CPE301 – SPRING 2019

Design Assignment 1a

Student Name: Daniel Senda

Student #: 5002362633

Student Email: sendad1@unlv.nevada.edu

Primary Github address: https://github.com/dsenda/Smiles

Directory: DA1a

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with YouTube video links (see template).

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

Used Atmel Studio 7 for this assignment.

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

The student decided to use the shift and add method to multiply two numbers without using the mul assembly instruction. The finished code, along with comments explaining every instruction, follows.

; DA1a.asm

; Daniel Senda

.org 0x000

start:

ldi r21, 0x0 ;Set register r23 to zero, "Zero register".

ldi r18, 0x0 ;Set register r24 to zero. "Low byte of final answer".

ldi r19, 0x0 ;Set register r24 to zero. "Mid byte of final answer".

ldi r20, 0x0 ;Set register r24 to zero. "High byte of final answer".

ldi r23, 0x0 ;Set register r23 to zero. "Overflow sum of multiplicand."

ldi r24, 0b11111111 ;Load low byte of multiplicand.

ldi r25, 0b11111111 ;Load high byte of multiplicand.

ldi r22, 0x11111111 ; Load 8-bit multiplier.

main\_loop:

lsr r22 ;Logical shift right of multiplier.

brcs lsb\_multiplier ;Branch to lsb\_multiplier if carry is set.

add r22, r21 ;add "zero register" and multiplier.

breq end ;If last result is zero, branch to end.

lsl r24 ;Logical shift left low byte of multiplicand.

rol r25 ;Rotate left with carry high byte of multiplicand.

rol r23 ;Rotate left with carry overflow sum of multiplicand.

jmp main\_loop ;Jump to main loop.

lsb\_multiplier:

add r18, r24 ;Add low byte of multiplicand with low byte of answer register.

adc r19, r25 ;Add with carry high byte of multiplicand with mid byte of answer register.

adc r20, r23 ;Add overflow sum of multiplicand with the high byte of answer register.

lsl r24 ;Logical shift left low byte of multiplicand.

rol r25 ;Rotate left with carry high byte of multiplicand.

rol r23 ;Rotate left with carry overflow sum of multiplicand.

jmp main\_loop ;Jump to main loop.

end: jmp end ;End loop, signifying that it finished.

