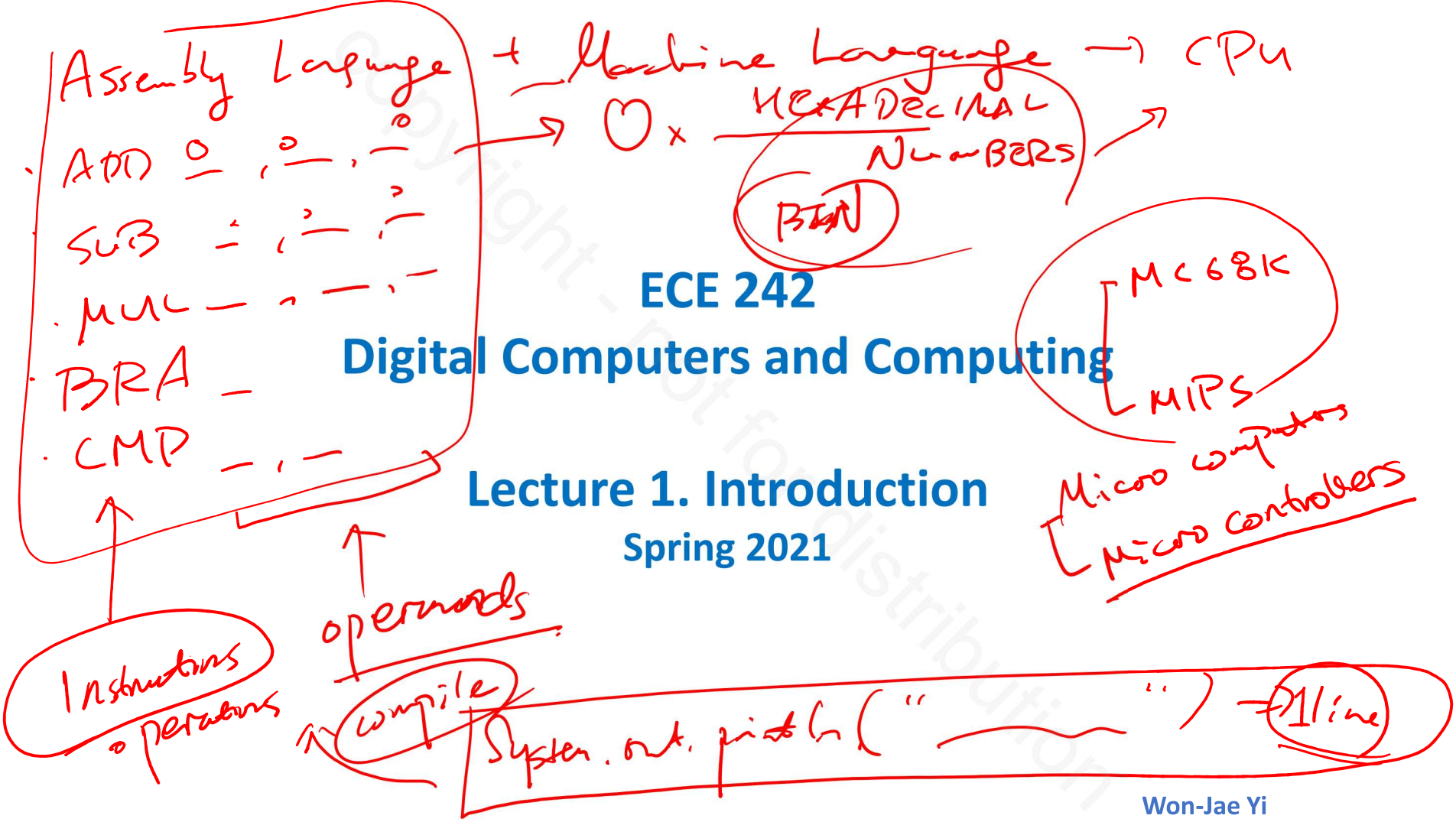
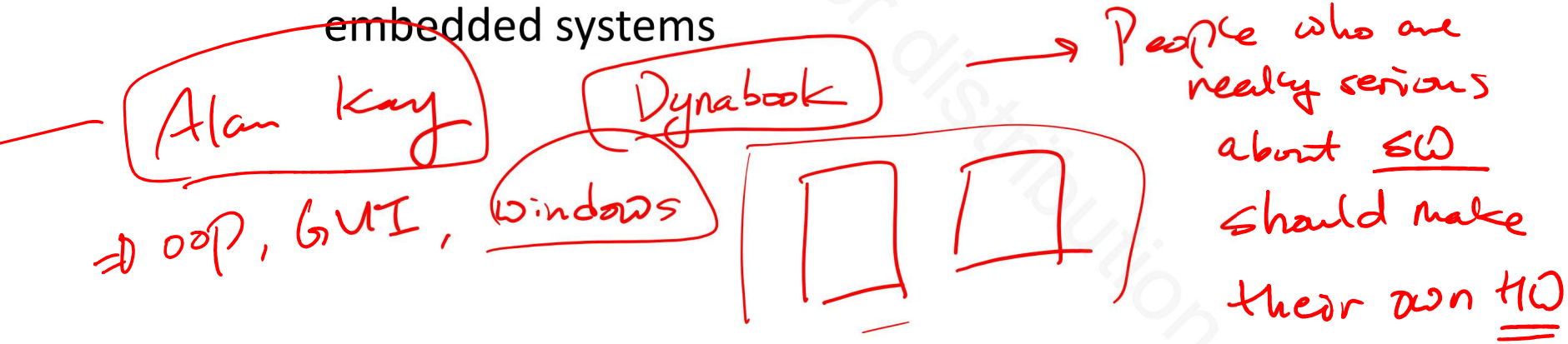


