CPSC 3303 Programming Project 2

Instructions For this assignment, you must write four programs in the **assembly language** you just started learning. The four programs will be included in ONE file inside ONE project. These programs should be implemented and tested using the *Visual Studio* IDE. For all these exercises, you cannot use built-in functions that perform these operations. In case of a doubt, check with your instructor.

For this assignment, you will build your own **byte** input/output functions using basic **character** input/output functions provided by the author of the textbook.

Objectives of this assignment:

- To build up your programming skills in the assembly language
- To get familiar with *Visual Studio*
- To stress on the fact that the memory (variables) ultimately are 0s and 1s
- To use some basic character input/output functions
- To build your own basic byte input/output functions.
- To distinguish between "numbers" and characters

Warning:

The executable you will build may get **blocked** by an Antivirus scanner on your system. In this case, read this from the Textbook author: "*Antivirus scanner software has improved greatly in recent years, and so have the number of viruses (one website reports 50,000 at present). Because of this, your computer's antivirus scanner may report a false positive when you build your program, and refuse to let you run it. There are a few workarounds: (1) You can add your project's bin/debug folder into an exclusion list in your antivirus configuration. This is my preferred approach, and it nearly always works. (2) You can suspend your realtime antivirus scanner software, but this will leave you open to malware for a short time. If you choose this option, be sure to temporarily disconnect your computer from the Internet. (3) You can send a copy of your program's EXE file to the antivirus software vendor, labeling it as a false positive. The virus scanner I use automatically uploads any questionable EXE file to their website and either quarantines or releases the file within about 30 minutes.*" *I used the first workaround, i.e., I added my project's /debug folder to the exclusion list in my antivirus configuration.*

Input/Output Built-in Functions Allowed:

The textbook author wrote some helpful functions. For this project, you **can** use only these two functions:

1) `ReadChar` : this function will allow you to read a character from the keyboard and receive its ASCII code in the register AL. This function is described in the textbook Page 165 (beginning/page).

2) `WriteChar` : this function will allow you to display on the console window (command window) a character whose ASCII code is stored in the register AL. See in the textbook Page 169 (start/page).

==========

For this project, you must ultimately design and implement the code for these two functions:

1) `WriteHexByte` : this function will allow you to display on the console window (command window) the hexadecimal number stored in the register AL. **Note** that for this project, you need only to write the code. In a future assignment, you will organize it as a function (procedure that you can reuse).

2) `ReadHexByte` : this function will allow you to read an 8-bit hexadecimal integer from the keyboard and receive it in the register AL.

Note that for this project, you need only to write the code for these two functions. In a **future** assignment, you will organize them as functions (i.e., **procedures** that you can call).

Note: the four exercises must be included in ONE project using ONE assembly file. Just separate in the file the exercises using clear comments.

Programming Exercise 1 (6 points):

The objective of this exercise is to write a program that converts a value ***v*** of a decimal digit ***d*** into the ASCII code of the digit ***d***. We assume that the register AL contains a value between 0 and 9. Therefore, AL will contain the ***value*** of a decimal digit '0' to '9'. You do not have to test AL for the range.

Your program 1) must store in the register DL the ***ASCII code*** of the character, and 2) must display the character using the function `WriteChar`.

Example: If AL will contain the **value** 4. Your program 1) must store in DL the ASCII code of '4', i.e. DL←34h, and 2) must display the **character** 4 using the built-in function `WriteChar`.

To test our program, you must initialize AL to various values varying from 0 to 9 (e.g., `MOV AL, 06h`) just before your instructions of the program you wrote.