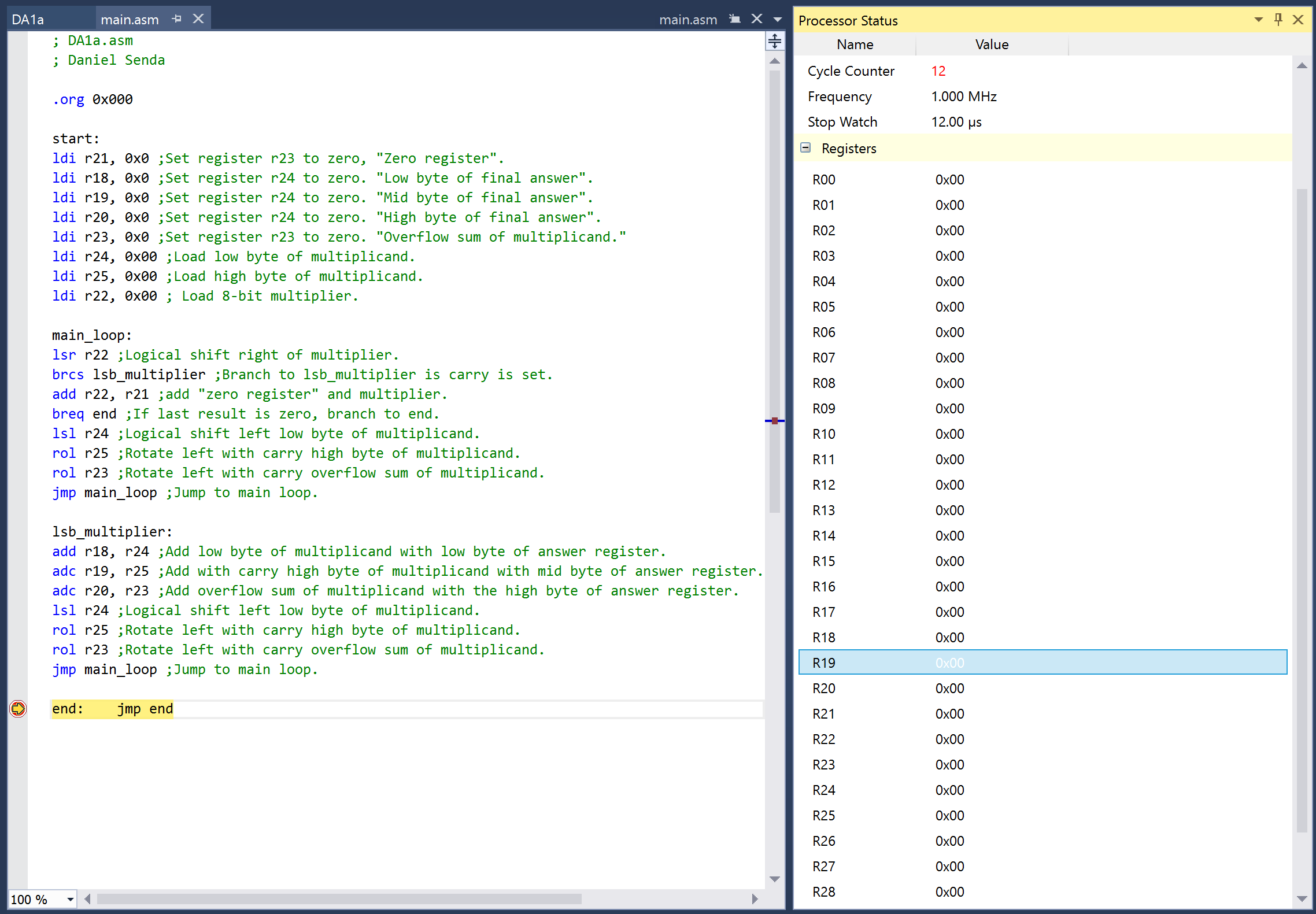
1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**

