1. (5 points) Use a loop instruction with indirect addressing to solve the problem
    a. Do not copy the elements to any other array
    b. Use the LOOP and XCHG instruction
    c. The input array, inputStr contains elements: “A”, “B”, “C”, “D”, “E”, “F”, “G”, “H”
    d. The array’s elements after running the program should look like “G”, “H”, “E”, “F”,” C”, “D”, “A”, “B”
        i. Submit the following
            1. Rename the asm file using your last name as Lastname1.asm
            2. Screenshot of the code and memory window showing the contents pf the variable inputStr

main.asm

```
;Student: Aparna Mandapaka
;Class: CSC 3210
;Assignment#: 3
;Description: Using the loop instrution with indirect addressing to solve array's elements

.386
.model flat, stdcall
.stack 4096

ExitProcess PROTO, dwExitCode:DWORD

.data
inputStr BYTE "A", "B", "C", "D", "E", "F", "G", "H"

.code
main PROC
    mov edi,0
    mov ebx, 0
    mov ecx, 0
    mov eax, 0
    mov ebx, OFFSET inputStr ; this is the address reference
    mov edi, OFFSET inputStr ; address reference
    add edi, 7
    mov ecx, 4 ; for this we will use loop

    L1:
        mov al, [edi]
        xchg [ebx], al
        xchg al, [edi]

        dec edi ; decrease
        inc ebx ; increases the loop
        Loop L1

        INVOKE ExitProcess,0   ≤ 1ms elapsed
main ENDP
END main
```

100 % No issues found Ln: 29 Ch: 17 Col: 23 TABS CRLF

Memory 1

Address: 0x00074000 Columns: Auto

```
0x00074000  48 47 46 45 44 43 42 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  HGFEDCBA........................
0x00074029  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................................
0x00074052  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................................
0x0007407B  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................................
0x00074A44  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Memory 1 Memory 2 Memory 3 Registers

main.asm

2. (5 point) Write an assembly program that does the following:
    a. Define the following value 0506-0307-0408-0102h in the .data segment using the 64 bit unsigned identifier named qVal
    b. You can subdivide the qVal value into 4 words - 0506, 0307, 0408, 0102
    c. Extract these words from qVal using PTR operator
    d. Find the sum of the words. The sum should be D17h
    e. Store the result in any 16-bit register
    f. The direction of adding two words goes from left to right
        i. Submit the following
            1. Rename the asm file using your lastname as Lastname2.asm
            2. Screenshot of the code and memory window showing the result in a 16-bit register

```
;Student: Aparna Mandapaka
;Class: CSC 3210
;Assignment#: 3
;Description:Writing an assembly code for the value 0506-0307-0408-0102h

.386
.model flat, stdcall
.stack 4096
ExitProcess PROTO, dwExitCode:DWORD

.data
qVal QWORD 0506030704080102h

.code
main PROC
;intilizing all the registers 0's
    mov edi, 0
    mov ebx,0
    mov ecx, 0
    mov eax,0
    mov edi, OFFSET qVal  ; storing the qVal address into the edi register
    mov ax,  WORD PTR [edi + 6] ; fetching the word from the adadress
    mov bx,  WORD PTR [edi + 4]
    mov cx,  WORD PTR [edi + 2]
    mov dx,  WORD PTR [edi]
    add ax, dx ;we are adding the values from dx register to ax
    add ax, cx ; adding the values from cx register to the ax register
    add ax, bx; adding the values from bx to ax register

    INVOKE ExitProcess, 0
main ENDP
END main
```

```
Registers
EAX = 00000D17 EBX = 00000307 ECX = 00000408 EDX = 005F0102 ESI = 005F1005 EDI = 005F4000 EIP = 005F1041 ESP = 00DBF8D0 EBP = 00DBF8DC EFL = 00000206
```

```
Address: 0x005F4000
0x005F4000  02 01 08 04 07 03 06 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
0x005F4049  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
0x005F4092  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
0x005F40DB  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
0x005F4124  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

