Number Representations

You will learn about the binary number system and how subtractions are performed on the computer.

How Does A Computer Work?

•Simple: something is either in one state or another.

•All parts of modern computers work this way.

•This two state approach is referred to as *bi*nary (bi = two, for 2 states).

What Is Binary?

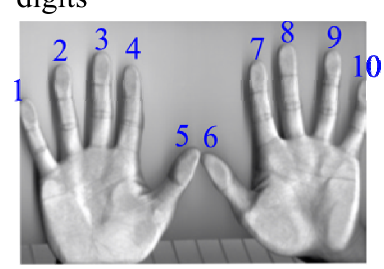
- (What you know): Binary is a method of representing information that uses two states.
- (What you may not be aware of): The number system that you are familiar (decimal) uses 10 states to represent information.

How Is Decimal Used To Store Numeric Information

- Base 10
  - 10 unique symbols are used to represent values

0
1
2
3
4
5
6
7
8
9
10
:

The number of digits is based on…the number of digits



The largest decimal value that can be represented by a single decimal digit is 9 = base(10) - 1

How Does Counting In Decimal Work?

0
1
2
3
4
5
6
7
8
9

Column 1 counts through all 10 possible values

For the next value, column 1 resets back to zero and column 2 increases by one

10
11
12
13
14
15
16
17

Column 1 counts through all 10 possible values, column 2 remains unchanged

18
19

Column 1 counts through all 10 possible values, column 2 remains unchanged

20
21
22
23
24
25
26
27
28
29
30
31
Etc.

For the next value, column 1 resets back to zero and column 2 increases by one

Decimal: Summary

- Base ten
- Employs ten unique symbols (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- Each digit can only take on the value from 0 – 9
  - Once a column has traversed all ten values then that column resets back to zero (as does it's right hand neighbours) and the column to it's immediate left increases by one.

Binary: Summary

- Base two
- Employs two unique symbols (0 and 1)
- Each digit can only take on the value 0 or the value 1
  - Once a column has traversed both values then that column resets back to zero (as does it's right hand neighbours) and the column to it's immediate left increases by one.

Counting In Binary

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

Storing Information With Binary

- Text: ASCII represents simple alphanumeric information

8 bits:
1 used for error checking
7 for the alphanumeric information = 128 combinations

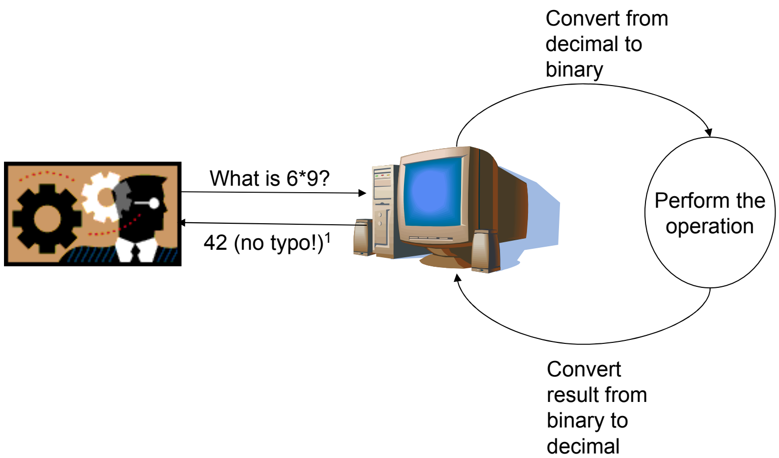
- Text: beyond simple English representations
  - Arabic, Dutch, Chinese, French, German, Spanish etc.
  - Representing this expanded text information uses additional bits:
    - 16 bits = 65,536 combinations
    - 24 bits = 16,777,216 combinations

Storing Other Information (2)

- Colors: using ~16 million colors can present a ‘true life’ representation, how are the color combinations encoded?

The Bigger Picture

•How does binary fit in when using a computer.