Won-Jae Yi

Embedded Systems with Microcontrollers/Microcomputers

- Some combination of computer hardware and software
- Programmable and/or reconfigurable OTA
- Designed for specific functions and/or features
- We are living in the world flooded with embedded systems



Examples of Embedded Systems

- Home appliances
  - A/C, refrigerator, microwave oven
- Mobile devices
  - smartphone, tablet, smart watch
- Automotive
  - HUD, ADAS, cruise control
- Military
  - Missiles, satellites
- Gaming Consoles
  - XBOX, PlayStation
- Almost all electronics are equipped with embedded systems

Microcontroller/Microcomputer

- A small computer on a single chip
  - Processor, control unit, input/output, memory, clock
- Typically "embedded" inside some device that they control

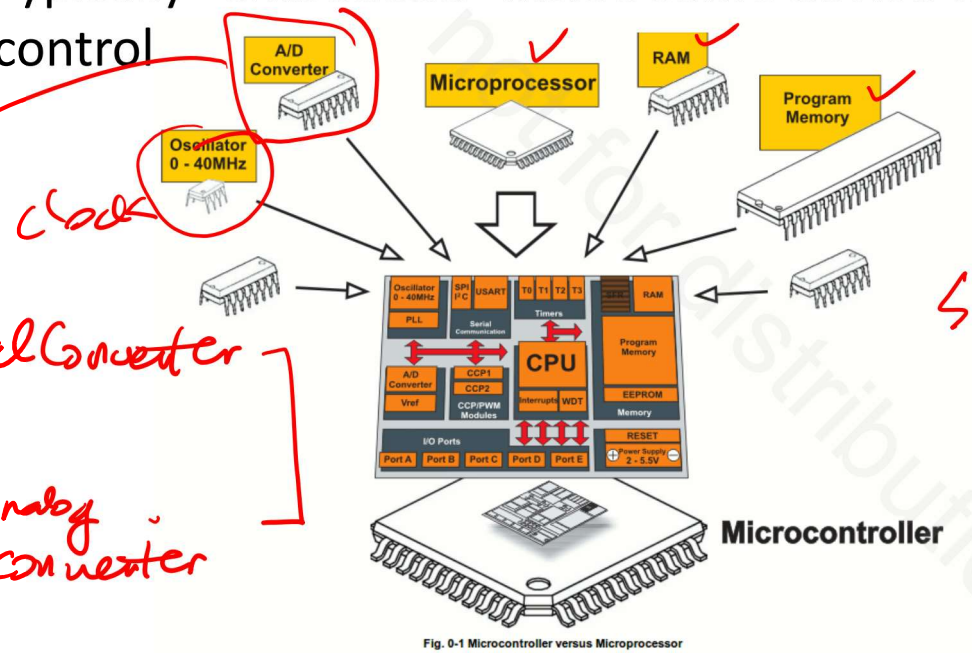
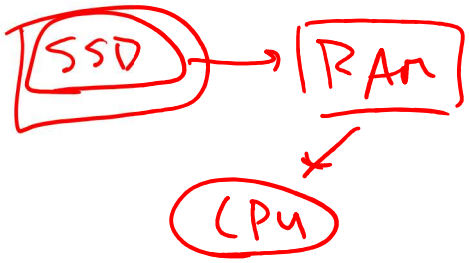
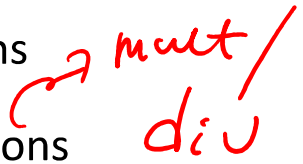


Fig. 0-1 Microcontroller versus Microprocessor

Microcontroller Components



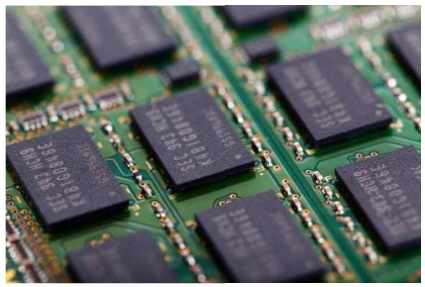
- Processor (CPU)
  - Execute programs with serial execution of instructions
  - Addition, subtraction, bit-wise AND/OR, shift operations
  - Registers (fast storage for load/store data from memory)



