

CSE1301
Computer Programming
Lecture 29:
Number Representation (Part 1)

1

Topics

- Binary numbers
- Signed Magnitude
- Two's Complement
- Excess-*k*

2

Representing Characters

- ASCII encoding (American Standard Code for Information Interchange)
- Each character represented by a one-byte (8-bit) number
- Other representations possible too:
 - EBCDIC (used by older IBM machines)
 - Unicode (16-bit character codes, provides enough codes for characters from all modern languages)

3

Representing Integers

- We represent integers using a base-10 positional notation.
- So the number 90210 means:
 $9 \times 10^4 + 0 \times 10^3 + 2 \times 10^2 + 1 \times 10^1 + 0 \times 10^0$

4

Representing Integers

- Computers represent integers using a base-2 positional notation.
- So the number 101011 means:
 $1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
 $1 \times 32 + 0 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$
43

5

Representing Integers

- Curiously, ye Olde Englishe pub-keepers used the same system:
- So the number 101011 means:
 $1 \times \text{gallon} + 0 \times \text{pottle} + 1 \times \text{quart} + 0 \times \text{pint} + 1 \times \text{chopin} + 1 \times \text{gill}$
 $1 \times 4.61 + 0 \times 2.31 + 1 \times 1.11 + 0 \times 0.61 + 1 \times 0.31 + 1 \times 0.11$
6.1 litres

6

Representing Integers

- The first few binary numbers are:

0000.....	0
0001.....	1
0010.....	2
0011.....	3
0100.....	4
0101.....	5
0110.....	6
0111.....	7
1000.....	8
1001.....	9
1010.....	10
1011.....	11
1100.....	12
1101.....	13
1110.....	14
1111.....	15

7

Converting decimal to binary

- We've seen that you convert binary to decimal by multiplying and adding.
- So it's no surprise that you convert decimal to binary by dividing and subtracting (well, remaindering, actually).

8

Converting decimal to binary

123	
$\frac{-2}{61}$	→ remainder 1
$\frac{-2}{30}$	→ remainder 1
$\frac{-2}{15}$	→ remainder 0
$\frac{-2}{7}$	→ remainder 1
$\frac{-2}{3}$	→ remainder 1
$\frac{-2}{1}$	→ remainder 1
$\frac{-2}{0}$	→ remainder 1

9

Converting decimal to binary

123	
$\frac{-2}{61}$	→ remainder 1
$\frac{-2}{30}$	→ remainder 1
$\frac{-2}{15}$	→ remainder 0
$\frac{-2}{7}$	→ remainder 1
$\frac{-2}{3}$	→ remainder 1
$\frac{-2}{1}$	→ remainder 1
$\frac{-2}{0}$	→ remainder 1



10

Converting decimal to binary

123		1111011
$\frac{-2}{61}$	→ remainder 1	↑
$\frac{-2}{30}$	→ remainder 1	
$\frac{-2}{15}$	→ remainder 0	
$\frac{-2}{7}$	→ remainder 1	
$\frac{-2}{3}$	→ remainder 1	
$\frac{-2}{1}$	→ remainder 1	
$\frac{-2}{0}$	→ remainder 1	

11

Converting decimal to binary

102	
$\frac{-2}{51}$	→ remainder 0
$\frac{-2}{25}$	→ remainder 1
$\frac{-2}{12}$	→ remainder 1
$\frac{-2}{6}$	→ remainder 0
$\frac{-2}{3}$	→ remainder 0
$\frac{-2}{1}$	→ remainder 1
$\frac{-2}{0}$	→ remainder 1

12

Converting decimal to binary

102		
<u>51</u>	→	remainder 0
51		
<u>25</u>	→	remainder 1
25		
<u>12</u>	→	remainder 1
12		
<u>6</u>	→	remainder 0
6		
<u>3</u>	→	remainder 0
3		
<u>1</u>	→	remainder 1
1		
<u>0</u>	→	remainder 1

