12/1/22

CS251- System Programming

Processor   Memory   I/O

these two gets merged
All these are discussed in COA

Processor
Memory
interaction b/w them

We don't use "CPU" ^the word in these days. We usually ^take the word "processor".

→ Earlier it used to be like centralised ~~processory~~ processing, now we have processing task of computer system at geographically distributed locations. That's why the word central doesn't have much significance.

We studied
- Data path
- Control path
- Various aspects of Memory design
  - RAM
  - Secondary Memory

. Modern Computers ← havae lots of advance capabilities
- Fast processors
- large memory
- Good Network support
- Sofisticated I/O.

All ^these things are possible only by

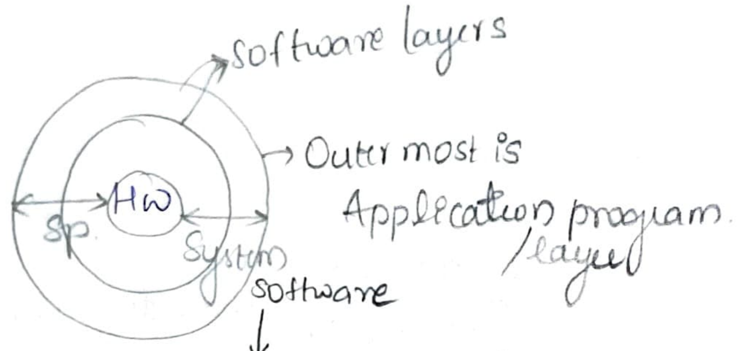
Instructing a computer using → Machine language
↓
00010011000
0's and 1's.

{ GAP btw interaction with computer : [interaction btw hardware and computer (M/c L)]

Expectation of computer system.

Bridging the Gap is carried out by "System Software".

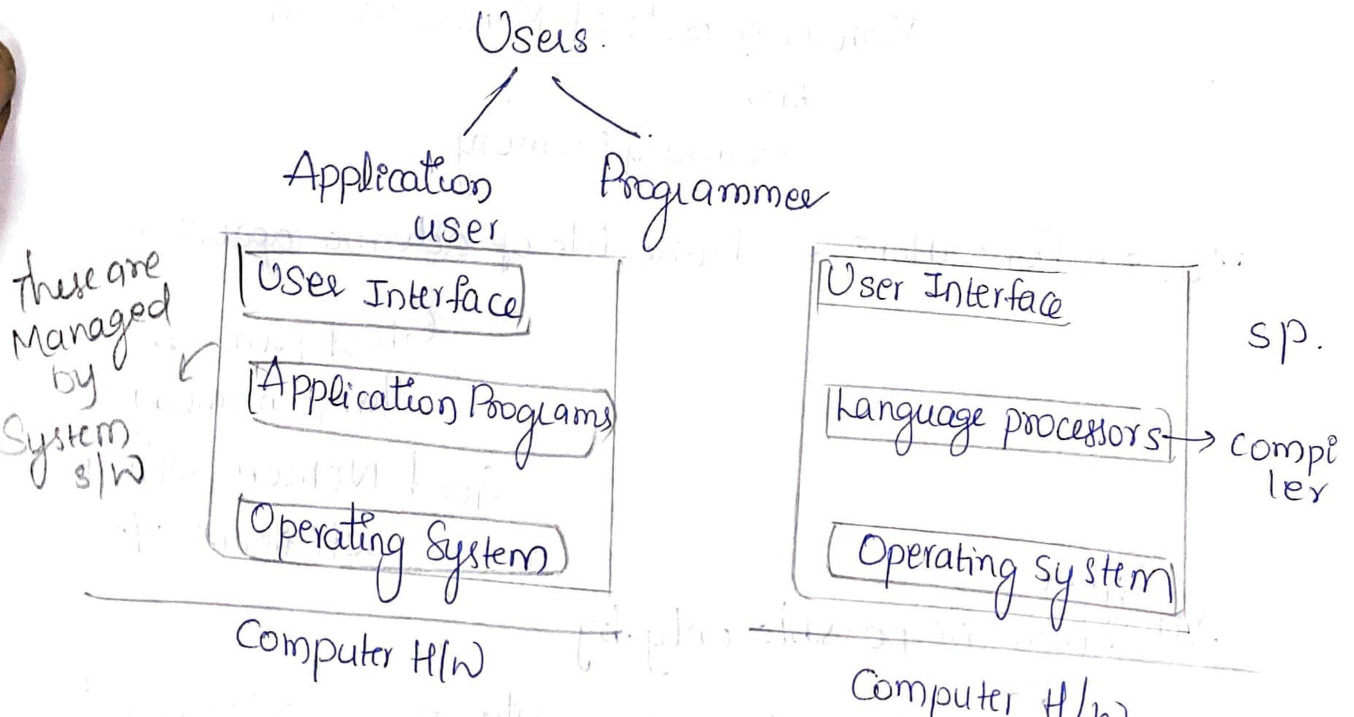
Layered view
↓
Hardware
System software



immediately encircles the hardware are System software or System program

What system programs do?

