Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2023), B.Sc. in CSE (Day)

Children learning App

Course Title: ***Microprocessors and Microcontrollers Lab***
Course Code: ***CSE 304***
Section: ***213-D2***

Students Details

Name	ID
Hridoy Debnath	213002239
Nadib Rana	213002247

Submission Date: 28/12/2023
Course Teacher's Name: ***Sudip Ghoshal***

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

1 Introduction **3**

1.1 Overview . . . 3

1.2 Motivation . . . 3

1.3 Problem Definition . . . 3

1.3.1 Problem Statement . . . 3

1.3.2 Complex Engineering Problem . . . 4

1.4 Design Goals/Objectives . . . 5

1.5 Application . . . 5

2 Design/Development/Implementation of the Project **6**

2.1 Introduction . . . 6

2.2 Project Details . . . 6

2.3 Implementation . . . 7

2.4 Flowchart . . . 18

3 Performance Evaluation **19**

3.1 Simulation Environment/ Simulation Procedure . . . 19

3.2 Results Analysis/Testing . . . 19

3.2.1 Result_portion_1: . . . 19

3.2.2 Result_portion_2 . . . 20

3.2.3 Result_portion_3 . . . 20

3.2.4 Result_portion_4 . . . 21

3.2.5 Result_portion_5 . . . 22

3.3 Results Overall Discussion . . . 22

4 Conclusion **23**

4.1 Discussion . . . 23

4.2 Limitations . . . 23

4.3 Scope of Future Work 24

Chapter 1

Introduction

1.1 Overview

In this project, we want to build a learning environment with some fun for 2 years to 10 years children. The "Children Learning App" is an educational software designed to operate within the EMU8086 microprocessor emulator, providing a foundational learning platform for children. This application is made using assembly language, offering a unique approach to early education by integrating the simplicity of learning alphabets and numbers. So we hope children enjoy the system, learn more, and develop day-to-day.

1.2 Motivation

For this project, we chose of EMU-8086 environment and it is driven by the challenge of developing educational content in a unique and stimulating environment and also enhancing our assembly language potential to do better projects and learn new things when facing difficulties. when we research children's learning, we find that they do not prefer to learn traditionally, so if we can interestingly learn things, they will learn better, we want to create an environment where they can learn words by pressing the keyboard key and for every alphabet key they see 5 words on based on this key like(A for apple, Asia, alphabet, Australia, argument)that and for the number key they press number like(1or5or8) and they will see written it like(one, five, eight) also in the same time they know how the number comes or what is before the number of it.

1.3 Problem Definition

1.3.1 Problem Statement

The "Children Learning App" aims to fill this void by employing assembly language within the EMU8086 emulator to create an accessible, engaging learning application. This project confronts the challenge of blending the simplicity required for children's

educational software with the complexity of low-level programming, aiming to introduce young learners to educational content through the platform. A notable challenge in the project was creating an intuitive user experience within the constraints of assembly language and the emulator's capabilities.

1.3.2 Complex Engineering Problem

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	Explain how to address
P1: Depth of knowledge required	In-depth knowledge of assembly language programming is crucial. This includes an understanding of CPU architecture, memory management, data representation, and register management. The project will address this thorough research and practical application, guided by educational resources and emulator documentation.
P2: Range of conflicting requirements	The application must be simple enough for children to use yet designed within the constraints of assembly language. Balancing user-friendliness with technical limitations requires iterative design and user testing to find an optimal compromise.
P3: Depth of analysis required	Need depth analysis in assembly language such as:(register management, data segment, comparing, jump instruction, and leveling) these are all topics we need to analyze very much
P4: Familiarity of issues	The issues of educational application development in a low-level language are not common. Addressing this unfamiliarity involves consulting domains in education, and reviewing case studies of similar projects.
P5: Extent of applicable codes	Applicable codes refer to both educational standards for early learning tools and programming best practices. The project will align with these by following curriculum guidelines and coding standards for assembly language.
P6: Extent of stakeholder involvement and conflicting requirements	Parents, educators, and children are key stakeholders. But 2 years to 8 years children is our focused stakeholders.
P7: Interdependence	The project's components are interdependent; educational content must match the technical capabilities of the EMU8086, and user interface design depends on both.

1.4 Design Goals/Objectives

- To develop an accessible, engaging, and user-friendly application that introduces children to the basics of the English alphabet and numerical digits.

- To demonstrate the potential of assembly language in creating educational tools, thereby bridging the gap between complex programming and elementary learning.

