

8086 ASSEMBLY MONOALPHABETIC SUBSTITUTION ENCRYPTION SYSTEM

MINOR PROJECT REPORT

By

**SRIJONI CHOWDHURY (RA221002610210)
SAYAL SINGH (RA2211002610218)
MOHAMMAD BASITH KALAM RIZVI (RA2211026010219)**

Under the guidance of

Dr.R.A.Karthika

In partial fulfilment for the Course

of

21CSS201T– Computer Organization and Architecture

in the Department of Computational Intelligence



FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSS201T– Computer Organization and Architecture** entitled in "**Monoalphabetic Substitution Encryption System using 8086 Assembly Language**" is the bonafide work of **Srijoni Chowdhury(RA2211026010210)** , **Sayal Singh(RA2211026010218)** and **Rizvi(RA2211026010219)** who carried out the work under my supervision.

SIGNATURE

Dr.R.A.Karthika

COA– Course Faculty

Assistant Professor

Department of Computational Intelligence

SRM Institute of Science and Technology

Kattankulathur

SIGNATURE

Dr Annie Uthra R

Head of the Department

Professor

Department of Computational Intelligence

SRM Institute of Science and Technology

Kattankulathur

ABSTRACT

The X86 Assembly Monoalphabetic Substitution Encryption System is a low-level, hardware-dependent cryptographic software designed to secure data through the implementation of a monoalphabetic substitution cipher at the assembly language level on x86 architecture. This system aims to provide a basic yet illustrative example of encryption techniques, offering a fundamental understanding of how encryption works at the machine code level.

The system leverages the x86 assembly language to perform encryption operations, replacing each character in the plaintext with a corresponding character from a predefined substitution key. The substitution key is a mapping of characters from the plaintext alphabet to the ciphertext alphabet, ensuring that each character is replaced consistently. This system relies on the monoalphabetic substitution method, which makes it suitable for educational purposes and limited security applications.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal**, for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman**, for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. C. Malathy**, Professor, Department of Networking and Communication and Course Coordinator **Dr. R.A.Karthika**, Assistant Professor, Department Computational Intelligence for their constant encouragement and support.

We are highly thankful to our Course project Faculty **Dr.R.A.Karthika**, Assistant Professor, Department Computational Intelligence for her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **Head of the Department, Dr.R.Annie Uthra, Professor, Department of Computational Intelligence** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

CHAPTER NO	CONTENTS	PAGE NO
1	INTRODUCTION	10
	1.1 Objective	10
	1.2 Introduction	10
2	SOFTWARE and HARDWARE REQUIREMENT	11
3	CONCEPT/WORKING PRINCIPLE	11
4	APPROACH/METHODOLOGY/PROGRAM	12
5	FLOWCHART	20
6	EXPERIMENT RESULTS & ANALYSIS	21
7	CONCLUSION	24
8	REFERENCES	25

1.INTRODUCTION

1.1Objective

- **Low-Level Programming Skills:** It provides an opportunity to learn and practice assembly language programming, a valuable skill for understanding and working with computer hardware and embedded systems.
- **Cryptography Understanding:** Through the implementation of a monoalphabetic substitution cipher, the project helps users comprehend fundamental cryptographic principles. It illustrates the concept of character substitution, which is a core component of many encryption algorithms, and fosters an appreciation for the nuances of secure data communication.
- **Algorithm Development:** This program encourages users to develop, modify, or enhance the encryption system's algorithm and substitution key. This allows for experimentation and exploration of various aspects of cryptographic design and implementation, fostering creativity and problem-solving skills.
- **Source Code Exploration:** Users can explore the project's source code to gain insights into how characters are substituted between plaintext and ciphertext. This provides a practical example of how encryption algorithms are coded and implemented in low-level programming languages.
- **Security Awareness:** While the monoalphabetic substitution cipher is not suitable for secure, real-world applications, the project highlights its weaknesses and reinforces the importance of using stronger encryption techniques for sensitive data. It raises awareness about the limitations and vulnerabilities of simple ciphers.
- **Foundation for Further Study:** The project can serve as a foundation for further exploration of cryptography and assembly language programming. Users who are interested in more advanced cryptographic concepts or in-depth x86 assembly programming can build upon the knowledge and experience gained from this project.

1.2Introduction

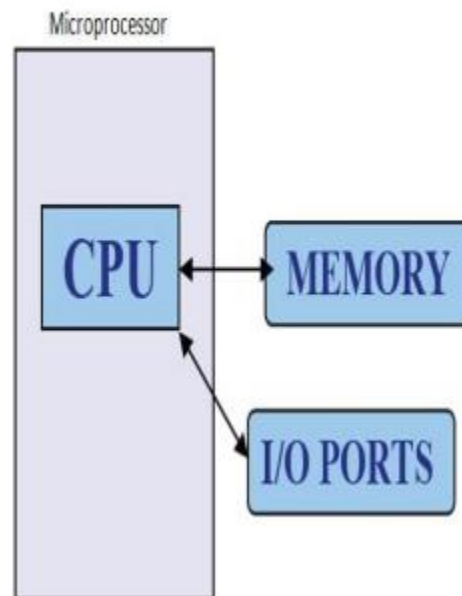
The "8086 Assembly Monoalphabetic Substitution Encryption System" is a fascinating exploration of the convergence of low-level programming and the realm of cryptography. This project delves into the intricate world of encryption, employing 8086 assembly language to implement a monoalphabetic substitution cipher—a fundamental encryption technique. The endeavor is designed to serve as an educational endeavor and a practical exercise, offering enthusiasts and aspiring programmers a unique opportunity to understand the inner workings of encryption at the machine code level.

2. SOFTWARE AND HARDWARE REQUIREMENT

- x86 emulator emu8086: EMU8086 is a popular and user-friendly integrated development environment (IDE) and emulator for the x86 architecture assembly language. It is primarily used for programming and testing assembly language code for Intel 8086 and 8088 microprocessors, which are early microprocessor models that were significant in the history of computing.
- Intel 8086 Assembly Language: Intel 8086 assembly language is a low-level programming language used for programming the Intel 8086 and Intel 8088 microprocessors. These processors were part of the x86 family and played a crucial role in the early history of personal computing. Intel 8086 assembly language is known for its close-to-hardware nature, allowing programmers to have precise control over the processor's operations. It's commonly used for educational purposes, for understanding computer architecture, and for tasks that require precise control over hardware. Despite its age, it remains a foundation for understanding the principles of assembly language and low-level programming.

3. CONCEPT/WORKING PRINCIPLE:

Microprocessor: is a CPU on a single chip. Third Generation (16-bit Microprocessor) were introduced in 1978. The 16-bit processors with a performance like minicomputers. Third Generation Microprocessors can be represented by Intel's 8086, Zilog Z800 and 80286.



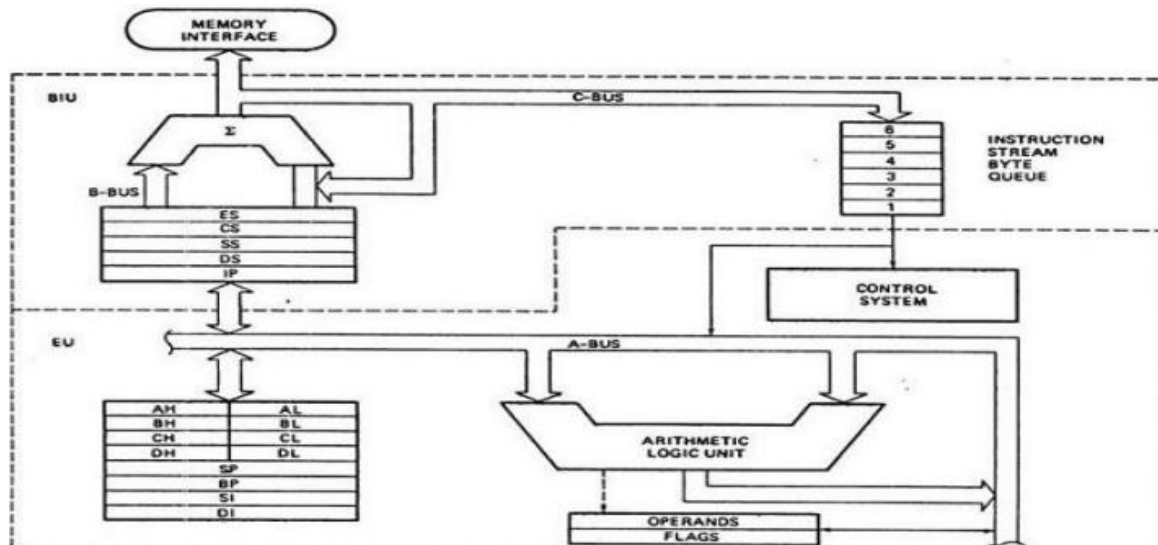
Diag1 - Structure of Microprocessor

8086 Microprocessor

The 8086 microprocessor is an 8-bit/16-bit microprocessor designed by Intel in the late 1970s. It is the first member of the x86 family of microprocessors, which includes many popular CPUs used in personal computers.

The architecture of the 8086 microprocessor is based on a complex instruction set computer (CISC) architecture, which means that it supports a wide range of instructions, many of which can perform multiple operations in a single instruction. The 8086 microprocessor has a 20-bit address bus, which can address up to 1 MB of memory, and a 16-bit data bus, which can transfer data between the microprocessor and memory or I/O devices.

