CPE301 – SPRING 2020

Design Assignment 1A

Student Name: Minsung Cho

Student #: 2001446442

Student Email: chom3@unlv.nevada.edu

Primary Github address: <https://github.com/cho-minsung>

Directory: https://github.com/cho-minsung/assignment1A.git

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

A computer

1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A**

No code was initial.

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**

|  |
| --- |
| .DEF RESULT1 = R31 ;define results. They are defined so that it reads from left to right. |
| .DEF RESULT2 = R30 |
| .DEF RESULT3 = R29 |
| .DEF RESULT4 = R28 |
| .DEF RESULT5 = R27 |
| .DEF RESULT6 = R26 |
| .DEF RESULT7 = R25 |
| .DEF RESULT8 = R24 |
| .DEF MULTIPLICAND4 = R16 ;define multiplicand |
| .DEF MULTIPLICAND3 = R17 |
| .DEF MULTIPLICAND2 = R18 |
| .DEF MULTIPLICAND1 = R19 |
| .DEF MULTIPLIER4 = R20 ;define multiplier |
| .DEF MULTIPLIER3 = R21 |
| .DEF MULTIPLIER2 = R22 |
| .DEF MULTIPLIER1 = R23 |
|  |
| LDI RESULT1, 0x0 ;clear result |
| LDI RESULT2, 0x0 |
| LDI RESULT3, 0x0 |
| LDI RESULT4, 0x0 |
| LDI RESULT5, 0x0 |
| LDI RESULT6, 0x0 |
| LDI RESULT7, 0x0 |
| LDI RESULT8, 0x0 |
| LDI MULTIPLICAND4, 0x54 ; 1410685718 as an example |
| LDI MULTIPLICAND3, 0x15 |
| LDI MULTIPLICAND2, 0x5b |
| LDI MULTIPLICAND1, 0x16 |
| LDI MULTIPLIER4, 0x3e ; 1041407088 is also the random code. |
| LDI MULTIPLIER3, 0x12 |
| LDI MULTIPLIER2, 0x9c |
| LDI MULTIPLIER1, 0x70 |
| SUB RESULT8, RESULT8 ;to clear the carry |
|  |
| TESTB1: ;these test loops are to check if the multiplier is not zero. if it not zero, it subtracts 1 and add the multiplicand to the result. |
| TST MULTIPLIER1 |
| BREQ TESTB2 |
| JMP LOOP |
|  |
| TESTB2: |
| TST MULTIPLIER2 |
| BREQ TESTB3 |
| JMP LOOP |
|  |
| TESTB3: |
| TST MULTIPLIER3 |
| BREQ TESTB4 |
| JMP LOOP |
|  |
| TESTB4: |
| TST MULTIPLIER4 |
| BREQ END |
| JMP LOOP |
|  |
| LOOP: ;I used the archaic way of adding the multiplicand many times(number of multiplier) because although this may take a lot of cycles, it does its job. |
| ;I tried to do it the shift and add way but I couldn't figure out how to, since the architecture is very different from the familiar MIPS. |
| SUBI MULTIPLIER1, 1 |
| SBCI MULTIPLIER2, 0 |
| SBCI MULTIPLIER3, 0 |
| SBCI MULTIPLIER4, 0 |
| ADD RESULT1, MULTIPLICAND1 |
| ADC RESULT2, MULTIPLICAND2 |
| ADC RESULT3, MULTIPLICAND3 |
| ADC RESULT4, MULTIPLICAND4 |
| JMP TESTB1 ;This will go through a lot of time and end with the infinite loop at the end. |
|  |
| END: |
| JMP END |

