35 pts

Name: ______________________________________

Class Day / Time: ______________________________________

Due Date: ______________________________________

Lab #12 – Advanced Procedures

1) Write a procedure **Avg** to find the average of collection of doubleword-size integers in an array. Procedure **Avg** will have three parameters:
(1) the address of the array
(2) the number of integers in the array (passed as a doubleword)
(3) the address of a doubleword at which to store the result.

Use the stack to pass arguments to the procedure. Register contents should be unchanged by the procedure; that is, registers, including the flags. Register, which are used in the procedure should be saved at the beginning of the procedure and restored before returning. Allocate stack space as needed for local variables. Use the ret instruction with no operand.

2) For problem # 1 above, write an assembly test driver, that is, a simple main program that will input appropriate values, call the procedure you created in the problem # 1, and output results. The main program must remove arguments from the stack.

3) Declare a procedure named **MultArray** that receives two pointers to arrays of doublewords, and a third parameter indicating the number of array elements.

4) Create a PROTO directive for the procedure in the problem # 3 above.

5) How many bytes of stack space would be used by the Factorial procedure when calculating 5!?

6) Modify the **ArraySum** procedure below, which calculates the sum of an array of doublewords, to receive the arguments on the stack. The modified procedure will be responsible to adjust the stack to remove the arguments before return (STDCALL). Make sure you include the necessary instructions to create the stack frame.

```
ArraySum PROC
; Receives: ESI points to an array of doublewords,
;   ECX = number of array elements.
; Returns: EAX = sum
;-----------------------------------------------------
        mov eax,0           ; set the sum to zero
L1:     add eax,[esi]       ; add each integer to sum
        add esi,4 ; point to next integer
        loop L1   ; repeat for array size
        ret
ArraySum ENDP
```

7) Declare a local variable named **pArray** that is a pointer to an array of doublewords.

8) Declare a local variable named **buffer** that is an array of 20 bytes.

9) Declare a local variable named **pwArray** which points to a 16-bit unsigned integer.

10) Declare a local variable named **myByte** that holds an 8-bit signed integer.

11) Declare a local variable named **myArray** that is an array of 20 doublewords