UECS1013 Introduction to Computer Organisation and Architecture

2014/May/Lecture 1

INTRODUCTION

- **History of Computers**
- **What is a Computer?**
  – Multi-level Computer Units
  – Types of Computer
- **Functional Units**
  – Memory
  – Input and Output Unit
  – CPU
  – Bus
- **Basic Operational Concepts**

What is a Computer?

Type	Price ($)	Example application
Disposable computer	1	Greeting cards
Embedded computer	10	Watches, cars, appliances
Game computer	100	Home video games
Personal computer	1K	Desktop or portable computer
Server	10K	Network server
Collection of Workstations	100K	Departmental minisupercomputer
Mainframe	1M	Batch data processing in a bank
Supercomputer	10M	Long range weather prediction

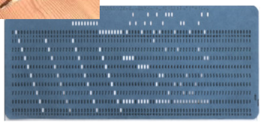
In general, a computer is a machine that can solve problems for people by carrying out instructions given to it.

3

History of Computers

Mechanical Computers

- Abacus invented in Babylonia in 3000BC
- Adding machine by Blaise Pascal (1642)
- IBM first electromechanical computer (using relays) designed by Howard Aiken (1937) was based on *punched cards*. It was used to calculate tables of mathematical functions.
- Holes on punched card were mechanically sensed and used to control the automatic sequencing of a list of calculations



4

History of Computers

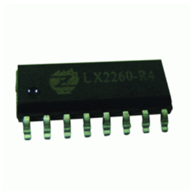
- ***1st Generation Computers*** (1940s to early 1950s) – based on *vacuum tube*s technology.
  - 1943 – ENIAC: first fully electronic computer, designed by John Mauchly (100 to 1000 times faster than mechanical computers)
  - 1946 – EDVAC: first **stored program** computers, designed by John von Neumann
  - Magnetic core memories were developed
  - Magnetic *tape* storage devices were developed
  - Programmes were written in assembly language

- ***2nd Generation Computers*** (late 50s to early 60s) – based on *transistors* (solid state) technology.
  - more reliable, less expensive, low heat dissipation
  - IBM 7000 series, DEC PDP-1.
  - Magnetic core memories were prominent.
  - Magnetic *disk* storage devices were developed
  - High-level languages such as Fortran were developed. Compilers were used to translate high-level languages to assembly languages

5

History of Computers

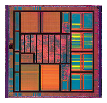
- ***3rd Generation Computers*** (late 60s to early 80s) – integrated circuits (IC).
  - Texas Instrument and Fairchild Semiconductor developed the capability to fabricate many transistor on a single silicon chip, called integrated circuits (IC)
  - IBM 360 series, DEC PDP-8
  - Many transistors packed into single container → low prices, high packing density
  - Introduction to ***pipelining*** (multiple instructions at the same time), ***cache*** (memory appear faster) and ***virtual memory*** (memory appears bigger)



6

History of Computers

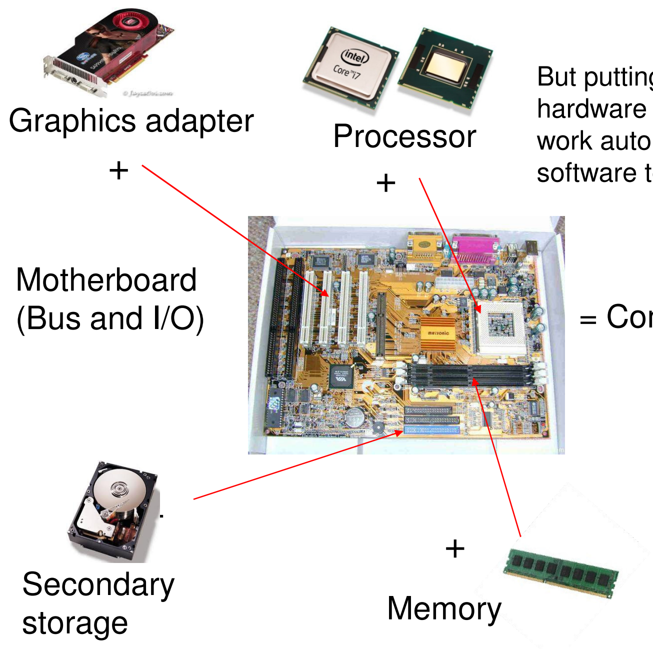
- ***4th Generation Computers*** - Very Large Scale Integration (VLSI)
  - Advances in processing/manufacturing technology → tens of thousands of transistors could be placed on a single chip → small size, low-cost, large memory, ultra-fast PCs to supercomputers
  - Driving force: Intel, National Semiconductor, Motorola, Texas Instrument and Advanced Micro Devices
  - Embedded computer systems and portable notebooks become widespread
  - Supercomputers and grid computers emerges as the upper end computing systems for weather forecasting, scientific and engineering computations and simulations



- ***5th Generation Computers*** (Parallel Computing)
  - Multi-core processors
  - Massively parallel, Cloud/Grid Computing, etc.

7

What is a Computer?



But putting together these hardware does not mean it will work automatically! You need software to manage them....

= Computer hardware

8

What is a Computer?

