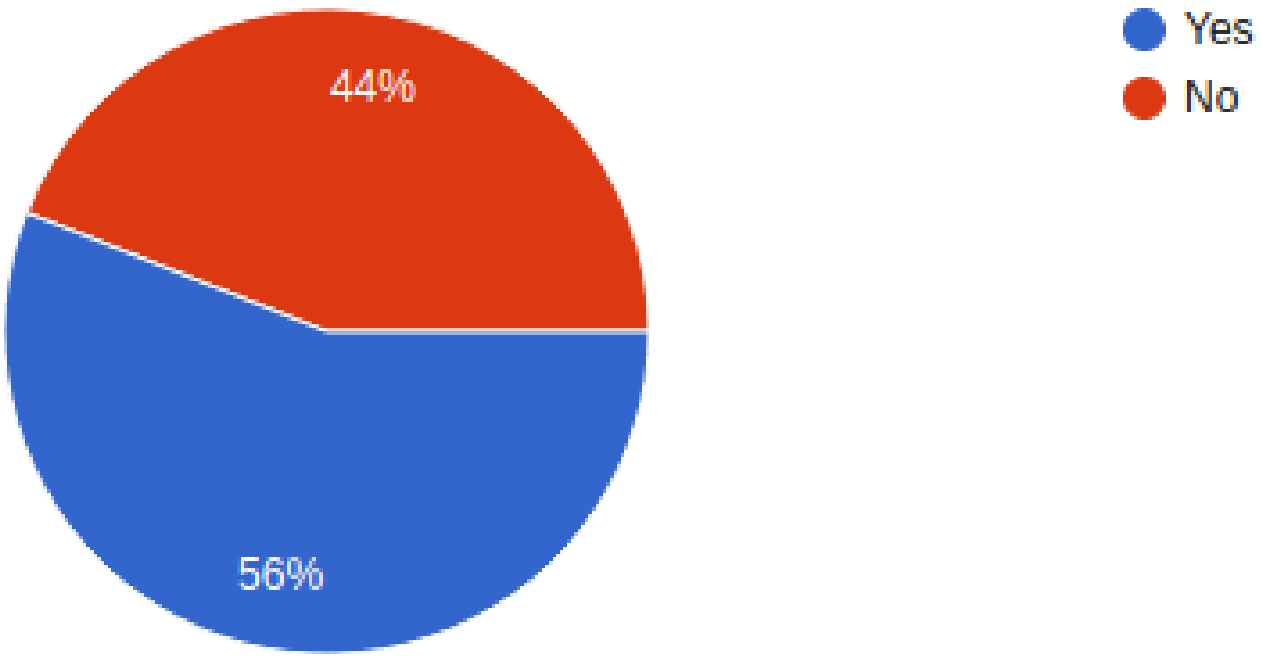
CSD101: Introduction to computing and programming (ICP)

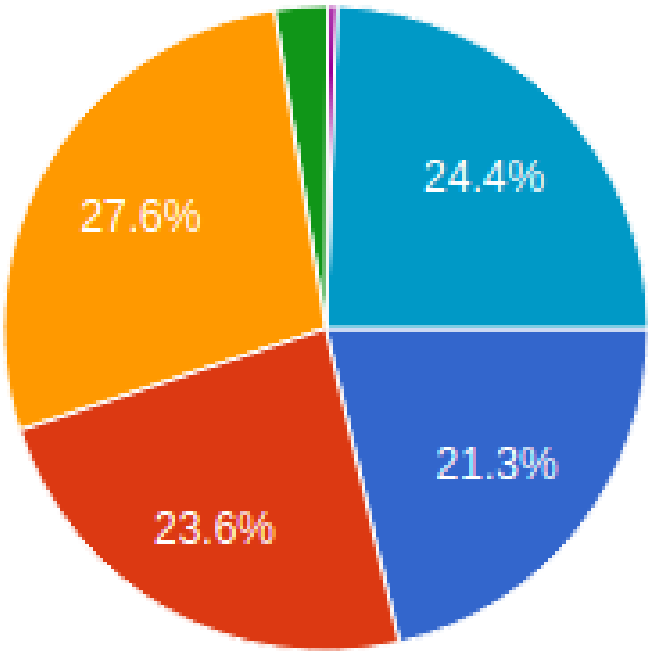
Some more organizational details

- SNU ERP (registered list for course), Blackboard - LMS for course.
  - **SNU ERP and BB are separate and you must be registered in both.**
  - If you are not registered in ERP you are not in the course and cannot be registered in BB.
  - If you are registered in SNU ERP but NOT on BB then send an email to manish.dhawan@snu.edu.in.
  - If you don't have a lab slot then choose one and send mail to ziaur.rehman@snu.edu.in.
  - For any other registration related problems contact Dean UG (deanugoffice@snu.edu.in).
- How to watch recorded lectures on Blackboard (BB)?
- How to access lab assignments/quizzes/tests on BB and upload answers?
- Lab slot assignments, MS Teams links for labs, TA assignment will be up either by tonight or on Sunday on the course website 'Labs' page.