The 8086 microprocessor has a segmented memory architecture, which means that memory is divided into segments that are addressed using both a segment registers and an offset. The segment register points to the start of a segment, while the offset specifies the location of a specific byte within the segment. This allows the 8086 microprocessors to access large amounts of memory, while still using a 16-bit data bus.



Diag2 : Architecture of 8086 Microprocessor

Memory is divided into various sections called segments.

- Code segment : where you store the program.
- Data segment : where the data is stored.
- Extra segment : mostly used for string operations.
- Stack segment : used to push/pop.

General purpose registers are used to store temporary data within the microprocessor.

AX, BX, CX, DX are the General-Purpose Registers. These are 16-bit registers. They are divided into 8-bit registers. These registers perform operations on 8-bit instructions.

Microprocessors also consist of Pointer Registers, Index Registers and Segment Registers.

Encryption & Decryption

Encryption: Encryption is a way of scrambling data so that only authorized parties can understand the information. In technical terms, it is the process of converting human-readable plaintext to incomprehensible text, also known as ciphertext. In simpler terms, encryption takes readable data and alters it so that it appears random. Encryption requires the use of a cryptographic key: a set of mathematical values that both the sender and the recipient of an encrypted message agree on.

Decryption: It is the process of transforming data that has been rendered unreadable through encryption back to its unencrypted form. In decryption, the system extracts and converts the garbled data and transforms it to texts and images that are easily understandable not only by the reader but also by the system. Decryption may be accomplished manually or automatically.

4. APPROACH

Encryption can be done in various ways. One of the simplest methods is monoalphabetic cipher. A monoalphabetic cipher is a substitution cipher where each letter of the plain text is replaced with another letter of the alphabet. It uses a fixed key which consists of the 26 letters of a “shuffled alphabet”.

This program implements this method to encrypt/decrypt a string of characters.

1. This program uses the following table as the reference for character substitution.

Plain text	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Cipher text	q	w	e	r	t	y	u	i	o	p	a	s	d	f	g	h	j	k	l	z	x	c	v	b	n	m

2. The program handles lowercase and uppercase characters.
3. This program omits all spaces in the result.

4.1 PROGRAM

ORG 0100H

JMP start.

newline EQU 0AH ; \n

cret EQU 0DH ; \r

bcksp EQU 08H ; \b

;

; Hard-coded string:

;

hardcoded_string DB hi there! This is an encrypted message', cret, newline, '\$' input_string DB 259

dup ('\$') ; Reserved area for input string (256 chars + \r \n \$)

; Messages and dialogues to be displayed in the UI:

;

message_welcome DB newline, 'Welcome to the monoalphabetic encryption system! ', cret, newline

DB 'Please choose how you wish to proceed:', cret, newline

DB '1- Enter string as input (max: 256 chars)', cret, newline

DB '2- Use hard-coded string', cret, newline, '\$'

message_using_hc DB '=====', cret, newline

DB 'USING YOUR HARDCODED STRING', cret, newline

DB '=====', cret, newline, '\$'

message_using_input DB '=====', cret, newline

DB 'Please enter your string below', cret, newline

DB '=====', cret, newline, '\$'

message_try_again DB cret, newline, 'Give it one more try? (y/n)', cret, newline, '\$'

message_press_key DB 'Press any key to exit...\$'

message_display_org DB cret, newline, 'Your original string: \$'

message_display_enc DB cret, 'Encrypted message: \$'

message_display_dec DB cret, 'Decrypted message: \$'

message_encrypting DB 'Encrypting...\$'

message_decrypting DB 'Decrypting...\$'

; Just for reference -----> 'abcdefghijklmnopqrstuvwxyz'

encryption_table_lower DB 97 dup (' '), 'qwertyuiopasdfghjklzxcvbnm'

decryption_table_lower DB 97 dup (' '), 'kxvmcnophqrszyijadlegwbuft'

; We leave 97(61H) blank spaces before the start of the table

; as the ASCII value of 'a' = 61H

encryption_table_upper DB 65 dup (' '), 'QWERTYUIOPASDFGHJKLZXCVBNM'

```

decryption_table_upper DB    65 dup (' '), 'KXVMCNOHPQRSZYIJADLEGWBUFT'
; We leave 65(41H) blank spaces before the start of the table
; as the ASCII value of 'A' = 41H

```

```

start:    LEA    DX, message_welcome
          MOV    AH, 09
          INT    21H
          MOV    AH, 0
          INT    16H
          CMP    AL, '2'    ; User chose to use the hardcoded string
          JE     use_hc
          CMP    AL, '1'    ; User chose to enter a string
          JNE    start
          CALL   get_input
          JMP    start_process

```

```

use_hc:   LEA    DX, message_using_hc
          MOV    AH, 09
          INT    21H
          LEA    SI, hardcoded_string

```

```

start_process:

```

```

; Display original string

```

```

          LEA    DX, message_display_org
          MOV    AH, 09      ; value of AH is adjusted as operation of int 21H depends on
its value

```

```

          INT    21H        ; at AH = 09, int 21H outputs string at DS:DX
          LEA    DX, SI
          MOV    AH, 09
          INT    21H

```

```

; Encrypt:

```

```

          LEA    DX, message_encrypting ; Display message
          MOV    AH, 09
          INT    21H
          MOV    AH, 1      ; value of AH is adjusted as operation of encrypt_decrypt
depends on its value

```

```

          CALL   encrypt_decrypt ; AH = 1 means monoalphabetic encryption, 0 means
monoalphabetic decryption, else do nothing

```

```

; Display result on the screen:

```

```

          LEA    DX, message_display_enc
          MOV    AH, 09      ; value of AH is adjusted as operation of int 21H depends on
its value

```

```

          INT    21H        ; at AH = 09, int 21H outputs string at DS:DX
          LEA    DX, SI
          MOV    AH, 09
          INT    21H

```

```

; Decrypt:

```

```

          LEA    DX, message_decrypting ; Display message
          MOV    AH, 09
          INT    21H
          MOV    AH, 0      ; AH = 0 means monoalphabetic decryption
          CALL   encrypt_decrypt

```

Display result on the screen:

```
    LEA    DX, message_display_dec
    MOV    AH, 09      ; value of AH is adjusted as operation of int 21H depends on
its value
    INT    21H        ; at AH = 09, int 21H outputs string at DS:DX
    LEA    DX, SI
    MOV    AH, 09
    INT    21H
```

;Display try again dialogue

```
try_again:    LEA    DX, message_try_again
    MOV    AH, 09
    INT    21H
    MOV    AH, 0
    INT    16H
    CMP    AL, 'y'
    JE     start
    CMP    AL, 'n'
    JNE    try_again
```

; Wait for any key...

```
    LEA    DX, message_press_key
    MOV    AH, 09
    INT    21H
    MOV    AH, 0      ; value of AH is adjusted as operation of int 16H depends on
its value
    INT    16H        ; at AH = 00, int 16H waits for keystroke from the keyboard (no
echo)
```

RET

; si - address of string to encrypt

encrypt_decrypt PROC NEAR

PUSH SI

next_char: MOV AL, [SI]

CMP AL, '\$' ; End of string?

JE end_of_string

CMP AL, ' ' ;<--- Beginning of space check

JNE cont ; Since this was a college assignment, One of my requirements
was to omit spaces in my result so

CALL omit_space ; you can just remove these 4 lines and the omit_space
subroutine if you do not wish to do that.

JMP next_char ;<--- End of space check

cont: CALL enc_dec_char

INC SI

JMP next_char

end_of_string: POP SI

RET

encrypt_decrypt ENDP

; Subroutine to send space to the end of the string (after '\$')

; You can delete this subroutine if you do not need it

;

```

omit_space      PROC  NEAR
                PUSH  SI          ; The reason I send the space after the '$'
                                ; is to handle several consecutive spaces without
omit_space_loop: MOV  AL, [SI+1]  ; entering an infinite loop as opposed to just swapping
                MOV   [SI+1], ' ' ; the ' ' character with the character after it
                MOV   [SI], AL
                INC    SI
                CMP    [SI-1], '$'
                JNE    omit_space_loop
                POP     SI
                RET
omit_space      ENDP

```

; Subroutine to convert character with the
; appropriate table (encrypt/decrypt)(uppercase/lowercase)

```

enc_dec_char     PROC  NEAR
                PUSH  BX
                CMP    AL, 'a'
                JB     check_upper_char
                CMP    AL, 'z'
                JA     skip_char
                CMP    AH, 1          ; AH = 1 means monoalphabetic encryption, since we are
working with lower case, we use encryption_table_lower
                JE     encrypt_lower_char
                CMP    AH, 0          ; AH = 0 means monoalphabetic decryption, since we are
working with lower case, we use decryption_table_lower
                JNE    skip_char
                LEA     BX, decryption_table_lower
                JMP     translate_char
encrypt_lower_char: LEA     BX, encryption_table_lower
                JMP     translate_char
check_upper_char:  CMP    AL, 'A'
                JB     skip_char
                CMP    AL, 'Z'
                JA     skip_char
                CMP    AH, 1          ; AH = 1 means monoalphabetic encryption, since we are
working with upper case, we use encryption_table_upper
                JE     encrypt_upper_char
                CMP    AH, 0          ; AH = 0 means monoalphabetic decryption, since we are
working with upper case, we use decryption_table_upper
                JNE    skip_char
                LEA     BX, decryption_table_upper
                JMP     translate_char
encrypt_upper_char: LEA     BX, encryption_table_upper
translate_char:    XLATB
                MOV     [SI], AL
skip_char:        POP     BX
                RET
enc_dec_char     ENDP

