



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

LAB REPORT NO: 02
Course Title: Compiler Lab
Course Code: CSE-304 Section:213-D1

Lab Experiment Name: Implementation of conditional statement using assembly language.

Student Details

Name		ID
1.	Irteja Mahmud	213902016

Lab Date : 12-10-2023
Submission Date : 19-10-2023
Course Teacher's Name : Sudip Ghoshal

Lab Report Status

Marks:
Comments:.....

Signature:.....
Date:.....

1. TITLE OF THE LAB EXPERIMENT

Implementation of conditional statement using assembly language.

2. OBJECTIVES/AIM

The objective of this program is to know about the use and implementation of conditional statement using assembly language.

3.PROCEDURE / ANALYSIS / DESIGN

Problem 1: Print two characters in reverse alphabetic order using assembly language.

Step 1: Initialize the program and memory.

- Set the program's origin to 100h.
- Define the memory model as "small."
- Allocate a stack of 100h bytes.

Step 2: Define data section.

- Declare variables in the data section:

Step 3: Start of the code section.

Step 4: Setup data segment and display the first prompt.

- Load the data segment address into AX from '@data'.
- Set DS (data segment) to the value in AX.
- Set AH to 9h (print string function).
- Load the offset address of 'txt1' into DX.
- Trigger interrupt 21h to display the "Input your first char: \$" message.

Step 5: Read the first character.

- Set AH to 1h (read character function).
- Trigger interrupt 21h to read a character from the user.
- Store the character in BL.

Step 6: Display a newline.

- Set AH to 9h (print string function).
- Load the offset address of 'next' into DX.
- Trigger interrupt 21h to print a newline and carriage return.

Step 7: Check if the character is a vowel.

- Compare the character in AL (copied from BL) with a set of vowel characters (both uppercase and lowercase).
- If the character matches a vowel, jump to the 'VL' (vowel) label.

Step 8: Display a message for consonant.

- Load the offset address of 'CONSONANT' into DX.
- Set AH to 9h (print string function).
- Trigger interrupt 21h to display the "CONSONANT" message.
- Return from the 'main' procedure using 'ret.'

Step 9: Display a message for vowel.

- The program reaches this point if the character is a vowel.
- Load the offset address of 'VOWEL' into DX.
- Set AH to 9h (print string function).
- Trigger interrupt 21h to display the "VOWEL" message.

Step 10: End of the main procedure.

- End the 'main' procedure using 'main endp.'

Step 11: End of the program.

- Mark the end of the program using 'end main.'

Problem 2: Take a number from user, print whether the given number is odd or even.

Step 1: Initialize the program and memory.

Step 2: Define data section.

- Declare variables in the data section:

Step 3: Start of the code section.

Step 4: Setup data segment and display the first prompt.

- Load the data segment address into AX from '@data'.
- Set DS (data segment) to the value in AX.
- Set AH to 9h (print string function).
- Load the offset address of 'txt1' into DX.
- Trigger interrupt 21h to display the "Input your first char: \$" message.

Step 5: Read the first character.

- Set AH to 1h (read character function).
- Trigger interrupt 21h to read a character from the user.
- Store the character in BL.

Step 6: Display a newline.

- Set AH to 9h (print string function).
- Load the offset address of 'next' into DX.

- Trigger interrupt 21h to print a newline and carriage return.

Step 7: Check if the character is a vowel.

- Compare the character in AL (copied from BL) with a set of vowel characters (both uppercase and lowercase).
- If the character matches a vowel, jump to the 'VL' (vowel) label.

Step 8: Display a message for consonant.

- Load the offset address of 'CONSONANT' into DX.
- Set AH to 9h (print string function).
- Trigger interrupt 21h to display the "CONSONANT" message.
- Return from the 'main' procedure using 'ret.'

Step 9: Display a message for vowel.

- The program reaches this point if the character is a vowel.
- Load the offset address of 'VOWEL' into DX.
- Set AH to 9h (print string function).
- Trigger interrupt 21h to display the "VOWEL" message.

Step 10: End of the main procedure.

- End the 'main' procedure using 'main endp.'

Step 11: End of the program.

- Mark the end of the program using 'end main.'

Problem 3: Find out the largest number between two numbers using assembly language.

Step 1: Initialize the program and data section.

- Set the program's origin to 100h.
- Define the data section and declare variables:

Step 2: Start of the code section.

