The Morse Code Translator in Assembly Language

Hizkia William Eben
Faculty of Engineering
Universitas Indonesia
Depok, Indonesia
williamhizkia@gmail.com

Luthfi Rahman Hardy
Faculty of Engineering
Universitas Indonesia
Depok, Indonesia
luthfirh31@gmail.com

Marinus Martin
Faculty of Engineering
Universitas Indonesia
Depok, Indonesia
martin.marinus03@gmail.com

Muhammad Sulton Tauhid
Faculty of Engineering
Universitas Indonesia
Depok, Indonesia
msulton55@gmail.com

***Abstract*—Morse Code is one of the most recognized and used codes that designed to facilitate the communication from the sender to its receiver. Morse Code was used broadly in military services especially in delivering important messages in precarious situation. Originally, this code is delivered chiefly by using the telegram.**

This program was developed to help the users to either write or read the messages sent via Morse Code. Using this program, users can either decipher or encipher their messages to understand the code's usage. This project is programmed, compiled, and emulated in Assembly language, using emu8086 virtual emulator.

Keywords—Assembly Language; Encipher; Decipher; Morse Code; Messages; Programming.

I. INTRODUCTION

Morse Code was first invented by an American painter and inventor named Samuel Finley Breese Morse, OIC (April 27, 1791 – April 2, 1872). He invented this code in May 24, 1844 with the first message sent was "What hath God wrought," in his demonstration in Washington DC, with monetary help of federal support.

Morse got an innovation to create this code because when he was working as a painter at New York, the horse messenger delivered a letter from his father that his wife was dead because of worsening sickness. However, as he arrived to his house at New Haven, he was already been buried. Heartbroken for his unaware, of his wife's sickness and death, caused by how the message was slowly to be delivered, he decided to create a rapid long distance communication way.

For his invention, he was collaborating with an American machinist and inventor, Alfred Vail. Along with New York University Professor, Leonard Gale, in achieving the technological breakthrough of getting the telegraphic signal to travel long distances over wire.

In 1847, Morse received a patent for his invention and from there, The Morse telegraphic apparatus was officially adopted as the standard for European telegraphy in 1851. Before the invention of wireless signal and satellite communication, Morse code was mainly used to deliver the messages for far distances in quite quick duration.

A. Morse Code

Morse code is a code where it encoded the characters into a set of dots and dashes (dots or dash) that still used as a telecommunication device. This code was named with its founder's name, Samuel F.B. Morse.

In the early days, Morse code was used to send short text messages over long distance by means of the so-called electrical telegraph via (long) wires. The transmitting operator used a Morse key (switch) to turn the electric current on and off in the rhythm of the Morse codes.

One can use Morse code via auditory, by blowing a whistler, using oral sounds, or operating the telegraph, or via visual, by operating a strobe light (signal lamp) or flashlight. Morse code was case-insensitive and able to transmit all 26 English letters, several non-English letters, Arabic numerals, and procedural signals (prosing's).

Each Morse code symbol is stylized with variation of dots and dashes. There are rules.in how long does the dots, dashes, interval between letters, and interval between words. Even though in the real condition, for a rapid transmission of communications, these rules are going simplified. Here are the chart containing 26 English letters and Arabic numerals from 0 – 9 in Morse code.

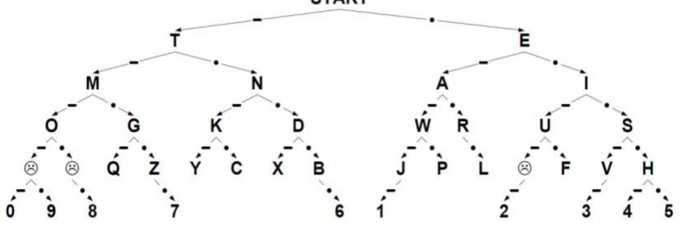
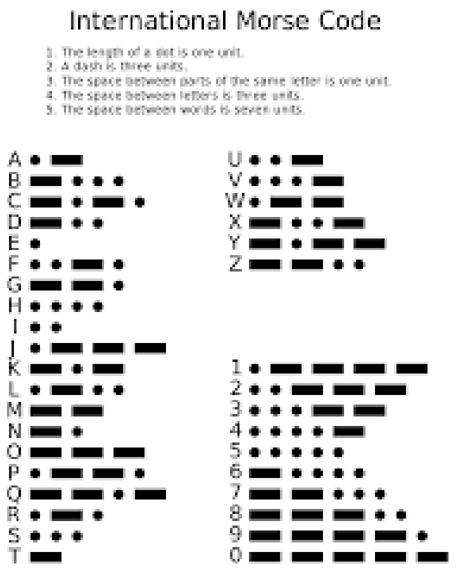


