COMP 10280
Programming I (Conversion)

John Dunnion

School of Computer Science
University College Dublin

COMP 10280 Programming I (Conversion)/Lecture 2

Outline

Decimal Numbers

- Numbers that we use are made up of decimal digits
- Each decimal digit can have a value 0–9
- Combining digits, we can form larger numbers by representing them as integers in base 10
- $215_{10} = (5 \times 10^0) + (1 \times 10^1) + (2 \times 10^2) =$
  $(5 \times 1) + (1 \times 10) + (2 \times 100) =$
  $5 + 10 + 200 =$
  215

Binary Numbers

- Numbers in a computer are made up of binary digits or "bits"
- Each binary digit can have one of two values: 0 or 1
- Combining binary digits (bits), we can form larger numbers by representing them as integers in base 2
- $11010111_2 =$
  $(1 \times 2^0) + (1 \times 2^1) + (1 \times 2^2) + (0 \times 2^3) + (1 \times 2^4) + (0 \times 2^5) + (1 \times 2^6) + (1 \times 2^7) =$
  $(1 \times 1) + (1 \times 2) + (1 \times 4) + (0 \times 8) + (1 \times 16) + (0 \times 32) + (1 \times 64) + (1 \times 128) =$
  $1 + 2 + 4 + 0 + 16 + 0 + 64 + 128 =$
  215

Binary Numbers

- Binary is the natural number base for computers
- For larger units of data, we name numbers according to the “binary thousand”, $2^{10} = 1024$
- 1 KiloByte (KByte or KB) = 2^{10}bytes $=$ 1024 bytes
- 1 MegaByte (MByte or MB) = 2^{20} bytes $=$ 1 048 576 bytes
- 1 GigaByte (GByte or GB) = 2^{30} bytes $=$ 1 073 741 824 bytes
- But see later!
- It is often convenient to represent numbers in base 8 (octal) or base 16 (hexadecimal)
- Base 8: Digits 0–7; Base 16: Digits 0–9, A–F
- $215_{10} = 327_8 = D7_{16} = 11010111_2$

Von Neumann Architecture

- Central Processing Unit
  - Arithmetic and Logic Unit
  - Registers
- Control Unit
  - Instruction Registers
  - Program Counter
- Primary Memory
  - Instructions
  - Data
- Secondary Memory
- Input and Output Mechanisms

Central Processing Unit (CPU)

- Often just referred to as the *Processor* or *Core*
- The “brain” of the computer
- Reads instructions from memory and executes them
- Operates at *very* high speeds
- Speed measured in Hertz
- A 2.8 GigaHertz (2.8GHz) computer operates at 2.8 billion (2 800 000 000) clock cycles per second
- The clock cycle is a switch between a high voltage and a low voltage
- Usually one simple instruction per clock cycle
- Many modern computers have more than one CPU (“Multi-Core”, “Dual-Core”, “Quad-Core”)
- Intel i7 Quad-Core 2.8GHz Processor

Components of the CPU

- Arithmetic and Logic Unit (ALU)
  - Arithmetic and logic circuits that execute the operations of the CPU's instruction set
- Registers
  - Set of registers that hold the *operands* for all ALU *operations*
- Control Unit
  - Logic that sequences each step in the execution of an instruction
- Bus Interface Unit
  - Set of registers at the interface between the CPU and the System Bus

System Memory

- System Memory holds all of the *active* data that the computer is using
- Random Access Memory (“RAM”)
- Volatile
- Initially RAM is empty
- Each program or data file uses part of memory
- When a program finishes, the memory it used is freed for other uses
- Memory measured in MBytes/GBytes
- 8GBytes memory
- Also have Read-Only Memory (“ROM”)

Random Access Memory (RAM)

- Ordered sequence of locations in which to store data
- Each location has a unique address
- One byte of data can be stored at each location
- In an 8-bit computer architecture, the address is 8 bits long. This means that there are $2^8 = 256$ locations that can be addressed directly
- In a 32-bit computer architecture, there are 2^{32} = 4GBytes of memory that can be addressed directly
- Most modern computers use a 64-bit architecture
- A program's code and the data it needs are loaded into memory while the program is executing
- Most modern computers have a hierarchical memory: one or more memory caches

System Bus

- A *bus* is a *channel* (eg a set of wires) over which data flows between two components
- Most of the internal system components, including the CPU, memory, storage devices, etc communicate with each other over one or more buses
    - Memory bus: Connects the CPU and the memory
    - Input/Output bus: Connects performance-critical components (eg video cards, disk drives, etc) to the CPU and the memory
- Performance of a bus is measured by the *bandwidth*: total amount of data that can be transferred per second
    - The width of the bus: the number of bits that can flow simultaneously over the bus (eg 8-bit, 32-bit)
    - The speed of the bus: the amount of data that can be transferred per second (eg 66MHz, 800MHz)
    - *Bandwidth* = Width $\times$ Speed (eg 8Gbits/second)

Hard Disk

- Long-term memory
- Non-volatile
- Storage for Operating System, programs/applications and data
- Organised into file systems, directories (“folders”) and files
- Electro-mechanical Magnetic media: Disk(s)
- Solid-State Drive: Flash memory

MegaHertz and MegaBytes

- Hertz: Measure of frequency (Cycles per second)
- 1 KiloHertz = 1 000 Hertz = 1 000 cycles/second
- 1 MegaHertz = 1 000 000 Hertz = 1 000 000 cycles/second

