

COMP1036 (CSF)

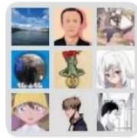
Computer Fundamentals

27/09/2024

Dr. Libin Hong and Dr. Wooi Ping Cheah

WeChat group

- Scan the barcode below to join the CSF 2024/25 WeChat group



群聊: Computer
Fundamentals 2024-2025



Valid until Oct 2.

该二维码7天内(10月2日前)有效, 重新进入将更新

Student Transfer and Others

- Students in the Faculty of Science and Engineering are allowed to transfer to other programs within the Faculty in the first two teaching weeks of Semester 1 in Year 2, subject to the approval from the Head of relevant school/department and the capacity at the University.
- Therefore, given some students won't start until the third week, first two weeks must be light.

Outline

- Who is teaching the module?
- Admin
- How will this module be taught and assessed?
- What is this module about?

Who is teaching the module

- Dr. Libin Hong
 - Research interests: Hyper-heuristic, Metaheuristic, Computational Intelligence, Artificial Intelligence
 - Previously worked at Insigma Hengtian Software Co., Ltd and State Street Corporation (China/US), teaching international students from over 20 countries at Hangzhou Normal University.
- Dr. Wooi Ping Cheah
 - Research interests: Soft Computing, Knowledge Engineering, Software Engineering, Artificial Intelligence, Programming Paradigms
 - Previously worked in the Multimedia University, Malaysia.

