UNIVERSITY OF MALAYA

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITI MALAYA

SEMESTER 2, SESSION 2023/2024
WIA1003 COMPUTER SYSTEMS AND ARCHITECTURE
LAB REPORT

Occ	3		
Members	No.	Name	Matric No.
	1.	NAZRUL IKRAM BIN NAZARULANUAR	23054536
	2.	TENGKU ANAS ZAINAL ABIDIN	22002125
	3.	DENNIS AIMIN OON BIN JEFFREY OON	22001610
Lecturer	MR. EMRAN BIN MOHD TAMIL		

Introduction

This report covers the implementation and observations of two assembly language programs designed to fulfill specific tasks using indexed addressing and register operations. The assignments involved calculating the sum of gaps between elements in an array and generating a sequence of numbers. The provided solutions are written in x86 assembly language using Irvine32 library functions.

Observations

Task 1: Sum of Gaps Between Array Elements

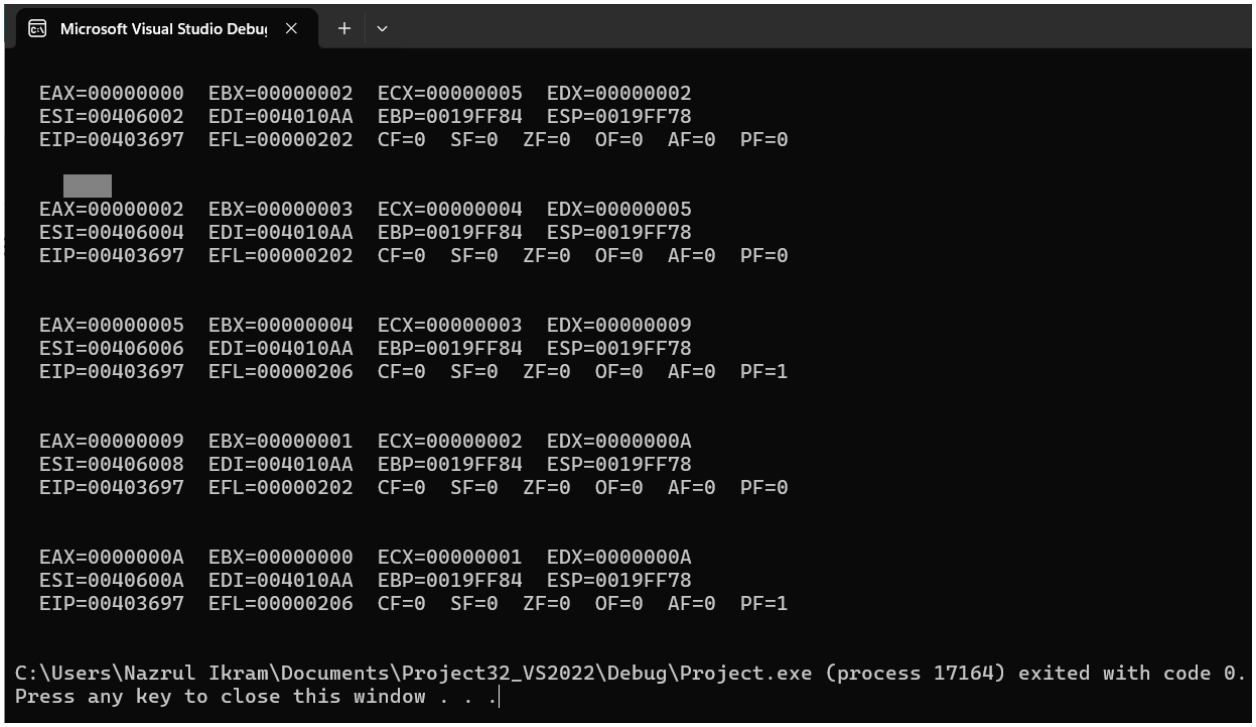
Objective: Write a program with indexed addressing to calculate the sum of all gaps between array elements. The array is of word type, sequenced in non-decreasing order. The specific array given is {0, 2, 5, 9, 10}, with gaps of 2, 3, 4, and 1, summing up to 10.

