Lecture 7.2

Internal Data Representation

Why binary representation makes sense?

- Inside computers, all data is represented as 0's and 1's
- Why binary representation makes sense?
- Because …
  - The RAM is electric,
  - The disk is magnetic,
  - The communications is optical,
  - The CPU logic is true/false
- All bi-state artifacts!
- The binary system is a perfect match to the way computers work.

A RAM segment (arbitrary snapshot)

…	…
10102501	00110110001001100011011000110110
10102502	10101010000101010100101010110110
10102503	10110110001101101011011010100110
10102504	10110010001101101010011010100110
10102505	00110010000000011011011010110110
10102506	10000000000011001011011010110110
10102507	00110110001001100011011000110110
10102508	10101010000101010100101010110110
10102509	10110110001101101011011010100110
10102510	10110010001101101010011010100110
10102511	00110010000000011011011010110110
10102512	10000000000011001011011010110110
10102513	00110110001001100011011000110110
10102514	10101010000101010100101010110110
10102515	10110110001101101011011010100110
10102516	10110010001101101010011010100110
…	…

Basic Program Elements: lecture outline

- Number systems
  - Binary
  - Octal
  - Hexadecimal
  - Decimal
- Internal representations of
  - Positive integers
  - Negative integers
  - Fractional numbers
- Internal representations of
  - Characters and strings
  - Images
  - Music
  - Other media types.

Number systems (well, some of them)

unary	decimal	binary	3-bit register
	0	0	000
*	1	1	001
**	2	10	010
***	3	11	011
****	4	100	100
*****	5	101	101
******	6	110	110
*******	7	111	111
********	8	1000	overflow
*********	9	1001	overflow
**********	10	1010	overflow

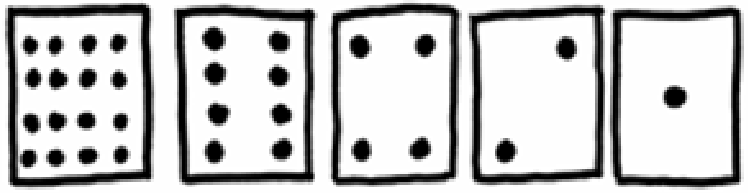
How many things can you represent with two signs?

1 bit	2 bit	3 bit	4 bit
0	00	000	0000
1	01	001	0001
	10	010	0010
	11	011	0011
		100	0100
		101	0101
		110	0110
		111	0111
			1000
			1001
			1010
			1011
			1100
			1101
			1110
			1111

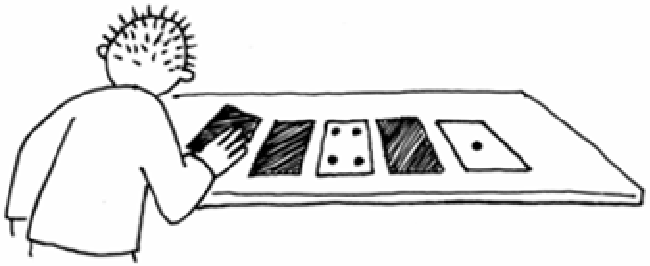
- A binary system based on *n* bits can represent 2^n different things
- Each time we add a bit to the representation, we can represent twice as many things
- We can use these bit combinations to create agreed-upon codes of …
  - Numbers
  - Characters
  - Images
  - Etc.

The binary system

Let's use cards to represent numbers:

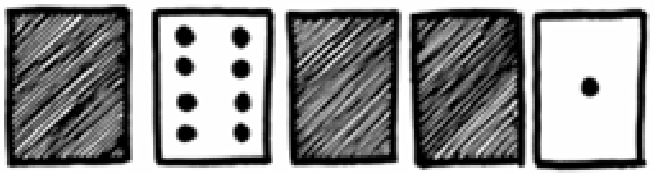


Using the cards to represent numbers



- What is the smallest number that we can represent?
- The largest?
- Can we represent any number within this range?
- Is the representation unique?
- How can you tell that a number is odd / even?

Binary representation of 9



0 1 0 0 1 = 9

- How to convert binary to decimal?
- How to convert a decimal to binary?
- How to add 1 to a given number?