Results of survey-1 (225 respondents)



Results of survey-2



- No programming
- Beginner level
- Intermediate level (Have written s small programs)
- Proficient level (Have done at lea large project, participate in conte
- Expert level (Have done several projects, participate in contests)
- Not applicable (answer to earlier question is No)

Computer - what is it - an abstract description

A computer is a device that:

1. Understands (that is able to represent) the following kinds of **DATA**:
   - Whole numbers both +ve, -ve - that is integers.
   - Decimal or real numbers (only approximately) due to finite representation.
   - Characters (of all kinds).
2. can carry out a sequence of operations on data. The sequence of operations is usually called a **PROGRAM**.
3. The computer has a **fixed way** to execute the program using the data as needed.
4. Both the program and the data are stored in the computer. The program is stored in consecutive locations in memory so that it can be executed in sequence.

Representing data I

- The lowest level of data representation is a **bit** (binary digit).
- A bit can be implemented by any device that can be in two states - 0 and 1.
- For use in computers bits are chunked together into **bytes** (1 byte=8 bits) and **words** (1 word = 2/4/8 bytes).
- A **register** is a sequence of bits treated as a unit where data can be stored/ accessed/ manipulated. Registers are present in the CPU. The registers in the ALU are the locus of arithmetic and logical operations.
- Data representation:
  - Whole numbers are represented in binary with an optional bit for sign.
  - Real or floating point numbers are represented by a *mantissa*+*exponent* notation and two bits for sign (one for mantissa and one for the exponent).

Representing data II

- Characters are represented by a defined mapping (called encoding). Two popular encodings are: ASCII (7-bits), Unicode (UTF-8, UTF-16, UTF-32). ASCII is a proper subset (first 128 codepoints) of UTF-8. Unicode implements all the characters of all living and dead scripts and special characters (e.g. punctuation, emojis etc.).
- Most systems (OS, websites etc) use UTF-8 and some use UTF-16. UTF-32 is rare.

- Due to finite representations. There are limits on the size and precision with which numbers can be represented.

