



Xi'an Jiaotong-Liverpool University

西交利物浦大學

Welcome to Microprocessor Systems

An introduction to CPT210

Module Information

About the module instructor, teaching organisation.

Module in a nutshell

Year: 2024-2025

Date of Commencement: Week 2

Originating School: School of Advanced Technology

Semester: 2

Credit level: Level 2 (Year 3)

Credit value: 2.5

Students: ~544 (from all departments across the school)

Teaching Team & Office hours

Teaching staff (3)



Filbert JUWONO

Co-lecturer

- Monday 14:00 – 16:00
- Office at SC248



Lijie YAO

Module Leader

- Tuesday 16:00 – 18:00
- Office at SC564D



Kok Hoe WONG

Lab Instructor

- Wednesday 09:00 – 11:00
- Office at SD431

Aims of module

- The aim of this module is to provide students, who have no previous knowledge of microprocessor systems, with a good knowledge of how microprocessors work as well as their architecture.
- This includes developing a sufficient learning of assembly language to enable them to write and analyse simple programmes.
- The students learn about different data formats and basic microprocessor concepts, such as assembly, functions, and stacks.

Module components

- Lectures

- Introduction
- Data Representations
- ARM Assembly
- Assemblers & Instruction Encoding
- ARM Functions & Stacks
- Pipelining & Exceptions Handling



Lecture 1 - 4 (Week 2 - 5)



Lecture 5 - 8 (Week 9 - 12)

- Labs

- Hands-on ARM Coding Practice using VisUAL2



Labs 1 - 2 (Week 6 & 8)

- Tutorials

- Assessment Trial-run & Formal Assessment



Tutorials 1 - 2 (Week 12 & 13)

Method of delivery (1/2)

Lectures (8) - **Onsite**

- **Week 2-5 & 9-12**
 - D1/1 & D2/1: Tuesdays, 11:00 - 12:50, SB123
 - D1/2 & D2/2: Tuesdays, 14:00 - 15:50, SD102

Tutorials (2) & Assessment - **Onsite**

- **Week 12 & 13**
 - D1/1: Fridays, 09:00 - 10:50, SC169
 - D1/2: Fridays, 11:00 - 12:50, SD102

Labs (2) - **Onsite**

- **Week 6 & 8**
 - D1/1: Monday, 09:00 - 10:50, CBG13
 - D1/2: Monday, 09:00 - 10:50, CBG15E
 - D1/3: Monday, 11:00 - 12:50, CBG13
 - D1/4: Monday, 11:00 - 12:50, CBG15E
 - D1/5: Tuesday, 11:00 - 12:50, CBG13
 - D1/6: Tuesday, 11:00 - 12:50, CBG15E
 - D1/7: Wednesday, 11:00 - 12:50, CBG15E

- Check your assigned group on LM
- Come to the assigned session, with correct location, date, and time
- No change of groups.

Method of delivery (2/2)

Check frequently the [course website on Learning Mall](#):

After you log in, [CPT210](#) should appear in the list of courses
(if not, send us an email: CPT210@xjtlu.edu.cn)

Announcements will be posted on:

[Learning Mall CPT210 course website](#) (and possibly sent by email)

Check your [university e-mail account](#) at least once every day

Contacting & questioning:

Pose questions on Learning Mall

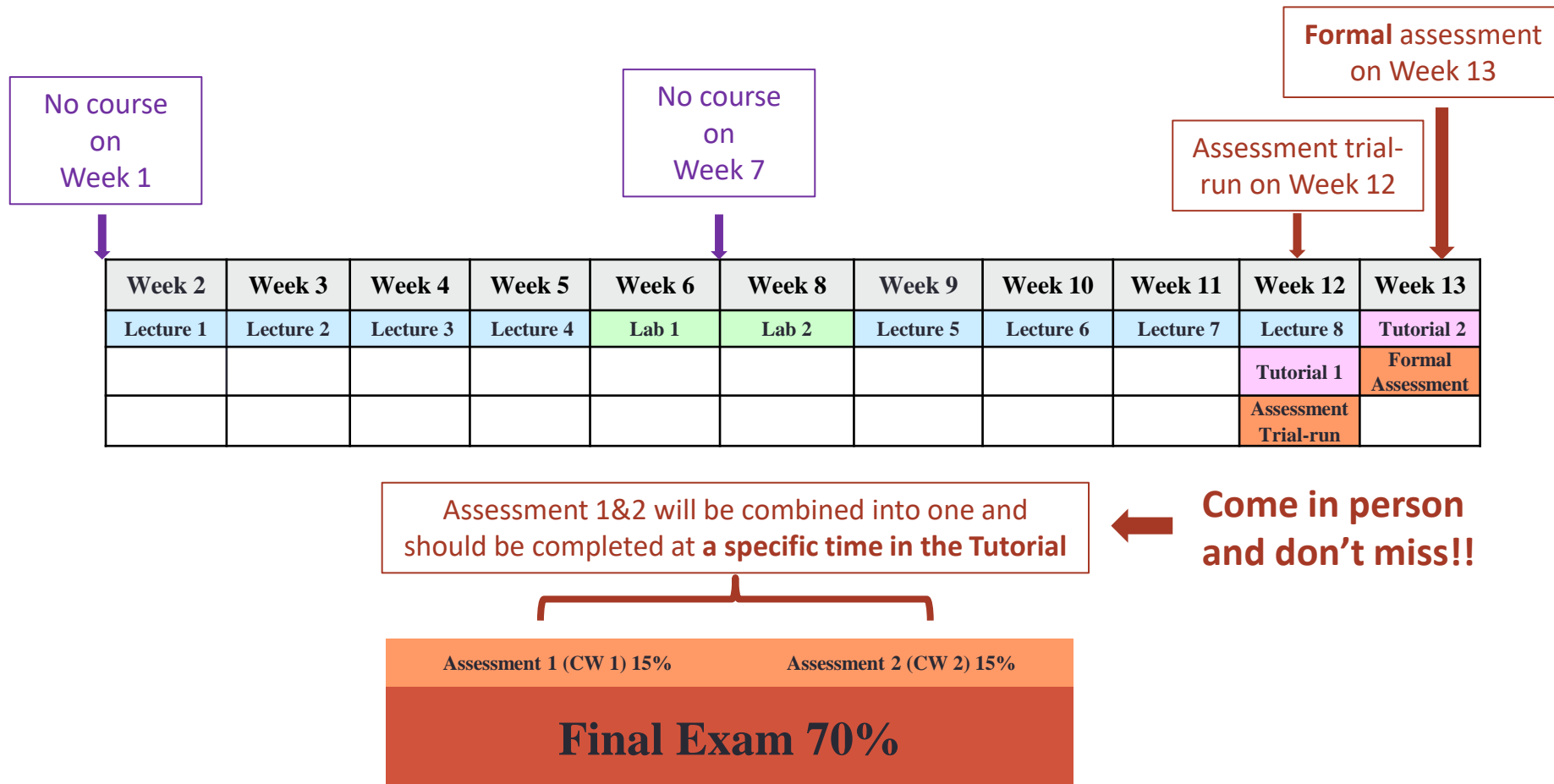
Book timeslots with TAs (reservation will be release very soon on Learning Mall)

