INEL 4206. PAssignment 1*:
400 Points

Note 1. Your code should compile and run easily (without any further effort) by MS Visual C++ when we are grading it.
Note 2. Some sample programs from the text book, Chapter 3, are already in the course web page.

Program 1. Write a program (using c and inline assembler) that

- receives an integer number, name *n*, from input, and
- calls a function, name *sub10*,
- function *sub10* subtracts 10 from *n,*
- then, the program prints n and ends.

Note 1. The body of *sub10* must be all in assembly.
Note 2. sub10 receives *n* as a parameter.

Example: if the program receives 15 as its input, it should print 5.

Program 2. Write a program (using c and inline assembler) that

- defines an array, name *table*, of 52 bytes containing letters AB ... Zab ... z,
- then in a loop receives from the console a number, name *n*, between 0 and 52,
- if *n* is not between 0 and 52, the program ends,
- if *n* is 0, the program prints the whole *table*,
- if *n* is between 1 and 52, it calls a function, name *swapElements*,
- function *swapElements* swaps *table*[0] and *table*[*n*-1].

Note 1. The body of *swapElements* must be all in assembly.
Note 2. *swapElements* receives *table* and *n* as parameters.

Example: if the program receives 0, 5, 0, 10, 4, 0, 55, it should print the following:
0
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmanopqrstuvwxyz
5
0
EBCDAFGHIJKLMNOPQRSTUVWXYZabcdefghijklmanopqrstuvwxyz
10
4
0
DBCJAFGHIEKLMNOPQRSTUVWXYZabcdefghijklmanopqrstuvwxyz
55

---

* **IMPORTANT NOTE:** Assignments that require **p**rogramming in **a**ssembly are called PAssignmet. You could interpret that PAssignments are the path to **pass** the course if you do them properly. Please also note that PAssignments require individual interviews. In the interview, you could be questioned about any single line in your code. Moreover, you should be able to make some changes in your code right away. If you cannot pass an interview, you get 0 for that PAssignment.

Program 3. Write a program (using c and inline assembler) that

- defines an array, name *table*, of 26 bytes,
- then it calls a function, name *initializeTable*,
- function initializeTable does the following:
    - it initialize register A with the ASCII code of letter 'a',
    - it initialize register B with the address of *table*
    - then in a loop,
        - stores the ASCII code of letter 'a' in *table*[0]
        - stores the ASCII code of letter 'b' in *table*[1]
        - …
        - stores the ASCII code of letter 'z' in *table*[25]
- then the program, prints the whole table and ends.

Note 1. The body of *initializeTable* must be all in assembly.
Note 2. *initializeTable* receives *table* as a parameter.

Program 4. Write a program (using c and inline assembler) that

- receives four characters (of one byte each) from the console, and
- stores them in c1, c2, c3, and c4, and then
- calls a function, name *reverse*,
- function *reverse* reverses the order of those four characters,
- then, the program prints c1, c2, c3, and c4; and ends.

Note 1. The body of *reverse* must be all in assembly.
Note 2. *reverse* receives c1, c2, c3, and c4 as parameters.
Note 3. Only use push and pop operations in function *reserve*.

Example: if the program receives BcFG as its input, it should print GFcB.

Program 5. Write a program (using c and inline assembler) that

- takes as input from the console an integer *w*
- call a function named *transparentRegister*
- the function *transparentRegister* stores *w* in register *EAX*
- in the function you must add 9 to *EAX* contents without losing the previous value
- the, the program prints the original value after the addition has taken place.

Note 1. The body of *transparentRegister* must be all in assembly.
Note 2. Function *transparentRegister* must use the stack opcodes: push and pop
Note 3. Function *transparentRegister* takes as input w

Program 6. Write a program (using c and inline assembler) that

- takes as input from the console an integer named *dividend* and other named *divisor*
- call a function named *fullDiv*

* **IMPORTANT NOTE:** Assignments that require **p**rogramming in **a**ssembly are called PAssignmet. You could interpret that PAssignments are the path to **pass** the course if you do them properly. Please also note that PAssignments require individual interviews. In the interview, you could be questioned about any single line in your code. Moreover, you should be able to make some changes in your code right away. If you cannot pass an interview, you get 0 for that PAssignment.

- function *fullDiv* performs division by the input divisor until the quotient is less than or equal 0
- then the program, prints the number of iterations needed to get the quotient to be less or equal to 0

Note 1. Both inputs *dividend* and *divisor* must be greater than zero
Note 2. The body of *fullDiv* must be all in assembly.
Note 3. Function *fullDiv* takes as input the dividend and the divisor and the *maxIter* operand.
Note 4. *maxIter* will store the number of iterations

Program 7. Write a program (using c and inline assembler) that

- takes as input from the console the operands of a multiplication
- inputs are the multiplicand operand and the multiplier operand and result
- call a function named *successiveAddition*
- the function *successiveAddition* performs multiplication without using the MUL or IMUL opcode
- then the program prints the value of the multiplication product

Note 1. The body of *successiveAddition* must be all in assembly.
Note 2. Function *successiveAddition* takes as input the multiplicand and the multiplier and result.

Program 8. Write a program (using c and inline assembler) that

- takes as input from the console 10 integers and stores them in an array table
- call a function named *average*
- the function *average* performs addition element by element and then find the average value

Note 1. The body of *average* must be all in assembly.
Note 2. Average must sum table[0]+table[1]+….+table[n-1] and then find the average value (this could be a real number)
Note 3. Function *average* takes as input table and result

Program 9. Write a program (using c and inline assembler) that

- receives three integer numbers from the console
- calls a function, name *sort*
- function *sort* arranges the three input numbers in non-decreasing order,
- then, the program prints the resulting list and ends.

Note 1. The body of sort must be all in assembly.
Note 2. Function sort takes three integers as input.
Example: if the program receives 9 7 100 as input, it should print 7 9 100.

The deadline is **Sunday, July 7 at 8:00PM.** **Send it as ONE text file ATTACHED to your email to both ahchinaei@ece.uprm.edu and earzuaga@ece.uprm.edu.**
Subject of the email: INEL4206, PA1

* **IMPORTANT NOTE:** Assignments that require **p**rogramming in **a**ssembly are called PAssignmet. You could interpret that PAssignments are the path to **pass** the course if you do them properly. Please also note that PAssignments require individual interviews. In the interview, you could be questioned about any single line in your code. Moreover, you should be able to make some changes in your code right away. If you cannot pass an interview, you get 0 for that PAssignment.