- Computer = Hardware + Software.
  - *Hardware*: physical components for computation/processing; should be simple, fast, reliable.
  - *Software*: set of instructions to perform tasks to specifications; should be flexible, user-friendly, sophisticated.
- Computers are information processors

$$\begin{tikzcd}\text{Raw data} \arrow[r] & \boxed{\text{Computer system}} \arrow[r] & \text{Processed information}\end{tikzcd}$$

Data Units:

- The hardware components of a computer system represent information as **0** (as 0V) or **1** (as either 1.8V or 3.3V or 5V).
- Therefore, data must be represented using binary digits consisting of only 0 and 1.

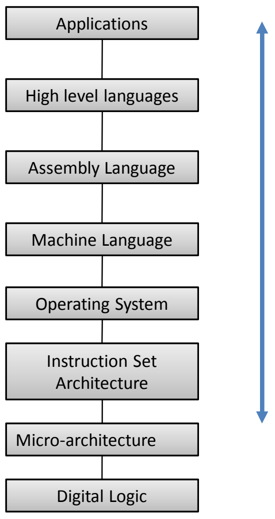
1 bit (binary digit): one of two values (0 or 1)

1 byte: 8-bits

1 word: 1, 2, or 4 bytes, or more (depends on ALU)

9

Multi-level Architecture



Software

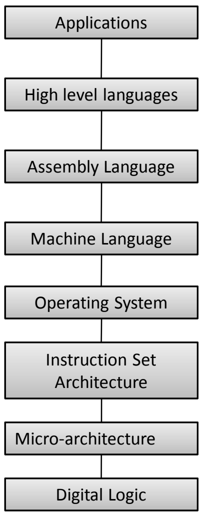
- High-level abstraction
- Closer to a human understanding of the problem
- Easier to program. Do not require much understanding of the computer hardware

- Low-level implementation
- Closer to how the machine solves the problem.
- Difficult to program. Requires in-depth knowledge of the computer hardware

Actual physical implementation

10

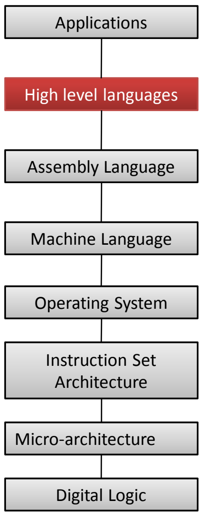
Multi-level Architecture



1. There is a large gap between what is convenient for people (the code can be understood and written easily by a human) and what is convenient for computers (code is in binary form).
2. **Abstraction** implies that any layer can hide the details of the layers below it from the layers above it.
3. For example, Instruction Set Architecture (ISA) is the abstraction layer between hardware and low level software layer.
4. Programmers are only interested in the top level but people interested in understanding how a computer really works must study all the levels.

11

High Level Languages

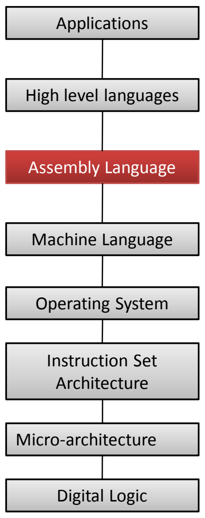


1. For a *software developer* to develop ***applications*** such as a library catalogue system, Office, Winzip, game, etc.
2. Examples of high level languages:

   C++, Matlab, Java, Java Script, etc.
3. High level languages are much easier for human to understand than assembly language program and are designed to be used by application programmers with problems to solve.
4. Example C++ code for C= A + B
   ```
   int A, B, C;    // variable declaration
   …
   C = A + B;      // add operation
   ```
5. Programs are generally translated to lower levels by ***compilers***.

12

Assembly Language Level



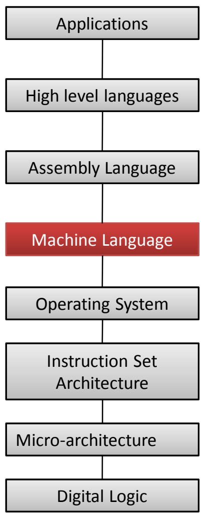
1. For *computer engineers* to develop the software to control the operations of a system, e.g., walkie talkie, routers, etc.
2. A symbolic form for the underlying language. The language is specific to the hardware. Example: ARM/MIPS
3. Assembly language (low level language) programs are usually easier to create than machine language program but still *requires an in-depth knowledge of the computer architecture* (hardware resources).
4. Example code for C = A + B

```
…
LDR R0, A          ; Load A into register R0
LDR R1, B          ; Load B into register R1
ADD R2, R1, R0     ; Add R0 and R1 and save result to R2
STR R0, C          ; Save the result  into C
```

5. An ***assembler*** translates programs written in an assembly language into a sequence of machine instructions.

13

Machine Language Level

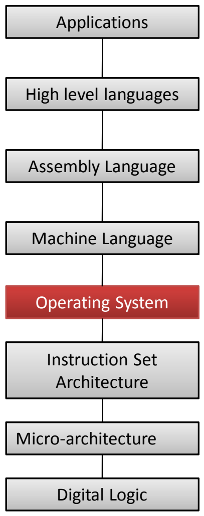


1. Native codes for the machine, consisting of strings of 1's and 0's and stored as binary numbers.
2. The very lowest possible level at which you can program a computer. Difficult for a human programmer to program but useful for debugging purposes.
3. Specific to the hardware: the machine language code assembled for Intel x86 system cannot run on a ARM platform
4. Example machine to add two variables (C=A+B)

