CPE301 – SPRING 2019

Design Assignment 1A

Student Name: Rocky Yasuaki Gonzalez

Student #: 5003229733

Student Email: gonzar14@unlv.nevada.edu

Primary Github address: https://github.com/rockyg1995/ihswppdar.git

Directory:

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**

N/A, (Atmel Studio 7 Project Only)

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

;

; DA1A - Rocky Gonzalez.asm

;

; Created: 2/5/2019 6:54:39 PM

; Author: rocky

; Summary: The following program executes the function for a Multiplication

; by utilizing a 16-bit Multiplicand and 8-bit Multiplicand, then storing the

; product into 3 registers (24-bit Product)

.include <m328pdef.inc> ; Include library for .SET and .ORG directives

.SET MULT1l = 0xff ; Set value for Lower 16-bit Multiplicand

.SET MULT1u = 0xff ; Set value for Upper 16-bit Multiplicand

.SET MULT2 = 0xff ; Set value for 8-bit Multiplicand

.ORG 0 ; Start Data Collecting from the Origin '0x00'

; Register Assignments

ldi r24, MULT1l ; Load in value for Lower 16-bit 1st Multiplicand

ldi r25, MULT1u ; Load in value for Upper 16-bit 1st Multiplicand

ldi r22, MULT2 ; Load in value for 8-bit 2nd Multiplicand

push r22 ; Save 8-bit Multiplicand Value into Stack

push r24 ; Save 16-bit Lower Multiplicand Value into Stack

push r25 ; Save 16-bit Upper Multiplicand Value into Stack

ldi r17, 0x01 ; Load in value representing increment (For Carry SREG)

clr r18 ; Clear bits in Lower 24-bit Product

clr r19 ; Clear bits in Middle 24-bit Product

clr r20 ; Clear bits in Upper 24-bit Product

; --------------------------------------------------------------------------------

; Check to see if Iterative Addition occurs

chckif:

subi r24, 0 ; Check if 'R24 > 0'

breq else ; Go to 'else' if 'R24 == 0'

rjmp repeat ; Execute Iterative Addition Loop

else: subi r25, 0 ; Check if 'R25 > 0'

breq end ; End if '1st Multiplicand' iteration is complete (R25 = 0, R24 = 0)

dec r25 ; Otherwise Decrement 'R25'

rjmp repeat ; Execute Iterative Addition Loop

; --------------------------------------------------------------------------------

; Iterative Addition Loop

repeat:

add r18, r22 ; Iterate Adding '2nd Multiplicand' by '1st Multiplicand' times

brcs prod1 ; If Overflow in R18, increment value into R19

cont: dec r24 ; Decrement R24

brne repeat ; If 'R24 > 0', repeat Iterative Addition

rjmp chckif ; Otherwise, Check if '1st Multiplicand' iteration is complete

; --------------------------------------------------------------------------------

; Incrementing Middle 24-bit Product

prod1: clc ; Clear Carry in Status Register

add r19, r17 ; Increment Middle 24-bit Product (R19)

brcs prod2 ; If Overflow is set, Increment Upper 24-bit Product

rjmp cont ; Continue Original Loop

; Incrementing Upper 24-bit Product

prod2: clc ; Clear Carry in Status Register

add r20, r17 ; Increment Upper 24-bit Product (R20)

rjmp cont ; Continue Original Loop

; --------------------------------------------------------------------------------

end: pop r25 ; Restore 16-bit Upper Multiplicand Value into Stack

pop r24 ; Restore 16-bit Lower Multiplicand Value into Stack

pop r25 ; Restore 8-bit Multiplicand Value from Stack / Stack Empty

endf: rjmp endf ; Loop End of Program

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

;

; DA1A - Verification - Rocky Gonzalez.asm

;

; Created: 2/10/2019 5:11:38 PM

; Author: rocky

;

; Summary: The following program executes the function for a Multiplication

; by utilizing a 16-bit Multiplicand and 8-bit Multiplicand, then storing the

; product into 3 registers (24-bit Product)

.include <m328pdef.inc> ; Include library for .SET and .ORG directives

.SET MULT1l = 0xff ; Set value for Lower 16-bit Multiplicand

.SET MULT1u = 0xff ; Set value for Upper 16-bit Multiplicand

.SET MULT2 = 0xff ; Set value for 8-bit Multiplicand

.ORG 0 ; Start Data Collecting from the Origin '0x00'

