

Lecture 1

Number Systems and Binary Arithmetic

Text: Chapter 1

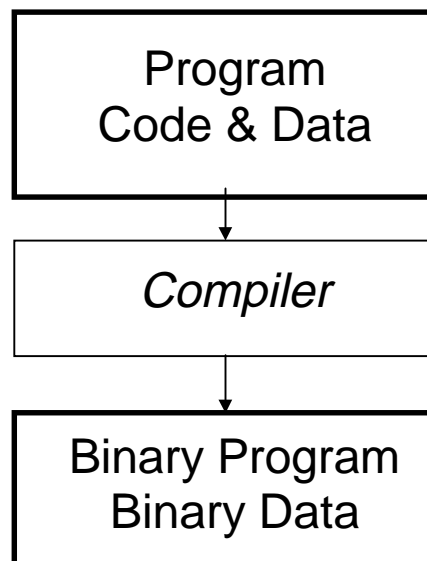
What is inside a computer that makes it work?

Important fact:

All information in the machine is in
BINARY

Binary is a number system composed of only ones and zeros ... 1001010110101001001001010101011011010

Programming has been made easy by compilers...

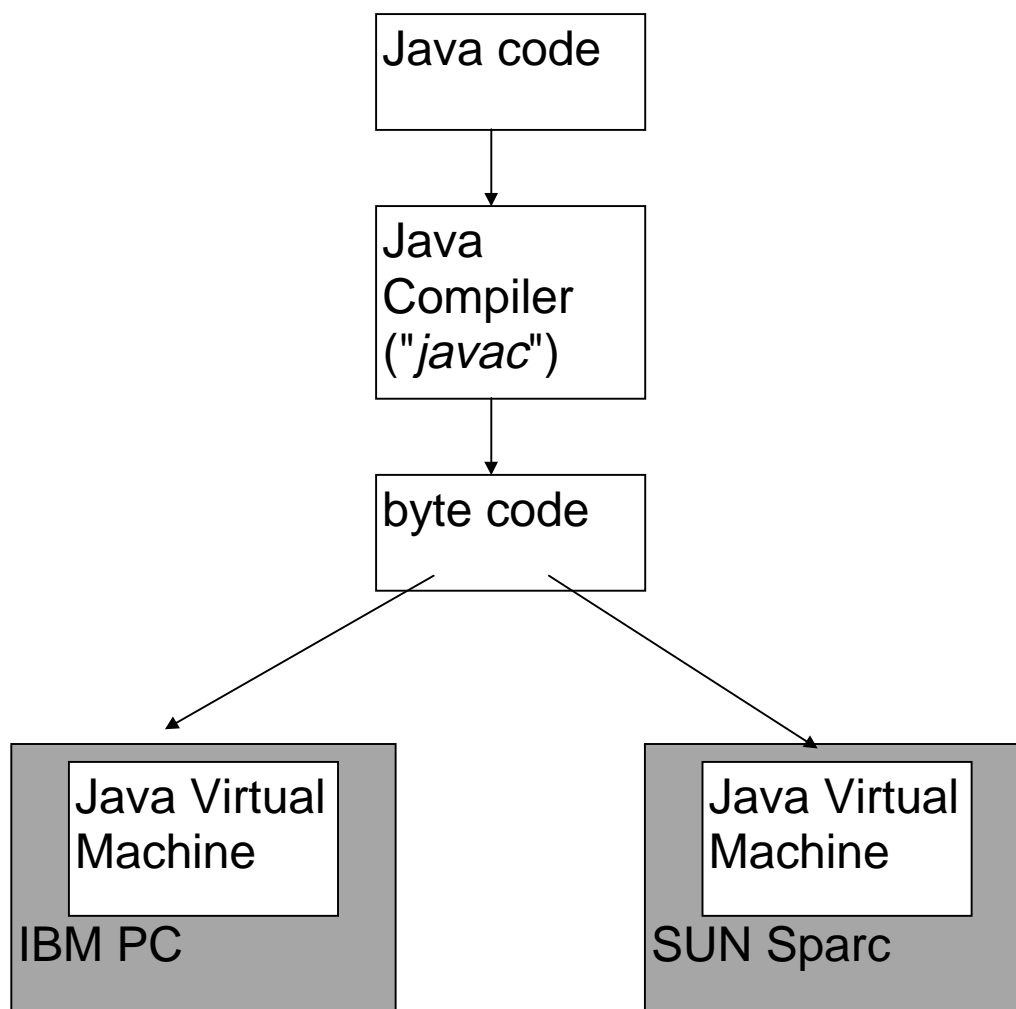


So, the components inside a computer contain **ONLY** binary information, and the computer can process **ONLY** binary information.