- This project is to improve skills on assembly language and solve various problems

1.5 Application

The "Children Learning App" developed using assembly language on the EMU8086 emulator has practical applications in educational settings, specifically for young learners(age:2-10). Its primary real-world application is to serve as a foundational tool in early childhood education centers and homes, helping children learn and reinforce their knowledge of the alphabet and numbers.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

Our project children's learning application such an application that helps our little children to grow up with many words as per our alphabet in the English language and also learn numbers from this application. here basically students learn by pressing the keyboard key we all know that our little children are very interested in mobile, tab, calculator, mouse, or keyboard because they can do pressing on it. we see in our house that little children are always interested in this device and in the project, we use the keyboard as the student's favorite .so here students press a key and see 5 words for each alphabet key press and press the number, they also learn what actually that that number is. [1] [2] [3].

2.2 Project Details

Our project children's learning application is developed with assembly language. In this project basically, we can implement the concept of low-level language that we learn in our classroom such as we learn (taking input, showing output, comparing, looping, methods, and some different instructions like(JMP, JE, JL Etc)) we implement most of the topic in the project we don't add any new topic that are not discuss in class. so in this project when the user runs the code they see a welcome interface and also see the message that press any key for start if the user presses any key on the keyboard they will see two options 1. study and another *Exit from here if user can go study he/she need to press 1 or if user don't want to go with study and want to leave from this interface they need to press * key after pressing * key they will see thank you message but if any user don't press 1 or * and presses others word from keyboard they again see 1. study *Exit massage. after that, if the user press 1 our program shows STUDY and some messages and last instructs the user to input the alphabet or number so here if the user presses any alphabet in capital words or small words they see the same 5 different words based on the given alphabet and if the user press 5 they see FIVE and a message that is "A number that is more than four" and again ask for alphabets or a number and at the last,

if the user press * the program shows a thank you message and the PROGRAM returns control TO THE OPERATING SYSTEM.

2.3 Implementation

In this section, I will Explain all the code details of my project: First, I open the EMU-8086 application on my computer and then click on new, and a blank window(page) comes here we need to do code.
First, we need to declare the size of the program and here I use Model small
.MODEL SMALL
; After that, we need to declare the stack size:
.STACK 100H
;and then we declare the data segment under
.DATA

```
TESTT DB "                        1.STUDY OR *.Exit$"
TESTT1 DB "                        STUDY",0dh,0ah
       DB "  Hope you will learn 5 words for every Alphaber &",0dh,0ah
       DB " learn the number and it's origin or before number ",0dh,0ah
       DB "
",0dh,0ah
       DB "       Input any Alphabet or Digit$",0dh,0ah

 FOR DB " for $"

Z1 DB " ZERO.The absence of all magnitude or quantity.
The origin of any kind of measurement$"
Z2 DB " ONE. The first whole number above zero.$"
Z3 DB " TWO.A number that is one more than one.$"
Z4 DB " THREE.A number that is one more than 2$"
Z5 DB " FOUR.A number that is one more than three.$"
Z6 DB " FIVE.A number that is one more than four.$"
Z7 DB " SIX.A number that is one more than five.$"
Z8 DB " SEVEN.A number that is one more than six.$"
Z9 DB " EIGHT.A number that is one more than seven.$"
Z10 DB " NINE.A number that is one more than eight$"

MSG db "                          ASSEMBLY PROJECT", 0dh,0ah
    db "                        ====================", 0dh,0ah
    db "                   ==== Children Learning App ====", 0dh,0ah
        db "                        ====================", 0dh,0ah
        db "                        Press any key to start...", 0dh,0ah
        db "                         $", 0dh,0ah


a     db "Apple,Accident,America,Animal,Asia.$"
b     db "Ball,Bank,Bicycle,Black,Brother$"
```

```
c       db "Cat , Call , Carry , Captain , Chair .$"
d       db "Dog , Design , Desk , Dairy , Dictionary .$"
e       db "Egg , Easy , Early , Effect , Electric$"
f       db "Fan , Feel , Feet , Female , Flight .$"
g       db "Goat , Game , Gentlemen , Glass , Girl .$"
h       db "Hen , Hair , Half , Hardware , Height , Happy .$"
i       db "Ink , Important , Increse , India , Iron , Island .$"
j       db "Jackfruit , Job , Join , Jump , Just .$"
k       db "Kite , Keyboard , Key , Killed , King .$"
l       db "Lion , Laptop , Lady , Large , Law .$"
m       db "Mobile , Machine , make , main , Maneger .$"
n       db "Nose , New , Never , Newspaper , Nation .$"
o       db "Orange , Online , Opinion , One , Order .$"
p       db "Pen , Practice , President , Press , Promise .$"
q       db "Queen , Question , Quickly , Quite .$"
r       db "Rose , Robot , Radio , Railway , Rain .$"
s       db "Sheep , School , Science , Sea , Search .$"
t       db "Tiger , Teacher , Temperature , Test , Tax .$"
u       db "Umbrella , Uncle , Under , Upset , Use .$"
v       db "Violin , Valley , Video , Village , Voice .$"
w       db "Window , Weekend , Weight , West , Weather .$"
x       db "X-ray , Xebec , Xenia , Xenic , Xenon .$"
y       db "Yolk , Yourself , Yes , Yet , Yard .$"
z       db "Ziraf , Zero , Zoom , Zonal , Zebra .$"

TNX DB "                ===o THANK YOU FOR YOUR PARTICIPATION
o===$"
INV DB "                                INVALID INPUT .$"
SPACE DB "                                $"
; in the data segment section we declared many variables such as
(TEST, TESTT, FOR, Z1-Z10, A-Z etc)
```