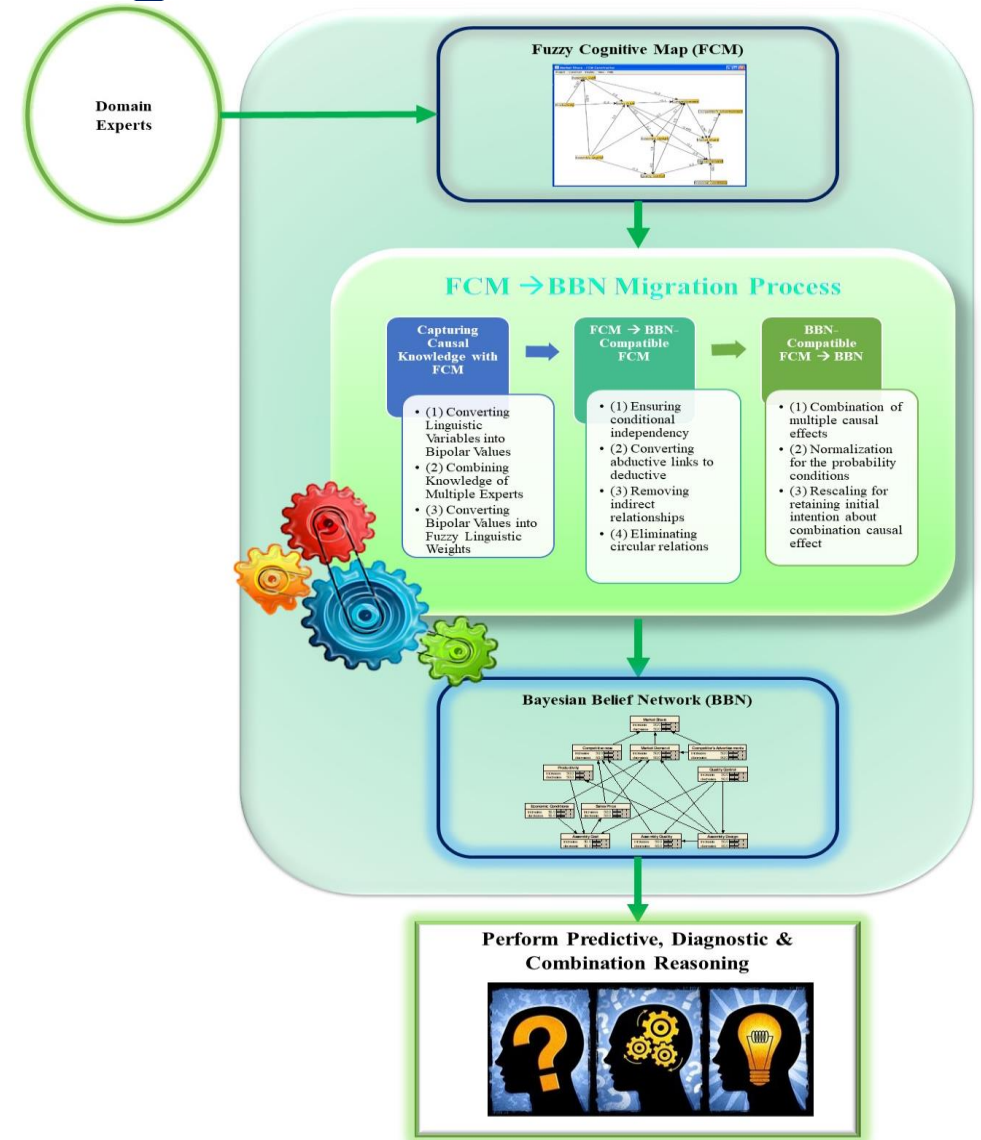
Research interests - Libin

- Hyper-heuristic
 - A method that selects or generates heuristics to solve computational search problems, addressing the challenge of 'heuristics to choose heuristics'
- Metaheuristic
 - A high-level algorithmic framework that provides guidelines for developing heuristic optimization algorithms
- Computational Intelligence
 - Set of nature-inspired computational methodologies and approaches to address complex real-world problems
- Artificial Intelligence
 - A field of study that focuses on creating machines that can learn, reason, and act in ways that are usually associated with human intelligence



Research interests – Wooi Ping

- Soft Computing (Fuzzy Cognitive Map, Bayesian Belief Network)
- Knowledge Engineering (Modeling, Representation, Reasoning)
- Software Engineering (Software Maintenance, Process Modeling)
- Artificial Intelligence (Machine Learning, Data Mining, Causal Discovery)
- Programming Paradigms (Logic, Functional, Object-Oriented)



Teaching - Libin

- Computer Fundamental (Qualifying Year)
- Introduction to Computing Systems
- Programming Fundamentals
- Object-Oriented Programming
- Algorithm and Data Structures
- JSP/Servlet Programming
- Application Development Based on J2EE Framework

Teaching - Wooi Ping

- Computer Fundamental (Qualifying Year)
- Programming and Algorithms (Qualifying Year)
- Programming Paradigms (Qualifying Year)
- Computer Security (Part II)
- Computer Graphics (Part II)

Research opportunities

- You may find the profiles of Professors
 - <https://research.nottingham.edu.cn/en/organisations/school-of-computer-science/persons/>
- Identify your own research interests and match them with professors
- Undergraduate research
- Summer research
- GRP project (Part I, a bit research, more on Software Engineering)
- Final-Year project (Part II)

Contact hours, Libin

- Contact
 - Office: PMB417
 - Email: Libin-Hong@nottingham.edu.cn
- Office hours
 - Thursday, 16:00-18:00
- Outside office hours
 - Drop an email to book appointment

Contact hours, Wooi Ping

- Contact
 - Office: PMB323
 - Email: Wooi-Ping.Cheah@nottingham.edu.cn
- Office hours
 - Wednesday, 12:00-14:00
- Outside office hours
 - Drop an email to book appointment.

Outline

- Who is teaching the module?
- Admin
- How will this module be taught and assessed?
- What is this module about?

Admin

- Plagiarism
 - Very strict on plagiarism.
 - We do check the code similarity.
 - The offender will be reported to School or University, and subject to disciplinary actions.
- 10 credits - workload 100 hours

Lectures and labs

- Lectures
 - Friday, 9:00-11:00, **TB-329**.
- Lab
 - Friday, 14:00-16:00, **IAMET-406**. (Group 1)
 - Friday, 16:00-18:00, **IAMET-406**. (Group 2)

Moodle

- **Formal source of communication**
 - Lecture materials, lecture/lab recordings,
 - Announcements on coursework and updates.
- Do check regularly for Moodle updates, especially on coursework requirements.

Lectures

- Start at 9:00 sharply
- Finish at 10:50 with a 10 minute break in the middle
- Attendance will be recorded
- I will attempt to answer questions after lectures
- Feedback will be arranged
 - Feel free to let us know how we are doing

