COMSC 260

Fall 2019

Programming Assignment 11

Worth 11.36 points (1.136% of your grade)

DUE: Monday, 12/9/19 by 11:59 P.M. on Canvas

START by downloading the 260_assign11.asm file from Canvas

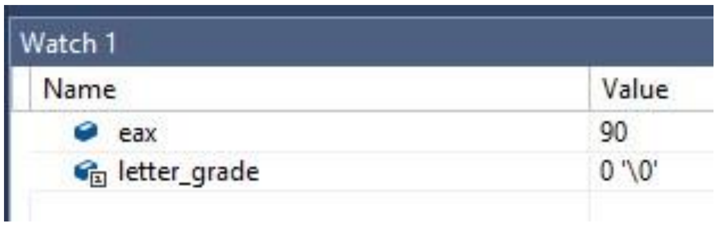
NOTE: Your submission for this assignment should be a single **.asm** file and a single **.pdf** file.
The following naming convention should be used for naming your files: **firstname_lastname_260_assign11.asm and firstname_lastname_260_assign11.pdf**. The pdf file that you submit should contain the screenshots of your sample runs of the program (see below). For example, if your first name is "James" and your last name is "Smith", then your files should be named James_Smith_260_assign11.asm and James_Smith_260_assign11.pdf.

COMMENTS (worth 5% of your programming assignment grade): Your program should have at least **ten (10)** different detailed comments explaining the different parts of your program. Each individual comment should be, at a minimum, a short sentence explaining a particular part of your code. You should make each comment as detailed as necessary to fully explain your code. You should also number each of your comments (i.e., comment 1, comment 2, etc.). **NOTE: My comments do NOT count towards the ten comments!**

SAMPLE RUNS (worth 5% of your programming assignment grade): You should submit screenshots of at least **five (5)** different sample runs of your program. Each sample run needs to use a different input for register EAX, and your sample runs should **NOT** be the same as the sample runs that are used in this write-up for the assignment.

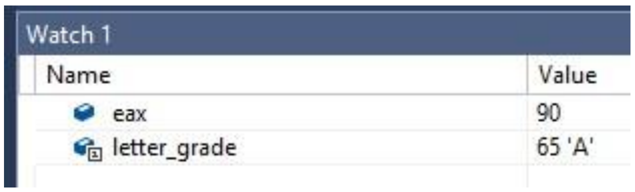
You should also number each of your sample runs (i.e., sample run 1, sample run 2, etc.). All of your sample runs should follow this format – for each individual sample run, screenshot (1) the value placed into register EAX at the beginning of the program and (2) the value in the letter_grade variable at the end of the program. For example:

(1)



Name	Value
eax	90
letter_grade	0 '\0'

(2)



Name	Value
eax	90
letter_grade	65 'A'

IMPORTANT NOTE: In this class (for this assignment and ANY other assignment) **the high-level directives from section 6.7 of chapter 6 are NOT allowed to be used under any circumstances.**

This means things such as .if, .else, .repeat, .while, etc. must **NOT** be used in this assignment, or any other assignment in this class. Use of these high-level directives would make the assignments too easy, so they are **NOT** allowed for this reason.

When your program needs to make decisions, it needs to do so using the conditional jump instructions that we are learning about in class.

OK: conditional jump instructions such as je, jne, jecxz, ja jnae, jg jcxz, jp, jnp, jb, jnbe, jp, jz, js, etc.

NOT OK: High-level directives such as .if, .while, .else, .repeat, etc. (**do NOT use anything from section 6.7 of chapter 6 in any of your assignments!!!**)

For your programming assignment you will be implementing the following procedure:

calculate_grade
Uses the numeric score in eax to determine the corresponding letter grade based on the following criteria*: - If the score is 90 – 100, the grade is an A - If the score is 80 – 89, the grade is a B - If the score is 70 – 79, the grade is a C - If the score is 60 – 69, the grade is a D - If the score is less than 60, the grade is an F

*The grading scale could be changed at any time by changing the A_grade, B_grade, C_grade, and D_grade constants in the program. The program should still work correctly even if these constants were changed in the future.