Go to office hours and have in-person discussion with teaching team

Send emails to CPT210@xjtlu.edu

Please, always use your [university e-mail account](#) if you want to contact teaching team


Delivery schedule (Tentative)



Lab materials

- Onsite (TAs will be around)
- Tasks & example codes posted on Learning Mall
- VisUAL2: software to emulate ARM coding

Assessment 1&2

- Assessment 1 & 2 will be combined into one and take **30% of the module mark**
- **On-site** online assessment that happens during Tutorial → You **MUST** come to the **tutorial in person**
- You need to complete this assignment individually
- Photography is not allowed A red circle with a diagonal slash over a black camera icon.
- Questions are exacted from lectures and labs
- **Scheduled during a specific time of your tutorial sessions** (about 40 minutes)
- **Bring your own LAPTOPS with full batteries** (not tablet, not cell phone)
- **Missing the assessment = missing 30% of the module mark**

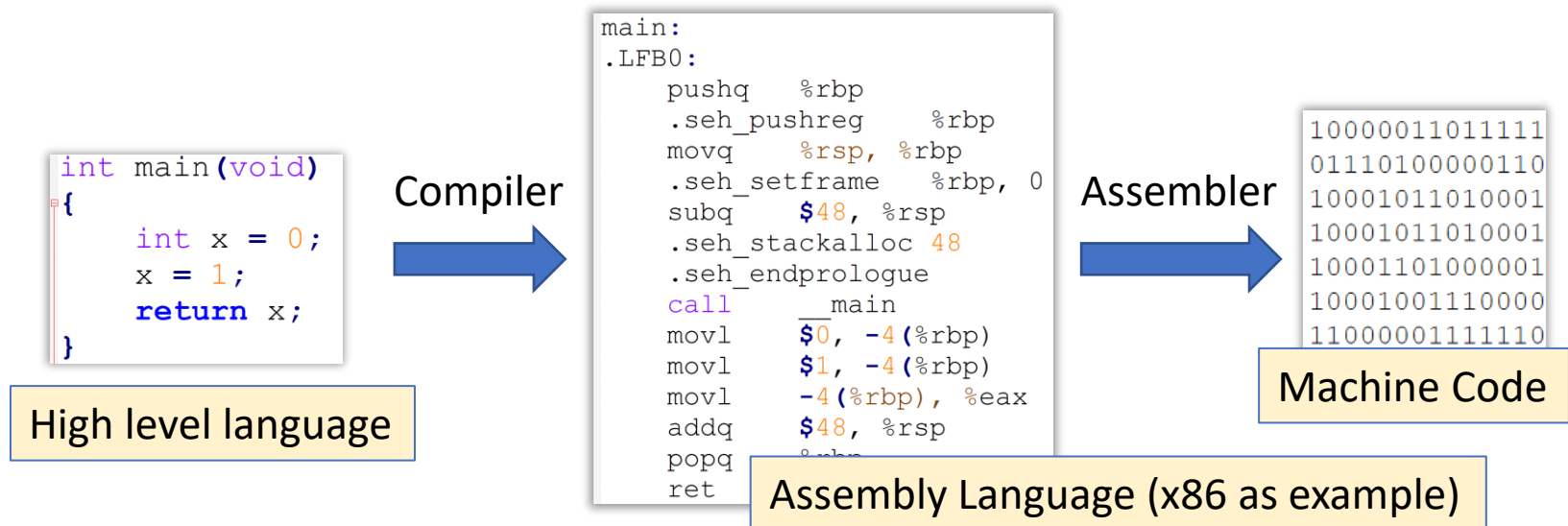


Attendance and absence

- Lectures, Labs, and Tutorials:
 - [Scan the QR code](#) on Attendance Management System (AMS)
- Absences:
 - Unreasonable absences are not allowed
 - [Support certificates](#) should be provided for each absence

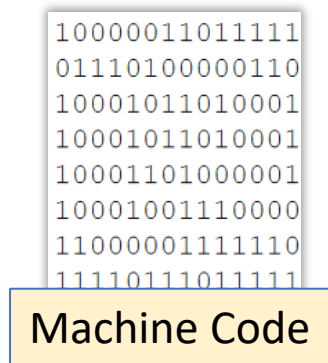
About this Module

- This module is a “bridge” between digital electronics (hardware) and high level languages (software).
- You will learn basic concepts required to understand how computers work
 - What programs are? How does your C/C++ code become the instructions that your computer understands?
 - How these instructions are executed in hardware?



About this Module

- This module is a “bridge” between digital electronics (hardware) and high level languages (software).
- You will learn basic concepts required to understand how computers work
 - What programs are? How does your C/C++ code become the instructions that your computer understands?
 - How are these instructions executed in hardware?



Focus of this Module

- We will focus on Instruction Set Architectures and their use through **Assembly Language Programming**.
- The architecture studied in this module will be **ARM (Advanced RISC Machine)**.

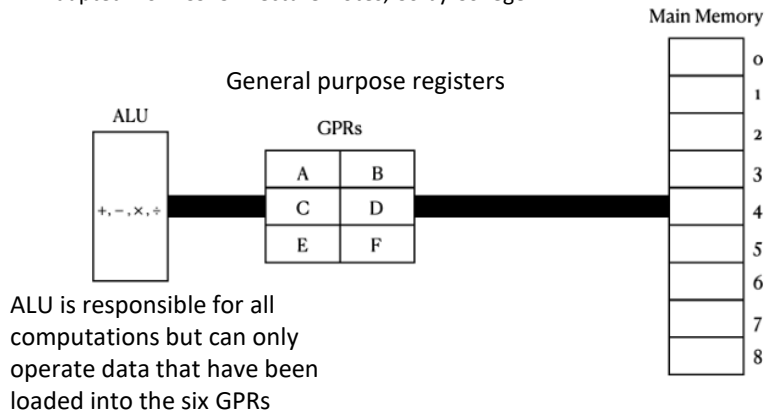
RISC vs CISC

- Microcomputer systems are designed according to which of the two components (hardware or software) should be optimized
 - Complex Instruction Set Computing (CISC) architecture
 - Reduced Instruction Set Computing (RISC) architecture

RISC vs CISC

(An Illustration)

Adapted from CS232 Lecture Notes, Colby College



Goal: calculate the product of two numbers, one stored at the location 2 and the other stored at the location 5 in the main memory, and store the product back at the location 2

CISC

Instruction: MUL 2 2 5

- Complex instruction
- Do not require the programmers to explicitly call any “load” or “store” functions to load the numbers from the main memory to GPRs or store the computing results back to the main memory

RISC

Instructions:

```
LOAD A 2 # load the number at the location 2 to register A
LOAD B 5 # load the number at the location 5 to register B
PROD C A B # conduct A x B and save the result back to C
STORE 2 C # store the value at C back at the location 2
```

- Idea: to make hardware simpler → “simple instructions”
- Compilers for RISC machines have to do more work
- More RAM is needed to store instructions

