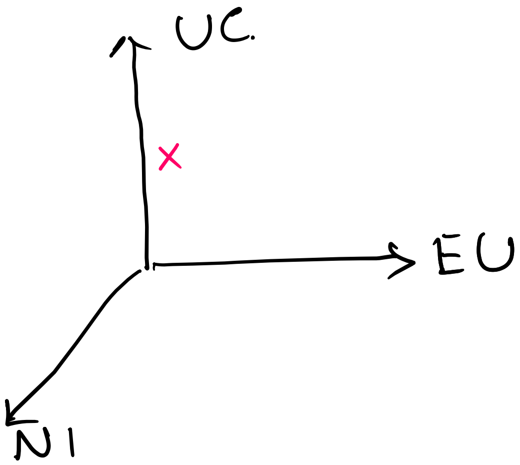
System Software.

- User convenience (UC)
- Efficient use. (EU)
- Non-interference (NI)



(A) (B)

→ Many factors

① Program development and production environment

Compiler

- Translate:

$$HLL \rightarrow \underline{ML}$$
$$\downarrow$$
Execution

- Analyses each HLL statement.
- Phase1: Program is compiled.
- Phase 2: ML instructions are generated.

→ Loop
  - Only once analyse

Interpreter.

- Does not generate ML Pgm.
- Analysis → P
  - Carries out desired computation.

↓

Keeps track of sequence 'P'

- Every time analyse.

- Program development
  ↓
  interpreter.

- Debugger
  - Stepwise
  → Interactive. debugging
    → Break points

② Making software portable.

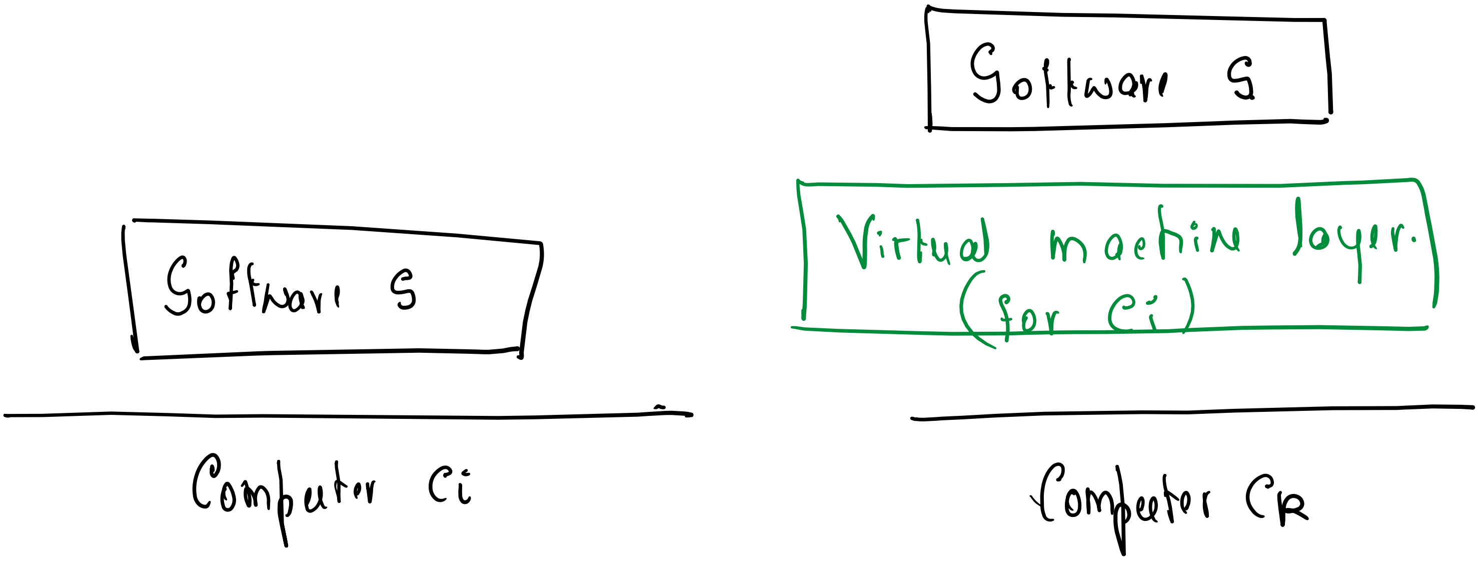
↓

Every computing environment.

