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: C:\Users\rocky\Documents\CpE 301+L - Embedded Systems Design\CpE 301\Repository\DesignAssignments\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

N/A, (Atmel Studio 7 Project Only)

2. 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
```

3. DEVELOPED (VERIFICATION) CODE OF TASK 2/A from TASK 1/A

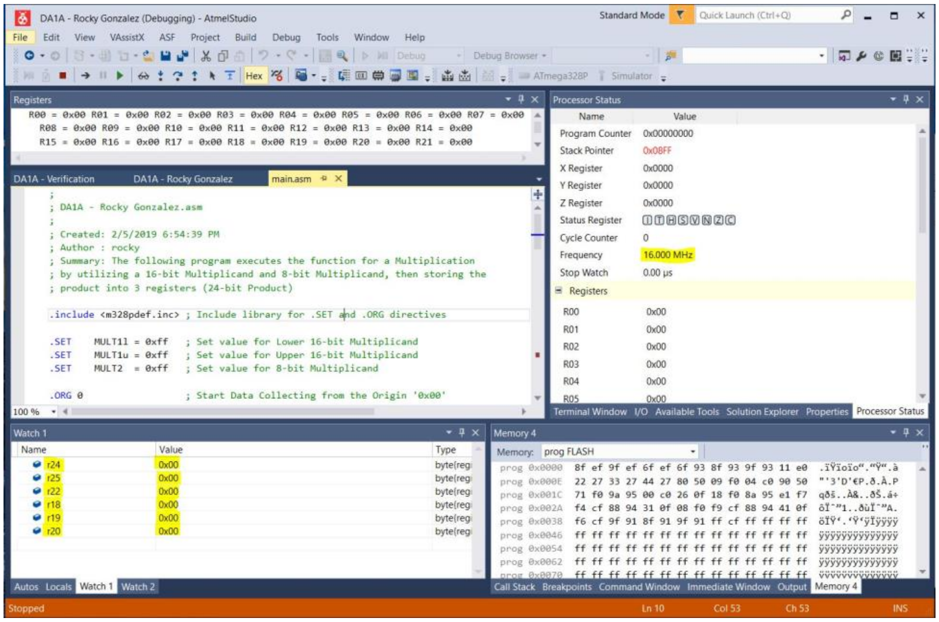
```
;
; DA1A - Verification - Rocky Gonzalez.asm
;
; Created: 2/10/2019 5:11:38 PM
; Author: rocky
;
; Summary: The following program executes the function for a Multiplication
; by utilizing a 16-bit Multiplicand and 8-bit Multiplicand, then storing the
; product into 3 registers (24-bit Product)

.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
```

4. SCHEMATICS

N/A (Assembly Coding Only)

5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)


DA1A - Rocky Gonzalez (Debugging) - AtmelStudio
Standard Mode
Quick Launch (Ctrl+Q)
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Debug Browser
Hex
ATmega328P
Simulator
Registers
DA1A - Verification
DA1A - Rocky Gonzalez
main.asm
100 %
Processor Status
Terminal Window I/O Available Tools Solution Explorer Properties Processor Status
Watch 1
Autos Locals Watch 1 Watch 2
Memory 4
Call Stack Breakpoints Command Window Immediate Window Output Memory 4
Stopped
Ln 10
Col 53
Ch 53
INS

```
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00
 R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00
 R15 = 0x00 R16 = 0x00 R17 = 0x00 R18 = 0x00 R19 = 0x00 R20 = 0x00 R21 = 0x00
```

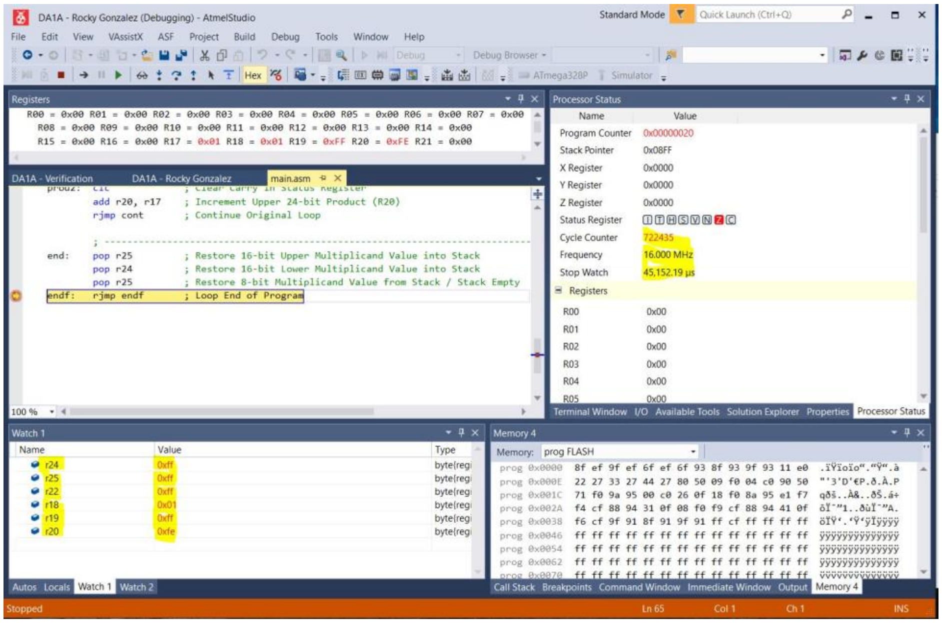
```
;
; DA1A - Rocky Gonzalez.asm
;
; Created: 2/5/2019 6:54:39 PM
; Author : rocky
; Summary: The following program executes the function for a Multiplication
; by utilizing a 16-bit Multiplicand and 8-bit Multiplicand, then storing the
; product into 3 registers (24-bit Product)