Introduction to Microprocessor Systems

Evolution of technology


Design of ISA

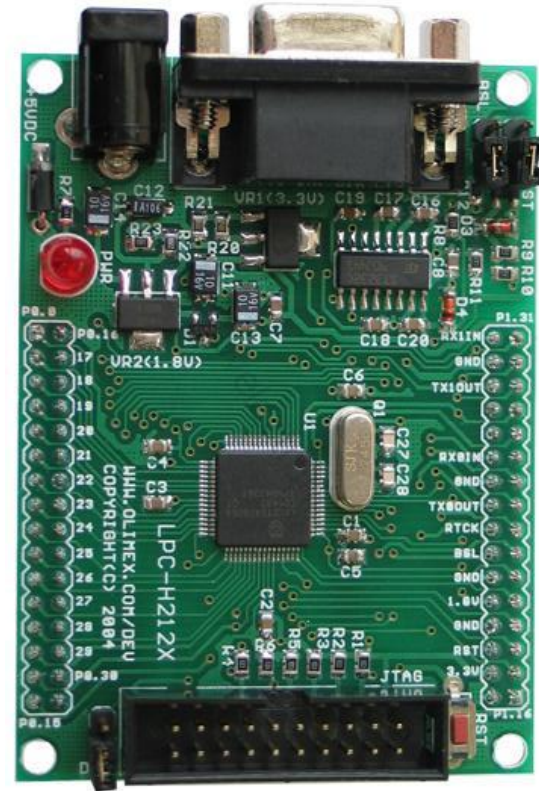
The Evolution of Computers

- Electronic Delay Storage Automatic Computer ([EDSAC](#))
 - First [practical](#) programmable computer, by UK. In Cambridge 1949.
 - On average, processes 650 instructions per second.
 - 1024 17-bit words of memory in [mercury ultrasonic delay lines](#).
 - 3000 valves, 12 kW power consumption, occupied a room 5m by 4m.
 - Early use to solve problems in meteorology, genetics and X-ray crystallography.



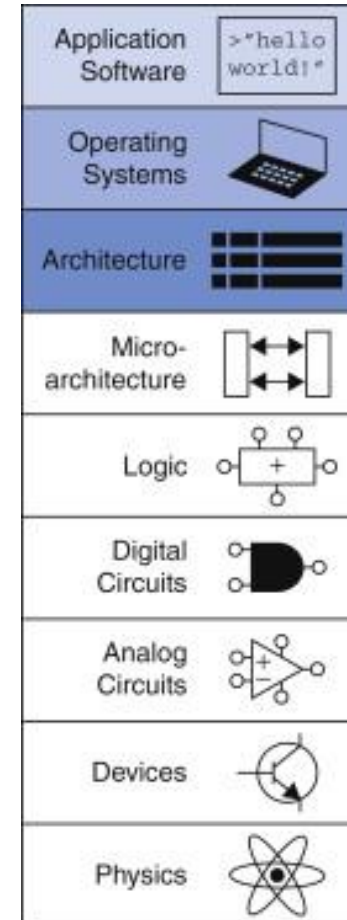
The Evolution of Computers

- ARM7
 - Up to 130 million instructions per second. 1995-2011
 - One of the most successful embedded processor.
- The picture shows LPC-H2124 header board. 
 - 256KB of program flash
 - 16KB of RAM.
 - The square chip in the middle is the microcontroller
- Original ARM design:
 - Steve Furber, Acro [RISC](#) Machine
 - Cambridge, 1985
- Evolution of computer architecture



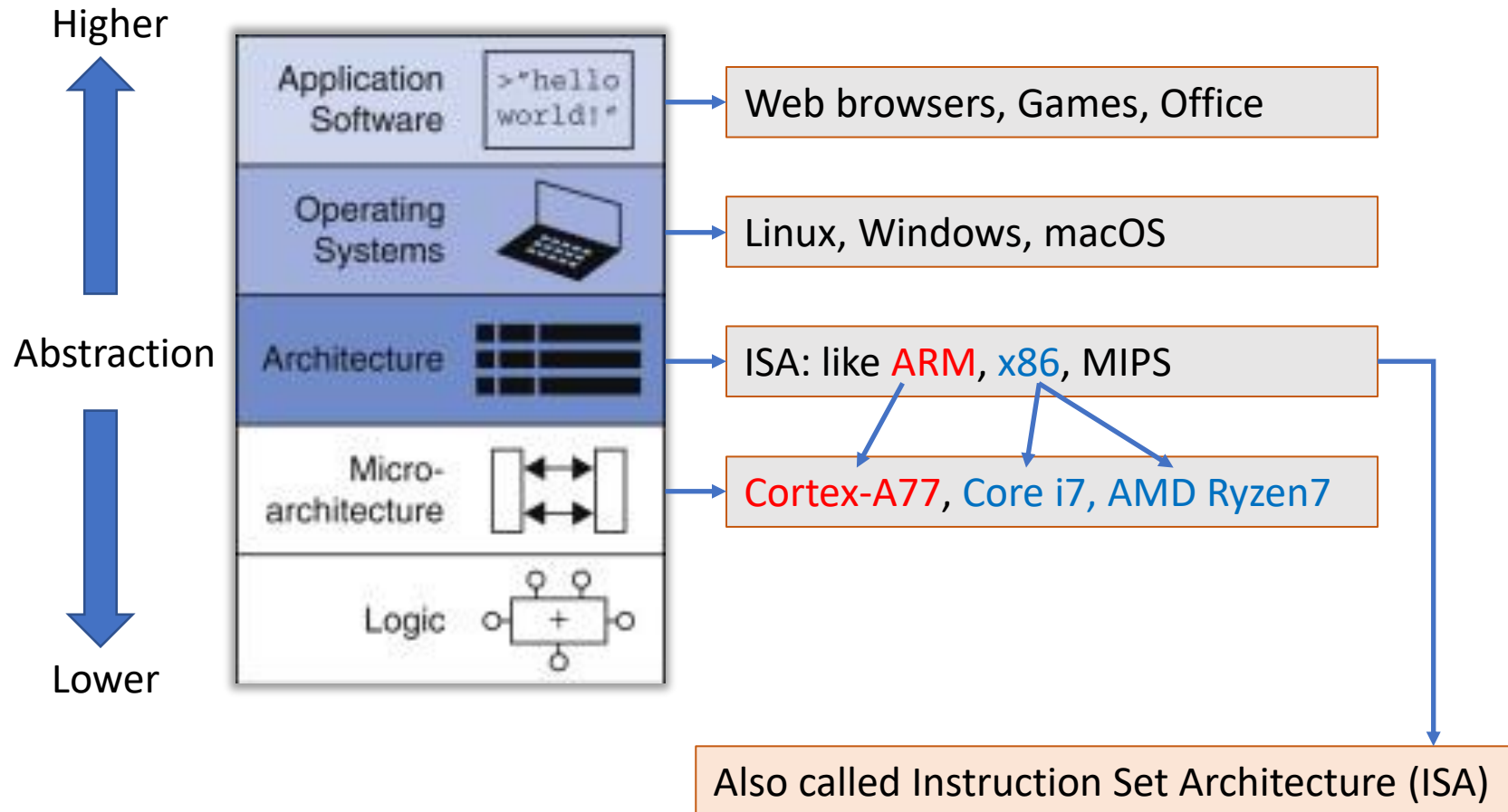
What is Computer Architecture?

