Introduction to programmable computing devices
Overview

• What is a computer?

• How was computer developed?

• How to encode instructions in past programmable devices?
What is a computer

- Computer:

  - A computer is a "reckoning" or computing device....
  - In fact: a computer is an *programmable* computing device (that helps humans do their chores)

- How was the computer developed/invented:

  - Humans invented *computing devices* in ancient times.
  - In the industrial revolution, *programmable machines* were invented
  - The computer is a *combination* of these 2 ideas
Overview

• What is a computer?

• How was computer developed?

• How to encode instructions in past programmable devices

• Logical (functional) view of a computer

• Program flow

• Types of instructions that a computer can execute
Some *computing devices* invented throughout history

- "Ancient" computing device: the **abacus**

**Toy abacus:**
- # beads in row 1 = unit value
- # beads in row 2 = 10's value
- # beads in row 3 = 100's value
- And so on

**Real abacus:**
- # beads in column 1 = unit value
- # beads in column 2 = 10's value
- # beads in column 3 = 100's value
- And so on
- 1 bead in upper half = 5 beads in lower half
- Value represented by abacus = 63571
The difference engine consists of a number of columns, numbered from 1 to N.

The machine is able to store one decimal number in each column.

The machine can only add the value of a column \( k + 1 \) to column \( k \) to produce the new value of \( k \).

Column N can only store a constant,

Column 1 displays (and possibly prints) the value of the calculation on the current iteration.
Programmable machines, programs and instructions

- Programmable machine:

  - *Programmable machine* = a device which *function* can be *altered* by a *program*

- Program:

  - *Program* = a *series of instructions* that accomplishes a *specific task*

  - Example:

    - Each *row* of holes in the *card board* is an *instruction* for some machine
    - The *position* of a hole *encodes* a *certain meaning*
    - The *program* consists of a *series of rows (= instructions)* on the card board
Examples of Programmable machines

**Mechanical piano/music box**

- A mechanical piano can play different songs
- The drum in the center of the piano contains spikes and rotates slowly
- Spikes at different position causes the piano to play a different note
- Different spike patterns (on different drums) will cause the piano to play a different song
Examples of Programmable machines

Mechanical loom

- The **mechanical loom** can weave fabric in **different patterns**
- The loom's movement is controlled by holes punched in a card
- **Different hole patterns** (on different cards) will cause the loom to weave a **different pattern**
Overview

• What is a computer?
• How was computer developed?
• How to encode instructions in past programmable devices
• Logical (functional) view of a computer
• Program flow
• Types of instructions that a computer can execute
Instruction encoding

• Instructions (= program instructions) tells a machine what to do
• Instructions are represented using an encoding method
• Example encoding:
  • 1 means add
  • 2 means subtract
  • And so on.

• Another example: encoding a song on paper
  • The location of a hole in the paper corresponds to a particular musical note
  • Encoding a song: a series of holes in the paper make the mechanical piano/music box play the notes of a song
Storing instructions and information
Overview

• Logical (functional) view of a computer
• Program flow
• Types of instructions that a computer can execute
Most common perception of a computer

- This is the most common view (perception) of a computer is as follows:
Most common perception of a computer (cont.)

- Component of the computer by their functionality:

  - **Input devices**: allow users to enter input to the computer (mouse, keyboard, microphone, camera)
  - **Output devices** allow the computer to display output to the user (monitor, printer, speaker)
  - **Input/output devices**: used by the computer to store data and/or communicate with other computers (CD-rom, floppy drive, hard drive, modem, network)
  - **Computer system** (that's the box in the middle of the picture)
Most common uses of a computer

• Today, the most popular usages of a computer are:

  • Web browsing
  • Play games
  • Text processing (for homework)

The operations of these tasks differ widely from each other.

