Cambridge IGCSE

Computer Science

Section 1

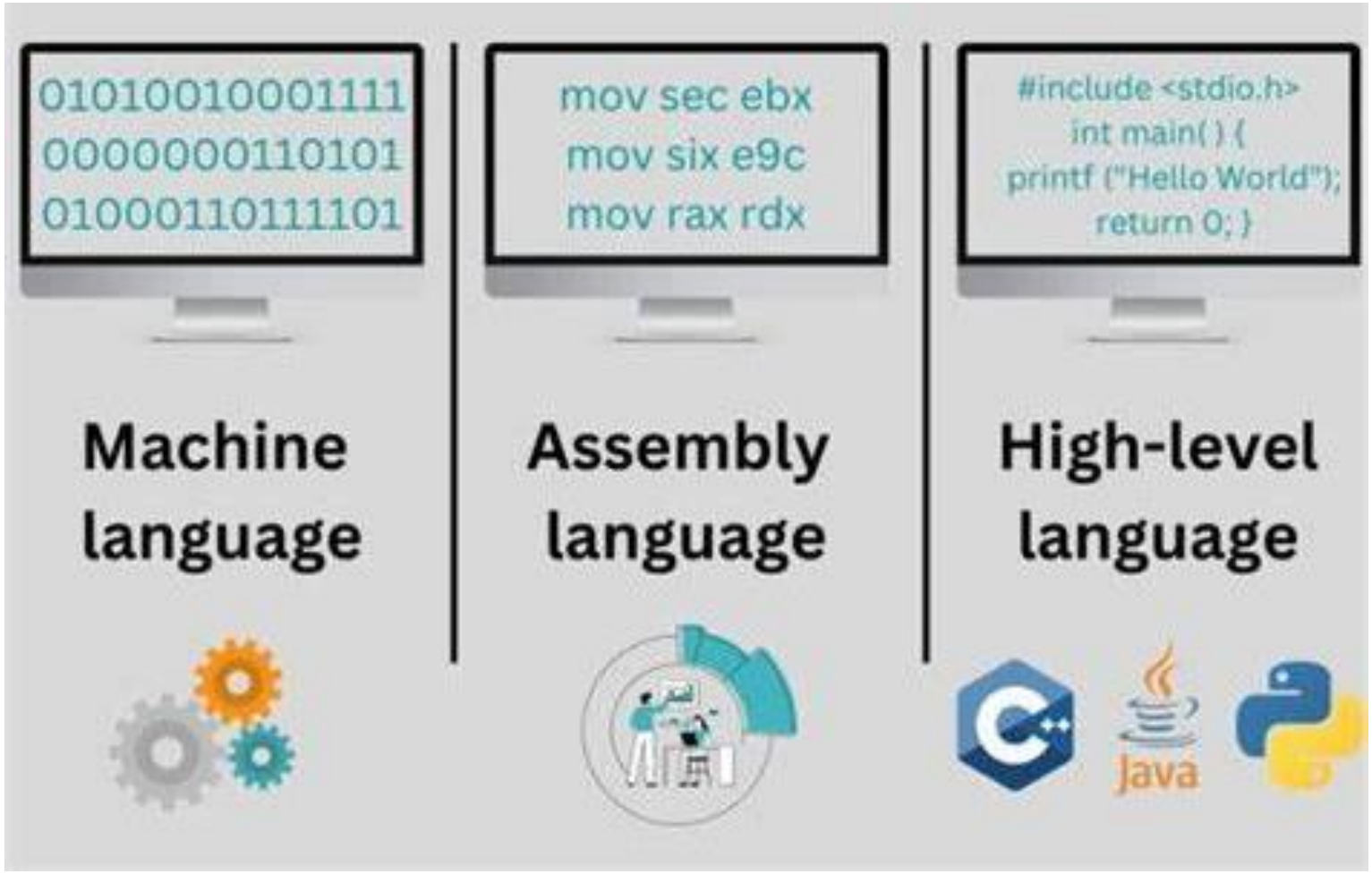
Types of programming language, translators and IDEs