↑

13

Converting decimal to binary

102			1100110
<u>51</u>	→	remainder 0	
51			
<u>25</u>	→	remainder 1	
25			
<u>12</u>	→	remainder 1	
12			
<u>6</u>	→	remainder 0	
6			
<u>3</u>	→	remainder 0	
3			
<u>1</u>	→	remainder 1	
1			
<u>0</u>	→	remainder 1	

↑

14

Converting decimal to binary

102			1100110
<u>51</u>	→	remainder 0	
51			
<u>25</u>	→	remainder 1	
25			
<u>12</u>	→	remainder 1	
12			
<u>6</u>	→	remainder 0	
6			
<u>3</u>	→	remainder 0	
3			
<u>1</u>	→	remainder 1	
1			
<u>0</u>	→	remainder 1	

↑

Most significant bit

15

Converting decimal to binary

102			1100110
<u>51</u>	→	remainder 0	
51			
<u>25</u>	→	remainder 1	
25			
<u>12</u>	→	remainder 1	
12			
<u>6</u>	→	remainder 0	
6			
<u>3</u>	→	remainder 0	
3			
<u>1</u>	→	remainder 1	
1			
<u>0</u>	→	remainder 1	

↑

Least significant bit

16

Representing Signed Integers

- That only takes care of the positive integers.
- To handle negative integers, we need to use one of the bits to store the sign.
- The rest of the bits store the absolute value of the number.
- Hence this is known as "signed magnitude" representation.

17

Representing Signed Integers

- If we use the first ("top") bit for the sign:

0000.....	+0
0001.....	+1
0010.....	+2
0011.....	+3
0100.....	+4
0101.....	+5
0110.....	+6
0111.....	+7
1000.....	-0
1001.....	-1
1010.....	-2
1011.....	-3
1100.....	-4
1101.....	-5
1110.....	-6
1111.....	-7

18

Representing Signed Integers

- If we use the first ("top") bit for the sign:

0000.....	+0
0001.....	+1
0010.....	+2
0011.....	+3
0100.....	+4
0101.....	+5
0110.....	+6
0111.....	+7
1000.....	-0
1001.....	-1
1010.....	-2
1011.....	-3
1100.....	-4
1101.....	-5
1110.....	-6
1111.....	-7

19

Adding binary numbers

- For individual bits there are only four possibilities:

$$\begin{array}{r}
 0 \\
 + 0 \\
 \hline
 0
 \end{array}
 \qquad
 \begin{array}{r}
 0 \\
 + 1 \\
 \hline
 1
 \end{array}
 \qquad
 \begin{array}{r}
 1 \\
 + 0 \\
 \hline
 1
 \end{array}
 \qquad
 \begin{array}{r}
 1 \\
 + 1 \\
 \hline
 10
 \end{array}$$

20

Adding binary numbers

- For multiple digits we do the same as in base-10: we add corresponding bits and carry the twos:

$$\begin{array}{r}
 11100101 \\
 + 1011101 \\
 \hline
 \end{array}$$

21

Adding binary numbers

- For multiple digits we do the same as in base-10: we add corresponding bits and carry the twos:

$$\begin{array}{r}
 \\
 11100101 \\
 + 1011101 \\
 \hline
 0
 \end{array}$$

22

Adding binary numbers

- For multiple digits we do the same as in base-10: we add corresponding bits and carry the twos:

$$\begin{array}{r}
 \\
 11100101 \\
 + 1011101 \\
 \hline
 10
 \end{array}$$

23

Adding binary numbers

- For multiple digits we do the same as in base-10: we add corresponding bits and carry the twos:

$$\begin{array}{r}
 \\
 11100101 \\
 + 1011101 \\
 \hline
 010
 \end{array}$$

24

Adding binary numbers

- For multiple digits we do the same as in base-10: we add corresponding bits and carry the twos:

$$\begin{array}{r} 11 \\ 11100101 \\ + \underline{1011101} \\ \hline 0010 \end{array}$$