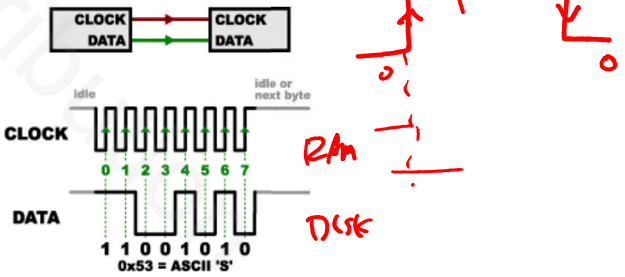
- Control Unit (Send/receive signals to control components)
  - Send/receive control signals to other components
  - Flow of the data between other units and operations

Microcontroller Components

- Memory
  - Program code (instruction) and data are stored
  - Addressable locations to load/store data
  - RAM, ROM, SD card, HDD, SSD



- Clock
  - Periodic signal to all components in microcomputer
  - Synchronization of data flow

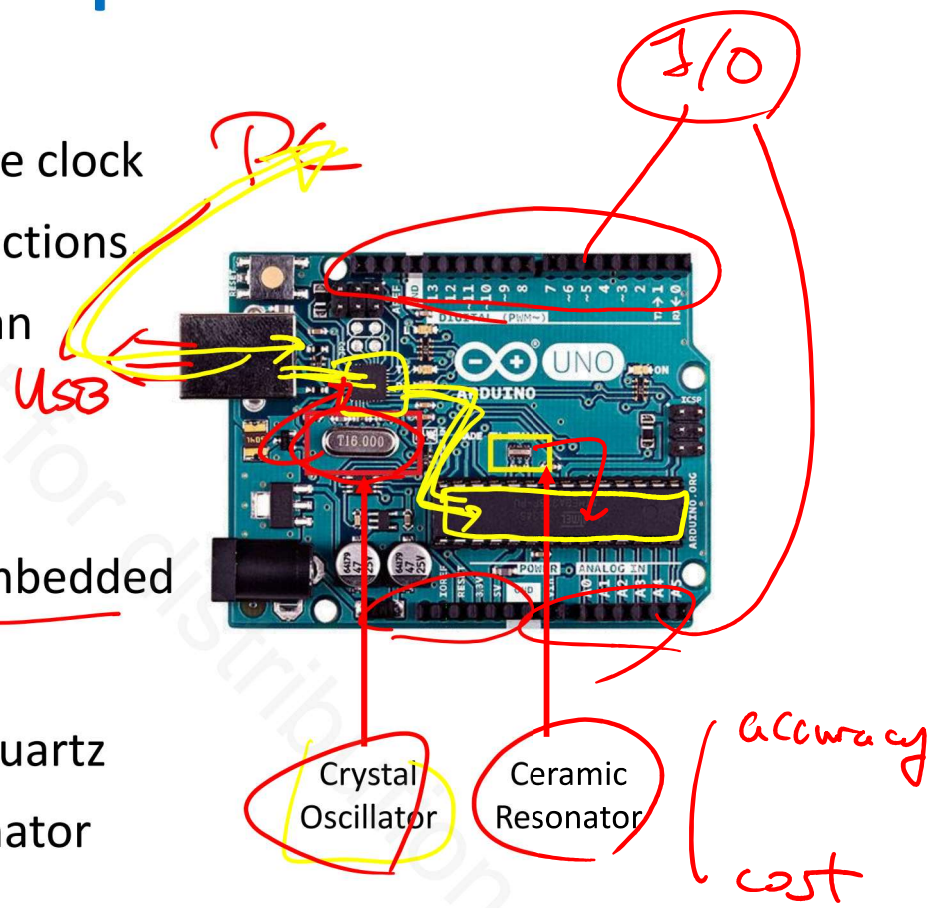


Microcontroller Components

- Input/Output (I/O)
  - Interface between internals of a microcontroller and outside world in the form of the pins of a microcontroller via special variables called “registers”
  - Registers are actual hardware memory locations inside the microcontroller
  - When assign a value to these registers, actual value in the hardware changes, and can be changed multiple times
  - Keyboard, LED/LCD Display, printers, USB drives, sensors, actuators, etc.