1 "*Hitchhicker's guide to the Galaxy*" by Douglas Adams

Converting From Binary To Decimal

•Start with some binary number to convert:

- E.g., 1 0 1. 1

•Label each of the binary digits:

- Starting with the digit immediately left of the decimal point and moving left (away from the decimal point) label the binary digits 0, 1, 2, 3 etc. in succession.
- Starting with the digit immediately right of the decimal point and moving right (away from the decimal) label the binary digits -1, -2, -3...

2 1 0 -1 ← **Position of each binary digit**

- E.g., 1 0 1. 1 ← **Binary number to be converted**

•Evaluate the expression: the binary digit raised to some exponent[1], multiply the resulting exponent by the corresponding digit and sum the resulting products.

Value in decimal = $(1x2^2) + (0x2^1) + (1x2^0) + (1x2^{-1}) = (1x4) + (0x2) + (1x1) + (1 * 1/2) = 4 + 0 + 1 + 0.5 = 5.5$

1 The value of this exponent will be determined by the position of the digit (value of the superscript)

Binary To Decimal: Other Examples

- $0101.11_2 = ????_{10}$
- $100000_2 = ????_{10}$
- $011111_2 = ????_{10}$

Decimal To Binary

Split up the integer and the fractional portions:

1) For the integer portion:

   a. Divide the integer portion of the decimal number by two.
   b. The remainder becomes the first integer digit of the number (immediately left of the decimal) in binary.
   c. The quotient becomes the new integer value.
   d. Divide the new integer value by two.
   e. The new remainder becomes the second integer digit of the binary number (second digit to the left of the decimal).
   f. Continue dividing until the quotient is less than two and this quotient becomes the last integer digit of the binary number.

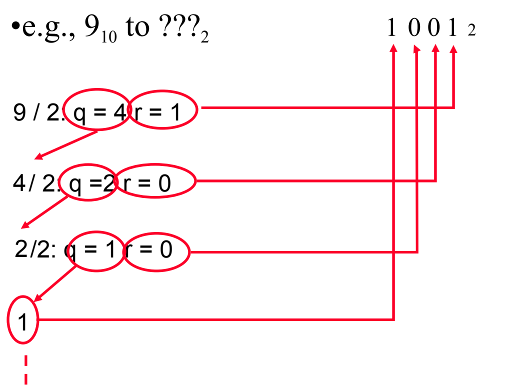
Decimal To Binary (2)

2) For the fractional portion:

a. Multiply by two.
b. The integer portion (if any) of the product becomes the first fractional digit of the converted number (first digit to the right of the decimal).
c. The remaining fractional portion of the product is then multiplied by two.
d. The integer portion (if any) of the new product becomes the second fractional digit of the converted number (second digit to the right of the decimal).
e. Keep multiplying by two until either the resulting fractional part of the product equals zero or you have the desired number of places of precision.

Decimal To Binary (3)

•e.g., 9_{10} to $???_2$



Stop dividing! (quotient less than target base)

Decimal To Binary: Other Examples

- $5.75_{10} = ????_2$
- $32_{10} = ????_2$
- $31_{10} = ????_2$

Addition In Binary: Five Cases

- Case 1: sum = 0, no carry out

$$\begin{array}{r} 0 \\ +\ \underline{0} \\ 0 \end{array}$$

Case 2: sum = 1, no carry out

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

Case 3: sum = 1, no carry out

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

Case 4: sum 0, carry out = 1

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

1 + 1 = 2 (in decimal)
= 10 (in binary)

Addition In Binary: Five Cases (2)

Case 5: Sum = 1, Carry out = 1

```
  1
   1
   1
+  1
 1 1
```

1 + 1 + 1 = 3 (in decimal)
= 11 (in binary)

Overflow: A Real World Example

•You can only represent a finite number of values

Overflow: Binary

•Occurs when you don't have enough bits to represent a value (“wraps around” to zero)

Binary (1 bit)	Value
0	0
1	1

0 0

1 1

: :

Binary (2 bits)	Value
00	0
01	1
10	2
11	3

00 0

01 1

10 2

11 3

: :

Binary (3 bits)	Value
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

000 0

001 1

: :

Overflow: Morale

•Regardless of the number of bits used to represent a number there always exists the possibility of an incorrect result due to overflow.