Labs

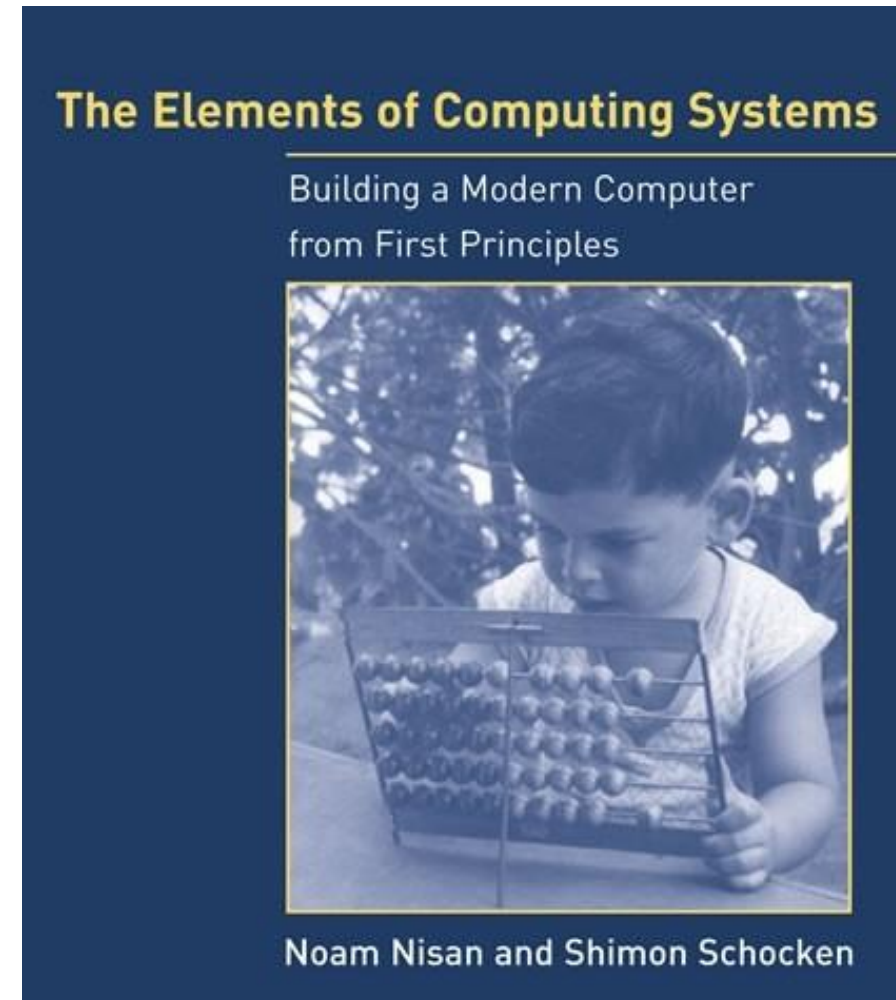
- (IAMET-406) – bring your own laptops
- 1st lab **will be simple for most of you**, just check you can log in, download software, run software, edit files, do basic checks etc.
- Attendance will be registered
 - **Warning letter will be issued if miss too many labs.**
 - If you can't make your day, try arranging a swap with someone (also let us know...)

Outline

- Who is teaching the module?
- Admin
- How will this module be taught and assessed?
- What is this module about?