The Java model

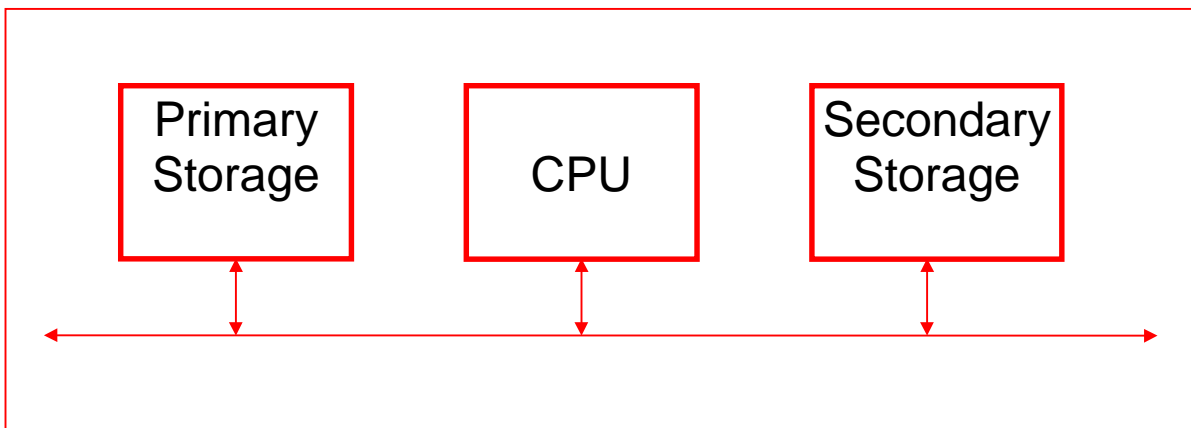
Because Java classes are platform-independent, it is up to the JAVA VIRTUAL MACHINE to interpret the binary byte code.

The JVM is **not** platform-independent. It must be a binary program which can be understood by the machine on which it is running.



What are the components of a Computer?

- Central Processing Unit
 - Program instruction execution.
 - Arithmetic
 - Decision making and program flow
- Primary Storage (Internal Memory)
 - Binary numbers only which may be either program instructions or program data.
- Secondary Storage
 - Long-term storage such as disk and tape.



All three units are connected by the **System Bus**

The Positional Number System

Representation of a number:

The value of a number is the sum of each digit when multiplied by the base raised to a power.

The power is a matter of the position of the number.

Positions: ..4 3 2 1 0 -1 -2 -3 -4 ...

Number: 1 2 3 . 4

$$123.4 = 1 * 10^2 + 2 * 10^1 + 3 * 10^0 + 4 * 10^{-1}$$

The digits are drawn from a set according to the base:

Base 10 {0,1,2,3,4,5,6,7,8,9}
Decimal

Base 2 {0,1}
Binary

Base 8 {0,1,2,3,4,5,6,7}
Octal

Base 16 {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
Hexadecimal

Conversion from Base N to Base 10

Method of Literal Expansion

The number can be represented as an expression using the sum of each digit multiplied by the base raised to a power which is the digit's position.

Simply solve that expression to determine the base 10 value of the number.

Example:

$$100101.1_2 = 2^5 + 2^2 + 2^0 + 2^{-1} = 32 + 4 + 1 + .5 = 37.5$$

$$7B4_{16} = 7 * 16^2 + B * 16^1 + 4 * 16^0 = 1792 + 176 + 4 = 1972$$

Conversion from Base 10 to Base N

Method of Division (For Integer Numbers)

Repeat

Divide the base 10 number by the new base;

Save the remainder;

Keep dividing the quotient

Until the quotient is zero.

The answer is found by reading the remainders last to first.

Example: Convert 297_{10} to Base 16

$$297 / 16 = 18 \text{ R. } 9$$

$$18 / 16 = 1 \text{ R. } 2$$

$$1 / 16 = 0 \text{ R. } 1 \text{ The answer is } 129_{16}$$

Method of Multiplication (For Fractional Numbers)

Repeat

Multiply base 10 number by the new base;

Record the integer portion;

Keep multiplying the fractional part

Until the fractional part is zero or
the desired accuracy is reached.

The answer is the integer parts read first to last.

Example: Convert $.203125_{10}$ to base 16:

$$.203125 * 16 = 3.25$$

$$.25 * 16 = 4.0$$

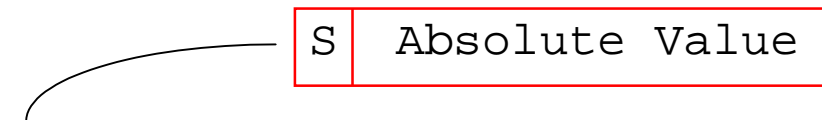
The answer is $.34_{16}$

Number Representation

To represent both positive and negative numbers.

Sign/Magnitude

Using an n-bit location, set aside the leftmost bit to record the sign, and use the right n-1 bits for the absolute value of the number.