Microcontroller Components

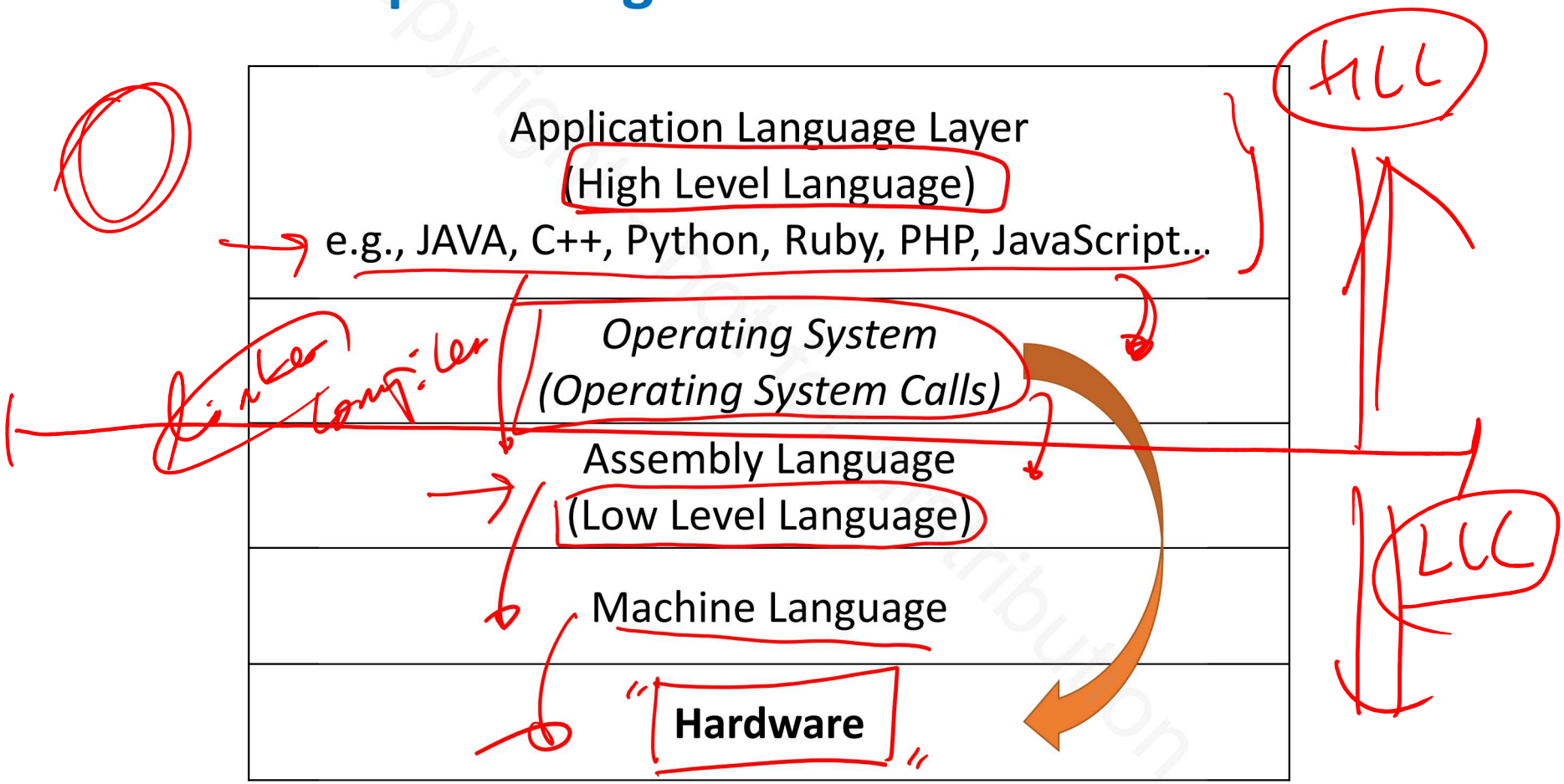
- Clock Signal Generator
  - Microcontrollers require clock cycles to execute instructions which is generated by an oscillator circuit
  - Used for synchronous functions within the embedded systems
  - e.g., crystal oscillator(quartz crystals), ceramic resonator



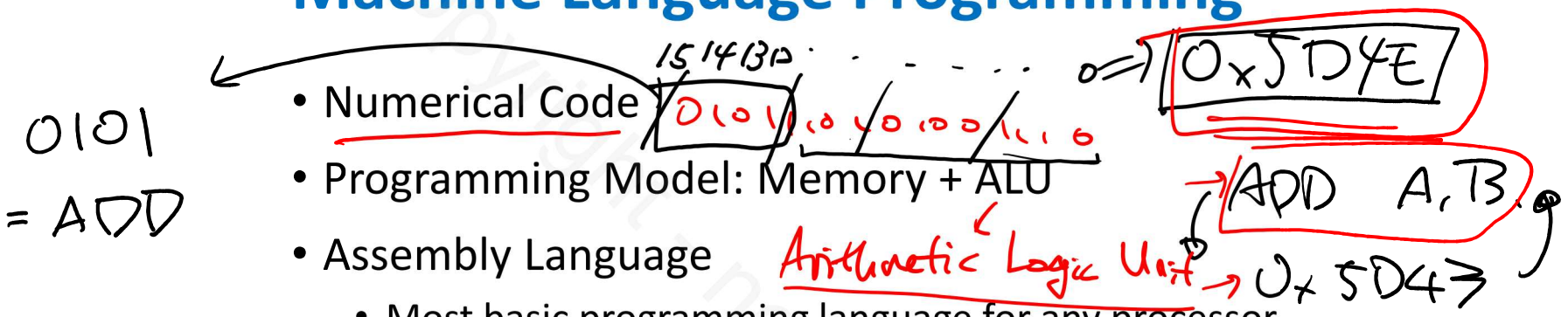
System Development

- PCs/Servers
- Use microcontrollers/microcomputers
  - More keen to software development compared others
  - Adding hardware is somewhat limited
  - Programming language depends on the platform
- Use FPGA (Field Programmable Gate Array)
  - Capable of becoming *any* digital circuit (within allowed space)
  - Hardware reconfiguration possible
  - VHDL/Verilog, C/C++/System C by HLS (High level synthesis)
  - MATLAB, LabVIEW
- Use ASIC (Application Specific Integrated Circuit)
  - Customized hardware configuration (VLSI techniques)
  - Full-custom design starting from RTL design