- Program → special features provided by OS or computer.
  → HLL

Virtual Machine concept

- Convenient
- VM: Abstract computer. → desired set of features.



— Software layer as shown. } Portability using VM.
↓
causes overhead.

→ Pascal Programming : 1970s → systematic Programming

→ VM → Designed.

→ Pascal compiler.

→ Code → Pascal VM. } P- code.

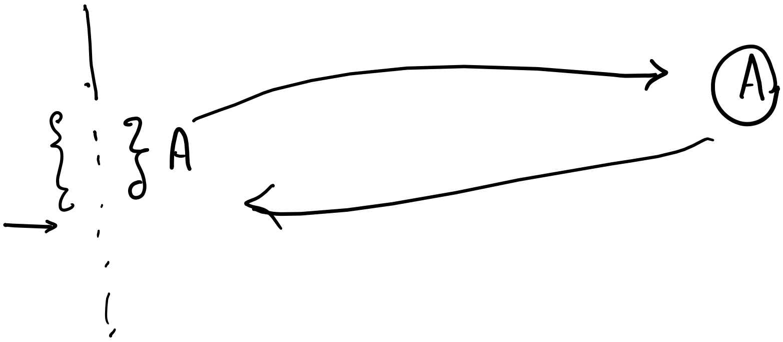
→ Java programming

→ JVM

→ Compiled → .Java byte code

3) Realizing benefits of the Internet

- Programs → remote computers → integrate the result

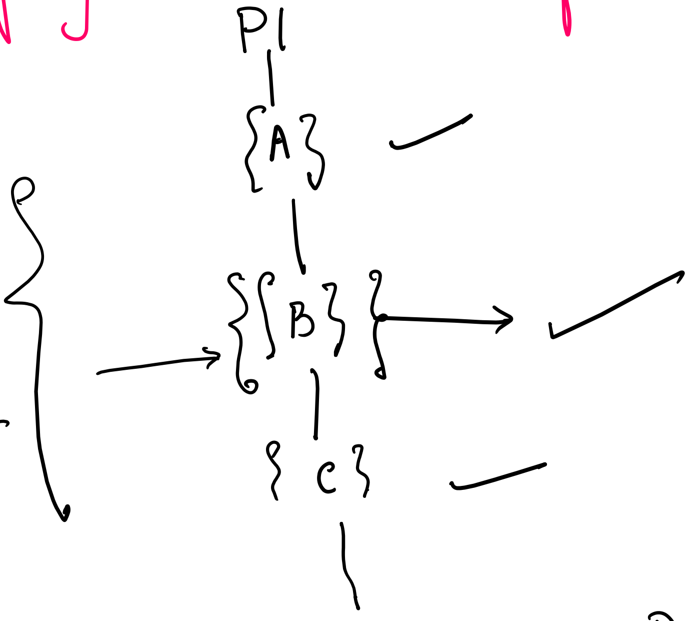


- Download → unknown program
  → Danger of interference.

- Web Server → Dynamic data.

4). Treating programs as components

→ Reuse.



Quickly.

↓

Support

Deep Learning

↓

Python.

Scripting languages.

↳ Unix shell script.

Task: Counting → unique names in files alpha.

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

- PERL
- PYTHON.
- TCL|TK.
- VB

- Class file.

5, Embedded system environment

Modern computers.

→ Imp requirement → embedded system.

↓

Real-time requirement.

→ Application.

→ Cross-compiler.

↓

Special compiler.

– Cross-platform software development.

6. Dynamic Specification, flexibility and Adaptive Software.

Views of System Software.

- User centric view
- System centric view.