Project Descriptions:

In this project, we are going to develop three programs as specified below using PC-SPIM (MIPS assembly instructions):

(1) IF ~ THEN ~ ELSE structure
(2) WHILE structure
(3) floating-point fraction numbers

The descriptions and requirements for each control structure are as follow:

(1) IF ~ THEN structure

When your program starts, your program prompts a user to enter two numbers. The first number is the number of the academic credit hours you have completed at SIUE, which should take an integer in the range of 0 through 280 (0 and 280 are included). The second number is the number of the academic credit hours you have transferred to SIUE, which should take an integer in the range of 0 through 36 (0 and 36 are included). The two numbers should be prompted by your program, one at a time (the credit hours at SIUE, followed by the credit hours transferred from other institutes). When a user enters any invalid number, your program should detect it and repeat the prompt to a user. When a user made an invalid input to the transferred credit hours, your program should prompt only for the transferred credit hours, but not for the credit hours completed at SIUE. After the two numbers are successfully entered, your program should determine your academic level based on the following grade criteria:

0-29: freshman
30-59: sophomore
60-89: junior
90 or more: senior

For example, if a user earned 48 for his (her) the credit hours earned at SIUE and 20 credit hours transferred from other academic institutions, “You are a junior” should be displayed. For each invalid input, “You made an invalid input for X. Enter a valid input for X.,” where X is:

For the first input: “the credit hours completed at SIUE”

For the second input: “the credit hours transferred from other academic institutions”

Requirements:

a. The above procedure must be implemented using IF-THEN structures implemented in MIPS assembly instructions.
b. Each input must be tested to confirm its validity

c. After the first input (a course score) is correctly made, the first input should never be repeated when a user makes an invalid input for the second input (i.e., “the credit hours transferred from other academic institutes”).

d. Your program should output a correct academic level (no garbage output, and etc.). See the following screen snapshot of the program.

e. Your program should terminate without any error or a warning message.

(2) WHILE structure

This program calculates a product of two numbers without using assembly instruction for multiplication (e.g., “mult”). When your program starts, your program prompts a user to enter two numbers (each number should be larger than 0 and less than 50). The product of two numbers should be calculated by adding the first number as many times as the second number. When a user enters any invalid number, your program should detect it and repeat the prompt to a user.

Requirements:

a. The above program should be implemented using a loop structure implemented in MIPS “add” assembly instructions, but not using any multiplication instruction.

b. Each input must be tested to confirm if it is a valid input.

c. If an input number is less than 0, display an error message, “the entered number is less than 0”. Then, prompt a user to make another input.

d. If an input number is larger than 50, display an error message, “the entered number is larger than 50”. Then, prompt a user to make another input.
e. After the first input is correctly made, the first input should never be repeated when a user makes an invalid input for the second input.

f. Your program should correctly calculate the product of the two numbers by using only instruction for addition (no multiplication).

g. Your program should terminate without any error or a warning message.

(3) floating-point fraction numbers

Most of processors handle fraction numbers using a different set of instructions (different from those integers), called “floating-point instructions”. This program first prompts a user to enter three numbers: the principal ($1.00-$500.00), the interest rate (1.00 to 50.00%) as a fraction number 0.01 to 0.50, and the target account balance. Note that the target account balance must be larger than the principal. Then, this program calculates how long (in terms of the number of years) will it take to achieve the target account balance. Your program should calculate the number of years as an integer. Thus, the number of the years necessary to achieve the target account balance should be rounded up, if the number is a fraction of a year. Some sample PC-SPIM assembly source code files will be provided for this program*1.

*1: “for loading a constant fraction number to a floating-point register” and “for copying the content of a floating-point register to another floating-point register”.

For Program #3, the following MIPS (PC-SPIM) floating-point instructions are suggested:

- add.s – single-precision floating-point addition
- mul.s – single-precision floating-point multiplication
- lwc1 – loading a single-precision floating-point constant from a memory address
- c.lt.s – floating-point numerical comparison of two floating-point registers
- bc1t – a conditional branch instruction used with c.lt.s

Requirements:

a. The above program should be implemented using MIPS assembly instructions.

b. If an invalid input is made, an error message should be displayed and your program should prompt a valid input.

c. After the first input or the second is successfully made, the first or the second input should never be prompted, especially when an invalid input is made to the second or the third input, except for a case if the target account balance is less than the principal.

d. The correct number of years that is necessary to achieve the target account balance should be displayed.

e. Your program should terminate normally (should not crash the system when your program leaves the system).
Submission:

Your source code files (one source code file for each of the three programs) should be e-mailed to Dr. Fujinoki (hfujino@siue.edu) by 1:15 P.M. on June 26th. Your program hard copy should be submitted at 1:15 P.M. in the classroom on June 26th. Failing to submit your program hard copy by 1:15 P.M. on June 26th will result in -5 points penalty to your project #1 grade (in 100-point scale).

- Extra-Credit Early-Submission*2 Deadline: 11:59:59 P.M., June 19th
- Free-Feedback Early-Submission*2 Deadline: 11:59:59 P.M., June 23rd
- Final Deadline: 1:15PM, June 26th

Note 1: Each program flow-control should be implemented as an independent program.

Note 2: In your program source code, include your name, student ID in your program header.

Note 3: It is required to use PC-SPIM posted to the CS312 course homes (programs developed for different versions of PC-SPIM will NOT be accepted for grading).

Note 4: This programming project is an individual programming project.

Note 5: Academic dishonesty in Project #1 will at least result in 0 point for the entire programming project credit for this course.

Late Submission:

- Penalty of -10% will be given for every 24 hours after the due (i.e., -10% for a submission within the first 24 hours after the due).
- Submission more than 48 hours after the due will not be accepted.

*2: No “late submission” for early deadlines
Grading Criteria:

- **Assembler error**: If Dr. Fujinoki cannot run your assembler source code file due to any assembler error: 0 ~ 20% of the credit for this assignment phase will be given.

- **Run-time error**: After your assembly source code file is successfully assembled to machine codes, if your program does not satisfy any requirement: 5% penalty for each minor problem. For any major failure, penalty depends on each such major problem.

- **The weight** of the three programs is: 25%, 25%, and 50% for program 1 (if-then), program 2 (while), and program 3 (floating-point) respectively.