S: 0 means positive

1 means negative

Examples: (using 8 bits)

```

3:  00000011
-1: 10000001
0:  00000000

```

It's easy to read the numbers, but doing arithmetic (using the normal rules) is difficult:

```

   7 00000111
+ -2 10000010
-----
- 9  10001001

```


One's Complement

All bits are part of the number, but the leftmost bit is still interpreted as the sign of the number. (As before, 0 means positive, 1 means negative)

Positive numbers are therefore represented by their binary values as long as the leftmost bit remains 0.

Negative numbers are represented by taking the absolute value and reversing (complementing) all the bits.

Examples: (Using 8 bits)

3:	00000011	
1:	00000001	
-1:	11111110	(1 is 00000001)
-18:	11101101	(18 is 00010010)

Arithmetic may be done directly in binary:

$$\begin{array}{r}
 7 \quad 00000111 \\
 + \quad -2 \quad 11111101 \\
 \hline
 1 \quad 00000100
 \end{array}$$

the answer is 5

carry
bit

The carry bit must be added to the "answer". This is called "End-around-carry".

Two's Complement

Positive Numbers are represented by their normal binary form and have a leftmost bit which is zero.

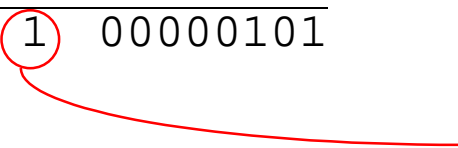
Negative Numbers are formed by taking the absolute value, forming the one's complement and adding one:

Example: Form the two's complement representation of -7 using 8 bits:

00000111	Positive 7
Form the one's complement:	
11111000	
+1	add one
11111001	Two's complement

Arithmetic is also simpler as there is no need for end-around-carry:

7	00000111	
+ -2	11111110	add negative 2
1	00000101	the answer is 5
		ignore carry bit



Range of Numbers

How large and small a number can be represented in a finite location of n bits?

With n -bits you can represent 2^n unique bit patterns, ranging in value from $0.. 2^n - 1$

Example:

With three bits, you can represent $2^3 (=8)$ bit patterns, numbered $0.. 2^3 - 1$ (i.e., $0..7$):

000, 001, 010, 011, 100, 101, 110, 111

Note that if we consider these numbers as represented in two's complement, half are positive (000, 001, 010, 011) and half are negative (100, 101, 110, 111).

$$\frac{2^n}{2} = 2^{n-1}$$

The range for sign/magnitude and one's complement becomes:

$$-(2^{n-1} - 1) .. 2^{n-1} - 1$$

And for two's complement (because there is no representation of negative zero) becomes:

$$(-2^{n-1}) .. 2^{n-1} - 1$$

Carry and Overflow

If two numbers are combined arithmetically (for example, by addition) and the result is incorrect because it is out of range an overflow is said to occur.

Overflow can be detected by examining the carry in to the leftmost column of addition and the carry out (the $n+1$ st bit).

Overflow \Leftrightarrow carry in \neq carry out

Examples:

$ \begin{array}{r} \textcircled{1}111110 \leftarrow \text{carry in} \\ +7 \quad 00000111 \\ + -2 \quad \underline{11111110} \\ \textcircled{1} \quad 00000101 \\ \text{carry out} \end{array} $	<p>carry in = carry out no overflow answer is correct</p>
---	---

$ \begin{array}{r} \textcircled{0} \\ -6 \quad 1010 \\ + -4 \quad \underline{1100} \\ \textcircled{1} \quad 0110 \end{array} $	<p>overflow</p>
---	-----------------

$ \begin{array}{r} \textcircled{1} \\ +6 \quad 0110 \\ ++4 \quad \underline{0100} \\ \textcircled{0} \quad 1010 \end{array} $	<p>overflow</p>
--	-----------------

Exercises - Lecture 1

1. Convert to base 10:

a) 10101b _____

b) 3D2h _____

c) A63Dh _____

2. Convert the base 10 number to the base in parenthesis:

a) 123 (16) _____

b) 54 (2) _____

3. Convert to the base in parenthesis:

a) 3479FB22h (2) _____

b) 100010010b (16) _____

4. Show the 8-bit Sign/Magnitude, One's Complement and Two's Complement:

a) 15 _____

b) -5 _____

c) -24 _____

5. Perform the following calculations (presuming 2's complement representation):

a) 00010011	b) 00010011	c) 3D72	d) 40D0
<u>+00001010</u>	<u>-00001010</u>	<u>+0E78</u>	<u>-0EB8</u>

6. Determine if each 4-bit two's complement calculation results in overflow (hint: for subtraction, ADD the two's complement):

a) 0011	b) 0011	c) 0100	d) 0100	e) 1000	f) 0110
<u>+0010</u>	<u>-0010</u>	<u>+0100</u>	<u>-0100</u>	<u>+0110</u>	<u>+0100</u>