; Register Assignments

ldi r24, MULT1l ; Load in value for Lower 16-bit 1st Multiplicand

ldi r25, MULT1u ; Load in value for Upper 16-bit 1st Multiplicand

ldi r22, MULT2 ; Load in value for 8-bit 2nd Multiplicand

clr r18 ; Clear bits in Lower 24-bit Product

clr r19 ; Clear bits in Middle 24-bit Product

clr r20 ; Clear bits in Upper 24-bit Product

; --------------------------------------------------------------------------------

; Utilize Multiplication Instruction/Store Value Into 24-bit Product 'R20:R19:R18'

mul r24, r22 ; Multiply Lower 16-bit Multiplicand w/ 8-bit Multiplicand -> R1:R0

add r18, r0 ; Store the lower data into the Lower 24-bit Product

add r19, r1 ; Store the upper data into the Middle 24-bit Product

mul r25, r22 ; Multiply Upper 16-bit Multiplicand w/ 8-bit Multiplicand -> R1:R0

add r19, r0 ; Add the lower data into the Middle 24-bit Product

brcs addc ; If 'Carry' in SREG set, Branch to add 'Carry' appropriately

add r20, r1 ; Otherwise, Add upper data into Upper 24-bit Product W/out 'Carry'

rjmp end ; Finish Program

addc: adc r20, r1 ; Add upper data into the Upper 24-bit Product With the 'Carry'

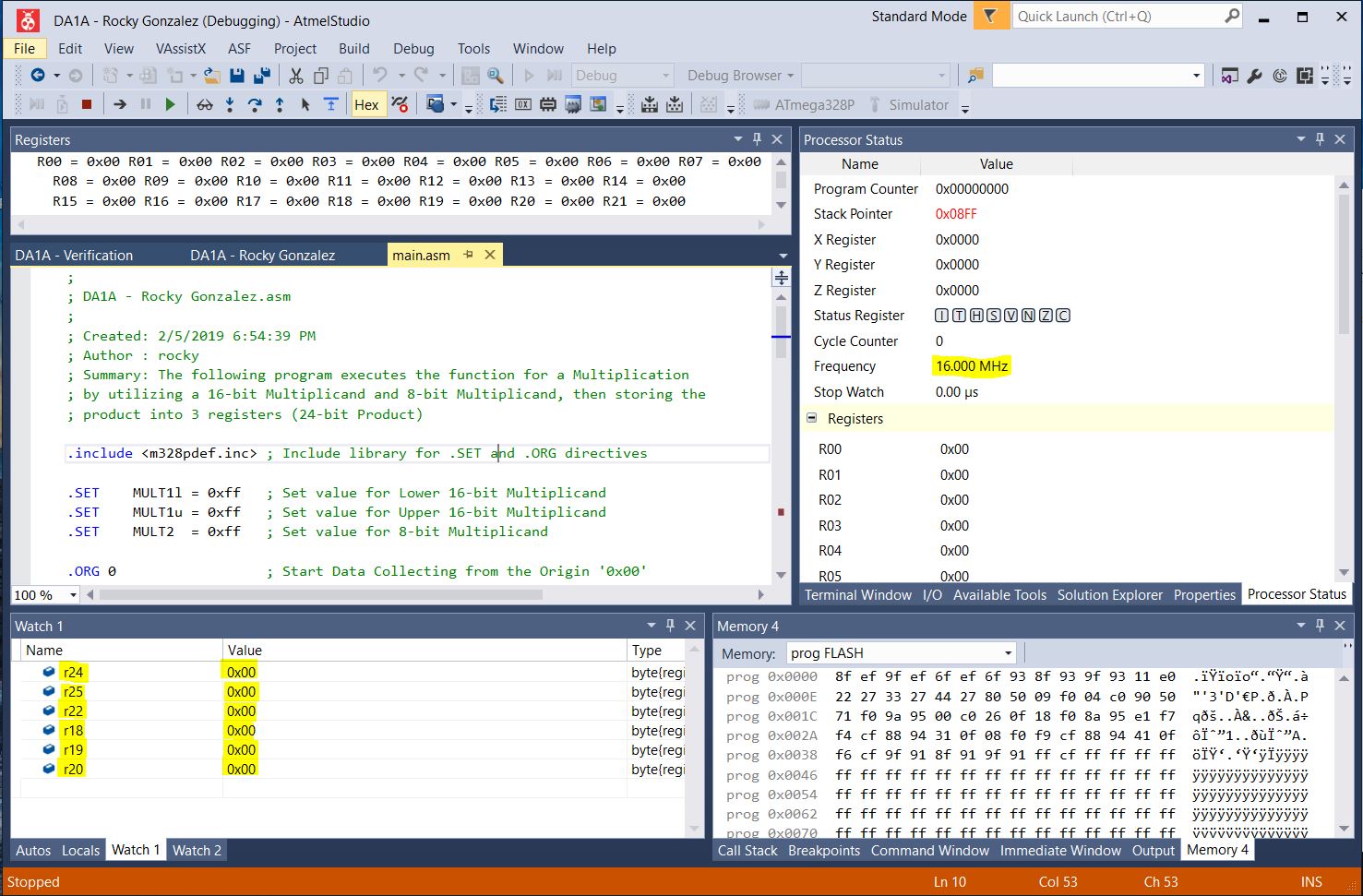
; --------------------------------------------------------------------------------