•Understanding how overflow works will help you determine where the errors may exist in your program and what is causing them.

Subtraction

- In the real world

  A - B
- In the computer

  A - B

Subtraction

- In the real world

  A - B
- ~~In the computer~~

  ~~A - B~~ ← **Not done this way!**

  A + (-B)

Binary Subtraction

•Requires the complementing of a binary number

- i.e., A – B becomes A + (-B)

•The complementing can be performed by representing the negative number as a twos complement value.

Complementing Binary Using The Twos Complement Representation

•For positive values there is no difference (no change is needed)

- e.g., positive seven (The 'A' in the expression A – B)

  0111 (regular binary)

  0111 (Twos complement equivalent)

•For negative values complement the number by negating the number: reversing (flipping) the bits (i.e., a 0 becomes 1 and 1 becomes 0) *and adding one to the result*.

- e.g., minus six (The 'B' in the expression A – B becomes A+(-B))

  -0110 (regular binary)

  1010 (Twos complement equivalent)

Interpreting The Bit Pattern: Complements

- Recall:
  - Positive values remain unchanged:
    - 0110 is the same value with both representations.
  - Negative values are converted through complementing:
    - Twos complement: negate the bits and add one
      - -0110 becomes 1010
- Problem: the sign must be retained (complements don't use a minus sign).
- Approach:
  - One bit (most significant bit/MSB or the signed bit) is used to indicate the sign of the number.
  - This bit cannot be used to represent the magnitude (size) of the number
  - If the MSB equals 0, then the number is positive
    - e.g. 0 bbb is a positive number (bbb stands for a binary number)
  - If the MSB equals 1, then the number is negative
    - e.g. 1 bbb is a negative number (bbb stands for a binary number)

Summary Of The Binary Representations

	Positive values are represented with:	Negative values are represented with:
Regular binary	No explicit symbol is needed (rarely is a plus '+' used) e.g., 100 vs. +100	A minus '-' sign e.g., **-**100
Twos complement	The sign bit (MSB) is zero e.g., **0**11	The sign bit (MSB) is one e.g., **1**00

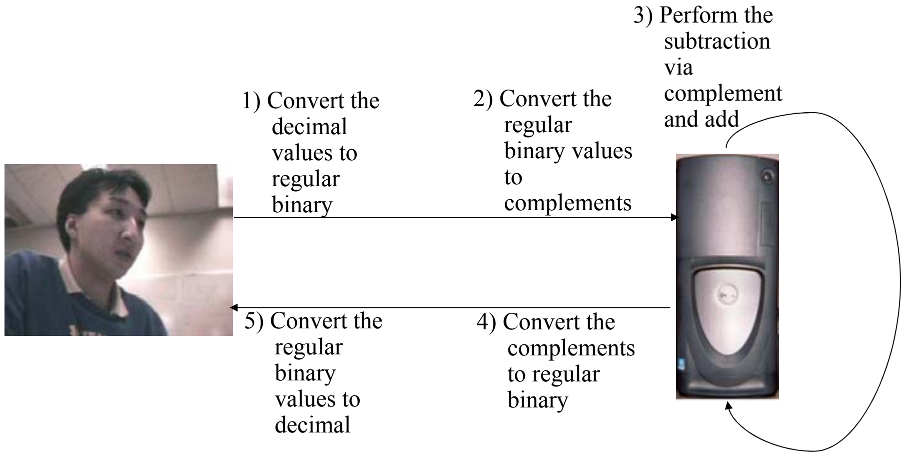
What You Already Should Know

- How to convert from decimal to binary.
- How to convert from binary to decimal.