C++	Assembly	Machine Language
int A, B, C;	…	…
…	LDR R1, A	E59F1010
C = A+B;	LDR R0, B	E59F0008
	ADD R2, R1, R0	E0812000
	STR R5, C	E58F5008

14

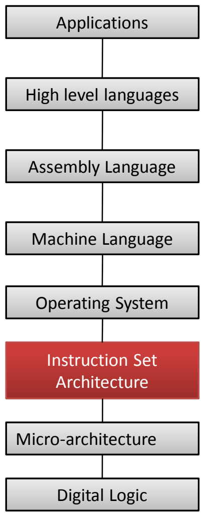
Operating System Level



1. A series of programs serves as a kind of protective shell or buffer to serve users and application programs from the complexities of the hardware.
2. Operating system is a resource manager:
   - Manages all resources
   - Decides between conflicting requests for efficient and fair resource usage (memory or CPU time)
   - Control shared resources so that different users are given fair share
3. Operating system is a control program
   - Controls execution of programs to prevent errors and improper use of the computer
   - Protect the system from incorrect or malicious program and users

15

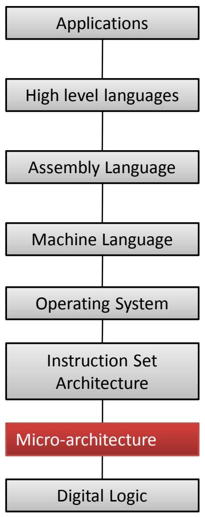
Instruction Set Level



1. The set of all ***instructions*** that is visible and available to a system-level programmer to control the operations of a PC.
2. The instructions are specific for a particular hardware. Example: The instruction set for ARM is different from MC68000 system.
3. The *abstraction layer* between the actual hardware and the low-level software.
4. May be carried out interpretively either by the microprogram or hardware at the micro-architecture level.
5. Example instructions:
   ADD Rx, Ry
   LDR, Rx, Ry

16

Microarchitecture Level

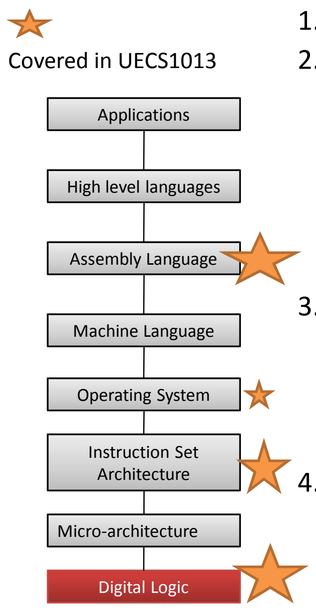


1. At this level, registers are connected to the ALU to form a **data path**, over which the data flow.
2. The operation of the data path (from registers to ALU) is controlled directly by a program called a **microprogram** or directly by hardware.
3. A microprogram is an interpreter for the instructions at ISA:

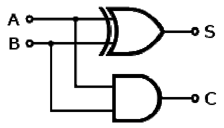
   e.g., ADD
   a. fetch instruction
   b. locate operands and bring into registers
   c. compute sum by the ALU

17

Digital Logic Level



1. The basic building block of a digital logic is ***gate***.
2. Example: How are the adder which adds two binary digits being implemented at the hardware (gate) level?



3. Each gate has one or more digital inputs (signals representing 0 or 1) and computes as output some simple functions of these inputs, e.g., AND or OR.
4. Gates can form the main computing engine. Gates can also form a 1-bit memory; some 1-bit memories can form registers for holding binary numbers.

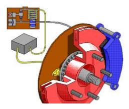
18

Computer Types

- **Embedded Computers**
  - Integrated into a larger system to automatically monitor and control a physical process .
  - Used for specific purpose rather than for general tasks.
  - Typical applications: industrial and home automation, appliances, telecommunication products and vehicles.
  - Examples:

Washing machines

Routers


ABS Braking System

19

Computer Types

- **Personal Computers**
  - Unlike *Embedded Computers*, personal computers are general-purpose – could be used to perform a variety of tasks
  - Used at different settings (homes, educational institution, business and engineering departments) and can be installed with a variety of application
- **Servers and Enterprise Systems**
  - Large computers that are shared by large number of *users*
  - Each *user* can be a personal computer over a public or private network.
  - Example: database & information processing for government

20

Computer Types

- **Supercomputer**
  - ❖ Used for highly demanding computations, e.g., weather forecasting, engineering design and simultation, scientific high costs.
  - ❖ Offering the highest performance
  - ❖ Most expensive and physically the largest

Year	Supercomputer	Peak speed (Rmax)	Location
2008	IBM Roadrunner	1.026 PFLOPS	Los Alamos, USA
		1.105 PFLOPS	
2009	Cray Jaguar	1.759 PFLOPS	Oak Ridge, USA
2010	Tianhe-IA	2.566 PFLOPS	Tianjin, China
2011	Fujitsu K computer	10.51 PFLOPS	Kobe, Japan
2012	IBM Sequoia	16.32 PFLOPS	Livermore, USA
2012	Cray Titan	17.59 PFLOPS	Oak Ridge, USA
2013	NUDT Tianhe-2	33.86 PFLOPS	Guangzhou, China