end: rjmp end ; Loop End of Program

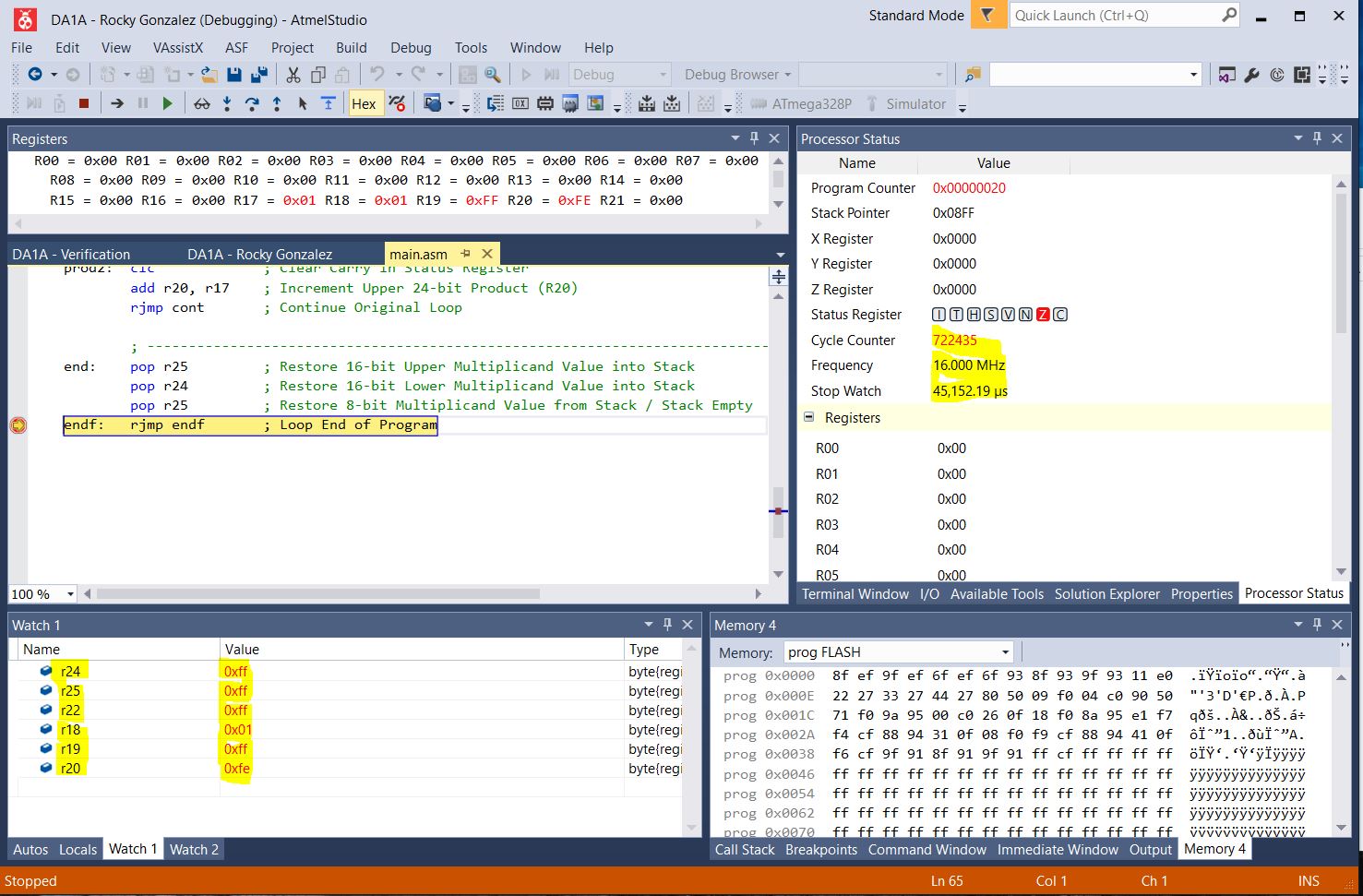
1. **SCHEMATICS**

N/A (Assembly Coding Only)

