Daniel Senda

CPE 310 DA1a

Due: 02-16-2019

**Design Assignment 1a**

Design assignment 1a had the student write, simulate, and demonstrate an assembly code using the AVR ATMEGA328p microcontroller. The following are the requirements:

* Perform a multiplication of a 16-bit multiplicand with an 8-bit multiplier without using the MUL instruction. Use iterative addition to perform the above multiplication.
* Registers R25:R24 hold the 16-bit multiplicand, R22 hold 8-bit multiplier, and R20:R19:R18 should hold the result.
* Verify your algorithm and answers using the AVR mul instruction or C or any high-level program.
* Determine the execution time @ 16MHz/#cycles of your algorithm using the simulation.

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.

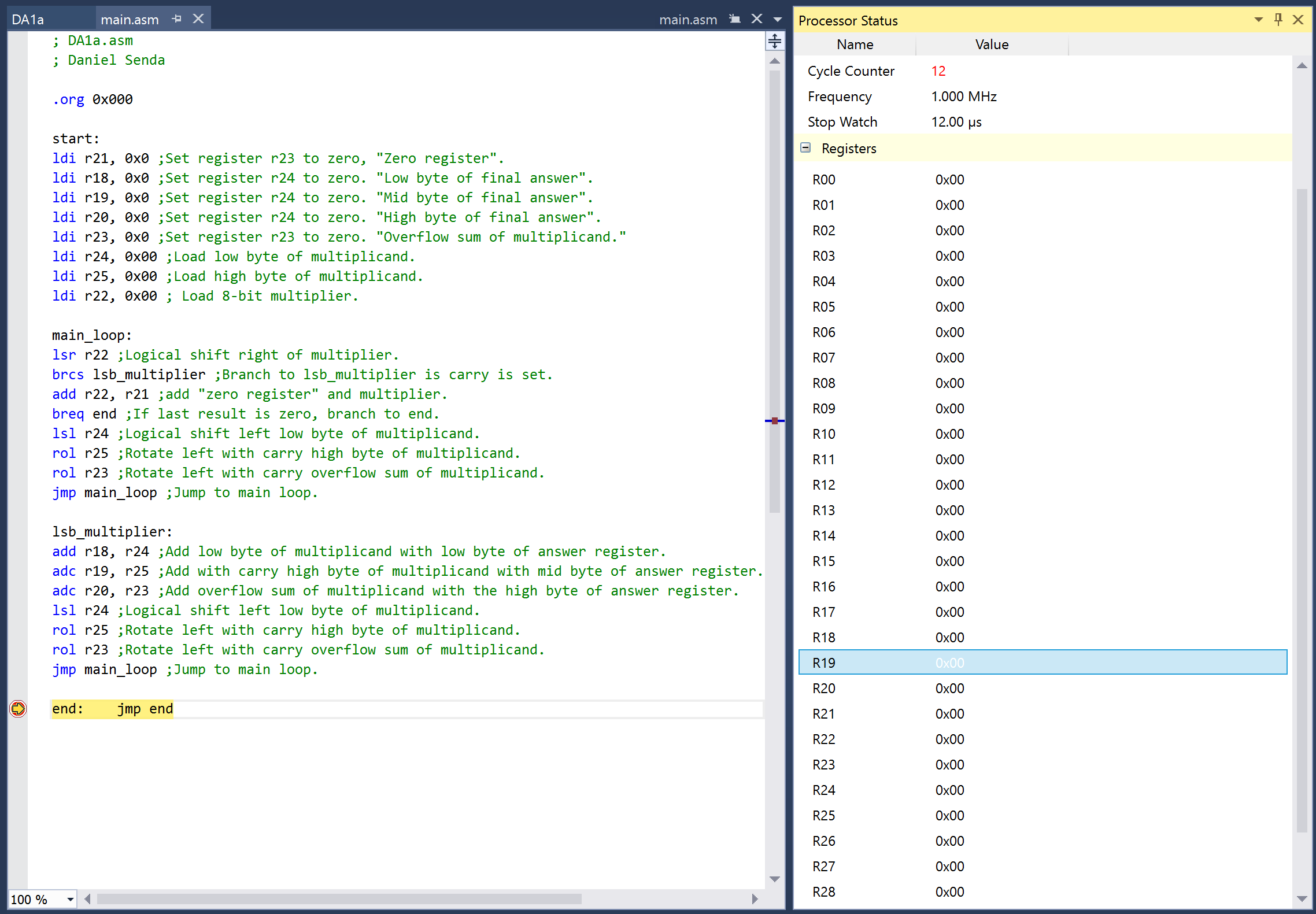
lsl r24 ;Logical shift left low byte of multiplicand.