3. (5 points) Consider the following code

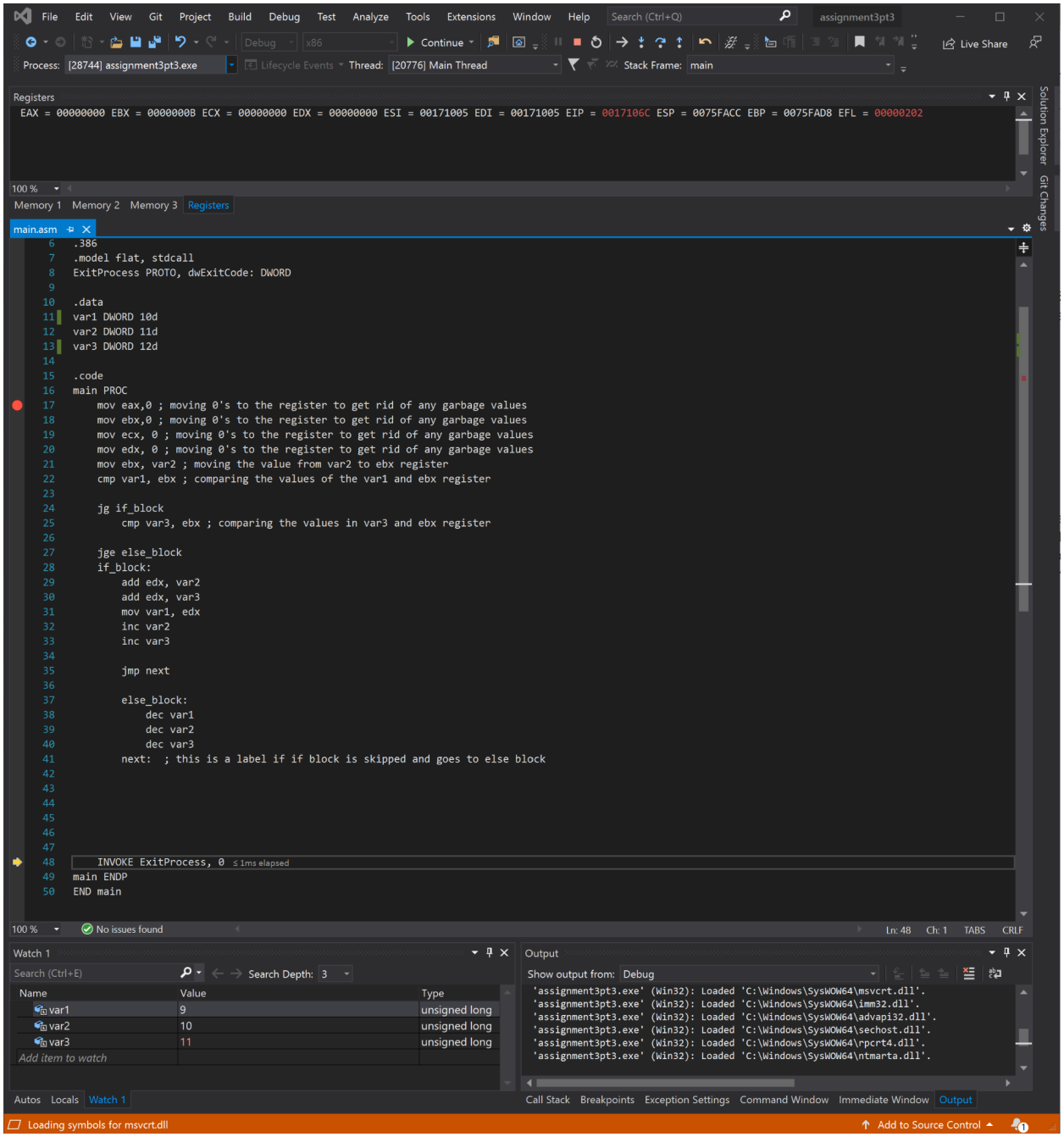
```
if (var1 > var2 ) OR (var3 < var2) {
        var1 = var2 + var3;
        var2++;
        var3++;
}
else{
        var1--;
        var2--;
        var3--;
}
```

Here var1, var2 and var3 are DWORD variables.

Var1 is initialized with 10(Decimal) var2 is initialized with 11 (Decimal) and var3 is initialized with 12 (decimal).

Translate the following code into assembly code (MASM)

You need to implement the logic of the if-else statement with compound condition.



Memory window on var1