```

```

; Subroutine to take input string from the user
; The subroutine handles if the user presses backspace: delete char + inc CX
; The subroutine allows the user to enter a maximum of 256 chars
get_input      PROC  NEAR
                LEA    DX, message_using_input
                MOV     AH, 09
                INT     21H
                LEA     SI, input_string
                MOV     AH, 1
                MOV     CX, 255      ; To set a cap for string input to 256 chars
                JMP     input_loop
backspace_entered: INC     CX          ; Increment CX in case user presses backspace as a
character is deleted
input_loop:     INT     21H
                MOV     [SI], AL
                CMP     AL, bcksp
                JNE     cont_input
                CMP     SI, offset input_string
                JE      input_loop    ;If the string is empty just loop again without affecting SI and
without incrementing CX
                MOV     [SI], ''
                CALL    omit_space
                DEC     SI
                JMP     backspace_entered
cont_input:     INC     SI
                CMP     AL, cret
                JE      terminate_string
                LOOP    input_loop    ; LOOP instead of JMP to incorporate (CX != 0000H) as a
jump condition
terminate_string: MOV     [SI-1], cret
                MOV     [SI], newline
                MOV     [SI+1], '$'
                LEA     SI, input_string
                RET
get_input      ENDP

end

```

Screenshots of code

```

103      MOV     AH, 09
104      INT     21H
105
106      ; Decrypt:
107      LEA     DX, message_decrypting    ; Display message
108      MOV     AH, 09
109      INT     21H
110      MOV     AH, 0
111      CALL    encrypt_decrypt    ; AH = 0 means monoalphabetic decryption
112
113      ; Display result on the screen:
114      LEA     DX, message_display_dec
115      MOV     AH, 09    ; value of AH is adjusted as operation of int 21H depends on its value
116      INT     21H    ; at AH = 09, int 21H outputs string at DS:DX
117      LEA     DX, SI
118      MOV     AH, 09
119      INT     21H
120
121      ; Display try again dialogue
122      try_again: LEA     DX, message_try_again
123      MOV     AH, 09
124      INT     21H
125      MOV     AH, 0
126      INT     16H
127      CMP     AL, 'y'
128      JE      start
129      CMP     AL, 'n'
130      JNE     try_again
131
132      ; Wait for any key...
133      LEA     DX, message_press_key
134      MOV     AH, 09
135      INT     21H
136      MOV     AH, 0
137      INT     16H    ; value of AH is adjusted as operation of int 16H depends on its value
138      ; at AH = 00, int 16H waits for keystroke from the keyboard (no echo)
139      RET
140
141
142
143      ; si - address of string to encrypt
144      encrypt_decrypt PROC NEAR
145      PUSH    SI
146      MOV     AL, [SI]
147      CMP     AL, '$'    ; End of string?
148      JE      end_of_string
149
150      CMP     AL, ' '    ;<--- Beginning of space check
151      JNE     cont
152      CALL    omit_space
153      JMP     next_char
154      ;<--- End of space check
155      cont:   CALL    enc_dec_char
156      INC     SI
157
158      next_char:
159      JMP     encrypt_decrypt
160
161      end_of_string:
162      RET
163
164      omit_space PROC NEAR
165      MOV     SI, [DI]
166      MOV     DI, [DI+1]
167      MOV     AL, [SI]
168      MOV     AH, 0
169      INT     16H
170      CMP     AL, ' '
171      JNE     cont
172      MOV     SI, [DI]
173      MOV     DI, [DI+1]
174      MOV     AL, [SI]
175      MOV     AH, 0
176      INT     16H
177      CMP     AL, ' '
178      JNE     cont
179      MOV     SI, [DI]
180      MOV     DI, [DI+1]
181      MOV     AL, [SI]
182      MOV     AH, 0
183      INT     16H
184      CMP     AL, ' '
185      JNE     cont
186      MOV     SI, [DI]
187      MOV     DI, [DI+1]
188      MOV     AL, [SI]
189      MOV     AH, 0
190      INT     16H
191      CMP     AL, ' '
192      JNE     cont
193      MOV     SI, [DI]
194      MOV     DI, [DI+1]
195      MOV     AL, [SI]
196      MOV     AH, 0
197      INT     16H
198      CMP     AL, ' '
199      JNE     cont
200      MOV     SI, [DI]
201      MOV     DI, [DI+1]
202      MOV     AL, [SI]
203      MOV     AH, 0
204      INT     16H
205      CMP     AL, ' '
206      JNE     cont
207      MOV     SI, [DI]
208      MOV     DI, [DI+1]
209      MOV     AL, [SI]
210      MOV     AH, 0
211      INT     16H
212      CMP     AL, ' '
213      JNE     cont
214      MOV     SI, [DI]
215      MOV     DI, [DI+1]
216      MOV     AL, [SI]
217      MOV     AH, 0
218      INT     16H
219      CMP     AL, ' '
220      JNE     cont
221      MOV     SI, [DI]
222      MOV     DI, [DI+1]
223      MOV     AL, [SI]
224      MOV     AH, 0
225      INT     16H
226      CMP     AL, ' '
227      JNE     cont
228      MOV     SI, [DI]
229      MOV     DI, [DI+1]
230      MOV     AL, [SI]
231      MOV     AH, 0
232      INT     16H
233      CMP     AL, ' '
234      JNE     cont
235      MOV     SI, [DI]
236      MOV     DI, [DI+1]
237      MOV     AL, [SI]
238      MOV     AH, 0
239      INT     16H
240      CMP     AL, ' '
241      JNE     cont
242      MOV     SI, [DI]
243      MOV     DI, [DI+1]
244      MOV     AL, [SI]
245      MOV     AH, 0
246      INT     16H
247      CMP     AL, ' '
248      JNE     cont
249      MOV     SI, [DI]
250      MOV     DI, [DI+1]
251      MOV     AL, [SI]
252      MOV     AH, 0
253      INT     16H
254      CMP     AL, ' '
255      JNE     cont
256      MOV     SI, [DI]
257      MOV     DI, [DI+1]
258      MOV     AL, [SI]
259      MOV     AH, 0
260      INT     16H
261      CMP     AL, ' '
262      JNE     cont
263      MOV     SI, [DI]
264      MOV     DI, [DI+1]
265      MOV     AL, [SI]
266      MOV     AH, 0
267      INT     16H
268      CMP     AL, ' '
269      JNE     cont
270      MOV     SI, [DI]
271      MOV     DI, [DI+1]
272      MOV     AL, [SI]
273      MOV     AH, 0
274      INT     16H
275      CMP     AL, ' '
276      JNE     cont
277      MOV     SI, [DI]
278      MOV     DI, [DI+1]
279      MOV     AL, [SI]
280      MOV     AH, 0
281      INT     16H
282      CMP     AL, ' '
283      JNE     cont
284      MOV     SI, [DI]
285      MOV     DI, [DI+1]
286      MOV     AL, [SI]
287      MOV     AH, 0
288      INT     16H
289      CMP     AL, ' '
290      JNE     cont
291      MOV     SI, [DI]
292      MOV     DI, [DI+1]
293      MOV     AL, [SI]
294      MOV     AH, 0
295      INT     16H
296      CMP     AL, ' '
297      JNE     cont
298      MOV     SI, [DI]
299      MOV     DI, [DI+1]
300      MOV     AL, [SI]
301      MOV     AH, 0
302      INT     16H
303      CMP     AL, ' '
304      JNE     cont
305      MOV     SI, [DI]
306      MOV     DI, [DI+1]
307      MOV     AL, [SI]
308      MOV     AH, 0
309      INT     16H
310      CMP     AL, ' '
311      JNE     cont
312      MOV     SI, [DI]
313      MOV     DI, [DI+1]
314      MOV     AL, [SI]
315      MOV     AH, 0
316      INT     16H
317      CMP     AL, ' '
318      JNE     cont
319      MOV     SI, [DI]
320      MOV     DI, [DI+1]
321      MOV     AL, [SI]
322      MOV     AH, 0
323      INT     16H
324      CMP     AL, ' '
325      JNE     cont
326      MOV     SI, [DI]
327      MOV     DI, [DI+1]
328      MOV     AL, [SI]
329      MOV     AH, 0
330      INT     16H
331      CMP     AL, ' '
332      JNE     cont
333      MOV     SI, [DI]
334      MOV     DI, [DI+1]
335      MOV     AL, [SI]
336      MOV     AH, 0
337      INT     16H
338      CMP     AL, ' '
339      JNE     cont
340      MOV     SI, [DI]
341      MOV     DI, [DI+1]