25

Adding binary numbers

- For multiple digits we do the same as in base-10: we add corresponding bits and carry the twos:

$$\begin{array}{r} 11 \\ 11100101 \\ + \underline{1011101} \\ \hline 00010 \end{array}$$

26

Adding binary numbers

- For multiple digits we do the same as in base-10: we add corresponding bits and carry the twos:

$$\begin{array}{r} 1 \\ 11100101 \\ + \underline{1011101} \\ \hline 000010 \end{array}$$

27

Adding binary numbers

- For multiple digits we do the same as in base-10: we add corresponding bits and carry the twos:

$$\begin{array}{r} 1 \\ 11100101 \\ + \underline{1011101} \\ \hline 1000010 \end{array}$$

28

Adding binary numbers

- For multiple digits we do the same as in base-10: we add corresponding bits and carry the twos:

$$\begin{array}{r} 1 \\ 11100101 \\ + \underline{1011101} \\ \hline 101000010 \end{array}$$

29

Adding binary numbers

- For multiple digits we do the same as in base-10: we add corresponding bits and carry the twos:

$$\begin{array}{r} 11100101 \rightarrow 229 \\ + \underline{1011101} \rightarrow \underline{+93} \\ 101000010 \rightarrow 322 \end{array}$$

30

Adding binary numbers

- But that only works for *unsigned* numbers.
- For signed numbers (in signed magnitude format) it's much more complicated.
- Try and work out a scheme yourself (hint: you need to consider four cases).

31

Two's complement representation

- Another representation scheme makes addition and subtraction easy, regardless of the signs of the operands.
- Rather than using the first bit as a sign bit, we give it a *negative* weight.
- In other words, if it's the eighth bit, its positional value is -2^7 , rather than $+2^7$

32

Two's complement representation

- So now the number 101011 means:

$$1 \times -2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$1 \times -32 + 0 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$$

$$-21$$

33

Two's complement representation

- Now the first few numbers are:

0000.....	+0
0001.....	+1
0010.....	+2
0011.....	+3
0100.....	+4
0101.....	+5
0110.....	+6
0111.....	+7
1000.....	-8
1001.....	-7
1010.....	-6
1011.....	-5
1100.....	-4
1101.....	-3
1110.....	-2
1111.....	-1

34

Two's complement representation

- Now the first few numbers are:

0000.....	+0
0001.....	+1
0010.....	+2
0011.....	+3
0100.....	+4
0101.....	+5
0110.....	+6
0111.....	+7
1000.....	-8
1001.....	-7
1010.....	-6
1011.....	-5
1100.....	-4
1101.....	-3
1110.....	-2
1111.....	-1

35

Negation with two's complement

- Even though the top bit indicates the sign, we can't just flip the bit to negate a number.
- To negate a two's complement number, we have to flip *all* its bits and then add 1:

00101010	(-0+0+32+0+8+0+2+0)	+42
11010101	(-128+64+0+16+0+4+1)	-43
11010110	(-43 + 1)	-42

36

Negation with two's complement

- Even though the top bit indicates the sign, we can't just flip the bit to negate a number.
- To negate a two's complement number, we have to flip *all* its bits and then add 1:

$$\begin{array}{r}
 11010110 \quad (-128+64+0+16+0+4+2+0) \quad -42 \\
 00101001 \quad (-0+0+32+0+8+0+0+1) \quad +41 \\
 00101010 \quad (+41 + 1) \quad +42
 \end{array}$$

37

Addition with two's complement

- The top bit acts just like all the other bits: it tells us whether to include the multiplier for the particular position).
- The fact that the multiplier is negative is irrelevant.
- So we can just add 2 two's complement numbers as if they were unsigned.