- *Computer architectures* represent the means of interconnectivity for a computer's hardware components as well as the mode of data transfer and processing exhibited.
 - Paul J. Fortier, Howard E. Michel, in [*Computer Systems Performance Evaluation and Prediction*](#), 2003
- The *architecture* is the programmer's view of a computer. It is defined by the instruction set (language) and operand locations (registers and memory). Many different architectures exist, such as ARM, x86, MIPS, SPARC, and PowerPC.
 - Sarah L. Harris, David Money Harris, in [*Digital Design and Computer Architecture*](#), 2016



What is Computer Architecture?

Architecture is about design, not just hardware.



Instruction Set Architecture (ISA)

- Instruction (or Operation Code) set.
 - ADD, SUB, RSB, etc.
- Organisation of programmable storage.
 - Where data is stored.
- Mode of addressing and accessing data items and instructions.
 - How data is stored/retrieved.
- Behaviour on exceptional conditions, e.g.:
 - Hardware divide by zero.
 - An interrupt occurred
- ISA is about what the computer does (not how it does).

Instruction Set Architecture (ISA)

- Examples of ISAs:
 - **8086/Pentium (x86) ISA:** Widely used, compatible with a lot of operating systems and software.
 - **ARM ISA:** Supports highly optimising compiler & operating system software for embedded applications.

x86 instruction

ADD	Add numbers
AND	Logical AND
MOV	Copy data between memory blocks/registers

ARM instruction

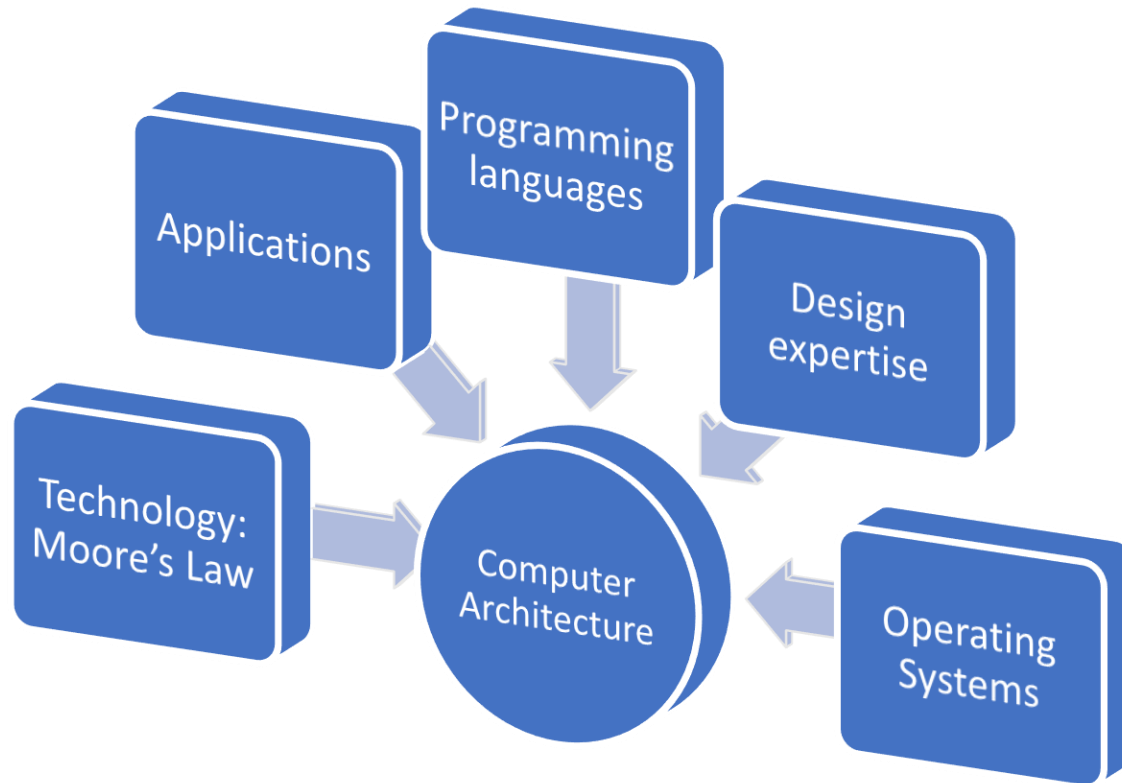
ADD	Add numbers
AND	Logical AND
LDR	Load from memory (can't save to memory)
STR	Store to memory (can't load from memory)

- Different implementations of the same ISA can run identical software.
 - Your compiled C program can run on any x86 computer as long as the operating system is the same and libraries are satisfied.

Discussion

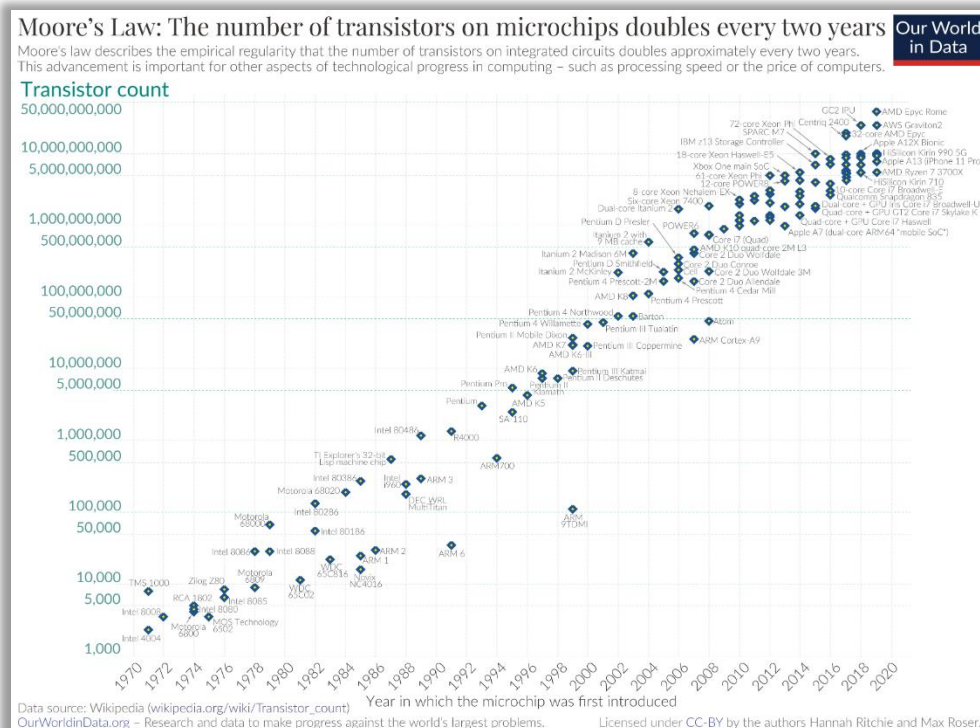
- Discuss the questions below based on your understanding and knowledge so far.
- Computer architecture affects:
 1. CPU design?
 2. Compiler design?
 3. Energy efficiency of the computer?
 4. Maximum memory size?
 5. UI design of the operating system (OS)?
 6. Security of the OS?

What Factors Influence Computer Architecture?



