Kulliyyah of Information and Communications Technology
Department of Computer Science

International Islamic University Malaysia
Semester II 2017/2018

Project Report

Computer Architecture

&

Assembly Language

CSC 3402

SECTION 02

GROUP NAME: Anything

LECTURER'S NAME: DR Hafizah Binti Mansor

Group Members

NAME	MATRIC NO.
Abid Ebna Saif Utsha	**1433527**
Mahfuzealahi Noman	**1515803**
Mohammad Nafees Bin Zaman	**1616357**
Sanjir Salsabil	**1526703**

Table of Contents:

1.0 Abstract:

As our life is getting more dependable on electronic things such as Mobile phones, Computers etc, we cannot think a single moment without those and so far, these things are very useful and easy to use. Even electronic transaction is a very popular way to transfer money from one account to another account is now just a matter of few seconds. It should also be mentioned that, we are more likely to spend times in social media. Even people are sharing their personal issues in social media rather than reality. In a nutshell, we can say that we are living in a virtual era. Virtual environment has a lot of advantages but one the major concern is how secure is that to share all the personal details in this environment. That is why people rely on putting a password or security gates by which they can store their data in a virtual environment without being stolen. As the security issue appears we wanted to implement a security interface which can be accessed only by the verified user.

1.1 Introduction:

Our Project is to make a security system that can only be accessed by verified user. As this course is about Computer Architecture & Assembly Language, so we have used assembly language (MIPS) to implement our desired work. MIPS assembly language is an instruction set & computer architecture based language for ***Microprocessor without Interlocked Pipeline Stages.*** It is a low-level language which was developed by John Hennessey. It has its own syntaxes. Unlikely to other programming languages like C++, Java, Python etc, it is not so flexible to use and that is why it is called a low-level language. In our system we try to utilize this language and make a user-friendly application.

1.2 Work Flow:

Our system is basically a general security system where users need to put their username and password to enter the system.

In the first step there will be a built-in security password which will be provided to the users who want to use our system. They must put the username and the password to enter the system.

There in the next step, they will find some options to do more with the system. One of the options are change password. So, if a user is not satisfied with the password or he/she thinks that the password is vulnerable then he/she can change the password.

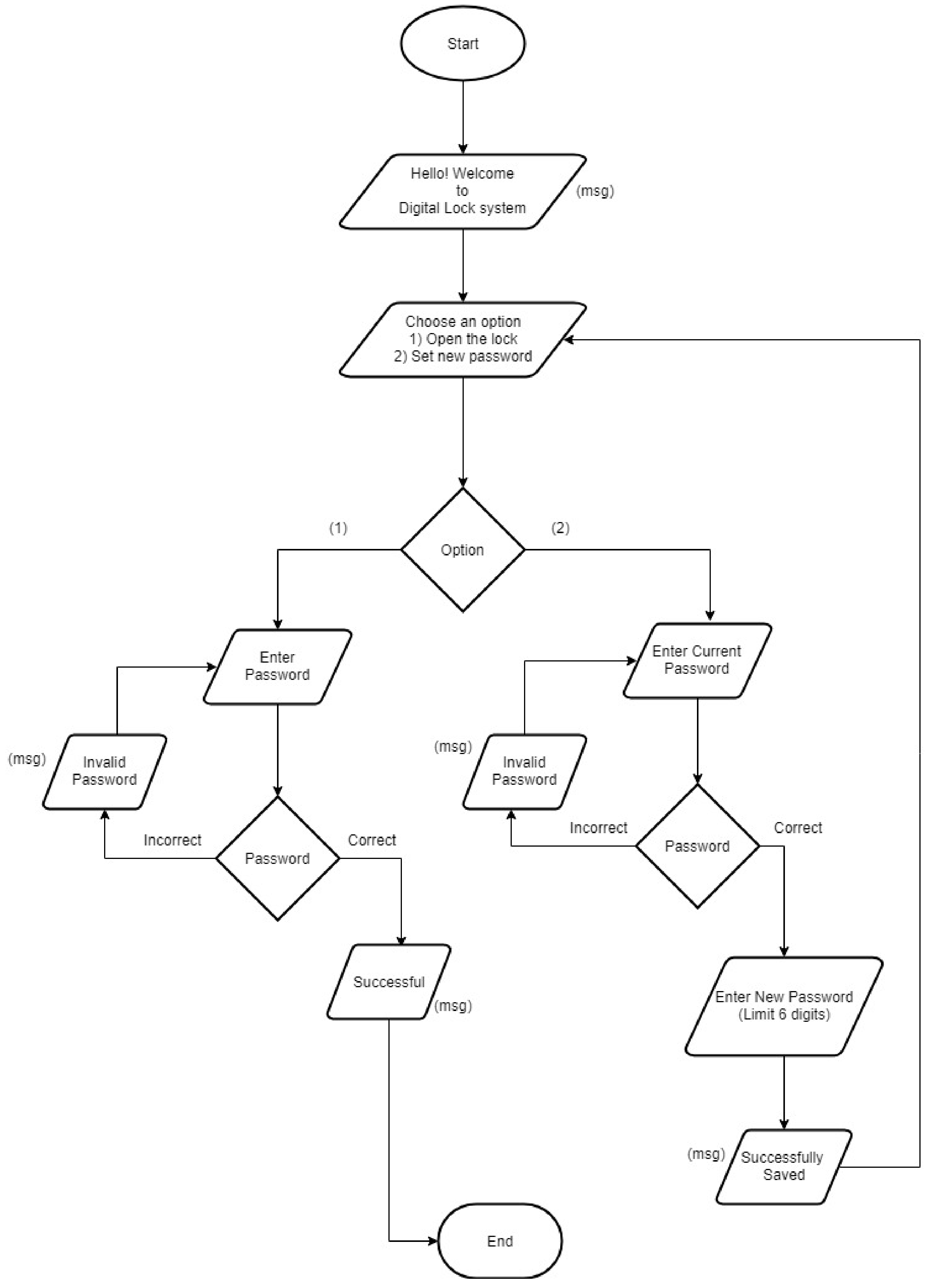
If someone unwanted enter the system and may try to change the password. That is why every time changing the password, user need to put his/her current password first. If it matches only then he/she can proceed.

