**Project Report on**

**A Python Interpreter with Assembly Language 8086**

Course Code: **CSE232**

Course Title**: Microprocessor, Embedded Systems, and IoT Lab**

**Submitted by:**

**Anamika Hossain Lily** (213-15-4246)

**Md Shefat Al Mahmud** (213-15-4364)

**A A M Mustahid** (213-15-4251)

**Istiaq Nezoom Mollah** (213-15-4269)

Department of Computer Science Engineering

Daffodil International University

**Submitted to:**

**Mr. Golam Rabbany**

Lecturer

Department of Computer Science Engineering

Daffodil International University



**A Python Interpreter with Assembly Language 8086**

**Abstract –** The project presents the development of a Python interpreter implemented in the 8086 Assembly Language with limited features. Building an interpreter of high-level programming languages like Python is a very complex and time-consuming task. So, it is not possible to build a complete Python interpreter overnight. The project aims to build an interpreter that can do some basic operations like input-output, work with variables, basic arithmetic operations, etc. The project is implemented with the emu8086 emulator.

**Keywords-** 8086 Assembly Language, Python, Interpreter, Input/output, Variable handling, Arithmetic Operations, emu8086

**I. Introduction**

Python has become the most popular language in recent years.[5] Python has gained immense popularity in both the software development industry and the academic world due to its ease of use and extensive libraries. But Python is slower than other languages like C, C++, Java, etc.

Assembly Programming language is a low-level programming language[1] and Python is a high-level programming language. Python is an easier and more human-readable programming language than assembly language.

An interpreter is a computer program that executes program instructions written in one of the lots of high-level programming languages. The interpreter converts the high-level program into an intermediate language, which it then executes, or it can parse the high-level source code and then execute the commands directly, line by line or statement by statement.

Assembly language is a key component of computer programming that resides at the intersection of human-readable code and machine language. Programmers use assembly language to interface directly with a computer's hardware, utilizing a set of symbolic names and mnemonic codes to represent the CPU's instructions. These instructions are architecture-specific and may perform activities such as moving data between memory locations, executing arithmetic calculations, and regulating the flow of a program.

Building an interpreter of Python with assembly is easy at all. It needs some years to build a complete Python interpreter. In this project, we added limited features of Python due to a shortage of time. We will add more functionalities in the future.

**II. Background & Motivation**

Assembly code can be cryptic and challenging to read due to its use of mnemonic codes and memory addresses. It lacks the structure and readability of high-level languages. Python is known for its clear and readable syntax, emphasizing code indentation and a clean, human-readable style. This makes it easier to understand and maintain. Writing code in Assembly is typically time-consuming due to its low-level nature. Programming tasks that are quick in high-level languages can take significantly more time in Assembly. Python is renowned for its rapid development capabilities. You can write and test code quickly, which is beneficial for prototyping and development efficiency.

Assembly code can be highly optimized for performance since it operates close to the hardware. It is suitable for tasks where speed is critical. Python is slower than low-level languages like Assembly due to its abstraction and interpretation.

The motivation of the project is to make an interpreter that interprets Python code faster than existing interpreters. Because Assembly offers precise control over both time and space complexity but comes at the cost of development time and complexity.

**III. Implementation**

The project is implemented by 8086 Assembly language. We used the emu8086 (fig: 3.1) emulator to write and execute the program. Used registers like AX, BX, CX, DX, SI, DI, etc.

