Comp1036 (CSF)
Computer Fundamentals

Dr Tianxiang Cui and Dr Wooi Ping Cheah

WeChat group

- Scan the barcode below to join the CSF 2022/23 WeChat group

Computer Fundamentals
2022-2023

Valid unti Sept 26.

该二维码7天内(9月26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 programmes 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. Tianxiang Cui
  - Research interests: Computational Intelligence, Operation Research, Machine Learning, Reinforcement Learning
  - Previously worked in Huawei (autonomous driving) and PingAn (quantitative trading)

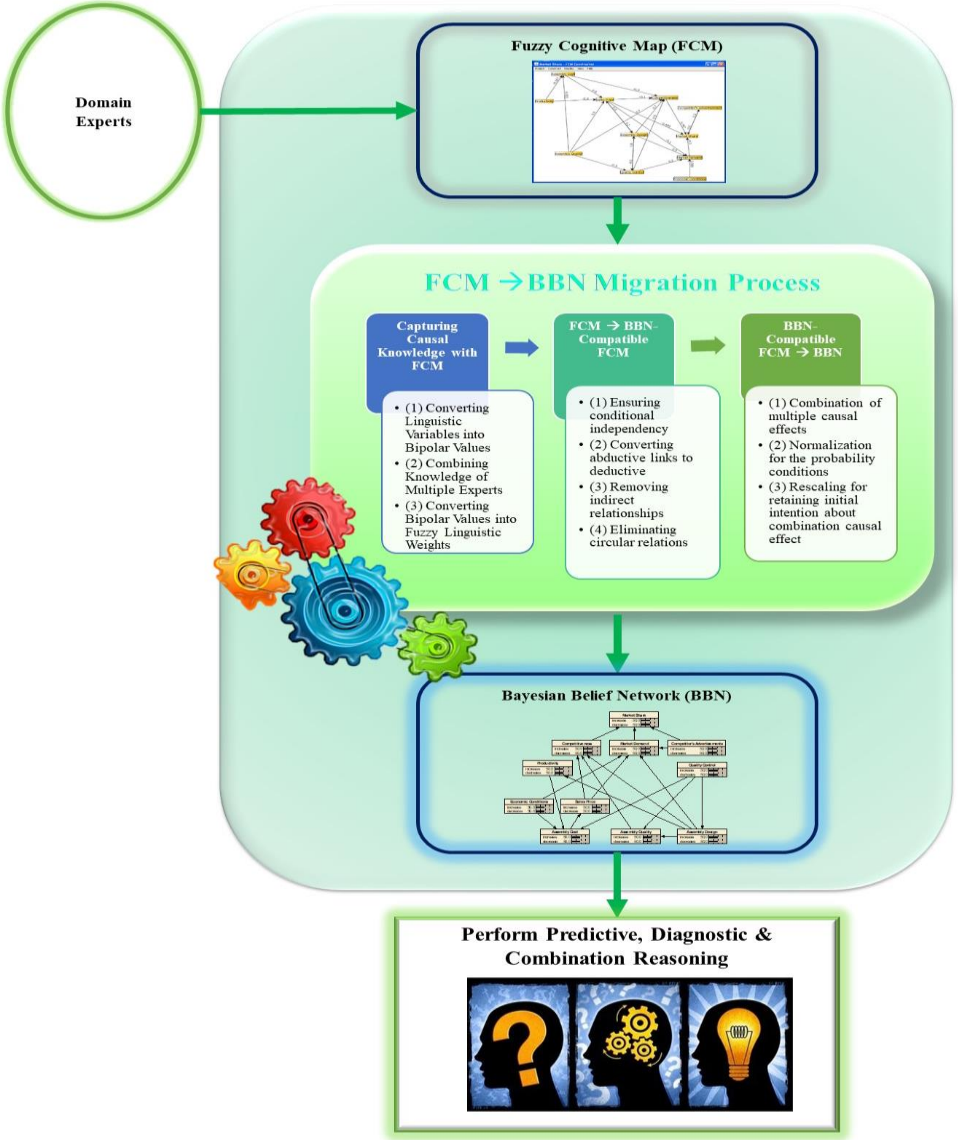
- Dr. Wooi Ping Cheah
  - PhD degree from Chonnam National University, South Korea.
  - Research interests: Soft Computing, Knowledge Engineering, Software Engineering, Artificial Intelligence, Programming Paradigms.