The last step is to exit the system and all the data will be saved with the new password.

2.0 Pseudocode:

```
1) Show welcoming message
2) Prompt the user to input options either 'Open the lock' or 'Set new password'
3) If user selected option (1) then, ask the user to input password.
    a) If the password is correct, then open the lock.
    b) If the password is not correct then, ask the user to input password again,
       repeat this step until user put correct password.
4) If user selected option (2) then, ask the user to input the current password.
    a) If the password is not correct, then ask the user to input current
       password again, repeat this step until user put correct password
    b) If the password is correct, then prompt the user to input new password.
       Then go to step no. (2) and ask the option to user.
```

3.0 Flow-Chart:



4.0 Obstacles:

One of the major problems that we faced is in the coding part. As it is a low-level language we need to put a great effort to learn it. After that, when we code many times we couldn’t produce our desired result. It was a very early beginning for us. Slowly we cop up with it. Nowadays people are not often use this language so we did not get enough references or resources by which we can solve our mistakes.

4.1 References

4.2 Summary:

The main purpose of this course is to introduce our self with assembly language. And the project’s purpose is to learn, practice and know how to implement assembly language in a real-life problem. Working in this project, was really a good experience for all of use. We shared our thoughts about assembly language and acquire a practical knowledge of it which will help our future works in this area.

We have successfully implement our security system using assembly language (MIPS). The system can secure information, which was our main target. Though it is only a security part interface but it can be used in any large system to secure information and putting some more options will help this system to be a more flexible system to use.

Appendix A:

Source Code:

```
.data

    welmsg: .asciiz "Hello! Welcome to Digital Lock System\n"
    menu: .asciiz "\nChoose one option 1. Open the Lock 2. Set new Password\n"
    pass: .asciiz "Enter Password: "
    invalid_msg: .asciiz "\nInvalid Password..."
    success_msg: .asciiz "\nSuccessful..."
    curr_pass: .asciiz "\nEnter Current Password: "
    new_pass: .asciiz "\nEnter new Password: (Limit 6 digits)"
    set_msg: .asciiz "\nSuccessfully saved..."
    password: .asciiz "123456"

    s1:   .asciiz

.text

   main:
       li $v0, 4
       la $a0, welmsg          # print welcome message
       syscall

   backtomenu:
       li $v0, 4
       la $a0, menu            # print menu option
       syscall
       li $v0, 5               # read option
       syscall
       add $s1, $zero, $v0     # store in $s1
       addi $s6, $zero, 1
       addi $s7, $zero, 2

       beq $s1, $s6, optOne
       beq $s1, $s7, optTwo




   optOne:
       li $v0, 4
       la $a0, pass            # print password message
       syscall

       la $s0, password

        li $v0,8                       # Input the password
        la $a0,s1
        li $a1,7
        syscall
```

```
        la $s1,s1                           #compare
        addi $t0, $t0, 0
        jal compare

        beq $v0, $0, success                #go to success
        j invalid                   #go to invalid



compare:
       add $t1, $s0, $t0
       lbu $t2, 0($t1)
       add $t1, $s1, $t0
       lbu $t3, 0($t1)
       slt $t4, $t2, $t3
       bne $t4, $0, notequal
       slt $t4, $t3,$t2
       bne $t4, $0, notequal
       beq $t2, $zero, equal

       addi $t0, $t0, 1 # i = i + 1
       j compare # next iteration of loop




notequal:
       addi $v0, $0, -1
       jr $ra
```

```
equal:
      addi $v0, $0, 0
      jr $ra



success:
     li $v0, 4
     la $a0, success_msg          # print success message
     syscall


     j exit


 invalid:
     li $v0, 4
     la $a0, invalid_msg          # print success message
     syscall


     j optOne


 exit:
       li $v0, 10
     syscall
```

```
optTwo:
      li $v0, 4
    la $a0, curr_pass           # print current password message
    syscall



    la $s0, password


      li $v0,8                        # Input the password
      la $a0,s1
      li $a1,7
      syscall



      la $s1,s1                                 #compare
      addi $t0, $t0, 0
      jal compare

      beq $v0, $0, currS                  #go to success
      j invalidS                      #go to invalid




currS:
    li $v0, 4
    la $a0, new_pass          # print success message
    syscall

    li $v0,8                            # Input the password
      la $a0,password
      li $a1,7
      syscall

    li $v0, 4
    la $a0, set_msg          # print success message
    syscall



    j backtomenu

invalidS:
    li $v0, 4
    la $a0, invalid_msg          # print success message
    syscall

    j optTwo
```