Source: Wikipedia

21

Computer Types

- **Grid Computing**
  - ❖ More cost effective than super computer
  - ❖ Links a large number of disparate personal computers and disk storage units in a physically distributed high-speed network (grid)
  - ❖ Grid computing relies on software (middleware) to divides/directs pieces of a program to a number of different computers.
  - ❖ Allows remote access to computing resources
  - ❖ Computing power is aggregated but with communication overhead
- **Cloud Computing**
  - ❖ Built upon the foundation of Grid Computing
  - ❖ Further offers on-demand resource provisioning. Allows companies immediate access to high computational power without having into developing new infrastructure by outsourcing

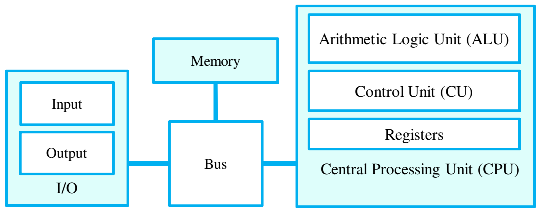
22

Computer Types

- **Cloud Computing (…cont)**
  - ❖ Providers can provide cloud services as 3 different services
  - a) Infrastucture as a Service (IaaS)
    - Cloud provider supplies the physical or virtual machines, raw storage, load balancers and network from their large pool of computing equipments installed in data centers.
    - Customer install their operating system images & software on the machines before deploying their application.
  - b) Platform as a Service (PaaS)
    - Cloud provider supplies the platform (operating system, programming language environment and web server)
    - Customer develops software solutions on the cloud platform (do not need to invest in infrastructure to start)
  - c) Software as a Service (SaaS)
    - Cloud provides and installs the application software
    - Customer to run the computational expensive application in the cloud network to enjoy the desired high performance

23

Functional Units



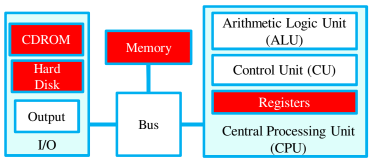
- CPU Components:
  - ❖ CPU: performs mathematical & logical operations, controls operations of a computer).
  - ❖ Memory: stores programs and intermediate data.
  - ❖ Input Devices: accept data from outside world.
  - ❖ Output Devices: presents data to the outside world.
  - ❖ Bus: communication between the components
- An analogy with Human Information Processors:
  - ❖ CPU – brain's reasoning powers
  - ❖ Memory – brain's memory
  - ❖ Input Devices – eyes, ears, sensory sub-system
  - ❖ Output Devices – mouth, hands, facial and body expressions
  - ❖ Bus – nerves

24

Memory

Primary Memory (Main Memory)

- All programs and their data must be hold in this memory while they are being executed
- Constructed from semiconductor bit cells, each capable of storing one bit.
- Memory are typically accessed in groups of fixed size called words. The number of bits in each word is called word length and is typically 16, 32 or 64 bits.
- Primary memory is typically RAM
- Memory access time: time to access one word.
- Random-access memory (RAM): memory where the access time is short and fixed regardless of its location



Cache

- An adjunct to the main memory, fabricated on the processor chip
- Much smaller and faster than the main memory
- Holds only part of the program and data currently being executed by the processor

25

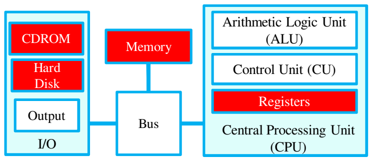
Memory

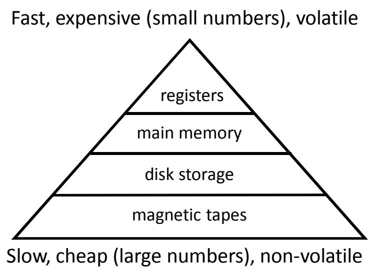
Register

- Fastest memory in the system
- Used to store the data that is currently being executed

Secondary Storage

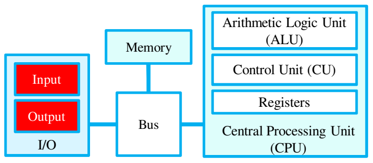
- Primary memory is expensive and is volatile (does not retain the content when power is turned off)
- Secondary storage is a non-volatile memory that is used to store large amounts of data and programs. Example: magnetic disks, optical disks, flash memory devices
- A magnetic disk can store 500 or more Gigabytes




Fast, expensive (small numbers), volatile
registers
main memory
disk storage
magnetic tapes
Slow, cheap (large numbers), non-volatile

26

Input and Output Unit



Input Units

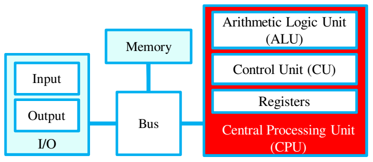
- Computer accepts coded information through input units
- Provides human-computer interaction: keyboards, mouse, touch screen
- Provides digital communication facilities such as Internet to allow computers to communicate
- Provides data transfer for secondary storage devices: hard disk (file transfer or load a program into memory), microphone (capture, audio input to be sampled and converted into digital codes for storage)

Output Units

- Counterpart of the input units. Used to send processed results to the outside world
- Example: printer, monitor, graphic display, etc.
- Some units performs both functions of input and output, e.g., touch screen, network card, etc.