① Translate the read of user to → Machine language
② Manage the resources of computer system
↳ Memory
I/O
processor.



Goals of system program

① User Convenience
② Efficient use of resources
③ Non Interference.

① User Convenience :

- Earlier We need devece to perform huge calculations - Deal with Numbers
- HLL → High level language programs have been executed now.
  - ↳ User wanted better services ↑
    - Command Line interface
    - Graphical user interface

② Efficient use

- Resources
  - Memory
  - Processor
  - Disks
  - I/O

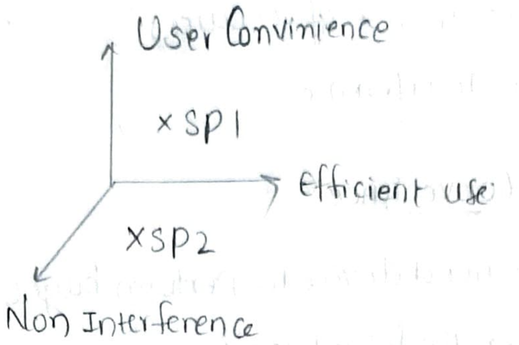
③ Non-interference

- Suitable security functions.
- Protective functions need to be implemented

- Classical stand-alone environments
  - Authentications {Password based}
- Computers - Connected to Internet
  - Security threats.
    - Trojan horse
    - Virus
    - worms

To reduce these interference effects,

OS → through various means.



SP → Ⓐ Ⓑ which is good/better/preferred?

↳ Difficult Question to answer

- It doesn't have unique answer
- A may be prefer over B under certain circumstances and viceversa

↓

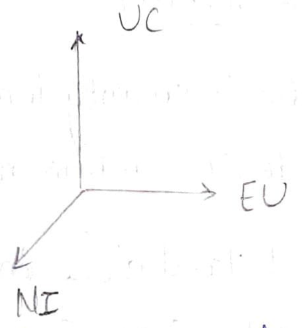
Factors → To Answer the question we need to know the factors.

① Program Development and production environment

19/1/22

System Software

- User convenience
- Efficient use
- Non-interferance



Though we have such measures, if someone ask how do we Compare btw system softwares.

It depends on many factors.

Factors :

① Program Development and production environment

We can imagine every system software as a point in 3-dimension and evaluate.

① Compiler

- Translate :
  HLL → ML
  ↓
  ready for execution

- It analysis each statement in HLL.

It has typically two phases.

Phase 1 : program is compiled

Phase 2 : ML instructions are generated (code generation)

When we use loops, statements in the loop are only once analysed.

② Interpreter

- Does not generate ML program
- ~~A~~ It analysis the program P, and directly carries out the desired computation.
  ↓
  It keeps track of sequence in which program 'p' gets executed

→ Every time it analysis entire loop

* During program development it is better to use interpreter than in production environment

③ Debugger
- Stepwise.

→ Interactive debugging
where we can even set break points in the program

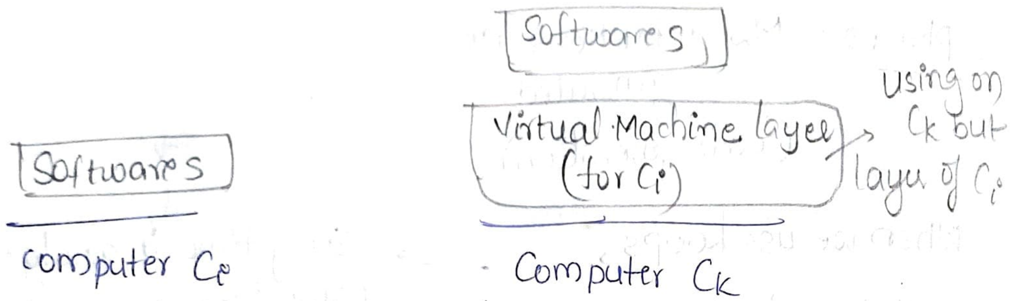
② Making a software portable
↓
possible to execute in every a computing environment other than where it developed.

if a   made
- Program →is, of special features provided by OS or specific computer
then it is difficult to make it portable.

