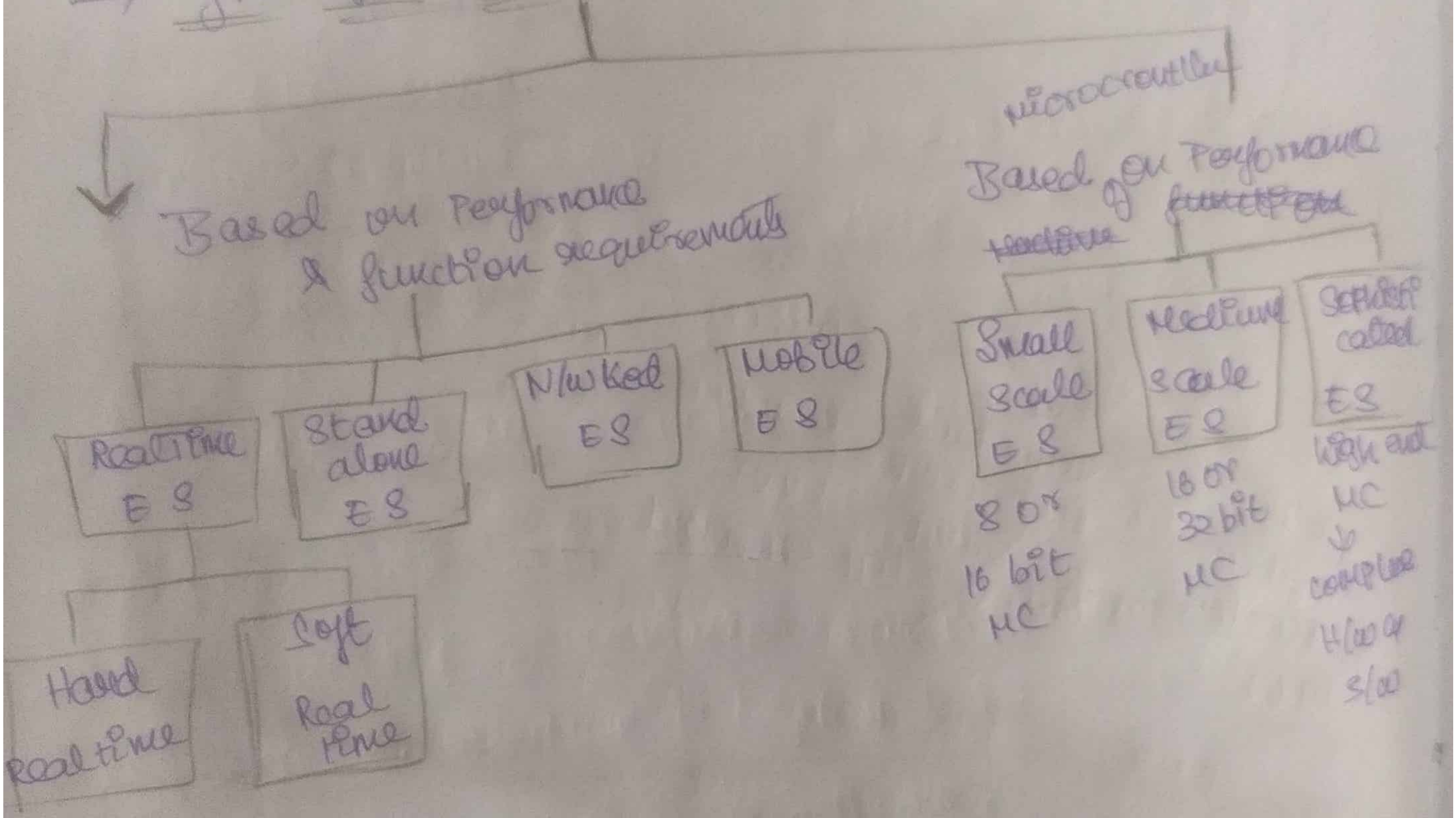
Unit - 2

→ **Embedded Systems** :- → special purpose. (Y) 2M

* It is a combination of Hardware & software to Perform a specific task throughout the life time.

→ **Types of Embedded Systems** :-



→ Types of computer :-

- Special purpose computer
  - → Developed to perform the specific.
  - → Eg:- calculator, washing machine ac etc...
- General purpose computer
  - → Developed by perform more than one task.
  - → Eg:- Laptop, Mobile phone etc.

23/6/22

→ Components of ES :-

ES Hardware components

- Power Supply
- Micro-controller
  - 8, 16, 64 bit.
- Memory
  - RAM, ROM, EPROM, flash
- Timer/Counter
  - ON, off
  - on compressed
  - off-compressed
- Communication Port
  - → * transmitting receive
  - * complex → USB → ext
  - * simple one mode of com
    - Serial
      - 1bit
    - Par
      - 8 bit.
- I/P O/P Port

Head
↓
stored

Volt
↓
erased

Non-vola
to
→ shut dr
backu

→ **Application :-**

* Consumer Electronics.
* Industrial Machinary.
* Automobiles
* Agriculture
* Processing Industrial Devices.
* Airplanes.
* Digital watches.
* Digital camera.
* Vending Machine
* AC
* Mobile device etc..