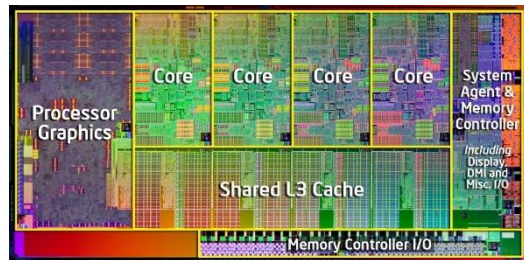
Moore's Law

- In 1965, Gordon E. Moore, co-founder of Intel, postulated that the number of transistors that can be packed into a given unit of space will double about every two years. (See on [Wikipedia](https://en.wikipedia.org/wiki/Moore's_Law))



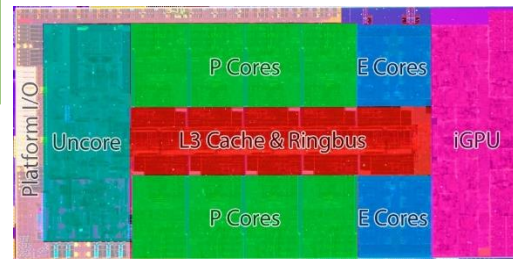
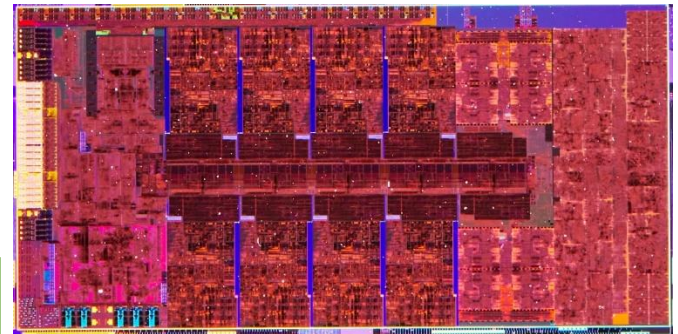
Moore's Law

- VLSI (very large scale integration) has increased speed and density of CPUs by shrinking dimensions
 - This is limited by size of atoms (Quantum effects), so will stop.
- Moore's Law predicted a doubling of computing power every two years.
 - This has held true but will stop soon without a change in technology or architecture (Quantum computing).



i7-2600k: 4 cores

i7-12700k:
8 P cores
8 E cores



Micro-architectures of ARM

- There are many architectures developed by ARM.
- Each architecture has many implementations (micro-architectures).
- Each micro-architecture has its own targeted purpose.
 - Cortex have A, M and R series.

Cortex-A	Cortex-R	Cortex-M
High performance	Real-time response	Low power usage
PCs, laptops, servers, networking equipment, and supercomputers.	medical equipment, vehicle steering, braking and signaling.	IoT and embedded devices, such as wearables and small sensors.



Micro-architectures of ARM

ISA	Cortex-A	Cortex-M	Cortex-R
ARMv7	Processor cores: <ol style="list-style-type: none"> 1. Cortex-A7 2. Cortex-A9 3. Cortex-A17 Products: <ol style="list-style-type: none"> 1. Nvidia Tegra 2 2. HiSilicon K3V2 	Processor cores: <ol style="list-style-type: none"> 1. Cortex-M0 2. Cortex-M3 Products: <ol style="list-style-type: none"> 1. Texas Instruments F28 2. Realtek RTL8710 	Processor cores: <ol style="list-style-type: none"> 1. Cortex-R4 2. Cortex-R8
ARMv8	Processor cores: <ol style="list-style-type: none"> 1. Cortex-A53 2. Cortex-A57 3. Cortex-A73 Products: <ol style="list-style-type: none"> 1. Apple A7 (iPhone 5) 2. Exynos 5433 (Galaxy Note 4) 3. Nvidia Tegra X1 	Processor cores: <ol style="list-style-type: none"> 1. Cortex-M23 2. Cortex-M33 Products: <ol style="list-style-type: none"> 1. Nuvoton M2351 2. GigaDevice GD32E230 3. NXP LPC5500 4. ST STM32 L5 	Processor cores: <ol style="list-style-type: none"> 1. Cortex-R52 2. Cortex-R82
ARMv9	Processor cores: <ol style="list-style-type: none"> 1. Cortex-A715 Products: <ol style="list-style-type: none"> 1. MediaTek Dimensity 9200 2. Qualcomm Snapdragon 8 Gen 2 		

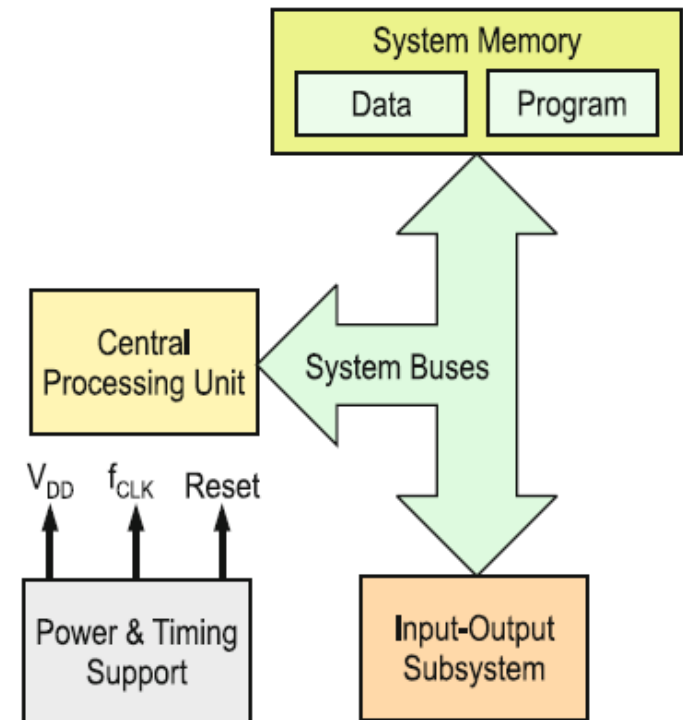
https://en.wikipedia.org/wiki/Template:Embedded_ARM-based_chips

The Organisation of Microprocessor Systems

Some basic terms and knowledge

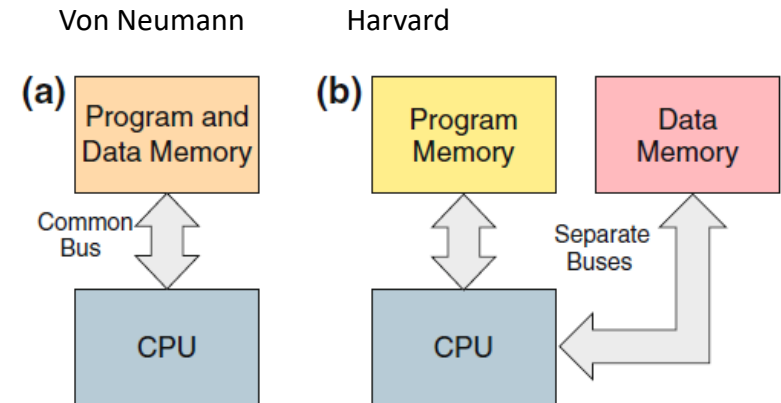
Microprocessor

- The minimal set of components required to establish a computing system
 - A Central Processing Unit (CPU)
 - System memory
 - Input/output (I/O) interface
- These components are interconnected by multiple sets of lines grouped according to their functions called **system buses**