Computer - hardware block diagram

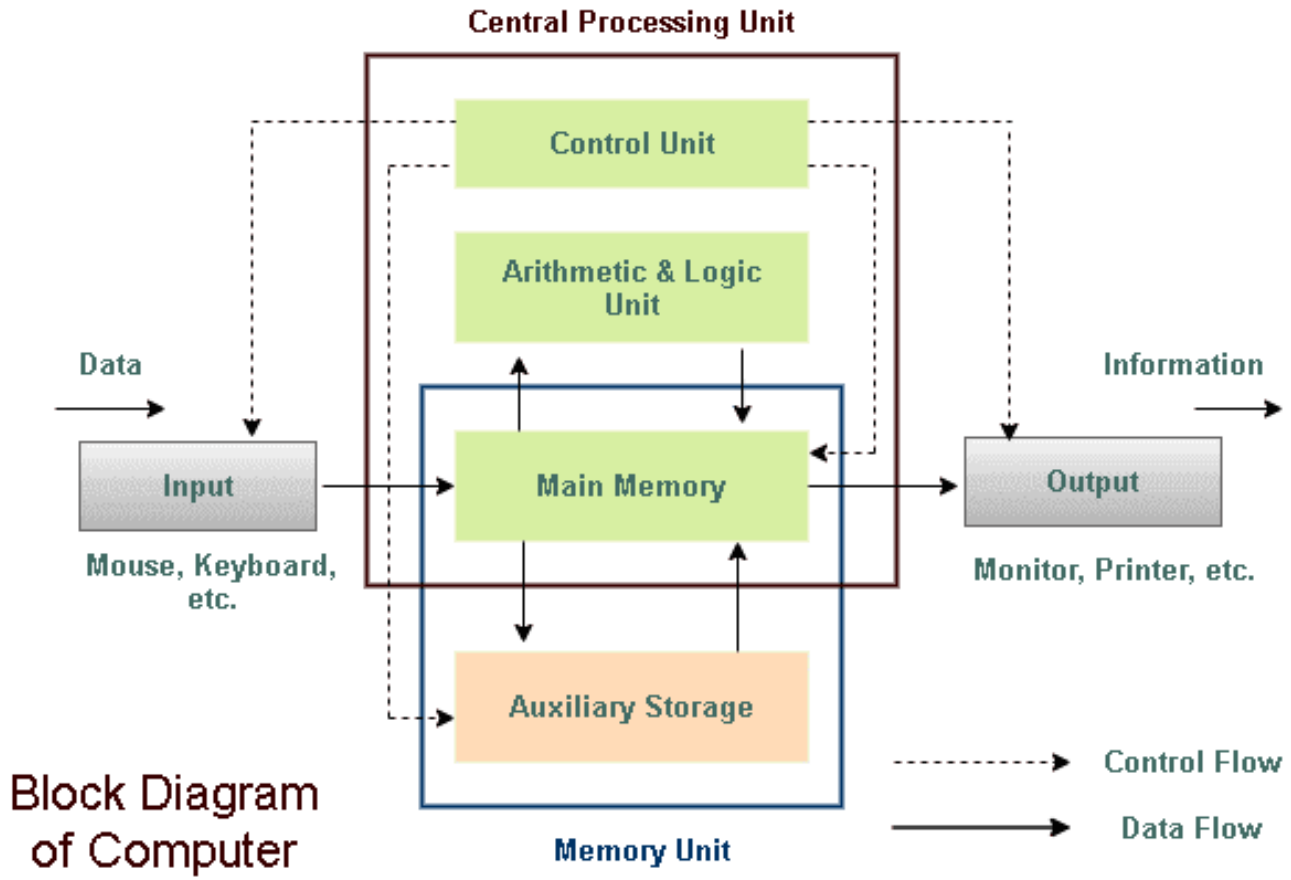


Figure: Computer hardware: logical, high-level block diagram. (src: tutorialsmate.com)

A very simple computer(VSC) I

- An **accumulator** is a register that holds one data element (i.e. an integer, real number or character (one or more depending on the encoding)). Their size is 2/ 4/ 8 bytes. Modern computers have 64-bit or 8-byte registers.
- **PC** (program counter) contains the address of the instruction to be executed next.
- Random access memory (RAM) addresses start from 1 and increase by 1 (byte). Access is typically in chunks of 4/ 8 bytes at a time.
- In principle the very simple computer is as powerful as any modern day computer.
- A VSC can be hard to program because it has a very simple, small set of operations on data (next slide).

A very simple computer(VSC) II

- For our purposes we assume that the VSC works in terms of units (decimal numbers and characters) and representations (internal representations of the current OS) that are convenient. For simplicity VSC works only with whole numbers.
- Memory size of VSC is 1000 units - addresses from 0 to 999.

Instruction set for the very simple computer I

Note: A is an address; acc stands for accumulator; @A means contents of address A; PC is program counter, N - number, S - string

Instrn code	**Instrn format**	**Meaning**
0	halt	Program halts
1	strt	Start of program
2	lodm A	acc = @A (load from memory)
3	lodn N	acc = N (acc loaded with number N)
4	lods S	acc = S (acc loaded with string S)
5	stor A	store acc at addr A
6	add A	acc = acc + @A
7	neg A	acc = -acc
8	jmp A	PC = A
9	jmp- A	if (acc<0) PC = A
10	jmp0 A	if (acc==0) PC = A
11	jmp+ A	if (acc>0) PC = A
12	inpn	acc = number read from input device
13	inps	acc = string read from input device
14	out	write acc to output device (string)
15	outl	write new line to output device

Writing code for VSC

Code for VSC is in the form of a sequence of lines of code. Each line has the following format:

Address	Operator code	Optional operand

The fields should be separated by one or more spaces. The lines of the program should always be stored in successive addresses else it is an error. To comment VSC code we use the # symbol. Anything after the # till the end of line is neglected.

Some simple problems

1. Output a message.
2. Read two numbers m and n and output the larger number.
3. Read two numbers m and n and output the product of the two numbers.
4. Read two positive integers m and n and output m^n.
5. Read two positive integers m and n and output the largest number that divides both m and n.

We write VSC code for the first two.
Problems 3 to 5 are for you to practice.

Examples of VSC programs - 1

Example 1

A VSC program that prints `Very Simple Computer`.

```
0 1
1 4 "Very Simple Computer" #load string in acc
2 14 #output contents of acc
3 15 #output newline
4 0  #halt
```

Examples of VSC programs - 2

Example 2

A VSC program that reads two numbers and prints the larger of the two numbers.

```
10 1
11 12        #Read first number - n1
12 5 500     #Store it at addr 500
13 12        #Read second number - n2
14 5 501     #Store it at 501
15 7         #n2 negated
16 6 500     #acc=n1-n2
17 11 24     #n1 is larger
18 2 501     #n2 is larger load it in acc
19 14        #output acc
20 4 " is larger or equal."     #load string
21 14        #output string
22 15        #output newline
23 0         #halt
24 2 500     #n1 is larger load it in acc
25 8 19      #jump to code at addr 19
```