Computer Organization

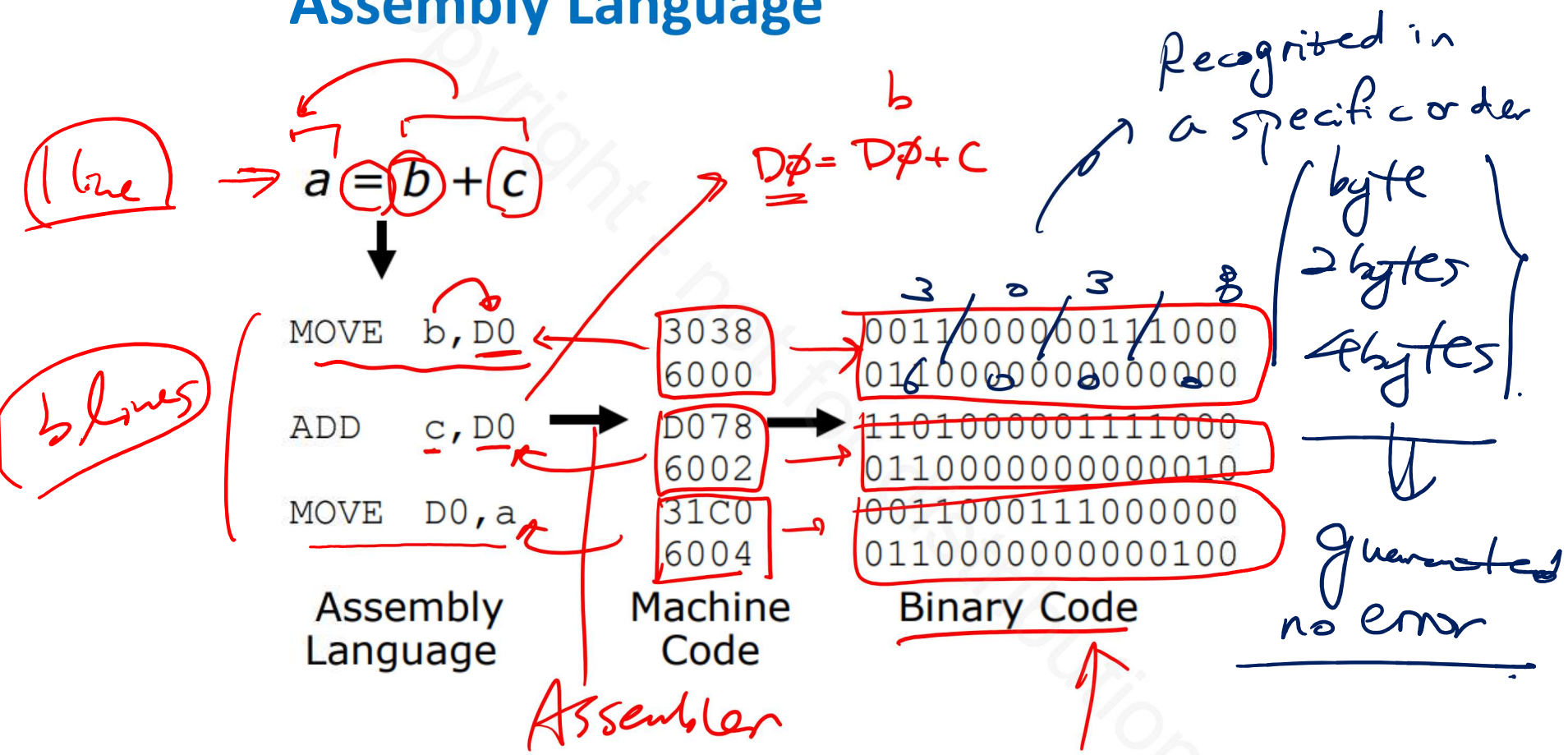


Machine Language Programming

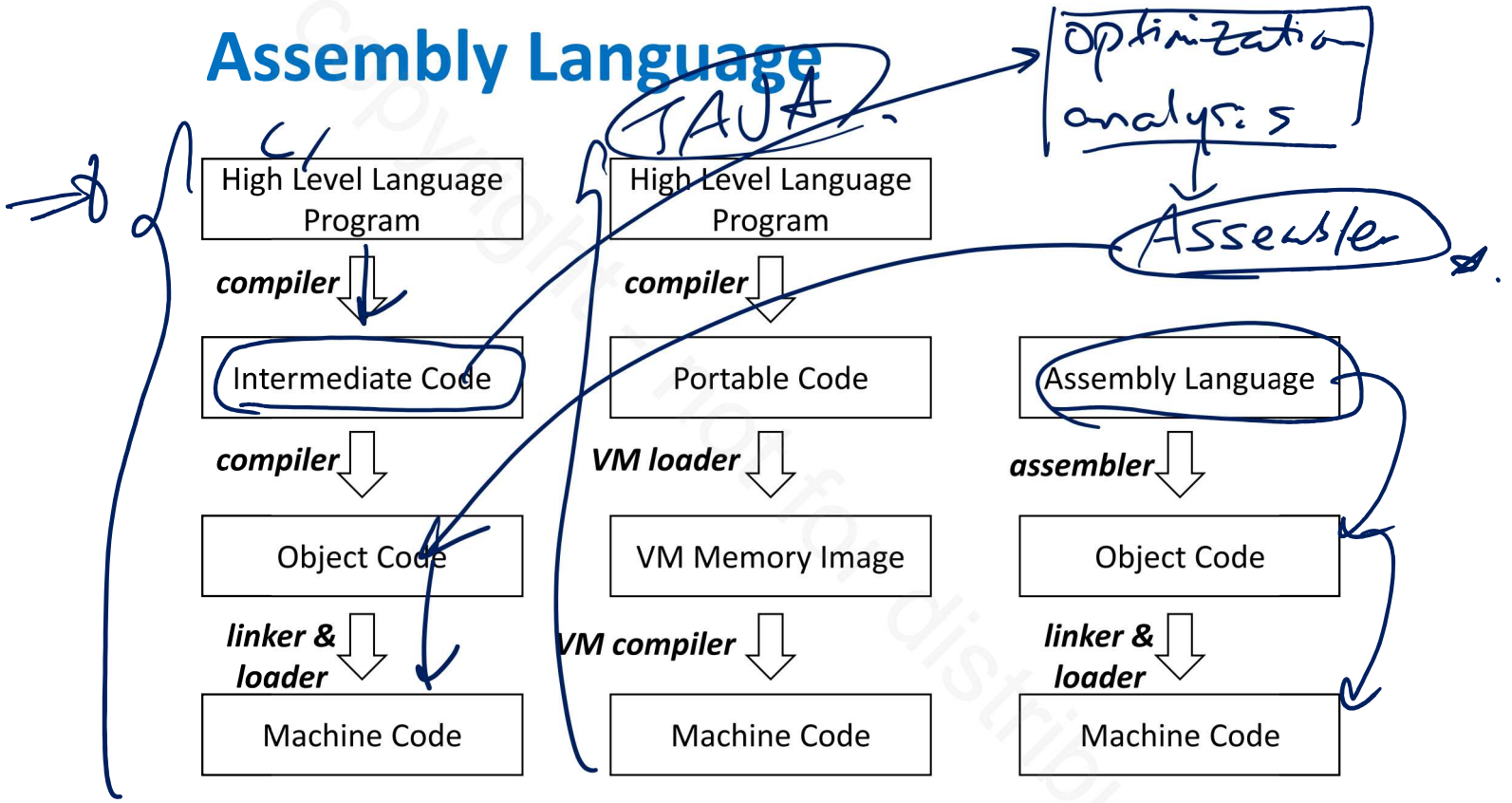


- Numerical Code
- Programming Model: Memory + ALU
- Assembly Language
  - Most basic programming language for any processor
  - Assembly language is *hardware specific*
    - Instructions that actually takes place inside the ALU
  - A programming language that enables human to directly implement operations on the physical CPU
  - A symbolic representation of the machine code needed to program a give CPU architecture
- Assembler
  - Converts assembly language program to machine code

Assembly Language



Assembly Language



- 1 line of High Level Language code can be multiple lines of machine code
- 1 line of Assembly Language code is 1 line of machine code