; Now, starting the code segment the main segment of our code, the main program starts from there. in this segment we have done all the conditions, checker, jumping, and all of my code that needed to run for this project:

```
.CODE
MAIN PROC       ;main function star from here

    MOV AX,@DATA ;import the data from the data segment
    MOV DS,AX  ;move the data AX to data segment

    INCLUDE 'EMU8086.INC' ;add the libary function

    MOV AH,9            ;for printing string
    LEA DX,MSG         ; Load effective address use for massage print
    INT 21H            ; interrupter call

    MOV AH,1              ;For taken input
```

```
    INT 21H
    TOP:                ;create level
    PRINTN ''    ; print new line using INCLUDE 'EMU8086.INC' libery fu
    MOV AH,9
    LEA DX,TESTT
    INT 21H

    PRINTN ''
    MOV AH,9
    LEA DX,SPACE
    INT 21H

    MOV AH,1
    INT 21H       ;take input from the user and store the value in al to
    MOV BL,AL

    PRINTN ''

    CMP BL,'1'    ;here compare the user input(store in bl) with '1'
    JE OPP1            ;( if JUMP-IF-EQUALto  OOP1 LEVEL then go opp1)

    CMP BL,'*'       ;same compare in this line
    JE OP2

    JMP TOP        ;if the bl value not match with 1 or * it again go TO
                   ;and start newly

    PRINTN ''
OPP1:
    PRINTN ''
    MOV AH,9
    LEA DX,TESTT1
    INT 21h

PRINTN ''

PRINTN ''

  OP1:
PRINTN ''

MOV AH,9
LEA DX,SPACE
INT 21H
MOV AH,1
INT 21H   ;take input from the user and store it al then again move al
MOV BL,AL
```

```
CMP BL,'*';compare on * if same then go to op2 level and show thank you
 JE OP2

 CMP BL,58 ; here compare bl with ascii value 58.
           ; if the ascii value less than 58 then go to NUMBER LEVEL
 JL NUMBER

 MOV AH,9
 LEA DX,FOR
 INT 21H
 CMP  BL,'A'
 JE C1
 CMP BL,'a'
 JE C1
   CMP  BL,'B'
 JE C2
 CMP BL,'b'
 JE C2
    CMP  BL,'C'
 JE C3
 CMP BL,'c'
 JE C3
  CMP  BL,'D'
 JE C4
 CMP BL,'d'
 JE C4
 CMP  BL,'E'
 JE C5
 CMP BL,'e'
 JE C5
   CMP  BL,'F'
 JE C6
 CMP BL,'f'
 JE C6
  CMP  BL,'G'
 JE C7
 CMP BL,'g'
 JE C7
  CMP  BL,'H'
 JE C8
 CMP BL,'h'
 JE C8
  CMP  BL,'I'
 JE C9
 CMP BL,'i'
 JE C9
  CMP  BL,'J'
 JE C10
```

```
CMP BL,'j'
JE C10
 CMP  BL,'K'
JE C11
CMP BL,'k'
JE C11
 CMP  BL,'L'
JE C12
CMP BL,'l'
JE C12
 CMP  BL,'M'
JE C13
CMP BL,'m'
JE C13
   CMP  BL,'N'
JE C14
CMP BL,'n'
JE C14
   CMP  BL,'O'
JE C15
CMP BL,'o'
JE C15
 CMP  BL,'P'
JE C16
CMP BL,'p'
JE C16
     CMP  BL,'Q'
JE C17
CMP BL,'q'
JE C17
 CMP  BL,'R'
JE C18
CMP BL,'r'
JE C18
 CMP  BL,'S'
JE C19
CMP BL,'s'
JE C19
   CMP  BL,'T'
JE C20
CMP BL,'t'
JE C20
    CMP  BL,'U'
JE C21
CMP BL,'u'
JE C21
 CMP  BL,'V'
JE C22
```

```
CMP BL,'v'
JE C22
  CMP  BL,'W'