Unit 4: Software

Objectives

- Explain what is meant by a high-level and low-level language, and give advantages and disadvantages
- Understand that assembly language is a form of low-level language that uses mnemonics
- Understand that an assembler is needed to translate an assembly language program into machine code
- Describe the operation of a compiler and an interpreter
- Explain the advantages and disadvantages of a compiler and an interpreter
- Explain the role of an IDE in writing program code and the common functions IDEs provide

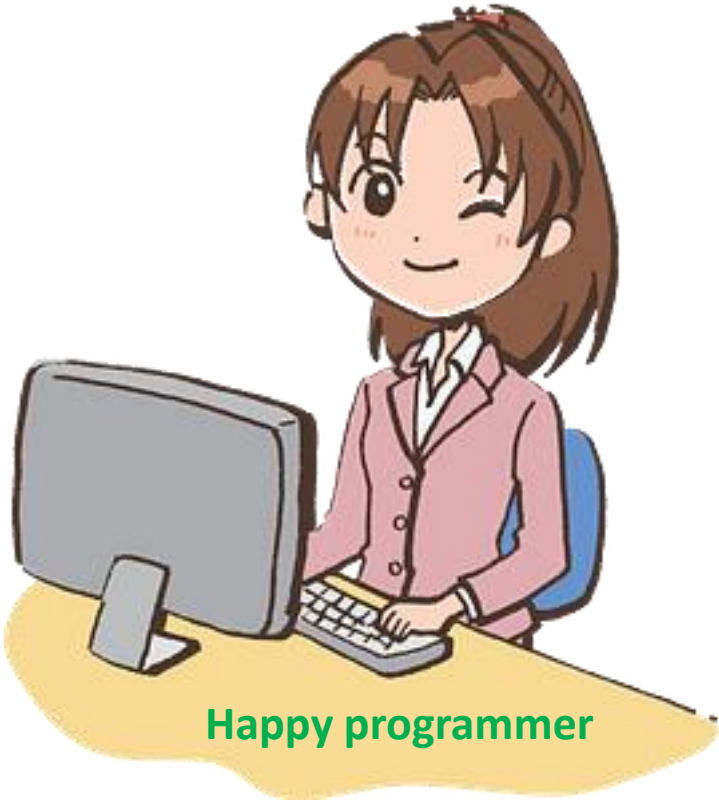
Programming languages



High-level programming languages

Are easier to read, write and understand than low-level as the language used is quite close to English.

Examples include Scratch, Python and Java.



Happy programmer

High-level programming languages

Advantages

Are easier to read, write and understand as the language used is quite close to English.

Programs can be written quickly and are easier to edit and change, than if they were written in a low-level language.

The language makes it is easier and quicker to find and fix bugs (errors).

Programs are portable - they can be used on different types of computer, they just need to be compiled (translated) for different machines.

High-level programming languages

Disadvantages

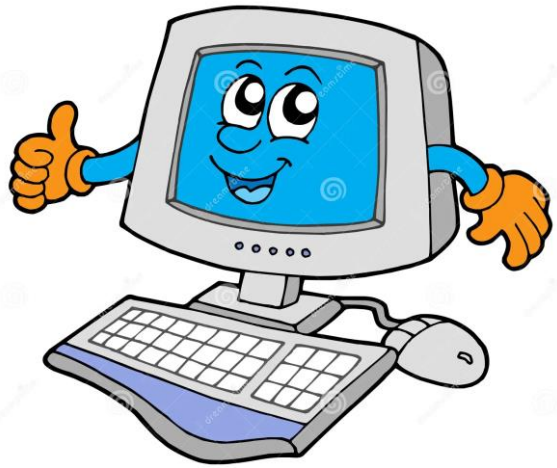
Programs have to first be translated into binary (machine code), and so, can be slower to execute (run).

The programs are larger, so take up more storage.

Programs may not be able make use of any special hardware.