Von Neumann and Harvard Architectures

- ❑ Program and data memories may or may not share the same system buses
- ❑ Systems with a single set of buses for accessing both programs and data are said to have a Von Neumann architecture
- ❑ Harvard architecture has physically separate address spaces for programs and data

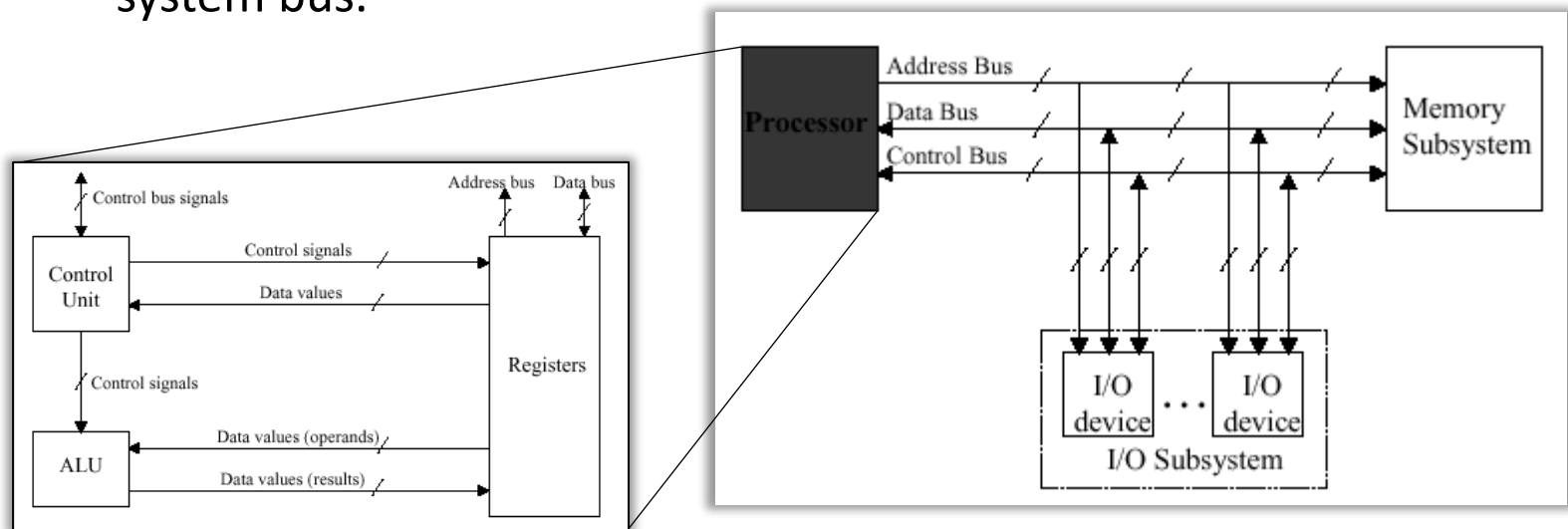


- MSP430 series uses a Von Neumann architecture
- Microchip PIC and Intel's 8051 utilize Harvard architecture

All ARM processors (before ARM9) use a Von Neumann architecture

Von Neumann machine

- Below is the typical organization of a modern Von Neumann machine.
 - Data and program are mostly stored in the computer memory separate from the process.
 - Registers in the processor [datapath](#) can also store small amount of data.
 - The address bus, data bus and control bus are also referred to as system bus.



System Buses

- A bus is simply a group of lines that perform a similar function
- Each line carries a bit of information and the group of bits may be interpreted as a whole
- The system buses are grouped in three classes:
 - Address bus
 - Data bus
 - Control bus

Address Bus

- The CPU interacts with only one memory register or peripheral device at a time
- Each register is uniquely identified with an identifier called address
- The set of lines transporting this address information is the address bus
- These lines are usually unidirectional and coming out from the CPU
- Addresses are usually named in hexadecimal notation
- The width of the address bus determines the size of the largest memory space that the CPU can address
- An address bus of m bits will be able to address at most 2^m different memory locations

Example

(Address Bus)

Determine how many different memory locations can be accessed and the address range (i.e., initial and final addresses) in hex notation with an address bus of (a) 12 bits, and (b) 22 bits

For 12 bits, there are $2^{12} = 2^2 \times 2^{10} = 4 \text{ K}$ different locations that can be addressed. In binary terms: 0b0000 0000 0000 to 0b1111 1111 1111, which in hex notation become 0x000 to 0xFFF

For 22 bits there are $2^{22} = 2^2 \times 2^{20} = 4\text{M}$ locations. So the address range is 0x000000 to 0x3FFFFFF

Data Bus

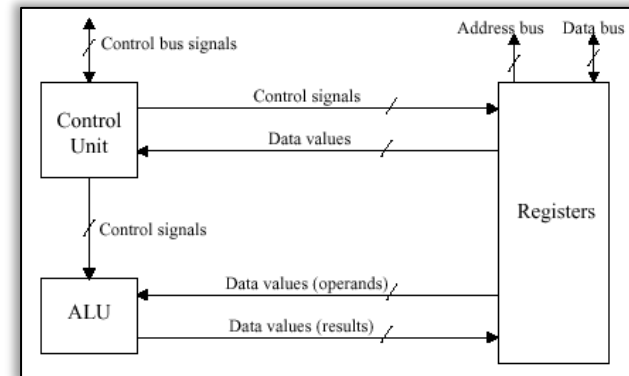
- The set of lines carrying data and instructions to or from the CPU is called the data bus
- A read operation occurs when information is being transferred into the CPU
- A data bus transfer out from the CPU into memory or into a peripheral device, is called a write operation
- The number of lines in the data bus determines the maximum data width the CPU can handle in a single transaction
 - An 8-bit data bus can transfer at most one byte in a single transaction
 - A 16-bit transaction would require two data bus transactions

Control Bus

- The control bus groups all the lines carrying the signals that regulate the system activity
- Most control lines are unidirectional
- Control signals include
 - those used to indicate whether the CPU is performing a read or write access
 - those that synchronize transfers saying when the transaction begins and ends
 - etc.

