

INTERNATIONAL DIPLOMA IN COMPUTING

INTERNATIONAL DIPLOMA IN BUSINESS COMPUTING

INFORMATICS COMPUTER DIPLOMA

HIGHER DIPLOMA IN BUSINESS & INFORMATION TECHNOLOGY

PROGRAM DESIGN (C1002/CS112)

ASSIGNMENT

TERM 2 2011

Student declaration:

*I declare that:*

* *I understand what is meant by plagiarism*
* *The implication of plagiarism has been explained to me by my institution*
* *This assignment is all my own work and I have acknowledged any use of the published and unpublished works of other people.*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Student’s signature:……………………………. | | | | | | Date:……………………… | | |
|  | | | | | |  | | |
| Total number of pages including this cover page | | | | | |  | | 8 |
|  | | |  | | |  | |  |
| Submission Date | |  | August 22, 2012 | | Due Date |  | | August 24, 2012 |
| Students’ ID | |  | 0521-0612-9667 | | Class Code |  | |  |
| Student’s Full Name | |  | Joeven A. Valenzuela | | |  | | |
| Lecturer’s Name | |  |  | Ms. Maychell Alcorin | |  | |  |
| OFFICIAL USE ONLY | | |  | | |  | | |
|  | |  | Marker’s comments | | |  | |  |
|  | |  |  | |  |  | |  |
| Marker’s name | Initial marks awarded | | | | | | /100 | |
|  | Penalty on late submission | | | | | |  | |
|  | Penalty for plagiarism | | | | | |  | |
|  | Final marks awarded | | | | | | /100 | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name of Student | : Joeven A.Valenzuela\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |  |  |
| Student ID | : 0521-0612-9667\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |  |  |
| Module Code | : C1002/CS112 | |  |  |  |  |
|  |  |  |  |  |  |  |
| Criteria |  | Base Mark | Graded Mark | Comments | |  |
|  |  |  |  |  |  |  |
| 1. Explain what you |  | 60 marks |  |  |  |  |
|  |  |  |  |  |  |
| understand by the term | |  |  |  |  |  |
| ‘Machine Language’. | |  |  |  |  |  |
| Write a short report (of | |  |  |  |  |  |
| 500 to 800 words), |  |  |  |  |  |  |
| making use of examples | |  |  |  |  |  |
| where relevant. |  |  |  |  |  |  |
|  | |  |  |  |  |  |
| 2. Describe at least two type | | 40 marks |  |  |  |  |
|  |  |  |  |  |  |
| of ‘Language Translator” | |  |  |  |  |  |
| that is widely used as | |  |  |  |  |  |
| they are before? Based on | |  |  |  |  |  |
| your understanding, write | |  |  |  |  |  |
| a short report (of 1000 to | |  |  |  |  |  |
| 1200 words), making use | |  |  |  |  |  |
| of examples where |  |  |  |  |  |  |
| relevant. |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Total |  | 100 marks |  |  |  |  |

**TABLE OF CONTENTS**

TABLE OF CONTENTS 3

ASSIGNMENT QUESTIONS 4

MACHINE LANGUAGE 5

COMPILER 6

INTERPRETER 7-8

REFERENCES 8

ASSIGNMENT QUESTIONS:

Regardless of what High-level programming language you use, you eventually need to

convert your program into machine language so that the computer can understand it. There are number of ways to do this such as:

• Compile the program

• Interpret the program

(a) Explain what you understand by the term ‘Machine Language’. Write a short report

(of 500 to 800 words), making use of examples where relevant. [40 marks]

(b) Describe at least two type of ‘Language Translator” that is widely used as they are

before? Based on your understanding, write a short report (of 1000 to 1200 words),

making use of examples where relevant. [60 marks]

1. **What is a Machine language?**

Machine language is a collection of binary numbers or bits that the computer flows and translates. It is the only terminology a computer is capable of knowing. There are 2 kinds of dialects which are used in computer systems. The first kind is known as high-level terminology. And the second kind is known as a low-level terminology. Advanced stage terminology indicates that it can be recognized by the human being. And a low-level terminology is a set of guidelines that a computer uses without any further translations. In this way, you can create a knowing for the computer dialects.

This device terminology is actually a known as set of guidelines and it is also known as "machine code”. In simple words, device terminology is terminology which can be quickly recognized by computer systems. It is known as the 1st generation of development terminology. It is the only terminology the computer straight acknowledges. Computers work with binary terminology and device terminology guidelines also use a series of binary numbers (1s and 0s) or a combination of letters and numbers that symbolizes binary numbers. So the computer can straight comprehend the system that is developed in device terminology. The binary numbers match to the on and off electrical states of a computer. It is a device reliant development terminology. Machine reliant means the system developed in one kind of device or computer could not be run on other kind computer or device. So programs developed in it terminology in one computer are not quickly portable to other computer systems. Another drawback of device terminology is that it is a very challenging terminology to comprehend and learn. If there is any problem in the system, written in device terminology, then it is very hard to find out the correct mistake.

Every system contains training. Guidelines can be used to read the next record in a file, move a prevent of information from one place to another, call a new window to the screen and so on. Before your computer can perform any system training, it must turn it into a binary value known as device terminology. An example of common device terminology training appears below

01011000011100000000000100000010

Device terminology training may look like a useless sequence of 0s and 1s, but it actually arranges pieces into categories that signify particular functions and storage places. The 32 bit training written here, for example goes information between two particular storage places. Similar instructions transfer information from storage to a sign-up and vice-versa, add or deduct principles in signs up, split and increase principles and so on. The set of miscoded machine-language instructions available to a computer is known as that pc's training set .Each pc foundation has its own device terminology. Machine dialects are the only dialects recognized by computer systems. While easily recognized by computer systems, device dialects are almost difficult for people to use because they include entirely of figures. Developers, therefore, use either a high-level development terminology or a set up terminology. A set up terminology contains the same instructions as a device terminology, but the instructions and factors have titles instead of being just figures.

