Project Title:

Simple Arithmetic Quiz App

Computer Organization and Assembly Language

Submitted by:

Jeremy Cube
Bismillah Constantino
Jericho Vince Tacata
Mark Edison Rosario
Piolo Brian Jizmundo

Bachelor of Computer Science 3 - 3

Submitted to:

Prof. Angelic P. Payne

Polytechnic University of the Philippines
Sta. Mesa, Manila

February 2022

I. Project Scope

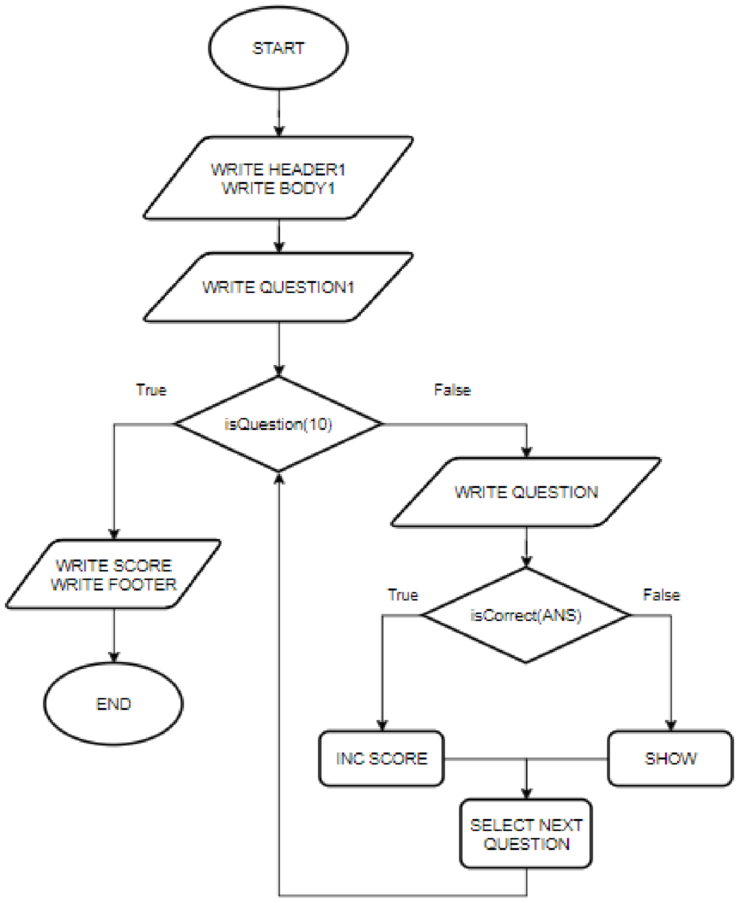
The scope of this project extends to a simple quiz application. It will only feature simple arithmetic problems. The purpose of this project is to test the knowledge of young students about basic math. The expected users of this application are Grade 3 students.

II. Objective

Technical - To fully build and design a working project using an assembly language, NASM.