AddTwo.asm

```
1   INCLUDE Irvine32.inc        ;QUESTION 1
2
3   .data
4   array WORD 0,2,5,9,10           ; Array Initialization with 5 elements
5   sum DWORD 0                 ; Variable to store the sum of differences
6
7   .code
8   main PROC
9       MOV ECX, LENGTHOF array     ; Load the length of the array (number of elements) into ECX
10      MOV ESI, OFFSET array       ; Load the address of the first element of the array into ESI
11      MOV EAX, 0                  ; Initializes EAX to 0
12      MOV EBX, 0                  ; Initializes EBX to 0
13      MOV EDX, 0                  ; Initializes EDX to 0
14
15  L1:
16      MOV AX, [ESI]               ; Move the current element pointed to by ESI into AX
17      MOV BX, [ESI + TYPE array]  ; Move the next element in the array into BX
18      SUB BX, AX                  ; Subtract the current element (AX) from the next element (BX)
19      ADD sum, EBX                ; Add the result of the subtraction to the sum
20      MOV EDX, sum                ; Moves the current value of sum into the EDX
21      ADD ESI, TYPE array         ; Move ESI to point to the next element in the array
22      call DumpRegs               ; Displays the current state of the registers
23      Loop L1                     ; Decrement ECX and repeat the loop if ECX is not zero
24
25  exit
26  main ENDP
27  END main
28
```

- The program initializes an array with the given elements.
- It uses a loop to iterate through the array, calculating the difference between consecutive elements.
- The differences are accumulated into a variable sum.
- The loop utilizes the ESI register to point to the current element and moves to the next element in each iteration.
- The DumpRegs function is called for debugging purposes to display the current state of the registers.

Output:


Microsoft Visual Studio Debu
EAX=00000000 EBX=00000002 ECX=00000005 EDX=00000002
ESI=00406002 EDI=004010AA EBP=0019FF84 ESP=0019FF78
EIP=00403697 EFL=00000202 CF=0 SF=0 ZF=0 OF=0 AF=0 PF=0
EAX=00000002 EBX=00000003 ECX=00000004 EDX=00000005
ESI=00406004 EDI=004010AA EBP=0019FF84 ESP=0019FF78
EIP=00403697 EFL=00000202 CF=0 SF=0 ZF=0 OF=0 AF=0 PF=0
EAX=00000005 EBX=00000004 ECX=00000003 EDX=00000009
ESI=00406006 EDI=004010AA EBP=0019FF84 ESP=0019FF78
EIP=00403697 EFL=00000206 CF=0 SF=0 ZF=0 OF=0 AF=0 PF=1
EAX=00000009 EBX=00000001 ECX=00000002 EDX=0000000A
ESI=00406008 EDI=004010AA EBP=0019FF84 ESP=0019FF78
EIP=00403697 EFL=00000202 CF=0 SF=0 ZF=0 OF=0 AF=0 PF=0
EAX=0000000A EBX=00000000 ECX=00000001 EDX=0000000A
ESI=0040600A EDI=004010AA EBP=0019FF84 ESP=0019FF78
EIP=00403697 EFL=00000206 CF=0 SF=0 ZF=0 OF=0 AF=0 PF=1
C:\Users\Nazrul Ikram\Documents\Project32_VS2022\Debug\Project.exe (process 17164) exited with code 0.
Press any key to close this window . . .

Task 2: Generate Sequence of Numbers

Objective: Write an assembly code to generate a sequence of numbers when a number is initialized. Starting with the number 8, the program should generate sequences as shown in the assignment details.