The binary system

- Computing the decimal value of a binary number:

$$(10011)_{two} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19$$

- The general scheme:

$$(x_n x_{n-1} \ldots x_0)_b = \sum_{i=0}^{n} x_i \cdot b^i$$

- This scheme works with any given number system with base b:

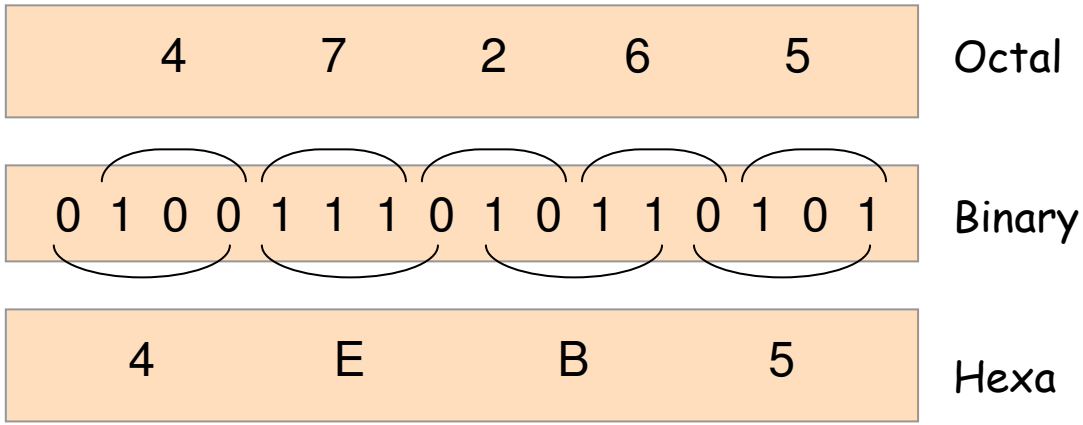
$$(9038)_{ten} = 9 \cdot 10^3 + 0 \cdot 10^2 + 3 \cdot 10^1 + 8 \cdot 10^0 = 9038$$

Basic Program Elements: lecture outline

- Number systems
  - Binary
  - Octal
  - Hexadecimal
  - Decimal
- Internal representations of
  - Positive integers
  - Negative integers
  - Fractional numbers
- Internal representations of
  - Characters and strings
  - Images
  - Music
  - Other media types.

Binary, octal, hexadecimal

Binary	Octal	Hexa
0	0	0
1	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
		9
		A
		B
		C
		D
		E
		F



- Binary-octal-hexa conversions are easy
- It is customary to display internal representation of data inside computers using octal or hexa.

Examples where octal and hexa are used in computing

Unix file permissions

(r = read, w = write, x = execute)

0	---	no permission
1	--x	execute
2	-w-	write
3	-wx	write and execute
4	r--	read
5	r-x	read and execute
6	rw-	read and write
7	rwx	read, write and execute

(Owner, Group, Others)

777	-rwxrwxrwx	rwx for all
754	-rwxr-xr--	rwx for owner, rx for group, r for others
124	---x-w-r--	x for owner, w for group, read for others

Unix internal representation of file permissions (octal)

URLs

```
http://www.example.com/name%20with%20spaces
```

(Two unrelated examples)

