Binary Operations

CS 3330

Samira Khan
University of Virginia
Jan 31, 2017

AGENDA

- Logistics
- Review from last Lecture
- Binary Operations
  - Logical Operations
  - Bitwise Operations
  - Examples

Feedbacks

- Quizzes are hard
  - Use the book, lectures, internet, whatever you can
  - Past quizzes are available online
- For the quiz question with: typedef struct bar { int x; } foo; I tried compiling struct foo *c; in C, and it compiled fine. So why is it wrong?
  - Dropped it
  - Good catch
  - Ask in Piazza, so that others can learn, too

Feedbacks

- Is a string still a string if it is expressed in bits? that seems like more of a philosophical question to me I'm referring to the quiz question
  - Yes
  - There is no “abcde…”
  - Everything is just 0s and 1s
  - It is just how you interpret it

AGENDA

- Logistics
- Review from last Lecture
- Binary Operations
  - Logical Operations
  - Bitwise Operations
  - Examples

Undefined Behavior

- C FAQ definition:
- *Anything at all can happen*
  - *Standard imposes no requirements;*
  - *Program may fail to compile,*
  - *or it may execute incorrectly,*
  - *or it may do exactly what the programmer intended*

Adding one to INT_MAX

if we add one to the largest representable integer, is the result negative?

```
#include <limits.h>
#include <stdio.h>


int main (void)
{
  printf ("%d\n", (INT_MAX+1) < 0);
  return 0;
}
```

Undefined, output could be zero or one

More Undefined Behavior

- Attempting to modify a string literal
  - char *p = "wikipedia"; *// valid C*
  - p[0] = 'W'; *// undefined behavior*
- Integer division by zero
  - int x = 1;
  - **return** x / 0; *// undefined behavior*
- Certain pointer operations
  - int arr[4] = {0, 1, 2, 3};
  - int *p = arr + 5; *// undefined behavior*
- Increment and assignment
  - i = i++ + 1; *// undefined behavior*

Undefined Behavior

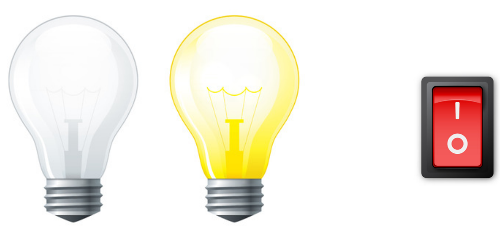
- Pros:
  - Simplifies the compiler's job
  - Generates very efficient code
  - Example: increment INT_MAX
    - Does not need to worry about overflows and result become negative
- Cons:
  - Misbehaved programs
  - Security issues
  - Example: Array out of bound

AGENDA

- Logistics
- Review from last Lecture
- Binary Operations
  - Logical Operations
  - Bitwise Operations
  - Examples

How do digital computers represent numbers?

- Made of transistors (think as a switch)
- Have only two states: ON, OFF



- Can only use 0 and 1 to represent numbers
- 3 → 0011, 10 → 1010

Binary Operations

- We need to operate on these binary numbers
- Arithmetic Operations
  - ADD, SUB, MUL, DIV
- Logical Operations
  - AND, OR, NOT
- Bitwise Operations
  - AND, OR, NOT, XOR, SHIFT

Logical Operations

Operator	Description	Example A = 2, B = 0

Logical Operations

Operator	Description	Example A = 2, B = 0
Logical AND (&&)	Both the operands are non-zero → condition becomes true	A && B = False/0

Logical Operations

Operator	Description	Example A = 2, B = 0
Logical AND (&&)	Both the operands are non-zero → condition becomes true	A && B = False/0
Logical OR (||)	Any of the two operands is non-zero → the condition becomes true	A || B = 1/true

Logical Operations

Operator	Description	Example A = 2, B = 0
Logical AND (&&)	Both the operands are non-zero → condition becomes true	A && B = False/0
Logical OR (||)	Any of the two operands is non-zero → the condition becomes true	A || B = 1/true
Logical NOT (!/~)	Reverse the logical state	!(A && B) = 1/true