AddTwo.asm

```
INCLUDE Irvine32.inc    ;QUESTION 2

.data
num BYTE 8          ; Defines a byte-sized variable named num and initializes it with the value 8.

.code
main PROC
    MOV EAX,0       ; Initializes EAX register to 0 (Display value).
    MOV EBX,0       ; Initializes EBX register to 0 (Initiate value).
    MOV ECX,0       ; Initializes ECX register to 0 (Outer loop counter for L1).
    MOV EDX,0       ; Initializes EDX register to 0 (Inner loop counter for L2).
    MOVZX EDX,num   ; Loads the value of num (8) into EDX register, zero-extending the byte to a double word.
    MOVZX ECX,num   ; Loads the value of num (8) into ECX register, zero-extending the byte to a double word.

L1:
    MOV EAX, EBX    ; Moves the value of EBX into EAX. Initially, both are 0.
    MOV EDX, ECX    ; Moves the value of ECX into EDX. Initially, ECX is 8 (from num).

L2:                 ; Display value.
    INC EAX         ; Increments the value in EAX by 1.
    call WriteDec   ; Print the value in EAX as a decimal number.
    DEC EDX         ; Decrements the value in EDX by 1.
    JNZ L2          ; Jumps back to the L2 label if EDX is not zero.

call Crlf           ; Print a newline.
INC EBX             ; Increments the value in EBX by 1.
DEC ECX             ; Decrements the value in ECX by 1.
JNZ L1              ; Jumps back to the L1 label if ECX is not zero.

exit
main ENDP
END main
```

- The program initializes the variable num with the value 8.
- It uses two nested loops to generate and print the sequences.
- The outer loop (L1) controls the starting point of each sequence.
- The inner loop (L2) generates and prints the current sequence.
- The WriteDec function is used to print the current value of EAX, and Crlf prints a newline after each sequence.
- The INC and DEC instructions are used to increment and decrement the values in the registers for controlling the loops.

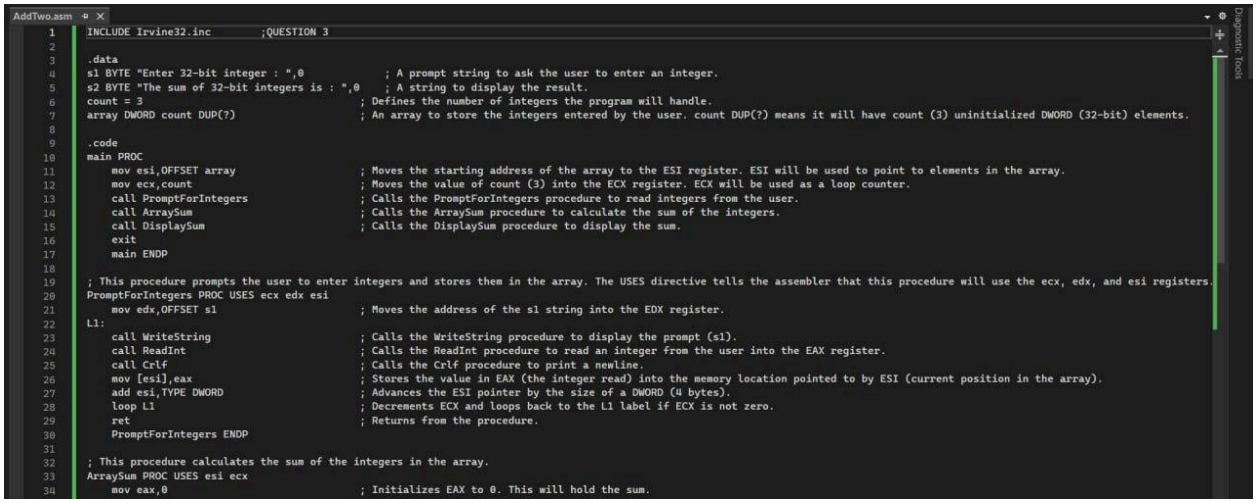
Output:

Microsoft Visual Studio Debu

```
12345678
2345678
345678
45678
5678
678
78
8

C:\Users\Nazrul Ikram\Documents\Project32_VS2022\Debug\Project.exe (process 13464) exited with code 0.
Press any key to close this window . . .
```