Research interests - Tianxiang

- Operation Research
  - Deal with the development and application of advanced analytical methods to improve decision-making
- Computational Intelligence
  - Set of nature-inspired computational methodologies and approaches to address complex real-world problems
- Machine Learning
  - An area of AI concerned with development of techniques which allow machines to learn
- Reinforcement Learning
  - Framework for learning to solve sequential decision making problems

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 - Tianxiang

- Computer Fundamental (Qualify Year)
- Software Engineering (Qualify Year)
- Computer Vision (Part II)

Teaching - Wooi Ping

- Computer Fundamental (Qualify Year)
- Programming and Algorithms (Qualify Year)

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)
- DT project (Part II)

Contact hours, Tianxiang

- Contact
  - Office: PMB426
  - Email: tianxiang.cui@nottingham.edu.cn
- Office hours
  - Thursday, 15:00-17:00
- 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, 15:00-17:00
- 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
  - Tuesday, 16:00-18:00, DB-A05+.
- Lab
  - Tuesday, 13:00-15:00, **IAMET-406+**. (Group 1)
  - Friday, 14:00-16:00, **IAMET-406+**. (Group 2)
- Tencent meeting live streaming (id: 911-047-1316)
- Recording:
  - The recorded lectures/labs will be uploaded to Panopto and shared on Moodle

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.
- https://moodle.nottingham.ac.uk/course/view.php?id=118756

Lectures

- Start at 16:00 sharply
- Finish at 16:50 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

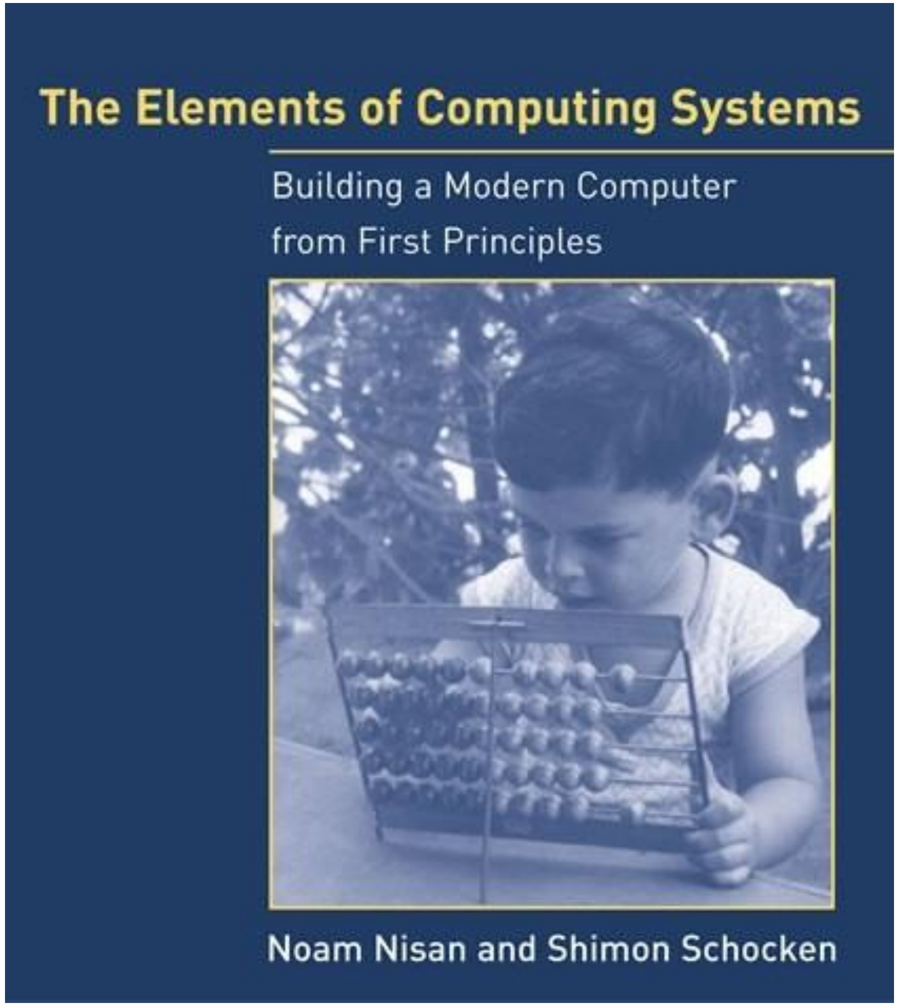
- (IAMET-406) – bring your own laptops
- 1st lab **will be simple for most of you**, just check you can log in, download software, run software, edit files, do basic checks etc.
- Attendance will be registered
  - **Warning letter will be issued if miss too many labs.**
  - If you can't make your day,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