UI -> NO user Interface

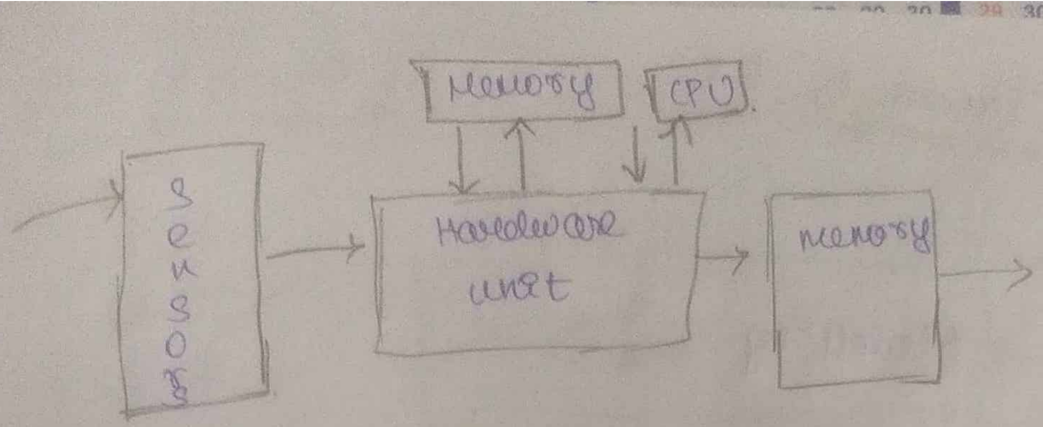
GUI -> user

→ **Automated System → no human interference.**

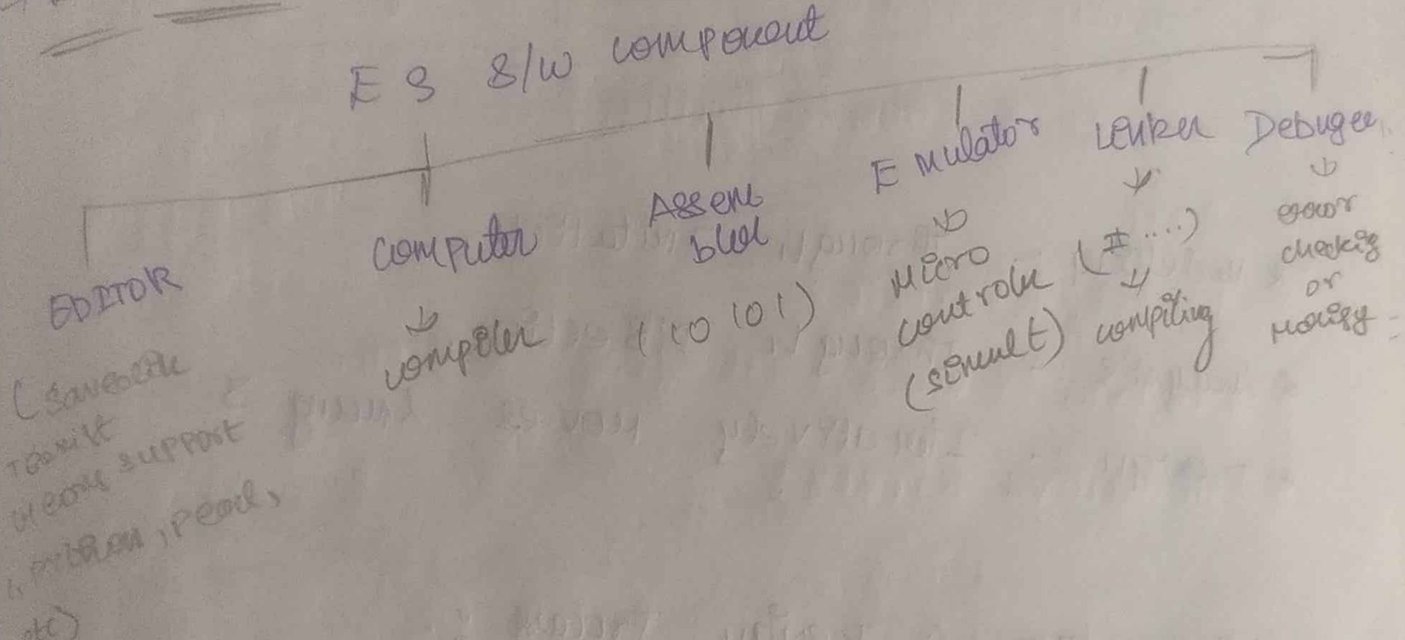
* Conveyor belt with sensor count no. of filled bottles, display on computer screen without human Interferenced.