→ HLL
helps to make portable.

Virtual Machine concept :

- Convenient method to implement portability
- VM is an Abstract computer that has all desired set of features



- Ci, Ck are of diff environments
- I want the software S to which is developed in Ci, to run on Ck without any modification
- It is realised/run on software layer as shown.
↓
portability achieved using VM causes overhead.

→ Pascal programming : 1970's (developed in)
to perform systematic programming
Programs written in pascal are portable.
Virtual machine for pascal is specifically designed.

If you take a pascal compiler
- it will generate code for pascal VM
Called Pcode

→ Java programming :

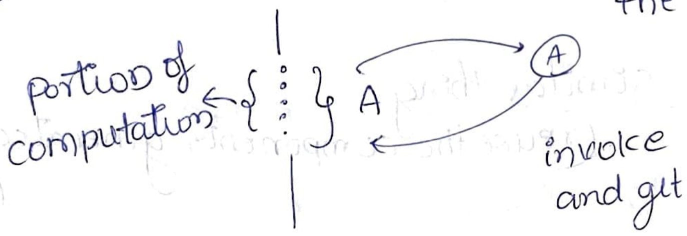
↳ JVM (executed on)
(Java Virtual Machine)

Java program when compiled, Java bytecode is generated

Java byte code is portable. (the program which is obtained after compilation is portable)

③ Realizing benefits of the Internet

- Programs - located on remote computers and integrate the result to the present program

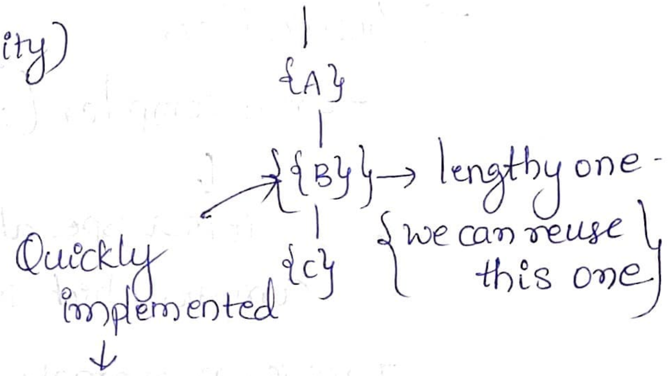


- Download → Unknown program.
↳ but in doing so, Danger of interferance.

- Web server → gives Dynamic data
↓
time varying data

④ Treating programs as components

→ Reuse. (gives this facility)



↓

we require some kind of support

This scenario is scene in deep learning.
This support is given by Scripting languages like UNIX shell script

We have a task :- Counting unique names in a file named alpha

```
cat alpha | sort | uniq | wc -l
```

cat → display content of file.

| → pipe symbol

sort, uniq → commands

sort → sort the content

uniq → seperate uniq words

wc → word count → Count words

- PERL
- PYTHON
- TCL/TC
- Visual Basic (VB)

} Scripting languages (include UNIX shell)

- Class-file → similar thing (reuse the components generated in a program)

⑤ Embedded system environment

Modern computers.

→ It posses imp requirement of embedded system environment

↓ this gives real time requirement.

→ Application

- Cross compiler (this application is acheived by cross compilation)

↓ it is a special compiler which runs in a Computer which is rich in resources.

This is an example of cross platform software development

⑥ Dynamic specification, flexibility & Adoptive software

in static v/s dynamic,

static is easy to handle

features of program are specified before execution

during the program things gets changing

↳ difficult to handle, becoz it requires extra management further poses execution time overhead.

flexibility → capability of to broaden the choice in the specification

Ex:- User defined data types are supported, then software is more flexible.