Schedule - To complete this small-scale project within 1 week

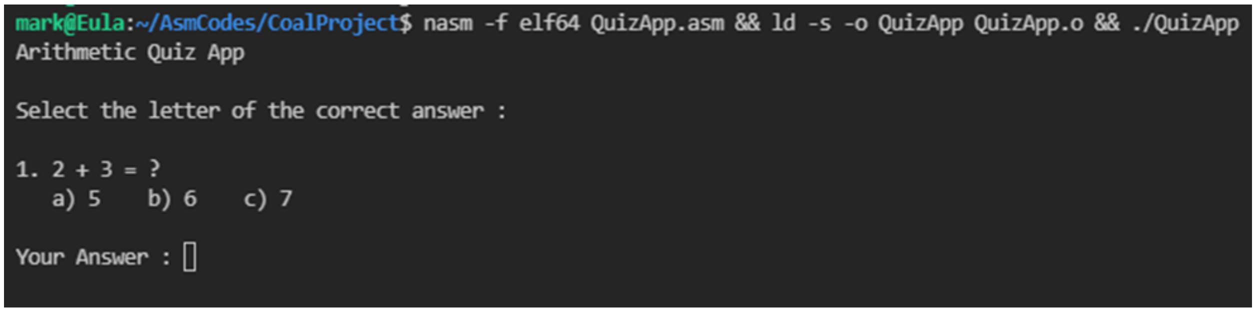
III. Flowchart



IV. Instruction Manual

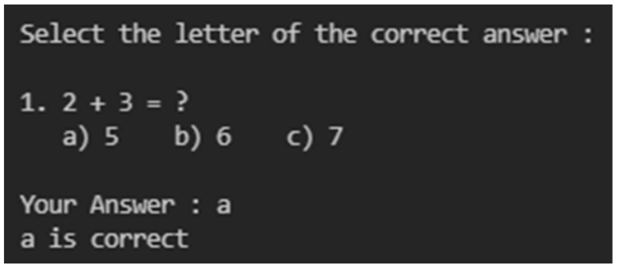
1. Run the program by executing the command : ./QuizApp
2. The program shall display a welcome message.
3. The program shall display the first question and three choices.
4. The user shall select the letter of his answer and input the letter in the command line.
5. The program will display the next questions and the user will provide his answer for each one.
6. When the questions end, the program will display the user's score.

V. Sample Input and Output

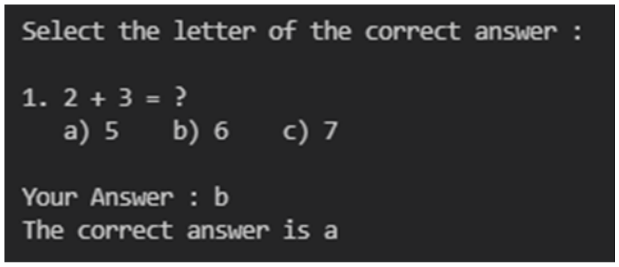


```
mark@Eula:~/AsmCodes/CoalProject$ nasm -f elf64 QuizApp.asm && ld -s -o QuizApp QuizApp.o && ./QuizApp
Arithmetic Quiz App

Select the letter of the correct answer :

1. 2 + 3 = ?
   a) 5    b) 6    c) 7

Your Answer : 
```

After the successful compilation of the assembly program. The program will then run and display the following messages.

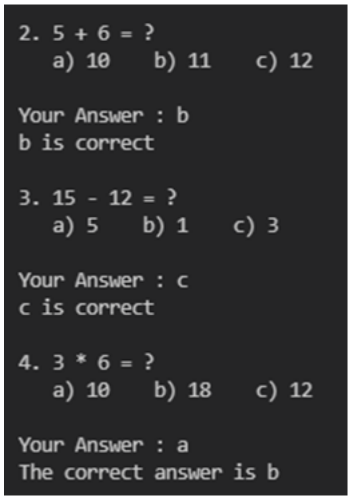


```
Select the letter of the correct answer :

1. 2 + 3 = ?
   a) 5    b) 6    c) 7

Your Answer : a
a is correct
```

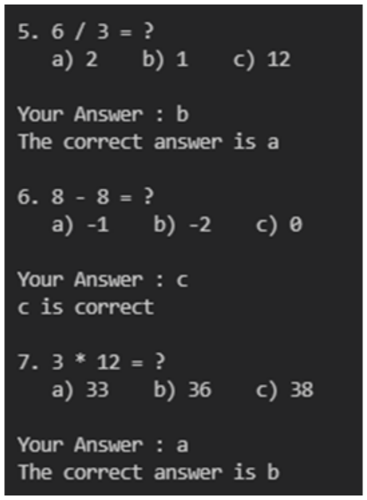


```
Select the letter of the correct answer :

1. 2 + 3 = ?
   a) 5    b) 6    c) 7

Your Answer : b
The correct answer is a
```

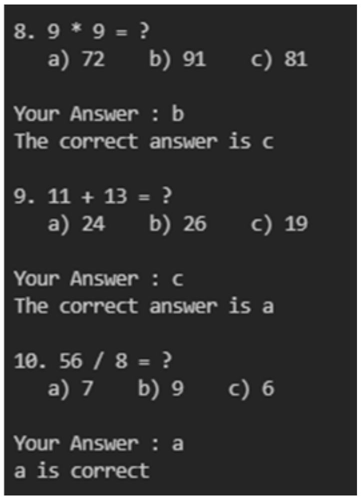
The program will ask the user's answer. A message will be displayed depending on the correctness of the user's answer.



```
2. 5 + 6 = ?
   a) 10    b) 11    c) 12

Your Answer : b
b is correct

3. 15 - 12 = ?
   a) 5    b) 1    c) 3

Your Answer : c
c is correct

4. 3 * 6 = ?
   a) 10    b) 18    c) 12

Your Answer : a
The correct answer is b
```