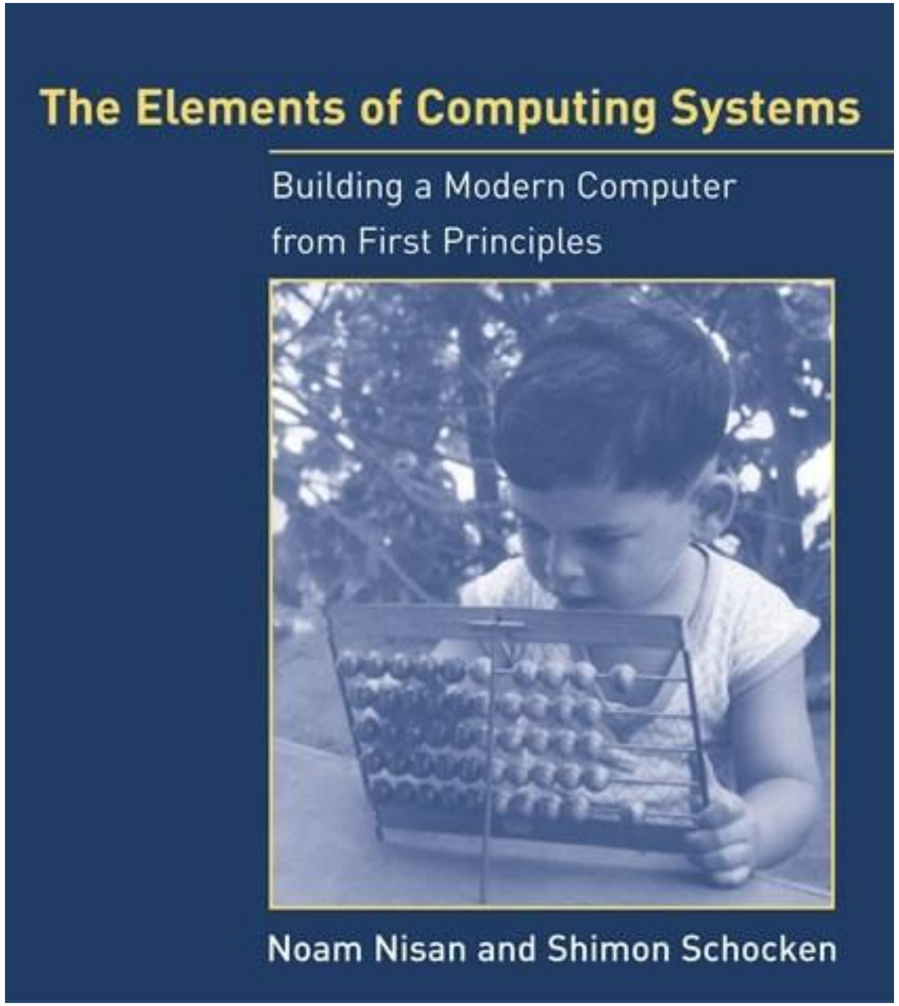
How will this module be taught

- Recommended Text books
  - Noam Nisan and Shimon Schocken. The elements of computing systems: building a modern computer from first principles. MIT press, 2005.
- 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, 2005.**
- 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 -50% of final mark
  - 1 month to finish (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

	2017/18	2018/19	2019/20	2020/21
1st	**47%**	**47%**	**47%**	**60%**
II-1	19%	24%	19%	13%
II-2	11%	12%	11%	10%
3rd	11%	8%	11%	8%
Fail	12%	9%	12%	9%
Number of students	160	188	160	168

COVID update

- Some student will be (at least) starting the module online,
  - These students will be supported by **online lectures** and **online labs via Tencent Meeting**.
- Talk to us and/or your personal tutor if you have any problems attending lectures or keeping up with the module.

