CSE115 / CSE503
Introduction to Computer Science I

Dr. Carl Alphonce
343 Davis Hall
alphonce@buffalo.edu

Office hours:

Tuesday 10:00 AM – 12:00 PM*

Wednesday 4:00 PM – 5:00 PM

Friday 11:00 AM – 12:00 PM

OR request appointment via e-mail

**Tuesday adjustments: 11:00 AM – 1:00 PM on 10/11, 11/1 and 12/6*

ROADMAP

Last time

Inheritance

Today

Inheritance

int representation in detail

Coming up

floating point representation

search

ANNOUNCEMENTS

EXAM 2 NOTICE

DATE: Tuesday November 15

TIME: 8:45 PM – 9:45 PM

LOCATION: as for exam 1

COVERAGE:

lecture material from 9/26 up to and including 11/04
lab material labs 4 – lab 9

readings: all assigned up to and including 13.4

HAVE A CONFLICT? ***Let me know by tonight***

Send e-mail with PDF of documentation.

Subject line: [CSE115] Exam 2 conflict documentation

BRING: your UB card

NO ELECTRONICS: cell phone, calculator, etc.

REVIEW

inheritance

Default constructor

Constructor chaining

this / super

potential errors

int

Primitives in Java

- Java has eight primitive types
  - boolean
  - integral types:
    - signed: **long, int, short, byte**
    - unsigned: char
  - floating point types: double, float
- Values of the primitive types are not objects
  - no properties
  - no capabilities

int

values: 0, 1, -1, 2, -2, …

maximum int: 2147483647 = $+2^{(32-1)}-1$

minimum int: -2147483648 = $-2^{(32-1)}$

operations: + - * / %

5+2 = 7	+: (int,int) → int
5-2 = 3	-: (int,int) → int
5*2 = 10	*: (int,int) → int
5/2 = 2 (quotient)	/: (int,int) → int
5%2 = 1 (remainder)	%: (int,int) → int

integral types' representations

representation used differs according to whether type is signed (byte, short, int, long) or unsigned (char):

- signed integral values are represented using "two's complement" representation
- unsigned integral values are represented using "binary" representation

size of representation differs according to type:

- byte is 1 byte wide (1 byte = 8 bits)
- short is 2 bytes wide
- int is 4 bytes wide
- long is 8 bytes wide

main point: values of different types have different representations – you can't "mix and match"!

types of operators for type int

Notice that all of these operators take two int arguments, and produce an int result.

There is hardware circuitry to perform these operations.

Two's complement

- uses a fixed-width encoding
- encodes a limited range of values
- encodes both negative and non-negative values
- familiar properties hold
  - ✓ unique representation of zero ($0 = -0$)
  - ✓ $x + 0 = 0 + x = x$
  - ✓ $x = -(-x)$
  - ✓ $x + (-x) = 0$
  - ✓ $x - y = x + (-y)$
- this last property lets us use addition circuitry to perform subtraction (to subtract y from x, negate y and add to x)

Bit pattern interpretation

half of bit patterns (those with a zero in the leftmost bit) are for non-negative values, and are interpreted just as base 2 (binary) numbers are

the assignment of values to the remaining bit patterns is done as described on the board bit patterns

000

001

010

011

100

101

110

111

binary interpretation

BINARY

000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

same bit patterns

BINARY		TWO'S COMPLEMENT
000	0	000
001	1	001
010	2	010
011	3	011
100	4	100
101	5	101
110	6	110
111	7	111

some patterns have same interpretation

BINARY		TWO'S COMPLEMENT	
000	0	000	0
001	1	001	1
010	2	010	2
011	3	011	3
100	4	100	
101	5	101	
110	6	110	
111	7	111	

two's complement

BINARY		TWO'S COMPLEMENT	
000	0	000	0
001	1	001	1
010	2	010	2
011	3	011	3
100	4	100	-4
101	5	101	-3
110	6	110	-2
111	7	111	-1

negation

To find representation of -x given the representation of x:

1. find the one's complement of x

   do this by flipping all the bits in the representation (1 becomes 0, 0 becomes 1)

2. find the two's complement of the result

   do this by adding one to the one's complement, ignoring any overflow carry

Example

Using a 4-bit wide representation, find the representation of -3:

start with representation of +3: 0011

compute its one's complement: 1100

compute its two's complement: 1101

Therefore, the representation of -3 is 1101

Exercise: verify that the desirable properties hold!

Addition

Keep in mind: representation has a fixed width!

Add as usual, but ignore overflow carry.

Extra pattern?

Since -0 = 0, there is one leftover “negative” bit string

Let that represent a negative number, -8 in the case of a 4-bit wide representation

In general, range of values for a k-bit wide two’s complement representation is from $-2^{(k-1)}$ to $+2^{(k-1)}-1$

For 4-bit wide representation: -8 to +7

Rollover

What happens when you add 1 to 7 in the 4-bit wide scheme?

0111 + 0001 = 1000

The answer is -8 (!)