Programming Exercise 2 (24 points): (Implementing *WriteHexByte*)

The objective of this exercise is to write a program that implements the function ***WriteHexByte*** to display in hexadecimal on the console the byte contained in the register AL. We assume that AL contains a hexadecimal number that uses only **decimal** digits (i.e. 0 to 9). Therefore, AL will contain a value such that both nibbles take only the values from 0 to 9. For example, if AL = 94h = $(1001\ 0100)_2$, then this content is valid because the most significant nibble $(1001)_2$ and the least significant nibble $(0100)_2$ are in the range [0-9]. The value 5Ah is **invalid** because the least significant nibble ($(1010)_2$ = Ah) is not in the range [0-9]. Given a valid value in AL, your program must :

1) store in the register DH the ASCII code of the most significant nibble.
2) store in the register DL the ASCII code of the least significant nibble.
3) Display (using `WriteChar`) on the console the character stored in DH (*recall that* `WriteChar` *uses AL for the character to display*).
4) Display (using `WriteChar`) on the console the character stored in DL.
5) Display 'h' to indicate that the number is a hexadecimal number.
6) *'Display'* the 'line feed' character (ASCII code is 0Ah) to go to the next line.
7) *'Display'* the 'carriage' character (ASCII code is 0Dh) to go to the next line.

Displaying a 'line feed' and a 'carriage return' will produce a new line. Check what happens if you do not display 'line feed' and/or a 'carriage return'.

Example : If AL contains the number 94h, your program 1) must store 39h (ASCII code of the character '9') in DH , 2) must store 34h (ASCII code of the character '4') in DL, 3) must display the characters '9', '4', 'h', 'line feed', and 'carriage return'. Recall that 'line feed' is **ONE** character whose ASCII code is 0Ah and 'carriage return is **ONE** character whose ASCII code is 0Dh

To test our program, you must initialize AL to various values varying from 00h to 99h (e.g., `MOV AL, 47h`) just before your instructions of the program you wrote. Do **NOT** use hexadecimal digits.

Programming Exercise 3 (6 points):

The objective of this exercise is to write a program that will read a **decimal** digit ***d*** from the keyboard, convert the ASCII code of a digit ***d*** into the value ***v*** of ***d,*** and store ***v*** in the register **DL** .

You must use the function ***`ReadChar`*** to input a decimal digit ***d*** from the keyboard. As described above, the function ***ReadChar*** will return in the register AL the ASCII code of the decimal digit ***d***. We assume that the user will input ONLY decimal digits ('0' to '9'). Therefore, the register AL will contain the ***ASCII code*** of a decimal digit '0' to '9', i.e., 30h to 39h. You do not have to test AL. Your program must store in the register DL the ***value*** of the digit.

Example: If the user enters the digit '4', AL will contain the **ASCII code** 34h. Your program must store in DL the **value** of the digit '4', i.e., the value 04h. Ultimately, DL ← 04h.

Programming Exercise 4 (24 points):

The objective is to write a program that implements the function ***ReadHexByte*** to input from the keyboard a byte (i.e., two digits) and store it in the register AL. You must write the program to implement the function ***ReadHexByte.*** In order to achieve this, you must read consecutively two decimal digits from the keyboard (use `ReadChar` for each digit). We assume that the user will enter only TWO decimal digits (i.e. '0' to '9'). The first digit entered will be set as the most significant nibble of AL and the second digit entered will be set as the least significant nibble of AL. Process appropriately the two characters and store in AL the value meant by the user.

Example: suppose the user enters the digit '9' followed by the digit '4'. This means that the user means the byte 94h. Your program must ultimately store in AL the value 94h = $(1001\ 0100)_2$. The digit '9' 'produced the most significant nibble $(1001)_2$ and the digit '4' 'produced the least significant nibble $(0100)_2$.

What you need to turn in:

- Electronic copies of your report (standalone) and source code (zipped) of your program. The four exercises must be included in the same project. Clearly indicate the separation between them by using comments in the source program. Your project folder must be zipped folder named ***m4-name***, where *name* is your lastname. Zip the project folder and post it with the standalone report on Canvas. Submit separately (not inside the zipped folder) the report as a Microsoft Word or PDF file.
- Your report must:
    - State whether your code works.
    - Clearly explain how to compile and execute your code.
    - If needed/applicable, report/analyze (as appropriate) the results. The quality of analysis and writing is critical to your grade.
- Good writing and presentation are expected.

How this assignment will be graded:

The program compiles and executes with minor bugs. The code is well-designed and commented.	90% credit
The program compiles and executes with major bugs.	40% credit
The program compiles but does not produce any meaningful result.	5% credit

If the instructor needs additional communications/actions to compile and execute your code, then a 30% penalty will be applied. If the turn-in instructions are not correctly followed, 10 points will be deducted.