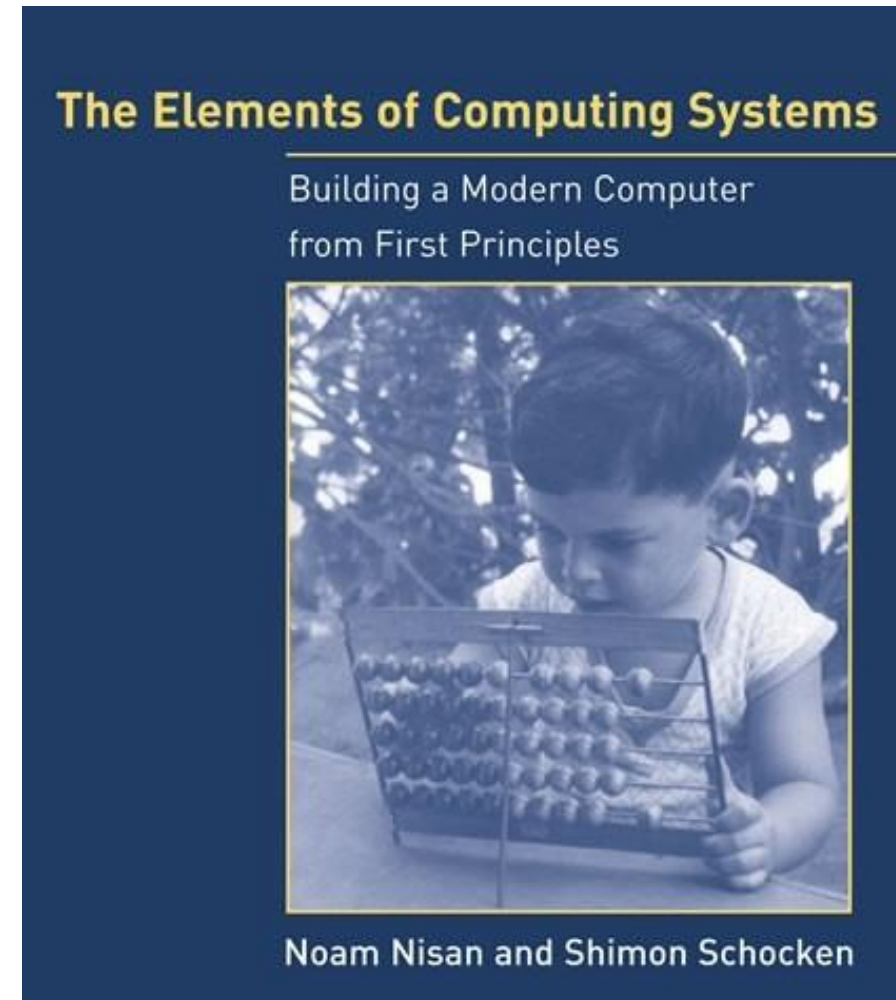
How will this module be taught

- Recommended Text books
 - Noam Nisan and Shimon Schocken. The elements of computing systems: building a modern computer from first principles. MIT press, 2021.
- Lectures
 - Will loosely follow chapters 1-8 of Nisan and Schocken.
- Labs
 - Also will loosely follow tasks from that book.



Textbook

- Nisan, Noam, and Shimon Schocken. *The elements of computing systems: building a modern computer from first principles*. MIT press, 2021.
- Patterson, David A., and John L. Hennessy. *Computer Organization and Design MIPS Edition: The Hardware/Software Interface*. Newnes, 2013.
- More online information:
 - From Nand to Tetris, Building a Modern Computer From First Principles.
 - <https://www.nand2tetris.org/>



How will this module be assessed

- Coursework
 - 2 pieces (Coursework 1, Coursework 2) - 50% of final mark
 - 1 month to finish each coursework (roughly)
 - Online submission
 - You can submit it early...
- Exam
 - 50% of final mark
 - 1 hour
 - More updates on exam during the last lecture

Coursework

- This will involve writing software using the nand2tetris simulators.
 - HardwareSimulator
 - CPUEmulator
 - VMEmulator
- This will be tested using test scripts
- Exact test scripts will **NOT** be given to you, but sample test scripts will be provided.
- The software will be described
 - Writing software according to a set of requirements
- Previously students have been asked to write a simple calculator.
 - One student finished the coursework in **one afternoon!**

More on coursework

- You can ask clarification questions on Moodle but...
 - You can't ask “is this right?” We do not assess your CW until you submit it.
- You can ask for help on debugging your programme, but
 - Not on implementing the coursework.
- Make sure that you attend all the lab sections and are able to do the lab exercises independently.
 - Coursework questions are usually built on the lab exercises.
- A good enough coursework score will mean you can pass the module before you even do the exam. (**50% of the final mark!**)

Previous assessment results

	2018/2019	2019/2020	2020/2011	2021/2022	2022/2023	2023/2024
1st	47%	47%	47%	60%	49%	50%
II-1	19%	24%	19%	13%	16%	25%
II-2	11%	12%	11%	10%	11%	7%
3rd	11%	8%	11%	8%	7%	7%
Fail	12%	9%	12%	9%	17%	11%
Number of students	160	188	160	168	181	210

1st: 70 and above **II-1:** 60-69 **II-2:** 50-59 **3rd:** 40-49

Outline

- Who is teaching the module?
- Admin
- How will this module be taught and assessed?
- What is this module about?

What is this module about

- **Summary of Content**

- *This module gives a basic understanding of the **fundamental architecture** of computers. This module will introduce how the simple **building blocks of digital logic** can be put together in different ways to build an entire computer. It will also show how modern computer systems are constructed of **hierarchical layers** of functionality which build on and **abstract the layers** below. You will spend four hours per week in lectures and computer labs for this module.*

