Skill Assessment #2 Assembly Program (.asm)

● Graded

Student

HARRY KIM

Total Points

50 / 50 pts

Autograder Score

50.0 / 50.0

Passed Tests

1.1) File named properly: Assignment_02.asm (5/5)
2.1) Assemble the code (5/5)
3.1) Simple test run: (5/5)
3.2) Simple test run: (5/5)
3.3) Simple test run: (5/5)
3.4) Simple test run: (5/5)
3.5) Simple test run: (5/5)
3.6) Simple test run: (5/5)
3.7) Simple test run: (5/5)
3.8) Simple test run: (5/5)

Question 2

Requirements met (Didn't use more advanced instructions such as shifts, multiplication, or branching (looping) instructions and is less than 25 lines of code) **0** / 0 pts

– 20 pts Used shifts, multiplication, or branching

– 10 pts Code not shorter than about 25 lines of code

✓ **– 0 pts** Correct

Question 3

On time (deductions for late, etc.) **0** / 0 pts

✓ **– 0 pts** On time

– 5 pts Late submission

Autograder Results

Assignment 02 Automated Tests

1.1) File named properly: Assignment_02.asm (5/5)

```
My script searches for and renames files as appropriate.
Searching for .asm files (any .asm is OK)...
submission/Assignment_02.asm found.  Copying it to the test directory as Assignment_02.asm
```

2.1) Assemble the code (5/5)

```
Program assembled OK.
```

3.1) Simple test run: (5/5)

```
Your program's input/output is shown below.

Please enter a non-zero int test number: 123
Your solution produces this result: 563
(End of your program's output.)

Your output contains an exact match for the expected output.
```

3.2) Simple test run: (5/5)

```
Your program's input/output is shown below.

Please enter a non-zero int test number: 2021
Your solution produces this result: 116
(End of your program's output.)

Your output contains an exact match for the expected output.
```

3.3) Simple test run: (5/5)

```
Your program's input/output is shown below.

Please enter a non-zero int test number: 42
Your solution produces this result: 513
(End of your program's output.)

Your output contains an exact match for the expected output.
```

3.4) Simple test run: (5/5)

Your program's input/output is shown below.

```
Please enter a non-zero int test number: 123456789
Your solution produces this result: 374
(End of your program's output.)
```

Your output contains an exact match for the expected output.

3.5) Simple test run: (5/5)

Your program's input/output is shown below.

```
Please enter a non-zero int test number: 8675309
Your solution produces this result: 262
(End of your program's output.)
```

Your output contains an exact match for the expected output.

3.6) Simple test run: (5/5)

Your program's input/output is shown below.

```
Please enter a non-zero int test number: 3810
Your solution produces this result: 568
(End of your program's output.)
```

Your output contains an exact match for the expected output.

3.7) Simple test run: (5/5)

Your program's input/output is shown below.

```
Please enter a non-zero int test number: 5005
Your solution produces this result: 658
(End of your program's output.)
```

Your output contains an exact match for the expected output.

3.8) Simple test run: (5/5)

Your program's input/output is shown below.

```
Please enter a non-zero int test number: -999
Your solution produces this result: 566
(End of your program's output.)
```

Your output contains an exact match for the expected output.

Submitted Files

Assignment_02.asm   Download

```
# This small assembly language program is part of problem S4
# in assignment #2.  Students are to fill in one piece of
# code below, while leaving the rest of the program entirely
# unchanged.  (Any changes to any other parts of the program
# or any changes to comments outside of the student work
# area may cause tests to fail.)
#
# Instructions:
#      1. Locate the clearly-marked student work section below.
#      2. Put your name -after- the # Author:
#      2. Do not change other provided comments or code.
#      3. Place your solution to S4 within the work section.
#          (You will add additional lines.)
#      4. Test/debug your solution
#      5. Turn in this file (in its entirety).
#
# Notes:  My test program will alter code outside of the
# student work section.  Your solution must exist entirely
# within the student work section.
#
# Feel free to examine the rest of the code.
#
# CS/ECE 3810 - Fall 2021

# The data area of memory:

        .data

# In this problem, you'll treat the memory locations
# below, labeled 'Cypher', as an array.  I fill in this
# array with randomized data (based on the test number).
# The array is 32 integers long (128 bytes long), but
# you'll only use a few of them.

Cypher:
        .space 128

# These are the text messages for my portions of the code.

Prompt:
        .asciiz "Please enter a non-zero int test number: "

Result:
        .asciiz "Your solution produces this result: "

# The text, or program, area of memory:

        .text