27

Central Processing Unit (CPU)

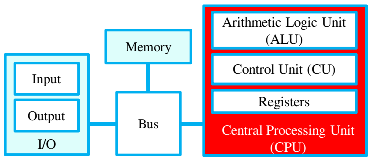


- Execute programs (a set of instructions to accomplish a certain task) that are stored in the main memory by fetching and then executing them one after another.
- Arithmetic Logical Unit (ALU)
  - Performs *arithmetic* and *logical* operations such as addition, subtraction, multiplication, comparison of numbers, etc.
  - These operations is initiated by bringing the required operands into the processor, where the operation is performed by the ALU
  - For example, when adding two numbers in the memory, these two numbers are first be brought into the processor before addition is carried out by ALU.
  - The sum may then be stored in the memory or retained in the CPU (register) for immediate use.
- Registers are *high speed memory* used to *temporarily* store the variables in an instruction *currently executed* by the CPU.

28

Central Processing Unit (CPU)

- Execute programs (a set of instructions to accomplish a certain task) that are stored in the main memory by fetching and then executing them one after another.

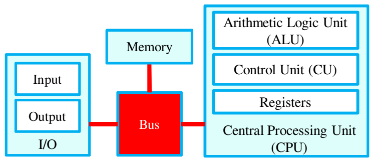


- Control Unit co-ordinates the operations between different units
  - Control circuits are responsible for generating the timing signals and control signals that govern a particular computer operation (e.g. addition, or data transfer)
  - *Timing signals* synchronizes the computer operation so that all steps are executed in the right sequence.
  - *Control signals* controls the data transfer. For example, to perform REG1 + REG2, the first and the second input to the ALU is directed from REG1 and REG2, respectively and not other registers
  - In reality, a control unit is not a well-defined, physical structure but is rather physically distributed throughout the computer.

29

Bus

- Buses are communication links allowing data, instructions and memory addresses to be referenced between different components (CPU, main memory and I/O devices) in a computer system.
- A bus is a group of lines such that a word of data can be transferred (one bit per line) at a time.
- ***Bus arbiter*** (not shown in the diagram) is controller that decides which requesting device is granted access to the bus when multiple devices share the same bus.



30

Basic Operational Concepts

- An **instruction** specifies an operation and its **data** operands. For example, the instruction ADD R4, R3, R2 for the ARMS processor adds the content of R3 and R2 and stores the result in R4.
- The basic instructions type are:

Load	Read a data operand from memory or an input device into the CPU
Store	Write a data operand from a CPU register to memory or an output device
Operate	Perform an arithmetic or logic operation on data operands in CPU registers
Flow	Control the execution flow (*if-else* and *while* statements)

- A **program** is a sequence of instructions, executed one after another, to perform a particular task.
- All programs and its data needs to be stored in the **main memory** before it can be executed.

31

Basic Operational Concepts

Sample program for the ARM processor:

- Write a program to perform the following operation:

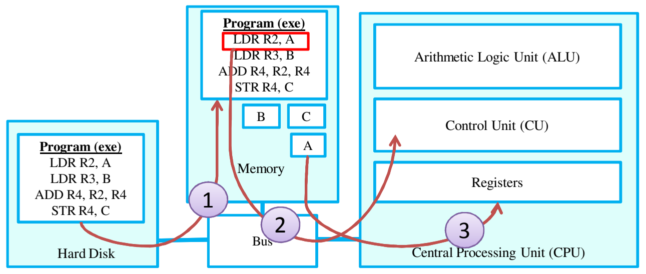
  C = A + B

- A, B, and C, are **labels** representing memory word addresses;
- Ri are processor registers

```
LDR     R2, A           ; Load A into CPU register 2
LDR     R3, B           ; Load B into CPU register 3
ADD     R4, R2, R3      ; Add register 3 and 2 and save into register 4
STR     R4, C           ; Store result into memory location C
```

32

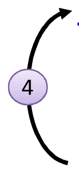
Basic Operational Concepts



The basic operation of a Computer:

(1) Load:

Program (***instruction***+***data***) is loaded from storage unit into memory.



(2) Fetch:

By default, CPU fetches the **instructions** (of the program) from the memory *line by line* sequentially. The control unit decodes the instruction to identify the *instruction* (load) with its *field* (A and R2)

(3) Execute:

Each instruction line is executed by CPU.

33

NUMBER SYSTEM

- **Positional Notation Number System**
  - Decimal Number (Base 10)
  - Binary (Base 2)
  - Octal (Base 8)
  - Hexadecimal (Base 16)
  - Conversion from other Bases to Base 10
  - Range of a Number System (Integers)
  - Fractions

34

Positional Notation Number System

- **Number System**: a mathematical notation used to represent numbers of a given set by using symbols in a consistent manner.
- **Position Independent Number Systems:** a number system where the number of digits representing a number does not reflect the true value of the number.

  Example : Roman: I, II, III, IV, V, ….
- **Positional Notation Number Systems** defines a finite set of numeric symbols/digits and then use *multiple digits* to represent numbers that are larger than the number of symbols.

  Example: Decimal: 1, 2, 3, 4, …, 10, 11, …., 99, 100, …
- For these systems, the **Base** of a number system represents the number of digits that are defined:
  - **Binary** (Base 2): 0, 1
  - **Octal** (Base 8): 0, 1, 2, 3, 4, 5, 6, 7
  - **Decimal** (Base 10): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - **Hexadecimal** (base 16): 0, 1, 2, 3, 4, 5, 7, 8, 9, A, B, C, D, E, F

35

Positional Notation Number System

Why do we focus on binary, octal, decimal and hexadecimal number system whereas we can also have Base 3, Base 5, Base 6, …?