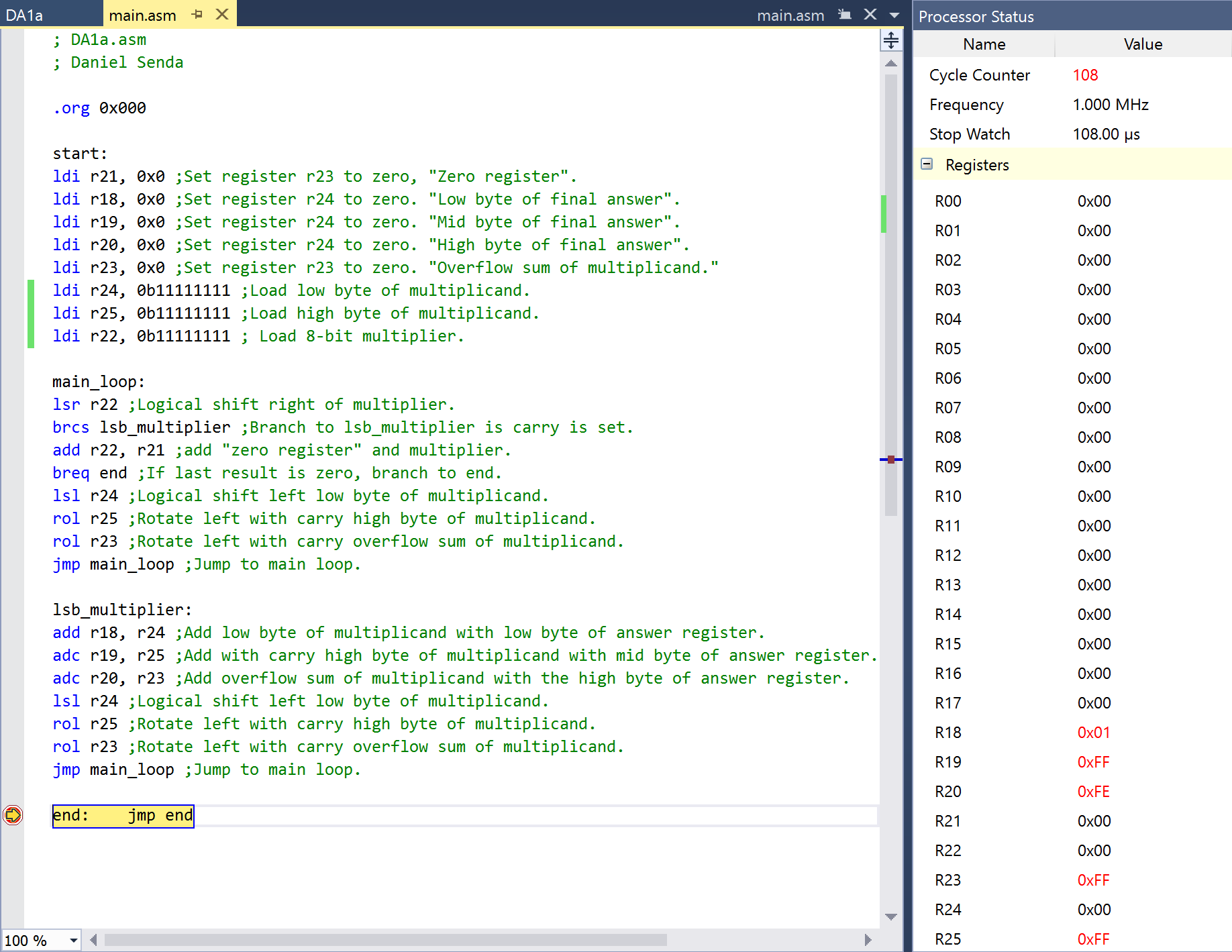
Not applicable.