→ **Electrical circuit :-**

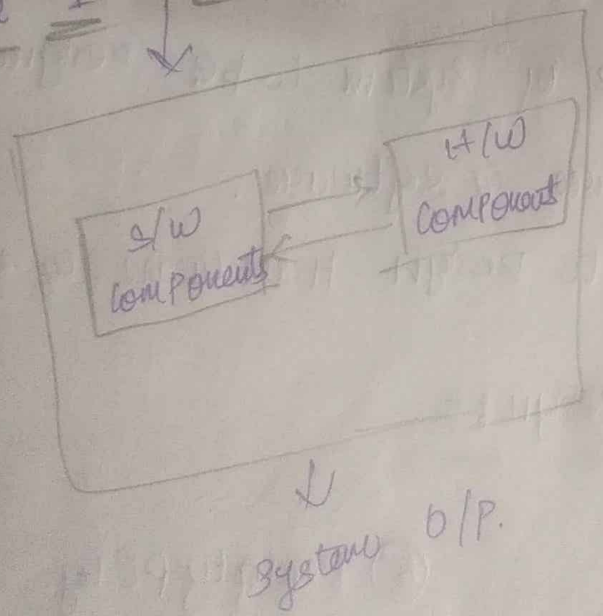
* Printed Circuit Board
* Resistors
* Capacitors
* Inductors
* Transistors,
* DIODES
* Integrated circuit
* LED (Light Emit Diode)
* LCD. (Liquid crystal display)



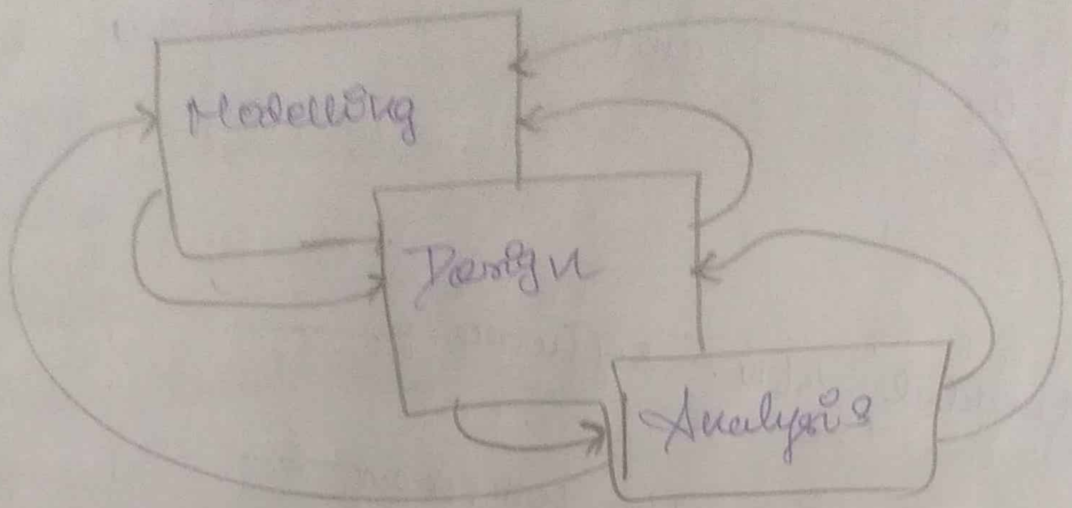
→ Embedded System Software :-



→ System Input :-



→ Design Process of ES :-



* Modelling :- Through Imitation.

* Analysis :- Through Dissection.

* Design :- Iteratively moves Among 3

→ Major Area of Design Process :-

* Sound Software & Hardware specifications.

* formulate architecture of system to be Designed.

* partitioning Hardware & Software.

* Iterative Approach to Design Hardware & Software.

