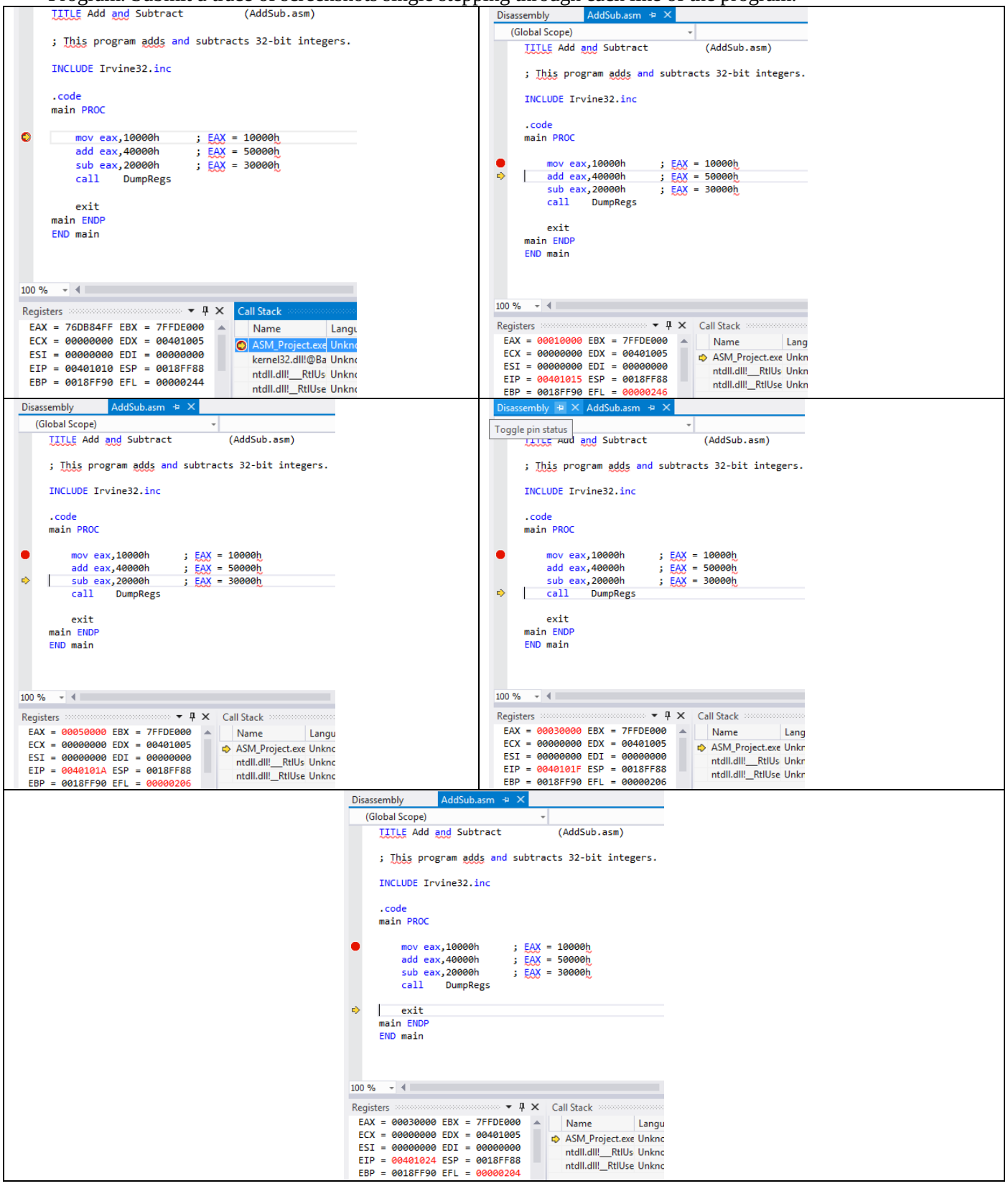
1. Single step through the program created in the previous assignment (HW3)—the AddSub Program. Submit a trace of screenshots single stepping through each line of the program.



2. Declare whether each flag is set (‘1’ for set, ‘0’ for not set) following each instruction. Show the function of the instructions by hand.

