Republic of the Philippines
CAMARINES SUR POLYTECHNIC COLLEGES
Nabua, Camarines Sur

ISO 9001:2015

COLLEGE *of* COMPUTER STUDIES

CS 318 – Architecture and Organization

LEARNING TASK (BASIC ASSEMBLY INSTRUCTIONS PART 2)

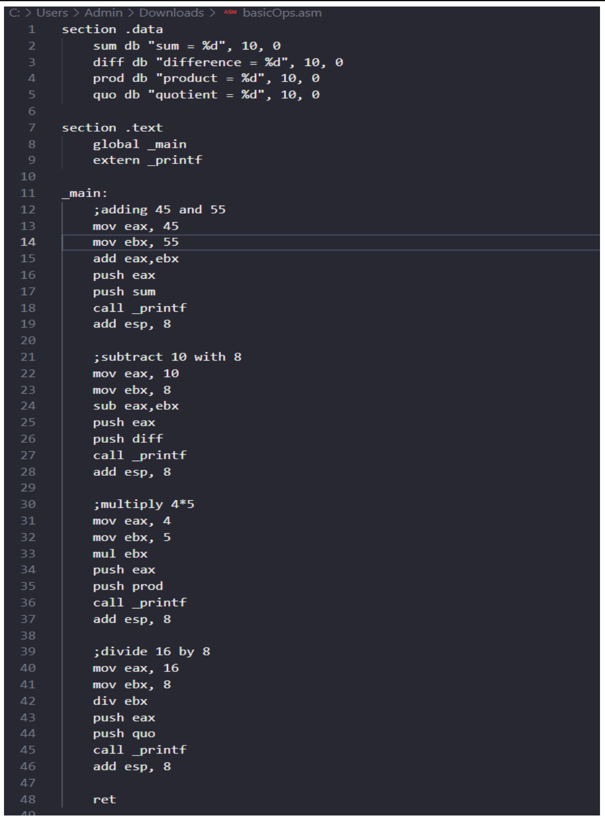
NAME: JOSHUA JERICHO D. BARJA SECTION: BSCS-3A

SAMPLE RUN

Step-by-step sample run of your assembly program with explanation.

Step 1

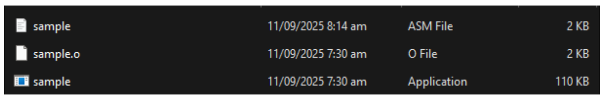
Create your assembly code that is properly working. It should use 32-bit architecture only.



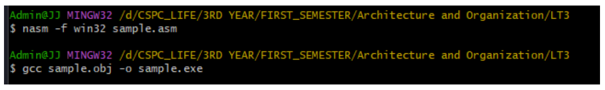
```
C: > Users > Admin > Downloads > basicOps.asm
section .data
    sum db "sum = %d", 10, 0
    diff db "difference = %d", 10, 0
    prod db "product = %d", 10, 0
    quo db "quotient = %d", 10, 0

section .text
    global _main
    extern _printf

_main:
    ;adding 45 and 55
    mov eax, 45
    mov ebx, 55
    add eax,ebx
    push eax
    push sum
    call _printf
    add esp, 8

    ;subtract 10 with 8
    mov eax, 10
    mov ebx, 8
    sub eax,ebx
    push eax
    push diff
    call _printf
    add esp, 8

    ;multiply 4*5
    mov eax, 4
    mov ebx, 5
    mul ebx
    push eax
    push prod
    call _printf
    add esp, 8

    ;divide 16 by 8
    mov eax, 16
    mov ebx, 8
    div ebx
    push eax
    push quo
    call _printf
    add esp, 8

    ret
```

Step 2

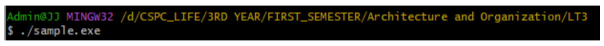
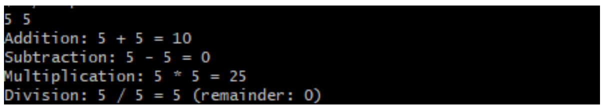
1. *Create file that has a .asm extension (e.g, sample.asm)*
2. *In your dedicated terminal, run the command <nasm -f win32 sample.asm -o sample.o> then hit enter.*
3. *After the first bash/command, type in another bash/command to perform linking <gcc sample.asm -o sample.exe> then press enter to execute.*



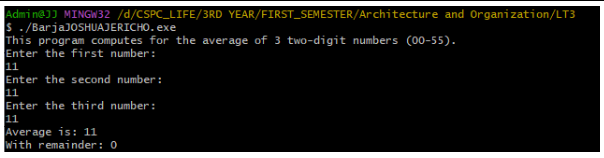
sample	11/09/2025 8:14 am	ASM File	2 KB
sample.o	11/09/2025 7:30 am	O File	2 KB
sample	11/09/2025 7:30 am	Application	110 KB