→ Important Steps in Design :-

① Requirements

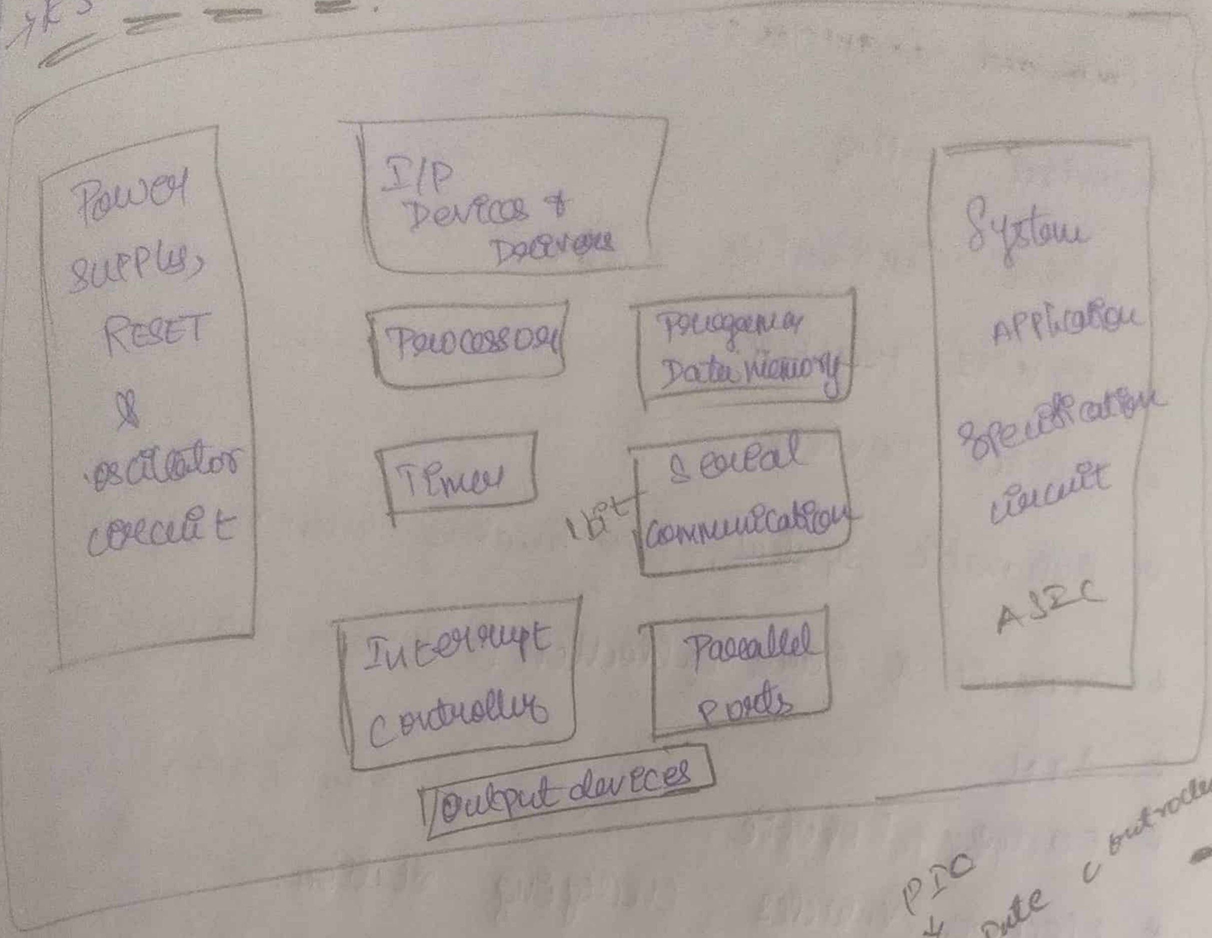
② System specification

③ Functional Design

④ Architecture Design

⑤ Prototyping

→ ES Hardware :-



PIC
↓
PSI Data controller
peripheral

→ other Hardwares :-

* ADC

* DAC

* LED

* LCD

* keypad

→ Software :-

C, C++, DOTNET, Simulator - MASM.

compiler - KEIL

ASIC
↓
APP Spl Inte circuit

→ Requirements :-

* Reliability

* cost - effectiveness

* Low power consumption

* Efficient → use of processing Power.

* Appropriate Execution time.

→ Embedded Examples :-

- * central heating
- * Global Positioning system
- * Fitness Tracker.
- * Medical Devices
- * Automatic system. (lock breaking system)
- * Transmit & Fare Collection (Everything, Automatic Machine)
- * ATM (withdraw cash)
- * Factory Robotic (Mech jobs, sensors etc)
- * Electric vehicles charging station. (assembly line, monitoring welding)
- * Interactive Kiosts.