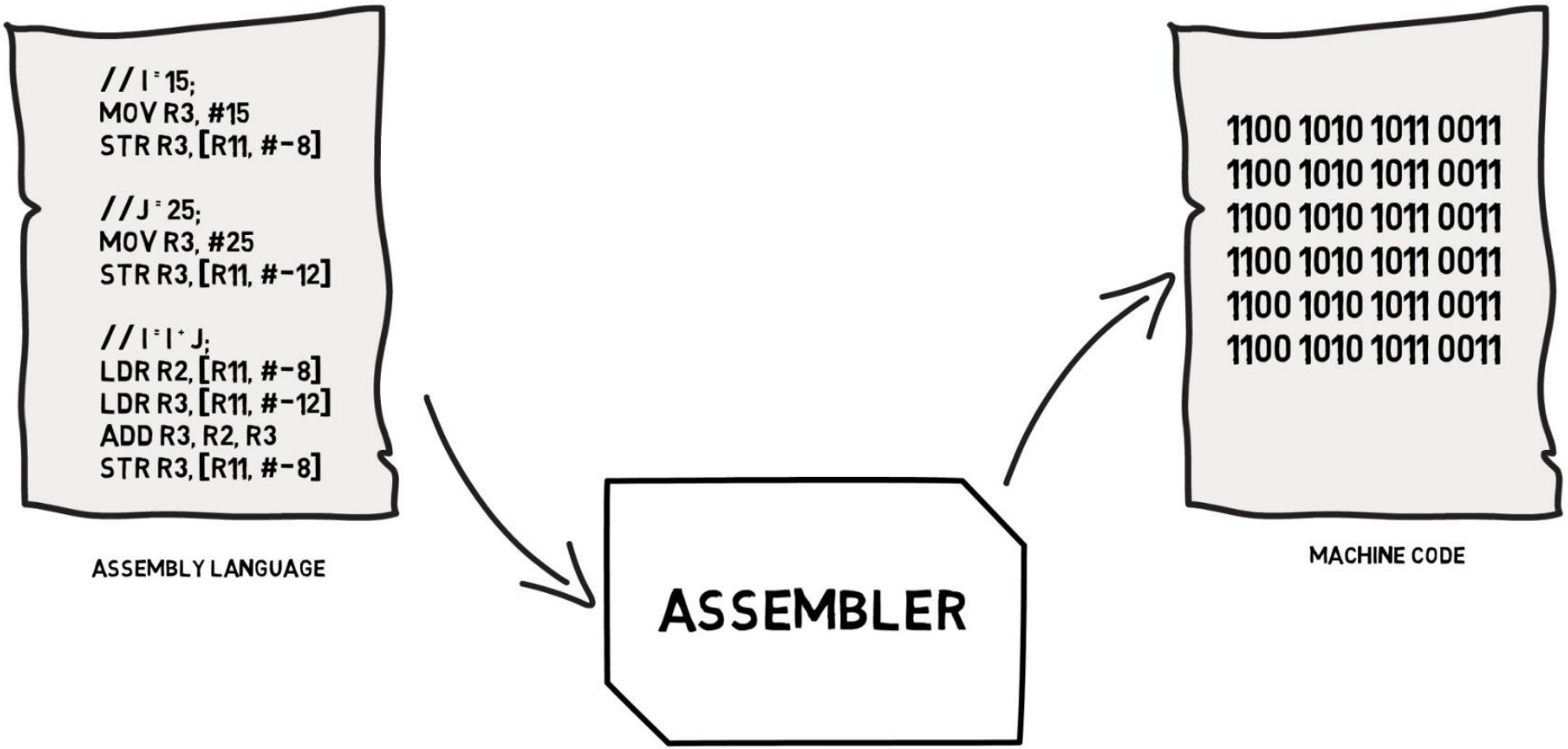
Low-level programming languages

Low level languages are much closer to what the computer actually understands - machine code, which is written in binary.



Happy computer

Low-level programming languages



Low-level programming languages

Low level languages are much closer to what the computer actually understands - machine code, which is written in binary.

They may use mnemonics (abbreviated words used instead of machine code binary).

An example is assembly language / code.