38

Addition with two's complement

- The mechanics are exactly the same as in the earlier example:

$$\begin{array}{r}
 11100101 \\
 + 01011101 \\
 \hline
 \end{array}$$

39

Addition with two's complement

- The mechanics are exactly the same as in the earlier example:

$$\begin{array}{r}
 \\
 11100101 \\
 + 01011101 \\
 \hline
 0
 \end{array}$$

40

Addition with two's complement

- The mechanics are exactly the same as in the earlier example:

$$\begin{array}{r}
 \\
 11100101 \\
 + 01011101 \\
 \hline
 10
 \end{array}$$

41

Addition with two's complement

- The mechanics are exactly the same as in the earlier example:

$$\begin{array}{r}
 \\
 11100101 \\
 + 01011101 \\
 \hline
 010
 \end{array}$$

42

Addition with two's complement

- The mechanics are exactly the same as in the earlier example:

$$\begin{array}{r}
 11 \\
 11100101 \\
 + \underline{01011101} \\
 \hline
 0010
 \end{array}$$

43

Addition with two's complement

- The mechanics are exactly the same as in the earlier example:

$$\begin{array}{r}
 1 \\
 11100101 \\
 + \underline{01011101} \\
 \hline
 00010
 \end{array}$$

44

Addition with two's complement

- The mechanics are exactly the same as in the earlier example:

$$\begin{array}{r}
 1 \\
 11100101 \\
 + \underline{01011101} \\
 \hline
 000010
 \end{array}$$

45

Addition with two's complement

- The mechanics are exactly the same as in the earlier example:

$$\begin{array}{r}
 1 \\
 11100101 \\
 + \underline{01011101} \\
 \hline
 1000010
 \end{array}$$

46

Addition with two's complement

- The mechanics are exactly the same as in the earlier example:

$$\begin{array}{r}
 1 \\
 11100101 \\
 + \underline{01011101} \\
 \hline
 101000010
 \end{array}$$

47

Addition with two's complement

- The mechanics are exactly the same as in the earlier example (except we only have 8 bits so we throw away the extra one):

$$\begin{array}{r}
 1 \\
 11100101 \\
 + \underline{01011101} \\
 \hline
 101000010
 \end{array}$$

48

Addition with two's complement

- The mechanics are exactly the same as in the earlier example (except we only have 8 bits so we throw away the extra one):

$$\begin{array}{r}
 11100101 \rightarrow -27 \\
 + \underline{01011101} \rightarrow +93 \\
 01000010 \rightarrow 66
 \end{array}$$

49

Addition with two's complement

- But things don't always work out so neatly.
- Because we have only a limited number of bits, some sums are too big to be represented correctly.

50

Addition with two's complement

- Consider the addition of two large positive numbers:

$$\begin{array}{r}
 01100101 \\
 + \underline{01011101}
 \end{array}$$

51

Addition with two's complement

- Consider the addition of two large positive numbers:

$$\begin{array}{r}
 1 \\
 01100101 \\
 + \underline{01011101} \\
 0
 \end{array}$$

52

Addition with two's complement

- Consider the addition of two large positive numbers:

$$\begin{array}{r}
 1 \\
 01100101 \\
 + \underline{01011101} \\
 10
 \end{array}$$

53

Addition with two's complement

- Consider the addition of two large positive numbers:

$$\begin{array}{r}
 1 1 \\
 01100101 \\
 + \underline{01011101} \\
 010
 \end{array}$$

54

Addition with two's complement

- Consider the addition of two large positive numbers:

$$\begin{array}{r} 111 \\ 01100101 \\ + 01011101 \\ \hline 0010 \end{array}$$

55

Addition with two's complement

- Consider the addition of two large positive numbers:

$$\begin{array}{r} 1111 \\ 01100101 \\ + 01011101 \\ \hline 00010 \end{array}$$

56

Addition with two's complement

- Consider the addition of two large positive numbers:

$$\begin{array}{r} 11111 \\ 01100101 \\ + 01011101 \\ \hline 000010 \end{array}$$

57

Addition with two's complement

- Consider the addition of two large positive numbers:

$$\begin{array}{r} 111111 \\ 01100101 \\ + 01011101 \\ \hline 1000010 \end{array}$$

58

Addition with two's complement

- Consider the addition of two large positive numbers:

$$\begin{array}{r} 111111 \\ 01100101 \\ + 01011101 \\ \hline 11000010 \end{array}$$

59

Addition with two's complement

- Consider the addition of two large positive numbers:

$$\begin{array}{r} 01100101 \rightarrow 101 \\ + 01011101 \rightarrow +93 \\ \hline 11000010 \rightarrow -62 \end{array}$$

- Oops!
- This is known as *overflow*.