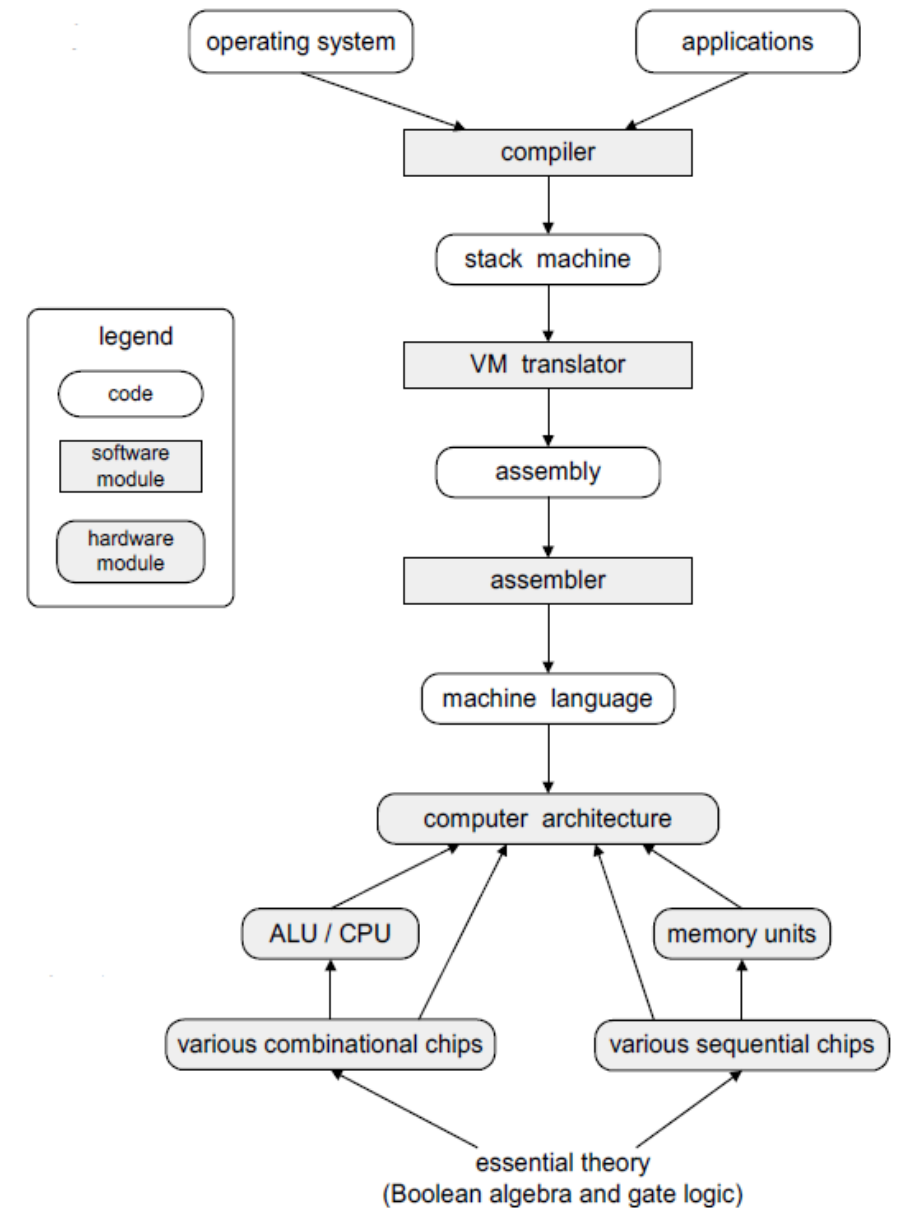
- **Education Aims**

- *To give a broad understanding of the **internal operation** and **structure** of computer. To show how a computer is built up from a relatively simple **digital circuit** by successive elaboration to form a number of **logical layers of functionality**; to show that hardware and software are often equivalent in this context. To allow the student to appreciate the typical facilities and mechanisms which underlie the operation of various high-level programming operations and facilities. To allow the student to appreciate the key conceptual steps which underlie the evolution or realisation of a **conventional stored-program digital computer**.*

