CPE301 – SPRING 2020

Design Assignment 1A

Student Name: Mateo Markovic

Student #: 2001338139

Student Email: Markom1@unlv.nevada.edu

Primary Github address: <https://github.com/mateom99/submission_da>

Directory: DesignAssignments/DA1A

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

.include <m328pdef.inc>

.org 0x00

start:

CLR R0 ; Make sure R0 is zero for later use

LDI R19, 0xAB ; Load the two MSB of the multiplicand into R19

LDI R18, 0x75 ; Load the next two bytes of the multiplicand into R18

LDI R17, 0xF1 ; Load the next two bytes of the multiplicand into R17

LDI R16, 0xBB ; Load the two LSB of the multiplicand into R16

LDI R23, 0x01 ; Load the two MSB of the multiplier into R23

LDI R22, 0x2F ; Load the next two bytes of the multiplier into R22

LDI R21, 0xFE ; Load the next two bytes multiplier into R21

LDI R20, 0x23 ; Load the two LSB of the multiplier into R20

CLR R31 ; Initialize the result registers to zero

CLR R30 ; Initialize the result registers to zero

CLR R29 ; Initialize the result registers to zero

CLR R28 ; Initialize the result registers to zero

CLR R27 ; Initialize the result registers to zero

CLR R26 ; Initialize the result registers to zero

CLR R25 ; Initialize the result registers to zero

CLR R24 ; Initialize the result registers to zero

multiply:

ADD R24, R16 ; Add the first two LSB of the multiplicand with the first two LSB of the result

ADC R25, R17 ; Add the next two bytes of the multiplicand with the first next two bytes of the result with above carry

ADC R26, R18 ; Add the next two bytes of the multiplicand with the first next two bytes of the result with above carry

ADC R27, R19 ; Add the next two bytes of the multiplicand with the first next two bytes of the result with above carry

ADC R28, R0 ; Store any leftover carry

ADC R29, R0 ; Store any leftover carry

ADC R30, R0 ; Store any leftover carry

ADC R31, R0 ; Store any leftover carry

SUBI R20, 1 ; Decrease the multiplier by 1

SBC R21, R0 ; If R20 underflew, decrease the next bit

SBC R22, R0 ; If R21 underflew, decrease the next bit

SBC R23, R0 ; If R22 underflew, decrease the next bit

CPI R20, 0 ; Check if the first byte is zero

BRNE multiply ; if not we need to keep multiplying

CPI R21, 0 ; check the next byte if it is zero

BRNE multiply ; if not we need to keep multiplying

CPI R22, 0 ; check the next byte if it is zero

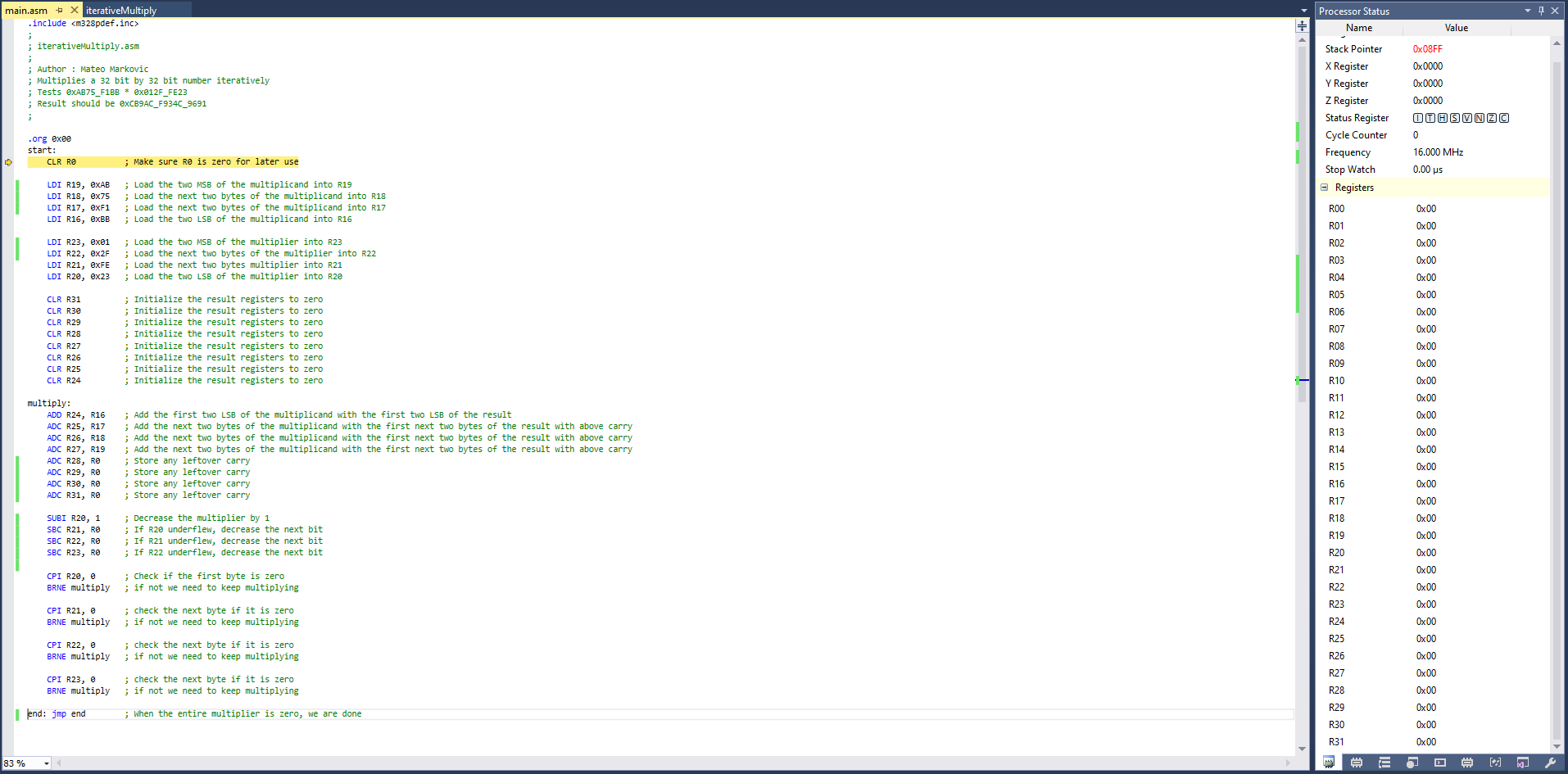
BRNE multiply ; if not we need to keep multiplying