= Give alert from Monitor charging, Charging etc

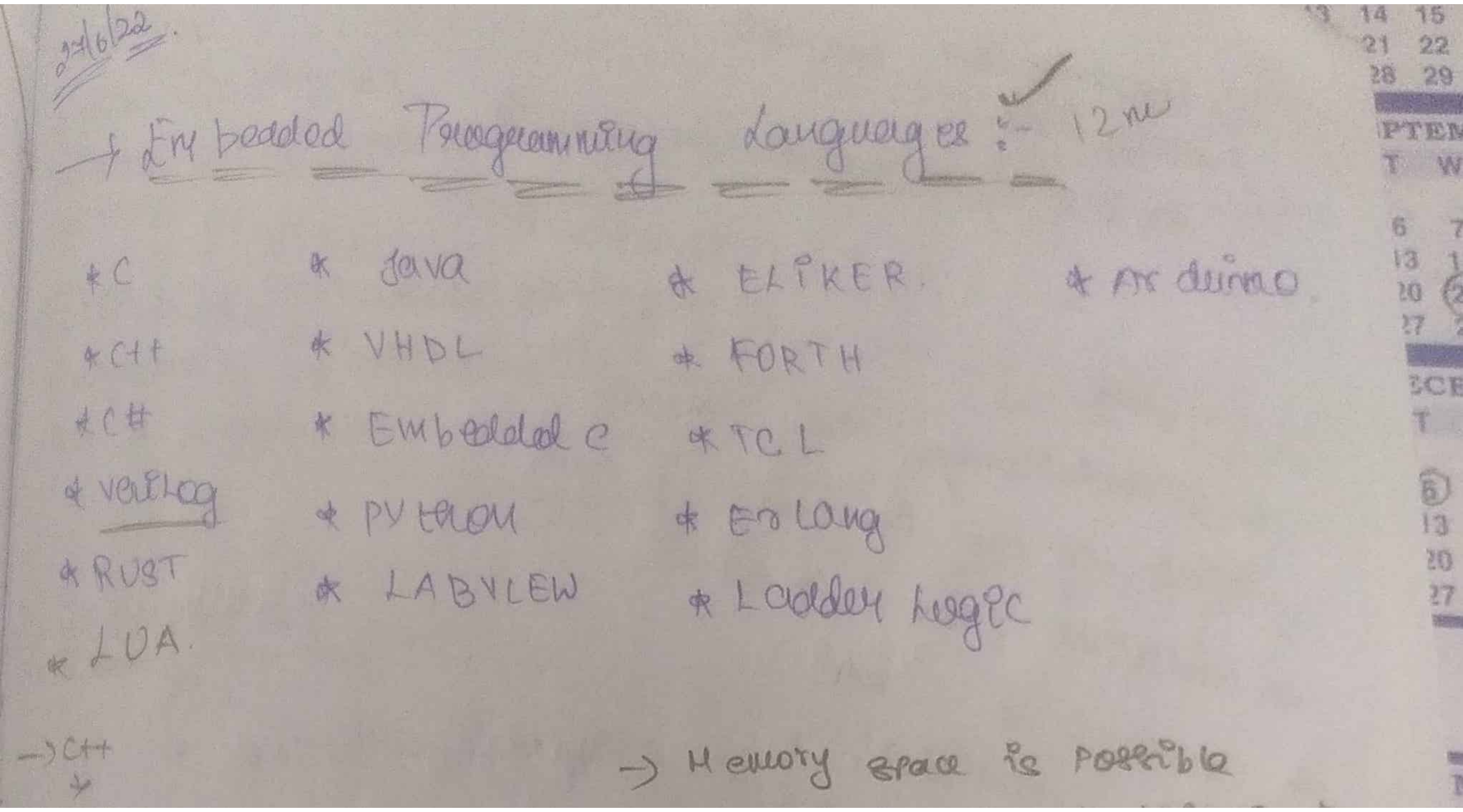
Teja

Teja Sri

* Inter: Intuative experience

* users

free -> complex washing Machine



27/6/22.

→ Embedded Programming Languages :- 12m

*C

*C++

*C#

*verilog

*RUST

*LUA.

*Java

*VHDL

*Embedded C

*Python

*LABVLEW

*ELIKER.