342      MOV     AL, [SI]
343      MOV     AH, 0
344      INT     16H
345      CMP     AL, ' '
346      JNE     cont
347      MOV     SI, [DI]
348      MOV     DI, [DI+1]
349      MOV     AL, [SI]
350      MOV     AH, 0
351      INT     16H
352      CMP     AL, ' '
353      JNE     cont
354      MOV     SI, [DI]
355      MOV     DI, [DI+1]
356      MOV     AL, [SI]
357      MOV     AH, 0
358      INT     16H
359      CMP     AL, ' '
360      JNE     cont
361      MOV     SI, [DI]
362      MOV     DI, [DI+1]
363      MOV     AL, [SI]
364      MOV     AH, 0
365      INT     16H
366      CMP     AL, ' '
367      JNE     cont
368      MOV     SI, [DI]
369      MOV     DI, [DI+1]
370      MOV     AL, [SI]
371      MOV     AH, 0
372      INT     16H
373      CMP     AL, ' '
374      JNE     cont
375      MOV     SI, [DI]
376      MOV     DI, [DI+1]
377      MOV     AL, [SI]
378      MOV     AH, 0
379      INT     16H
380      CMP     AL, ' '
381      JNE     cont
382      MOV     SI, [DI]
383      MOV     DI, [DI+1]
384      MOV     AL, [SI]
385      MOV     AH, 0
386      INT     16H
387      CMP     AL, ' '
388      JNE     cont
389      MOV     SI, [DI]
390      MOV     DI, [DI+1]
391      MOV     AL, [SI]
392      MOV     AH, 0
393      INT     16H
394      CMP     AL, ' '
395      JNE     cont
396      MOV     SI, [DI]
397      MOV     DI, [DI+1]
398      MOV     AL, [SI]
399      MOV     AH, 0
400      INT     16H
401      CMP     AL, ' '
402      JNE     cont
403      MOV     SI, [DI]
404      MOV     DI, [DI+1]
405      MOV     AL, [SI]
406      MOV     AH, 0
407      INT     16H
408      CMP     AL, ' '
409      JNE     cont
410      MOV     SI, [DI]
411      MOV     DI, [DI+1]
412      MOV     AL, [SI]
413      MOV     AH, 0
414      INT     16H
415      CMP     AL, ' '
416      JNE     cont
417      MOV     SI, [DI]
418      MOV     DI, [DI+1]
419      MOV     AL, [SI]
420      MOV     AH, 0
421      INT     16H
422      CMP     AL, ' '
423      JNE     cont
424      MOV     SI, [DI]
425      MOV     DI, [DI+1]
426      MOV     AL, [SI]
427      MOV     AH, 0
428      INT     16H
429      CMP     AL, ' '
430      JNE     cont
431      MOV     SI, [DI]
432      MOV     DI, [DI+1]
433      MOV     AL, [SI]
434      MOV     AH, 0
435      INT     16H
436      CMP     AL, ' '
437      JNE     cont
438      MOV     SI, [DI]
439      MOV     DI, [DI+1]
440      MOV     AL, [SI]
441      MOV     AH, 0
442      INT     16H
443      CMP     AL, ' '
444      JNE     cont
445      MOV     SI, [DI]
446      MOV     DI, [DI+1]
447      MOV     AL, [SI]
448      MOV     AH, 0
449      INT     16H
450      CMP     AL, ' '
451      JNE     cont
452      MOV     SI, [DI]
453      MOV     DI, [DI+1]
454      MOV     AL, [SI]
455      MOV     AH, 0
456      INT     16H
457      CMP     AL, ' '
458      JNE     cont
459      MOV     SI, [DI]
460      MOV     DI, [DI+1]
461      MOV     AL, [SI]
462      MOV     AH, 0
463      INT     16H
464      CMP     AL, ' '
465      JNE     cont
466      MOV     SI, [DI]
467      MOV     DI, [DI+1]
468      MOV     AL, [SI]
469      MOV     AH, 0
470      INT     16H
471      CMP     AL, ' '
472      JNE     cont
473      MOV     SI, [DI]
474      MOV     DI, [DI+1]
475      MOV     AL, [SI]
476      MOV     AH, 0
477      INT     16H
478      CMP     AL, ' '
479      JNE     cont
480      MOV     SI, [DI]
481      MOV     DI, [DI+1]
482      MOV     AL, [SI]
483      MOV     AH, 0
484      INT     16H
485      CMP     AL, ' '
486      JNE     cont
487      MOV     SI, [DI]
488      MOV     DI, [DI+1]
489      MOV     AL, [SI]
490      MOV     AH, 0
491      INT     16H
492      CMP     AL, ' '
493      JNE     cont
494      MOV     SI, [DI]
495      MOV     DI, [DI+1]
496      MOV     AL, [SI]
497      MOV     AH, 0
498      INT     16H
499      CMP     AL, ' '
500      JNE     cont
501      MOV     SI, [DI]
502      MOV     DI, [DI+1]
503      MOV     AL, [SI]
504      MOV     AH, 0
505      INT     16H
506      CMP     AL, ' '
507      JNE     cont
508      MOV     SI, [DI]
509      MOV     DI, [DI+1]
510      MOV     AL, [SI]
511      MOV     AH, 0
512      INT     16H
513      CMP     AL, ' '
514      JNE     cont
515      MOV     SI, [DI]
516      MOV     DI, [DI+1]
517      MOV     AL, [SI]
518      MOV     AH, 0
519      INT     16H
520      CMP     AL, ' '
521      JNE     cont
522      MOV     SI, [DI]
523      MOV     DI, [DI+1]
524      MOV     AL, [SI]
525      MOV     AH, 0
526      INT     16H
527      CMP     AL, ' '
528      JNE     cont
529      MOV     SI, [DI]
530      MOV     DI, [DI+1]
531      MOV     AL, [SI]
532      MOV     AH, 0
533      INT     16H
534      CMP     AL, ' '
535      JNE     cont
536      MOV     SI, [DI]
537      MOV     DI, [DI+1]
538      MOV     AL, [SI]
539      MOV     AH, 0
540      INT     16H
541      CMP     AL, ' '
542      JNE     cont
543      MOV     SI, [DI]
544      MOV     DI, [DI+1]
545      MOV     AL, [SI]
546      MOV     AH, 0
547      INT     16H
548      CMP     AL, ' '
549      JNE     cont
550      MOV     SI, [DI]
551      MOV     DI, [DI+1]
552      MOV     AL, [SI]
553      MOV     AH, 0
554      INT     16H
555      CMP     AL, ' '
556      JNE     cont
557      MOV     SI, [DI]
558      MOV     DI, [DI+1]
559      MOV     AL, [SI]
560      MOV     AH, 0
561      INT     16H
562      CMP     AL, ' '
563      JNE     cont
564      MOV     SI, [DI]
565      MOV     DI, [DI+1]
566      MOV     AL, [SI]
567      MOV     AH, 0
568      INT     16H
569      CMP     AL, ' '
570      JNE     cont
571      MOV     SI, [DI]
572      MOV     DI, [DI+1]
573      MOV     AL, [SI]
574      MOV     AH, 0
575      INT     16H
576      CMP     AL, ' '
577      JNE     cont
578      MOV     SI, [DI]
579      MOV     DI, [DI+1]
580      MOV     AL, [SI]
581      MOV     AH, 0
582      INT     16H
583      CMP     AL, ' '
584      JNE     cont
585      MOV     SI, [DI]
586      MOV     DI, [DI+1]
587      MOV     AL, [SI]
588      MOV     AH, 0
589      INT     16H
590      CMP     AL, ' '
591      JNE     cont
592      MOV     SI, [DI]
593      MOV     DI, [DI+1]
594      MOV     AL, [SI]
595      MOV     AH, 0
596      INT     16H
597      CMP     AL, ' '
598      JNE     cont
599      MOV     SI, [DI]
600      MOV     DI, [DI+1]
601      MOV     AL, [SI]
602      MOV     AH, 0
603      INT     16H
604      CMP     AL, ' '
605      JNE     cont
606      MOV     SI, [DI]
607      MOV     DI, [DI+1]
608      MOV     AL, [SI]
609      MOV     AH, 0
610      INT     16H
611      CMP     AL, ' '
612      JNE     cont
613      MOV     SI, [DI]
614      MOV     DI, [DI+1]
615      MOV     AL, [SI]
616      MOV     AH, 0
617      INT     16H
618      CMP     AL, ' '
619      JNE     cont
620      MOV     SI, [DI]
621      MOV     DI, [DI+1]
622      MOV     AL, [SI]
623      MOV     AH, 0
624      INT     16H
625      CMP     AL, ' '
626      JNE     cont
627      MOV     SI, [DI]
628      MOV     DI, [DI+1]
629      MOV     AL, [SI]
630      MOV     AH, 0
631      INT     16H
632      CMP     AL, ' '
633      JNE     cont
634      MOV     SI, [DI]
635      MOV     DI, [DI+1]
636      MOV     AL, [SI]
637      MOV     AH, 0
638      INT     16H
639      CMP     AL, ' '
640      JNE     cont
641      MOV     SI, [DI]
642      MOV     DI, [DI+1]
643      MOV     AL, [SI]
644      MOV     AH, 0
645      INT     16H
646      CMP     AL, ' '
647      JNE     cont
648      MOV     SI, [DI]
649      MOV     DI, [DI+1]
650      MOV     AL, [SI]
651      MOV     AH, 0
652      INT     16H
653      CMP     AL, ' '
654      JNE     cont