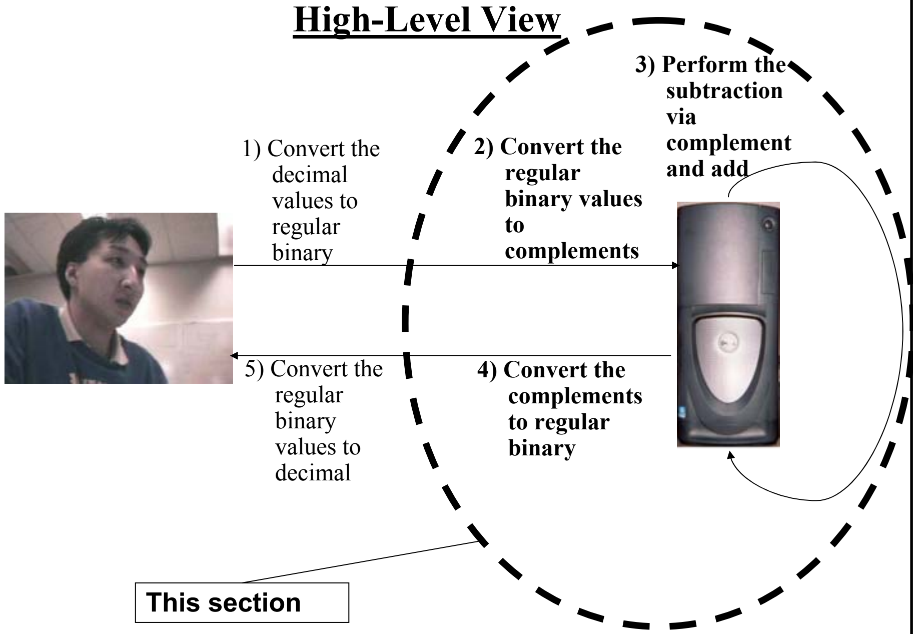
What You Will Learn

- How to subtract numbers with the complement and add technique:
  - The operation A – B is performed as A + (-B)

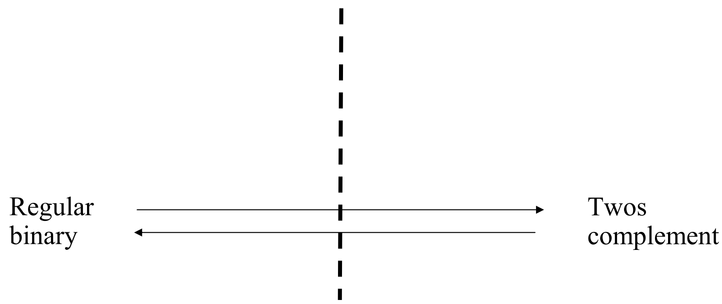
Binary Subtraction Via Complement And Add: A High-Level View



Binary Subtraction Via Complement And Add: A High-Level View



Reminder: Crossing The Boundary Between Regular And Signed Binary



Each time that this boundary is crossed (steps 2 & 4 from the previous slide) apply the rule:

1) Positive numbers pass unchanged
2) Negative numbers must be converted (complemented): negate and add one to the result

Binary Subtraction Through Twos Complements

1) *Convert from regular binary to a 2's complement representation* (check if it's preceded by a minus sign).
   a. If the number is not preceded by a minus sign, it's positive (leave it alone).
   b. If the number is preceded by a minus sign, the number is negative (complement it and discard the minus sign).
      i. Flip the bits.
      ii. Add one to the result.
2) Add the two binary numbers.
3) Check if there is overflow (a bit is carried out) and if so discard it.
4) *Convert the 2's complement value back to regular binary* (check the value of the MSB).
   a. If the MSB = 0, the number is positive (leave it alone).
   b. If the MSB = 1, the number is negative (complement it and precede the number with a negative sign).
      i. Flip the bits.
      ii. Add one to the result.

Binary Subtraction Through Twos Complements

	Step 1A: flip bits	Step 1B: add 1	Step 2: Add no's
e.g. 01000_2	01000_2	01000_2	01000_2
- $\underline{00010_2}$ →	$\underline{11101_2}$ →	$\underline{11110_2}$ →	$\underline{11110_2}$
			1 00110_2

