Regis University CC&IS – CS370

Programming Assignment 5

Documentation Due by: midnight Sunday of Week 5

Programming Due by: midnight Sunday of Week 6

Introduction

For this assignment you will create a program that allows a user to select a number base (in the range of 2 to 16, inclusive), enter digits for the number base, validate each digit as it is entered, and display the resulting value in base 2, 8, 10 and 16.

Requirements

Create a program that:

- Has six user-defined procedures (UDPs):
    - IsLegal – this UDP shall:
        - Accept an ASCII value in one register.
        - Accept a number base value in another register.
        - Determines whether the ASCII value is valid for the base.
        - Use the AsciiToDigit procedure
        - Return 1 if the value is valid.
        - Return 0 if the value is not valid.

    - AsciiToDigit – this UDP shall:
        - Accept an ASCII character value in a register.
        - Returns the numeric value related to the ASCII character value. For example, if ‘A’ (41h) or ‘a’ (61h) is passed in, then this procedure would return 10d.
        - Return -1 if the ASCII character is not valid.

    - DigitToAscii – this UDP shall:
        - Accept a numeric value in a register.
        - Returns the ASCII character value related to the numeric value. For example, if 15d is passed in, then this procedure would return ‘F’ or ‘f’ (you must choose one case for letters).
        - Return -1 if the numeric value is not valid.

    - WriteInteger – this UDP shall:
        - Accept a number in the EAX register.
        - Accept a number base value in the BL register.
        - Display the number in its base. For example, if 5 and 2 were passed in, this UDP would display 101.
        - Use the DigitToAscii procedure.
        - Not display leading zeroes.

    - ReadInteger – this UDP shall:
        - Accept a number base in the BL register.
        - Read characters from the keyboard one character at a time until the user presses the Enter key.
            - The valid characters are ‘0’ – ‘9’, ‘a’ – ‘f’ and ‘A’ – ‘F’.
        - Validate each character, as it is entered, to ensure it is a valid character for the number base. For example, entry of ‘G’ for base 7 is invalid, but ‘6’ would be valid.
            - If a character is valid, display it on the screen.
            - If a character is invalid, then do not display it.
        - Maintain a running final value as each character is entered.
            - For example, if a user chose base 7 and entered the digits 2, 6, 7, A and1:
                - The 7 and A would be ignored leaving 261_7.
                - Your running value would consist of calculating the decimal value of $(2 * 7^2) + (6 * 7^1) + (1 * 7^0) = 141$.
        - Uses the AsciiToDigit procedure.
        - Not use an array.
        - Return the result in the EAX register.
        - Return 0 if no valid characters were entered.

    - DisplayIntegers – this UDP shall:
        - Accept a number in the EAX register.
        - Use the WriteInteger procedure to display number in its Base 2, Base 8, Base 10 and Base 16 equivalents.

- Performs the following tasks in the main procedure:

    - Prompts the user for a number base value.
    -
    - Uses the ReadInteger procedure to get the base value from the user.
        - If the ReadInteger procedure call results in 0 then you will exit the program.
        - If the ReadInteger procedure call results in a value outside of the range of 2-16, then display an error message and prompt the user again.

    - Prompts the user for a number.

    - Uses the ReadInteger procedure to get the number from the user.

    - Uses the WriteInteger procedure to display the user-entered number in its Base 2, Base 8, Base 10 and Base 16 equivalents.

    - Allows the user to enter another base/number combination.

- Is formatted and commented as required by the *CS370 Coding and Documentation Standards*.

- Does not use the .IF, .REPEAT, or .WHILE directives of MASM.

Allowed Irvine Procedure Calls:

- Crlf
- ReadChar
- WriteChar
- WriteString

Sample Program Input and Output:

```
Enter base value (2 thru 16 or 0 to exit): 5
Enter a number in your chosen base: 1234

Base 2:  11000010
Base 8:  302
Base 10: 194
Base 16: C2

Enter base value (2 thru 16 or 0 to exit): 7
Enter a number in your chosen base: 1234

Base 2:  111010010
Base 8:  722
Base 10: 466
Base 16: 1D2

Enter base value (2 thru 16 or 0 to exit): 10
Enter a number in your chosen base: 1234

Base 2:  10011010010
Base 8:  2322
Base 10: 1234
Base 16: 4D2

Enter base value (2 thru 16 or 0 to exit): 16
Enter a number in your chosen base: 123abc

Base 2:  100100011101010111100
Base 8:  4435274
Base 10: 1194684
Base 16: 123ABC

Enter base value (2 thru 16 or 0 to exit): 0


Press any key to continue . . .
```

Program Documentation

Create your program document which contains:

- Algorithms as pseudo-code (C++ structure, but does not have to strictly adhere to the syntax of the language).
- A test plan and results. For an example of how to create your test plan, download the *CS370 Sample Test Plan*.
- A Unified Modeling Language (UML) Activity Diagram that shows how your program functions. For an example of how to create your activity diagram, download the *CS370 Sample Activity Diagram*.

