INSTRUCTIONS: Show your work (i.e., how you derived your answer or the reason behind your thinking) in addition to your answer. Budget your time wisely (e.g., do not spend too much time on a single question).

1. (4%) What is the program that combines object files into an executable program?
2. (4%) Explain the relationship between an assembly program and an assembler.
3. (4%) Which of the following are true? (multiple choices)
   (a) a directive is executed at runtime
   (b) an instruction is executed at runtime
   (c) a directive is executed at assembly time
   (d) an instruction is executed at assembly time
4. (4%) What is the memory byte order, from low to high address, of the following data definition?
   BigVal DWORD 12345678h
5. (4%) What is the value of the Overflow flag after the execution of code below?
   MOV AL, 88h
   ADD AL, 90h
6. (4%) What is the value of AL in hexadecimal representation after the execution of the instruction below?
   MOV AX, -68
7. (4%) What is the value of EAX after the execution of the code below?
   array WORD 100, 200,
   300, 3 DUP(350),
   400, 500, 700
   MOV EAX, SIZEOF array
8. (3%) (True/False) The LOOPE instruction jumps to a label when (and only when) the Zero flag is clear.
9. (6%) What are the values of the Carry flag and AL after the execution of the code below?
   MOV AL, 8Fh
   SHL AL, 2
10. (3%) (True/False) A procedure’s stack frame always contains caller’s return address and procedure’s local variables.
11. (4%) Assuming that a procedure contains no local variables, a stack frame is created by which sequence of actions at runtime?
    (a) EBP pushed on stack; arguments pushed on stack; procedure called; EBP set to ESP
    (b) arguments pushed on stack; EBP pushed on stack; ESP set to EBP; procedure called
    (c) arguments pushed on stack; procedure called; EBP pushed on stack; ESP set to EBP
    (d) arguments pushed on stack; procedure called; ESP set to EBP; EBP pushed on stack
12. (16%) Translate the following C code into assembly. (Note: No .IF or other directives are allowed.) You may assume that A and B are BYTE variables that are already defined.
   while (A > 10) {
      if ( ((A>=100) && (A<156)) || (B>20) )
         A=A-2;
      else {
         A=A-3;
         B=B+1;
      }
   }

(continue on the back side)
13. (20%) Trace the following code:

```assembly
.data
FinalResult DWORD 11223344h
.code
MOV AL, 3
MOV BL, 2
MOV ESI, OFFSET FinalResult
MOV ECX, 4
L1:
    MOV BYTE PTR [ESI], AL
    SUB BYTE PTR [ESI], BL
    MOV AL, BL
    MOV BL, BYTE PTR [ESI]
    INC ESI
    LOOP L1
```

(a) (10%) What is the value stored in FinalResult after the execution of the above code? Please write it in little endian order and hexadecimal form.

(b) (10%) Suppose the 3rd line in the code is changed to MOV ESI, 0 so as to initialize ESI to 0. Change the code within the loop to produce the same results.

14. (20%) The procedure `Factorial_no_stack` below calculates the factorial of integer N using a global variable instead of using the run-time stack.

```assembly
main PROC
    push 8
    call Factorial_stack
    exit
main ENDP

Factorial_no_stack PROC
    push ebp
    mov ebp, esp
    mov eax, [ebp+8]
    cmp eax, 0
    ja L1
    mov eax, 1
    jmp L2
L1: dec eax
    push eax
    call Factorial_no_stack
    mov ebx, [ebp+8]
    mul ebx
L2: pop ebp
    ret 4
Factorial_no_stack ENDP
```

(a) (6%) Explain why the procedure `Factorial_no_stack` does not work, while `Factorial_stack` works?

(b) (7%) Rewrite the procedure `Factorial_no_stack` so that it works. (Hint: Use an extra variable for the partial product, and consider when to perform `mul`.)

(c) (7%) Rewrite the procedure `Factorial_stack` so that the return value is also passed by the stack instead of by `eax`. 

---

Page 2/2