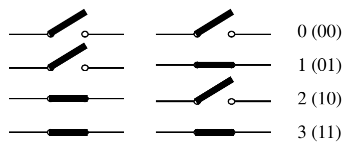
- Human are familiar with **Base 10 (Decimal)** number system
  a. Base 10 uses 10 digits (0, 1, 2, …9) to represent numbers
  b. Origin: counting on the fingers
  c. "Digit" from the Latin word *digitus* meaning "finger"
- A computer "has only two fingers" and therefore it operates in the **Base 2** (**Binary)** number system.
  a. The input to the logic gates contains only two possible values. The elementary storage units inside computer are electronic switches. Each switch holds one of two states: *on* (1) or *off* (0).
  b. Base 2 uses a **bit** (two *bi*nary digi*t*), 0 or 1, to represent numbers

- **Base 8 (Octal)** or **Base 16 (Hexadecimal)** falls within the same family (power of two) and can be converted easily.

36

Positional Notation Number System

- To represent a value larger than the total number of digits in a number system, we can use **multiple** digits.
  - For example, the number *twelve* cannot be represented using a single *decimal* digit (ten valid symbols only), and we have to use two digits (12_{10})
  - For example, 1 switch in a *binary* system can represent 2 values and two switches can represent 4 values.



- In this system, the position of the digits is assigned a certain weight based on the base value.
  - **Binary** (Base 2): weights are in the powers-of-2.
  - **Octal** (Base 8): weights are in the powers-of-8.
  - **Decimal** (Base 10): weights are in the powers of 10
  - **Hexadecimal** (base 16): weights are in the powers-of-16.

37

Positional Notation Number System

Notations:

- To differentiate between the different bases, the subscript value following the number itself denotes the base that the number is being represented. For example

  10100_2 (in base 2), 263_7 (in base 7)

  129_{10} (in base 10), $8AE7_{16}$ (in base 16)

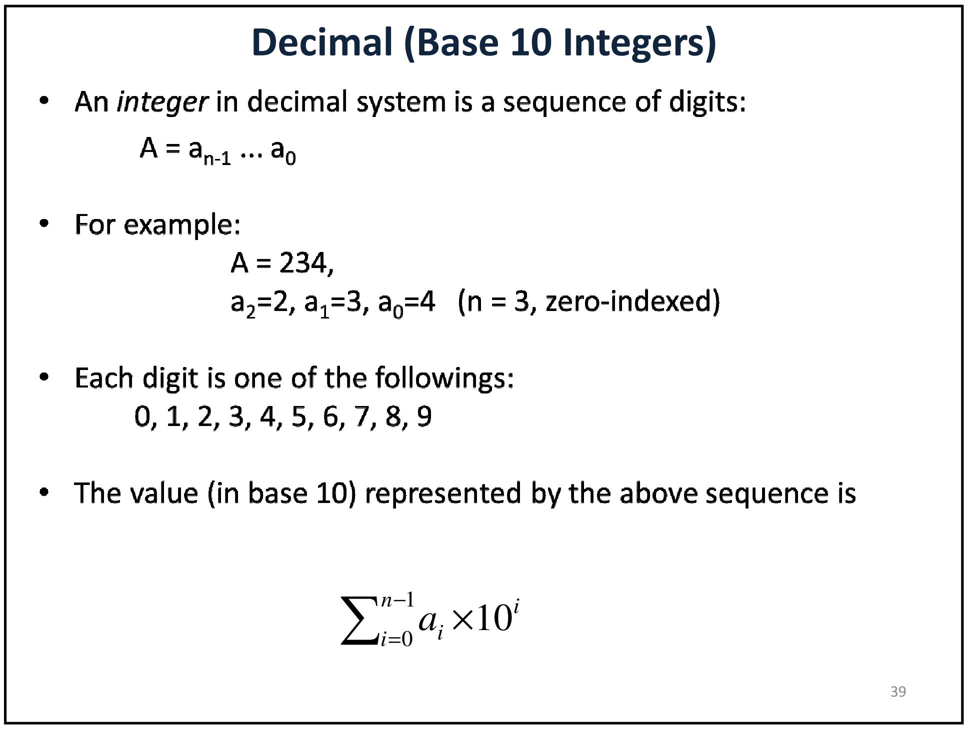
- The largest digit in a number must not be bigger than the base. For example, the largest digit in 10100_2 is 1 and the largest digit in 523_6 is 5.
- The same set of digits in different number system represents different numbers. For example:

  $111_2 \neq 111_8 \neq 111_{10} \neq 111_{16}$

  $1101111_2 \neq 157_8 = 111_{10} = 6F_{16}$

- A number is not followed by any subscript, the default base (Base 10) is assumed. For example, $124 = 124_{10}$.