.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'
```

Name	Value
Program Counter	0x00000000
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	0
Frequency	16.000 MHz
Stop Watch	0.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00

Name	Value	Type
r24	0x00	byte{reg
r25	0x00	byte{reg
r22	0x00	byte{reg
r18	0x00	byte{reg
r19	0x00	byte{reg
r20	0x00	byte{reg

```
Memory: prog FLASH
prog 0x0000  8f ef 9f ef 6f ef 6f 93 8f 93 9f 93 11 e0  .ïŸïoïo“.“Ÿ“.à
prog 0x000E  22 27 33 27 44 27 80 50 09 f0 04 c0 90 50  "'3'D'€P.ð.À.P
prog 0x001C  71 f0 9a 95 00 c0 26 0f 18 f0 8a 95 e1 f7  qðš..À&..ðŠ.á÷
prog 0x002A  f4 cf 88 94 31 0f 08 f0 f9 cf 88 94 41 0f  ôÏˆ”1..ðùÏˆ”A.
prog 0x0038  f6 cf 9f 91 8f 91 9f 91 ff cf ff ff ff ff  öÏŸ‘.‘Ÿ‘ÿÏÿÿÿÿ
prog 0x0046  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0054  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0062  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0070  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

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


DA1A - Rocky Gonzalez (Debugging) - AtmelStudio
Standard Mode
Quick Launch (Ctrl+Q)
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Debug Browser
Hex
ATmega328P
Simulator
Registers
DA1A - Verification
DA1A - Rocky Gonzalez
main.asm
100 %
Processor Status
Terminal Window I/O Available Tools Solution Explorer Properties Processor Status
Watch 1
Autos Locals Watch 1 Watch 2
Memory 4
Call Stack Breakpoints Command Window Immediate Window Output Memory 4
Stopped
Ln 65
Col 1
Ch 1
INS

```
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00
 R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00
 R15 = 0x00 R16 = 0x00 R17 = 0x01 R18 = 0x01 R19 = 0xFF R20 = 0xFE R21 = 0x00
```

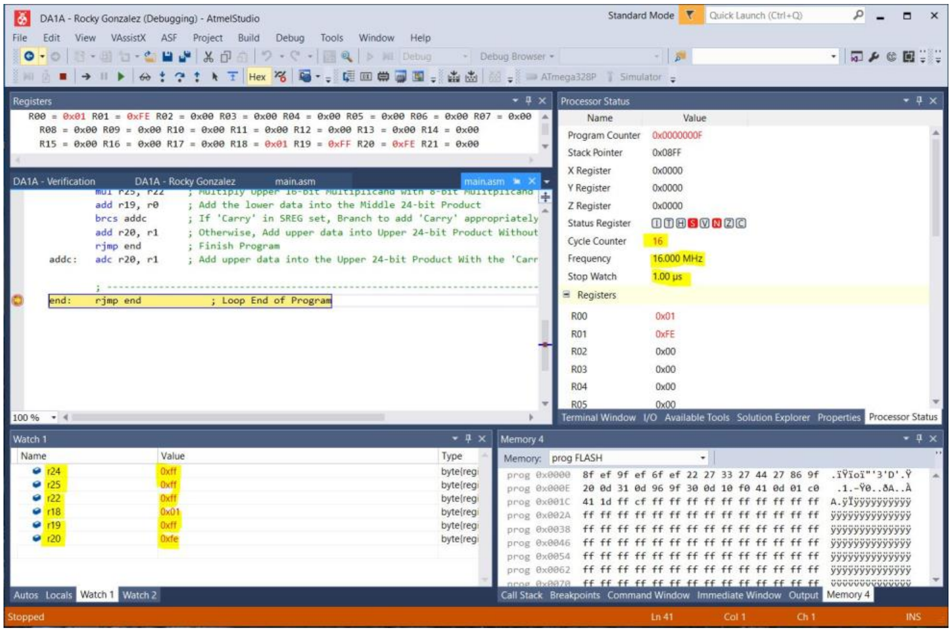
```
prod2:  clc            ; Clear Carry in Status Register
        add r20, r17   ; Increment Upper 24-bit Product (R20)
        rjmp cont      ; Continue Original Loop

        ; ------------------------------------------------------------------------------
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
```