JE C23
CMP BL,'w'
JE C23
  CMP  BL,'X'
JE C24
CMP BL,'x'
JE C24
  CMP  BL,'Y'
JE C25
CMP BL,'y'
JE C25
  CMP  BL,'Z'
JE C26
CMP BL,'z'
JE C26
          NUMBER:
   MOV AH,2
 MOV DL,8
 INT 21H
   CMP  BL,'0'
JE C27
   PRINTN ''
 CMP  BL,'1'
JE C28
 CMP BL,'2'
JE C29

 CMP  BL,'3'
JE C30

CMP BL,'4'
JE C31

 CMP  BL,'5'
JE C32

CMP BL,'6'
JE C33
  CMP  BL,'7'
JE C34

CMP BL,'8'
JE C35

 CMP  BL,'9'
```

```
JE  C36
 JMP  OP1

 MOV  AH, 9
 LEA  DX, INV
 INT  21H
    JMP  OP1

 C1 :
 MOV  AH, 9
LEA  DX, a
 INT  21H
 JMP  OP1

 C2 :
 MOV  AH, 9
LEA  DX, b
 INT  21H
    JMP  OP1

 C3 :
 MOV  AH, 9
LEA  DX, c
 INT  21H
    JMP  OP1

 C4 :
 MOV  AH, 9
LEA  DX, d
 INT  21H
 JMP  OP1

 C5 :
 MOV  AH, 9
LEA  DX, e
 INT  21H
    JMP  OP1

 C6 :
 MOV  AH, 9
LEA  DX, f
 INT  21H
    JMP  OP1

 C7 :
 MOV  AH, 9
LEA  DX, g
 INT  21H
```

```
    JMP  OP1

  C8 :
  MOV  AH, 9
LEA  DX, h
  INT  21H
    JMP  OP1

  C9 :
  MOV  AH, 9
LEA  DX, i
  INT  21H
    JMP  OP1

  C10 :
  MOV  AH, 9

  lEA  DX, j
  INT  21H
     JMP  OP1

  C11 :
  MOV  AH, 9
LEA  DX, k
  INT  21H
     JMP  OP1

  C12 :
  MOV  AH, 9
LEA  DX, l
  INT  21H
     JMP  OP1

  C13 :
  MOV  AH, 9
LEA  DX,m
  INT  21H
    JMP  OP1

  C14 :
  MOV  AH, 9
LEA  DX, n
  INT  21H
    JMP  OP1

  C15 :
  MOV  AH, 9
LEA  DX, o
```

```
  INT  21H
   JMP  OP1

  C16 :
 MOV  AH, 9
LEA  DX, p
  INT  21H
   JMP  OP1

  C17 :
 MOV  AH, 9
LEA  DX, q
  INT  21H
   JMP  OP1

  C18 :
 MOV  AH, 9
LEA  DX, r
  INT  21H
   JMP  OP1

  C19 :
 MOV  AH, 9
LEA  DX, s
  INT  21H

  JMP  OP1

  C20 :
 MOV  AH, 9
LEA  DX, t
  INT  21H
   JMP  OP1

  C21 :
 MOV  AH, 9
LEA  DX, u
  INT  21H
   JMP  OP1

  C22 :
 MOV  AH, 9
LEA  DX, v
  INT  21H
   JMP  OP1

  C23 :
 MOV  AH, 9
```

```
LEA  DX, w
  INT  21H
     JMP  OP1

  C24 :
  MOV  AH, 9
LEA  DX, x
  INT  21H
    JMP  OP1

  C25 :
  MOV  AH, 9
LEA  DX, y
  INT  21H
     JMP  OP1

  C26 :
  MOV  AH, 9
LEA  DX, z
  INT  21H
  JMP  OP1

  C27 :
  MOV  AH, 9
LEA  DX, z1
PRINTN  ' '
  INT  21H
  JMP  OP1

     C28 :
  MOV  AH, 9
LEA  DX, z2
  INT  21H
  JMP  OP1

        C29 :
  MOV  AH, 9
LEA  DX, z3
  INT  21H
  JMP  OP1

       C30 :
  MOV  AH, 9
LEA  DX, z4
  INT  21H
  JMP  OP1

        C31 :
```

```
  MOV AH,9
 LEA DX,z5
  INT 21H
  JMP OP1

       C32:
  MOV AH,9
 LEA DX,z6
  INT 21H
  JMP OP1

      C33:
  MOV AH,9
 LEA DX,z7
  INT 21H
  JMP OP1

       C34:
  MOV AH,9
 LEA DX,z8
  INT 21H
  JMP OP1

       C35:
  MOV AH,9
 LEA DX,z9
  INT 21H
  JMP OP1

      C36:
  MOV AH,9
 LEA DX,z10

  INT 21H
  JMP OP1
 OP2:
 PRINTN ''
 PRINTN ''

  MOV AH,9
  LEA DX,TNX
  INT 21H

  MOV AH,4CH    ;this line is end the main function
  INT 21H

  MAIN ENDP
END MAIN
```