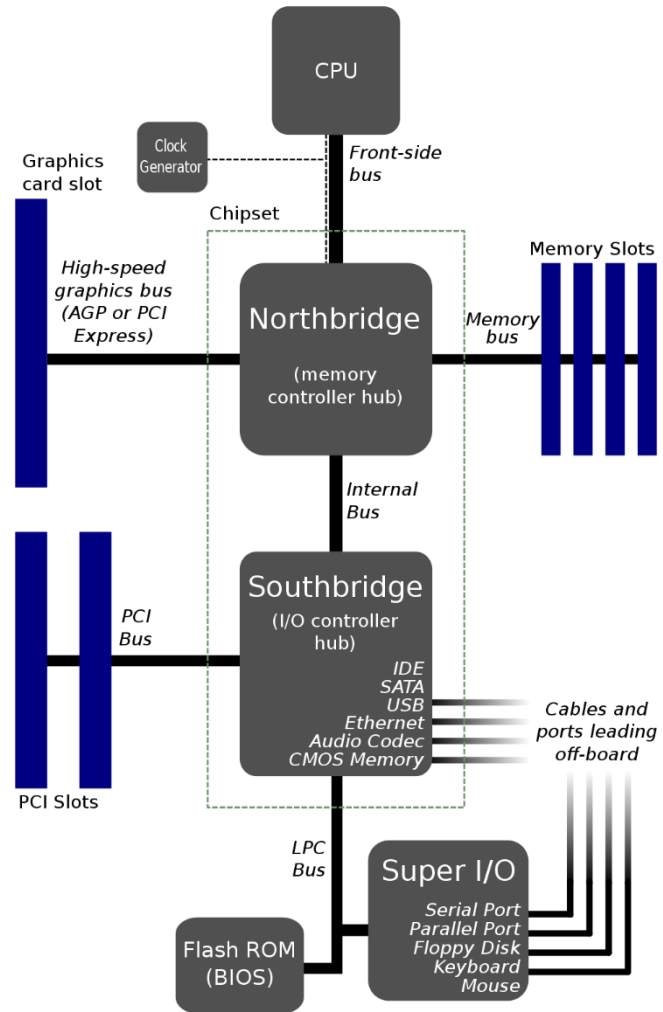
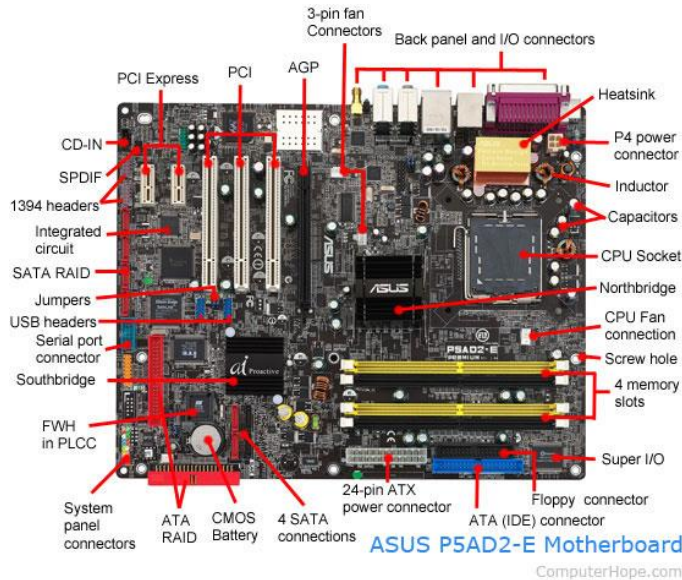
CPU

- **Registers:** Very fast but limited memory space embedded inside the CPU.
- **Arithmetic logic unit (ALU):** Carries out arithmetic and bitwise operations.
 - Arithmetic: + - * /
 - Bitwise: & | ^ ~
- **Control unit:** Coordinator of memory, ALU, registers, I/O devices.
- **CPU example workflow:**
 1. Fetch the instruction
 2. Fetch the operands
 3. Execute the instruction
 4. Write the results.
 5. Back to 1.



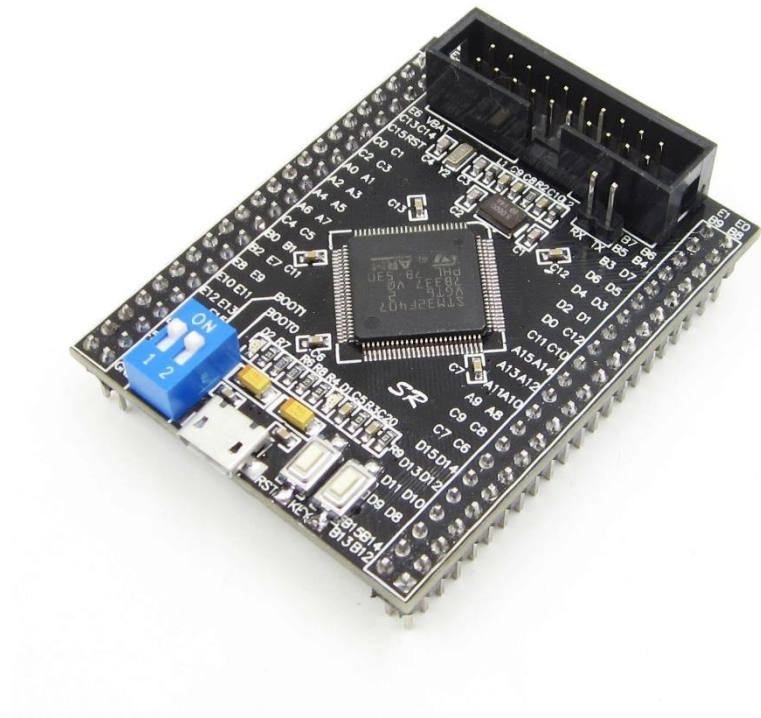
Internal Organisation of PC

- The image on the right shows the structure of a PC we see in daily life.
 - You can install operating systems on it.



MCU Board

- The image on the right is an ARM MCU board
 - **MCU**: Microcontroller Unit
 - It only runs one single program.
- You cannot install full operating systems on MCU.
 - Lacks memory management unit (MMU).
 - Some details will be covered later in this module.

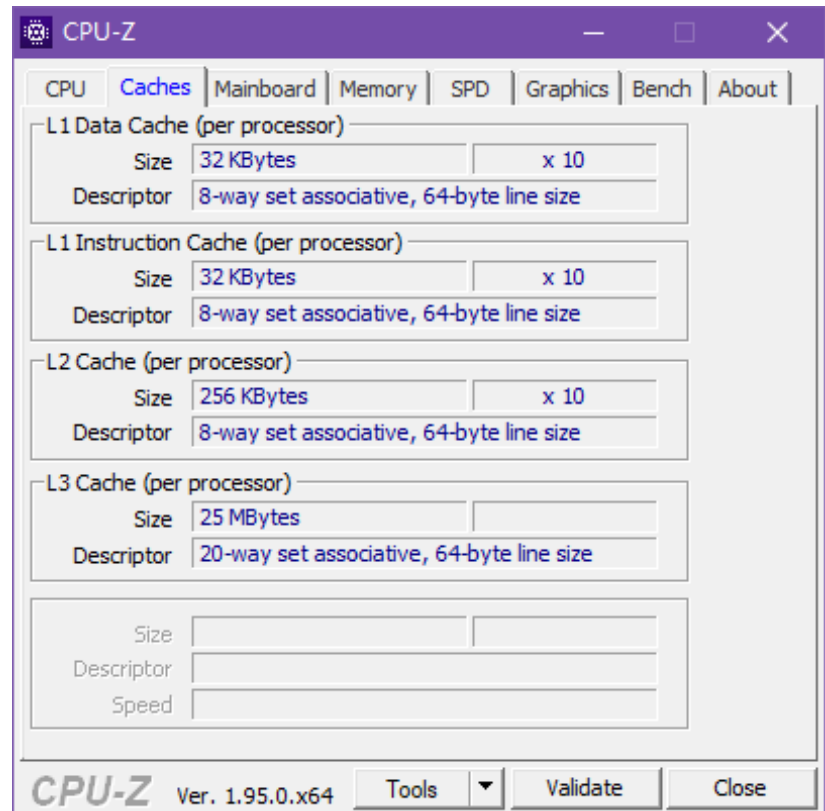
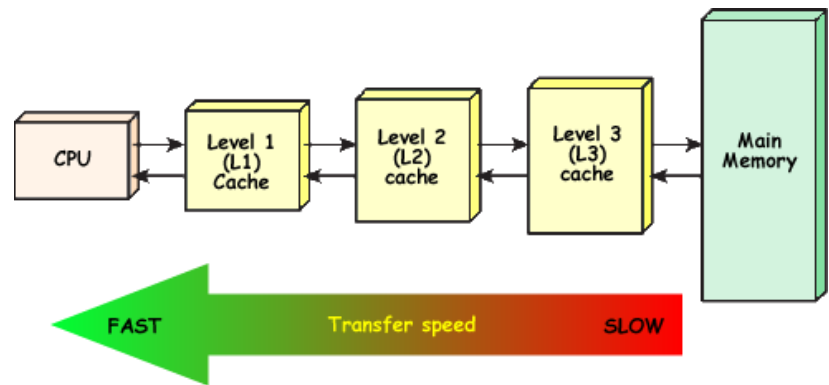


