HOMEWORK 5

Directions: *This assignment is due by **11:00 a.m. on Friday, October 31**. Submit your answer to the programming problem (Question 1) as a single .asm file in Canvas. Submit your answers to the remaining questions as a PDF in Canvas.*

Scoring (40 total points): *#1: 30 pts • #2–5: 10 point total*

Background

Integer values in the range [0h, Fh] can be represented using only four bits. (An 8-bit unit is called a byte; a 4-bit unit is called a "nibble." Yes, seriously.)

When storage space is a concern—e.g., when compressing data, or when designing network protocols—it is common to "pack" several integer values together, e.g., by packing two consecutive 4-bit values into one 8-bit value. As an example, the BYTE array

01 0B 0C 03 05 08

can be "packed" into a 3-byte array, storing two values per byte:

1B C3 58.

When the number of bytes in the original array is odd, put a 0 in the low nibble of the last byte, e.g.,

05 0A 0F ⇒ 5A F0

Assignment

1. (Programming)
   - Download **HW5.asm** from Canvas.
   - In the TITLE line, replace TODO: YOUR NAME with your name.
   - Fill in the *IsOdd* procedure, as described in the code. It is a STDCALL procedure that receives one stack argument—a 32-bit, signed integer value—and returns 1 or 0 if the integer is odd or even, respectively.
   - Fill in the *Pack* procedure. It is a STDCALL procedure that receives three stack arguments. The first is a BYTE input array, where each value is between 0h and Fh, inclusive. The second argument is the number of bytes in the input array. The third argument is an output array with half as many bytes as the input array.[1] The procedure should "pack" pairs of consecutive bytes from the input array, as described above, storing the result in the output array.
   - The output should be:

```
Is odd? +3 => +1
Is odd? +2 => +0
Is odd? +1 => +1
Is odd? +0 => +0
Is odd? +1 => +1
Is odd? +2 => +0
Is odd? +3 => +1
Is odd? -2147483648 => +0
Is odd? -2147483639 => +1
Is odd? +2147483646 => +0
Is odd? +2147483645 => +1
Input (10 bytes):       1 2 3 4 F E D C B A
Result (5 bytes):       12 34 FE DC BA
Input (3 bytes):        B 2 D
Result (2 bytes):       B2 D0
Input (1 bytes):        A
Result (1 bytes):       A0
Input (6 bytes):        C 0 E 3 6 4
Result (3 bytes):       C0 E3 64
Input (0 bytes):
Result (0 bytes):
```

[1] If the length n of the input array is odd, the output array contains $(n + 1)/2$ bytes—enough to store the packed data.

2. The *ftp* program (a common Unix utility used to transfer files to/from a remote machine) has a number of options, including options to
   - Attempt to use an anonymous login, rather than asking the user to enter a username and password.
   - Enable interactive prompting when multiple files are transferred at once
     (i.e., don't ask the user to confirm every single transfer)
   - Enable verbose mode, which displays informative messages during file transfers.

   a. Suppose you are implementing *ftp* in assembly language and want to store these three options in a bit set. Define a symbolic constant for each of the three options.

```
USE_ANONYMOUS_LOGIN          = _________
ENABLE_INTERACTIVE_PROMPTING = _________
ENABLE_VERBOSE_MODE          = _________
```

   b. Suppose you will store the set of options selected by the user in a BYTE variable called `options`. By default, *ftp* will *not* use anonymous logins, it *will* use interactive prompting, and verbose mode *is* enabled. To represent this set of defaults, what should the value of options be? (Use the symbolic constants above in your declaration; do not use "magic numbers.")

```
.data
options BYTE _____________________________________________________
```

   c. Write sequences of assembly language instructions to change the value of `options` as follows:

   (i) Enable the ENABLE_VERBOSE_MODE option (or, if it's already enabled, leave it enabled).

   (ii) Disable the USE_ANONYMOUS_LOGIN option (or, if it's already disabled, leave it).

   (iii) Toggle the ENABLE_INTERACTIVE_PROMPTING option – i.e., turn if off if it's on, and turn it on if it's off.

   (iv) If the ENABLE_VERBOSE_MODE option is *enabled*, jump to the label *displayResponse*.

   (v) If the USE_ANONYMOUS_LOGIN option is *disabled*, jump to the label *inputCredentials*.

3. The LAHF (Load AH from Flags) instruction copies the low byte of the EFLAGS (bits 0–7) into the AH register. Write a sequence of instructions that will set EAX to FFFFFFFFh if the zero flag is set and 00000000h otherwise, using only LAHF and bitwise operations (no conditional jumps). Briefly comment each instruction to explain why your solution works.

4. Suppose you want to swap the values in AH and AL. Write an instruction to do this using a bitwise rotation.

5. When an Internet Protocol (IP) packet is transmitted across a network, the first byte of the packet is divided into two parts: the high four bits represent the version number, and the low four bits represent the "Internet Header Length" or IHL. Both are 4-bit unsigned integer values.

    a. Suppose the first byte of the IP packet is stored in DL. Write a sequence of instructions that will load the version number into EAX (zero-extending it from 4 bits to 32 bits).

    b. Suppose the first byte of the IP packet is stored in DL. Write a sequence of instructions that will load the Internet Header Length into EAX (zero-extending it from 4 bits to 32 bits).