2.4 Flowchart

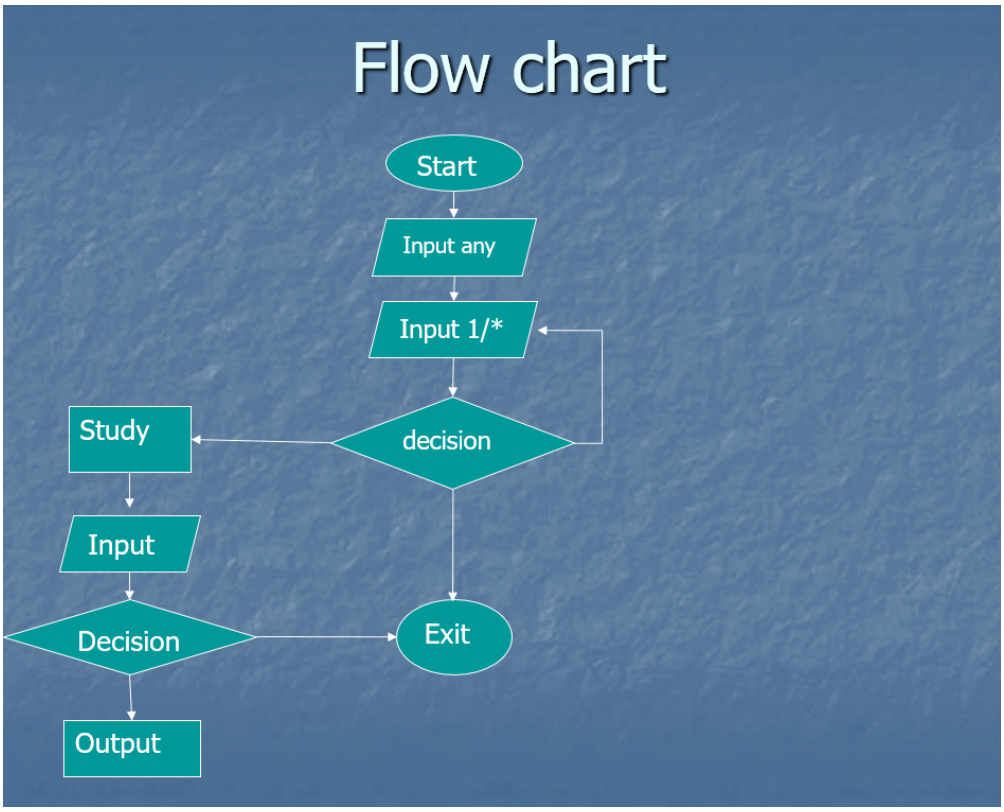

Figure 2.1: Flowchart of children learning application project

Chapter 3

Performance Evaluation

Our project run successfully and we test multiple times from presses of many alphabets and numbers. we can't identify any error. we also test it for children. and we also tested that every word that I inserted into our project was the right word.