- http://modulecatalogue.nottingham.ac.uk/ningbo/asp/main_search.asp

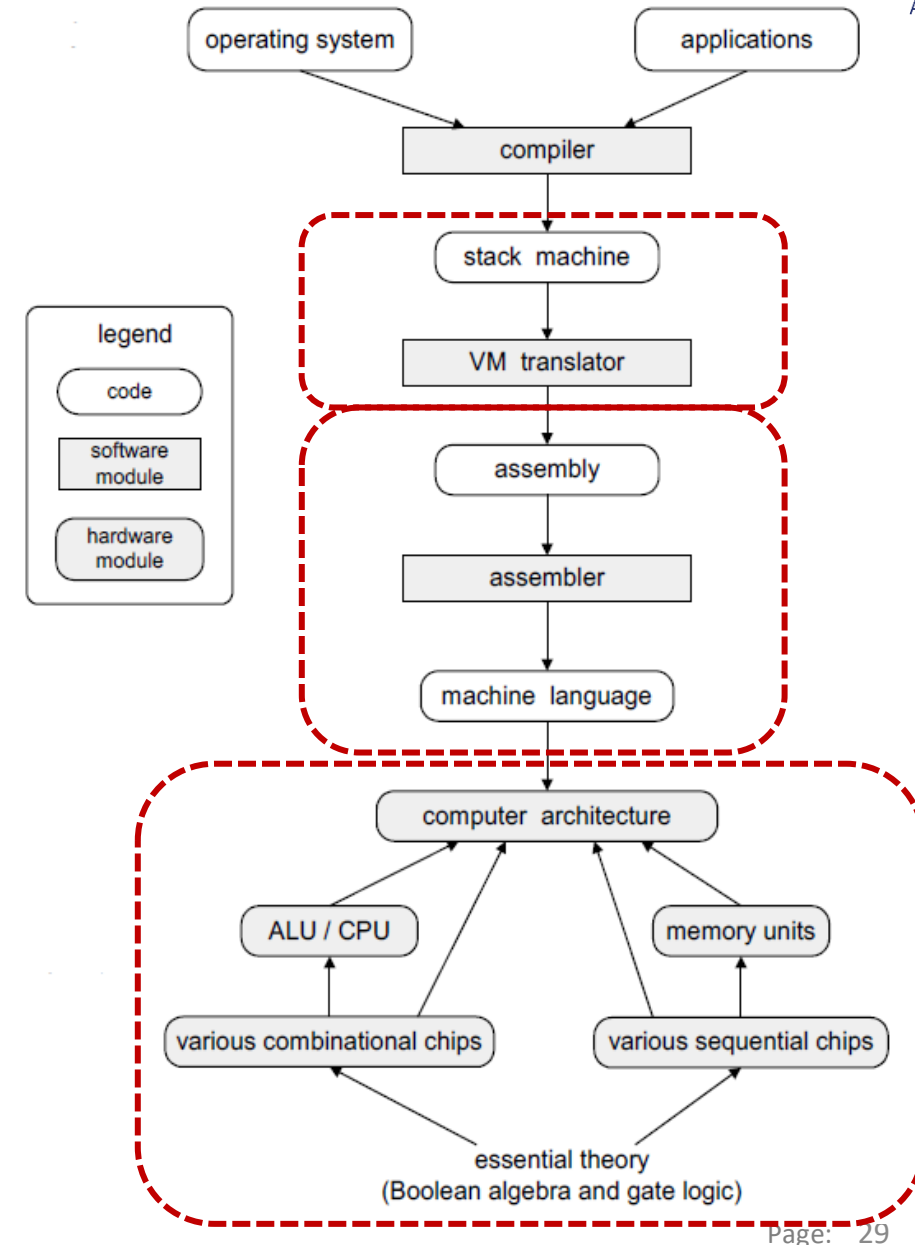
What is this module about

- A **10-credit module**, bottom up approach to how computers work
- Starts at gate logic 11010101010110
- How to build simple computing devices
- How join these devices together
- ...all the way up to high level systems
- Compilers, OS, Networking, High level languages taught elsewhere, will be only dealt with briefly here.



Programming Languages

- Hardware Description Language (HDL)
 - Use to construct the hardware, e.g. chips and ALU
 - Hardware Simulator.
- Machine Language
 - Binary machine language
 - Assembly language (Hack assembly)
 - CPU Emulator.
- Virtual Machine Language
 - Stack machine mode
 - VM Emulator.



Levels of Programming Code

- Compiler – A program that translates high-level language statements into assembly language statements
- Assembly language – A symbolic representation of machine instructions
- Assembler – A program that translates a symbolic version of instructions into the binary versions
- Machine language – A binary representation of machine instructions
- Instruction – A command that computer hardware understands and obeys

High-level
language
program
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

Compiler

Assembly
language
program
(for MIPS)

