**CPE301 – SPRING 2019**

Design Assignment DA1A

Student Name: YeEun Lee

Student #:2001614222

Student Email: leey81@unlv.nevada.edu

Primary Github address: https://github.com/yeeun219/submission\_da.git

Directory: cpe301\DesignAssignments\DA1A

# Task1. Multiply using iterative addition.

;

; AssemblerApplication5.asm

;

; Created: 2019-02-17 3:02:44 am

; Author : llje2

;

.def Res1 = R18

.def Res2 = R19 ;storage of result

.def Res3 = R20

.def m1L = R24 ;16-bit multiplicand lower 8 bits

.def m1M = R25 ;16-bit multiplicand upper 8 bits

.def m2 = R22 ;8-bit multiplier

.equ m1 = 0x0103

initiate:

ldi m1M,HIGH(m1) ; upper 8 bits of m1 to m1M

ldi m1L,LOW(m1) ; lower 8 bits of m1 to m1L

ldi m2,0x03

multiply:

ldi r17,8 ; count of loop, multiplier is 8bit,so count of loop is 8

ldi r16, 1;the number compared with multiplier

ldi r26, 0 ;the number of the shift in loop\_if

loop: AND m2,r16 ;check whether the bit is 1 in order

sub m2,r16

BREQ if\_label ;if the bit is 1 , go if\_label

jmp next ; if the bit is 0 , go next

if\_label:

mov r1, r26 ;not to change value of r26

sub r1,r0

BREQ if\_not ;if r1 is 0, go if\_not then add

;multiplicand to result not shift

loop\_if: ;a shift of r26(the number of the shift)

LSL m1L

LSL m1M

dec r26

BRNE loop\_if

if\_not: ADD res1,m1L ;add left shifted multiplicand to result

ADC Res2,m1M

next:

LSL r16 ; 1 left shift of the comparing number

inc r26 ;increase count of shift

ldi m1M,HIGH(m1) ; reinitiate because of changing value

ldi m1L,LOW(m1)

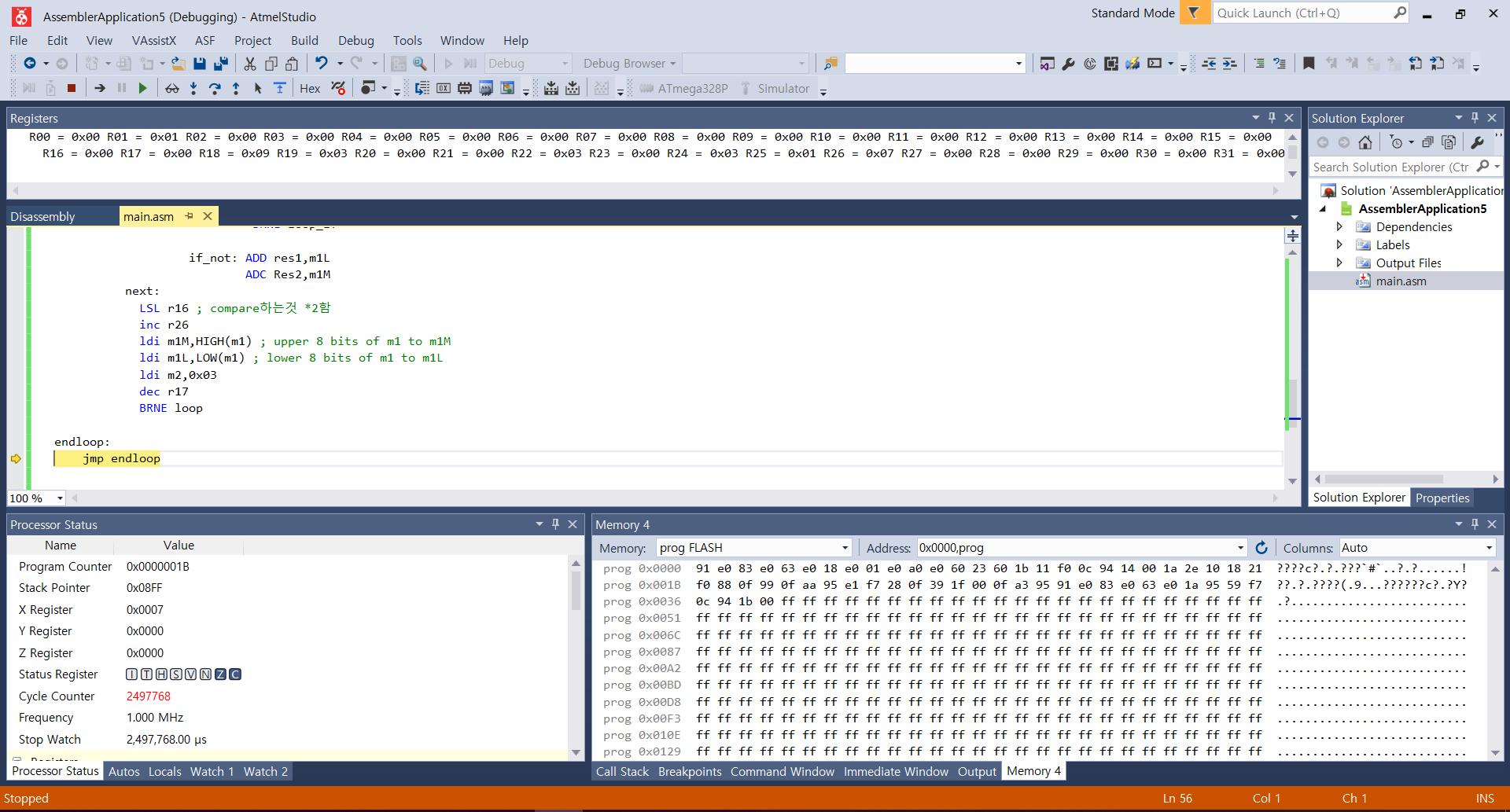
ldi m2,0x03

dec r17 ; decrease count of loop

BRNE loop

endloop:

jmp endloop



The snapshot you can see is result of Task1 (The result in R20:R19:R8 is 0X 000309 ).