The hexa value `%20` is `32` in decimal, representing "space" in ASCII

Examples where octal and hexa are used in computing

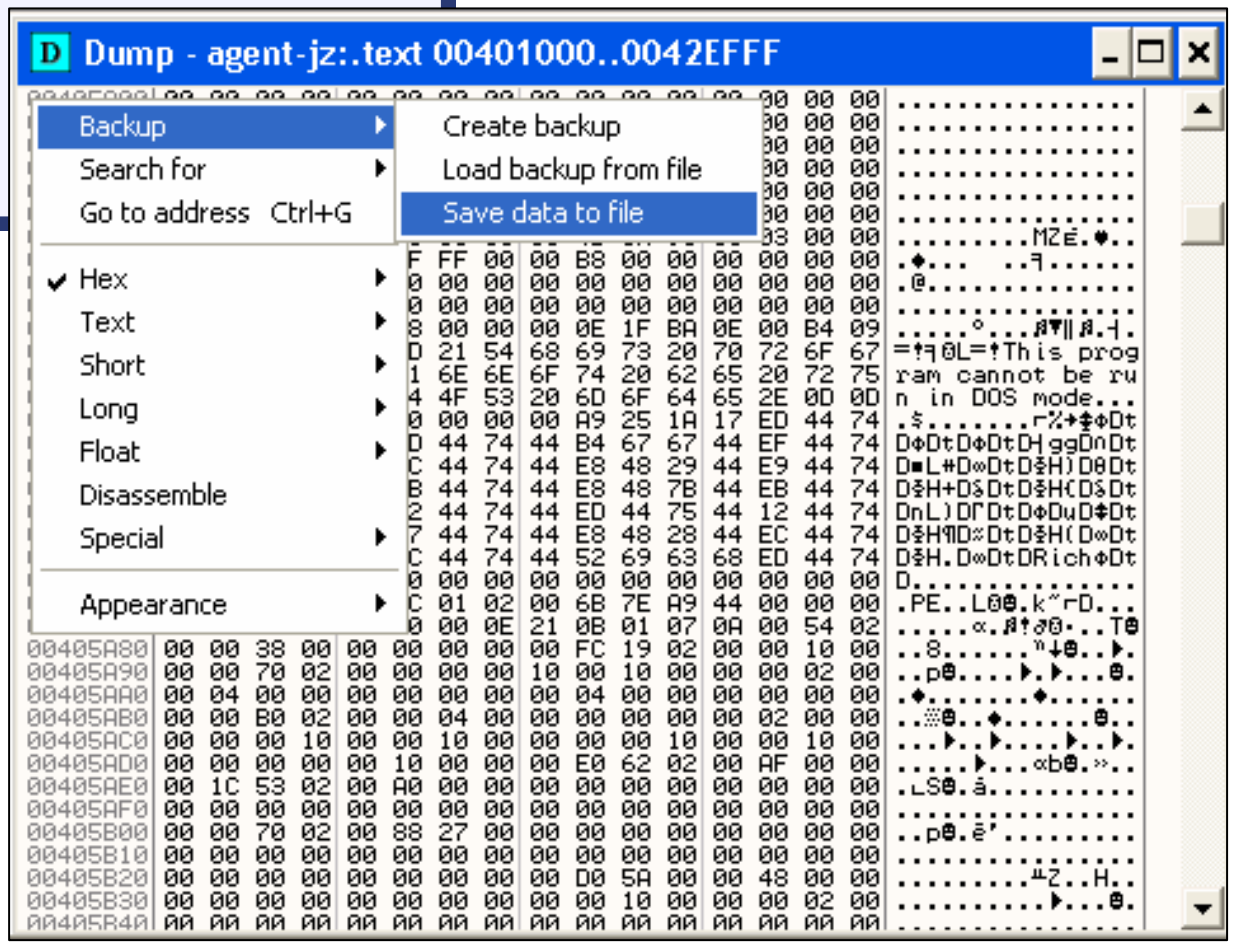
```
public class Bla {
  public static void main(String[] args) {
    Foo f = new Foo();
    System.out.println("Value of f variable: " + f);
  }
}

class Foo {
}
```

```
C:\WINDOWS\system32\cmd.exe

D:\demo>javac Bla.java

D:\demo>java Bla
Value of f variable: Foo@de6ced
```



Memory dump = a snapshot of a memory segment

Sometimes used for hard core debugging, or hacking

Memory addresses and contents are usually shown in hexa.

(The Java program and the memory dump are unrelated)

Basic Program Elements: lecture outline

- Number systems
  - Binary
  - Octal
  - Hexadecimal
  - Decimal
- Internal representations of
  - Positive integers (done)
  - Negative integers
  - Fractional numbers
- Internal representations of
  - Characters and strings
  - Images
  - Music
  - Other media types.

Representing negative numbers

Assuming 4-bit codes:

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

2 - 5 = 2 + (-5) =

```
  0 0 1 0
+ 1 0 1 1
---------
  1 1 0 1  = -3
```

The 2's complement system:

- Divide all possible codes to two subsets
- Assign the codes that begin with `0` to positive numbers
- Assign the codes that begin with `1` to negative numbers
- To negate a number:
  leave all trailing `0`'s and first `1` intact;
  flip all the remaining bits

Implications:

We know how to represent
`byte`, `short`, `int`, `long`, and `char`

Basic Program Elements: lecture outline

- Number systems
  - Binary
  - Octal
  - Hexadecimal
  - Decimal
- Internal representations of
  - Positive integers
  - Negative integers
  - Fractional numbers
- Internal representations of
  - Characters and strings
  - Images
  - Music
  - Other media types.