Von Neumann bottleneck

- In modern computers the bus connecting the CPU to external memory modules cannot keep up with the execution speed of the CPU.
- The slowdown of the bus is called the *von Neumann bottleneck*.
- Almost all modern CPU chips include some cache memory, which is connected to the other CPU components with much faster internal buses.
- The cache memory closest to the CPU commonly has a Harvard architecture configuration to achieve higher throughput of data processing.

CPU Cache

- Another Component of CPU.
- Lies in between CPU registers and memory.
 - Access speed: Register > cache > memory
 - Manufacture cost: Register > cache > memory
 - Storage capacity: Memory > cache > register
- Before looking for a certain piece of data in memory, CPU will first look for it in the cache.



Memory



- Data and Program instructions are stored in the **primary memory**.
 - A byte is a memory unit for storage
 - A memory chip is full of such bytes.
 - **Secondary memory** refers to external storages
 - CD, Floppy, Hard disk, USB storage etc.
- Technology limitations affects memory size.
- For 32-bit CPUs
 - Address bus is 32-bit wide.
 - Each memory location refers to 1 Byte (8 bits)
 - Can address up to $2^{32} = 4294967296 = 4 \text{ GB}$
 - The lowest address is 0×00000000
 - The highest address is $0 \times \text{FFFFFFFF}$

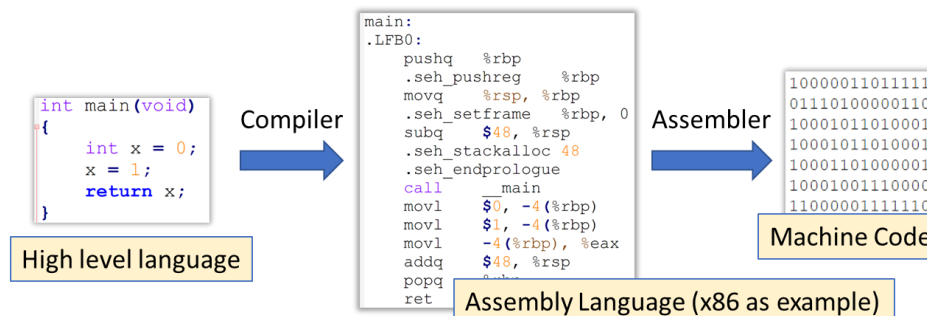
Address	Content
0xFFFFFFFF	0x8b
0xFFFFFFF	0x11
.	.
.	.
.	.
.	.
0x00000001	0xAA
0x00000000	0xCD

Accessing Memory

- **Data** and **instructions** are accessed and manipulated in fixed-length chunks.
- **Data sizes in C**: 1 byte (char), 2 bytes (short), 4 bytes (int, float), 8 bytes (long)
 - For data that is less than the size of its `sizeof (type)`, leading zeros/ones will be added.
 - A `char` with value 1 is stored as 00000001 in the memory.
 - A `char` with value -12 is stored as 11110100.
 - A `short` with value -12 is 1111111111110100.
 - Negative binary numbers will be taught later.
- **Data sizes in ARMv7 assembly**: byte, half-word (2 bytes), word (4 bytes)
 - No data type associated.
 - No sign/unsigned number indicator.

Accessing Memory

- Data and instructions are accessed and manipulated in fixed-length chunks.
- Instruction: each instruction (usually) has a length of one **word**.
 - Words in a 32-bit system are usually 32 bits long.
 - Words in a 64-bit system are usually 64 bits long.
 - Somehow related to the width of the data and address bus.
 - BUT, **some ARM instructions are half-word long**.
 - In x86, the length of instructions may vary.
 - Usually, word size = address size (pointer size) = register size.



Byte Ordering

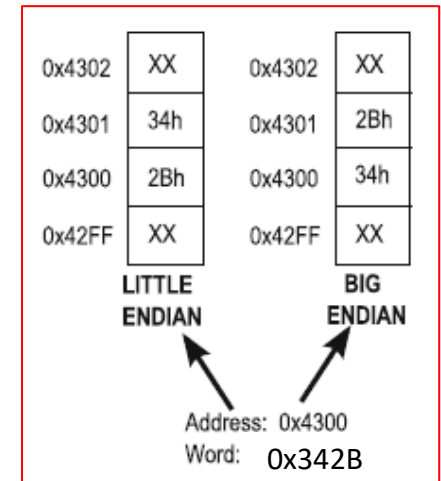
- Assume we have a 32-bit int value:
 - 0x AA BB CC DD
- How should its bytes be ordered in memory?
 - Arrangement 1

Value	DD	CC	BB	AA
Address	0x1	0x2	0x3	0x4

- Arrangement 2

Value	AA	BB	CC	DD
Address	0x1	0x2	0x3	0x4

- The ordering of bytes is called **endianness**.
 - The first arrangement method is called **little endian**.
 - The second arrangement method is called **big endian**.



Byte Ordering

- Big-endian architectures:
 - SPARC, z/Architecture
 - Least significant byte has highest address.
 - DD at 0x04
- Little-endian architectures:
 - x86, x86-64
 - Least significant byte has lowest address.
 - DD at 0x01
- Both endianness are supported:
 - ARM, MIPS, PowerPC
 - Can be specified



Important Notes about Byte Ordering

- Byte ordering only affects basic data units (word), composite data units like arrays or struct/class are NOT affected by byte ordering.

```
int y[] = {0x41424344, 0x45464748};
```

- Big-endian

Value	0x41	0x42	0x43	0x44	0x45	0x46	0x47	0x48
Address	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08

y[0]

y[1]

- Little-endian

Value	0x44	0x43	0x42	0x41	0x48	0x47	0x46	0x45
Address	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08

Alignment

- Instructions that read/write data from/to memory views the memory as a series of chunks of fixed-sized data.
- Each chunk has:
 - The address (of the first byte) of that chunk of data.
 - The length of data.
- If an instruction loads 4 bytes in one go, then the address of the 4-byte data must be a multiply of 4.
- This is called memory **alignment**.
 - Aligned vs unaligned access.