3.1 Simulation Environment/ Simulation Procedure

3.2 Results Analysis/Testing

3.2.1 Result_portion_1:

The output of our project set in this section.
First, when we run the program we will this interface

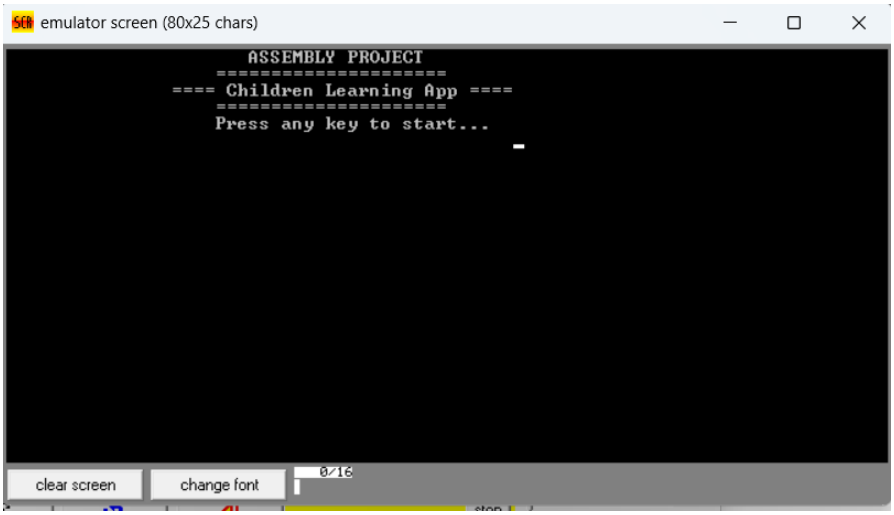

Figure 3.1: First interface of our project

3.2.2 Result_portion_2

After pressing any key we will see Figure (3.2) this interface

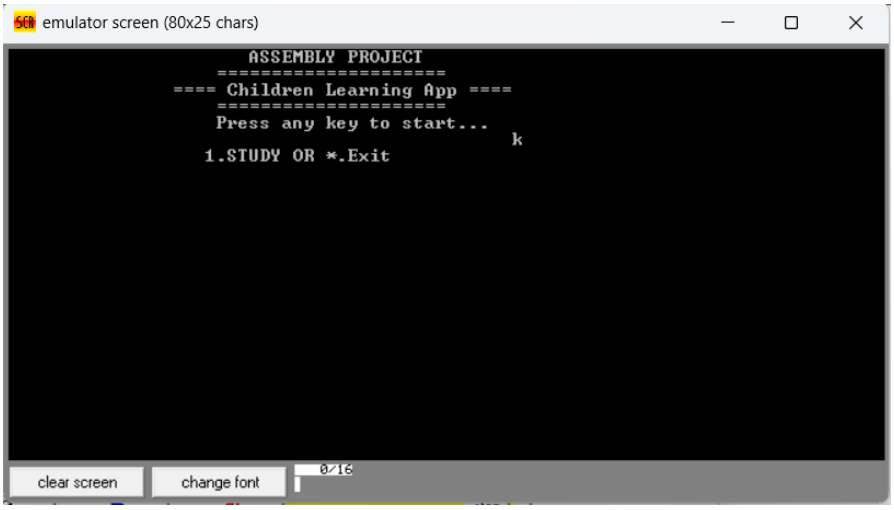

Figure 3.2: After pressing any key from Keyboard

This section offer us to go with STUDY or Exit.For star study we press 1 and for leave the page we use * key.

3.2.3 Result_portion_3

Now if I press 1 then see this type of interface in Figure (3.3)

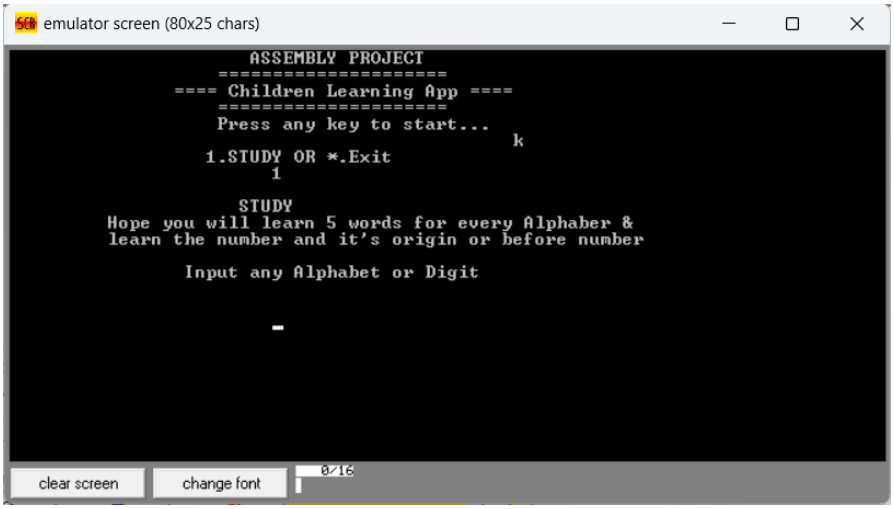

Figure 3.3: Output of After pressing 1 from Keyboard

When we press 1 we see this type of interface and the interface offer us for input alphabet or number also we will see some massage.

