National University
Of Computer and Emerging Sciences

EE2003 – Computer Organization and Assembly Language (Sp’24) Mar 2024

Assignment: 02, **Weight:** 3.0, Due Date: 2 Jun, **CLO:** 2

Teacher: Dr Muhammad Usman Abbasi

Name: Tazmeen Afroz **Roll No:** P22-9252 **Section:** BAI-4A

Note:

Plagiarism will be marked zero straight away to all parties involved.

Subroutines:

1. Make separate subroutines for add, subtract, multiply and divide and then perform all these operations between two numbers of your choice using these subroutines and passing them the numbers as parameters on the stack. Also, store the results for each of the operations in the variables shown in the starter code.

```
[org 0x0100]
jmp start
operand1: dw 5
operand2: dw 2
sum_result: dw 0
subtraction_result: dw 0
multiplication_result: dw 0
division_result: dw 0

add_numbers:
  push bp
  mov bp, sp
  push ax
  push bx
  mov ax, [bp+8]        ; Load the first operand
  add ax, [bp+6]        ; Add the second operand
  mov bx, [bp+4]        ; Get the address of the result variable
```

```
 mov [bx], ax        ; Store the result
  pop bx
  pop ax
  pop bp
  ret 6        ; Clean up the stack and return

subtract_numbers:
  push bp
  mov bp, sp
  push ax
  push bx
  mov ax, [bp+8]      ; Load the first operand
  sub ax, [bp+6]      ; Subtract the second operand
  mov bx, [bp+4]      ; Get the address of the result variable
  mov [bx], ax        ; Store the result
  pop bx
  pop ax
  pop bp
  ret 6        ; Clean up the stack and return

multiply_numbers:
  push bp
  mov bp, sp
  push ax
  push bx
  mov ax, [bp+8]      ; Load the first operand
  mul word [bp+6]     ; Multiply by the second operand
  mov bx, [bp+4]      ; Get the address of the result variable
  mov [bx], ax        ; Store the result
  pop bx
  pop ax
  pop bp
  ret 6        ; Clean up the stack and return

divide_numbers:
  push bp
  mov bp, sp
  push dx
  push ax
  push bx
  xor dx, dx          ; Clear dx for division
  mov ax, [bp+8]      ; Load the first operand
  div word [bp+6]     ; Divide by the second operand
  mov bx, [bp+4]      ; Get the address of the result variable
  mov [bx], ax        ; Store the result
  pop bx
  pop ax
  pop dx
  pop bp
  ret 6         ; Clean up the stack and return

start:
  ; Addition
  mov ax, [operand1]
  push ax
  mov ax, [operand2]
  push ax
  mov ax, sum_result
  push ax
  call add_numbers

  ; Subtraction
  mov ax, [operand1]
  push ax
  mov ax, [operand2]
  push ax
  mov ax, subtraction_result
  push ax
  call subtract_numbers

  ; Multiplication
  mov ax, [operand1]
  push ax
  mov ax, [operand2]
```

```
push ax
mov ax, multiplication_result
push ax
call multiply_numbers

; Division
mov ax, [operand1]
push ax
mov ax, [operand2]
push ax
mov ax, division_result
push ax
call divide_numbers

; Terminate program
mov ax, 0x4c00
int 0x21
```

2. Perform recursion in assembly language using subroutines of your choice.

Note: As the assignment is based on subroutines, make sure to pay attention to the syntax of your subroutines. Marks will be deducted for any syntax error or semantic error in your subroutines.

```
[org 0x0100]
jmp start

factorial:
  push bp
  mov bp, sp
  sub sp, 2

  mov ax, [bp+4]
  cmp ax, 1
  jbe base_case

  push ax
  dec ax
  push ax
  call factorial
  add sp, 2
  mov [bp-2], ax

  mov ax, [bp+4]
  mul word [bp-2]

  jmp fac_end

base_case:
  mov ax, 1

fac_end:
  mov sp, bp
  pop bp
  ret
```

```
start:
  mov ax, 5
  push ax
  call factorial
  add sp, 2
  mov bx, ax

  mov ax, 0x4c00
  int 0x21
```