Instruction #	*Instruction*	*CF*	*SF*	*ZF*	*OF*
0	mov ax, 7F00h				
1	add ax, 100h	**1**	**0**	**0**	**0**
2	add al, 0h	**0**	**0**	**1**	**0**
3	add al, 0FFh	**0**	**0**	**0**	**0**
4	sub ah, 080h	**0**	**0**	**1**	**0**

3. Given the following data declarations:

```
.data
Alpha  BYTE 2Bh, 23h, 4Ah
Beta   DWORD 222h
Delta  DWORD 312h
Iota   DWORD 434h
Zeta   WORD 124h
```

A. Write an instruction that moves *Beta* into *EAX* and adds *Iota* to the same register

```
mov EAX, Beta
add  EAX, Iota
```

B. Write a set of instructions that adds all the elements of the array *Alpha* into *AL*

```
;Method One
.data
Alpha BYTE 2Bh, 23h, 4Ah
.code
       mov esi, OFFSET Alpha
       mov AL, [esi]        ;2B
       add AL, [esi+1]      ;4E
       add AL, [esi+2]      ;98
```

```
;Method Two (Indirect Operands)
.data
Alpha BYTE 2Bh, 23h, 4Ah
.code
       mov esi, OFFSET Alpha
       mov AL, [esi]        ;2B
       INC esi
       add AL, [esi]        ;4E
       INC esi
       add AL, [esi]        ;98
```

```
;Method Three (Loop)
.data
Alpha BYTE 2Bh, 23h, 4Ah
.code
       mov edi, OFFSET Alpha
       mov ecx, LENGTHOF Alpha
       mov AL, 0
L1:
       add AL, [edi]
       add edi, TYPE Alpha
       loop L1
```

C. Write a set of instructions that moves *Delta* into *EAX*, adds the value stored in *Zeta* to the same register

```
.data
Delta DWORD 312h
Zeta WORD 124h
.code
mov EAX, Delta       ;EAX = 312h
add AX, Zeta         ;EAX = 436h
```

D. Write an instruction that moves the last two bytes in *ALPHA* into *CX*.

```
.data
Alpha BYTE 2Bh, 23h, 4Ah
.code
movzx CX, Alpha+2
```

E. What are the contents of C*X* subsequent to part D of this question?
CX = 0000

4. Fill in the requested register values after executions of the instructions:
Show the memory map. NOTE: Whoever created the question forgot to place a '0' to the beginning AACCh, BCCBh, AB71h, BB00h.

```
.data
myBytes       BYTE   24h, 25h, 26h, 27h
myWords       WORD   2233h, 3355h, 3456h, AACCh, BCCBh
myDoubles     DWORD  BB00h, 2345h, 3456h, AB71h, 44h
myPointer     DWORD  myDoubles

.code
mov esi, OFFSET myBytes
mov ax, WORD PTR [esi+2]          ; A.   AX   =   2726
mov eax, DWORD PTR myWords        ; B.   EAX  =   33552233
mov esi, myPointer
mov ax, WORD PTR [esi+8]          ; C.   AX   =   3456
mov ax, WORD PTR [esi+1]          ; D.   AX   =   00BB
mov ax, WORD PTR [esi-6]          ; E.   AX   =   3456
```

Memory Map

	DWORD	WORD	BYTE	offset
myBytes			24	0
			25	1
			26	2
			27	3
myWords		2233	33	0
			22	1
		3355	55	2
			33	3
		3456	56	4
			34	5
		AACC	CC	6
			AA	7
		BCCB	CB	8
			BC	9

		DWORD	WORD	BYTE	offset
myPointer	myDoubles	0000BB00	BB00	00	0
				BB	1
			0000	00	2
				00	3
		00002345	2345	45	4
				23	5
			0000	00	6
				00	7
		00003456	3456	56	8
				34	9
			0000	00	10
				00	11
		0000AB71	AB71	71	12
				AB	13
			0000	00	14
				00	15
		00000044	0044	44	16
				00	17
			0000	00	18
				00	19

5. What is returned by the following operations in connection to the given array:

   .data
   myArray DWORD 12 DUP(23), 3, 34, 467 , 2365, 47899

   A. TYPE myArray
      4_{10}, 4_{16}

   B. LENGTHOF myArray
      17_{10}, 11_{16}

   C. SIZEOF myArray
      68_{10}, 44_{16}

6. Fill in the requested register values after executions of the instructions (Do not let your eyes deceive you. There are some mov**S**x instructions and some mov**Z**x instructions.):

```
.code
mov bx, 0ACD8h
movzx eax, bx                    ; A.   EAX =  0000ACD8h
movzx edx, bl                    ; B.   EDX =  000000D8h
movzx cx, bh                     ; C.   CX =   00ACh

mov bx, 0DD34h
movsx eax, bx                    ; D.   EAX = FFFFDD34
movsx edx, bl                    ; E.   EDX = 00000034
movsx cx, bh                     ; F.   CX =  FFDD
```

Note: This is also in HW 3.

7. Write a program that prints *War Eagle* on your screen. You can use the following. Assemble and generate the output using MASM and Visual Studio. Embed your output in your submission.
   Source code:

```
TITLE  PRINT WAR EAGLE
COMMENT !
This program prints "WAR EAGLE" on the screen
with a carriage return and line feed.
!

INCLUDE Irvine32.inc
.data
message BYTE "WAR EAGLE",0dh,0ah,0

.code
main PROC
       mov edx, offset message
       Call WriteString
       exit

main ENDP

END main
```

   Output:

```
WAR EAGLE
Press any key to continue . . .
```