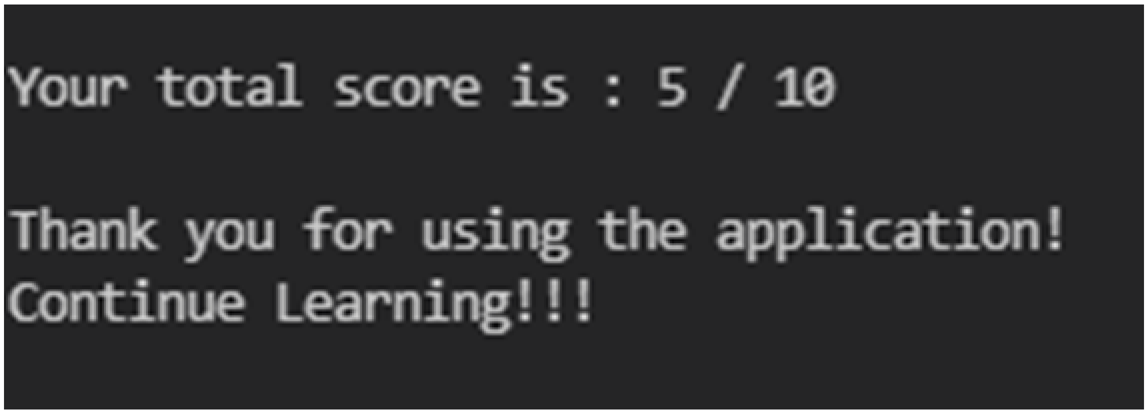


```
5. 6 / 3 = ?
   a) 2    b) 1    c) 12

Your Answer : b
The correct answer is a

6. 8 - 8 = ?
   a) -1    b) -2    c) 0

Your Answer : c
c is correct

7. 3 * 12 = ?
   a) 33    b) 36    c) 38

Your Answer : a
The correct answer is b
```



```
8. 9 * 9 = ?
   a) 72    b) 91    c) 81

Your Answer : b
The correct answer is c

9. 11 + 13 = ?
   a) 24    b) 26    c) 19

Your Answer : c
The correct answer is a

10. 56 / 8 = ?
   a) 7    b) 9    c) 6

Your Answer : a
a is correct
```

The program will continue to show the question and choices, and ask for the user's answer. The program will end at the 10th question.



```
Your total score is : 5 / 10

Thank you for using the application!
Continue Learning!!!
```

The program will display the accumulated score of the user. The program will end after displaying the following messages.

