Computer systems

course "Essentials of computing systems"

Feb.2022

©João M. Fernandes

contents

contents

1. Levels of a computer

2. Organization of a computer

3. Instruction-level parallelism

computer

- A computer is a device that can do work for its users by executing instructions given to it.
- A computing device is considered as a computer if it has the following characteristics:
  - **Data-dependent instruction sequence**. A computer must include conditional branch constructors, whose execution depends on values that can only be calculated during execution time.
  - **Data-dependent data selection**. A computer must be able to use data to determine the actual data to be used in calculations.
- It is a calculator if it lacks either or both characteristics.

program

- The sequence of instructions that describe the task to be performed is the program.
- The electronic circuits of each computer are able to directly execute a limited set of simple instructions, into which all its programs must be converted before they can be executed.
- These simple instructions are operations like:
  - add two numbers;
  - test if a number is equal to zero;
  - copy a piece of data from one part to another;
  - decide what instruction to execute next based on the evaluation of some logical condition.
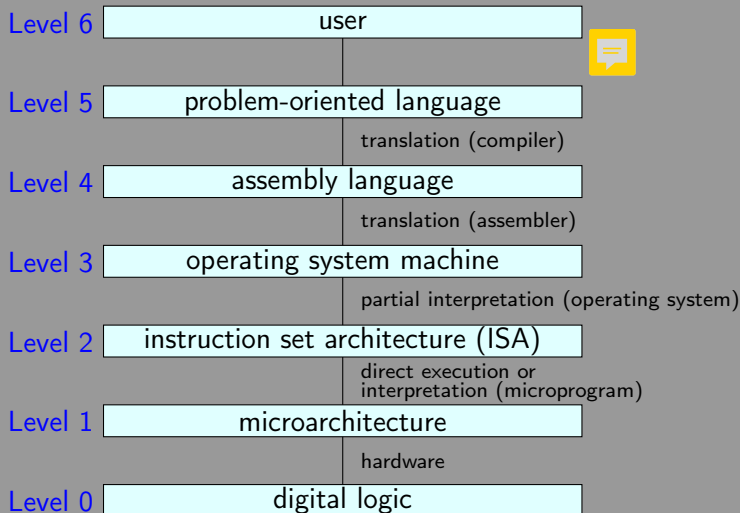
machine language

- The primitive instructions constitute a language, called a machine language.

- Programmers use these instructions to operate the computer.

- Programming with machine languages is usually difficult and tedious for people.

- The machine language exists mainly to be executed by the computer.

- New languages can be constructed on top of the machine language to rise the abstraction level.

a seven-level computer

Level 6	user
Level 5	problem-oriented language
	translation (compiler)
Level 4	assembly language
	translation (assembler)
Level 3	operating system machine
	partial interpretation (operating system)
Level 2	instruction set architecture (ISA)
	direct execution or interpretation (microprogram)
Level 1	microarchitecture
	hardware
Level 0	digital logic

Levels 4 and 3

- A fundamental frontier exists between levels 4 and 3.
- The lowest three levels are intended for running the interpreters and translators needed to support the highest levels.
- These interpreters/translators are developed by systems programmers (specialised in constructing virtual machines).
- Levels 4 and above are intended for the applications programmer with a given problem to solve in a specific domain.
- The machine languages of levels 1, 2, and 3 are numeric.
- Programs in these levels consist of long series of numbers (bits), which are appropriate for machines.
- At level 4 and above, the languages contain words and mnemonics that humans are able to easily understand.

programmer's perspective

- Each level represents a distinct abstraction of the computer, with different concepts, objects and operations.
- This division is convenient as it permits those that work at a given level to be unaware of what happens in the other levels.
- From a programmer's perspective, it is relevant to understand how computers work at the lower levels.
- Whenever a program written in a high-level language is not behaving as expected, it is seldom necessary to analyse what happens at the lower levels.

contents

von Neumann architecture

- A computer is a complex system that contains millions of electronic components.
- To describe a computer, it is essential to follow an hierarchical approach.
- Almost all uniprocessor (or scalar) computers follow the so-called von Neumann architecture.
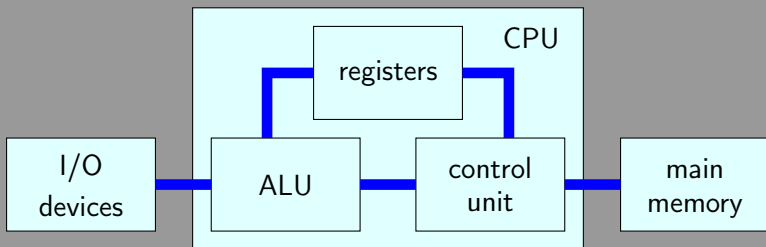
von Neumann architecture