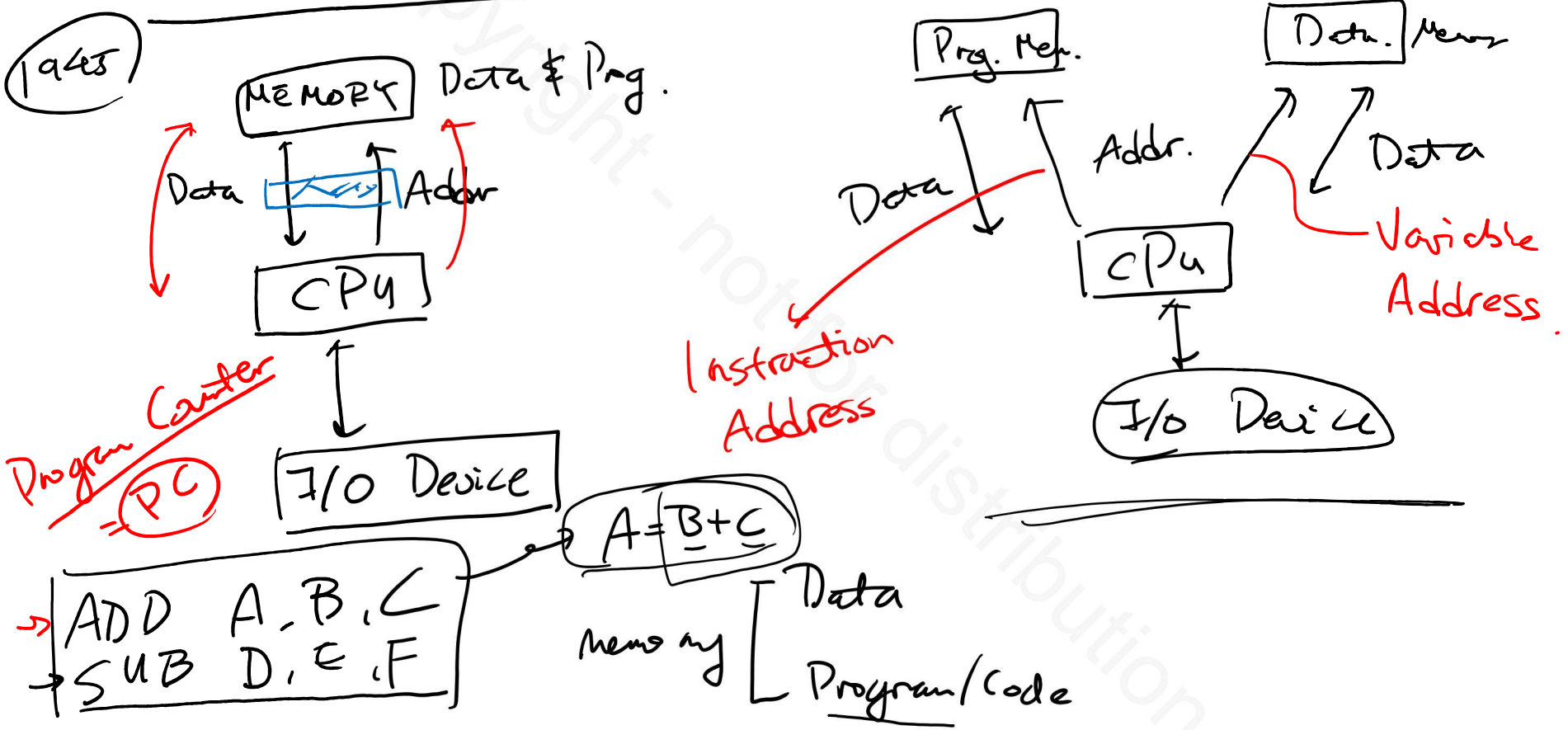
Why do we need to know Assembly Language??

- Hardware Perspective
  - Understand how a computer works at the machine level
  - Understand how hardware actually works
  - Understand Von Neumann Machine structure and find its limitations
    - **Von Neumann Machine Structure**
      - Stored program computer (program & data stored in memory)

```
pc = 0;
do {
  instruction = memory[pc++];
  decode(instruction);
  fetch(operands);
  execute();
  store(results);
} while (instruction != halt);
```

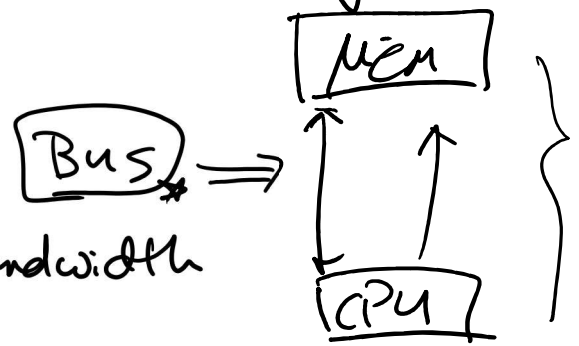
- Software Perspective
  - Foundation of many abstract concepts in software come from assembly language and computer architecture

Von Neumann Arch. Vs. Harvard Arch.

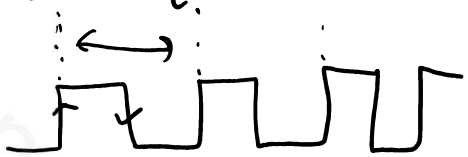


Von Neumann Arch.

features (good) : changes SW only!
→ less HW...
→ since command, program, data all reside on same memory



bad? : · bottleneck on memory bandwidth
· inefficiency using CPU.
→ can only run 1 command per cycle.



MC68000 & MIPS

+1 (ARM)

- Throughout this course, you will learn two different types of assembly languages for two different architectures
  - MC68000 for CISC Architecture
    - Complex Instruction Set Computer
  - MIPS for RISC Architecture
    - Reduced Instruction Set Computer

MC68K

⇒ Many different types of instructions available

⇒ only essential types of instructions available.