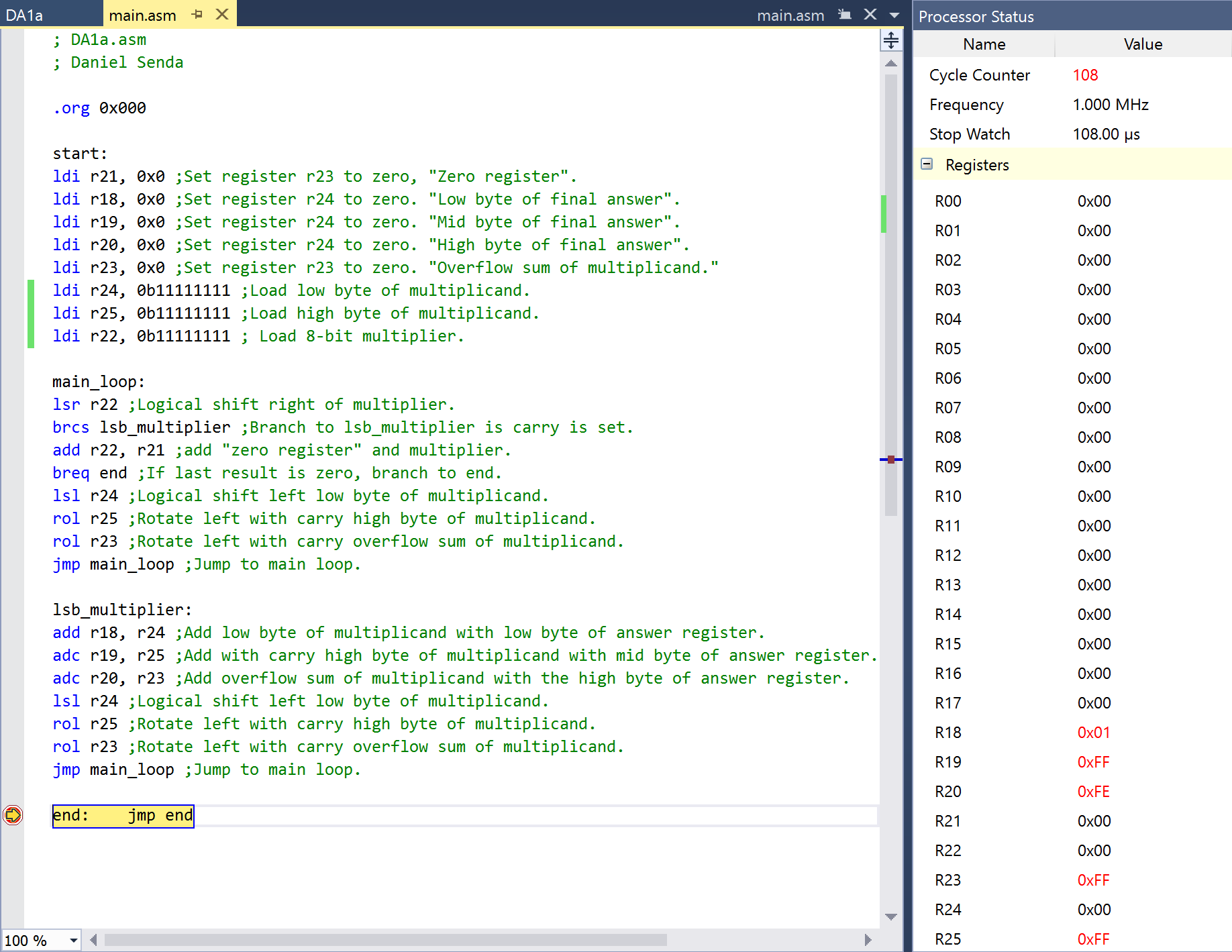
rol r25 ;Rotate left with carry high byte of multiplicand.