And yet, they are accomplished using the same machine (a computer) through executing a different computer program.
Hardware and Software

- Computer jargon:
  - **Hardware** = the *physical parts* of a computer
    - The case containing the computer
    - Keyboard
    - Terminal
    - Mouse
    - Etc
Hardware and Software (cont.)

- **Software** = the *computer programs* that you run with a computer
  - Web browser
  - PC games
  - Microsoft Word
  - Microsoft Excel
  - Etc
Hardware and Software (cont.)

- We will first study how a computer (hardware) is connected together so it can execute computer programs.
- Then we will study what computer software does.
Logical view of a computer (hardware)

- Logical (functional) view of a computer

![Diagram showing logical view of a computer](image-url)
Logical view of a computer (hardware) (cont.)

- The **input devices**, **output devices** and **input/output (I/O) devices** are called **peripheral devices**
- A **computer system** consists of:
  - The **Central Processing Unit** (a.k.a. the **CPU**)
  - The **memory** or **Random Access Memory (RAM)**

In this webnote, we will first study the **computer (RAM) memory**
Functionality of the RAM (Memory)

• Structure of the Memory (RAM)

  • The RAM consists of multiple memory cells:

  • Each memory cell is uniquely identified by its memory address

  • Memory addresses always start at zero (0)

  • The last memory address depends on the amount of memory installed in the computer system
Operation of the memory

- **Memory** can store and recall (retrieve) values for the CPU:

Each **memory cell** can store one **number**

**Example:**

- Memory location 0 stores the value 13
- Memory location 1 stores the value 3
- Memory location 2 stores the value 0
- Memory location 3 stores the value 45
- ...

```
<table>
<thead>
<tr>
<th>RAM (memory)</th>
<th>1 memory cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>45</td>
</tr>
</tbody>
</table>
```

In this example:
Operation of the memory (cont.)

• Each memory cell can store and recall a value by the command by the CPU:

• A memory cell is like the STO/RCL function of a calculator:

• Numbers that are stored in a memory cell can encode:

  • a instruction
  • a piece of information
Operation of the memory (cont.)

- Computer (RAM) memory:

  - The **RAM memory** works just like the **Store/recall** buttons in the above demo.
  
  - The **RAM memory** is under the control of the **CPU**:
    
    - The **CPU** can store a value in a specific memory location in the **RAM memory**
    - The **CPU** can **recall** the stored value later when it needs it.
Storing information in memory cells using numbers

- Information can be stored as *numbers* by using an encoding method.
- An *encoding method* is simply an *agreement* on a representation of some facts by specific *numbers*.
- 2 common type of things are represented by *numbers* inside a computer:

1. The *instructions* that tells the computer what to do
2. Various kinds of *information* that are stored and manipulated by the computer.
Computer programs (Software)

• Computer program:

  • A computer program (or software) is a (very long) list of instructions that are executed by the computer

Schematically: what a computer program look like

"add x to y"
"subtract x from y"
"multiply x with y"
....
Computer programs (Software) (cont.)

- The instructions are not represented in English, but by some number
  (Some programs contains over a billion instructions !)

- Another commonly used name for computer program is computer application
  I will use these 2 terms interchangeably
Representing computer instructions

• Representing computer instructions by numbers:

  • A computer can perform Mathematical operations and logical operations:

    Example:
    • Add
    • Subtract
    • Multiply
    • Divide
    • Compare 2 numbers
    • And 2 logical value
    • Or 2 logical value
Representing *computer instructions* (cont.)

- Each **operation** is represented by a **unique encoding**

**Example:**
- $0 = \text{add}$
- $1 = \text{subtract}$
- $2 = \text{multiply}$
- And so on...
Computer programs - revisited

• Computer program: (the naked truth)

  • A computer program (or software) is a (very long) list of numbers that represents instructions that are executed by the computer

• Schematically: what a computer program look something like

  1256
  875
  7263
  ....

Each number represents a computer instruction
Representing *information*

- A *computer* is used to process *information*

  - How is *information* stored inside a *computer* ???
Representing information (cont.)