655      MOV     SI, [DI]
656      MOV     DI, [DI+1]
657      MOV     AL, [SI]
658      MOV     AH, 0
659      INT     16H
660      CMP     AL, ' '
661      JNE     cont
662      MOV     SI, [DI]
663      MOV     DI, [DI+1]
664      MOV     AL, [SI]
665      MOV     AH, 0
666      INT     16H
667      CMP     AL, ' '
668      JNE     cont
669      MOV     SI, [DI]
670      MOV     DI, [DI+1]
671      MOV     AL, [SI]
672      MOV     AH, 0
673      INT     16H
674      CMP     AL, ' '
675      JNE     cont
676      MOV     SI, [DI]
677      MOV     DI, [DI+1]
678      MOV     AL, [SI]
679      MOV     AH, 0
680      INT     16H
681      CMP     AL, ' '
682      JNE     cont
683      MOV     SI, [DI]
684      MOV     DI, [DI+1]
685      MOV     AL, [SI]
686      MOV     AH, 0
687      INT     16H
688      CMP     AL, ' '
689      JNE     cont
690      MOV     SI, [DI]
691      MOV     DI, [DI+1]
692      MOV     AL, [SI]
693      MOV     AH, 0
694      INT     16H
695      CMP     AL, ' '
696      JNE     cont
697      MOV     SI, [DI]
698      MOV     DI, [DI+1]
699      MOV     AL, [SI]
700      MOV     AH, 0
701      INT     16H
702      CMP     AL, ' '
703      JNE     cont
704      MOV     SI, [DI]
705      MOV     DI, [DI+1]
706      MOV     AL, [SI]
707      MOV     AH, 0
708      INT     16H
709      CMP     AL, ' '
710      JNE     cont
711      MOV     SI, [DI]
712      MOV     DI, [DI+1]
713      MOV     AL, [SI]
714      MOV     AH, 0
715      INT     16H
716      CMP     AL, ' '
717      JNE     cont
718      MOV     SI, [DI]
719      MOV     DI, [DI+1]
720      MOV     AL, [SI]
721      MOV     AH, 0
722      INT     16H
723      CMP     AL, ' '
724      JNE     cont
725      MOV     SI, [DI]
726      MOV     DI, [DI+1]
727      MOV     AL, [SI]
728      MOV     AH, 0
729      INT     16H
730      CMP     AL, ' '
731      JNE     cont
732      MOV     SI, [DI]
733      MOV     DI, [DI+1]
734      MOV     AL, [SI]
735      MOV     AH, 0
736      INT     16H
737      CMP     AL, ' '
738      JNE     cont
739      MOV     SI, [DI]
740      MOV     DI, [DI+1]
741      MOV     AL, [SI]
742      MOV     AH, 0
743      INT     16H
744      CMP     AL, ' '
745      JNE     cont
746      MOV     SI, [DI]
747      MOV     DI, [DI+1]
748      MOV     AL, [SI]
749      MOV     AH, 0
750      INT     16H
751      CMP     AL, ' '
752      JNE     cont
753      MOV     SI, [DI]
754      MOV     DI, [DI+1]
755      MOV     AL, [SI]
756      MOV     AH, 0
757      INT     16H
758      CMP     AL, ' '
759      JNE     cont
760      MOV     SI, [DI]
761      MOV     DI, [DI+1]
762      MOV     AL, [SI]
763      MOV     AH, 0
764      INT     16H
765      CMP     AL, ' '
766      JNE     cont
767      MOV     SI, [DI]
768      MOV     DI, [DI+1]
769      MOV     AL, [SI]
770      MOV     AH, 0
771      INT     16H
772      CMP     AL, ' '
773      JNE     cont
774      MOV     SI, [DI]
775      MOV     DI, [DI+1]
776      MOV     AL, [SI]
777      MOV     AH, 0
778      INT     16H
779      CMP     AL, ' '
780      JNE     cont
781      MOV     SI, [DI]
782      MOV     DI, [DI+1]
783      MOV     AL, [SI]
784      MOV     AH, 0
785      INT     16H
786      CMP     AL, ' '
787      JNE     cont
788      MOV     SI, [DI]
789      MOV     DI, [DI+1]
790      MOV     AL, [SI]
791      MOV     AH, 0
792      INT     16H
793      CMP     AL, ' '
794      JNE     cont
795      MOV     SI, [DI]
796      MOV     DI, [DI+1]
797      MOV     AL, [SI]
798      MOV     AH, 0
799      INT     16H
800      CMP     AL, ' '
801      JNE     cont
802      MOV     SI, [DI]
803      MOV     DI, [DI+1]
804      MOV     AL, [SI]
805      MOV     AH, 0
806      INT     16H
807      CMP     AL, ' '
808      JNE     cont
809      MOV     SI, [DI]
810      MOV     DI, [DI+1]
811      MOV     AL, [SI]
812      MOV     AH, 0
813      INT     16H
814      CMP     AL, ' '
815      JNE     cont
816      MOV     SI, [DI]
817      MOV     DI, [DI+1]
818      MOV     AL, [SI]
819      MOV     AH, 0
820      INT     16H
821      CMP     AL, ' '
822      JNE     cont
823      MOV     SI, [DI]
824      MOV     DI, [DI+1]
825      MOV     AL, [SI]
826      MOV     AH, 0
827      INT     16H
828      CMP     AL, ' '
829      JNE     cont
830      MOV     SI, [DI]
831      MOV     DI, [DI+1]
832      MOV     AL, [SI]
833      MOV     AH, 0
834      INT     16H
835      CMP     AL, ' '
836      JNE     cont
837      MOV     SI, [DI]
838      MOV     DI, [DI+1]
839      MOV     AL, [SI]
840      MOV     AH, 0
841      INT     16H
842      CMP     AL, ' '
843      JNE     cont
844      MOV     SI, [DI]
845      MOV     DI, [DI+1]
846      MOV     AL, [SI]
847      MOV     AH, 0
848      INT     16H
849      CMP     AL, ' '
850      JNE     cont
851      MOV     SI, [DI]
852      MOV     DI, [DI+1]
853      MOV     AL, [SI]
854      MOV     AH, 0
855      INT     16H
856      CMP     AL, ' '
857      JNE     cont
858      MOV     SI, [DI]
859      MOV     DI, [DI+1]
860      MOV     AL, [SI]
861      MOV     AH, 0
862      INT     16H
863      CMP     AL, ' '
864      JNE     cont
865      MOV     SI, [DI]
866      MOV     DI, [DI+1]
867      MOV     AL, [SI]
868      MOV     AH, 0
869      INT     16H
870      CMP     AL, ' '
871      JNE     cont
872      MOV     SI, [DI]
873      MOV     DI, [DI+1]
874      MOV     AL, [SI]
875      MOV     AH, 0
876      INT     16H
877      CMP     AL, ' '
878      JNE     cont
879      MOV     SI, [DI]
880      MOV     DI, [DI+1]
881      MOV     AL, [SI]
882      MOV     AH, 0
883      INT     16H
884      CMP     AL, ' '
885      JNE     cont
886      MOV     SI, [DI]
887      MOV     DI, [DI+1]
888      MOV     AL, [SI]
889      MOV     AH, 0
890      INT     16H
891      CMP     AL, ' '
892      JNE     cont
893      MOV     SI, [DI]
894      MOV     DI, [DI+1]
895      MOV     AL, [SI]
896      MOV     AH, 0
897      INT     16H
898      CMP     AL, ' '
899      JNE     cont
900      MOV     SI, [DI]
901      MOV     DI, [DI+1]
902      MOV     AL, [SI]
903      MOV     AH, 0
904      INT     16H
905      CMP     AL, ' '
906      JNE     cont
907      MOV     SI, [DI]
908      MOV     DI, [DI+1]
909      MOV     AL, [SI]
910      MOV     AH, 0
911      INT     16H
912      CMP     AL, ' '
913      JNE     cont
914      MOV     SI, [DI]
915      MOV     DI, [DI+1]
916      MOV     AL, [SI]
917      MOV     AH, 0
918      INT     16H
919      CMP     AL, ' '
920      JNE     cont
921      MOV     SI, [DI]
922      MOV     DI, [DI+1]
923      MOV     AL, [SI]
924      MOV     AH, 0
925      INT     16H
926      CMP     AL, ' '
927      JNE     cont
928      MOV     SI, [DI]
929      MOV     DI, [DI+1]
930      MOV     AL, [SI]
931      MOV     AH, 0
932      INT     16H
933      CMP     AL, ' '
934      JNE     cont
935      MOV     SI, [DI]
936      MOV     DI, [DI+1]
937      MOV     AL, [SI]
938      MOV     AH, 0
939      INT     16H
940      CMP     AL, ' '
941      JNE     cont
942      MOV     SI, [DI]
943      MOV     DI, [DI+1]
944      MOV     AL, [SI]
945      MOV     AH, 0
946      INT     16H
947      CMP     AL, ' '
948      JNE     cont
949      MOV     SI, [DI]
950      MOV     DI, [DI+1]
951      MOV     AL, [SI]
952      MOV     AH, 0
953      INT     16H
954      CMP     AL, ' '
955      JNE     cont
956      MOV     SI, [DI]
957      MOV     DI, [DI+1]
958      MOV     AL, [SI]
959      MOV     AH, 0
960      INT     16H
961      CMP     AL, ' '
962      JNE     cont
963      MOV     SI, [DI]
964      MOV     DI, [DI+1]
965      MOV     AL, [SI]
966      MOV     AH, 0