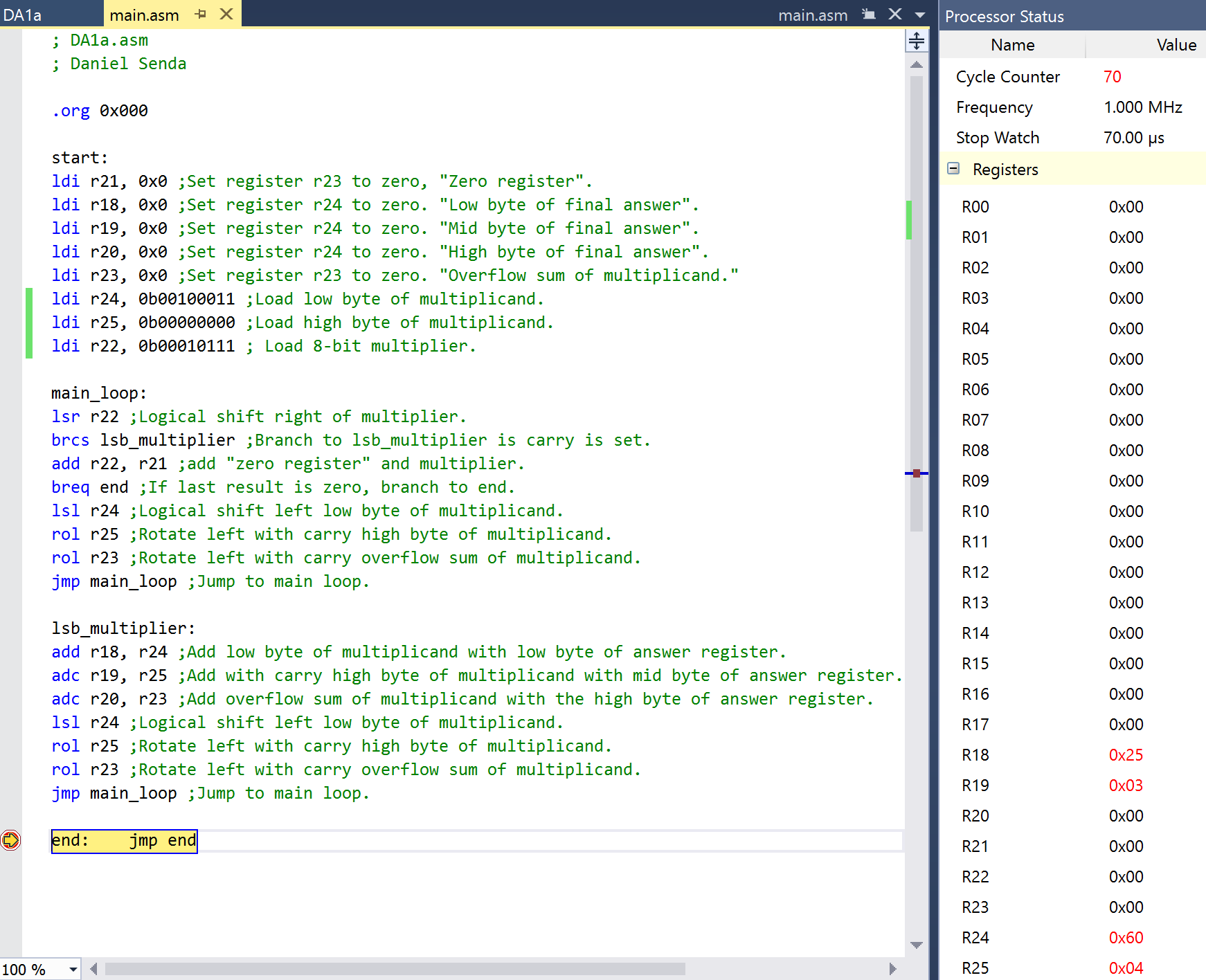
rol r23 ;Rotate left with carry overflow sum of multiplicand.

jmp main\_loop ;Jump to main loop.

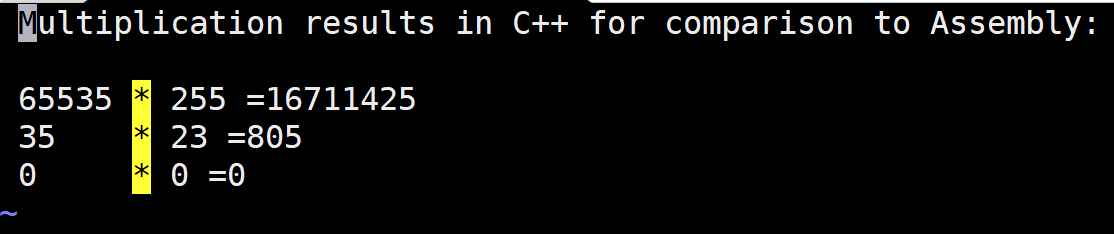
end: jmp end ;End loop, signifying that it finished.

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.  


This concludes the documentation of the digital assignment 1a.