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

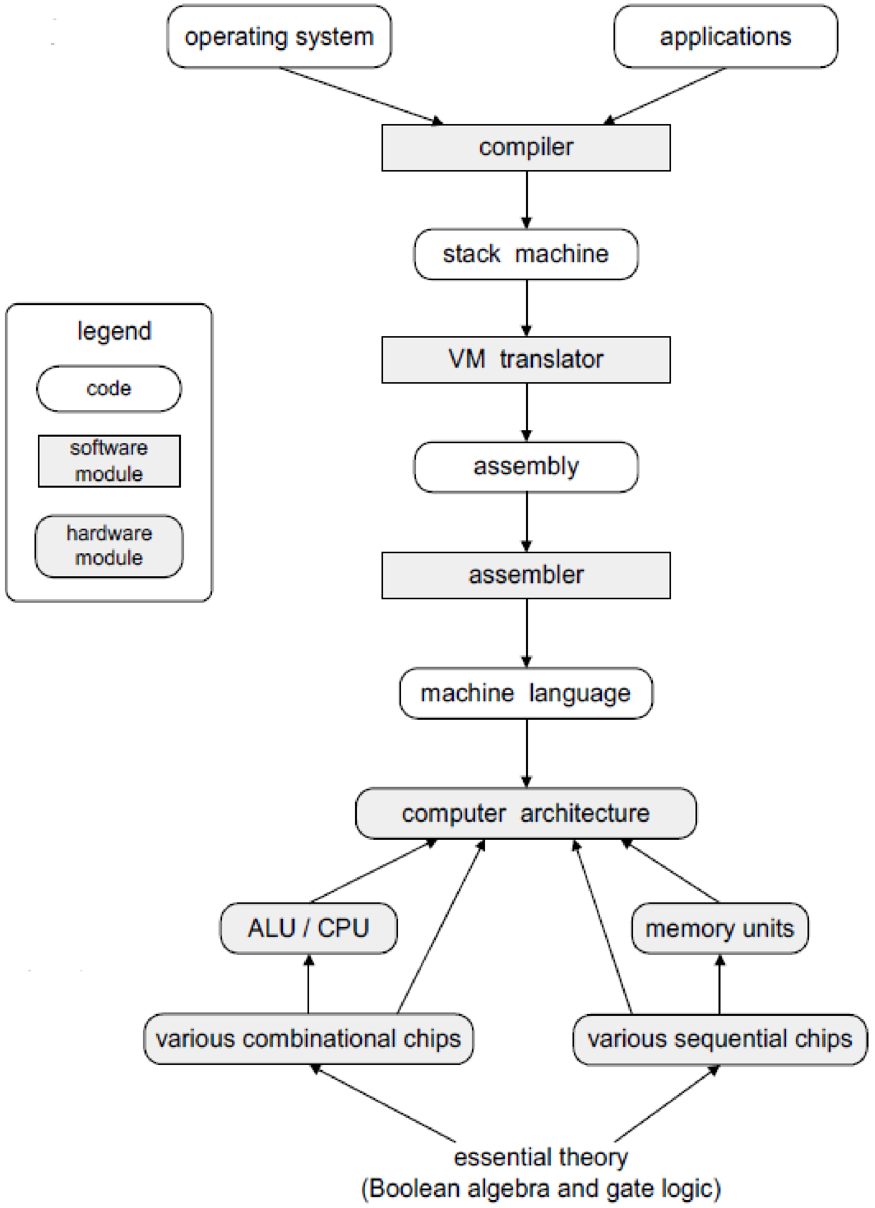
- **Summary of Content**
  - *This module gives a basic understanding of the fundamental architecture of computers. This module will introduce how the simple building blocks of digital logic can be put together in different ways to build an entire computer. It will also show how modern computer systems 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