VI. Source Code

```
;  SUBJECT: Computer Organization and Assembly Language
;   COURSE AND SECTION: BSCS 3-3
;
;   PROJECT: Quiz App
;
;   Five (5) Members
;   Leader:
;       Rosario, Mark Edison
;   Member(s):
;       Constantino, Bismillah
;       Cube, Jeremy
;       Jizmundo, Piolo Brian
;       Tacata, Jericho Vince

SYS_EXIT  equ 1

SYS_WRITE equ 4
STDOUT    equ 1

SYS_READ  equ 3
STDIN     equ 2

;MACROS
%macro compare 4
    mov ecx, [%1]
    cmp ecx, %2
    JE  %3
    JNE %4
%endmacro

%macro eatbuffer 0
    ;READ and STORE USER INPUT TO NUM VARIABLE
    mov eax, SYS_READ
    mov ebx, STDIN
    mov ecx, ans
    mov edx, 1
    int 0x80
%endmacro

%macro write_MSG 2
    ;WRITE MESSAGE
    mov eax, SYS_WRITE          ;system call number (sys_write)
    mov ebx, STDOUT             ;file descriptor (stdout)
    mov ecx, %1                 ;message to write
    mov edx, %2                   ;message length
    int 0x80                      ;call kernel
%endmacro

%macro write_QA 6
    ;WRITE QUESTION, CHOICES, AND PROMPT USER INPUT

    ;WRITE Question
    mov eax, SYS_WRITE            ;system call number (sys_write)
    mov ebx, STDOUT               ;file descriptor (stdout)
    mov ecx, %1                   ;message to write
    mov edx, %2                   ;message length
    int 0x80                      ;call kernel

    ;WRITE Answers
    mov eax, SYS_WRITE            ;system call number (sys_write)
    mov ebx, STDOUT               ;file descriptor (stdout)
    mov ecx, %3                   ;message to write
    mov edx, %4                   ;message length
    int 0x80                      ;call kernel

    ;WRITE AnswerPrompt
    mov eax, SYS_WRITE            ;system call number (sys_write)
    mov ebx, STDOUT               ;file descriptor (stdout)
    mov ecx, %5                   ;message to write
    mov edx, %6                   ;message length
    int 0x80                      ;call kernel

    ;READ and STORE USER INPUT TO NUM VARIABLE
    mov eax, SYS_READ
    mov ebx, STDIN
    mov ecx, ans
    mov edx, 1
    int 0x80
%endmacro

%macro incBy1 1
    mov edx, [%1]
    inc edx
    mov [%1], edx
%endmacro

section .data                 ;Constants
    ans1 DB 'a', 0xa, 0xa
    ans2 DB 'b', 0xa, 0xa
    ans3 DB 'c', 0xa, 0xa
    ans4 DB 'b', 0xa, 0xa
```

```
ans5 DB 'a', 0xa, 0xa
ans6 DB 'c', 0xa, 0xa
ans7 DB 'b', 0xa, 0xa
ans8 DB 'c', 0xa, 0xa
ans9 DB 'a', 0xa, 0xa
ans10 DB 'a', 0xa, 0xa

score DB '0'
lenScore equ $ - score

incr DB '1'

HEADER1 DB 'Arithmetic Quiz App', 0xa, 0xa
lenH1 equ $ - HEADER1

;HEADER2 DB 'Press Enter to start the quiz : ', 0xa
;lenH2 equ $ - HEADER2

BODY1 DB 'Select the letter of the correct answer : ', 0xa, 0xa
lenB1 equ $ - BODY1

BODY2 DB 'Your Answer : ',
lenB2 equ $ - BODY2

BODYT DB ' is correct ', 0xa, 0xa
lenBT equ $ - BODYT

BODYF DB 'The correct answer is '
lenBF equ $ - BODYF

FOOTER1 DB 'Your total score is : '
lenF1 equ $ - FOOTER1

FOOTER2 DB ' / 10 ', 0xa, 0xa
lenF2 equ $ - FOOTER2

FOOTER3 DB 'Thank you for using the application!', 0xa
lenF3 equ $ - FOOTER3

FOOTER4 DB 'Continue Learning!!!', 0xa, 0xa
lenF4 equ $ - FOOTER4

Q1 DB '1. 2 + 3 = ?', 0xa
lenQ1 equ $ - Q1

QA1 DB '   a) 5    b) 6    c) 7', 0xa, 0xa
lenQA1 equ $ - QA1
```

```
Q2 DB '2. 5 + 6 = ?', 0xa
lenQ2 equ $ - Q2

QA2 DB '   a) 10    b) 11    c) 12', 0xa, 0xa
lenQA2 equ $ - QA2

Q3 DB '3. 15 - 12 = ?', 0xa
lenQ3 equ $ - Q3

QA3 DB '   a) 5    b) 1    c) 3', 0xa, 0xa
lenQA3 equ $ - QA3

Q4 DB '4. 3 * 6 = ?', 0xa
lenQ4 equ $ - Q4

QA4 DB '   a) 10    b) 18    c) 12', 0xa, 0xa
lenQA4 equ $ - QA4

Q5 DB '5. 6 / 3 = ?', 0xa
lenQ5 equ $ - Q5

QA5 DB '   a) 2    b) 1    c) 12', 0xa, 0xa
lenQA5 equ $ - QA5

Q6 DB '6. 8 - 8 = ?', 0xa
lenQ6 equ $ - Q6

QA6 DB '   a) -1    b) -2    c) 0', 0xa, 0xa