2. Types of Language Translator

**Compilers**

## Since a components is capable of understanding only device stage guidelines, so it is necessary to turn the guidelines of a system published in advanced stage terminology to device guidelines before the system can be implemented by the computer. We have seen that set up dialects are use an assembler to perform this transformation process. In case of an advanced stage terminology, this job is carried out by a compiler. Though, a compiler is a converting system that transforms the guidelines of an advanced stage terminology into device terminology. A compiler is so known as “byte code” because it gathers a set of device terminology guidelines for every system training of an advanced stage terminology. A system published by a designer in an advanced stage terminology is known as resource system. After a designer creates a resource system it will be become device terminology by a computer, it is termed as an item system.

## A resource system containing one recognized by the computer will not be complied into an item system. The compiler will print out a suitable concept showing this, along with a list of known as mistake information which indicates the kind of mistakes dedicated. The mistake diagnostics is an important aid to the designer. A compiler, however, cannot identify sensible mistakes. It can only identify lexical mistake in the system, we often known as it “syntax error”. It cannot know one's objectives.

## It’s accountable for converting a resource value, which is published in a development terminology, into a target terminology. This is most commonly done to make an exe system. A compiler is mainly used for programs that turn a resource value into a set up terminology or device value, which are both a lower stage terminology than the resource value. There are different types of compilers, for example, a cross-compiler is one that will run a collected system on a computer whose operating system or CPU is different from the one on which the compiler runs; and a decompiler transforms from a low stage terminology to a higher one. It is a special system that procedures claims published in a particular development terminology and transforms them into device terminology, a "binary program" or "code," that a computer brand uses. It is a system published in a high-level terminology cannot be run directly on any computer. First, it has to be become device terminology.

## A compiler takes a high-level-language system and transforms it into an exe machine-language system. Once the interpretation is done, the machine-language system can be run any number of times, but of course it can only be run on one kind computer (since each kind computer has its own device language). If the system is to run on another kind computer it has to be re-translated, using a different compiler, into the appropriate device terminology.

## Interpreters

A demonstration is another kind of demonstration used for changing impressive stage 'languages' into program value. It requires one claim of an impressive stage 'languages' and changes it into a program training which is instantly implemented. Translations and performance different for each statement knowledgeable in the impressive stage terminology program. In other terms, demonstration changes one training, and the management program completes the producing program value, the next training is modified, and the management program perform it value training, and so on. This varies from a compiler which merely changes the whole resource program into items program and is not involved in its performance. In situation of a compiler, the whole resource program is become a comparative program terminology program. So, chronic choice is not necessary. However, in situation of a demonstration, no item value is stored for upcoming use because the demonstration and the performance procedure different. When training is used; it must once again be regarded and become program terminology.

This program can provide out the suggestions that are published in a development terminology. It can do this in a variety of ways: It can instantly perform the resource code; it can figure out the resource value into an efficient value (also known as impressive representation) and implement this instantly away; or it particularly provides out a precompiled value that has been created and stored by a compiler. Interpreters provide developers some benefits that compilers do not. Considered 'languages' are much simpler to understand than gathered 'languages', which is excellent for starting developers. A demonstration allows the designer knows instantly when and where issues are available in the code; gathered applications create the designer wait until the program is finish. Interpreters therefore can be much simpler to use and produce more immediate results; however the resource value of a regarded terminology cannot run without the demonstration. There is a substitute to obtaining a high-level terminology program. Instead of using a compiler, which changes the program all at once, you can use an interpreter, which changes it instruction-by-instruction, as needed. A demonstration is a program that features much like a CPU, with a kind of fetch-and-execute style. To be able to perform a program, the demonstration features in a style in which it continually goes one training from the program, chooses what is necessary to bring out that training, and then features the appropriate machine-language buys to do so

One use of interpreters is to perform high-level terminology applications. For example, the development terminology 'Basic' is usually implemented by an demonstration rather than a compiler. However, interpreters have another purpose: they can let you use a machine-language program appropriate for one kind pc on a absolutely different kind pc, thus doing a last time demonstration organization. However, there is a devious problem with interpreters - they can gradually performance significantly, perhaps by an aspect of three. Collected applications are normally quicker than regarded applications. Many high-level development 'languages' have the choice of using an demonstration instead of a compiler. Some of these 'languages' particularly use an demonstration. A demonstration features very in a different way from compilers and assemblers. It changes applications into machine-executable type whenever they are implemented. It researches and completes each range of resource value, in buy, without looking at the whole program. Instead of complicated a phase before program performance, an demonstration methods the program as it is being implemented.

References:

ANONYMOUS. 2012. Basics of Computer. [http://www.basicsofcomputer.com/translato rs .htm](http://www.basicsofcomputer.com/translato%20rs%20.htm). August 22, 2012

SWINDELLS,JOHN. What is Compiler, Interpreter, Assembler? http://www.blurtit.co m/q 769296.html . August 22, 2012

ANONYMOUS. 2012. What is machine language? [http://www.computerhope.com/ jarg on/m/machlang.htm](http://www.computerhope.com/%20jarg%20on/m/machlang.htm). August 22, 2012

ANONYMOUS. 2012. Research & Articles for Language Translation. http://www.book rags.com/research/language-translators-wcs/. August 22, 2012

ANONYMOUSE. 2012. What is machine language? – A word definition from Webopedia Computer Dictionary http://www.webopedia.com/TERM/ M/machine \_language.html. August 22, 2012

Copy right © 2012