

COSC2406 SS19 – Assembly Language Programming

Assignment 6 - Chapters 7 & 8

Due: August 12th, 2019 by 11:55pm

NOTE: Only single operand multiplication and division instructions are allowed.

Q1: [100] Create a program with an array of 12 SWORDS initialized with all zeros. Then present the user with the following menu of options:

- 1 – Populate the array with random numbers (user supplied range)**
- 2 – Multiply the array with a user provided multiplier**
- 3 – Divide the array with a user provided divisor**
- 4 – Mod the array with a user provided divisor**
- 5 – Print the array**
- 0 - Exit**

For option #1, ask the user for the low value and the high value of the random number range. Use this to call a function `genRandomNum(sdword low, sdword high)` which accepts named parameters and uses the INVOKE method of being called; the result should be returned in EAX. Use this function to populate the array. At least one local variable, created on the stack **without** the use of the LOCAL directive, must be used by the procedure.

For options #2, #3 and #4, ask the user to provide the multiplier or divisor for the required operation. For each operation, use a procedure which accepts the offset of the DWORD to be multiplied or divided and the multiplier or divisor as unnamed stack parameters. The result of the operation will be stored in the same DWORD. The stack parameters must be removed by the procedure when it returns.

For option #5, the printed values should be signed integers, comma separated, with { } brackets around the list. A procedure must be used to complete this task and parameters must be passed using either named parameters (with INVOKE) or by pushing any parameters onto the stack.

Additional Requirements:

- A. Each menu option (except exit) should call a procedure which is passed the array of numbers. Some procedures will call other procedures as described above. Use a separate procedure for each task. Even printing the menu and collecting a valid response can be a procedure called by the main method in a loop. The loop will use the return value to determine what procedure to call next or managed the exit of the program if zero is entered.
- B. ***Only option#5 will print the array values – but every other procedure should provide some appropriate message when complete.***
- C. All procedures must preserve register values whenever possible.
- D. Any procedure returning an answer must do so in either the EAX register or using an in-out stack parameter.
- E. No procedure can directly reference an array. Only use indirect operands with the offset and size of the array being passed as a **parameter on the stack**.
- F. Fully comment the code including header comments, comments at the start of each procedure, and line codes neatly in a right column. (Penalty of up to 20% for not doing this)

COSC2406 SS19 – Assembly Language Programming

- G. Your procedures must demonstrate both the use of parameter passing using unnamed parameter values (refer to these using only the EBP register) and named parameters created using a parameter list.
- H. Your procedures must demonstrate creating local unnamed stack variables (referred to using EBP) and local variables created using LOCAL.
- I. Your program must demonstrate both INVOKE and CALL operations for executing procedures.
- J. All INVOKE operations must have matching PROTO declarations.

Grading: Options 1 & 5 – 5 marks each; Options 2, 3, 4 – 15 marks each; menu and other structure 5 marks; demonstration and explanation 20 marks.

For all of these problems, a right-side column of comments must be included to explain the logic of your code. Writing the logic in Java or pseudo-code is acceptable. When I read your comments, I should clearly see your algorithm. Also, there must be proper comments at the start of every procedure as shown on pages 146-47, and comments with your name, course code, date, assignment/question number, and problem description at the top of every program. Marks will be deducted for missing commenting. *Failure to comment the code properly or to format like the code in the textbook will result in a penalty of up to 25%.*

Demonstrate and explain your working programs to the TA, Johnny Console (giconsole@algonau.ca) by Aug 14th. The demonstration and explanation of your code will represent 25% of the assignment grade – the other 75% of the grade will come from grading the code and how well the programs meet the specification and requirements of the question.

IMPORTANT:

- a) **If you do not demonstrate your programs to the TA by the end of the demonstration date, you will receive a grade of zero.**
- b) **If you are unable to explain most or all of your code and the logic of your programs, then the maximum grade possible will be 50%.**