*FORTH

*TCL

*Erlang

*Ladder Logic

*Arduino

→ C++

↓

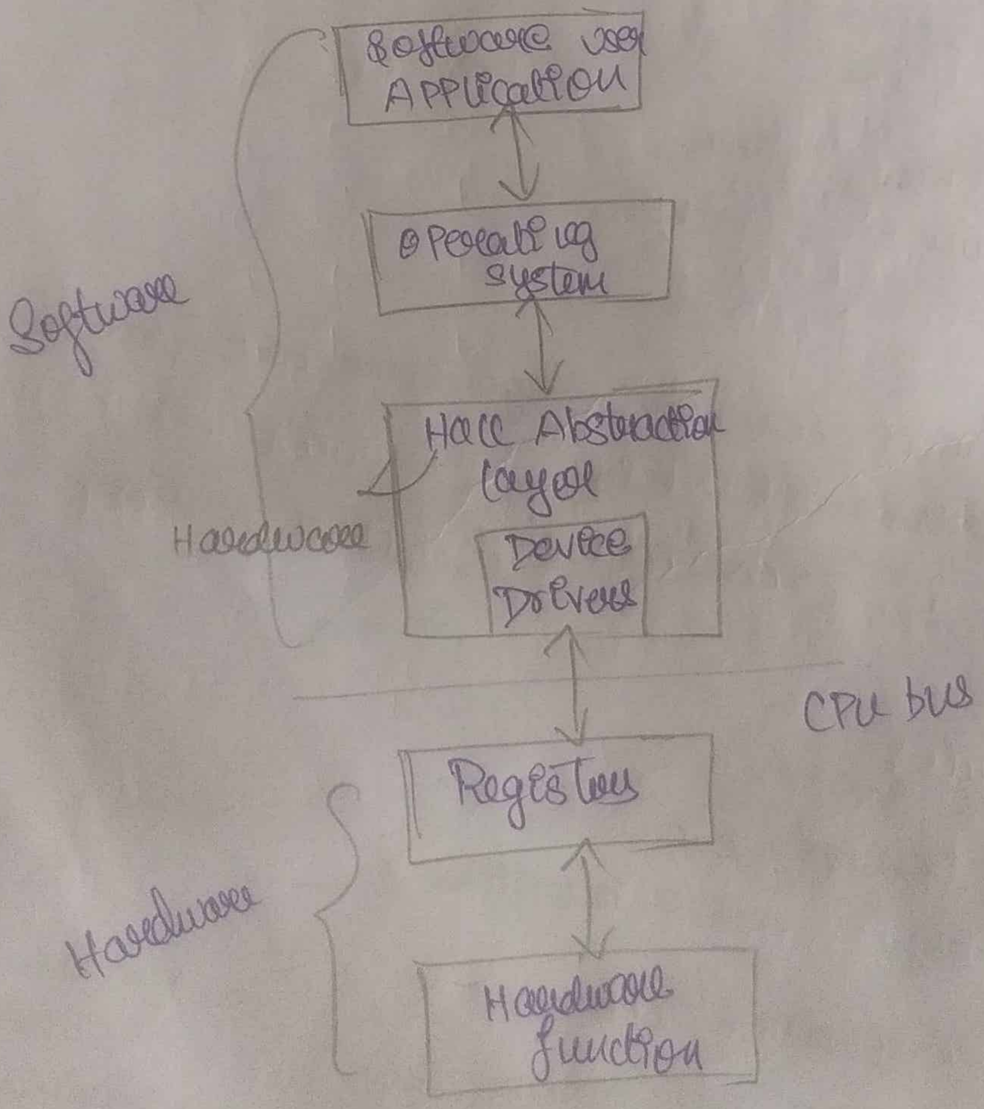
→ Memory space is possible

30/6/22

Abstraction :-

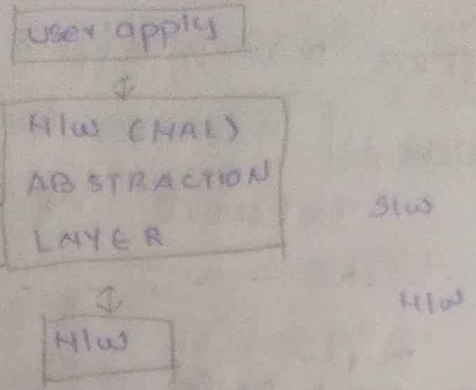
(Hardware devices,

tu



* Allow Users to Interact with Hardware Devices.

* Allows Device Independent Programming by standard call to Hardware.



* No need to underst and physical Interface

→ Interperetation :-

* realize :- Semantics of a language

* Instructions visible to Programmer are Implemented by Low level Processor, which Interpret Program visible Instructions.

→ Translation :-

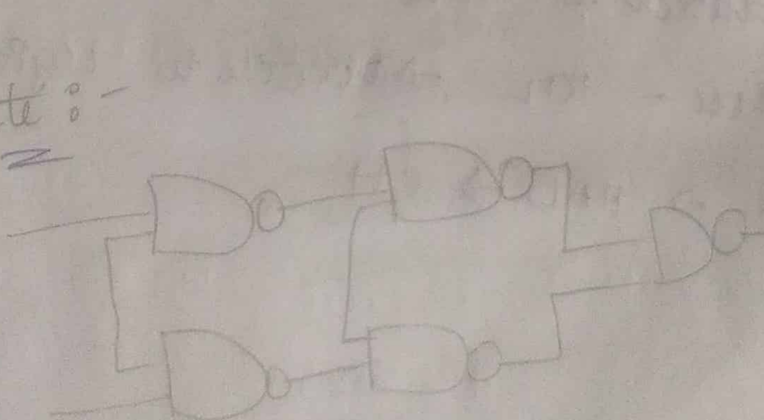
* Easy form of abstraction.

* RTL (Register transfer Level)

Behavioural Synthesis :-

→ Logical synthesis.

