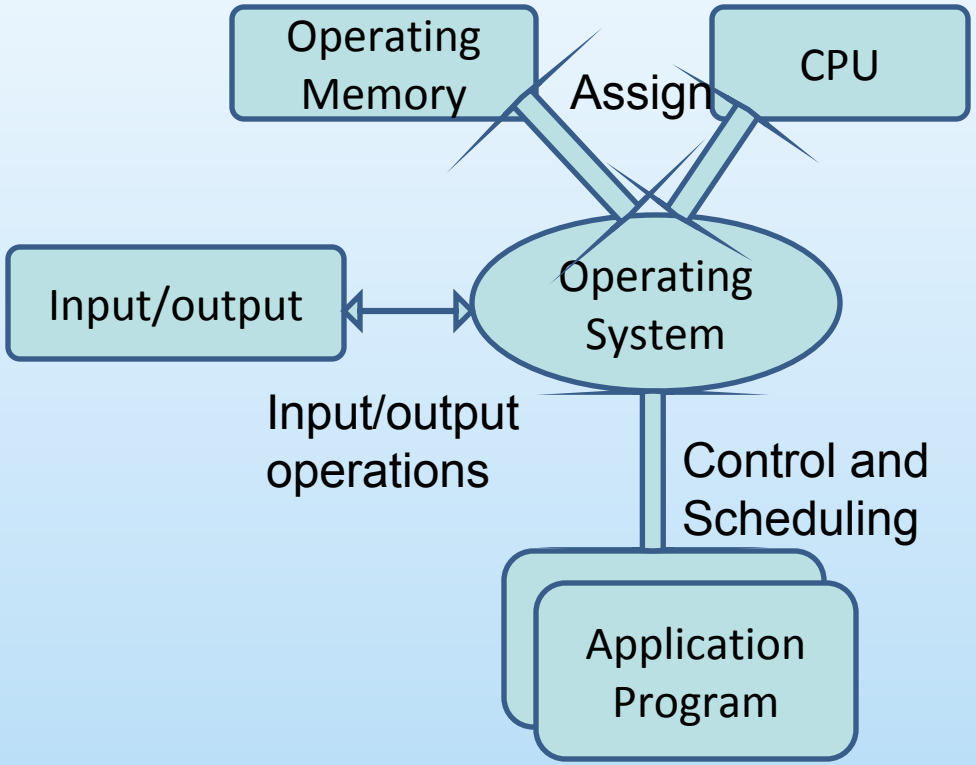
Personal Computer Devices

Computing Program, Operating System and Computing Resources

- Computing Program – sequence of the commands executed by computing unit
- Application Program – computing program running for some application purpose (e.g. Power Point, Game, etc.)
- Operating System – computing program intended for
  - Scheduling of application programs
  - Control of application programs
  - Resources (memory, CPU) allocation for application programs
  - Performing of the Input/output operations for application programs



Operating Memory and CPU are the computing resources

2

Information Basics

- Minimal information unit is one bit with two possible values: 0, 1
- Minimal information unit addressable in as operating memory as in external storage is byte
- One byte contains 8 bits with 2^8 (256) possible combinations, that is numbers as one combination may present one number
- 1024 bytes = 1 Kilobyte (KB)
  1024 Kilobytes = 1 Megabyte (MB)
  1024 Megabytes = 1 Gigabyte (GB)
  1024 Gigabytes = 1 Terabyte (TB)
- Non-programmer says that there are 1000 bytes in one Kilobyte and programmer says that there are 1024 meters in one kilometer
- How many combinations may be presented in 1 TB: only $2^{1024*1024*1024*1024*8}$ = ??? ; 1 TB = 262144 books “War and Peace”

3

Number Systems Conversion

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

Formula for conversion from binary to decimal

$2^{n-1} + 2^{n-2} + \ldots + 2^{0}$

Where n – number of digits

Method for conversion from decimal to binary

-Better to use calculator ☺

- Sequential division of decimal number on 2 until the last division result is 1 ☹

- 1 concatenated with sequence of all remainders from last to first forms the binary presentation

Think of decimal number 4 with binary presentation 100

Hexadecimal Numbering System

Decimal	Binary	Hexadecima
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Think of how to convert binary number to hexadecimal and vise versa

?