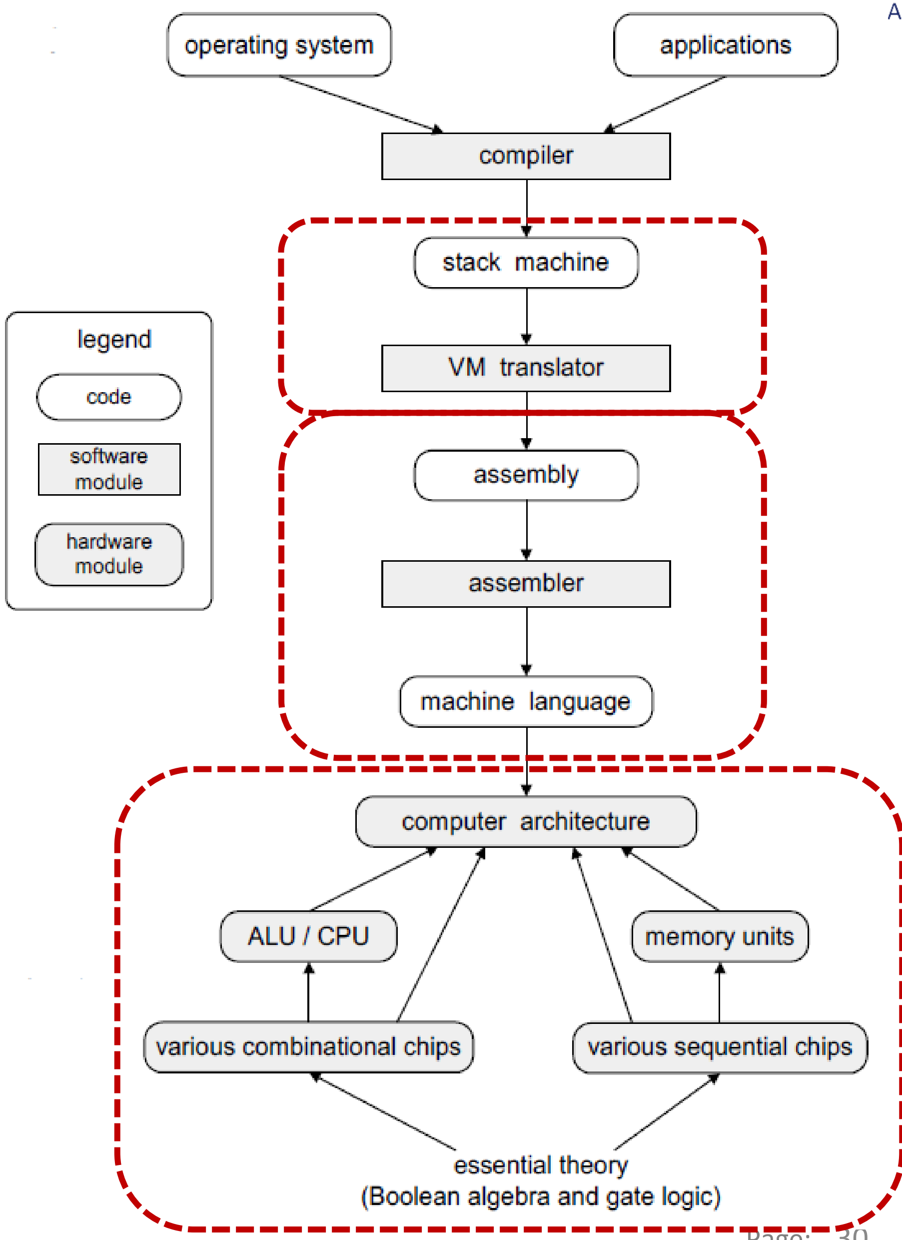
- A **10-credit module**, bottom up approach to how computers work
- Starts at gate logic 1101010101010110
- How to build simple computing devices
- How join these devices together
- ...all the way up to high level systems
- Compilers, OS, Networking, High level languages taught elsewhere, will be only dealt with briefly here.



From: Schocken, Shimon, Noam Nisan, and Michal Armoni. "A synthesis course in hardware architecture, compilers, and software engineering." *ACM SIGCSE Bulletin*. Vol. 41. No. 1. ACM, 2009.