An assembler is needed to translate assembly language into machine code.

```
11
12      MOV AX,BCD
13      MOV BH,AH
14      MOV BL,AL
15
16      AND AH,0F0H
17      AND BH,0FH
18      AND AL,0F0H
19      AND BL,0FH
20
21      MOV CL,04H
22      ROL AH,CL
23
24      MOV CL,04H
25      ROL AL,CL
26
27      ADD AX,3030H
28      ADD BX,3030H
29
30      LEA SI,ASCII
31
32      MOV [SI],AH
33      INC SI
34
35      MOV [SI],BH
36      INC SI
37
38      MOV [SI],AL
39      INC SI
40
41      MOV [SI],BL
42      INC SI
43
44      LEA DX,ASCII
45      MOV AH,9
46      INT 21H
47
48      MOV AH,4CH
49      INT 21H
```

Low-level programming languages

Assembly Example

```
        .MODEL SMALL            ;select small model
        .DATA                   ;start data segment
DATA1   DW     2000H            ;define DATA1
DATA2   DW     3000H            ;define DATA2
        .CODE                   ;start code segment
        .STARTUP                ;start program
        LEA SI,DATA1            ;address DATA1 with SI
        MOV DI,OFFSET DATA2     ;address DATA2 with DI
        MOV BX,[SI]             ;exchange DAT1 with DATA2
        MOV CX,[DI]
        MOV [SI],CX
        MOV [DI],BX
        .EXIT
        END
```

Low-level programming languages

When a processor chip is made, it is designed to understand and execute a set of machine code instructions (OpCodes), unique to that chip - see Instruction Sets

One step up from machine code, is assembly code.

Each machine code instruction is given a mnemonic (name), so that it is easier for people to write the assembly code.

A mnemonic is an abbreviated 缩写 word used instead of the machine code binary.

Low-level programming languages

Compared to machine code (binary), programs are easier to write in Assembly Language, but still require lots of training.

Mnemonics can be used instead of 0s and 1s.
Example: `STO bx E402`

Assembly language requires an assembler to be converted / translated into machine code.

Low-level programming languages

Advantages

They can directly manipulate and use all features of the hardware - because they are written specific to a machine.

The programs are smaller and more memory efficient - the code does not take up very much primary memory space.

No need for a compiler or interpreter.

They are quicker to execute (run).

Low-level programming languages

Disadvantages

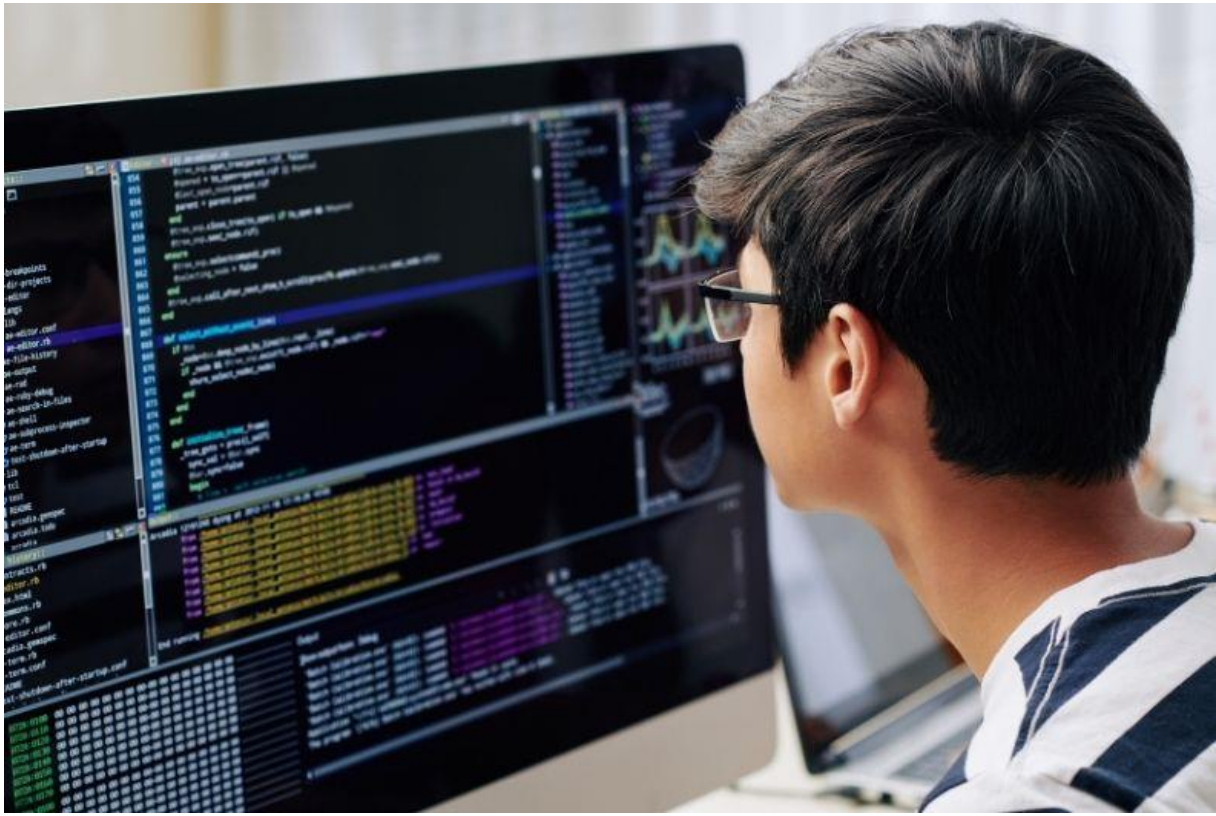
They are harder to read and understand than high-level languages.