lenQA6 equ $ - QA6

Q7 DB '7. 3 * 12 = ?', 0xa
lenQ7 equ $ - Q7

QA7 DB '   a) 33    b) 36    c) 38', 0xa, 0xa
lenQA7 equ $ - QA7

Q8 DB '8. 9 * 9 = ?', 0xa
lenQ8 equ $ - Q8

QA8 DB '   a) 72    b) 91    c) 81', 0xa, 0xa
lenQA8 equ $ - QA8

Q9 DB '9. 11 + 13 = ?', 0xa
lenQ9 equ $ - Q9

QA9 DB '   a) 24    b) 26    c) 19', 0xa, 0xa
```

```
    lenQA9 equ $ - QA9

    Q10 DB '10. 56 / 8 = ?', 0xa
    lenQ10 equ $ - Q10

    QA10 DB '   a) 7    b) 9    c) 6', 0xa, 0xa
    lenQA10 equ $ - QA10


section .bss                ;Variables
    ans resb 1

section .text
    global _start           ;must be declared for using gcc

    _start:                 ;tell linker entry point

    write_MSG HEADER1, lenH1
    write_MSG BODY1, lenB1

    ;Q1
    write_QA Q1, lenQ1, QA1, lenQA1, BODY2, lenB2
    compare ans, 'a', True1, False1

    True1:
        incBy1 score
        write_MSG ans1, 1
        write_MSG BODYT, lenBT
        JMP _Qs2

    False1:
        write_MSG BODYF, lenBF
        write_MSG ans1, 3
        JMP _Qs2

    _Qs2:
        eatbuffer
        write_QA Q2, lenQ2, QA2, lenQA2, BODY2, lenB2
        compare ans, 'b', True2, False2

    True2:
        incBy1 score
        write_MSG ans2, 1
        write_MSG BODYT, lenBT
        JMP _Qs3

    False2:
```

```
    write_MSG BODYF, lenBF
    write_MSG ans2, 3
    JMP _Qs3

_Qs3:
    eatbuffer
    write_QA Q3, lenQ3, QA3, lenQA3, BODY2, lenB2
    compare ans, 'c', True3, False3

True3:
    incBy1 score
    write_MSG ans3, 1
    write_MSG BODYT, lenBT
    JMP _Qs4

False3:
    write_MSG BODYF, lenBF
    write_MSG ans3, 3
    JMP _Qs4

_Qs4:
    eatbuffer
    write_QA Q4, lenQ4, QA4, lenQA4, BODY2, lenB2
    compare ans, 'b', True4, False4

True4:
    incBy1 score
    write_MSG ans4, 1
    write_MSG BODYT, lenBT
    JMP _Qs5

False4:
    write_MSG BODYF, lenBF
    write_MSG ans4, 3
    JMP _Qs5

_Qs5:
    eatbuffer
    write_QA Q5, lenQ5, QA5, lenQA5, BODY2, lenB2
    compare ans, 'a', True5, False5

True5:
    incBy1 score
    write_MSG ans5, 1
    write_MSG BODYT, lenBT
    JMP _Qs6
```

```
False5:
    write_MSG BODYF, lenBF
    write_MSG ans5, 3
    JMP _Qs6

_Qs6:
    eatbuffer
    write_QA Q6, lenQ6, QA6, lenQA6, BODY2, lenB2
    compare ans, 'c', True6, False6

True6:
    incBy1 score
    write_MSG ans6, 1
    write_MSG BODYT, lenBT
    JMP _Qs7

False6:
    write_MSG BODYF, lenBF
    write_MSG ans6, 3
    JMP _Qs7

_Qs7:
    eatbuffer
    write_QA Q7, lenQ7, QA7, lenQA7, BODY2, lenB2
    compare ans, 'b', True7, False7

True7:
    incBy1 score
    write_MSG ans7, 1
    write_MSG BODYT, lenBT
    JMP _Qs8

False7:
    write_MSG BODYF, lenBF
    write_MSG ans7, 3
    JMP _Qs8

_Qs8:
    eatbuffer
    write_QA Q8, lenQ8, QA8, lenQA8, BODY2, lenB2
    compare ans, 'c', True8, False8

True8:
    incBy1 score
    write_MSG ans8, 1
    write_MSG BODYT, lenBT
    JMP _Qs9
```

```
False8:
    write_MSG BODYF, lenBF
    write_MSG ans8, 3
    JMP _Qs9

_Qs9:
    eatbuffer
    write_QA Q9, lenQ9, QA9, lenQA9, BODY2, lenB2
    compare ans, 'a', True9, False9

True9:
    incBy1 score
    write_MSG ans9, 1
    write_MSG BODYT, lenBT
    JMP _Qs10

False9:
    write_MSG BODYF, lenBF
    write_MSG ans9, 3
    JMP _Qs10

_Qs10:
    eatbuffer
    write_QA Q10, lenQ10, QA10, lenQA10, BODY2, lenB2
    compare ans, 'a', True10, False10

True10:
    incBy1 score
    write_MSG ans10, 1
    write_MSG BODYT, lenBT
    JMP _exit

False10:
    write_MSG BODYF, lenBF
    write_MSG ans10, 3
    JMP _exit

_exit:

    write_MSG FOOTER1, lenF1
    write_MSG score, lenScore
    write_MSG FOOTER2, lenF2
    write_MSG FOOTER3, lenF3
    write_MSG FOOTER4, lenF4
```

```
;EXIT
mov eax, SYS_EXIT           ;system call number (sys_exit)
int 0x80                    ;call kernel
```