967      INT     16H
968      CMP     AL, ' '
969      JNE     cont
970      MOV     SI, [DI]
971      MOV     DI, [DI+1]
972      MOV     AL, [SI]
973      MOV     AH, 0
974      INT     16H
975      CMP     AL, ' '
976      JNE     cont
977      MOV     SI, [DI]
978      MOV     DI, [DI+1]
979      MOV     AL, [SI]
980      MOV     AH, 0
981      INT     16H
982      CMP     AL, ' '
983      JNE     cont
984      MOV     SI, [DI]
985      MOV     DI, [DI+1]
986      MOV     AL, [SI]
987      MOV     AH, 0
988      INT     16H
989      CMP     AL, ' '
990      JNE     cont
991      MOV     SI, [DI]
992      MOV     DI, [DI+1]
993      MOV     AL, [SI]
994      MOV     AH, 0
995      INT     16H
996      CMP     AL, ' '
997      JNE     cont
998      MOV     SI, [DI]
999      MOV     DI, [DI+1]
1000     MOV     AL, [SI]
1001     MOV     AH, 0
1002     INT     16H
1003     CMP     AL, ' '
1004     JNE     cont
1005     MOV     SI, [DI]
1006     MOV     DI, [DI+1]
1007     MOV     AL, [SI]
1008     MOV     AH, 0
1009     INT     16H
1010     CMP     AL, ' '
1011     JNE     cont
1012     MOV     SI, [DI]
1013     MOV     DI, [DI+1]
1014     MOV     AL, [SI]
1015     MOV     AH, 0
1016     INT     16H
1017     CMP     AL, ' '
1018     JNE     cont
1019     MOV     SI, [DI]
1020     MOV     DI, [DI+1]
1021     MOV     AL, [SI]
1022     MOV     AH, 0
1023     INT     16H
1024     CMP     AL, ' '
1025     JNE     cont
1026     MOV     SI, [DI]
1027     MOV     DI, [DI+1]
1028     MOV     AL, [SI]
1029     MOV     AH, 0
1030     INT     16H
1031     CMP     AL, ' '
1032     JNE     cont
1033     MOV     SI, [DI]
1034     MOV     DI, [DI+1]
1035     MOV     AL, [SI]
1036     MOV     AH, 0
1037     INT     16H
1038     CMP     AL, ' '
1039     JNE     cont
1040     MOV     SI, [DI]
1041     MOV     DI, [DI+1]
1042     MOV     AL, [SI]
1043     MOV     AH, 0
1044     INT     16H
1045     CMP     AL, ' '
1046     JNE     cont
1047     MOV     SI, [DI]
1048     MOV     DI, [DI+1]
1049     MOV     AL, [SI]
1050     MOV     AH, 0
1051     INT     16H
1052     CMP     AL, ' '
1053     JNE     cont
1054     MOV     SI, [DI]
1055     MOV     DI, [DI+1]
1056     MOV     AL, [SI]
1057     MOV     AH, 0
1058     INT     16H
1059     CMP     AL, ' '
1060     JNE     cont
1061     MOV     SI, [DI]
1062     MOV     DI, [DI+1]
1063     MOV     AL, [SI]
1064     MOV     AH, 0
1065     INT     16H
1066     CMP     AL, ' '
1067     JNE     cont
1068     MOV     SI, [DI]
1069     MOV     DI, [DI+1]
1070     MOV     AL, [SI]
1071     MOV     AH, 0
1072     INT     16H
1073     CMP     AL, ' '
1074     JNE     cont
1075     MOV     SI, [DI]
1076     MOV     DI, [DI+1]
1077     MOV     AL, [SI]
1078     MOV     AH, 0
1079     INT     16H
1080     CMP     AL, ' '
1081     JNE     cont
1082     MOV     SI, [DI]
1083     MOV     DI, [DI+1]
1084     MOV     AL, [SI]
1085     MOV     AH, 0
1086     INT     16H
1087     CMP     AL, ' '
1088     JNE     cont
1089     MOV     SI, [DI]
1090     MOV     DI, [DI+1]
1091     MOV     AL, [SI]
1092     MOV     AH, 0
1093     INT     16H
1094     CMP     AL, ' '
1095     JNE     cont
1096     MOV     SI, [DI]
1097     MOV     DI, [DI+1]
1098     MOV     AL, [SI]
1099     MOV     AH, 0
1100     INT     16H
1101     CMP     AL, ' '
1102     JNE     cont
1103     MOV     SI, [DI]
1104     MOV     DI, [DI+1]
1105     MOV     AL, [SI]
1106     MOV     AH, 0
1107     INT     16H
1108     CMP     AL, ' '
1109     JNE     cont
1110     MOV     SI, [DI]
1111     MOV     DI, [DI+1]
1112     MOV     AL, [SI]
1113     MOV     AH, 0
1114     INT     16H
1115     CMP     AL, ' '
1116     JNE     cont
1117     MOV     SI, [DI]
1118     MOV     DI, [DI+1]
1119     MOV     AL, [SI]
1120     MOV     AH, 0
1121     INT     16H
1122     CMP     AL, ' '
1123     JNE     cont
1124     MOV     SI, [DI]
1125     MOV     DI, [DI+1]
1126     MOV     AL, [SI]
1127     MOV     AH, 0
1128     INT     16H
1129     CMP     AL, ' '
1130     JNE     cont
1131     MOV     SI, [DI]
1132     MOV     DI, [DI+1]
1133     MOV     AL, [SI]
1134     MOV     AH, 0
1135     INT     16H
1136     CMP     AL, ' '
1137     JNE     cont
1138     MOV     SI, [DI]
1139     MOV     DI, [DI+1]
1140     MOV     AL, [SI]
1141     MOV     AH, 0
1142     INT     16H
1143     CMP     AL, ' '
1144     JNE     cont
1145     MOV     SI, [DI]
1146     MOV     DI, [DI+1]
1147     MOV     AL, [SI]
1148     MOV     AH, 0
1149     INT     16H
1150     CMP     AL, ' '
1151     JNE     cont
1152     MOV     SI, [DI]
1153     MOV     DI, [DI+1]
1154     MOV     AL, [SI]
1155     MOV     AH, 0
1156     INT     16H
1157     CMP     AL, ' '
1158     JNE     cont
1159     MOV     SI
```

```

edit: C:\emu8086\MySource\COA2.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
157 end_of_string: JMP next_char
158 POP SI
159 RET
160 encrypt_decrypt ENDP
161
162
163
164 ; Subroutine to send space to the end of the string (after '$')
165 ; You can delete this subroutine if you do not need it
166
167
168 omit_space PROC NEAR
169 PUSH SI ; The reason I send the space after the '$'
170 MOV SI, [SI+1] ; is to handle several consecutive spaces without
171 MOV [SI+1], ' ' ; entering an infinite loop as opposed to just swapping
172 INC SI ; the ' ' character with the character after it
173 INC SI
174 CMP [SI-1], '$'
175 JNE omit_space_loop
176 POP SI
177 RET
178 omit_space ENDP
179
180
181
182
183 ; Subroutine to convert character with the
184 ; appropriate table (encrypt/decrypt)(uppercase/lowercase)
185 enc_dec_char PROC NEAR
186 PUSH BX
187 CMP AL, 'a'
188 JB check_upper_char
189 CMP AL, 'z'
190 JA skip_char
191 CMP AH, 1 ; AH = 1 means monoalphabetic encryption, since we are working with lower case, we use encryption
192 JE encrypt_lower_char ; AH = 0 means monoalphabetic decryption, since we are working with lower case, we use decryption
193 CMP AH, 0
194 JNE skip_char
195 LEA BX, decryption_table_lower
196 JMP translate_char
197 encrypt_lower_char: LEA BX, encryption_table_lower
198 JMP translate_char
199 check_upper_char: CMP AL, 'A'
200 JB skip_char
201 CMP AL, 'Z'
202 JA skip_char
203 CMP AH, 1 ; AH = 1 means monoalphabetic encryption, since we are working with upper case, we use encryption
204 JE encrypt_upper_char ; AH = 0 means monoalphabetic decryption, since we are working with upper case, we use decryption
205 CMP AH, 0
206 JNE skip_char
207 LEA BX, decryption_table_upper
208 JMP translate_char
209 encrypt_upper_char: LEA BX, encryption_table_upper
210 translate_char: XLATB
211
212
213
214
215
216
217
218 ; Subroutine to take input string from the user
219 ; The subroutine handles if the user presses backspace: delete char + inc CX
220 ; The subroutine allows the user to enter a maximum of 256 chars
221 get_input PROC NEAR
222 LEA DX, message_using_input
223 MOV AH, 09
224 INT 21H
225 LEA SI, input_string
226 MOV AH, 1
227 MOV CX, 255 ; To set a cap for string input to 256 chars
228 JMP input_loop
229 input_loop: INC CX ; Increment CX in case user presses backspace as a character is deleted
230 INT 21H
231 MOV [SI], AL
232 CMP AL, 8 ; backspace
233 JNE cont_input
234 MOV SI, offset input_string
235 JE input_loop ; If the string is empty just loop again without affecting SI and without incrementing CX
236 MOV [SI], ' '
237 CALL omit_space
238 DEC SI
239 JMP backspace_entered
240 cont_input: INC SI
241 CMP AL, 0
242 JE terminate_string
243 LOOP input_loop ; LOOP instead of JMP to incorporate <CX != 0000H> as a jump condition
244 MOV [SI-1], 0
245 MOV [SI], newline
246 MOV [SI+1], '$'
247 LEA SI, input_string
248 RET
249 get_input ENDP
250
251 end
252
253
254
255