# Task3 Multiply using AVR mul instruction.

Code : The change part of task1 is using mul instruction.

.def Res1 = R18

.def Res2 = R19

.def Res3 = R20

.def m1L = R24 ;16-bit multiplicand lower 8 bits

.def m1M = R25 ;16-bit multiplicand upper 8 bits

.def m2 = R22 ;8-bit multiplier

.equ m1 = 0xAAAA

ldi m1M,HIGH(m1) ; upper 8 bits of m1 to m1M

ldi m1L,LOW(m1) ; lower 8 bits of m1 to m1L

ldi m2,0x55 ; 8-bit constant to m2

; Multiply using AVR mul instruction

;

mul m1L,m2 ; Multiply LSB

mov Res1,R0 ; copy result to result register

mov Res2,R1

mul m1M,m2 ; Multiply MSB

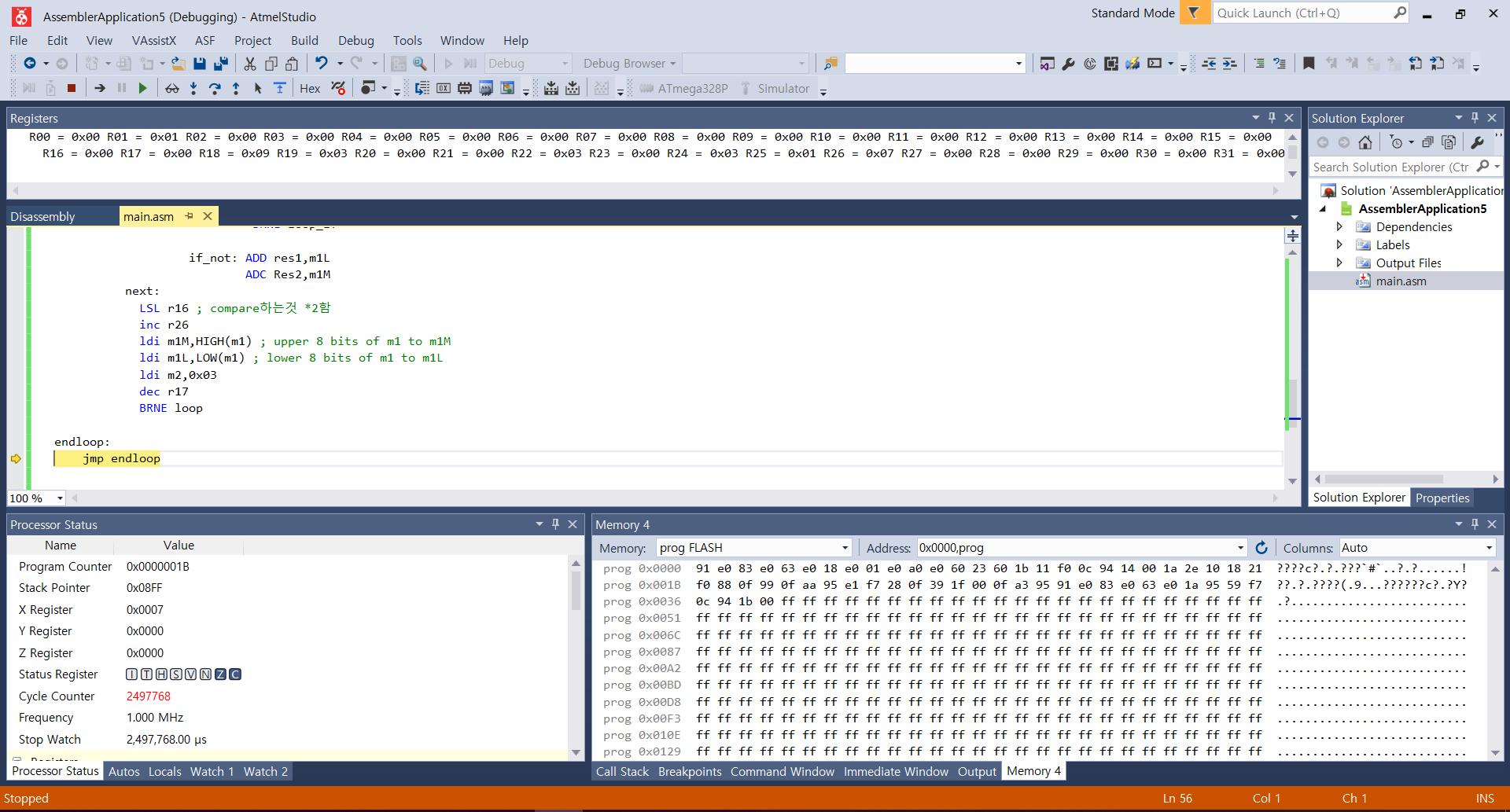
mov Res3,R1 ; copy MSB result to result byte 3

add Res2,R0 ; add LSB result to result byte 2

brcc NoInc ; if not carry, jump

inc Res1

NoInc:



The snapshot you can see is same result of Task1 (The result in R20:R19:R8 is 0X 000309 ).