Fig.1 Morse Code Pattern Chart



International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	● ▬	U	● ● ▬
B	▬ ● ● ●	V	● ● ● ▬
C	▬ ● ▬ ●	W	● ▬ ▬
D	▬ ● ●	X	▬ ● ● ▬
E	●	Y	▬ ● ▬ ▬
F	● ● ▬ ●	Z	▬ ▬ ● ●
G	▬ ▬ ●		
H	● ● ● ●		
I	● ●		
J	● ▬ ▬ ▬		
K	▬ ● ▬	1	● ▬ ▬ ▬ ▬
L	● ▬ ● ●	2	● ● ▬ ▬ ▬
M	▬ ▬	3	● ● ● ▬ ▬
N	▬ ●	4	● ● ● ● ▬
O	▬ ▬ ▬	5	● ● ● ● ●
P	● ▬ ▬ ●	6	▬ ● ● ● ●
Q	▬ ▬ ● ▬	7	▬ ▬ ● ● ●
R	● ▬ ●	8	▬ ▬ ▬ ● ●
S	● ● ●	9	▬ ▬ ▬ ▬ ●
T	▬	0	▬ ▬ ▬ ▬ ▬

Fig.2 International Morse Code Chart

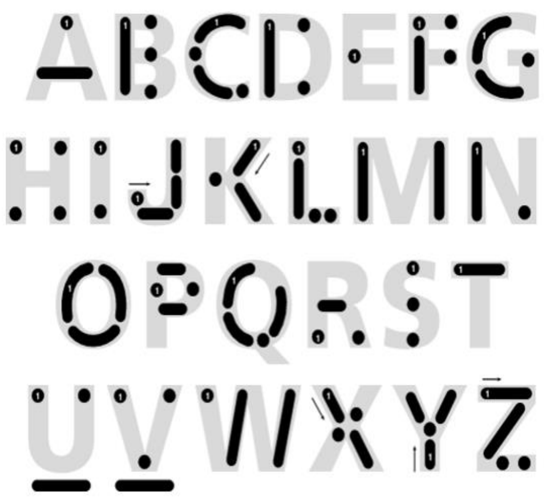


Fig.3 Morse Code Patterns for each character

B. *Assembly Language*

Assembly Language is the basic "computational" language used to program the computer by manipulation the memory and registers inside the microprocessor. Assembly is a low-level programming language that mimicking how the architecture's machine code instructions with the program's statements written in Assembly.

Assembly language is specific toward particular computer architecture or, sometimes, operating system. However, most of it can be used in any operating system since Assembly language is considered as the "closest" language in assembling how the computer's machine operates. Because of its simplicity and similarity to how the computer works, this language only need an assembler without interpreter or compiler like many high-level programming languages.

Assembly is usually written in one statement per machine instruction. Assembler directives, macros, procedures, and multiple symbolic labels of program and memory locations are often also supported. This source code can be run directly to the computer's microprocessor, or within the emulator installed in the computer.

II. Parts of the Program

A. *Main Menu*

After user opens the program, one will see the main menu part that provides four choices, which are:

1. Encode

   This is a choice where user wants to encode the string message into a Morse code.

2. Decode

   This is a choice where user wants to decode (translate) the Morse code into a string message

3. Show Morse Code Table

   This option will show the letter and its Morse code list from International Morse Code chart table.

4. Exit

User can select the preferred function one of them by inserting an integer input as previously provided. After user used the program, it will ask the user to go back to main menu and select another or same choice until user inserts input "4" (Exit).

To build this menu, the programmers called a procedure named CLEAR_SCREEN before inserting several interrupt functions, such as: MOV AH, 9H to show the main menu contents and MOV AH, 1H to wait for the user's input. In addition, some comparisons instruction are added to let the program's work jump into the preferred code section.

B. *Encode*

In the encoding section, user will be prompted to input a string message in uppercase and the program shows its encryption into its Morse code afterwards.

To build this section, the programmers used several procedures functions, such as: GET_INPUT (To get what user is inputting), CHECK_LETTER (to check the letter and find its appropriate Morse code), BEEPS (to show and sound dot ("dots") in Morse code), BEEPL (to show and sound dash ("dash") in Morse code).

The output produced by these program's instructions are the message in Morse code and how does it sound like.