• Representing various kinds of information by numbers:

  • Same technique is used to encode any type of information
  • Example: encoding gender information
    - 0 = male
    - 1 = female

  • Example: encoding marital status information
    - 0 = single
    - 1 = married
    - 2 = divorced
    - 3 = widowed
How can we tell what a number stored in the computer mean?

- In the previous examples, we saw that the number 0 can mean:
  - Add (in instruction encoding)
  - male (in gender information)
  - single (in marital status information)

- We can only tell what is the meaning of the number 0 if we are given:
  - context information
Now we can tell what a number stored in the computer mean!

• If the computer is executing an instruction, then:
  • The number 0 means "perform an add operation"

• If the computer is examining gender information, then:
  • The number 0 means "male"

• If the computer is examining marital status information, then:
  • The number 0 means “single”
Computer memory and the binary number system
The switch is a *memory device*

- The electrical switch is a *memory device*:

- The electrical switch can be in one of these 2 states:
  - off (we will call this state 0)
  - on (we will call this state 1)
Memory cell used by a computer

- One switch can be in one of 2 states
- A row of $n$ switches:

  ![Diagram of a row of switches](https://www.123f.com)

  can be in one of $2^n$ states!
The *binary number system*

- The binary number system uses 2 digits to encode a number:
  - 0 = represents no value
  - 1 = represents a unit value

- That means that you can *only use* the digits 0 and 1 to write a *binary number*

  - Example: some binary numbers
    - 0
    - 1
    - 10
    - 11
    - 1010
    - and so on.
The *binary number* system (cont.)

- The **value** that is *encoded (represented)* by a **binary number** is computed as follows:

<table>
<thead>
<tr>
<th>Binary number</th>
<th>Value encoded by the binary number</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{n-1} \ d_{n-2} \ ... \ d_1 \ d_0$</td>
<td>$d_{n-1} \times 2^{n-1} + d_{n-2} \times 2^{n-2} + ... + d_1 \times 2^1 + d_0 \times 2^0$</td>
</tr>
</tbody>
</table>
The *binary number* system (cont.)

Example:

<table>
<thead>
<tr>
<th>Binary number</th>
<th>Value encoded by the binary number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$0 \times 2^0 = 0$</td>
</tr>
<tr>
<td>1</td>
<td>$1 \times 2^0 = 1$</td>
</tr>
<tr>
<td>10</td>
<td>$1 \times 2^1 + 0 \times 2^0 = 2$</td>
</tr>
<tr>
<td>11</td>
<td>$1 \times 2^1 + 1 \times 2^0 = 3$</td>
</tr>
<tr>
<td>1010</td>
<td>$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 8 + 2 = 10$</td>
</tr>
</tbody>
</table>
What does all this have to do with a computer?

- Recall what we have learned about the Computer RAM memory:

  - The RAM consists of multiple memory cells:

    Each memory cell stores a number
What does all this have to do with a computer? (cont.)

- The connection between the computer memory and the binary number system is:

  - The computer system uses the binary number encoding to store the number

Example:

<table>
<thead>
<tr>
<th>How we perceive it:</th>
<th>The reality:</th>
</tr>
</thead>
<tbody>
<tr>
<td>address of memory cell</td>
<td>address of memory cell</td>
</tr>
<tr>
<td>0</td>
<td>000...000</td>
</tr>
<tr>
<td>1</td>
<td>000...001</td>
</tr>
<tr>
<td>2</td>
<td>000...010</td>
</tr>
<tr>
<td>3</td>
<td>000...011</td>
</tr>
</tbody>
</table>

- Each byte has 8 bits

A memory address is 32 bits long!!!
What does all this have to do with a computer? (cont.)

- Note: the address is also expressed as a binary number

A computer can have over 4,000,000,000 bytes (4 Gigabytes) of memory.