Representing fractional numbers

Regular representation:	532.81	0.00791
Floating point representation:	$53281 * 10^{-2}$	$791 * 10^{-5}$
Normalized:	$5.3281 * 10^{-4}$	$7.91 * 10^{-3}$

Significand **Exponent**

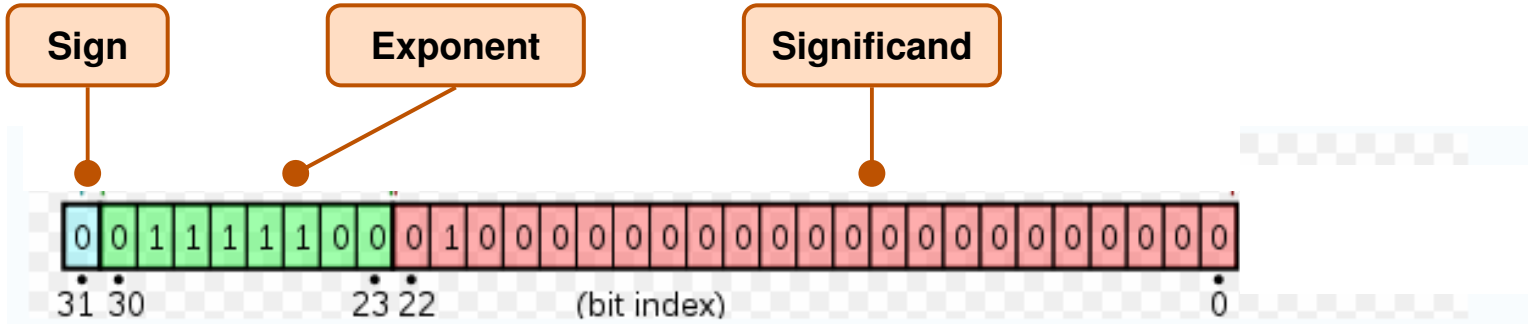
- In normalized form, there is only one digit before the decimal point, and this digit must not be zero. The decimal point "floats" to affect the necessary correction
- Thus, we can represent fractional numbers as pairs of two numbers: the significand and the exponent

<u>Floating point arithmetic</u> (example):

```
123456.7 + 101.7654 = (1.234567 * 10^5) + (1.017654 * 10^2)
                    = (1.234567 * 10^5) + (0.001017654 * 10^5)
                    = (1.234567 + 0.001017654) * 10^5
                    = 1.235584654 * 10^5
```

All arithmetic operations are done using floating point representation.
The numbers are converted to decimals only when we wish to read or write a value.

Floating point in binary



value (32 bit sequence) = $(-1)^{\text{sign bit}} * \text{significand} * 2^{\text{exponent}}$

Each bit in the significand represents a value that halves, as follows:

`bit 22 = 0.5, bit 21 = 0.25, bit 20 = 0.125, bit 19 = 0.0625 ...`

This may lead to slight accuracy anomalies.

<u>Implications:</u>

This is how `float` and `double` are represented.

Basic Program Elements: lecture outline

- Number systems
  - Binary
  - Octal
  - Hexadecimal
  - Decimal
- Internal representations of
  - Positive integers
  - Negative integers
  - Fractional numbers
- Internal representations of
  - Characters and strings
  - Images
  - Music
  - Other media types.

Representing non-numeric data

The data on the right can be part of a

- Spreadsheet
- Word document
- JPG Image
- Video stream
- Medical record
- Operating system code
- Java code
- . . .

<u>So what is it?</u>

- There's is no way to tell
- The semantics of the data makes sense only in the context in which we use it.

A RAM segment (arbitrary snapshot)

. . .	. . .
10102501	00110110001001100011011000110110
10102502	10101010000101010100101010110110
10102503	10110110001101101011011010100110
10102504	10110010001101101010011010100110
10102505	00110010000000011011011010110110
10102506	10000000000011001011011010110110
10102507	00110110001001100011011000110110
10102508	10101010000101010100101010110110
10102509	10110110001101101011011010100110
10102510	10110010001101101010011010100110
10102511	00110010000000011011011010110110
10102512	10000000000011001011011010110110
10102513	00110110001001100011011000110110
10102514	10101010000101010100101010110110
10102515	10110110001101101011011010100110
10102516	10110010001101101010011010100110
. . .	. . .

