```html
```

Assembly Program to Compute Sum and Difference of Two Floating Point Numbers

Sub-Subject: Computer Science (Assembly Language Programming)

Topic: Implementing User-Defined Procedures for Floating Point Operations

Given Data:

- Implement a user-defined procedure for the sum of two floating-point numbers.
- Implement a user-defined procedure for the difference of two floating-point numbers.
- Check if the numbers are not equal before proceeding with the computation.

Introduction:

This assembly program is implemented using x86 assembly language, specifically targeting the FPU (Floating Point Unit). It defines two user-defined procedures/functions to calculate the sum and the difference of two floating-point numbers, ensuring the operations are performed only if the numbers are not equal.

```
section .data
    num1    dd  5.5                 ; Define first floating-point number
    num2    dd  3.2                 ; Define second floating-point number
    resultSum dd 0.0                ; Variable to store the sum
    resultDiff dd 0.0               ; Variable to store the difference

section .text
    global _start                   ; Entry point for the program

_start:
    ; Load numbers into FPU registers and compare
    fld     dword [num1]            ; Load num1 into ST(0)
    fld     dword [num2]            ; Load num2 into ST(0) and push ST(0) to ST(1)
    fucompp                         ; Compare ST(0) and ST(1) and pop both
    fstsw   ax                      ; Store FPU status word in AX
    sahf                            ; Load AH into FLAGS status register
    jne     calculate               ; Jump if numbers are not equal

    ; Exit the program if numbers are equal
    mov     eax, 1                  ; System call number (sys_exit)
    xor     ebx, ebx                ; Exit code 0
    int     0x80                    ; Call kernel

calculate:
    ; Call procedure to compute sum
    push    dword [num2]
    push    dword [num1]
    call    sum
    add     esp, 8                  ; Clean up the stack

    ; Call procedure to compute difference
    push    dword [num2]
    push    dword [num1]
    call    diff
    add     esp, 8                  ; Clean up the stack

    ; Exit the program
    mov     eax, 1                  ; System call number (sys_exit)
    xor     ebx, ebx                ; Exit code 0
    int     0x80                    ; Call kernel

sum:
    ; Compute the sum of two floating-point numbers
    fld     dword [esp + 4]         ; Load first argument into ST(0)
    fld     dword [esp + 8]         ; Load second argument into ST(0)
    fadd    st(1), st(0)            ; Add ST(0) to ST(1)
    fstp    dword [resultSum]       ; Store the result in memory
    fstp    st(0)                   ; Clean up the stack (pop ST(0))
    ret

diff:
    ; Compute the difference between two floating-point numbers
    fld     dword [esp + 4]         ; Load first argument into ST(0)
    fld     dword [esp + 8]         ; Load second argument into ST(0)
    fsub    st(1), st(0)            ; Subtract ST(0) from ST(1)
    fstp    dword [resultDiff]      ; Store the result in memory
    fstp    st(0)                   ; Clean up the stack (pop ST(0))
    ret
```

Explanation for Each Step:

1. Data Section Initialization:

`num1` and `num2` are declared to hold the floating-point numbers. `resultSum` and `resultDiff` are declared to store the results of the sum and the difference operations. Explanation: This area initializes the memory to hold the required floating-point numbers and results.

2. Entry Point and Initial Comparison:

The `_start` label marks the entry point. `fld dword [num1]` and `fld dword [num2]` load the floating-point numbers into the FPU registers for comparison. The `fucompp` instruction compares the two numbers and updates the FPU status word. Explanation: Ensures the program checks if the numbers are equal before any operation.

3. Exit if Numbers are Equal:

After comparison, if the numbers are equal, it triggers an exit. Explanation: If numbers are equal, the program will not proceed with further calculations.

4. Calling Sum Procedure:

The two numbers are pushed onto the stack as arguments, followed by a call to the `sum` procedure. `add esp, 8` cleans up the stack after the procedure call. Explanation: The stack mechanism is utilized to pass arguments to the sum procedure.

5. Calling Diff Procedure:

The same stack mechanism is used to call the `diff` procedure. Explanation: This follows the exact mechanism used for the sum procedure, ensuring cleanliness and organization.

6. Exit Program:

System call to exit the program with `exit code 0`. Explanation: Ensures proper termination of the program.

7. Sum Procedure:

Loads arguments from the stack and computes their sum using `fadd`. Stores the result in `resultSum`. Explanation: The procedure handles floating-point addition and memory storage efficiently.

8. Difference Procedure:

Loads arguments from the stack and computes their difference using `fsub`. Stores the result in `resultDiff`. Explanation: This procedure accurately computes and stores the difference between two floating-point numbers.

Final Solution:

The assembly program properly checks if the given floating-point numbers are unequal, then computes and stores their sum and difference using user-defined procedures.

```
```