CSCE 4114
bit twiddling (bashing) in C

David Andrews

dandrews@uark.edu

Bit Twiddling in C

- Lecture will not cover common programming constructs and abstract data structures. The semantics are the same as you learned in PFI and PFII.
- If you are not familiar with C or C++ syntax many good general reviews are available online.
- We will cover the bit manipulation operations that are common and convenient for programming embedded systems.

Bit Twiddling in C

- C was developed for writing operating systems, it is close to the machine.
  - Which OS was it created for and used to write ?
- Can Address specific memory locations
  - Can move data between memory and registers
  - Can declare register variables
- Basic data types of machine
  - Based on original Data Types of Target Mainframe
  - Needed for OS interactions with device's registers
- Operators included to manipulate single bits within these data types

Data Types

Table 2.2.1: Data types.

Type	Minimum size*	Range	Notes
signed char	8	-128 to 127	
unsigned char	8	0 to 255	
signed short	16	-2^{15} to $2^{15}-1$	2^{16} is 65,536
unsigned short	16	0 to $2^{16}-1$	
signed long	32	-2^{31} to $2^{31}-1$	2^{32} is about 4 billion
unsigned long	32	0 to $2^{32}-1$	
signed int	N	-2^{N-1} to $2^{N-1}-1$	*Though commonly used, we avoid these due to undefined width*
unsigned int	N	0 to $2^{N}-1$	

*The size of integer numeric data types can vary between compilers, for reasons beyond our scope. The following table lists the sizes for numeric integer data types used in this material along with the minimum size for those data types defined by the language standard.

Figure 2.2.1: Variable declarations.

```
unsigned char  ucI1;
unsigned short usI2;
signed   long  slI3;
unsigned char  bMyBitVar;
```

Feedback?

Bit Manipulation

- C has standard bit-manipulation operators.

& Bit-wise AND

| Bit-wise OR — a |= 0x4; /* Set bit 2 */

^ Bit-wise XOR — d ^= (1 << 5); /* Toggle bit 5 */

~ Negate (one's comp)

>> Right-shift — g <<= 2; /* Multiply g by 4 */

<< Left-shift — e >>= 2; /* Divide e by 4 */

What do these do ?

b &= ~0x4;

c &= ~(1 << 3);

Bit Manipulation

- C has standard bit-manipulation operators.

& Bit-wise AND	
| Bit-wise OR	a |= 0x4; /* Set bit 2 */
^ Bit-wise XOR	d ^= (1 << 5); /* Toggle bit 5 */
~ Negate (one's comp)	
>> Right-shift	g <<= 2; /* Multiply g by 4 */
<< Left-shift	e >>= 2; /* Divide e by 4 */

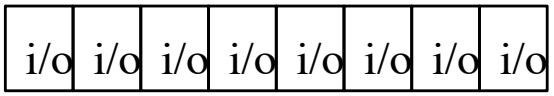
What do these do ?

b &= ~0x4; /* Clear bit 2 */

c &= ~(1 << 3); /* Clear bit 3 */

Example Usage

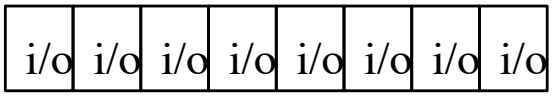
base_addr = 0x00000400



- Suppose you have a Control Register that sets directions for 8 input/output devices
- 1 := input
- 0 := output
- Device is already configured, and you want to check device #2.
- Write down the code in C……

Example Usage

base_addr = 0x00000400



- Suppose you have a Control Register that sets directions for 8 input/output devices
- 1 := input
- 0 := output
- Device is already configured, and you want to check device #2.

```
int mask = 0x4;                 /* 00000100 */
A = *base_addr & mask;          /*A only has bit 2 */
```

Ternary Operator

if-then-else style:

```
int a = 10, b = 20, c;

if (a < b) {
      c = a;
}
else {
      c = b;
}
```

Ternary Operator

```
int a = 10, b = 20, c;

c = (a < b) ? a : b
```

Ternary Operator

Write a function using the ternary operator to set the kth bit of a word to either {0, 1}.

```
// x: 8-bit value.
//k: bit position to set, range is 0-7.
// b: set bit to this, either 1 or 0
unsigned char SetBit(
unsigned char x, unsigned char k, unsigned char b)
```

Ternary Operator

Write a function using the ternary operator to set the kth bit of a word to either {0, 1}.

```
// x: 8-bit value.
//k: bit position to set, range is 0-7.
// b: set bit to this, either 1 or 0
unsigned char SetBit(
unsigned char x, unsigned char k, unsigned char b)

{ return (b ? (x | (0x01 << k)) : (x & ~(0x01 << k)) ); }
```

Ternary Operator

Write a function using the ternary operator that returns the value of the kth bit in an 8-bit word.

// x: 8-bit value.

//k: bit position to set, range is 0-7.

Ternary Operator

Write a function using the ternary operator that returns the value of the kth bit in an 8-bit word.

```
// x: 8-bit value.
//k: bit position to set, range is 0-7.

unsigned char GetBit(unsigned char x, unsigned char k)
{ return ((x & (0x01 << k)) != 0); }
```