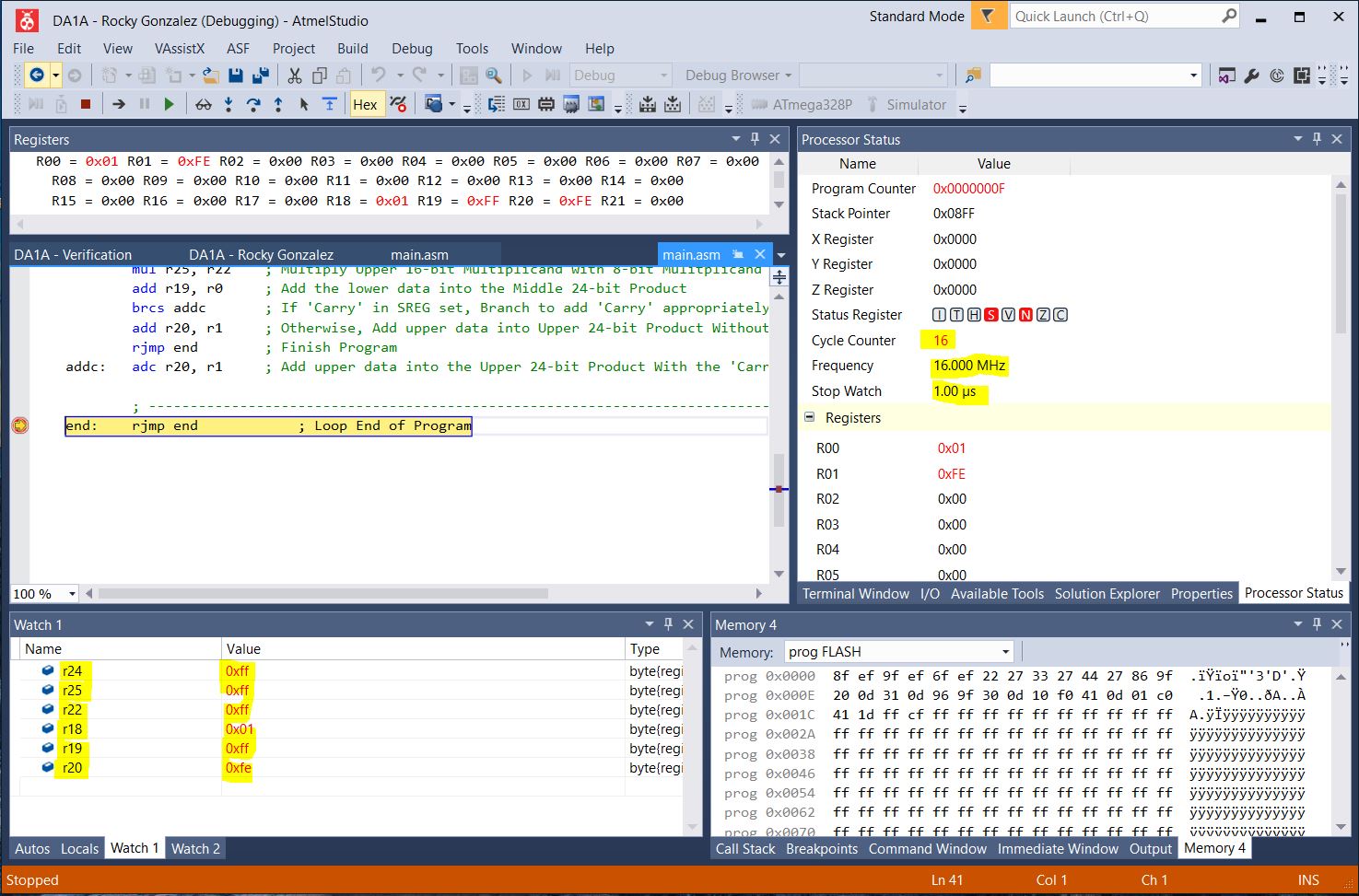
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**



*Figure 1a – Before Start of Iterative Addition (Multiplication)*



*Figure 1b – Output of Iterative Addition (Multiplication)*



*Figure 1c – Output of Multiplication Instruction (Verification)*

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

N/A (Assembly Coding Only)

1. **VIDEO LINKS OF EACH DEMO**

https://youtu.be/oaMX\_D1M-9E

1. **GITHUB LINK OF THIS DA**

**Student Academic Misconduct Policy**

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

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

Rocky Gonzalez