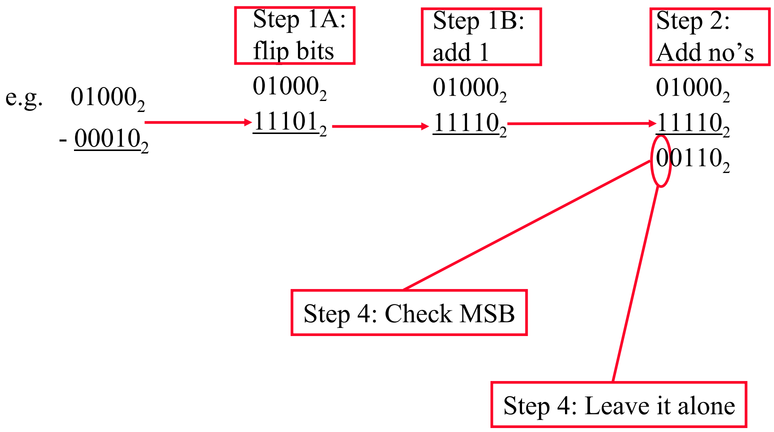
Step 3: Check for overflow

Binary Subtraction Through Twos Complements

	Step 1A: flip bits	Step 1B: add 1	Step 2: Add no's
e.g. 01000_2	01000_2	01000_2	01000_2
- $\underline{00010_2}$ →	$\underline{11101_2}$ →	$\underline{11110_2}$ →	$\underline{11110_2}$
			00110_2

Step 3: Discard it

Binary Subtraction Through Twos Complements



Overflow: Regular Binary

- Occurs when you don't have enough bits to represent a value (wraps –around to zero)

Binary (1 bit)	Value
0	0
1	1

0 0
: :

Binary (2 bits)	Value
00	0
01	1
10	2
11	3

00 0
: :

Binary (3 bits)	Value
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

000 0
: :

Overflow: Signed

- In all cases it occurs do to a "shortage of bits".
- Subtraction – subtracting two negative numbers results in a positive number.

  e.g. - 7

      - <u>1</u>

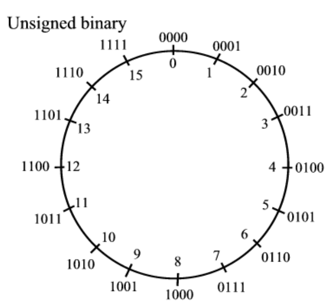
      + 7
- Addition – adding two positive numbers results in a negative number.

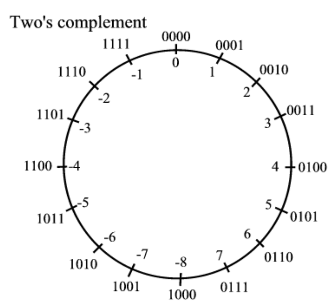
  e.g. 7

      + <u>1</u>

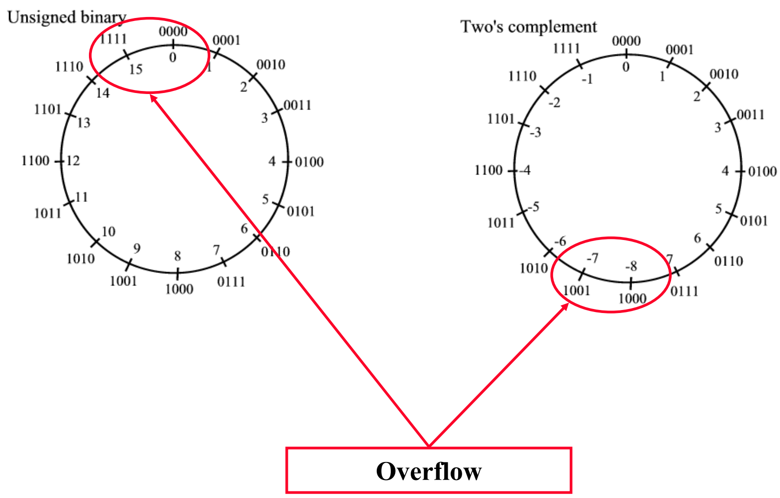
      - 8

Summary Diagram Of The Binary Representations





Summary Diagram Of The Binary Representations



After This Section You Should Now Know

- How binary plays a role in the operation of a computer
- How the binary number system works
- How to convert between decimal and binary
- Binary addition
- Binary subtraction via the complement and add technique
- How signed and unsigned overflow work