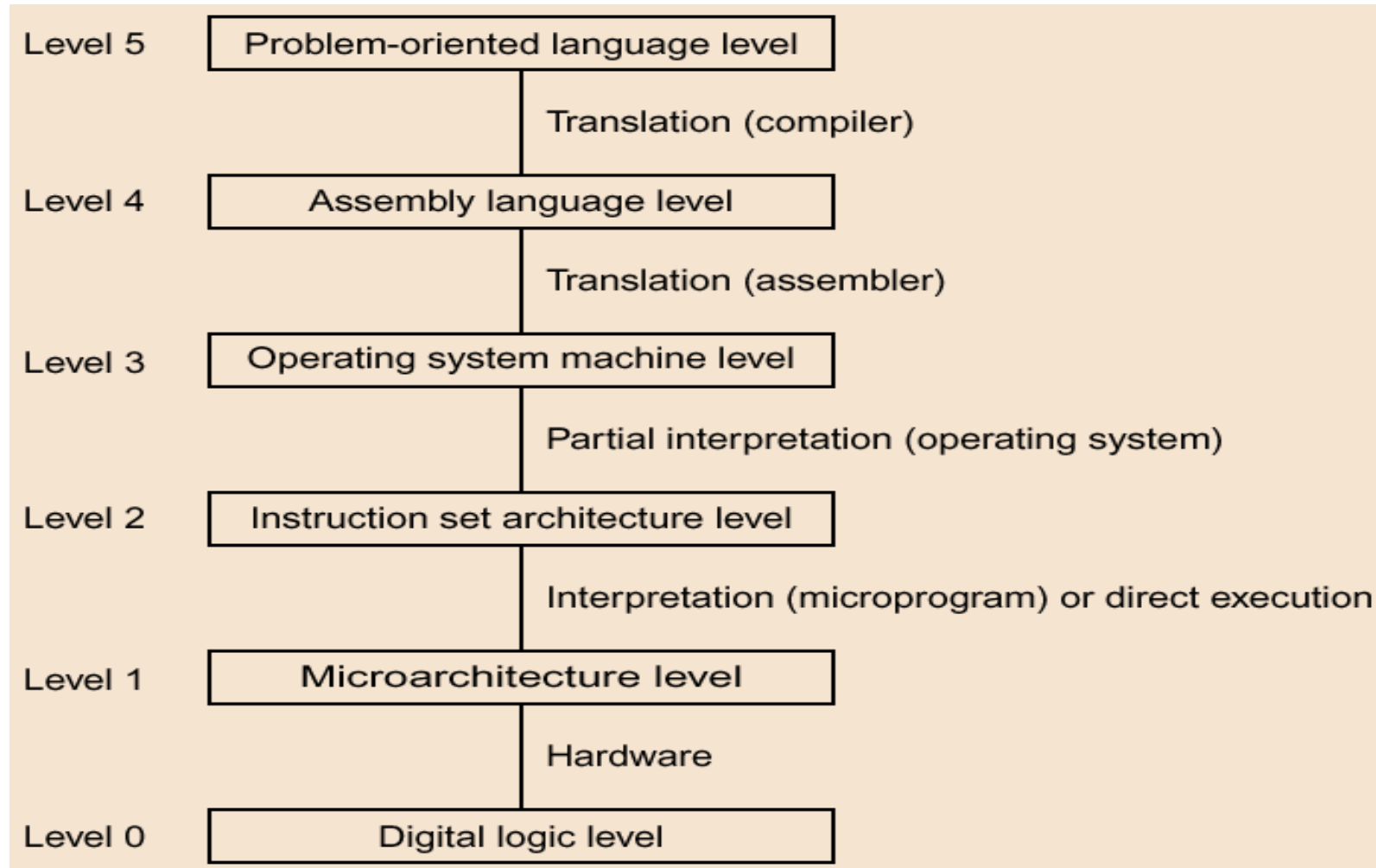
```
swap:
    multi $2, $5, 4
    add   $2, $4, $2
    lw    $15, 0($2)
    lw    $16, 4($2)
    sw    $16, 0($2)
    sw    $15, 4($2)
    jr    $31
```

Assembler

Binary machine
language
program
(for MIPS)

```
000000001010001000000000100011000
00000000100000100001000000100001
10001101111000100000000000000000
100011100001001000000000000000100
101011100001001000000000000000000
101011011110001000000000000000100
000000111110000000000000000001000
```

Computer Systems Hierarchy



Learning Outcome

- To be able to understand simple assembly language programs.
- To understand the major components (especially hardware) that make up a computer system.
- To be able to program in assembly language.

Content Plan (provisional!)

	Week beginning	Lecture	Lab
Week 1	16 th Sep	No lecture	No lab
Week 2	23 rd Sep	Introduction to COMP1036 (Libin Hong)	No lab
Week 3	30 th Sep	No lecture	No lab
Week 4	7 th Oct	Boolean Logic (Libin Hong)	Download and test Nand2tetris tools (Libin Hong)
Week 5	14 th Oct	Boolean Arithmetic (Libin Hong)	Overview of the Hardware Description Language (HDL) (Libin Hong)
Week 6	21 st Oct	Sequential Logic and ALU (Libin Hong)	Combinational logic: using the logic gates (Libin Hong)
Week 7	28 th Oct	Memory and CSF History (Libin Hong)	Sequential logic/ALU (Libin Hong)
Week 8	4 th Nov	Machine Language I (Wooi Ping Cheah)	Coursework 1 (Libin Hong)
Week 9	11 th Nov	Machine Language II (Wooi Ping Cheah)	Machine language (Wooi Ping Cheah)
Week 10	18 th Nov	Input-Output & Assembler (Wooi Ping Cheah)	Assembler: Basic language translation techniques (Wooi Ping Cheah)
Week 11	25 th Nov	Virtual Machine I (Wooi Ping Cheah)	Virtual machine: abstraction (Wooi Ping Cheah)
Week 12	2 nd Dec	Virtual Machine II (Wooi Ping Cheah)	Virtual machine: implementation (Wooi Ping Cheah)
Week 13	9 th Dec	Revision (Libin Hong & Wooi Ping Cheah)	Coursework 2 (Wooi Ping Cheah)
Week 14	16 th Dec	No lecture	No lab