Program Submission

Submit each of program documents to the Assignment 5 Dropbox.

Before submitting your program deliverables, you MUST name them as follows:

Lastname-Assn5-Program.asm
Lastname-Assn5-ProgramDocument.docx

For example:

Jones-Assn5-Program.asm
Jones-Assn5-ProgramDocument.docx

Grading

The following rubric will be used to grade your program.

Rating Category	**Exemplary**	**Partially Proficient**	**Basic (needs work)**	**Not Demonstrated**
Documentation	Documentation is well written, clearly explains what the code is accomplishing. Includes complete and accurate **file** and **function** headers, as detailed in the CS370 coding standards, along with additional in-line comments at all appropriate and necessary locations.	Documentation includes file header, function headers, and inline comments, but may lack clarity or details in some instances, OR may have violated some minor details of the CS370 coding standards.	Documentation is incomplete and/or incorrect and/or formatted incorrectly. Violated significant details of the CS370 coding standards or the underlying program intent.	Only a few (or no) comments in program.
Data Storage	Constants used where appropriate, correct data types used for variables. Constant, variables defined within correct program scope. Followed CS370 coding standards naming conventions and used descriptive names for all identifiers.	A few minor errors in data declaration scope, data typing or assignment, or may have violated some minor details of the CS370 coding standards.	One or more major errors, or may have violated significant details of the CS370 coding standards.	Constants not used, identifier names are not descriptive, and/or there are multiple errors in the data types assigned.
Program Input (and Input Processing)	Reads data correctly, correctly handling multiple records per line. Recognizes end of line correctly. Recognizes end of file correctly, whether the file contains a newline on the last line or not.	Last data line is duplicated or not read, but rest of lines are handled correctly OR other minor issues in reading and processing data.	Does not correctly handle multiple records per line, OR major/many issues in reading and processing data.	Was not able to read data from the file.
Program Processing: Calculations & Logic	Individual calculations and summations performed correctly. Keeps track of counts correctly. All if statements written correctly and efficiently.	Problems with ONE of the program calculations and/or if statement not written correctly and efficiently.	Multiple problems with program calculations or if statements.	All calculations incorrect.

Display Output	All required screen output is displayed neatly in columns under correct headers. Display pauses for long files.	Problems with display formatting or pausing, OR does not satisfy one of the requirements.	Multiple problems with display output.	No display output produced.
Assembly Language Constructs	Demonstrates understanding of program structure, control structures, and file structure. Appropriate use of language with correct parameter passing and no global variable usage. Follows all of the coding standards for code usage.	Most parameters are defined and passed correctly with no global variable usage, but there are minor errors (e.g. parameter passed by reference when it should be passed by value).	Multiple problems with parameter definition and/or usage, OR global variables were used OR several major errors.	Does not demonstrate any understanding or program or control/data structures. Most parameters incorrectly defined or passed. Major coding standard violations.
Modularity (functional breakdown)	Program is modular in design (if applicable, functions with arguments, not including main) and all are logically organized with prototyping. Each module performs ONE well-defined task and is defined and called correctly. Program uses code efficiently. The program modules can be easily modified and/or reused with minimal work.	Program is modular in design and is mostly logically organized. Most of program has efficient use of code. May be missing one required function, contain a module that performs too many tasks, or have one function that is incorrectly defined or called. Some of the program modules can be easily modified and/or reused with minimal work.	Program is modular in design and some parts are logically organized. Multiple problems with parameter definition and/or usage. Program has little efficient use of code. More than one required function missing, or multiple problems with module definitions/calls. Modules could not be modified and/or reused, without substantial work.	Program is not modular in design and is not logically organized. Program has little efficient use of code and program compactness. The program modules cannot be easily modified and/or debugged.
Readability/ Miscellaneous	Code is exceptionally well organized and very easy to follow and has no issues.	Code is fairly easy to read, but there are some spacing, indentation, and/or other issues.	Code is readable only by someone who already knows what it is supposed to be doing and/or there are substantial spacing, indentation, and/or other issues.	The code is poorly organized and very difficult to read and/or has other major issues.
Test Plan	Test plan includes rationale and tests that validate ALL logic.	Missing parts of rationale or missing tests of some logic.	Missing all rationale and/or missing many tests of the logic.	No test plan submitted.

Activity Diagram	Activity diagram covers ALL logic.	Missing coverage of some logic.	Missing coverage of the majority of the logic.	No activity diagram submitted.
Delivery	Submitted on time	Submitted 1-3 days late (3% deducted per day late)	Submitted 4-7 days late	Not submitted within 1 week of due date