

VU LEARNING HUB

CS401 Assignment 1 Solution

Fall 2026

📢 Join our official WhatsApp Channel for instant VU updates, past papers & assignment solutions!

➡ **https://whatsapp.com/channel/0029VbDOzXMC6ZvZBUbW8k2n**

💬 Questions? Chat with us on WhatsApp for quick help & guidance!

➡ **https://wa.me/923135320064**

🌐 Visit our website for more past papers, notes & assignment solutions!

➡ **https://vulearninghub.xyz**

CS401 Assignment 1 Solution

Fall 2026 | Total Marks: 20 | Due Date: 15 November 2026

Name: VU Learning Hub

Question

Write 8086 assembly code in NASM syntax to process an array of unsigned 16 bit integers. The array starts at DS SI. CX holds the number of elements. The array can hold 0 to 20 elements. Store each result in the register given in the table. If the array is empty set all four registers to zero. If the sum goes over 65535 keep only its lower 16 bits.

Operation	Result Register	Marks
Sum of all elements	AX	5
Largest element	BX	5
Count of even elements	DX	5
Count of elements greater than 50	DI	5

Solution Code

Write this code only between the STUDENT SOLUTION markers in solution_template.asm. Do not change anything before the start marker or after the end marker.

```
; Sum of all elements in AX, largest in BX, even count in DX, count over 50 in DI.
; Empty array (CX = 0) leaves all four registers at zero.
; The sum keeps only its lower 16 bits because ADD wraps in 16 bits.

    xor ax, ax          ; sum = 0
    xor bx, bx          ; largest = 0
    xor dx, dx          ; even count = 0
    xor di, di          ; count over 50 = 0
    test cx, cx         ; empty array?
    jz done             ; yes, all four stay zero

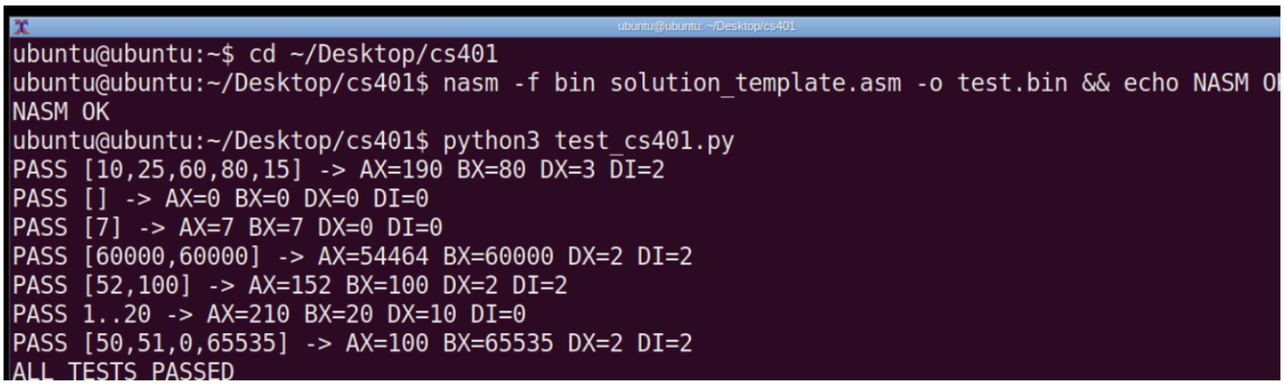
next_elem:
    mov bp, [si]        ; load the current element
    add ax, bp          ; add it to the sum
    cmp bp, bx          ; compare it with the largest so far
    jbe not_max         ; skip if it is not larger
    mov bx, bp          ; new largest
not_max:
    test bp, 1          ; check the lowest bit
    jnz not_even        ; skip if the number is odd
    inc dx              ; even number, count it
not_even:
    cmp bp, 50          ; compare with 50
    jbe not_gt50        ; skip if it is 50 or less
    inc di              ; over 50, count it
not_gt50:
    add si, 2           ; move to the next word
    loop next_elem      ; CX = CX - 1, repeat while CX is not zero
done:
```

How the Code Works

All four output registers start at zero. If CX is zero the code jumps to the end and all four stay zero. Each loop reads one element into BP. ADD puts it in the sum in AX. CMP checks it against the largest in BX. TEST checks the lowest bit to count even numbers in DX. CMP with 50 counts numbers over 50 in DI. SI moves to the next word and LOOP repeats until CX is zero.

Test Result

The code was assembled with NASM. It was tested on seven arrays. All seven tests passed.



Example array 10 25 60 80 15 gives AX 190 BX 80 DX 3 DI 2. Empty array gives all four registers zero. Array 60000 60000 gives AX 54464 because the sum keeps only its lower 16 bits.