- The von Neumann architecture has been extended in several ways to address some of its limitations.
- A well-known proposal is the inclusion of a system bus that connects the components of the computer.
- The system bus has three main elements: data bus, address bus, and control bus.
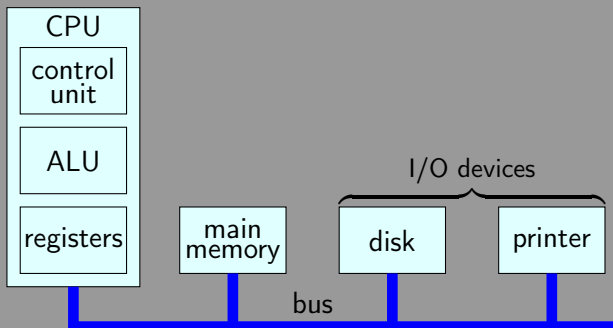
central processing unit (CPU)

- The CPU is the "brain" of the computer.
- The CPU is responsible for the execution of the program, stored as a sequence of machine language instructions in the main memory.
- Each instruction directs the CPU to perform some basic task.
- The CPU does all this mechanically, without understanding what is the purpose for executing the instruction.
- The CPU is composed of several distinct components: control unit, ALU, and registers.

CPU
control unit
ALU
registers
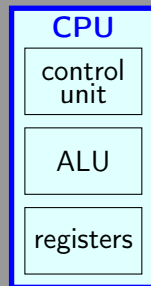
registers

- The CPU contains a small number of high-speed memory registers.
- A register is used to store temporary results and status information.
- All the registers have usually the same size.
- Registers can be read and written at a very high speed, since they are internal to the CPU.
- The instruction pointer (IP) indicates the address of the next instruction to be executed.
- The instruction register (IR) stores the instruction currently being executed. IR está no CU.

CPU

control unit

ALU
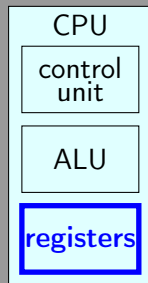
registers

arithmetic logic unit (ALU)

- The ALU performs basic operations such as addition, subtractions, and comparisons.
- Usually, an ALU has two data inputs and one data output.
- The operations carried on by the ALU affect the status registers, which indicate attributes related to the last arithmetic or logic operation.
- The operation that the ALU performs is signalised by the control unit, based on the instruction that is executed in each instant.

CPU
control unit
ALU
registers
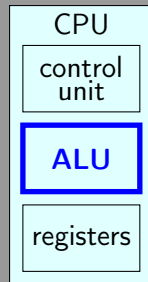
control unit

- The control unit is responsible for orchestrating the components of the computer.

- It ensures that the instruction that is being executed produces its expected effects.

- This is accomplished by activating in the right sequence the control signals that are sent to the computer components.

- This process is repeated indefinitely, while the computer is powered on.

- The CPU receives an instruction from the memory, decodes it, and executes it using data obtained from the memory or stored in the registers.

- When the execution of an instruction is finished, the cycle starts again for the next instruction.

CPU

control unit

ALU

registers
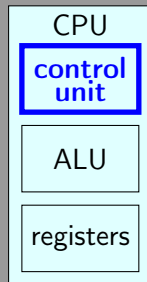
fetch-decode-execute cycle

- The control unit can be seen as a cyclic (micro)program that goes through the set of steps to execute the instructions of another program.
- The fetch-decode-execute cycle is a central to the operation of computers, since they all just execute instructions:
  1. Fetch the instruction from memory and store it into the IR.
  2. Determine the type of instruction.
  3. If a word in memory is used by the instruction, determine its location.
  4. Store the word into a CPU register, if needed.
  5. Produce the effects of the instruction, i.e., perform some simple operation dictated by the instruction.
  6. Change the IP to refer to the next instruction.

clock

- Most computer systems have a clock signal that acts like a heartbeat.
- The clock regulates the passage of time within the computer.
- A clock emits periodically a pulse.
- A clock cycle is the time between two consecutive ticks (represents one discrete time unit).
- The different operations performed by a processor are synchronised by the clock.
- All these operations begin with the pulse of the clock.
- The speed at which the instructions are executed depends on the frequency of the clock (cycles per second or Hertz).

main memory

- The main memory is the component of the computers, where programs and data are both stored.
- A memory consists of a number of cells or locations, each of which can store a piece of information.
- Each cell has a number, called its address, by which programs can refer to it.
- A memory with n cells uses addresses from 0 to $n-1$.
- All cells in a memory contain the same number of bits.
- The main memory is a random-access memory (RAM).
- RAM memories are volatile, which means that the information disappears if the supply voltage is turned off.

bytes and words

- The cell is the smallest addressable unit.
- Almost all modern computers use 8-bit (1 byte) cells.
- Bytes are grouped into words (a computer with a 32-bit word has 4 bytes/word).
- The word is also important since most instructions operate on entire words, for example, adding two words together.
- A 32-bit machine has 32-bit registers and instructions for manipulating 32-bit words.

little- and big-endian

- When storing a multibyte value in a byte-addressable memory, the order of the individual bytes is relevant.
- The number $11\ 0000\ 0000_2$ requires two bytes of memory: $0000\ 0000_2$ and $0000\ 0011_2$.