ARM

- We'll get started with MC68000 first, then MIPS architecture afterwards…
  - Theoretical backgrounds and programming assignments

"Exercise"

"Programming Assignment"

MC68000 Family

- Family of microprocessors/microcontrollers has brown to contain dozens of members, from newer versions of 16-bit MC68000 to the powerful 32-bit MC68060.

MC68000 (16-bit CPU)	Instruments, programmable controllers
MC68020, . . . , MC68060 (32-bit CPUs)	Personal computers, workstations and other sophisticated products
MC68302 (MC68000 CPU)	Communications processor
MC68332 (Microcontroller)	Automobile engine control, robotics, medical instrumentation
FlexCore (Combined CPU and logic)	Laser printers, memory controllers

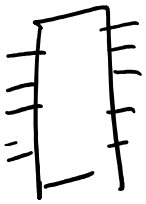
MC68000 Family

- MC68000 is the foundation of the MC68000 family
  - Basis of other family of MC68000
- The family concept assures
  - Processor line is adequately supported
  - Improved with time
  - One who is familiar with basic processor, would able to learn newer processors without hassle

One familiar with a programing language should able to adapt to others quickly.

Support for Various Families

Type	Support
Processors and support circuits	
8-, 16-, and 32-bit microprocessors and microcontrollers	Basic processor and enhanced versions available in various packages and speeds of operation
Embedded controllers	Lower cost processors for product design
Microcontrollers	Single chip with CPU, memory and other peripheral functions for product design
Peripheral interface circuits	Circuits for interfacing CPU to a wide range of peripheral devices
Special devices	Devices for floating-point mathematics, network control, and other applications
Software	
Operating systems	Various operating systems for program development, real-time applications, time sharing, or special purposes
Development software	Editors, assemblers, compilers, debugging programs
Applications software	Special-purpose programs for accounting, engineering analysis, etc.
Documentation	Manuals, application notes, data sheets
Product development	
Development systems	Complete systems for software development and hardware/software integration
Single-board computer modules	Processing units, memory modules, and other complete hardware subsystems

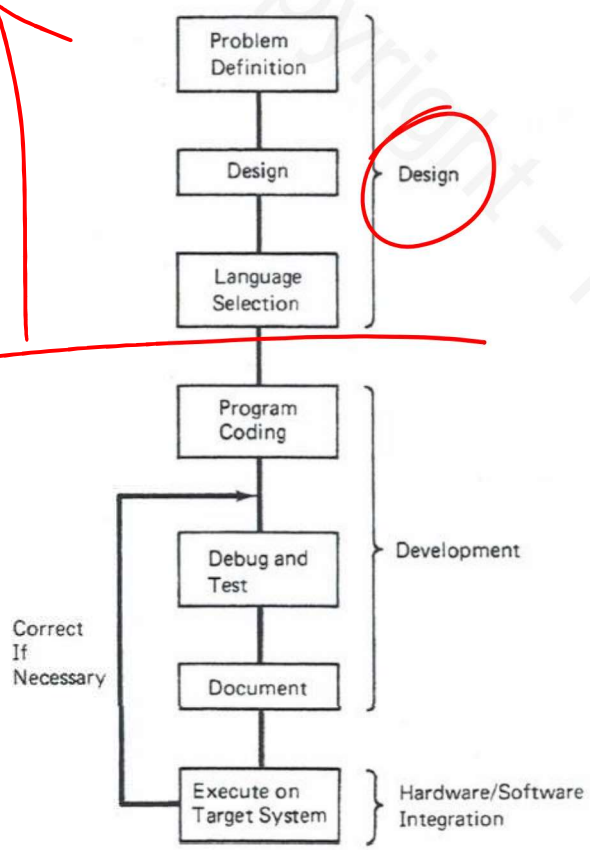


Support for MC68K Family

- Hardware Support
  - Understanding of the characteristics of hardware specs
  - CPU, interfacing chips, memory and peripheral devices
    - CPU, parallel I/O, serial I/O, RAM, ROM, ADC, etc.
  - Identify different circuitry/features available on the chip
- Software Support
  - **Application Software**: programs tailored to solve a specific problem
  - **Development Software**: Editor, assembler, or compiler to create applications programs
  - **Operating System**: Program to control CPU, I/O, storage

Datasheets / Spec. doc.

Software Support for MC68K



- Design before programming
  - Define the problem
  - Layout your design
  - Select the optimal language
- Development of software app.
  - Code, Debug, **Documentation**
  - Execute and Test

Development Software

- Operating Systems
  - Control program execution and file management
- Programs to create software
  - **Text editors**: create and edit source program
  - **Assemblers and debuggers**: translate and debug assembly language programs
  - **Compilers**: compile programs
  - **Linkage editors:** link separate program modules and create machine language code
- Cross-software
  - **Cross-assemblers, cross-compilers**: create 68K family machine language programs on another computer
  - **Simulators**: simulate execution of 68k family programs

Product Design (HW & SW)