```

```
# Note that I don't need a label here.  The first instruction
# of the text section is the first instruction that will be
# executed.

# Ask the user for a test number.

        la $a0, Prompt   # Put the address of the string into $a0
        li $v0, 4        # 4 -- Syscall for print string
        syscall

# Gather input, keep it as the seed.

        li $v0, 5        # 5 -- Syscall for read intenger
        syscall
        move $t4, $v0    # $t4 is the test number (random seed)

# Fill the data array repeatedly with random digits (so
# that the student solution has interesting digits to
# extract and print).  Note that I have written this
# code in a little bit of a confusing fashion to make it
# difficult to determine correct answers without
# either debugging the program or writing a correct
# solution.

        li $t0, 996      # $t0 is my loop and address counter
RandomLoop:
        andi $t5, $t4, 0x00000007
        addi $t5, $t5, 1
        la   $t6, Cypher
        andi $t1, $t0, 0x0000007c
        add  $t1, $t1, $t6
        sw   $t5, 0($t1)
        srl  $t5, $t4, 20
        srl  $t6, $t4, 23
        sll  $t4, $t4, 8
        xor  $t1, $t5, $t6
        andi $t1, $t1, 0x0000007f
        or   $t4, $t4, $t1
        addi $t0, $t0, -4
        bne  $t0 $zero, RandomLoop

# At this point, the array is filled with random digits.
# (It is ready for the student to solve S4.)
# Students may not assume that registers have specific
# values at this point.  Initialize any registers
# you use.

###########################################################
# Student work section BEGIN:  Put your code below.
# Author: Harry Kim

la $t6, Cypher # Loading in Cypher array
```

```
lw $t0,32($t6) # Ones place Cypher[8]
lw $t1,36($t6) # Tens place Cypher[9]
lw $t2,40($t6) # Hundreds place Cypher[10]

# "Multiplying" tens place.
add $t4, $t1, $t1
add $t3, $t4, $t4
add $t3, $t3, $t3
add $t1, $t3, $t4

# "Multiplying" hundreds place.
add $t4, $t2, $t2 # Do "times ten" twice instead of "times one hundred."
add $t3, $t4, $t4
add $t3, $t3, $t3
add $t2, $t3, $t4
add $t4, $t2, $t2
add $t3, $t4, $t4
add $t3, $t3, $t3
add $t2, $t3, $t4

# adding all numbers and setting it to $v0.
add $t4, $t0, $t1
add $v0, $t4, $t2

# Student work section END:  Put your code below.
############################################################

# At this point, the student solution should be in $v0.
# My code will print out $t0 below.

# Save the student's answer, $v0, for later use.

        move $s1, $v0

# Print out the result message.

        la $a0, Result   # Put the address of the string into $a0
        li $v0, 4        # 4 -- Syscall for print string
        syscall

# Print out the student's answer as an int.

        move $a0, $s1    # Put the student's answer in $a0
        li $v0, 1        # 1 -- Syscall for print int
        syscall

# Print out a newline.

        li $a0, 10       # 10 is the ASCII code for a newline
        li $v0, 11       # 11 -- Syscall for print ASCII char
        syscall

```

```
# Exit cleanly (instead of just falling off the bottom of the code).

        li $v0, 10       # 10 -- Syscall for exit program
        syscall




```