Example: Short-Circuit

```
#include <stdio.h>
int zero() { printf("zero()\n"); return 0; }
int one() { printf("one()\n"); return 1; }

int main() {
printf(">␣%d\n", zero() && one());
printf(">␣%d\n", one() && zero());
return 0;
}
```

OUTPUT

```
zero()
> 0
one()
zero()
> 0
```


AGENDA

- Logistics
- Review from last Lecture
- Binary Operations
  – Logical Operations
  – **Bitwise Operations**
  – Examples

Bitwise Operations

X	Y
0	0
0	1
1	0
1	1

Bitwise Operations

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Operates on each bit

Bitwise Operations

X	Y	X AND Y	X OR Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Operates on each bit

Bitwise Operations

X	Y	X AND Y	X OR Y	X XOR Y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Operates on each bit

Bitwise Operations

X	Y	X AND Y	X OR Y	X XOR Y	NOT X
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Operates on each bit

Bitwise Operations

```
  0010
& 0100
-----------
  0000
```

AND

```
  0010
| 0100
-----------
  0110
```

OR

```
  0010
^ 0100
-----------
  0110
```

XOR

```
~ 0100
-----------
  1011
```

NOT

```
   0010
>> 0001
-----------
   0001
```

SHIFT

What about negative numbers?

SHIFT Bitwise Operations

- What value should be placed in the MSB?
  - −10 >> 1 == ???
- -10 = 1111 … 1111 0110
- -10 >> 1 ==?111 … 1111 1011

- Option 1: copy sign bit → Arithmetic shift
- Option 2: always keep zero → Logical shift

Signed and Unsigned Shift

- /*signed*/ int x = −10;
- /* arithmetic: */
- x >> 1 == −5
- x >> 4 == −1

- unsigned int y = 0xFFFFFFF6;
- /* logical */
- y >> 1 == 0x7FFFFFFB
- y >> 4 == 0x0FFFFFFF

Undefined Behavior with SHIFTS

- 0 >> 32 → undefined behavior
- 0 << 32 → undefined behavior
- (long) 0 >> 32 → okay
- (long) 0 >> 64 → undefined behavior

AGENDA

- Logistics
- Review from last Lecture
- Binary Operations
  - Logical Operations
  - Bitwise Operations
  - **Examples**

RECAP: Binary Operations

- Bitwise Operations
  - AND, OR, NOT, XOR, SHIFT

Why do Bit Manipulation?

Why do Bit Manipulation?

- Faster than many complex operations
  - DIV 25-123 cycles
  - ADD/AND/CMP/OR/SUB/XOR 1 cycle
- Examples:
  - Suppose you are designing a reliable system, want to detect if any bit flipped
  - Parity: count the number of ones, store 1 if even
  - How many ones are there in a number?
  - Suppose you want to make DIV faster if it is a power of two
  - Is a number a power of two?

Intel® 64 and IA-32 Architectures Optimization Reference Manual

Building Blocks of Bit Manipulation

- Set a bit, keep other bits unchanged
  - 0 → 1; 1 → 1
- Reset/Clear a bit, keep other bits unchanged
  - 1 → 0; 0 → 0
- Toggle a bit, keep other bits unchanged
  - 1 → 0; 0 → 1
- Extract and shift
  - 1010 1111 1010 1010 → 0000 0000 0000 1111

Building Blocks of Bit Manipulation

- Set a bit
  - 0 → 1; 1 → 1
  - X | 1
  - Example:
  - Set the third bit of 1010
  - 1**0**10 | 0100 = 1**1**10
  - $X_3\mathbf{X_2}X_1X_0$ | 0100 = $X_3\mathbf{1}X_1X_0$