Adding one to the largest magnitude positive number yields the largest magnitude negative number.

unexpected negation

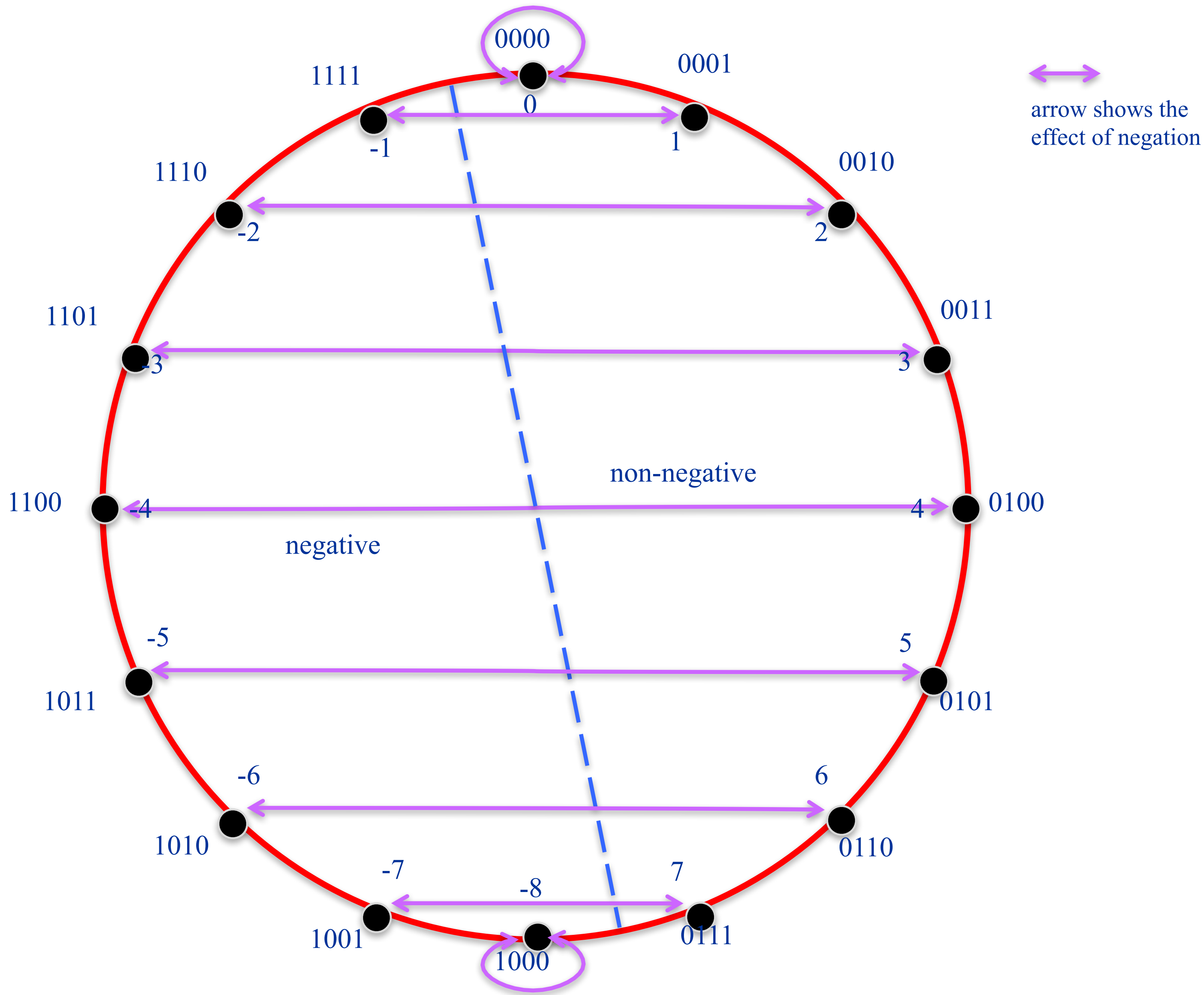
The negative of the largest magnitude negative number is itself.

In the 4-bit wide scheme, -(-8) is -8.

Understand the representation!

It is important that you understand the limitations of working with fixed-width representations.

See next slide for a visualization of a 4-bit wide two's complement representation scheme.


0000
0001
1111
0
1
-1
1110
0010
-2
2
1101
0011
-3
3
non-negative
1100
-4
4
0100
negative
-5
5
1011
0101
-6
6
1010
0110
-7
-8
7
1001
0111
1000
arrow shows the effect of negation

All four signed integral types use the two's complement representation scheme

The width of the representations differ, and therefore the possible range of values:

TYPE	BITS	BYTES	MIN	MAX
byte	8	1	-128	+127
short	16	2	-32,768	+32,767
int	32	4	-2,147,483,648	+2,147,483,647
long	64	8	-9,223,372,036,854,775,808	+9,223,372,036,854,775,807

That's 9 quintillion, 233 quadrillion, 372 trillion, 36 billion, 854 million, 775 thousand, 8 hundred and 8.

In general, for an n-bit wide representation, the range of possible values is

$$-2^{(n-1)} \rightarrow +2^{(n-1)} - 1$$