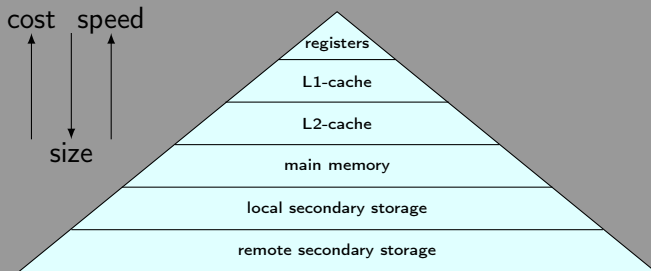|		
...	2103	...
00000000	2104	00000011
00000011	2105	00000000
...	2106	...

Little-endian   Big-endian
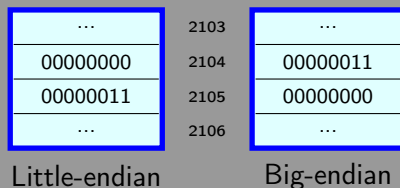
stored-program computer

- Digital computers possess a great versatility, as they can execute whatever programs their users wish.
- This flexibility is related to the ability of a computer to load the programs in its internal memory.
- By changing the program in the memory, one changes the problem the computer is solving.
- This justifies the notion of a stored-program computer, one that can be used for any problem.
- During a computation, a given instruction may also be modified, i.e., changed to a different instruction
- This gives rise to self-modifying code, which is straightforward to perform at the machine-level.

memory hierarchy

- The main memory of a computer is never enough.
- The traditional solution to store a great quantity of data is to use a memory hierarchy.

cost speed

size

| registers |
| L1-cache |
| L2-cache |
| main memory |
| local secondary storage |
| remote secondary storage |

- The aim is to establish a system with cost almost as low as the cheapest level and speed almost as fast as the fastest level.
- All data in a given level can be found in the next lower level.

memory hierarchy

level	1	2	3	4
name	registers	cache	main memory	disk storage
typical size	< 1 kB	< 16 MB	< 512 GB	> 1 TB
implementation	CMOS	SRAM	DRAM	magnetic disk
access time (ns)	0.25–0.5	0.5–25	50–250	5,000,000
bandwidth (MB/s)	50,000–500,000	5000–20,000	2500–10,000	50–500

Adapted from [Hennessy and Patterson; 2007]

I/O devices

- Computers are built so that they can be expanded with new I/O (input/output) devices: hard disk, keyboard, mouse, monitor, printer, network interface, scanner.

- An I/O device is a hardware system used by humans or systems to communicate with a computer.

- The CPU must communicate with and control these devices.

- For each device, there is a device driver, which is software that the CPU executes when it has to deal with that device.

- A new device can be installed by connecting the device into the computer and installing its device driver software.

- The device driver allows the CPU to communicate with the respective device.
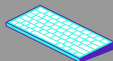
polling

- When a given device produces data that needs to be processed, somehow the CPU needs to take some action.
- With polling, the CPU keeps checking for incoming data over and over.
- Whenever it detects that data was made available by the device, it processes it.
- Polling is very simple, but is also very inefficient.

interrupts

- Interrupts are generally used instead of polling.
- An interrupt is a signal sent by a device to the CPU to request its attention.
- The CPU reacts by suspending what it is doing to answer to the interrupt.
- The CPU saves information about what it is currently doing so that it can return to it afterwards.
- Then the CPU jumps to some fixed memory location and begins executing the instructions stored there.
- Those instructions correspond to the interrupt handler that does the processing necessary to respond to the interrupt.
- Once the interrupt is processed, the CPU returns to the process that was suspended.

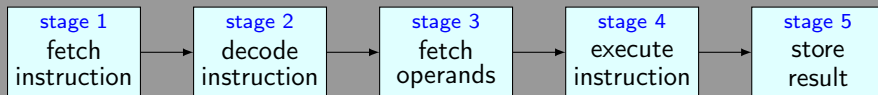# contents

improving computers

- Improving the characteristics of computers, like performance, cost, reliability, energy consumption, is a concern for computer engineers and architects.

- Performance is an inescapable issue, since making computers run faster is always relevant.

- One possible approach is to exploit parallelism, which can have different forms: instruction-, data-, and processor-level.