Step 3: Setup data segment and display the input prompt.

- Load the data segment address into AX from '@data'.
- Set DS (data segment) to the value in AX.
- Set AH to 9h (print string function).
- Load the offset address of 'txt1' into DX.
- Trigger interrupt 21h to display the "Input A Number: \$" message.

Step 4: Read a character from the user.

- Set AH to 01h (read character function).
- Trigger interrupt 21h to read a character from the user.
- Store the character in BL.

Step 5: Check if the number is divisible by 5.

- Copy the value from BL to AL for further processing.
- Load the value 5 into BH.
- Perform unsigned division (DIV) of AL by BH.
- Check the result in AH (remainder):
 - If AH is zero, jump to label 'l1' (number is divisible by 5).

Step 6: Display a message when the number is not divisible by 5.

- Set AH to 9h (print string function).
- Load the offset address of 'next' into DX to print a newline.
- Load the offset address of 'txt3' into DX to display the "Number is not Divisible By 5" message.
- Return from the 'main' procedure.

Step 7: Display a message when the number is divisible by 5.

- Set AH to 9h (print string function).
- Load the offset address of 'next' into DX to print a newline.
- Load the offset address of 'txt2' into DX to display the "Number is Divisible By 5" message.
- Return from the 'main' procedure.

Step 8: End of the program.

- Mark the end of the 'main' procedure using 'main endp.'
- End the 'code' section and the program using 'end main.'

4.IMPLEMENTATION

Problem 1:

```
org 100h
.model small
.stack 100h
.data

    msg1 db ?
    msg2 db ?
    txt1 dw "Input your first char: $"
    txt2 dw "Input your second char: $"
    txt3 dw "Sorted Oorder: $"
    next db 013,010,"$"
    CONSONANT db 0DH,0AH,'CONSONANT$'
    VOWEL db 0DH,0AH,'VOWEL$'

.code
main proc

    mov ax,@data
    mov ds,ax

    mov ah , 9h
    lea dx , txt1        ;displaying text 1
    int 21h              ;Input your first char:

    mov ah,1h
    int 21h              ;input
    mov bl, al

    mov ah , 9h
    lea dx , next        ;printing newline
    int 21h

    MOV AL,BL
    CMP AL,'A'
    JE VL
    CMP AL,'E'
    JE VL
    CMP AL,'I'
    JE VL
```

```

    CMP AL, 'O'
    JE VL                                ;checking vowel
    CMP AL, 'U'
    JE VL
    CMP AL, 'a'
    JE VL
    CMP AL, 'e'
    JE VL
    CMP AL, 'i'
    JE VL
    CMP AL, 'o'
    JE VL
    CMP AL, 'u'
    JE VL

    LEA DX, CONSONANT
    MOV AH, 9
    INT 21H
    ret

VL:
    LEA DX, VOWEL
    MOV AH, 9
    INT 21H

```

```

    main endp
end main

```

Problem 2:

```

org 100h
.data
txt1 dw "Input: $"
msg1 db 'Character$'
msg2 db 'Digit$'
msg3 db 'Others$'
next db 013, 010, "$"

.code

main proc
    mov ax, @data

```

```

mov ds, ax

mov ah, 9h
lea dx, txt1      ;displaying text 1
int 21h

                                ;input
mov ah, 01h
int 21h

mov bl, al      ; move A to bl register

cmp bl, 30h      ; compare bl with ascii value of 01
jge print       ; if greater or equal jump to label print

print:          ; label print

cmp bl, 39h      ; compare bl with ascii value of 09
jle print1      ; if less or equal jump to label print1
jg end          ; if greater or equal jump to label print

                                ; new line

mov ah, 02h      ; character output function
mov dl, 0Ah      ; line feed
int 21h          ; dos interrupt
mov dl, 0Dh      ; Carriage return
int 21h          ; dos interrupt

print1:
mov ax, @data
mov ds, ax

mov ah, 09h
lea dx, next
int 21h

mov ah, 09h
lea dx, msg2
int 21h
ret

```



```

end:

cmp bl, 41h
jge print2

print2:
cmp bl, 5Ah
jle print3
jge end1

print3:
mov ax, @data
mov ds, ax

mov ah, 09h
lea dx, next
int 21h

mov ah, 09h
lea dx, msg1
int 21h
ret

end1:
mov ax, @data
mov ds, ax

mov ah, 09h
lea dx, next
int 21h

mov ah, 09h
lea dx, msg3
int 21h
ret
main endp
endp