```

Diag5 -Code running in emu8086

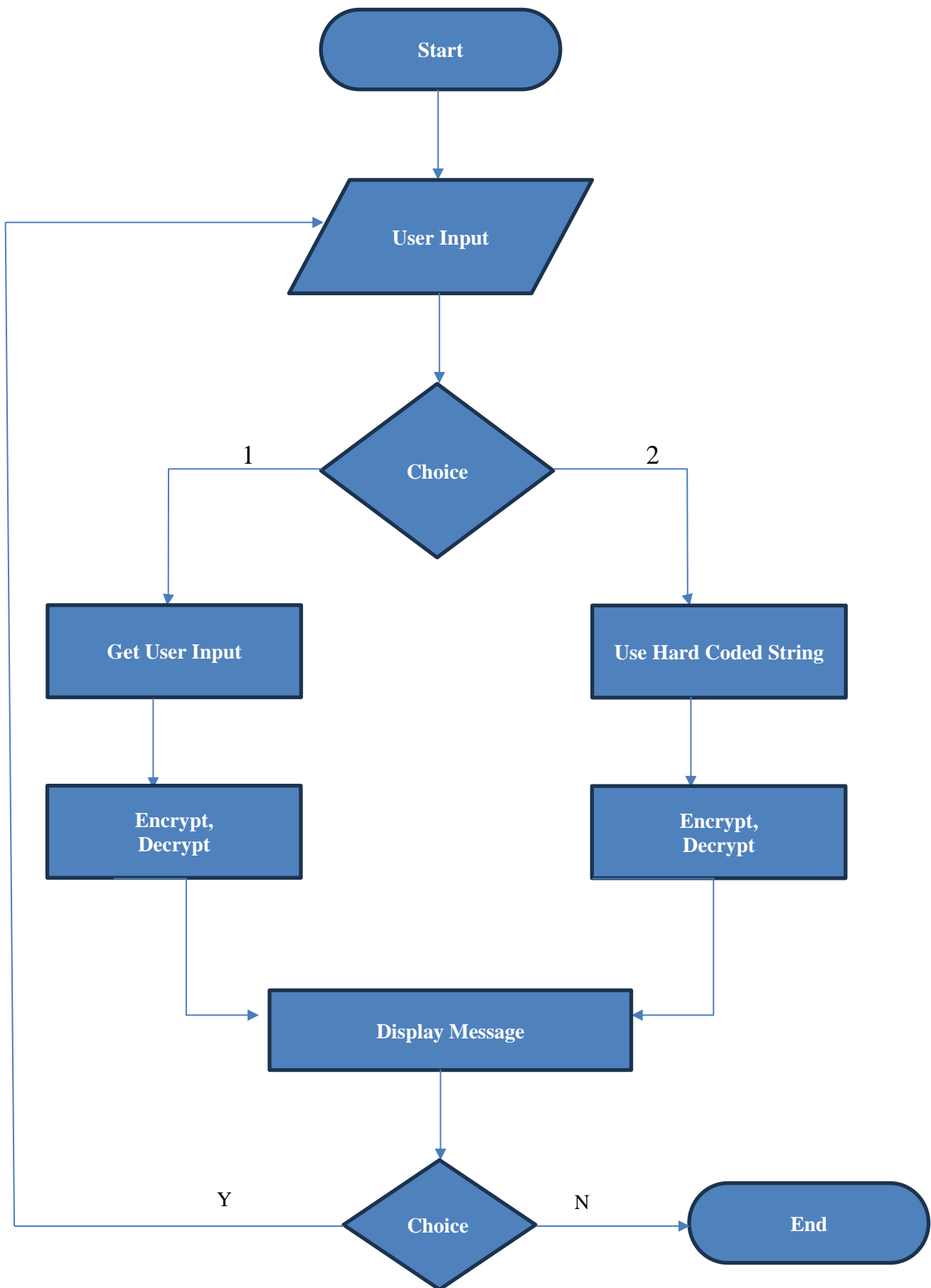
```

edit: C:\emu8086\MySource\COA2.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
202 JA skip_char
203 CMP AH, 1 ; AH = 1 means monoalphabetic encryption, since we are working with upper case, we use encryption
204 JE encrypt_upper_char ; AH = 0 means monoalphabetic decryption, since we are working with upper case, we use decryption
205 CMP AH, 0
206 JNE skip_char
207 LEA BX, decryption_table_upper
208 JMP translate_char
209 encrypt_upper_char: LEA BX, encryption_table_upper
210 translate_char: XLATB
211 MOV [SI], AL
212 POP BX
213 RET
214 enc_dec_char ENDP
215
216
217
218 ; Subroutine to take input string from the user
219 ; The subroutine handles if the user presses backspace: delete char + inc CX
220 ; The subroutine allows the user to enter a maximum of 256 chars
221 get_input PROC NEAR
222 LEA DX, message_using_input
223 MOV AH, 09
224 INT 21H
225 LEA SI, input_string
226 MOV AH, 1
227 MOV CX, 255 ; To set a cap for string input to 256 chars
228 JMP input_loop
229 input_loop: INC CX ; Increment CX in case user presses backspace as a character is deleted
230 INT 21H
231 MOV [SI], AL
232 CMP AL, 8 ; backspace
233 JNE cont_input
234 MOV SI, offset input_string
235 JE input_loop ; If the string is empty just loop again without affecting SI and without incrementing CX
236 MOV [SI], ' '
237 CALL omit_space
238 DEC SI
239 JMP backspace_entered
240 cont_input: INC SI
241 CMP AL, 0
242 JE terminate_string
243 LOOP input_loop ; LOOP instead of JMP to incorporate <CX != 0000H> as a jump condition
244 MOV [SI-1], 0
245 MOV [SI], newline
246 MOV [SI+1], '$'
247 LEA SI, input_string
248 RET
249 get_input ENDP
250
251 end
252
253
254
255

```

Diag6-Code running in emu8086

5. FLOWCHART



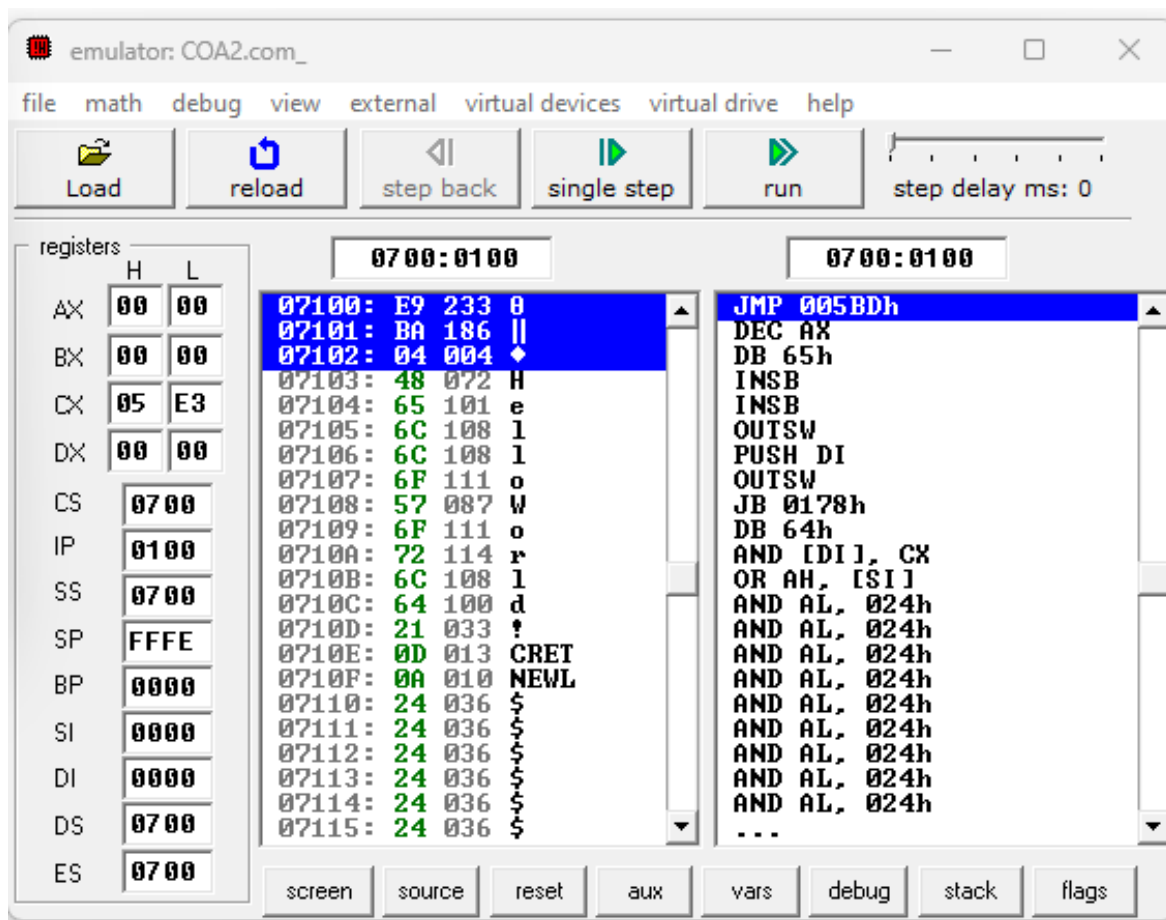
6. EXPERIMENT RESULTS & ANALYSIS

The program lets the user choose whether to enter a string or use the hard-coded one

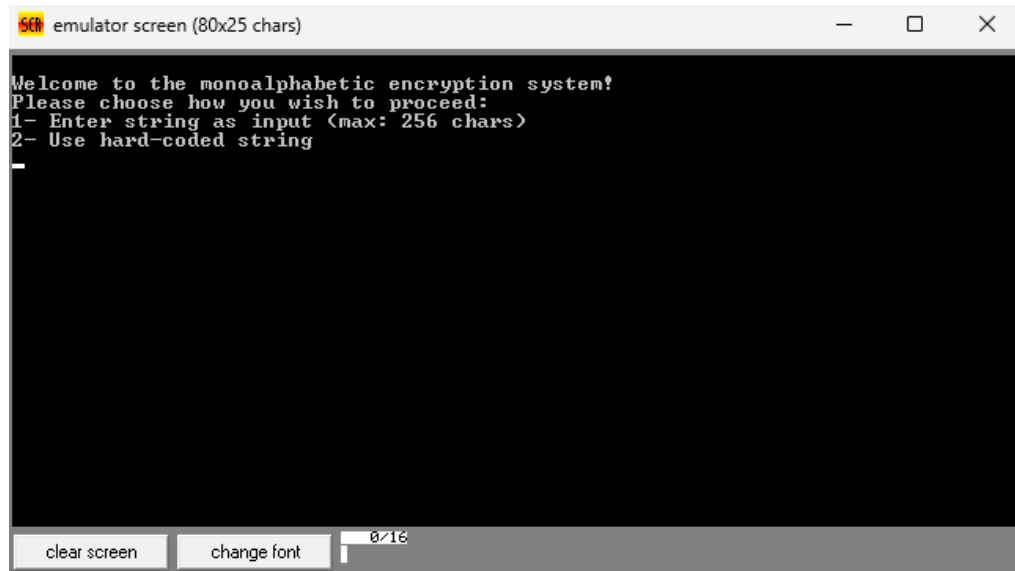
```
; Hard-coded string:
hardcoded_string      DB      'Welcome To encryption system', cret, newline, '$'

input_string          DB      259 dup '<'$>'      ; Reserved area for input string (256 chars + \r + \n + $)
```