XOR Gate :-



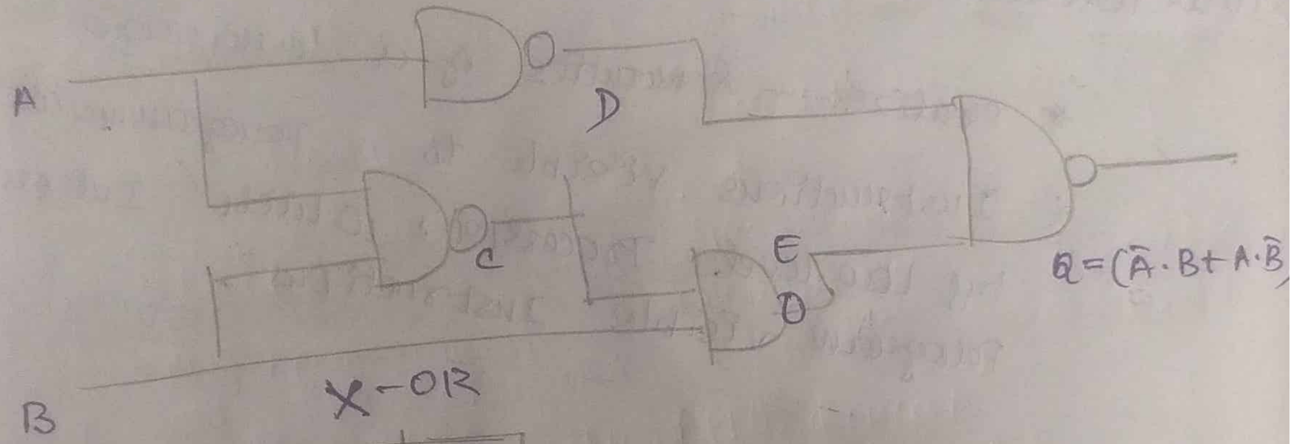
→ Extension :-

* Define New Datatypes.

* System → Hardware level.

Implement by extension of C++.

→ XOR using NAND Gate:-



X-OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

very case?

→ Interpretation - VLIW

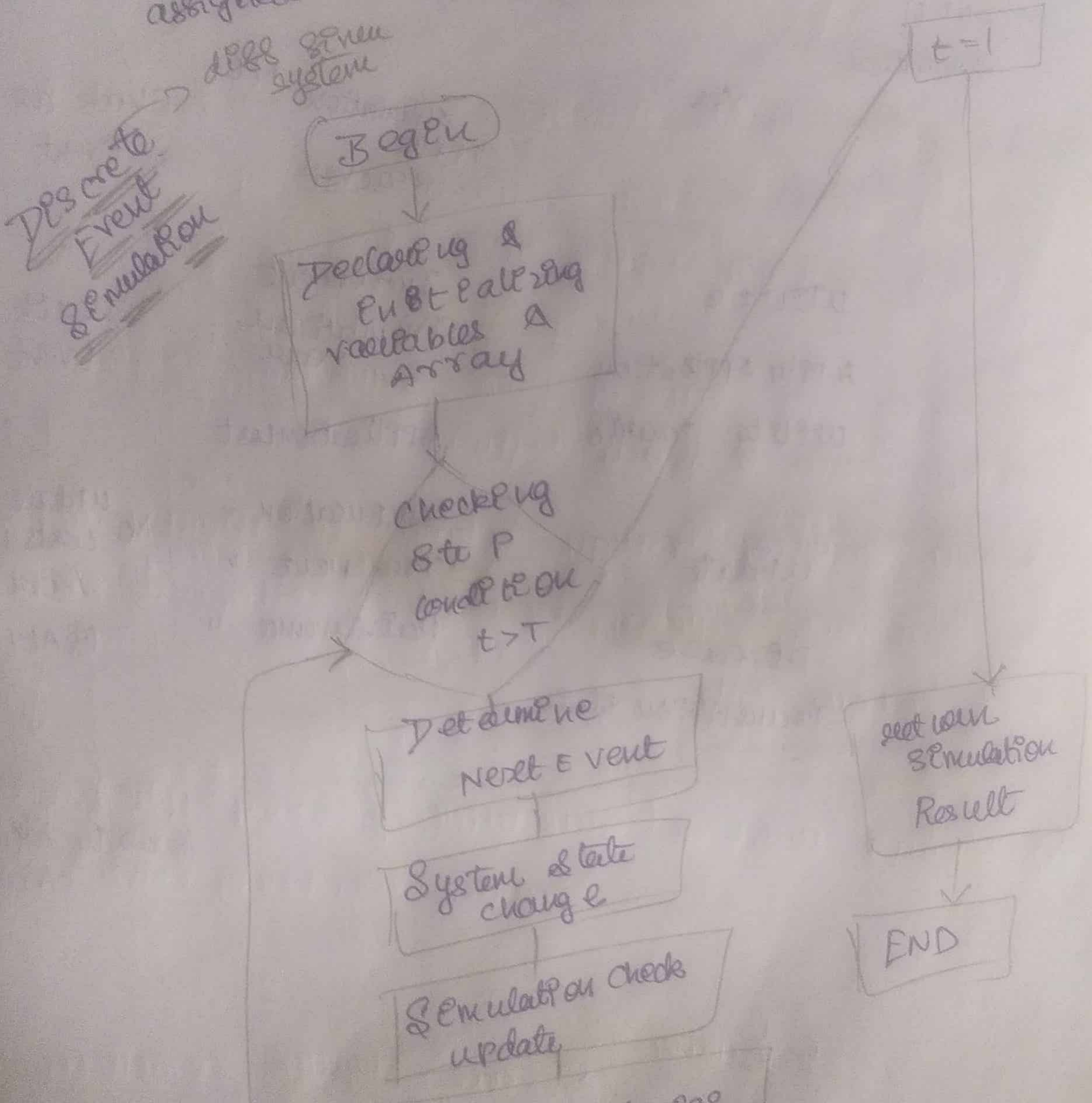
→ Translation - RTL → Direct to logic gates.