1. **SCHEMATICS**

No schematics

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

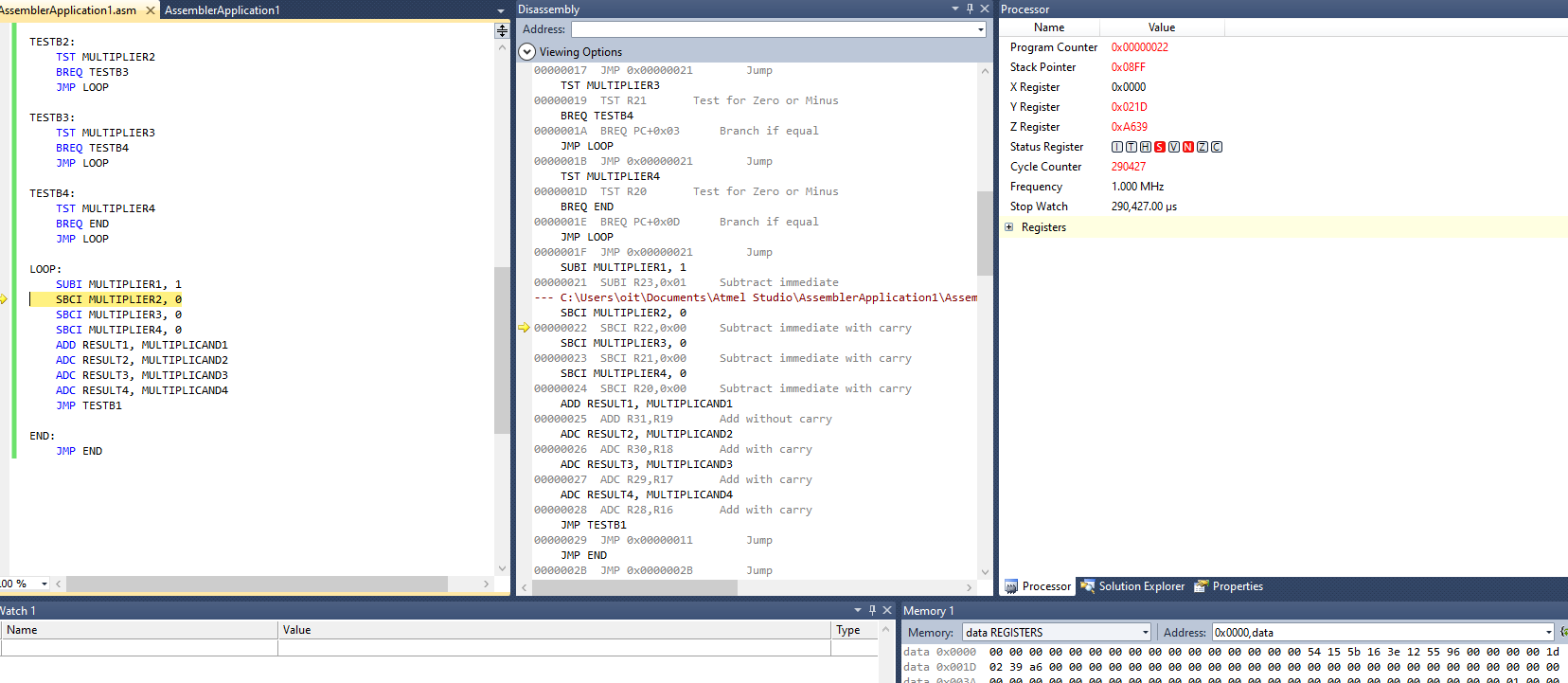


Figure 1. at the beginning

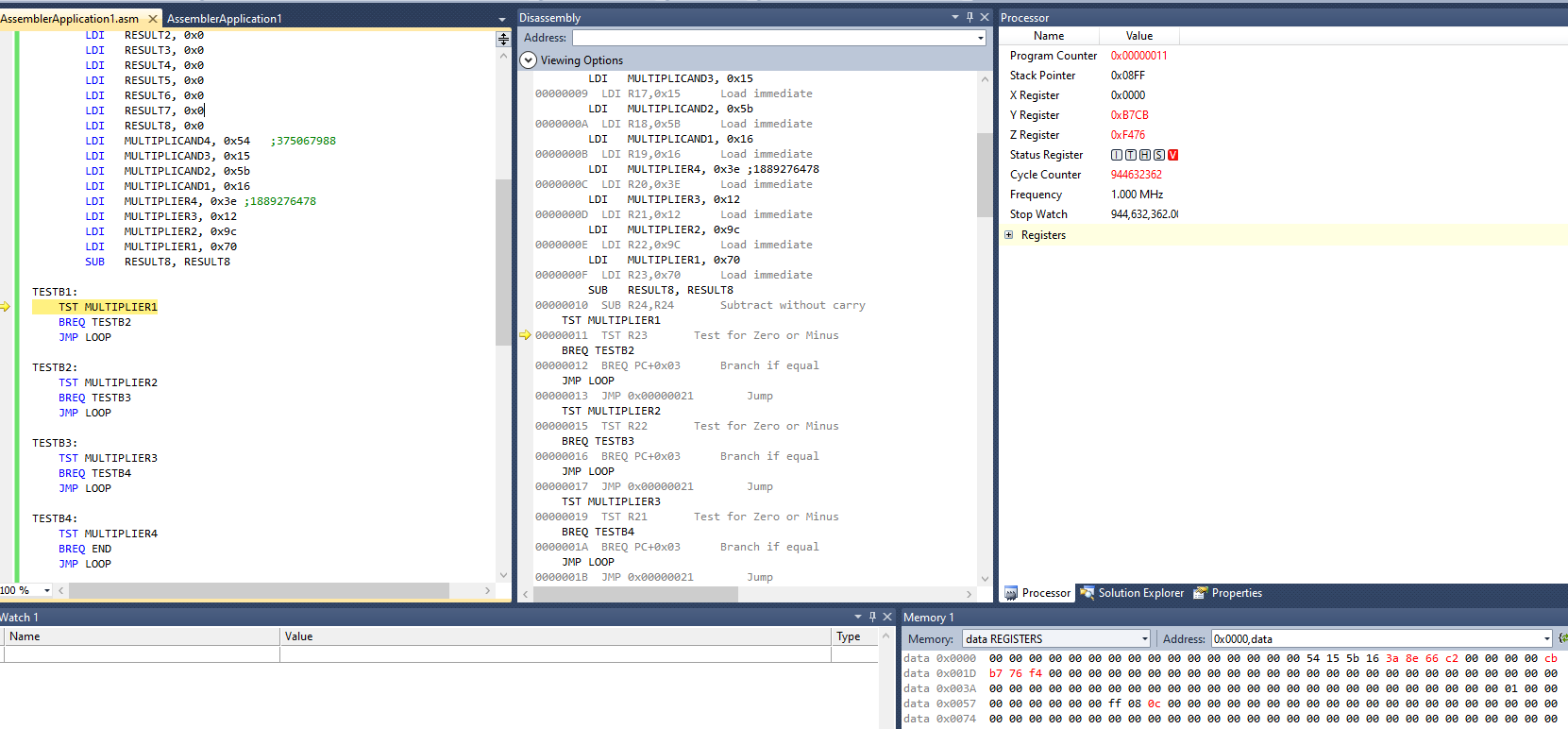


Figure 2. many cycles later…

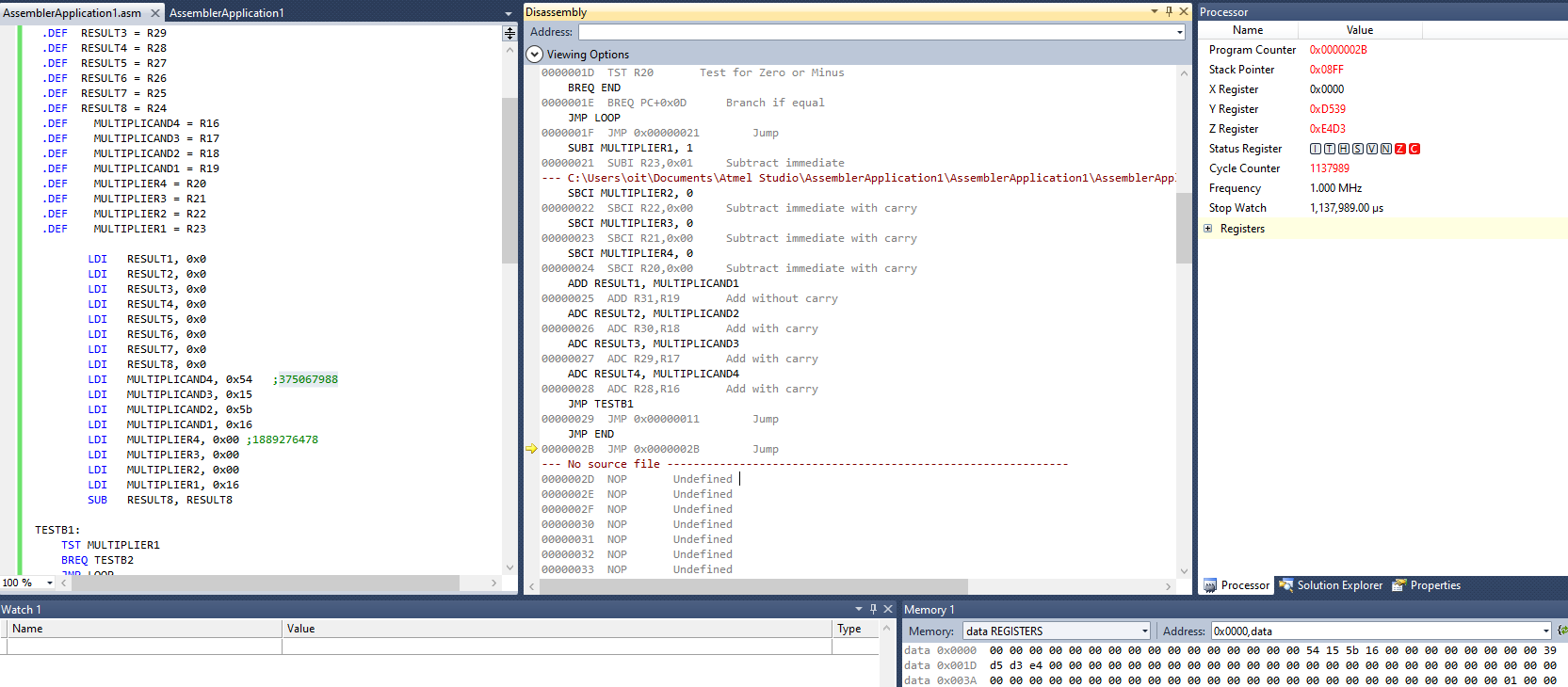


Figure 3. As you can see, with a simpler multiplier, you know the code works.

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

No board

1. **VIDEO LINKS OF EACH DEMO**

No link

1. **GITHUB LINK OF THIS DA**

<https://github.com/cho-minsung/assignment1A>

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Minsung Cho