LANGUAGE TRANSLATORS

A. HIGH AND LOW LEVEL LANGUAGES

Low – Level Languages
High-Level Languages
Example: Assembly Language
Example: Pascal, Basic, Java

Characteristics of LOW Level Languages:

- They are machine oriented: an assembly language program written for one machine will not work on any other type of machine unless they happen to use the same processor chip.
- Each assembly language statement generally translates into one machine code instruction, therefore the program becomes long and time-consuming to create.

Example:

<table>
<thead>
<tr>
<th>Low Level Assembly Code</th>
<th>Machine Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>10100101 01110001</td>
<td>LDA &amp;71</td>
</tr>
<tr>
<td>01101001 00000001</td>
<td>ADD #&amp;01</td>
</tr>
<tr>
<td>10000101 01110001</td>
<td>STA &amp;71</td>
</tr>
</tbody>
</table>

Characteristics of HIGH Level Languages:

- They are not machine oriented: in theory they are portable, meaning that a program written for one machine will run on any other machine for which the appropriate compiler or interpreter is available.
- They are problem oriented: most high level languages have structures and facilities appropriate to a particular use or type of problem. For example, FORTRAN was developed for use in solving mathematical problems. Some languages, such as PASCAL were developed as general-purpose languages.
- Statements in high-level languages usually resemble English sentences or mathematical expressions and these languages tend to be easier to learn and understand than assembly language.
- Each statement in a high level language will be translated into several machine code instructions.

Example:

<table>
<thead>
<tr>
<th>C Code</th>
<th>Low Level Assembly Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>number:= number + 1;</td>
<td>10100101 01110001</td>
</tr>
<tr>
<td></td>
<td>01101001 00000001</td>
</tr>
<tr>
<td></td>
<td>10000101 01110001</td>
</tr>
</tbody>
</table>

B. GENERATIONS OF PROGRAMMING LANGUAGES

<table>
<thead>
<tr>
<th>Generation</th>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>4th</td>
<td>4GLs</td>
</tr>
<tr>
<td>3rd</td>
<td>High Level Languages</td>
</tr>
<tr>
<td>2nd</td>
<td>Low-level Languages</td>
</tr>
<tr>
<td>1st</td>
<td>Machine Code</td>
</tr>
</tbody>
</table>
1. **MACHINE LANGUAGE – 1ST GENERATION**

In the early days of computer programming all programs had to be written in machine code. For example a short (3 instruction) program might look like this:

```
0111 0001 0000 1111
1001 1101 1011 0001
1110 0001 0011 1110
```

Machine code has several significant disadvantages associated it:

1. It is not intuitively obvious what a machine code instruction does simply from its encoding, consequently it is very difficult to read and write machine code.
2. The writing of machine code is extremely time consuming and error prone.
3. Many different machine codes exist (one for each make and type of computer).

Machine code executes directly without translation since it is the actual pattern of 0s and 1s understood by the computer’s memory.

2. **ASSEMBLY LANGUAGE – 2ND GENERATION**

Assembler languages were initially developed to address the disadvantages associated with machine code programming. They used symbolic codes instead lists of binary instructions. Consequently programming became more "friendly". An example of assembly code is given below:

```
MOV AX 01
MOV BX 02
ADD AX BX
```

In assembler language each line of the program corresponds to one instruction in machine code. For a program written in assembler language to be executable it must be translated into machine code using a translating program called an **assembler**.

Although use of assembly languages offers some advantages there are still a number of significant disadvantages associated with their use:

1. Each model of computer has its own assembly language associated with it.
2. Assembler programming still requires great attention to detail and hence remains both time consuming and tedious.
3. Because of (2) the risk of program error is not significantly reduced.

Note that there are some computer applications, such as interfacing with peripherals, where assembler language is still a necessity.

3. **HIGH LEVEL LANGUAGES – 3RD GENERATION**

High Level languages are so-called because they are independent of the particular computer. They are not machine oriented.

4. **FOURTH GENERATION LANGUAGES (4GLs) – THE 4TH GENERATION**

4GLs are the most modern high level languages today. The user must concentrate more on what is to be done to solve the problem while the how part is entirely (or most of it) left in the hands of the 4GL.
PROGRAM TRANSLATORS

As we have seen in other chapters, a computer can only understand programs defined using machine code. Consequently a program written for example in a high level language such as Java cannot be run directly. To execute a computer program written in high or low level language, it must be first be translated.

There are 3 types of system software used for translating the code that a programmer writes into a form that the computer can execute (i.e. machine code). These are:

1. Assemblers
2. Compilers
3. Interpreters

**Source Code** is the code that is input to a translator.

**Executable code** is the code that is output from the translator.

1. **Assembler**

An Assembler converts an assembly program into machine code.

![Assembly Process Diagram]

2. **Compiler**

A Compiler is a program that translates a high level language into machine code. The Turbo Pascal compiler, for example, translates a program written in Pascal into machine code that can be run on a PC.

![Compiler Diagram]

**Advantages of a Compiler**

1. Fast in execution
2. The object/executable code produced by a compiler can be distributed or executed without having to have the compiler present.
3. The object program can be used whenever required without the need to of re-compilation.
Disadvantages of a Compiler

1. Debugging a program is much harder. Therefore not so good at finding errors
2. When an error is found, the whole program has to be re-compiled

3. Interpreter

An Interpreter is also a program that translates high-level source code into executable code. However the difference between a compiler and an interpreter is that **an interpreter translates one line at a time and then executes it**: no object code is produced, and so the program has to be interpreted each time it is to be run. If the program performs a section code 1000 times, then the section is translated into machine code 1000 times since each line is interpreted and then executed.

Advantages of an Interpreter

1. Good at locating errors in programs
2. Debugging is easier since the interpreter stops when it encounters an error.
3. If an error is deducted there is no need to retranslate the whole program

Disadvantages of an Interpreter

1. Rather slow
2. No object code is produced, so a translation has to be done every time the program is running.
3. For the program to run, the Interpreter must be present
Language Translators - Questions

The box below shows part of a high-level language computer program. Part A shows the program before it has been translated. Part B shows the program after it has been translated.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>While colour = ‘Red’ do</td>
<td>1010 1101</td>
</tr>
<tr>
<td>Writeln (Colour, make);</td>
<td>1011 1111</td>
</tr>
<tr>
<td></td>
<td>1111 1001</td>
</tr>
<tr>
<td></td>
<td>1101 1110</td>
</tr>
<tr>
<td></td>
<td>1101 1101</td>
</tr>
</tbody>
</table>

1. Compilers, interpreters and assemblers are all translation programs. Which one of the three programs would be necessary to translate program A into program B.

2. What general name is given to high-level language code, similar to A, before it is translated?

3. What general name is given to the code, similar to B, produced after translation?

4. State what type of software package would be used to enter coding A into a computer and store it on magnetic disk.

5. Interpreters and assemblers differ in the way they translate computer programs and the type of programs they translate. State two such differences.

6. Three types of software that may be used to convert source code to object code are Assemblers, Compilers and Interpreters.

   a) What is the difference between source code and object code?
   b) What is the difference between an Assembler and a Compiler?
   c) What is the difference between a Compiler and an Interpreter?
   d) Why do programmers prefer High Level Languages to Low Level Languages?
   e) Why do programmers still feel the need to use Low Level programming Languages?