38



Decimal (Base 10 Integers)

- An *integer* in decimal system is a sequence of digits:

  $A = a_{n-1} \dots a_0$

- For example:

  A = 234,

  $a_2=2$, $a_1=3$, $a_0=4$ (n = 3, zero-indexed)

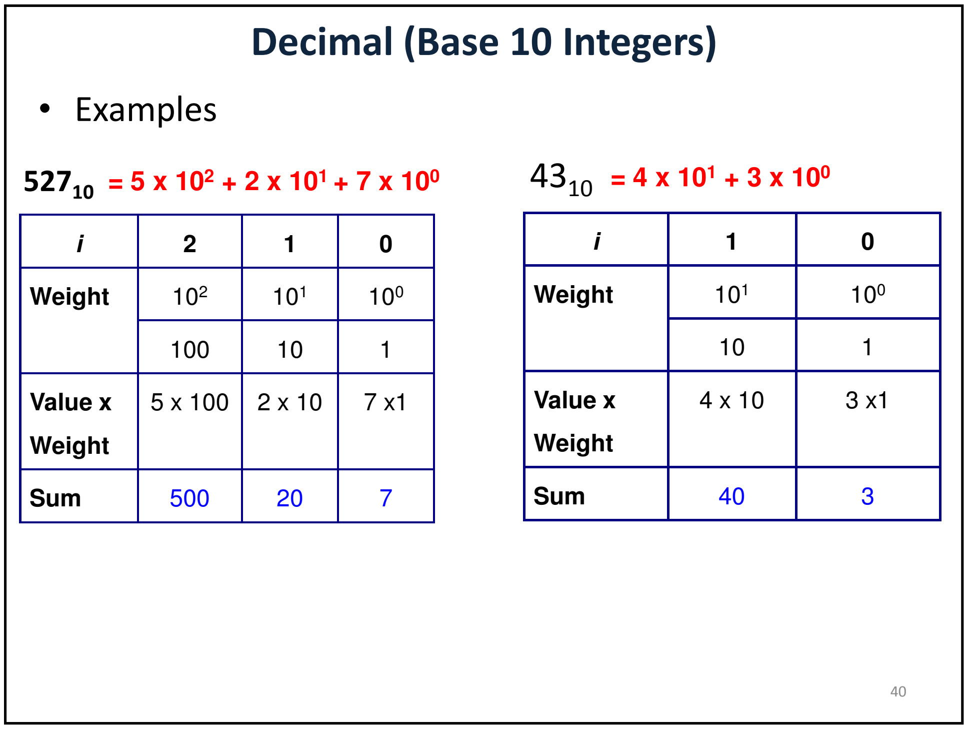
- Each digit is one of the followings:

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9

- The value (in base 10) represented by the above sequence is

$$\sum_{i=0}^{n-1} a_i \times 10^i$$

39



Decimal (Base 10 Integers)

- Examples

527_{10} = $5 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$

i	2	1	0
Weight	10^2	10^1	10^0
	100	10	1
Value x Weight	5 x 100	2 x 10	7 x1
Sum	500	20	7

43_{10} = $4 \times 10^1 + 3 \times 10^0$

i	1	0
Weight	10^1	10^0
	10	1
Value x Weight	4 x 10	3 x1
Sum	40	3

40

Binary (Base 2 Integers)

- A number in binary system is a ***sequence of bits***, possibly followed by a binary point and then a sequence of bits.
- The actual value (in Base 10) represented by the binary number is:

$$A = \sum_{i=0}^{n-1} a_i \times 2^i$$

$1101\ 0110_2$ = 128 + 64 + 16 + 4 + 2 = 214_{10}

i	7	6	5	4	3	2	1	0
Weight	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
Value x Weight	1x128	1x64	0x32	1x16	0x8	1x4	1x2	0x1
Sum for Base 10	128	64	0	16	0	4	2	0

Binary (Base 2 Integers)

- In computer terminology,

  1 Kilo (K) = 2^{10}

  1 Mega (M) =2^{20}

  1 Giga (G) =2^{30}

  1 Tera (T) =2^{40}

Examples:

4KB	= $2^2 \times 2^{10} = 2^{12}$ = 4,096 bytes
16MB	= $2^4 \times 2^{20} = 2^{24}$ = 16,777,216 bytes
2TB	= $2^1 \times 2^{40} = 2^{41}$ = 2,199,023,255,552 bytes

Octal (Base 8 Integers)

1. The possible digits in an octal number are:

   0, 1, 2, 3, 4, 5, 6, 7

2. The actual value (in Base 10) represented by the octal digit sequence $A_{n-1}...A_1A_0$ is:

$$\sum_{i=0}^{n-1} a_i \times 8^i$$

624_8 = **404_{10}**

Position	**2**	**1**	**0**
Weight	8^2 64	8^1 8	8^0 1
Value x Weight	6 x 64	2 x 8	4 x 1
Sum	384	16	4

43

Hexadecimal (Base 16 Integers)

1. The possible digits in a hexadecimal number are:

   0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

   where A, B, C, D, E, F stand for 10, 11, 12, 13, 14, 15 in decimal respectively.

2. The actual value (in base 10) represented by the hexadecimal digit sequence $A_{n-1}...A_1A_0$ is: $\sum_{i=0}^{n-1} a_i \times 16^i$

$6{,}704_{16}$ = **$26{,}372_{10}$**

Position	**3**	**2**	**1**	**0**
Weight	16^3 4,096	16^2 256	16^1 16	16^0 1
Value x Weight	6 x 4,096	7 x 256	0 x 16	4 x 1
Sum	24,576	1,792	0	4

44

Conversion from other bases to Base 10

- Convert the following numbers to Base 10:

$AEC_{16} = 10x16^2 + 14x16^1 + 12x16^0 = 2796_{10}$

$17_8 = 1x8^1 + 7x8^0 = 15_{10}$

$77_5 = 7x5^1 + 7x5^0 = 42_{10}$

$10101_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
$= 21_{10}$

45

Range of a Number System (Integers)

- The **range** of a number system is how many different numbers it can represent.
- The range *R* for *K* digits in Base *B* is:

$$\boxed{R = B^K}$$

- Conversely, the number of digits *K* required to represent a total of *R* numbers in a Base *B* number system is:

$$\boxed{K = \lceil log_B R \rceil}$$

46

Range of a Number System (Integers)

- Example:

  How many numbers can 1 bit represent? How about 4 bits?