→ Extension → HAL → C++

→ Discrete Event Simulation :-

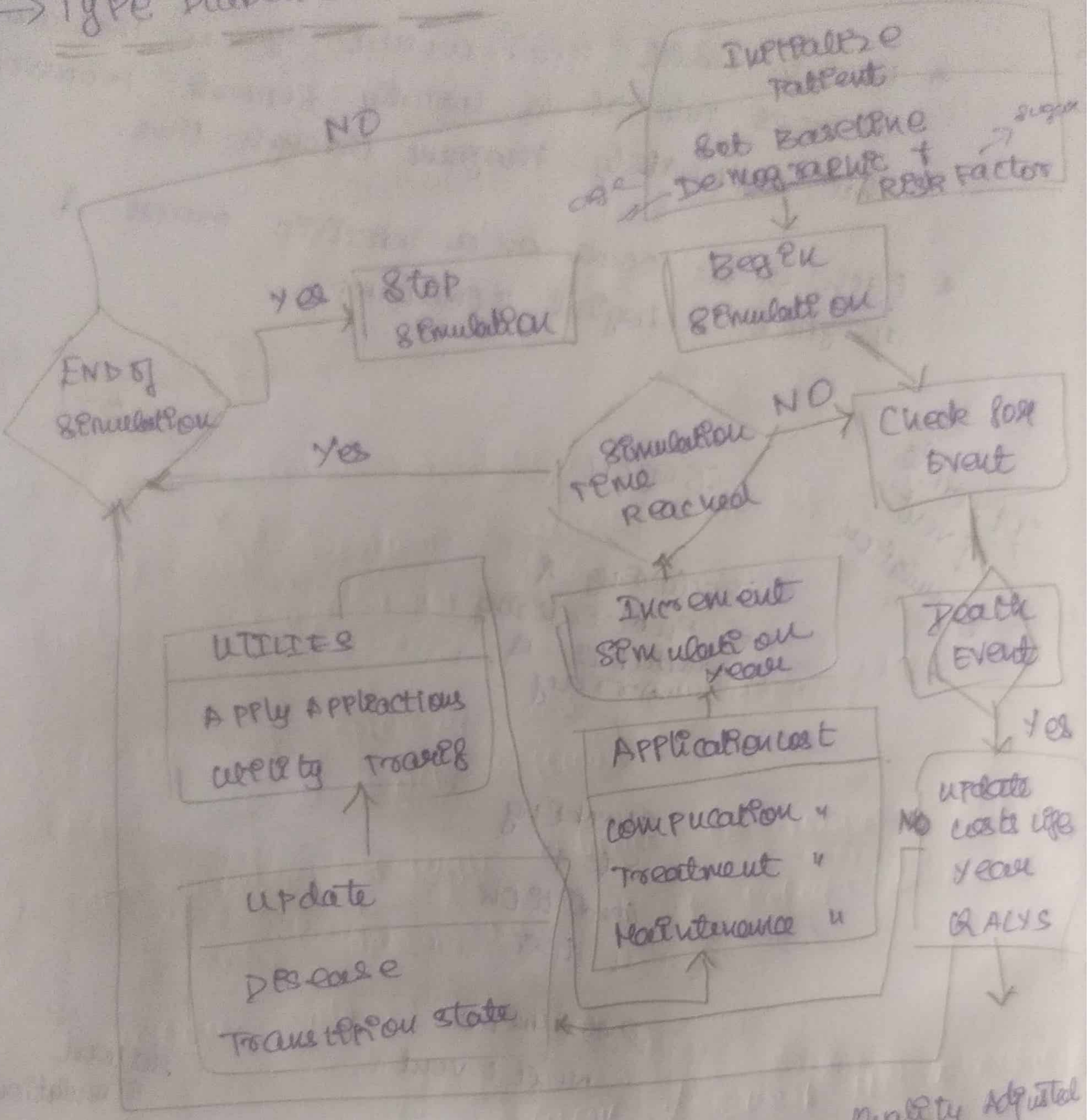
* Used to model real world system that can be dispersed into set of logically separate processors that autonomusly program through time.

* Each event occurs on a specific process & assigned a logical timestamp.



Discrete Event Simulation
diff given system
Begin
Declaring & initializing variables & Array
Checking stop condition t>T
t=1
Determine Next Event
System state change
Simulation check update
collect statistics
get the simulation Result
END

→Type Diabetic Patient :-



Quality Adjusted Year.