Representing characters and strings

```
public class Charade {
  public static void main(String[] args){
    System.out.println();
    String s1 = "It\'s just";
    String s2 = "a huge, incredible";
    String s3 = "charade.";
    System.out.println(s1+" "+s2+" "+s3);
  }
}
```

```
C:\WINDOWS\system32\cmd.exe

D:\demo>java Charade

It's just a huge, incredible charade.
```

There's a big difference between the way the data is rendered to our senses, and the way the data is stored in memory

This should not be surprising.

Symbol table

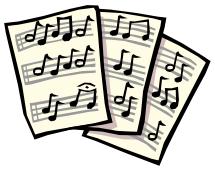
symb	addr	type
s1	10096	String
s2	10097	String
s2	10098	String

RAM (32-bit)

...				
10096				70603
10097				70606
10098				70611
...				
70603	9	73	116	96
70604	115	32	106	117
70605	115	116		
70606	18	97	32	104
70607	117	103	101	44
70608	32	105	110	99
70609	114	101	100	105
70610	98	108	101	
70611	7	99	104	97
70612	114	97	100	101
...				

Representing other media types

Text
(e.g. Ascii)



Sound
(e.g. MP3)

Images
(e.g. JPG)

Programs
(e.g. Bytecode)

From an agreed-upon, media-specific standard representation to bits

From bits to an agreed-upon, media-specific standard representation

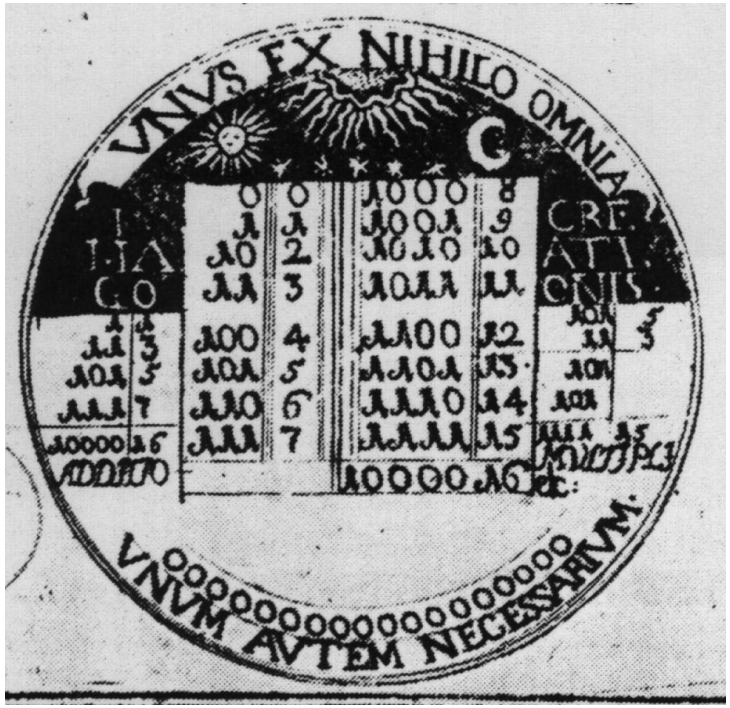
RAM

0011011000100110001101100011011 0
1010101000010101010010101011011 0
1011011000110110101101101010011 0
1011001000110110101001101010011 0
0011001000000000110110110101101 10
1000000000001100101101101011011 0
0011011000100110001101100011011 0
1010101000010101010010101011011 0
1011011000110110101101101010011 0
1011001000110110101001101010011 0
0011001000000000110110110101101 10
1000000000001100101101101011011 0

- The conversions are performed only when data is acquired or rendered
- The conversions are typically performed by programs running on device drivers.

On the shoulders of giants

Leibniz's medallion:



Goittfried Leibniz
(1646-1716)

- Leibniz complained that "All who are occupied with the reading or writing of scientific literature have assuredly very often felt the want of a common scientific language, and regretted the great loss of time and trouble caused by the multiplicity of languages employed in scientific literature"
- Leibniz dream: *Characteristica Universalis*": a universal, formal, language of reasoning in which any assertion can be proved or disproved automatically
- The dream's end: Turing and Goedl in 1930's.