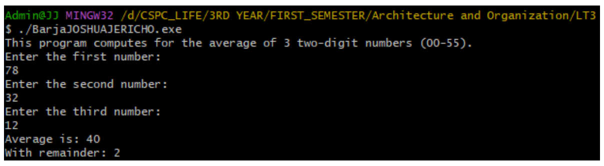


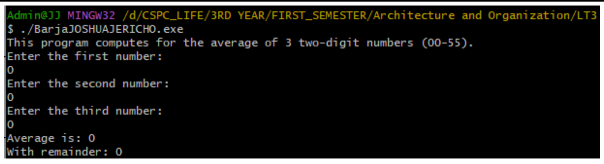
```
Admin@JJ MINGW32 /d/CSPC_LIFE/3RD YEAR/FIRST_SEMESTER/Architecture and Organization/LT3
$ nasm -f win32 sample.asm

Admin@JJ MINGW32 /d/CSPC_LIFE/3RD YEAR/FIRST_SEMESTER/Architecture and Organization/LT3
$ gcc sample.obj -o sample.exe
```

Step 3
1. *Execute the exe file using this bash/command <./sample.exe>*
2. *Check the code for errors.*
	 `Admin@JJ MINGW32 /d/CSPC_LIFE/3RD YEAR/FIRST_SEMESTER/Architecture and Organization/LT3`
	`$ ./sample.exe`
	 `5 5`
	`Addition: 5 + 5 = 10`
	`Subtraction: 5 - 5 = 0`
	`Multiplication: 5 * 5 = 25`
	`Division: 5 / 5 = 5 (remainder: 0)`

TEST CASES (Actual program)	*OUTPUT*

Input (11, 11, 11)	 `Admin@JJ MINGW32 /d/CSPC_LIFE/3RD YEAR/FIRST_SEMESTER/Architecture and Organization/LT3`
	`$ ./BarjaJOSHUAJERICHO.exe`
	`This program computes for the average of 3 two-digit numbers (00-55).`
	`Enter the first number:`
	`11`
	`Enter the second number:`
	`11`
	`Enter the third number:`
	`11`
	`Average is: 11`
	`With remainder: 0`

Input(78, 32, 12)	 `Admin@JJ MINGW32 /d/CSPC_LIFE/3RD YEAR/FIRST_SEMESTER/Architecture and Organization/LT3`
	`$ ./BarjaJOSHUAJERICHO.exe`
	`This program computes for the average of 3 two-digit numbers (00-55).`
	`Enter the first number:`
	`78`
	`Enter the second number:`
	`32`
	`Enter the third number:`
	`12`
	`Average is: 40`
	`With remainder: 2`

Input (0, 0, 0)	 `Admin@JJ MINGW32 /d/CSPC_LIFE/3RD YEAR/FIRST_SEMESTER/Architecture and Organization/LT3`
	`$ ./BarjaJOSHUAJERICHO.exe`
	`This program computes for the average of 3 two-digit numbers (00-55).`
	`Enter the first number:`
	`0`
	`Enter the second number:`
	`0`
	`Enter the third number:`
	`0`
	`Average is: 0`
	`With remainder: 0`

PROGRAM CODE

```
section .data
    header_msg db 'This program computes for the average of 3
two-digit numbers (00-55).', 10, 0
    input_1 db 'Enter the first number:', 10, 0
    input_2 db 'Enter the second number:', 10, 0
    input_3 db 'Enter the third number:', 10, 0
    ave_output db 'Average is: %d', 10, 0
    remainder db 'With remainder: %d', 0
    input_format db '%d', 0

section .bss
    num1 resd 1 ; use resd because scanf expects an int*
    num2 resd 1
    num3 resd 1
    rem resd 1

section .text
    global _main
    extern _scanf
    extern _printf

_main:
    push header_msg
    call _printf
    add esp, 4

    ; get user input
    push input_1
    call _printf
    add esp, 4 ; clean up 4 bytes
    push num1
    push input_format
    call _scanf
    add esp, 8

    push input_2
    call _printf
    add esp, 4 ; clean up 4 bytes
    push num2
    push input_format
    call _scanf
    add esp, 8

    push input_3
    call _printf
    add esp, 4 ; clean up 4 bytes
    push num3
    push input_format
    call _scanf
    add esp, 8

    ; compute for the average

    mov eax, [num3]
    add eax, [num2]
    add eax, [num1]

    mov edx, 0 ; remainder
    mov ecx, 3 ; divisor
    div ecx ; quotient

    ; move edx value to variable in order to print the actual value
    mov [rem], edx

    ;  average value
    push eax
    push ave_output
    call _printf
    add esp, 8

    ; remainder value
    push dword [rem]
    push remainder
    call _printf
    add esp, 8

    ret
```