C. *Decode*

In the decoding section, user will be prompted to input a set of Morse code characters in dots and dashes before the program will translate them into a readable message.

To build this section, the programmers used a special algorithm to read the pattern's input. First, the program slices the input when space is detected. Each time the program meets space ascii, then all the previous char will be moved to other arrays where the program can make a calculation for counting the said arrays. The algorithm is to count the total length of the array, then multiply by the content of its array. If it's dits, it will be multiplied by 1, if it's dahs, it will be multiplied by 2. Then for the next loop, the length will be decremented by 1. Hence, every alphabet will have its own special numbers to be compared with the inputs. If it matched, the program will interrupt outputs based on the matched characters,

The output produced by these program's instructions are a string of decrypted message.

III. Strength and Weakness

The program that we built and programmed will not free from the strength and weakness aspects. By knowing and understanding them, we can improve its work and debugging them from any errors or limitations in the future.

Here are several strengths and weaknesses we could find inside our program that will be described below.

A. Strengths

1. In option "Encode Message", the message that user inserted will be encrypted in two ways, which are in visual (its Morse code) and in aural ("dits" and "dahs").
2. Encode program will also give outputs with sound besides string outputs.
3. The program provides "Show Morse Code Table" option to the user who does not know how to write a Morse code.
4. Encode and decode supports characters deletion if the user feels like they input wrong characters.

B. Weaknesses/Known bugs

1. In option "Encode Message", user is unable to input the message in lowercase format (User only able to input it in uppercase).
2. In option "Decode Message", user is unable to input SPACE (" ") in Morse code (The message translated will contain no space).
3. In Decode, if the user tries to delete characters when there is no char left, then the user tried to type the Morse code to translate it, the output will be different than expected.
4. The program cannot encode or decode other than alphabets. Hence, numerical conversion is not possible.

IV. Work Divisions

The work divisions by its affiliates in Group 1 in creating, programming, creating Video explanation, and publishing this Assembly program are described below. The group members are:

1. Hizkia William Eben
   a) Brainstorming and Ideas Research (35%)
   b) Assembly Programming (90%)
   c) Report Writing (15%)
2. Luthfi Rahman Hardy
   a) Brainstorming and Ideas Research (15%)
   b) Assembly Programming (4%)
   c) Report Writing (85%)
3. Marinus Martin
   a) Brainstorming and Ideas Research (20%)
   b) Assembly Programming (6%)
4. Muhammad Sulton Tauhid
   a) Brainstorming and Ideas Research (20%)
   b) Video explanation (100%)

V. Conclusions

Assembly language can be applied into several useful programs, and one of them is this Morse Code Translator program. By understanding how Assembly language works in programming the computer.

In this paper, we presented a Morse Code Translator program as an implementation of Assembly programming we studied in class or in practicums. Although Morse code are not used as much as past times, by using this program, both users and programmers can study and understand the Morse code usage along with its contribution to message delivering in telecommunications.

References

[1] Crypto Museum, Visual symbolic representation of the morse code alphabet. © Copyright 1989, AG Reinhold, Cambridge, UK. Crypto Museum, November 2008.

[2] History of Morse, Nrich Math.org. Copyright © 1997 - 2019. University of Cambridge. All rights reserved.

[3] A. Michael Wood, Assembly Language: How To Learn To Code Assembly Today, April 4 2019, whoishostingthis.com

[4] The Editors of Encyclopaedia Britannica, "Morse Code | Invention, History, & Systems", Encyclopedia Britannica, 2019. [Online]. Available: https://www.britannica.com/topic/Morse-Code. [Accessed: 04- Apr-2019].

[5] M. Bellis, "Words on a Wire: A Brief Biography of Samuel Morse", ThoughtCo, 2019. [Online]. Available: https://www.thoughtco.com/biography-of-samuel-morse-1992165. [Accessed: 05- Apr- 2019].

[6] "Assembly Introduction", www.tutorialspoint.com, 2019. [Online]. Available: https://www.tutorialspoint.com/assembly_programming/assembly_introduction.htm. [Accessed: 04- Apr- 2019].

[7] The Editors of Encyclopaedia Britannica, "Assembly language | computer language", Encyclopedia Britannica, 2019. [Online]. Available: https://www.britannica.com/technology/assembly-language. [Accessed: 05- Apr- 2019].

[8] "IBM Knowledge Center", Ibm.com, 2014. [Online]. Available: https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.asma400/asmr102112.htm. [Accessed: 04- Apr- 2019]