60

Subtraction with two's complement

- $X - Y = X + (-Y)$
- Therefore to subtract two's complement numbers, negate the second operand and add.

61

Excess- k representation

- Yet another way to represent numbers in binary.
- For N bit numbers, k is $2^{N-1}-1$
- So for 4-bit integers, k is 7
- The value of each bit string is its unsigned value minus k .

62

Excess- k representation

- "Sliding ruler"

	Unsigned	Excess- k
0000.....	0	-7
0001.....	1	-6
0010.....	2	-5
0011.....	3	-4
0100.....	4	-3
0101.....	5	-2
0110.....	6	-1
0111.....	7	0
1000.....	8	1
1001.....	9	2
1010.....	10	3
1011.....	11	4
1100.....	12	5
1101.....	13	6
1110.....	14	7
1111.....	15	8

Note: A red double-headed arrow labeled $k=7$ spans from the 0 in the Unsigned column to the 7 in the Excess- k column.

63

Excess- k representation

- Now the first few numbers are:

0000.....	-7	(i.e. 0-7)
0001.....	-6	(i.e. 1-7)
0010.....	-5	(i.e. 2-7)
0011.....	-4	(i.e. 3-7)
0100.....	-3	(i.e. 4-7)
0101.....	-2	(i.e. 5-7)
0110.....	-1	(i.e. 6-7)
0111.....	+0	(i.e. 7-7)
1000.....	+1	(i.e. 8-7)
1001.....	+2	(i.e. 9-7)
1010.....	+3	(i.e. 10-7)
1011.....	+4	(i.e. 11-7)
1100.....	+5	(i.e. 12-7)
1101.....	+6	(i.e. 13-7)
1110.....	+7	(i.e. 14-7)
1111.....	+8	(i.e. 15-7)

64

Excess- k representation

- Now the first few numbers are:

0000.....	-7
0001.....	-6
0010.....	-5
0011.....	-4
0100.....	-3
0101.....	-2
0110.....	-1
0111.....	0
1000.....	+1
1001.....	+2
1010.....	+3
1011.....	+4
1100.....	+5
1101.....	+6
1110.....	+7
1111.....	+8

65

Excess- k representation

- Top bit is still the sign bit (but now 1 is +ve, 0 is -ve).
- Numerical order is the same as lexical order.
- Disadvantage in arithmetic (e.g. have to subtract a k after adding two excess- k numbers)

66

Excess- k representation: Example

- Convert -14 (decimal) to 8-bit excess- k
- What is k ?
 $k = 2^{N-1} - 1 = 2^{8-1} - 1 = 127$
- Find the number u such that $u - k = -14$.
 $u - 127 = -14$, implies $u = 113$
- Convert u to unsigned binary:
 001110001

67

Limitations of representations

- None of these number representations acts exactly like the integers.
- Infinitely many integers, but only 2^N binary numbers for a specific N -bit representation.
- That means some operations *will* overflow.
- If the program doesn't crash when that happens, it will simply produce wrong answers (which is worse!)

68

Limitations of representations

- Computer arithmetic is only an approximation of integer arithmetic.
- Many of the arithmetic laws you're used to don't always hold.

69

Reading

- Brookshear: Sections 1.4 to 1.7
- D&D: Appendix E
- Donald E. Knuth, The Art of Computer Programming, Vol. 2: Semnumerical Algorithms, 2nd edition, Addison-Wesley, 1981

70