2 = 0;
 printf("Enter value 2: ");
 scanf(" %d", &value2);

 unsigned int mask = 0x80000000;
 int result1 = mask & value1;
 printf("Mask: %08X Value: %08X Result: %08X\n", mask, value1, result1);

 int result2 = mask & value2;
 printf("Mask: %08X Value: %08X Result: %08X\n", mask, value2, result2);

 if (result1 == result2) printf("Your values have the same sign!\n");
 else printf("Your values have different signs!\n");
```

Exercise: Modify the previous program to test if two signed numbers have opposite signs

```
int value1 = 0;
 printf("Enter value 1: ");
 scanf(" %d", &value1);

 int value2 = 0;
 printf("Enter value 2: ");
 scanf(" %d", &value2);

 unsigned int mask = 0x80000000;
 int result1 = mask & value1;
 printf("Mask: %08X Value: %08X Result: %08X\n", mask, value1, result1);

 int result2 = mask & value2;
 printf("Mask: %08X Value: %08X Result: %08X\n", mask, value2, result2);

 if (result1 == result2) printf("Your values have the same sign!\n");
 else printf("Your values have different signs!\n");
```

Exercise: Use bit ops to test even/odd

Bit shifting

<< shifts bits to the left

>> shifts bits to the right

Example:

```
unsigned int right = 0xAA00;
unsigned int rightShift = left >> 8;
```

Logical VS arithmetic shifts

Logical shift right

- Fill in with zeros
- Unsigned int behavior

Arithmetic shift right

- Fill in with either 0 or 1 based on the sign bit
    - E.g. prepend with leftmost bit !
- Signed int behavior

Shift left always fills in zero

Bit shift: example

```
#include <stdio.h>

int main(int argc, char **argv) {
   /* Unsigned integer value: u_val. */
   unsigned int u_val = 0xFF000000;

   /* Signed integer value: s_val. */
   int s_val = 0xFF000000;

   printf("%08X\n", u_val >> 12);  // logical right shift
   printf("%08X\n", s_val >> 12);  // arithmetic right shift

   return 0;
}
```

Demo: Swapping bytes

Suppose a = 0xAABB. Write a program that swaps the lowest 2 bytes.

```
unsigned int a = 0xAABB;
 unsigned int leftMask = 0xFF00;
 unsigned int rightMask = 0x00FF;
 unsigned int left = (leftMask & a) ;
 unsigned int right = (rightMask & a) ;
 unsigned int leftShift = left >> 8;
 unsigned int rightShift = right << 8;
 unsigned flipped = leftShift | rightShift;

 printf("Left: %08X Right: %08X\n", left, right);
 printf("Left: %08X Right: %08X\n", leftShift, rightShift);
 printf("Before: %08X After: %08X\n", a, flipped);
```

Exercise: Swapping bytes

Variable	Bits 32-24	Bits 23-16	Bits 15-8	Bits 7-0
a	00	00	AA	BB
leftMask				
rightMask				
Left				
Right				
leftShift				
rightShift				
flipped				

Bit fields and masking

A **bit field** stores a set of booleans within a single value

A **flag** is a value that can be true or false

Example: Suppose we store 8 flags within a character data type

```
unsigned char c = 0x45; // binary is 0100 0101
                        // all flags are set to zero except for 3
```

Example: Dinner options

```
#define WATER 0x01 // mask that corresponds to the least significant bit
#define WINE 0x02 //mask corresponds to the second least significant bit
#define DINNER_ROLL
#define SALAD
#define SOUP
#define MAIN
#define DESSERT
#define COFFEE

// Fill in the remaining masks
```

Example: Dinner options

```
// In code, we would write
unsigned char options = WATER|SALAD|DESSERT;
```

What is the bit field corresponding to each of the following options?

Options	Bit Field
WATER, WINE, SALAD, MAIN, COFFEE	
WATER, SALAD, SOUP, MAIN, DESSERT	
WINE, MAIN, DESSERT, COFFEE	
WATER, SALAD, DESSERT	

Example: Dinner options

What options correspond to the following bit fields?

Bit Field	Options
0xAE	
0x31	
0x9B	
0x74	

Example: Dinner options

```
// Set someone’s dinner preferences to WATER, WINE, DESSERT
unsigned char options = WATER | WINE | DESSERT;

// Test whether someone choose the MAIN option.
if (options & MAIN) printf("You choose the MAIN\n");

// Test whether someone choose the WINE and DESSERT options.
if ((options & WINE) & (options & DESSERT))
{
  printf("You choose the WINE and DESSERT\n");
}
```

Example: File permissions use bit fields

```
alinen@sutekh:~/cs355/os-devel$ ls -l
-rw-r--r--  1 alinen alinen   70 Dec 10  2023 README.md
```

What does the above file permissions represent?

What numeric value corresponds to the following permissions (in octal)? –rw-r--r--

What numeric value corresponds to the following permissions (in octal)? –rwxr-x--x