3.2.4 Result_portion_4

In this part, user need to input any alphabet or number :

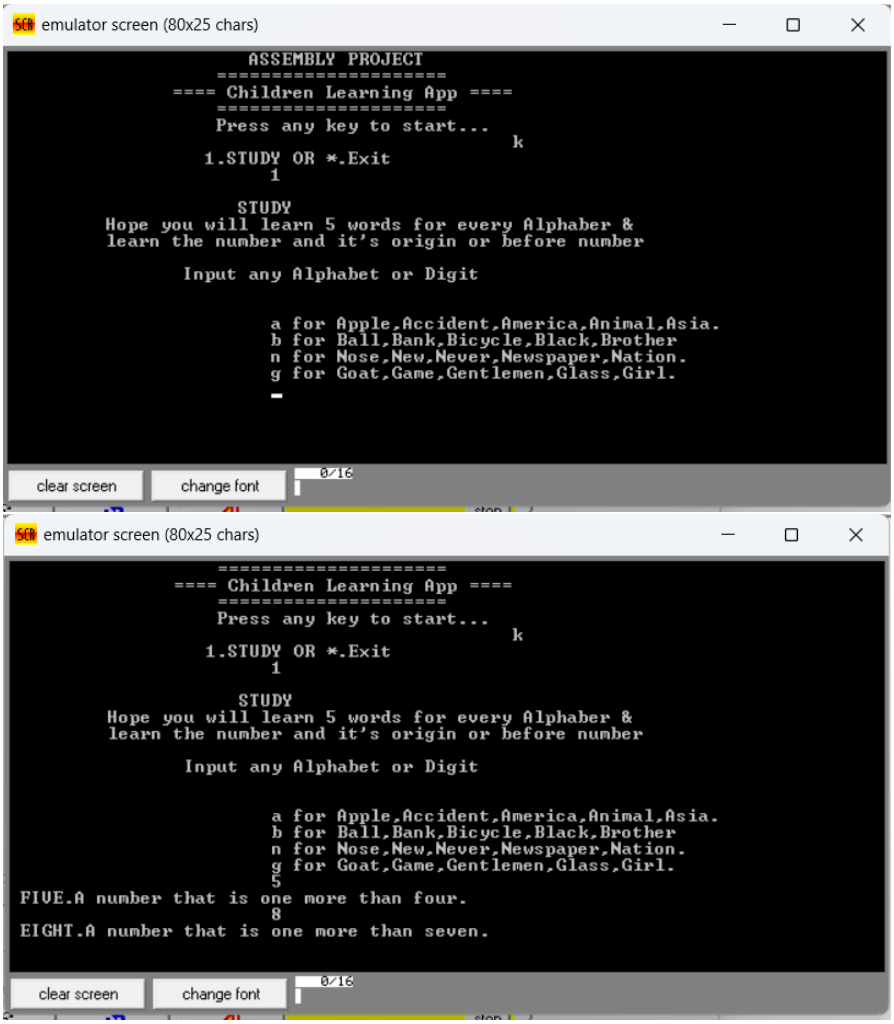

Figure 3.4: After pressing any key from Keyboard

For testing purpose i already press alphabets(a,b,n,g) and number(5,8)and see the result.