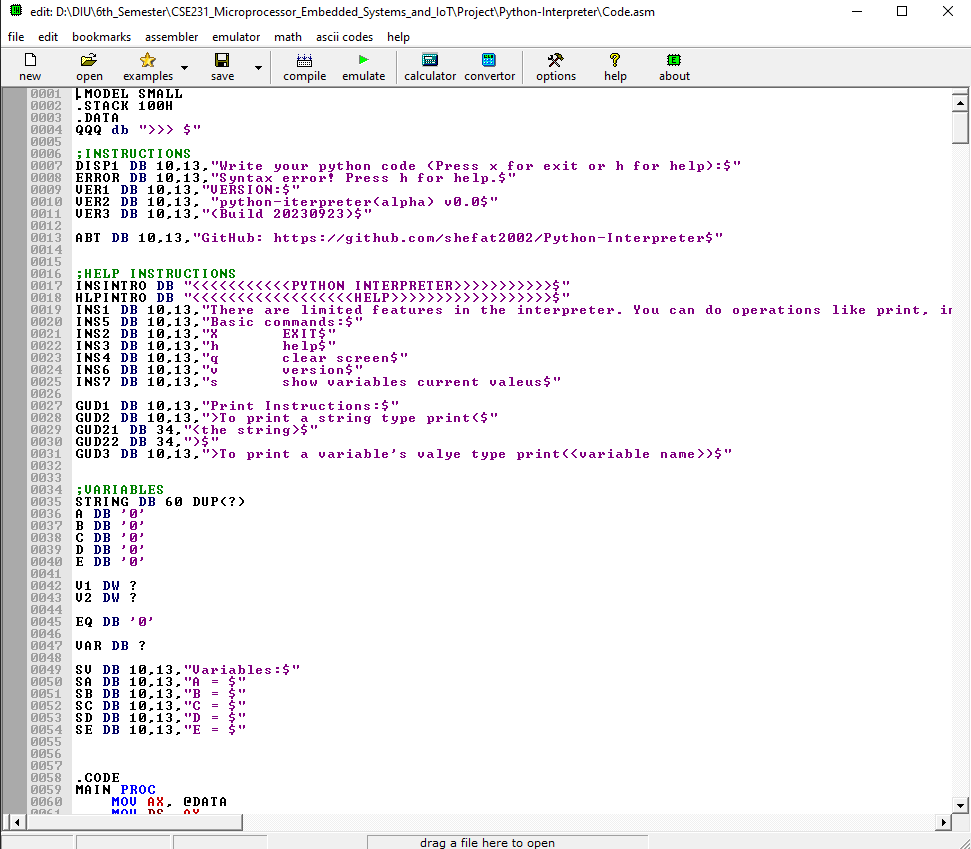


Fig:3.1

We added the following features:

**Input/Output:** Used SI register to input string from the users. If the command is to print something, the interpreter reads and gives the output(Fig:3.2). The interpreter can also return the output(Fig:3.3).



Fig: 3.2

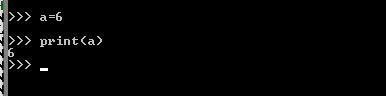


Fig: 3.3

**Variable:** There are limited variables. Users can take only 5 variables at a time which are a, b, c, d, and e. (Fig:3.4)

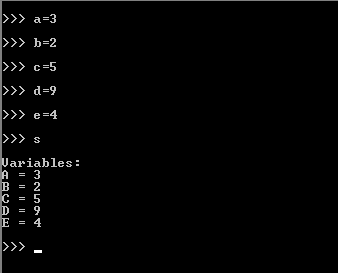


Fig: 3.4

**Arithmetic Operations**: Users can do arithmetic operations like summation, subtraction, multiplication, and division. For the executing arithmetic calculation, we used ADD, SUB, MUL, and DIV operations. (Fig:3.5)

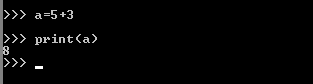


Fig: 3.5

**Error Handling:** The program can handle syntax errors.

**IV. Testing & Result**

The program was tested multiple times and fixed many bugs. The program can give appropriate output on basic operations of Python. But there are still some major weaknesses remain. Due to lack of time, our team was not able to fix all the issues.

**IV. Discussion**

The Development of the Python interpreter was a very complex and challenging task. It took more than 14 hours to complete. Due to a lack of knowledge of advanced assembly language topics, it makes more harder to implement and develop a complete project. The interpreter has lots of limitations. Because it is not possible to make an interpreter like Python in a day. Future work can focus on addressing these limitations and further optimizing the interpreter for improved performance and compatibility.

**V. References**

[1] Ytha Yu, Charles Marut-Assembly Language Programming Organization of the IBM PC (1992)

[2]T . L. Adams and R. E. Zimmerman. 1989. An analysis of 8086 instruction set usage in MS DOS programs. In Proceedings of the third international conference on Architectural support for programming languages and operating systems (ASPLOS III). Association for Computing Machinery, New York, NY, USA, 152–160. <https://doi.org/10.1145/70082.68197>

[3] M. Xu, "Teaching methods for “Assembly Language Programming” in Windows environment," 2011 IEEE International Symposium on IT in Medicine and Education, Guangzhou, China, 2011, pp. 156-159, doi: 10.1109/ITiME.2011.6130806.

[4] Qiang Zhang, Lei Xu, and Baowen Xu. 2022. RegCPython: A Register-based Python Interpreter for Better Performance. ACM Trans. Archit. Code Optim. 20, 1, Article 14 (March 2023), 25 pages. https://doi.org/10.1145/3568973

[5] Software TIOBE. 2022. TIOBE Index. Retrieved from https://www.tiobe.com/tiobe-index.