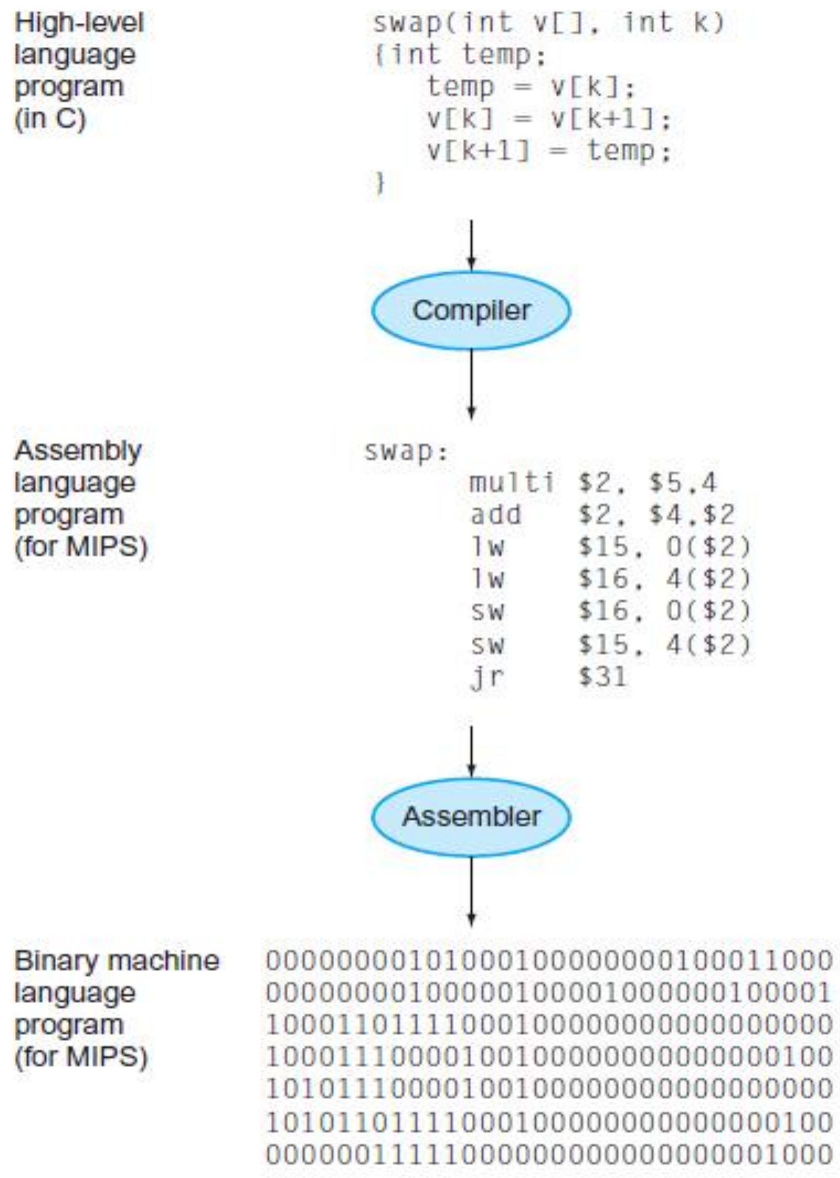
Programming Languages

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

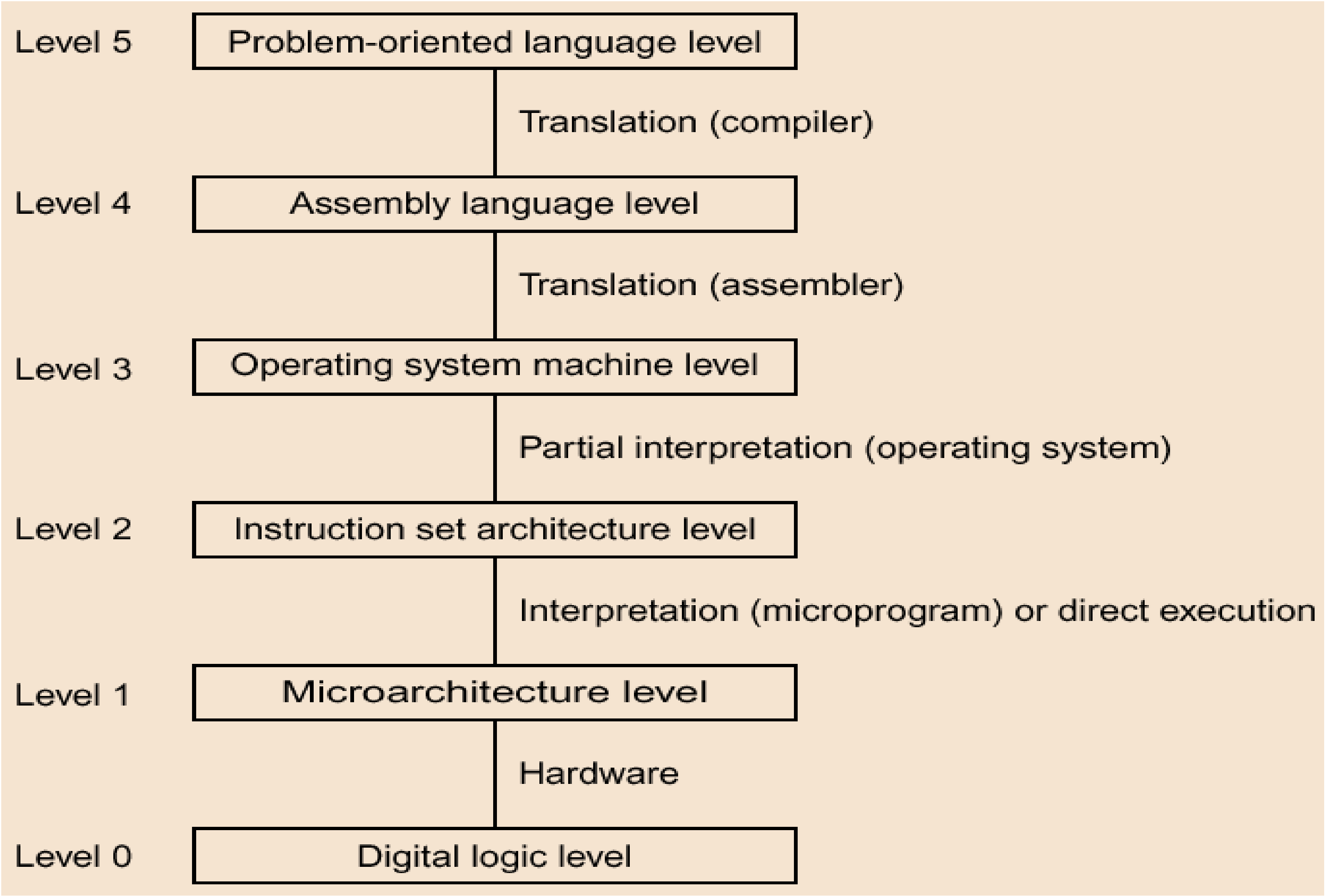


Levels of Programming Code

- 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



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	12th Sep	No lecture	No lab
Week 2	19th Sep	Introduction to COMP1036	No lab
Week 3	26th Sep	Boolean Logic	Download and test Nand2tetris tools (optional)
Week 4	3rd Oct	No lecture	No lab
Week 5	10th Oct	Boolean Arithmetic	Overview of the Hardware Description Language (HDL)
Week 6	17th Oct	Sequential Logic and ALU	Combinational logic: using the logic gates
Week 7	24th Oct	Memory and CSF History	Sequential logic/ALU
Week 8	31st Oct	Machine Language	Machine language
Week 9	7th Nov	Assembler	Coursework release
Week 10	14th Nov	Virtual Machine I	Assembler: Basic language translation techniques
Week 11	21st Nov	Virtual Machine II	Virtual machine: abstraction
Week 12	28th Nov	Compilers and High level languages	Virtual machine: implementation
Week 13	5th Dec	Revision	Coursework deadline
Week 14	12th Dec	TBD	No Lab

Why is this module important

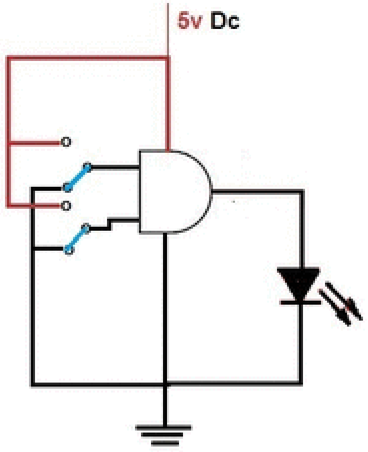
- Car analogy –
  - Most modules of CS degree is about how to ‘drive the car’ (Programming, DB, SE…)
    - how to drive it faster, safer, more efficiently
  - This module is how the car is **built** and even how to build the components that go into building the car
  - Not essential but most good drivers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

Why CSF?

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

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

}
```



Truth Table

A	B	Y = A.B
0	0	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, complier and all the way to high-level program.
- From hardware all the way to high-level program (C/C++).