- Byte: Measure of amount of information (8 bits)
- 1 KiloByte (KByte, KB) = 2^{10} Bytes = 1024 Bytes
- 1 MegaByte (MByte, MB) = 2^{20} Bytes = 1024 KBytes = 1 048 576 Bytes
- 1 GigaByte (GByte, GB) = 2^{30} Bytes = 1024 MBytes = 1 073 741 824 Bytes
- 1 TeraByte (TByte, TB) = 2^{40} Bytes
- 1 PetaByte (PByte, PB) = 2^{50} Bytes
- 1 ExaByte (EByte, EB) = 2^{60} Bytes
- 1 ZettaByte (ZByte, ZB) = 2^{70} Bytes
- 1 YottaByte (YByte, YB) = 2^{80} Bytes

Decimal v Binary

Decimal			Binary		
1000	KiloByte	KB	1024	KibiByte	KiB
1000^2	MegaByte	MB	1024^2	MebiByte	MiB
1000^3	GigaByte	GB	1024^3	GibiByte	GiB
1000^4	TeraByte	TB	1024^4	TebiByte	TiB
1000^5	PetaByte	PB	1024^5	PebiByte	PiB
1000^6	ExaByte	EB	1024^6	ExbiByte	EiB
1000^7	ZettaByte	ZB	1024^7	ZebiByte	ZiB
1000^8	YottaByte	YB	1024^8	YobiByte	YiB

Simple model of hardware

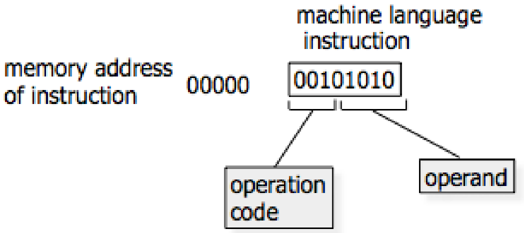
- The hardware of modern computers is very complex
- Programmers can ignore the low-level details of the computer's hardware and write programs using a much simpler model ("abstraction")
- However, sometimes it is necessary to know what is happening at the lower level of detail, especially when a program has to run as efficiently or as fast as possible

Systems Software

- The Operating System is the single most important program on the computer
- the Operating System has two main rôles:
  - It acts as an interface between the user of the computer and the computer's hardware and software
  - It facilitates the efficient use of the computer's resources
- The Operating System is a program, like any other
- It runs continuously from boot-up and controls the running of all other programs and applications
- Common Operating Systems include Unix, Linux, Microsoft Windows, Apple OS X, Apple iOS, Android.

Machine Code

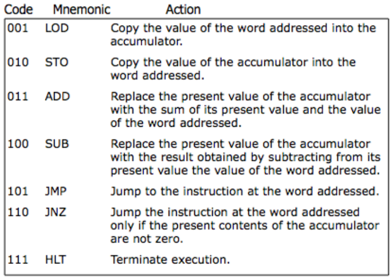
- A CPU can only understand program instructions that are expressed in terms of its machine language or machine instruction set (Machine code)
- To illustrate the concept of a machine language program, consider a simple CPU with a primitive machine language



- In this example, the operation (command to the control unit) is stored in the first three bits of the instruction, and the last five bits contain the operand, either the address of the data to be operated upon by the instruction or an address to which control is to be transferred

Assembly Language

- In Assembly Language, we replace the numeric command with a mnemonic
- We also replace numeric references to registers by labels



Code	Mnemonic	Action
001	LOD	Copy the value of the word addressed into the accumulator.
010	STO	Copy the value of the accumulator into the word addressed.
011	ADD	Replace the present value of the accumulator with the sum of its present value and the value of the word addressed.
100	SUB	Replace the present value of the accumulator with the result obtained by subtracting from its present value the value of the word addressed.
101	JMP	Jump to the instruction at the word addressed.
110	JNZ	Jump the instruction at the word addressed only if the present contents of the accumulator are not zero.
111	HLT	Terminate execution.

- The assembly language instructions directly reference components of the hardware, such as the CPU's registers

Machine/Assembly Language Programming

- To write a program of any size in machine language or assembly language is extremely difficult
- The instruction set is low-level, ie the model, or abstraction, is very close to the details of the computer hardware
- It is difficult to determine the purpose of a program by examining the code
- The instruction set is specific to a particular type of computer architecture. Only machines which have the same architecture can run the program. A completely new program must be written for any new architectures
- In order to make software development cost-effective it is necessary to have some way of writing programs which
    - can be easily run on different architectures
    - are easy to understand, modify and maintain

High-Level Language Programming

- Hence the need for high-level languages such as Python that are written for an abstraction that is close to the problem representation and that can be translated to run on many different types of computer architecture
- There are many other languages, such as C, C++, Java, Perl, Fortran, Lisp. We could roughly attempt to organise these languages from highest-level to lowest-level, according to how close the language is to a problem-oriented abstraction or to a hardware-oriented abstraction.
- Python allows us to create programs that are largely problem-oriented

Program Translation

- The program written by the programmer is called the source program
- A source program written in a high-level language must be converted, or translated, into machine/assembly code in order to run on the computer
- This process is called compilation and we say that a program is compiled into machine/assembly code. The program that does this is called the compiler
- The computer then executes (“runs”) the machine/assembly language translation of the source program
- Some languages are interpreted, ie the source code is executed directly by an interpreter
- Python is an interpreted language