- Instruction-level parallelism with pipelining explores individual instructions to obtain higher throughput.

pipeline analogy

Patterson and Hennessy, Computer organization and design, 5th ed., Morgan Kaufmann, 2014.
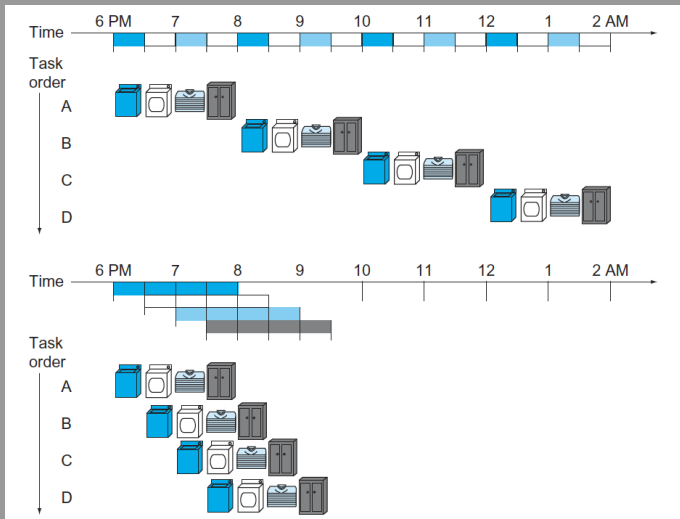
pipeline

- With pipelining, the process of executing instructions is divided in stages.
- Multiple instructions can be overlapped in execution.
- Each stage is responsible for a part of the process and all can run in parallel, thus speeding up the process.

stage 1 fetch instruction	stage 2 decode instruction	stage 3 fetch operands	stage 4 execute instruction	stage 5 store result

ideal operation of the pipeline

- Stages operate in parallel, which accelerates the process.
- Pipelining increases the number of instructions that are simultaneously under execution.
- Instruction throughput (number of instructions completed per time unit) is improved.
- Latency (time that is required to complete each individual instruction) is maintained.
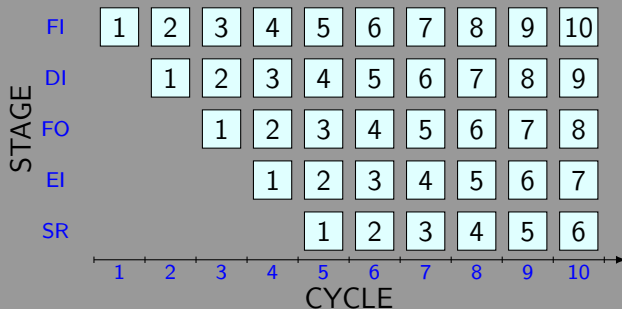
aspects affecting pipeline

- The maximum theoretical speed up is very unlikely to occur in practice, because it is difficult to balance the time it takes to complete each stage.
- All stages must be equally balanced, otherwise the faster stages must wait for the slower ones.
- Three major aspects negatively affect the theoretical speed up:
  1. resource conflicts,
  2. data dependencies,
  3. conditional branch statements.

Resource conflicts

- A resource conflict occurs whenever two pipeline stages need to simultaneously access the same resource (e.g., memory).
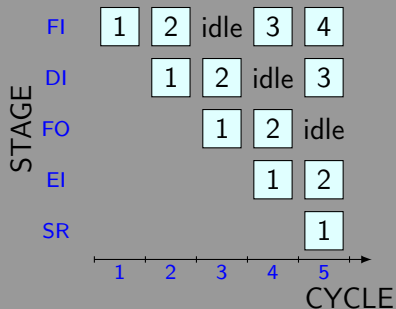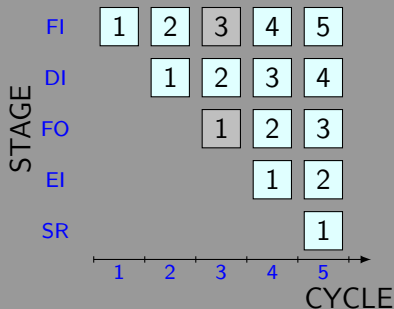
data dependency

- A data dependency occurs whenever the result of one instruction, not yet available, is also the operand of a subsequent instruction.

- Some solutions exist to handle these conflicts.

- By inserting a delay (e.g., a nop instruction) into the pipeline, sufficient time passes and the conflict disappears.

- Some architectures assume that this problem is solved by the compiler, which must reorder instructions.

branch instruction

- Problems in the pipeline may also be caused by a branch instruction that modifies the normal flow of a program.

- If the conditional branch is taken, the subsequent instructions must not be executed and the pipeline must be emptied.

- Some architectures predict the outcome of a conditional branch to try to identify the instructions that will be executed next.

- In some cases, compilers try to solve branching issues by rearranging the machine code to cause a delayed branch.

- Another alternative is to initiate fetches on both paths of the branch. When the branch is actually executed, the correct path is identified and the pipeline can be safely continued.