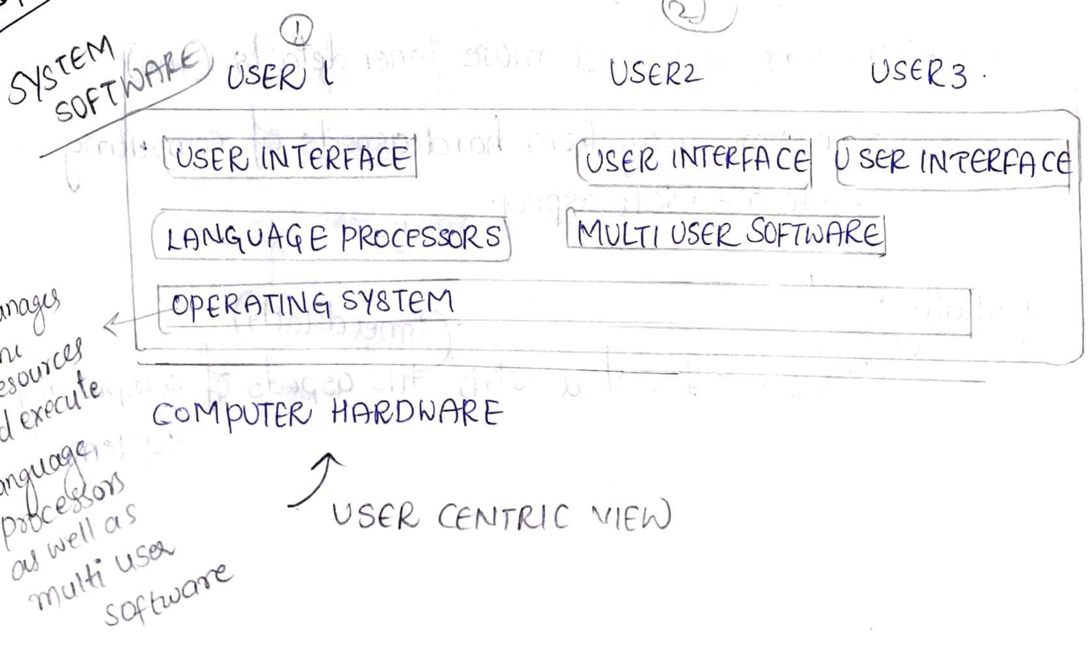
Adoptive software → that adjusts its own features and behavior according to its environment

Ex:- Plug and play capability of OS.

Views of system software

- User centric view
- System centric view

20/2/22

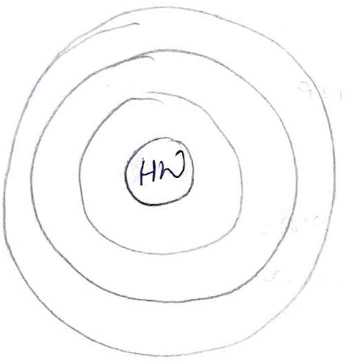


↗ User centric view   (program).
→ USER is developing something for some
Computational needs
- HLL, Assembly language.
For developing a program it needs converter

Compiler   Assembler
Loader, linkers
Debugger
Interpreter.

System centric view

- Efficient usage of computer system
  - Efficient utilization
- User convenience
- Non-interference



↳ what we Achieved by this layer centric view
↳ Abstracting more and more inner details (HW)
Astorical move.
→ Moving away from hard aspects of computing system → Soft aspect

Initially:)
Person → familiar with (inner details) HW aspects of computing system.

$\underbrace{0110}_{\text{operation code}}$ $\underbrace{011100}_{\text{operand 1}}$ $\underbrace{010110}_{\text{operand 2}}$ → instruction written in machine language

opcode — Data → Machine code

0110 → ADD instruction
0010 → SUB
0011 → MUL

⇓

Lesser Details

Next level

$\underbrace{\text{ADD}}_{\text{opcode}}$ $\underbrace{\text{A}}_{\text{operand 1}}$ B — Operand 2

This helps the user to remember.

Addressing modes.

ADD #A B

Mnemonic code ↳ one which aids to remember

SUB C D
MUL E F

Assembly level language.

M/c code
↓
Mnemonic code → Still not sufficient

⇓

Highest level

C = A+B;
X = Y+Z-P;

Mathematical Expression
- Statement;