It is more difficult to debug code (find errors).

```
EXAMPLE2: MOV   DPTR,#50H       ;init pointer to 0050H
          MOV   R7,#0           ;init count = 0
REPEAT:   MOVX  A,@DPTR         ;char = @pointer
          INC   DPTR            ;increment pointer
IF:       CJNE  A,#'0',$+3      ;if char >= '0' AND
          JC    UNTIL           ;
          CJNE  A,#'9'+1,$+3    ; char <= '9'
          JNC   UNTIL           ;
THEN:     INC   R7              ;then increment counter
UNTIL:    CJNE  A,#0,REPEAT     ;char is 00H
          MOV   A,R7            ;store count in acc
HERE:     SJMP  HERE
          END                   ;example 2
```

IDE - Integrated Development Environment

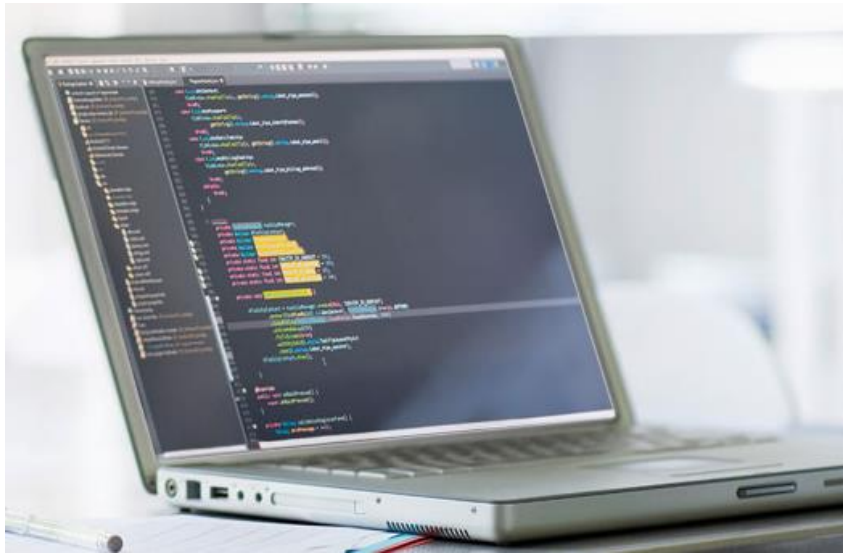
An Integrated Development Environment is used to write code and develop programs.



IDE - Integrated Development Environment

An Integrated Development Environment is used to write code and develop programs.

When you want to do a digital display, you use presentation software.



When you want to write an essay, you use word processing software.

When you want to calculate the accounts for a business, you use spreadsheet software.

IDE - Integrated Development Environment

An Integrated Development Environment is used to write code and develop programs.

IDEs include functions like:

- code editors
- run-time environment
- translators
- error diagnostics
- auto-completion
- auto-correction
- prettyprint