CPI R23, 0 ; check the next byte if it is zero

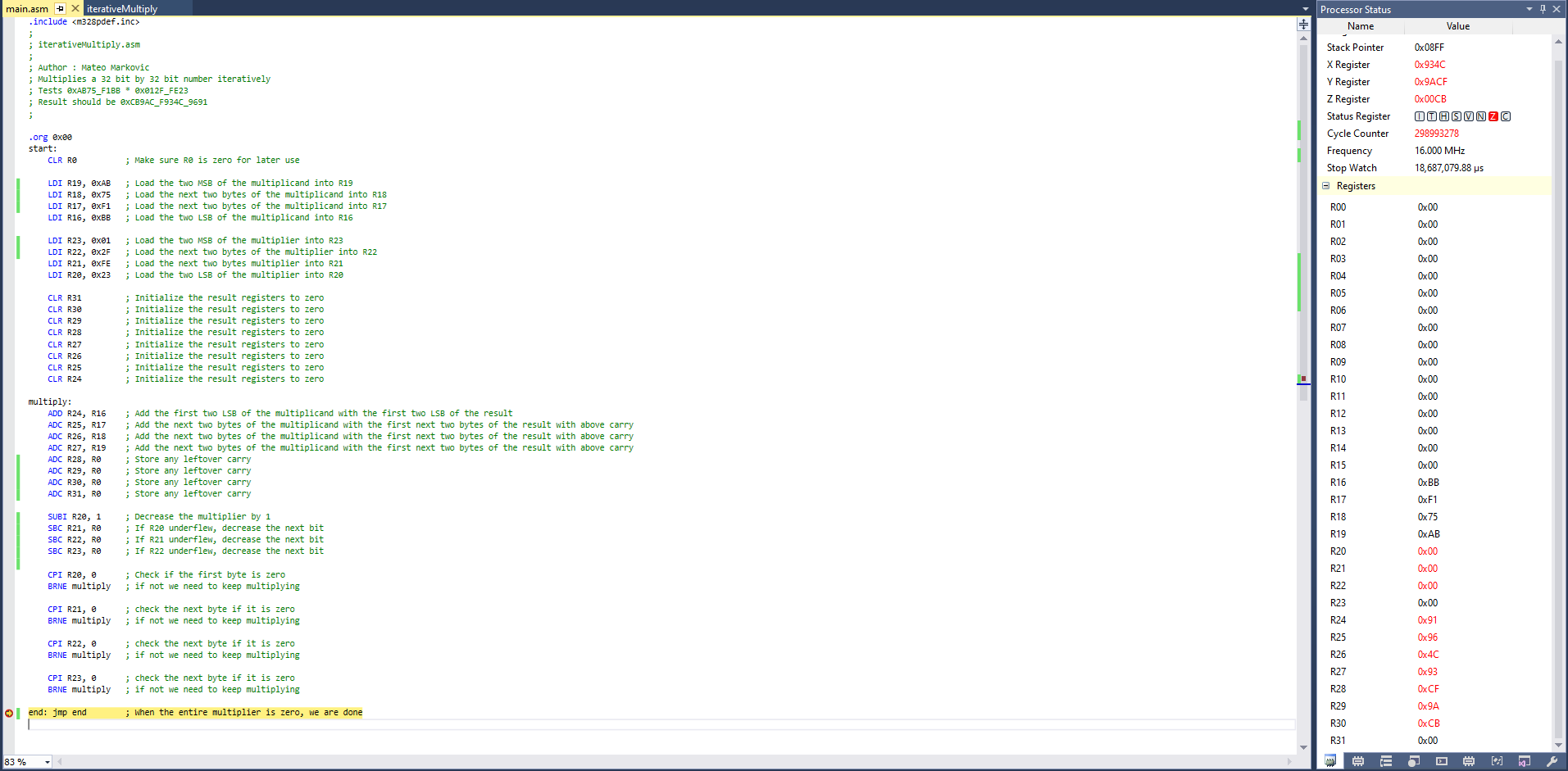
BRNE multiply ; if not we need to keep multiplying

end: jmp end ; When the entire multiplier is zero, we are done

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



Initial State before Execution



Result after Execution (Registers on right)

****

High Level Proof (Windows Programmer Calculator)

1. **EXECUTION TIME**

In the multiplication of 0xAB75F1BB \* 0x012FFE23 the program goes through 298993278 Cycles. When running at 16Mhz this takes 18.687 seconds to run (2.99x10^8/16x10^6).

1. **GITHUB LINK OF THIS DA**

<https://github.com/mateom99/submission_da/tree/master/DesignAssignments/DA1A>

**Student Academic Misconduct Policy**

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

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

Mateo Markovic