  1 bit , base 2 $\rightarrow$ represents up to 2^1 or 2 values (0 or 1)

  4 bits, base 2 $\rightarrow$ represents up to 2^4 or 16 values (0000, 0001, 0010, …, 1111)

  How many numbers can 2 digits in Base 8 represent?

  2 digits , base 8 $\rightarrow$ represents up to 8^2 or 64values

  00, 01, 02, …, 07
  10, 11, 12, …, 17,
  20, 21, 22, …, 27,
  …
  70, 71, 72, …, 77

47

Range of a Number System (Integers)

- Example:

  How many bits are required to represent 32, 64, 1024, 40 and 100 values?

  32 $\rightarrow$ requires $\lceil log_2 32 \rceil$ or 5 bits
  64 $\rightarrow$ requires $\lceil log_2 64 \rceil$ or 6 bits
  1024 $\rightarrow$ requires $\lceil log_2 1024 \rceil$ or 10 bits
  40 $\rightarrow$ requires $\lceil log_2 40 \rceil$ or 6 bits
  100 $\rightarrow$ requires $\lceil log_2 100 \rceil$or 7 bits

48

Fractions

- A number with fractional part (example 15.35_{10} in Base 10) is a sequence of digits, followed by a decimal point or **radix point** in general, and then a sequence of digits:
- Radix point separates the **integral part** (with positive powers of the base, with value ≥1 and ≤0) from the **fractional part** (with negative powers of the base, with 1<value<0).

  $a_n \ldots a_0 . a_{-1} \ldots a_{-k}$

  a_n : most significant digit; a_{-k} : least significant digit
- For the **fractional part**, the *successive* digits have the following values

  $1/B, 1/B^2, 1/B^3, 1/B^4, \ldots$

  or equivalently

  $B^{-1}, B^{-2}, B^{-3}, B^{-4}, \ldots$

  where B is the base of the number

49

Fractions

- 0.101011_2 **= 0.671875_{10}**

Position	-1	-2	-3	-4	-5	-6
Weight	2^{-1} 1/2	2^{-2} 1/4	2^{-3} 1/8	2^{-4} 1/16	2^{-5} 1/32	2^{-6} 1/64
Value x Weight	1 x 1/2	0 x 1/4	1x 1/8	0 x 1/16	1 x 1/32	1 x 1/64
Sum	.5		0.125		0.03125	0.015625

- $15.35_{10} = 1 \times 10^1 + 5 \times 10^0 + 3 \times 10^{-1} + 5 \times 10^{-2}$
- $6F.77_{16} = 6 \times 16^1 + 15 \times 16^0 + 7 \times 16^{-1} + 7 \times 16^{-2}$
  $= 111.46484375_{10}$
- $101.01_2 = 1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-2}$
  $= 5.25_{10}$

50

Fractions

Example:

- Convert $A3F.C_{16}$ to decimal

$$A3F.C_{16} = 10 \times 16^2 + 3 \times 16^1 + 15 \times 16^0 + 12 \times 16^{-1}$$
$$= 2623.75_{10}$$

- Convert 132.7_8 to decimal

$$132.7_8 = 1 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 + 7 \times 8^{-1}$$
$$= 90.875_{10}$$

51

Fractions

- Example:

$$0.011_2 = 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$$
$$= 0.0 + 0.25 + 0.125$$
$$= 0.375_{10}$$

- There may NOT be an **exact** relationship between fractional numbers in different number base. For example:

0.1_{10} cannot be represented exactly in binary (Base 2)

Equivalent value is $0.0001100110011\ldots_2$

0.1_3 cannot be represented exactly in decimal (Base 10)

Equivalent value $0.333333\ldots._{10}$

52

Fractions

- Moving the radix point one place to the right multiplies the number by the base number. Example:
  - $139.15_{10} \times 10^1 = 1391.5_{10}$
  - $139.15_{10} \times 10^2 = 13912_{10}$
  - $101.11_2 \times 2^1 = 1011.1_2$ (or equivalently $5.75_{10} \times 2_{10} = 11.5_{10}$)
  - $101.11_2 \times 2^3 = 101110_2$ (or equivalently $5.75_{10} \times 8_{10} = 46_{10}$)
- Move the number point one place to the left divides the number by the base number. Example:
  - $139.15_{10} / 10^1 = 13.915_{10}$
  - $139.15_{10} / 10^2 = 1.3915_{10}$
  - $101.11_2 / 2^1 = 10.111_2$ (or equivalently $5.75_{10} / 2_{10} = 2.875_{10}$)
  - $101.11_2 / 2^3 = 0.10111_2$ (or equivalently $5.75_{10} / 8_{10} = 0.71875_{10}$)

53

Review Question

Question 1:
Convert the following value to Base 10:
a) $FF02_{16}$
b) 776_8
c) $1001\ 0110_2$

Question 2:
How many numbers can be represented using
a) Base 10, 2 digits
b) Base 2, 16 digits

Question 3:

How much digits do we need to cover the unsigned value 23000 in Base 2, 10 and 16?

54