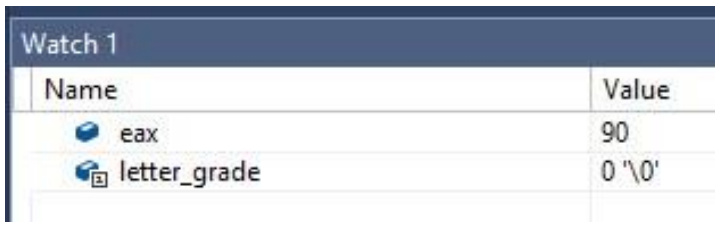
The idea is that in main, a number between 0 – 100 will be placed in register eax. Then, main will call the calculate_grade procedure. The calculate_grade procedure will determine the letter grade based on the value in eax, and then move the letter grade (‘A’, ‘B’, ‘C’, ‘D’, or ‘F’) into the letter_grade variable in the data segment.

For example, if value in eax is 92, then ‘A’ should be moved into letter_grade.

If the value in eax is 74, then ‘C’ should be moved into letter_grade, and so on.

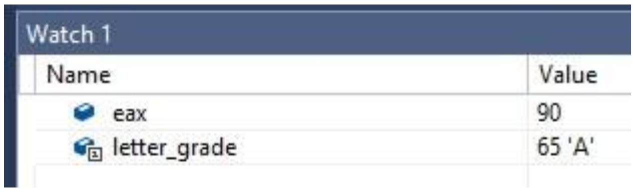
Sample run 1:

Initial state – BEFORE calling calculate_grade (90 is placed into eax)



Name	Value
eax	90
letter_grade	0 '\0'

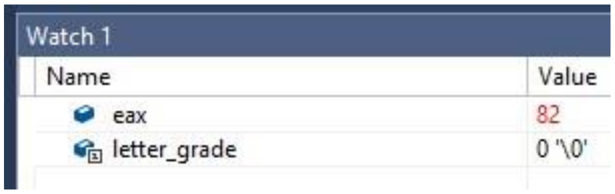
Final state – AFTER calling calculate_grade (‘A’ is placed into letter_grade because eax is 90)



Name	Value
eax	90
letter_grade	65 'A'

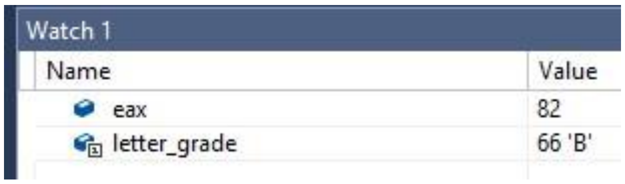
Sample run 2:

Initial state – BEFORE calling calculate_grade (82 is placed into eax)



Name	Value
eax	82
letter_grade	0 '\0'

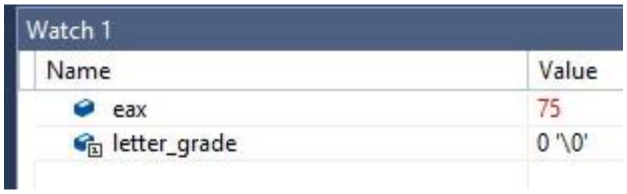
Final state – AFTER calling calculate_grade (‘B’ is placed into letter_grade because eax is 82)



Name	Value
eax	82
letter_grade	66 'B'

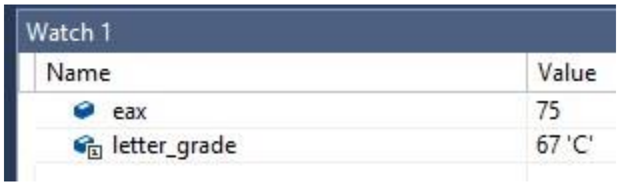
Sample run 3:

Initial state – BEFORE calling calculate_grade (75 is placed into eax)



Name	Value
eax	75
letter_grade	0 '\0'

Final state – AFTER calling calculate_grade (‘C’ is placed into letter_grade because eax is 75)



Name	Value
eax	75
letter_grade	67 'C'