Task 3: Sum of User-Input Integers

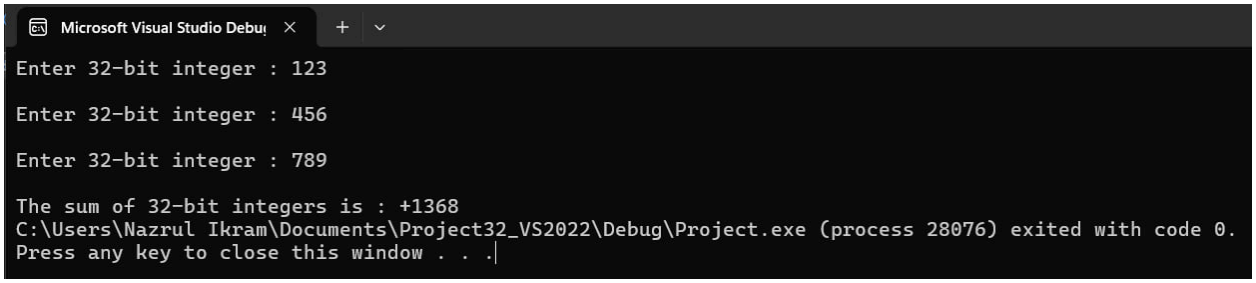
Objective: Write a program that prompts the user for three 32-bit integers, stores them in an array, calculates the sum of the array, and displays the sum on the screen.



```
INCLUDE Irvine32.inc        ;QUESTION 3

.data
s1 BYTE "Enter 32-bit integer : ",0            ; A prompt string to ask the user to enter an integer.
s2 BYTE "The sum of 32-bit integers is : ",0    ; A string to display the result.
count = 3                                      ; Defines the number of integers the program will handle.
array DWORD count DUP(?)                       ; An array to store the integers entered by the user. count DUP(?) means it will have count (3) uninitialized DWORD (32-bit) elements.

.code
main PROC
    mov esi,OFFSET array                       ; Moves the starting address of the array to the ESI register. ESI will be used to point to elements in the array.
    mov ecx,count                              ; Moves the value of count (3) into the ECX register. ECX will be used as a loop counter.
    call PromptForIntegers                     ; Calls the PromptForIntegers procedure to read integers from the user.
    call ArraySum                              ; Calls the ArraySum procedure to calculate the sum of the integers.
    call DisplaySum                            ; Calls the DisplaySum procedure to display the sum.
    exit
    main ENDP

; This procedure prompts the user to enter integers and stores them in the array. The USES directive tells the assembler that this procedure will use the ecx, edx, and esi registers.
PromptForIntegers PROC USES ecx edx esi
    mov edx,OFFSET s1                          ; Moves the address of the s1 string into the EDX register.
L1:
    call WriteString                           ; Calls the WriteString procedure to display the prompt (s1).
    call ReadInt                               ; Calls the ReadInt procedure to read an integer from the user into the EAX register.
    call Crlf                                  ; Calls the Crlf procedure to print a newline.
    mov [esi],eax                              ; Stores the value in EAX (the integer read) into the memory location pointed to by ESI (current position in the array).
    add esi,TYPE DWORD                         ; Advances the ESI pointer by the size of a DWORD (4 bytes).
    loop L1                                    ; Decrements ECX and loops back to the L1 label if ECX is not zero.
    ret                                        ; Returns from the procedure.
    PromptForIntegers ENDP

; This procedure calculates the sum of the integers in the array.
ArraySum PROC USES esi ecx
    mov eax,0                                  ; Initializes EAX to 0. This will hold the sum.
L1:
    add eax,[esi]                              ; Adds the value pointed to by ESI (current array element) to EAX.
    add esi,TYPE DWORD                         ; Advances the ESI pointer by the size of a DWORD (4 bytes).
    loop L1                                    ; Decrements ECX and loops back to the L1 label if ECX is not zero.
    ret                                        ; Returns from the procedure with the sum in EAX.
    ArraySum ENDP

; This procedure displays the sum of the integers.
DisplaySum PROC USES edx
    mov edx,OFFSET s2                          ; Moves the address of the s2 string into the EDX register.
    call WriteString                           ; Calls the WriteString procedure to display the result prompt (s2).
    call WriteInt                              ; Calls the WriteInt procedure to display the integer in EAX (the sum).
    ret                                        ; Returns from the procedure.
    DisplaySum ENDP
    END main
```

- The program initializes prompts and prepares an array to store user-input integers.
- It uses a procedure to prompt the user to enter integers and stores them in the array.
- Another procedure calculates the sum of the integers in the array.
- A final procedure displays the calculated sum.