So we need a 32 bytes to express the address
Computer memory

• A computer is an electronic device
• Structure of a RAM memory:

  • The RAM memory used by a computer consists of a large number of electronic switches
  • The switches are organized in rows
  • For historical reason, the number of switches in one row is 8
Computer memory (cont.)

Details

- In order to store text information in a computer, we need to encode:
  - 26 upper case letters ('A', 'B', and so on)
  - 26 lower case letters ('a', 'b', and so on)
  - 10 digits ('0', '1', and so on)
  - 20 or so special characters ('&', '%', '$', and so on)

for a total of about 100 different symbols

- The nearest even power $2^n$ that is larger than 100 is:
  - $2^7 = 128 \geq 100$

- For a reason beyond the scope of this course, an 8th switches is added
Computer memory (cont.)

• This is was a **portion** of the **RAM memory** looks like:

```
address of memory cell       RAM (memory)
000...000  00001101
000...001  00000011
000...010  00000000
000...011  00101101
```

• **What information** is stored in the RAM memory depends on:

  • The **type** of data (this is the **context information**)

    Example of **types**: marital status, gender, age, salary, and so on.

  • This determines the **encoding scheme** used to interpret the number
Computer memory jargon:

- **bit** = (binary digit) a *smallest* memory device
  A bit is in fact a switch that can remember 0 or 1
- (The digits 0 and 1 are digits used in the binary number system)

- **Byte** = 8 bits
  A byte is in fact one row of the RAM memory

- **KByte** = kilo byte = 1024 (= $2^{10}$) bytes (approximately 1,000 bytes)
- **MByte** = mega byte = 1048576 (= $2^{20}$) bytes (approximately 1,000,000 bytes)
- **GByte** = giga byte = 1073741824 (= $2^{30}$) bytes (approximately 1,000,000,000 bytes)
- **TByte** = tera byte
Combining adjacent memory cells

- A byte has 8 bits and therefore, it can store:

\[ 2^8 = 256 \text{ different patterns} \]

(These 256 patterns are: 00000000, 00000001, 00000010, 00000011, .... 11111111)
Combining adjacent memory cells (cont.)

- Each pattern can be encoded exactly one number:

  - 00000000 = 0
  - 00000001 = 1
  - 00000010 = 2
  - 00000011 = 3
  - ...
  - 11111111 = 255

Therefore, one byte can store one of 256 possible values (You can store the number 34 into a byte, but you cannot store the number 456, the value is out of range)
Combining adjacent memory cells (cont.)

- The computer can combine adjacent bytes (memory cells) and use it as a larger memory cell

Schematically:

2 bytes: 

```
  00 00 00 00 00 00 00 00
  01 01 01 01 01 01 01 01
```

one 16-bits memory cell:

```
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

A 16 bits memory cell can store one of $2^{16} = 65536$ different patterns.
Therefore, it can represent (larger) numbers ranging from: $0 – 65535$. 
Combining adjacent memory cells (cont.)

- Example: how a computer can use 2 consecutive bytes as a 16 bits memory cell:

  - The bytes at address 0 and address 1 can be interpreted as a 16 bits memory cell (with address 0)
Combining adjacent memory cells (cont.)

• When the computer accesses the RAM memory, it specifies:
  
  • The memory location (address)
  • The number of bytes it needs
Combining adjacent memory cells (cont.)

- The computer can also:
  
  - combine 4 *consecutive* bytes and use them as a 32 bits memory cell
    
    - Such a memory call can represent numbers ranging from: \( 0 - (2^{32} - 1) \) or \( 0 - 4294967295 \)

  - combine 8 *consecutive* bytes and use them as a 64 bits memory cell
    
    - Such a memory call can represent numbers ranging from: \( 0 - (2^{64} - 1) \) or \( 0 - 18446744073709551615 \)
Combining adjacent memory cells (cont.)

- There is no need (today) to combine 16 consecutive bytes and use them as a 128 bits memory cell
- But this may change in the future...