Operating Memory

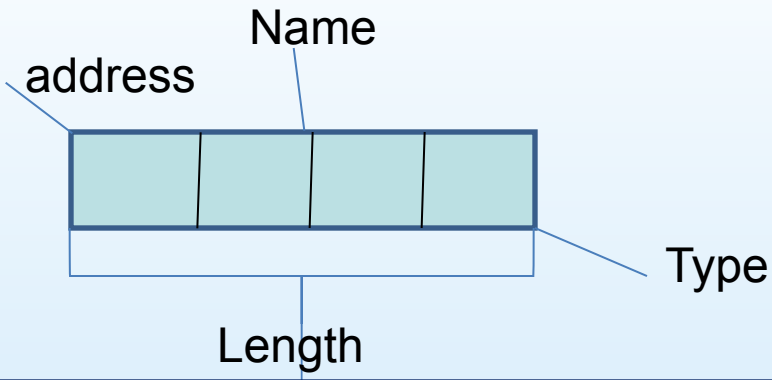
Address 0

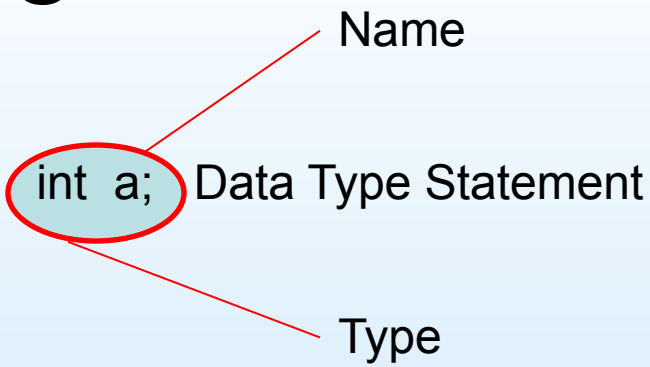
10011011	10011011	10011011	00101110	00101100	00101111	00101101	11011011	01111000
00101101	10011011	10011011	00101110	00101100	00101111	00101101	11011011	01111000
00101101	10011011	10011011	00101110	00101100	00101111	00101101	11011011	01111000
00101101	10011011	10011011	00101110	00101100	00101111	00101101	11011011	01111000
00101101	10011011	10011011	00101110	00101100	00101111	00101101	11011011	01111000
00101101	10011011	10011011	00101110	00101100	00101111	00101101	11011011	01111000
11011011	01111000	11011011	01111000	11011011	01111000	11011011	01111000	11011011
11011011	01111000	11011011	01111000	11011011	01111000	11011011	01111000	11011011

Address 1073741824

6

Variables





Variable – piece of operating memory defined in the program text by a programmer with type and name

Type defines length (number of bytes) and operations which may be performed with data containing in the piece of operating memory

Name defines reference to the piece of operating memory which will be converted to an address

int	integer numbers, length depends on CPU (usually 4 bytes) `int a = 5;`
char	single character, length of one byte `char c = 'a';`
float	floating-point number (value containing decimal places) `float pi = 3.1415926;`

Function printf – print formatted

Console – logic device with input/output

By default, console input matches keyboard as physical device and output does display as physical device

Function printf is intended for printing on console formatted data in the human form

#include <stdio.h> //header file containing function prototype

printf (“hello world”);

printf (“first operand is %d \n second operand is %d \n result is %d”, op1, op2, result);

Some formats:

%d – for presentation of an integer number in the decimal form

%o – for presentation of an integer number in the octal form

%x – for presentation of an integer number in the hexadecimal form

%c – for symbol presentation

Escape sequences

\n – new line

\t - tabulation

Operations with integers

`+ - * /`	arithmetic operations
`%`	division remainder (5 % 2 => 1)

`++  --`	Increment, decrement
	`a++` ⇔ `a = a + 1`
	`a--` ⇔ `a = a - 1`

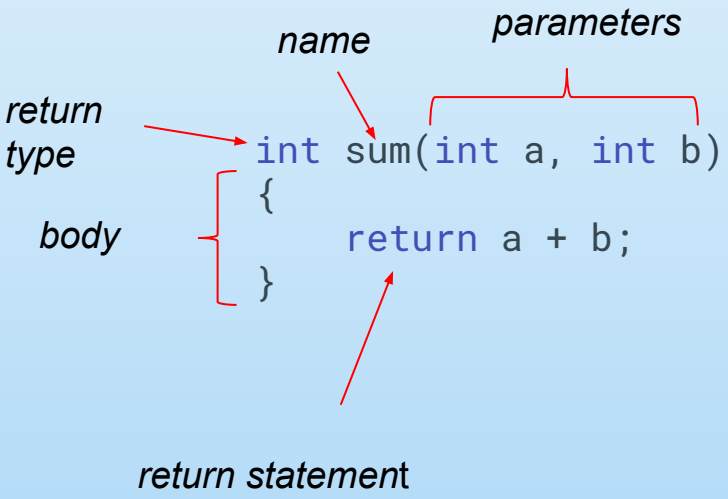
`+=, -=, /=, *=, %=`	shorthand operators
`b = a++` `b = a--`	shorthand increment and decrement in postfix form
`b = ++a` `b = --a`	shorthand increment and decrement in prefix form

Functions

A function is a set of statements that take inputs, do some specific computation and produces output.

The idea is to put some commonly or repeatedly done task together and make a function, so that instead of writing the same code again and again for different inputs, we can call the function.

Function example



The function takes two integer numbers as parameters, adds them and returns the result

Functions

Function to be **declared** before the first **function call**. Declaration is a promise to the compiler that later there will be an **implementation** of the function with the specified signature.
Signature of the function is a combination its return type, name and parameters.

```c
#include <stdio.h>
#include <stdlib.h>

int sum(int a, int b);

int main(void) {
    printf("3 + 5 = %d", sum(3, 5));
    return EXIT_SUCCESS;
}

int sum(int a, int b)
{
    return a + b;
}
```

function declaration

function call

function implementation