3.2.5 Result_portion_5

Now if I press * then the program shows a Thank you message and ends the program

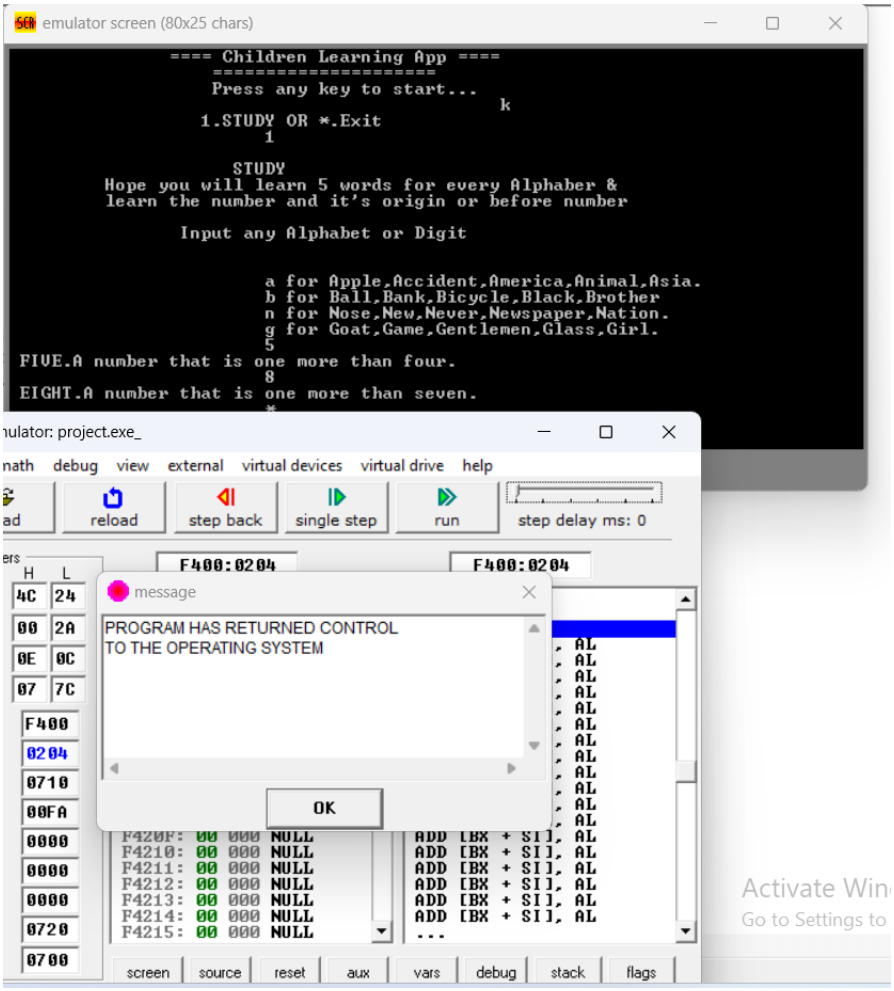

Figure 3.5: Output of After pressing * key from Keyboard

3.3 Results Overall Discussion

The overall results are satisfactory for me if I consider this a low-level project. Because we know in low-level language we have many boundaries if we consider it then I think it would be a good project for a student that actually we are. And details of our project I will explain before . I think this is a simple project and very easy to use and I also believed that every student can run it if once we explain it.

Chapter 4

Conclusion

4.1 Discussion

- Our project effectively takes a single character as input and executes different actions based on that input also Messages are displayed to the user.

- Our project successfully handles user input and demonstrates clear organization with the use of labels.

- The project appears straightforward, with potential complexity lying in the branching logic and handling different input cases.

- The Project likely provided an opportunity to practice assembly language programming, conditional branching, and input/output operations.

- Depending on the project specifics, we gained skills related to x86 assembly language, conditional branching, and user input handling.

- The project successfully accomplishes the objective by taking user input, branching based on characters, displaying messages, and terminating the program.

4.2 Limitations

Low-level and devoid of high-level abstractions is assembly language. Because of this, maintaining the code becomes more difficult. It may be difficult to add features in the future, and developers who are unfamiliar with Assembly may have a steep learning curve.
The only interactive features in the app are basic input and output. Although it gives information on alphabets and numbers, it is devoid of interactive features that could improve the learning process, such as games(visually) or quizzes.
Multimedia and visual elements are not well suited for assembly language. The app's

ability to visually engage users is limited because it only provides text-based information. The user experience may be greatly improved by adding multimedia or graphical elements, however doing so would necessitate switching to a higher level language.

The alphabet and numbers are the main focus of the current implementation. Because of the limitations of Assembly language, growing the app's capacity or adding more sophisticated features might not be feasible. In order to scale in the future, a more flexible programming language would need to be used.

4.3 Scope of Future Work

The following changes are put forward in order to address the problems and enhance the..
Chilled Learning App:
Transition to a Higher-Level Language: For greater readability, maintainability, and extensibility , think about moving the project to a higher-level language like Python or Java or C++.

Interactive Features: We can make learning more interesting for kids, include interactive components like games, quizzes, and multimedia content.

Support with Graphics:We will used HTML and CSS or other language for add graphics and images to improve the visual appeal and add excitement to the learning process.

Scalability:We will create a platform for the app that can readily accept new learning modules and a variety of educational resources.

References

[1] Teaching of 8088/86 programming with 8086 assembly emulator. *Journal of microprocessor Research*, 70:263–286, 2017.

[2] Douglas Laney. Documentation for emu8086, 2001.

[3] Amritos vlog. https://amritoo.hashnode.dev/a-beginners-guide-to-assembly-language-using-emu8086.