High level language

Harder

M/c code ↗ object program

↑ Translation ASSEMBLER

AL — Mnemonic code

↑ Translator. COMPILER

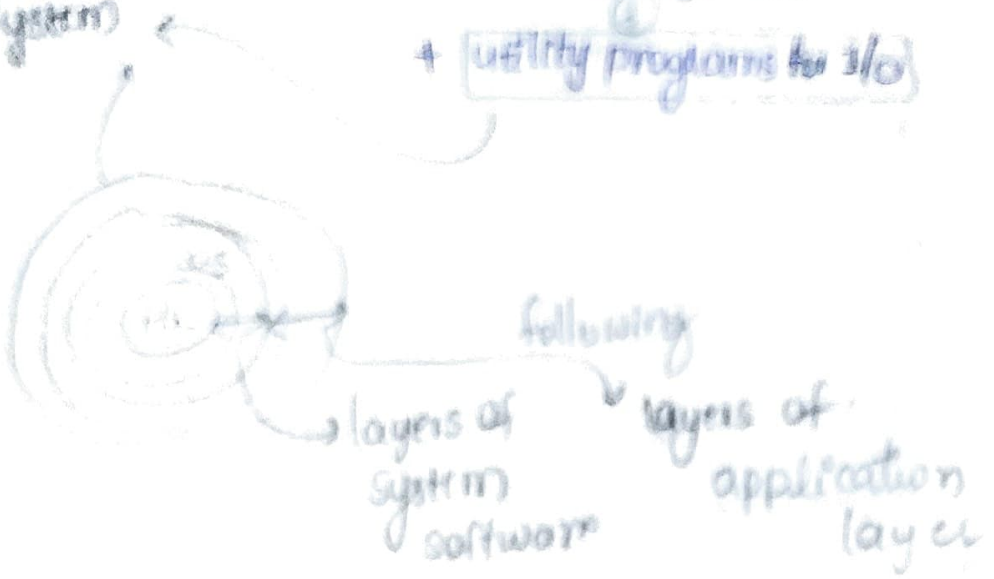
Statement ↳ HLL

↓

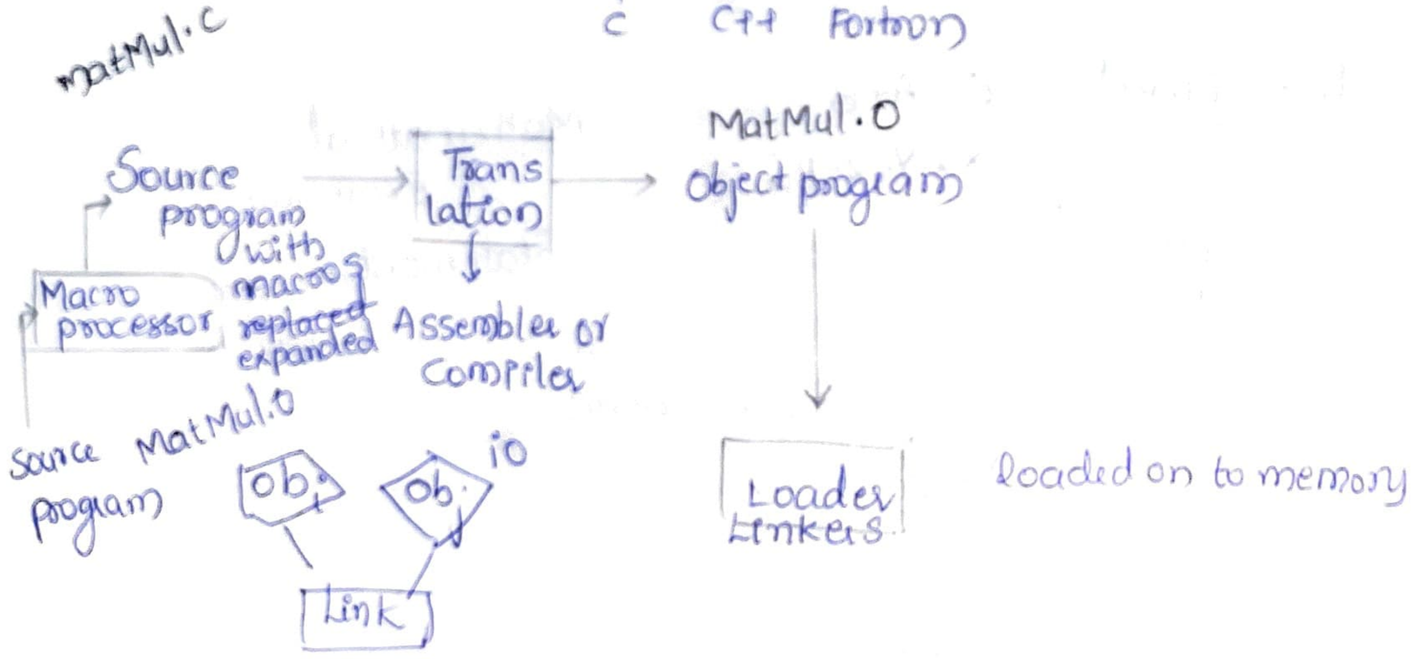
Softer

↳ Source program

Machine (hardware) + ① Translators
↓ + ② Library programs
Computing System + ③ utility programs for I/O



Software
- System software
  - OS
  - Translators
    - Assembler
    - Compiler
      - C
      - C++
      - Fortran
    - Interpreter
- Application software



Assembler
Loader
Linker
Macroprocessor
Text processor
└ Tool

Compilers
Interpreters
Debuggers
Operating system
Database Management system
Network connection