Why is this module important

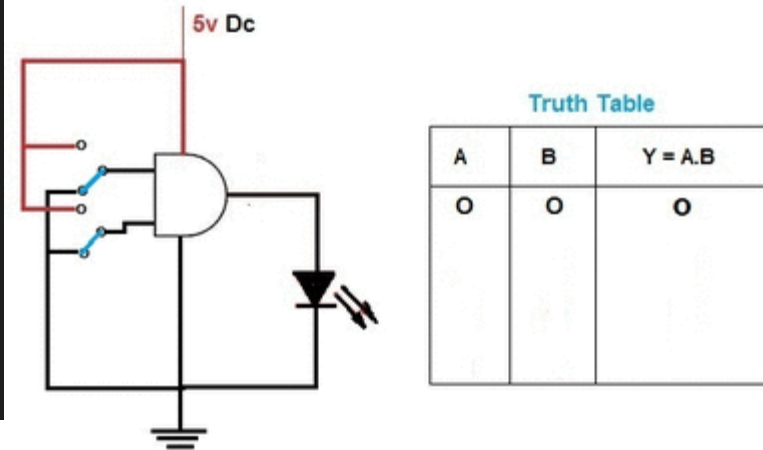
- Car analogy –
 - Most modules of CS degree is about how to ‘drive the car’ (Programming, DB, SE...)
 - how to drive it faster, safer, more efficiently
 - This module is how the car is **built** and even how to build the components that go into building the car
 - Not essential but most good drivers know how the car actually works



Why CSF?

```
/* C Hello World program */
#include<stdio.h>

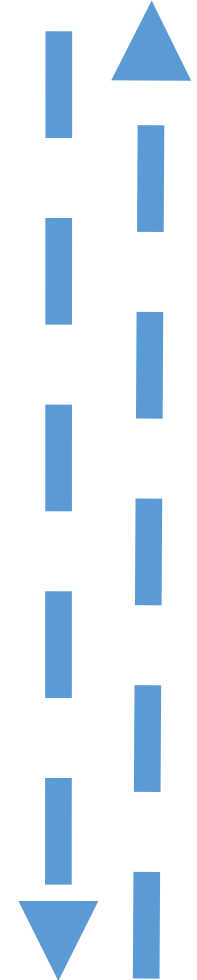
int main()
{
    printf("Hello World!\n");
    return 0;
}
```



- Programming is about giving a computer instructions that we want it to perform
- What happens when we run 'Hello World'? How does it make it appear on the screen?
- What happens when we press a key on the keyboard?

What does it take to run a program

- **Program is just a set of dead characters in a text file**
- This text file needs to be parsed
- Re-expressed as a lower level language
 - Convert something human friendly into something human unfriendly
 - Produces another text file, containing machine level code
- This machine code realised by a hardware architecture
 - Hardware devices – registers, memory, ALU
- Hardware devices constructed using logic gates
- Logic gates constructed from primitive gates (eg. NAND)
- Primitive gates constructed from switching devices (eg. Transistors)
-Physics.....



Understanding Performance

- The performance of a program depends on a combination of the effectiveness of hardware or software components
- Algorithm
 - Determines the number of I/O operations executed
- Programming language, compiler, architecture
 - Determine number of machine instructions executed per operation
- Processor and memory system
 - Determine how fast instructions can be executed
- I/O system (including OS)
 - Determines how fast I/O operations may be executed

Why CSF?

- Long ago, every computer specialist had a complete, transparent understanding of the inner workings of computers, this has been somewhat lost – the computer is a black box even to many modern computer science students
- The best way to understand computers is to build one from scratch (transistors -> logic gates -> adders -> ALU and Registers -> CPU)
- Without seeing how the interaction of elementary components work will leave you with an uneasy feeling that you don't fully know what's going on inside computers

Why CSF?

- Makes it possible to write computer programs that are faster and smaller
- Allows programmers to balance performance and relative cost of operations with appropriate programming choices
- Prepare for other CS courses, such as programming, compilers, operating systems, etc.

Summary

- From logic gates to CPUs (in terms of hardware)
- From machine language to virtual machine, compiler and all the way to high-level program.
- From hardware all the way to high-level program (C/C++).

