Air University
Department of Cyber Security
(Mid-Term Examination: Fall 2025)

Student Sign: _____

Subject: Computer Organization and Assembly Language
Class: BS-Cyber Security
Code: CS-226
Section: A & B
FM Name: Ms. Maryam Malik
FM Signature: _____

Total Marks: 50
Time:
Date: 5th Nov 2025
Max Duration: 2 Hours
HoD Signature: _____

Instructions:

- You are required to attempt **ALL Questions**.
- This is a closed book/notes exam.
- **Return question paper with the answer sheet**

No	Questions	CLO	Ma
1	a) What is the decimal representation of **11110010** signed binary numbers? b) Justify the following statement with example. "MASM does not prevent you from initializing a WORD with a negative value, but it's considered poor style." Why? c) Explain the concept of SIMD and how MMX registers utilize this feature and how many registers are used in 32bit and 64bit processor. d) How x86 processor's and 64-bit modes of operations differ from each other? e) What is the difference between BYTE 5 DUP(0) and BYTE 5 DUP(?)?	1	1
2	a. **Update** values of the flags **CF, OF, ZF, SF, PF, AF** after execution of each block which are part of a single program and write reasoning.	2	

```
.code

; block 1
  mov al, 250
  mov bl, 10
  add al, bl
```

```
.code

;block 2
mov al, 01000000b
mov bl, 01111111b
add al, bl
```

A video encoding program takes 80 minutes on a single processor.
You plan to upgrade to 8 cores, where 60% of the code can be parallelized.
Find the new execution time.
How many cores would be needed to achieve a **speedup of at least 3×**

Write an equivalent **x86 MASM assembly program** that performs the same computation. Use both **equate directives** (=, EQU), wherever suitable.

3 | 20

```
#include <iostream>
using namespace std;

int main() {
    int val1 = 8;
    int val2 = 12;
    int val3 = 5;
    int val4 = 3;
    int result;

    result = (val1 * 2) + (val2 - 4) - (val3 + val4);

    return 0;
}
```

Simulate the Fetch–Decode–Execute cycle for the three instructions. Show the contents of PC, IR, MAR, MDR, and ACC after each instruction, and give the final value stored in memory.

Opcode	instruction
0001	LOAD
0010	ADD
0011	SUB
0100	STORE