Name	Value
Program Counter	0x00000020
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	722435
Frequency	16.000 MHz
Stop Watch	45,152.19 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00

Name	Value	Type
r24	0xff	byte{reg
r25	0xff	byte{reg
r22	0xff	byte{reg
r18	0x01	byte{reg
r19	0xff	byte{reg
r20	0xfe	byte{reg

```
Memory: prog FLASH
prog 0x0000  8f ef 9f ef 6f ef 6f 93 8f 93 9f 93 11 e0  .ïŸïoïo“.“Ÿ“.à
prog 0x000E  22 27 33 27 44 27 80 50 09 f0 04 c0 90 50  "'3'D'€P.ð.À.P
prog 0x001C  71 f0 9a 95 00 c0 26 0f 18 f0 8a 95 e1 f7  qðš..À&..ðŠ.á÷
prog 0x002A  f4 cf 88 94 31 0f 08 f0 f9 cf 88 94 41 0f  ôÏˆ”1..ðùÏˆ”A.
prog 0x0038  f6 cf 9f 91 8f 91 9f 91 ff cf ff ff ff ff  öÏŸ‘.‘Ÿ‘ÿÏÿÿÿÿ
prog 0x0046  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0054  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0062  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0070  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Figure 1b – Output of Iterative Addition (Multiplication)


DA1A - Rocky Gonzalez (Debugging) - AtmelStudio
Standard Mode
Quick Launch (Ctrl+Q)
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Debug Browser
Hex
ATmega328P
Simulator
Registers
DA1A - Verification
DA1A - Rocky Gonzalez
main.asm
main.asm
100 %
Processor Status
Terminal Window I/O Available Tools Solution Explorer Properties Processor Status
Watch 1
Autos Locals Watch 1 Watch 2
Memory 4
Call Stack Breakpoints Command Window Immediate Window Output Memory 4
Stopped
Ln 41
Col 1
Ch 1
INS

```
R00 = 0x01 R01 = 0xFE R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00
  R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00
  R15 = 0x00 R16 = 0x00 R17 = 0x00 R18 = 0x01 R19 = 0xFF R20 = 0xFE R21 = 0x00
```

```
        mul r25, r22     ; Multiply Upper 16-bit Multiplicand with 8-bit Mulitplicand
        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 Without
        rjmp end         ; Finish Program
addc:   adc r20, r1      ; Add upper data into the Upper 24-bit Product With the 'Carr

        ; ------------------------------------------------------------------------
end:    rjmp end             ; Loop End of Program
```

Name	Value
Program Counter	0x0000000F
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	16
Frequency	16.000 MHz
Stop Watch	1.00 µs
⊟ Registers	
R00	0x01
R01	0xFE
R02	0x00
R03	0x00
R04	0x00
R05	0x00

Name	Value	Type
r24	0xff	byte{regi
r25	0xff	byte{regi
r22	0xff	byte{regi
r18	0x01	byte{regi
r19	0xff	byte{regi
r20	0xfe	byte{regi

Memory: prog FLASH

```
prog 0x0000  8f ef 9f ef 6f ef 22 27 33 27 44 27 86 9f  .ïŸïoï"'3'D'.Ÿ
prog 0x000E  20 0d 31 0d 96 9f 30 0d 10 f0 41 0d 01 c0   .1.–Ÿ0..ðA..À
prog 0x001C  41 1d ff cf ff ff ff ff ff ff ff ff ff ff  A.ÿÏÿÿÿÿÿÿÿÿÿÿ
prog 0x002A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0038  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0046  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0054  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0062  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0070  ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Figure 1c – Output of Multiplication Instruction (Verification)

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

N/A (Assembly Coding Only)

7. VIDEO LINKS OF EACH DEMO

https://youtu.be/oaMX_D1M-9E

8. GITHUB LINK OF THIS DA

C:\Users\rocky\Documents\CpE 301+L - Embedded Systems Design\CpE 301\Repository\DesignAssignments\DA1A

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"This assignment submission is my own, original work".
Rocky Gonzalez