```

Problem 3:

```

org 100h
.model small
.stack 100h

```

.data

```
txt1 dw "Input A Number: $"
txt2 dw "Number is Divisible By 5$"
txt3 dw "Number is not Divisible By 5$"
next db 013,010,"$"
```

.code

main proc

```
mov ax,@data
mov ds,ax
```

```
mov ah , 9h
lea dx , txt1      ;displaying text 1
int 21h            ;Input A Number:
```

```
mov ah,1h
int 21h            ;input1
mov bl, al
```

```
mov ah , 9h
lea dx , next      ;printing newline
int 21h
```

```
mov al,bl
mov bh,5
div bh
```

```
cmp ah,0
je l1
```

```
mov ah , 9h
lea dx , next      ;displaying text 1
int 21h
```

```
mov ah , 9h
lea dx , txt3      ;displaying text 1
int 21h
```

```
ret
```

```
l1:
```

```
mov ah , 9h
lea dx , next      ;displaying text 1
int 21h
```

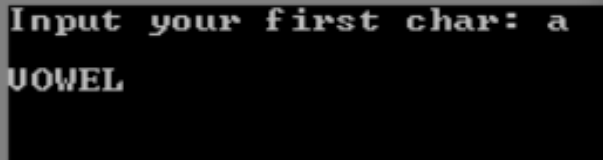
```
    mov ah , 9h
    lea dx , txt2      ;displaying text 1
    int 21h
    ret

main endp
end main
```

5.TEST RESULT / OUTPUT

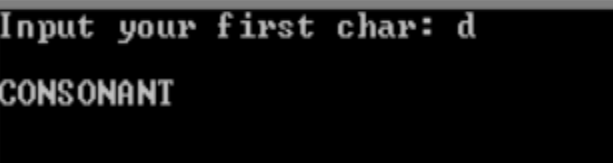
Problem 1 Output:

SCA emulator screen (80x25 chars)



Input your first char: a
VOWEL

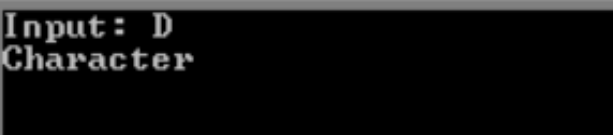
SCA emulator screen (80x25 chars)



Input your first char: d
CONSONANT

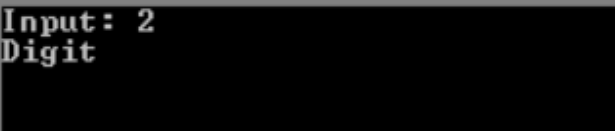
Problem 2 Output:

SCA emulator screen (80x25 chars)



Input: D
Character

SCA emulator screen (80x25 chars)



Input: 2
Digit

Problem 3 Output:

SCM emulator screen (80x25 chars)

```
Input A Number: 6  
Number is not Divisible By 5
```

SCM emulator screen (80x25 chars)

```
Input A Number: 5  
Number is not Divisible By 5
```

6.ANALYSIS AND DISCUSSION

In this lab report, we will analyze and reflect upon the three assembly language problems we've encountered. These problems serve as practical examples of how assembly language can be used for simple decision-making and condition-based output.

In Problem 1, the objective was to create an assembly program that identifies whether a given character is a vowel or a consonant. The program utilizes a straightforward approach by comparing the input character to a predefined set of vowel characters in both uppercase and lowercase forms. If the character matches any of the vowels, it is labeled as a "VOWEL," and if not, it is identified as a "CONSONANT.".

Problem 2 tasked us with designing an assembly program that identifies whether an input character is a digit, a letter, or neither. The program employs ASCII range comparisons to distinguish between digits and letters. If the input character falls within the ASCII range of digits (0-9), it is labeled as a "Digit." Otherwise, it is categorized as a "Character."

Problem 3 involves creating an assembly program to ascertain whether a given number is divisible by 5. The program takes an integer input and performs a division operation, checking whether the remainder is zero. If the remainder is zero, the program declares "Number is Divisible By 5"; otherwise, it states "Number is not Divisible By 5."

7. SUMMARY:

In all three problems, the programs follow a similar structure, which includes displaying prompts for user input, processing the input, and displaying the output based on specific conditions. These problems demonstrate how assembly language can be used for basic decision-making and conditional output.