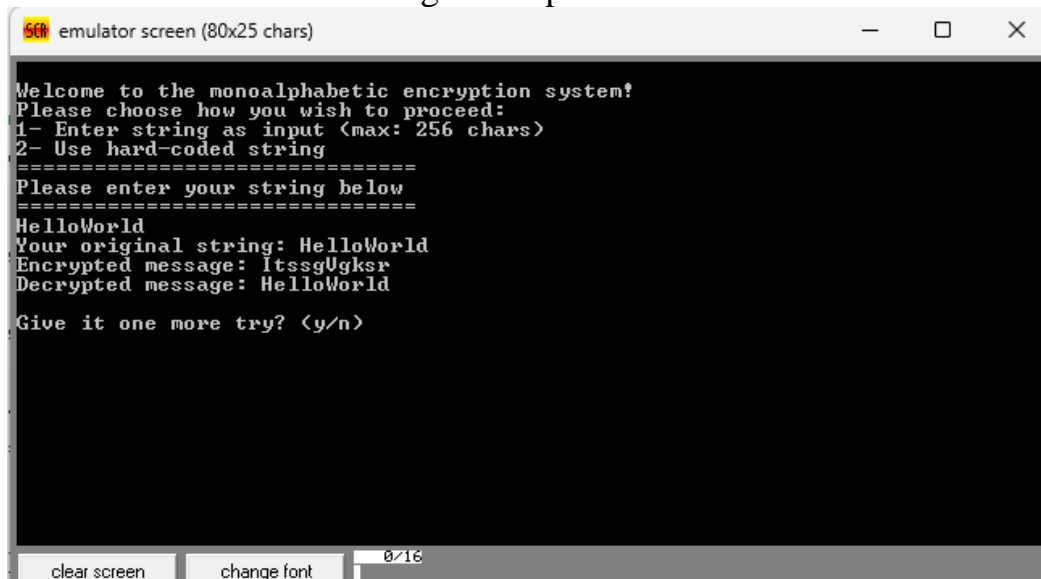
Diag8-Hard Coded String



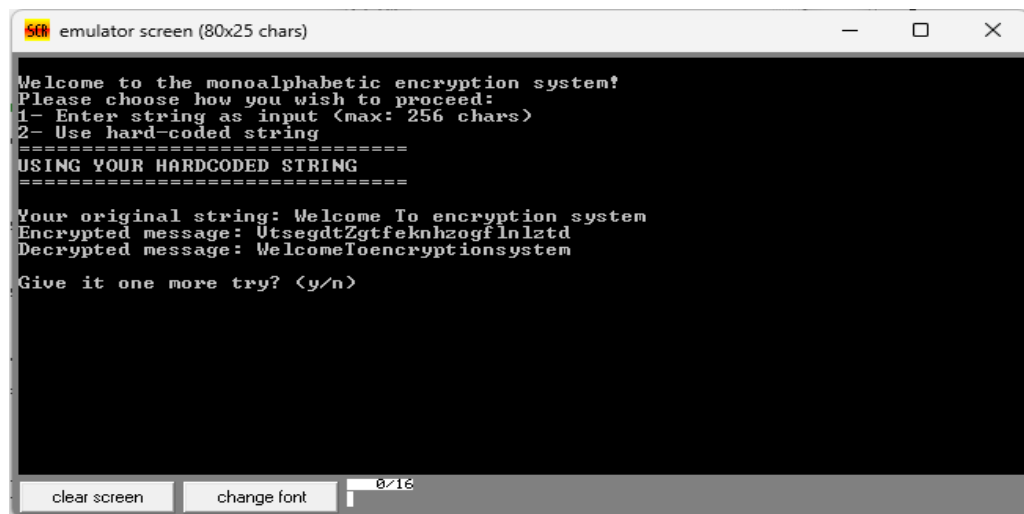
Diag9-Emulator



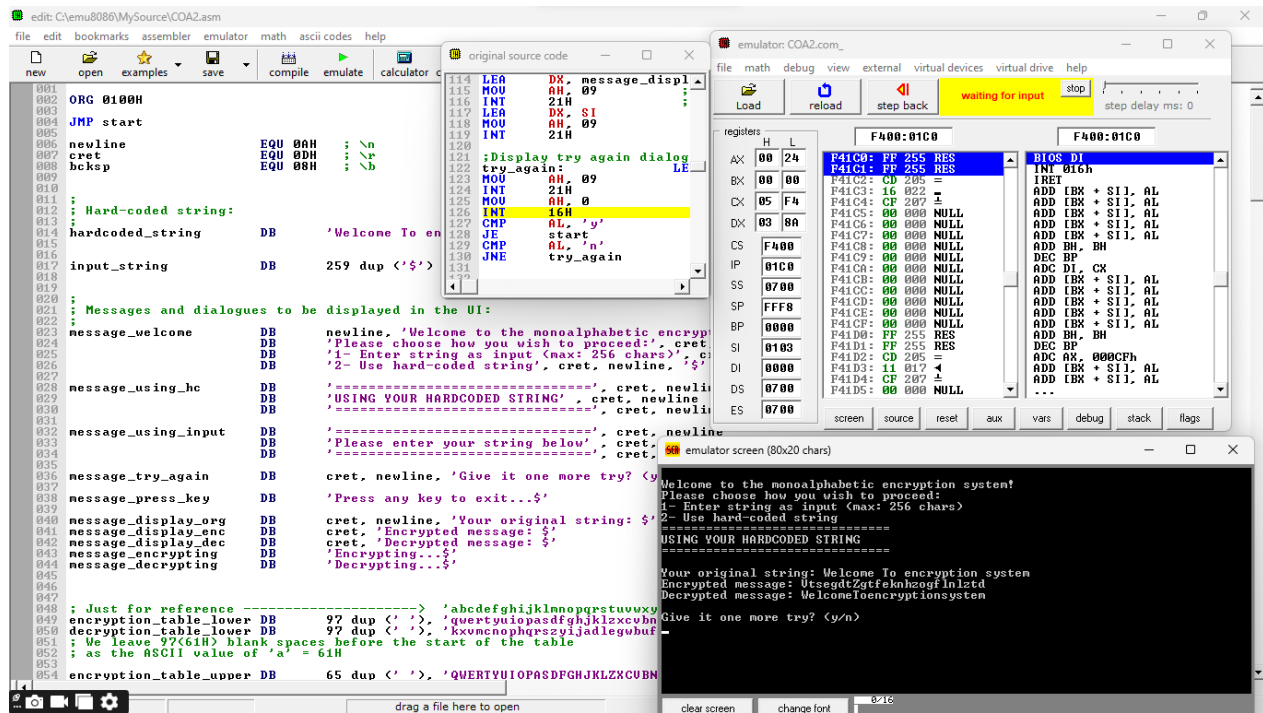
Diag10-OutputScreen



Diag11-Output with UserInput



Diag12-Output with Hard Coded String



Diag13-Program Execution

7. CONCLUSION

The monoalphabetic encryption system project serves as a valuable tool for data security and communication privacy. While this assembly code example provides a basic implementation, it highlights the need for encryption and decryption mechanisms in various real-world applications.

Importance of Encryption:

- **Data Protection:** In an increasingly digital world, the protection of sensitive information is paramount. Encryption helps safeguard data from unauthorized access and eavesdropping, ensuring confidentiality.
- **Privacy:** Individuals and organizations need encryption to maintain the privacy of their personal and confidential data, such as financial records, healthcare information, and communication.
- **Secure Communication:** Secure communication channels are essential, particularly in industries like finance, healthcare, and government, where transmitting sensitive information is routine.
- **E-commerce:** E-commerce platforms rely on encryption to protect customer data, including credit card information and personal details.
- **National Security:** Governments employ encryption to safeguard classified information and maintain national security.
- **Future Development:** The need for encryption and data security will continue to grow as technology evolves. In an era of increasing cyber threats and data breaches, projects like this serve as a foundation for developing advanced encryption methods and secure applications.

In conclusion, the monoalphabetic encryption system project underscores the significance of data security and privacy in our digital age. It not only provides a basic implementation but also encourages further exploration and development in the field of encryption. As technology advances, the demand for secure communication and data protection will remain a crucial consideration for individuals, organizations, and governments.

8.REFERENCES

<https://www.geeksforgeeks.org/architecture-of-8086/>

Unit2: Basic Structure of a Computer

<https://www.philadelphia.edu.jo/academics/qhamarsheh/uploads/emu8086.pdf>

<https://www.cloudflare.com/learning/ssl/what-is-encryption/>