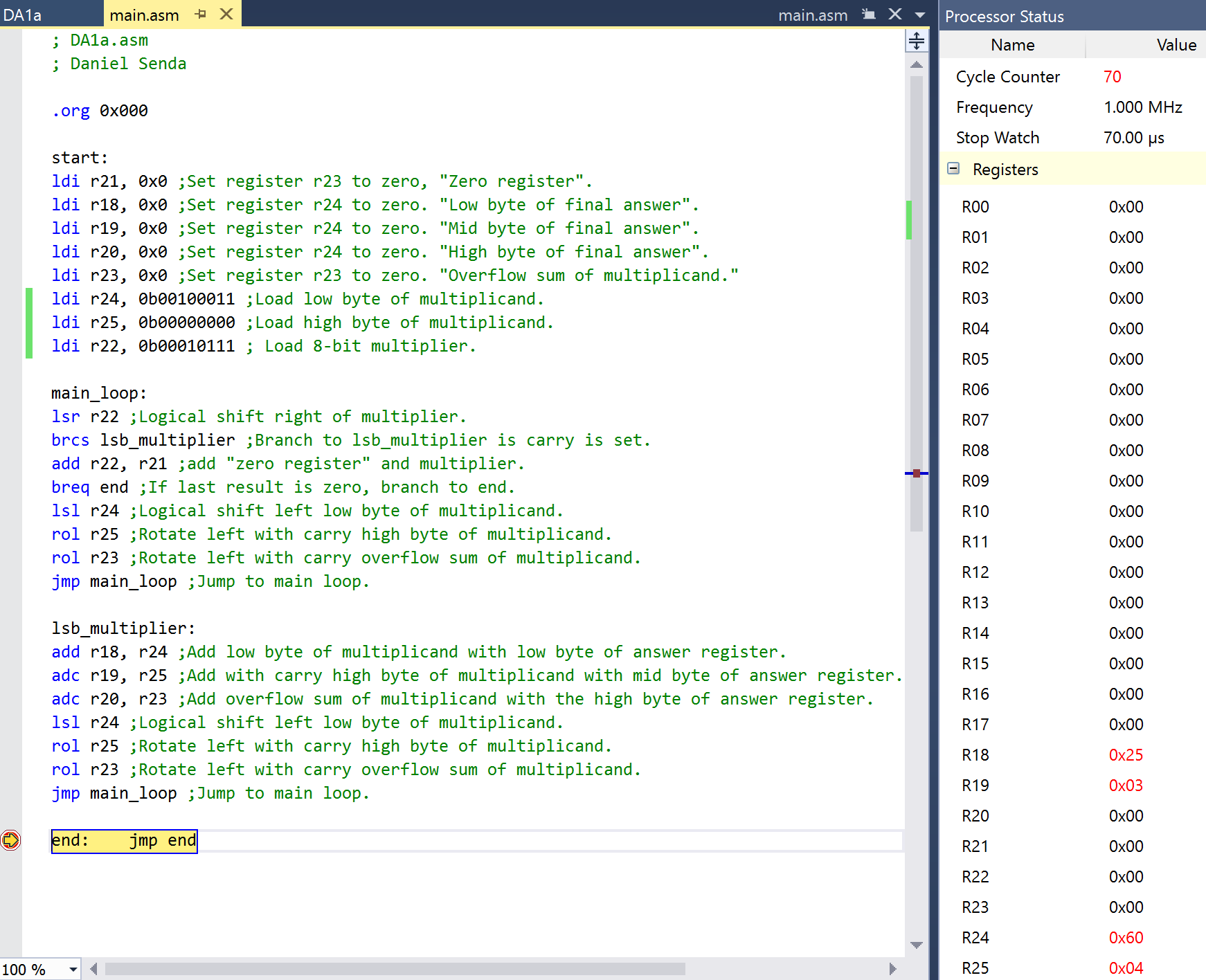
1. **SCHEMATICS**

Not applicable.

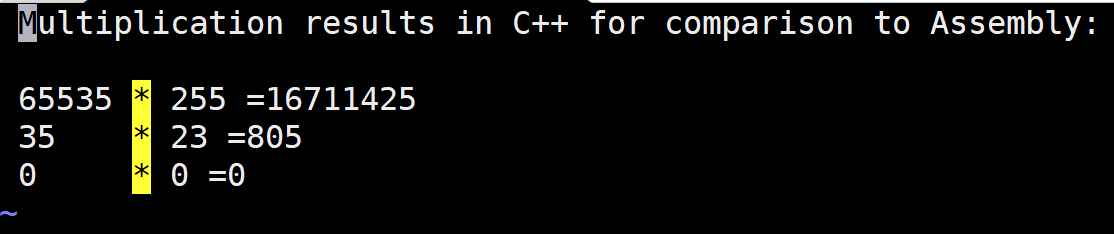
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

The following shows that 0\*0 = 0, as can be seen from the registers (R18-R20). This multiplication process takes up the least amount of clock cycles, which is 12.  


This calculation is 65535\*255 = 16,711,425 = 0xFEFF01. This is the longest run with 108 cycles.  


This calculation is 35\*23 = 805 = 0x325. This calculation took 70 cycles.  


After getting the results from the assembly code, they were confirmed using the following C++ code.  


The output file gave the following results, confirming the assembly results.  


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

Not applicable.

1. **VIDEO LINKS OF EACH DEMO**

Not applicable.

1. **GITHUB LINK OF THIS DA**

https://github.com/dsenda/Smiles/tree/master/DA1a

**Student Academic Misconduct Policy**

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

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

Daniel Senda