Output:



```
Microsoft Visual Studio Debu
Enter 32-bit integer : 123

Enter 32-bit integer : 456

Enter 32-bit integer : 789

The sum of 32-bit integers is : +1368
C:\Users\Nazrul Ikram\Documents\Project32_VS2022\Debug\Project.exe (process 28076) exited with code 0.
Press any key to close this window . . .
```

Task 4: Calculate Grade Based on User Input

Objective: Write a program that receives an integer value between 0 and 100, then displays a single capital letter on the screen based on the range the integer falls within.

AddTwo.asm

```
INCLUDE Irvine32.inc    ;QUESTION 4

.data
prompt BYTE "Enter mark (0-100): ",0  ; Prompt the user and display the result.
result BYTE "Grade: ",0               ; Prompt the user and display the result.
invalidMsg BYTE "Invalid input !", 0
num DWORD ?                           ; 32-bit memory location to store the integer entered by the user.

.code
main PROC
    call GetValue                    ; Call GetValue to get a valid mark from the user.
    call CalculateGrade              ; Calls the CalculateGrade procedure to determine the letter grade.
    exit                             ; Exit the program
main ENDP

GetValue PROC
    promptLoop:
        mov edx, OFFSET prompt       ; Moves the address of prompt into edx.
        call WriteString             ; Calls WriteString to display the prompt "Enter mark (0-100): ".
        call ReadInt                 ; Calls ReadInt to read an integer from the user, which stores the result in the eax register.

        ; Validate input
        cmp eax, 0                  ; Compare input with 0
        jl invalidInput             ; Jump if input is less than 0
        cmp eax, 100                ; Compare input with 100
        jg invalidInput             ; Jump if input is greater than 100

        mov num, eax                ; Moves the value in eax (the entered integer) into the memory location num.
        ret                         ; Return from GetValue if input is valid

    invalidInput:
        mov edx, OFFSET invalidMsg  ; Moves the address of invalidMsg into edx.
        call WriteString            ; Display invalid input message.
        call Crlf                   ; Move to a new line.
        jmp promptLoop              ; Jump back to promptLoop

GetValue ENDP
```

```

CalculateGrade PROC
    mov eax, num                     ; Move the valid input into eax for grade calculation.

    .IF eax>=90
        mov al,'A'                  ; Sets 'A' to al.
    .ELSEIF eax>=80
        mov al,'B'                  ; Sets 'B' to al.
    .ELSEIF eax>=70
        mov al,'C'                  ; Sets 'C' to al.
    .ELSEIF eax>=60
        mov al,'D'                  ; Sets 'D' to al.
    .ELSE
        mov al,'F'                  ; Sets 'F' to al.
    .ENDIF                          ; End if condition

    mov edx, OFFSET result          ; Moves the address of result into edx.
    call WriteString                ; Calls WriteString to display "Grade: ".
    call WriteChar                  ; Calls WriteChar to display the character in al (the letter grade).
    ret
CalculateGrade ENDP

END main
```

- The program initializes prompts and prepares a variable to store the user's input.
- It reads an integer value from the user and stores it.
- A procedure compares the entered integer against predefined ranges and determines the corresponding letter grade.
- The determined grade is then displayed on the screen.

Output:


Microsoft Visual Studio Debu
Enter mark (0-100): -10
Invalid input !
Enter mark (0-100): 123
Invalid input !
Enter mark (0-100): 86
Grade: B
C:\Users\Nazrul Ikram\Documents\Project32_VS2022\Debug\Project.exe (process 2336) exited with code 0.
Press any key to close this window . . .

Conclusion

The provided assembly language programs successfully achieve the objectives set out in the lab assignment. The first program calculates the sum of gaps between array elements using indexed addressing, while the second program generates a sequence of numbers based on the initial value. The third program sums user-input integers and displays the result, and the fourth program calculates and displays grades based on user input. All implementations demonstrate fundamental concepts of assembly language programming, including loop control, register operations, conditional statements, and the use of Irvine32 library functions for input and output.