Building Blocks of Bit Manipulation

- Reset/Clear a bit
  - 1 → 0; 0 → 0
  - X & 0
  - Example:
  - Clear the third bit of 1110
  - 1**1**10 & 1011 = 1**0**10
  - $X_3\mathbf{X_2}X_1X_0$ & 1011 = $X_3\mathbf{0}X_1X_0$

Building Blocks of Bit Manipulation

- Toggle a bit, keep other bits unchanged
  - 1 → 0; 0 → 1
  - X ^ 1
  - Example:
  - Toggle the third bit of the bit vector
  - 1**1**10 ^ 0100 = 1**0**10
  - 1**0**10 ^ 0100 = 1**1**10
  - $X_3\mathbf{X_2}X_1X_0$ | 0100 = $X_3\overline{X_2}X_1X_0$

Building Blocks of Bit Manipulation

- Extract and shift
  - 1010 1111 1010 1010 → 0000 0000 0000 1111
  - Mask: X & 0x0F00 = 0000 $\mathbf{X_3X_2X_1X_0}$ 0000 0000
  - Shift: X >> 8 = 0000 0000 0000 $\mathbf{X_3X_2X_1X}$

Example Problems

- Is any bit set in the bit vector?
- How many bits are set?
- How to manipulate colors in RGB?

Is any bit set in the bit vector?

- Naïve Solution

- Shift a bit and check if it is 1
- (X & 1) | ((X >> 1) & 1) | ((X >> 2) & 1) …
- 0011 0010 & 1 = 0
- 0001 1001 & 1 = 1
- 0000 1100 & 1 = 0
- ....

- Any other solution?

Is any bit set in the bit vector?

- Operate only on half
- Assume 16 bits integer
- A = (X >> 8) | X
- Take the upper half and or with X
- If any bit in X is set, lower half of a will have a set bit
- X = 0000 0100 0000 0000

```
  0000 0000 0000 0100     X>>8
| 0000 0100 0000 0000     X
---------------------------------
  -----  ------ 0000 0100     A
```


Is any bit set in the bit vector?

- A = (X >> 8) | X
- **A = (A >> 4) | A**
- A = (A >> 2) | A
- A = (A >> 1) | A
- return A & 1

```
  -----  ------ 0000 0000     A>>4
| -----  ------ 0000 0100     A
---------------------------------
  -----  -----  ----  0100    A
```


Is any bit set in the bit vector?

- A = (X >> 8) | X
- A = (A >> 4) | A
- **A = (A >> 2) | A**
- A = (A >> 1) | A
- return A & 1

```
  -----  ------ -----  0001     A>>2
| -----  ------ -----  0100     A
---------------------------------
  -----  -----  ----  -- 01     A
```

Is any bit set in the bit vector?

- A = (X >> 8) | X
- A = (A >> 4) | A
- A = (A >> 2) | A
- **A = (A >> 1) | A**
- return A & 1

```
   -----  ------ -----  -- 00      A>>1
|  -----  ------ -----  -- 01      A
---------------------------------
  -----  -----   ----   --  - 1     A
```


Example Problems

- Is any bit set in the bit vector?
- How many bits are set?
- How to manipulate colors in RGB?

Count Set bits

- Count = X & 1
- Count += (X >> 1) & 1
- Count += (X >> 2) & 1
- …

Example Problems

- Is any bit set in the bit vector?
- How many bits are set?
- How to manipulate colors in RGB?

How to manipulate colors in RGB?

- RGB 8 bits
  - Each color is represented using 8 bit
  - 0 to 255
- 32 bits: 0x 00 **BB** **GG** **RR**
- int blueMask = 0xFF0000

```
int greenMask = 0xFF00
int redMask = 0xFF;
int r = 12, g = 13, b = 14;
int bgrValue = (b << 16) + (g << 8) + r;
printf("blue:%d\n", ((bgrValue & blueMask) >> 16));
printf("red:%d\n", ((bgrValue & redMask)));
printf("green:%d\n", ((bgrValue & greenMask) >> 8));
```


Binary Operations

CS 3330

Samira Khan
University of Virginia
Jan 31, 2017