Jason Sikorski

CPE301 – SPRING 2016

Design Assignment 1

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

The student understands that all required components should be submitted in complete for grading of this assignment.

NO	SUBMISSION ITEM	COMPLETED (Y/N)	MARKS (/MAX)
0.	INITIAL CODE OF TASK A		
1.	INCREMENTAL / DIFFERENTIAL CODE OF TASK B		
2.	INCREMENTAL / DIFFERENTIAL CODE OF TASK C		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK D		
4	CALCULATION OF EXECUTION TIME @ 16MHZ		
5.	SCREENSHOTS OF EACH TASK OUTPUT		
6.	GITHUB LINK OF THE DA		

0	INITIAL CODE OF TASK 1/A		

FILE: header.inc

```
; Created: 2/19/2016 1:39:33 PM
; Author : jmsikorski
;
.ORG  0x20         ;Place after program code

.MACRO INITSTACK      ;Initialize Stack Pointer
   LDI   R16, HIGH(RAMEND)
   OUT   SPH, R16
   LDI   R16, LOW(RAMEND)
   OUT   SPL, R16
.ENDMACRO


.DEF  TEMP1 = R17     ;R17 is a temp register and its value is not saved
.DEF  COUNT = R18     ;R18 is used for loop counting
.EQU  Var_Address = 0x100;

Div3:              ;Checks if a value is divisible by 3
                   ;Value is stored in TEMP1 register
                   ;Algorithm subtracts odd bits and adds even bits
                   ;If the sum is 0, 3, or -3 then TEMP1 is divisible by 3

   PUSH  R16       ;Save R16 state
   PUSH  R25       ;Save R25 state
   PUSH  R22       ;Save R22 state
   PUSH  COUNT     ;Save Count state
   LDI   R25, 0    ;Clear summing register
   LDI   R22, 0    ;Store 0 value
   LDI   COUNT, 4  ;Loop Counter
   MOV   R16, TEMP1   ;Save TEMP1 value
Repeat3:
   LSL   TEMP1     ;Shift out MSB of TEMP1
   SBC   R25, R22  ;Subtract carry from count
   LSL   TEMP1     ;Shift out MSB of TEMP1
   ADC   R25, R22  ;Add carry to count
   DEC   COUNT     ;Decrement loop counter
   CPI   COUNT, 0  ;Check loop counter
   BRNE  Repeat3      ;Stay in loop
   CPI   R25, 0
   BREQ  Add3      ;If count != 0 don't add
   CPI   R25, 3
   BREQ  Add3      ;If count != 3 don't add
   CPI   R25, -3
   BREQ  Add3      ;If count != -3 don't add
   RJMP  End3
Add3:
   ADD   R23, R16  ;Add the value to R24
   ADC   R24, R22  ;Add the carry if any to R25
End3:
   POP   COUNT     ;Restore Count
   POP   R22       ;Restore R22
```

```
   POP   R25       ;Restore R25
   POP   R16       ;Restore R16
   RET             ;Return to program

Div7:              ;Checks if a value is divisible by 7
                   ;Value is stored in TEMP1 register
                   ;Algorithm divides 8 bits into 3 bit segments,
                   ;segments are summed, if sum = 7 then TEMP1 is
                   ;divisible by 7, if sum > 7, repeat summation
   PUSH  R16       ;Save R16 state
   PUSH  R25       ;Save R25 state
   PUSH  R22       ;Save R22 state
   PUSH  COUNT     ;Save Count state
   LDI   R25, 0    ;Clear summing register1
   LDI   R22, 0    ;Store 0 value
   LDI   COUNT, 3  ;Loop counter
   MOV   R16, TEMP1   ;Save TEMP1 value
Repeat7:
   SBRC  TEMP1, 0  ;Add 1 if bit 0 is 1
   SUBI  R25, -1
   SBRC  TEMP1, 1  ;Add 2 if bit 1 is 1
   SUBI  R25, -2
   SBRC  TEMP1, 2  ;Add 4 if bit 1 is 1
   SUBI  R25, -4
   LSR   TEMP1     ;Shift out 1st 3 bits
   LSR   TEMP1
   LSR   TEMP1
   DEC   COUNT     ;Decrement Counter
   CPI   COUNT, 0  ;Loop until Counter equals 0
   BRNE  Repeat7
   SUBI  R25, 7    ;Subtract 7 from sum register
   BRMI  End7      ;If Sum less than 0, not divisible by 7
   BREQ  Add7      ;If Sum equal 7, divisible by 7
   LDI   COUNT, 3  ;If greater than 0, sum again
   RJMP  Repeat7

Add7:
   ADD   R20, R16  ;Add the value to R20
   ADC   R21, R22  ;Add the carry if any to R21
End7:
   POP   COUNT     ;Restore Count
   POP   R22       ;Restore R22
   POP   R25       ;Restore R25
   POP   R16       ;Restore R16
   RET             ;Return to program
```

FILE: main.asm

```
.INCLUDE "header.inc"          ;Header file contains macros & subroutines
.ORG 0x00                      ;Begining of code

   INITSTACK                   ;Initialize the stack pointer

   LDI   ZL, LOW(Var_Address)  ;Load Z with variable address
   LDI   ZH, HIGH(Var_Address)
   LDI   TEMP1, LOW(RAMEND/2)  ;Load Temp1 with initial value
   LDI   COUNT, 25             ;Number of values to add
Get25:
```

```
   ST    Z+, TEMP1             ;Store Temp1 into (Z) then increment Z
   SUBI  TEMP1, -4             ;Update Temp1 value
   DEC   COUNT                 ;Decrement Counter
   CPI   COUNT, 0              ;Loop until Counter equals 0
   BRNE  Get25

End1:
   RJMP  End1                  ;Infinite loop
```

1.	INCREMENTAL / DIFFERENTIAL CODE OF TASK B		

File main.asm

```
.EQU  Num = 25                 ;Variable for number of values (Added)
   LDI   COUNT, Num            ;Number of values to add    (Changed line 9)

GetNum:  (Changed line 10)

   BRNE  GetNum   (Changed line15)

(Incremental Code at line 16)

   LDI   ZL, LOW(Var_Address)  ;Load Z with array index[0]
   LDI   ZH, HIGH(Var_Address)
   LDI   COUNT, Num            ;Variable Counter
Divide7:
   LD    TEMP1, Z+             ;Load TEMP1 with the value in (Z) post increment Z
   CALL  Div7                  ;Call Div7 Subroutine
   DEC   COUNT                 ;Decrement counter
   CPI   COUNT, 0              ;Repeat until Counter equals 0
   BRNE  Divide7

End1:
   RJMP  End1                  ;Infinte Loop
```

2	INCREMENTAL / DIFFERENTIAL CODE OF TASK C		

File main.asm

```
(Differential Code at line 22)
Divide:
   LD    TEMP1, Z              ;Load TEMP1 with the value in (Z), post increment Z
   CALL  Div7                  ;Call Div7 Subroutine
   LD    TEMP1, Z+             ;Reload TEMP1
   CALL  Div3                  ;Call Div3 Subroutine
   DEC   COUNT                 ;Decrement Counter
   CPI   COUNT, 0              ;Loop until Counter equals 0
   BRNE  Divide

End1:
   RJMP  End1                  ;Infinite Loop
```

3	INCREMENTAL / DIFFERENTIAL CODE OF TASK D		

File main.asm

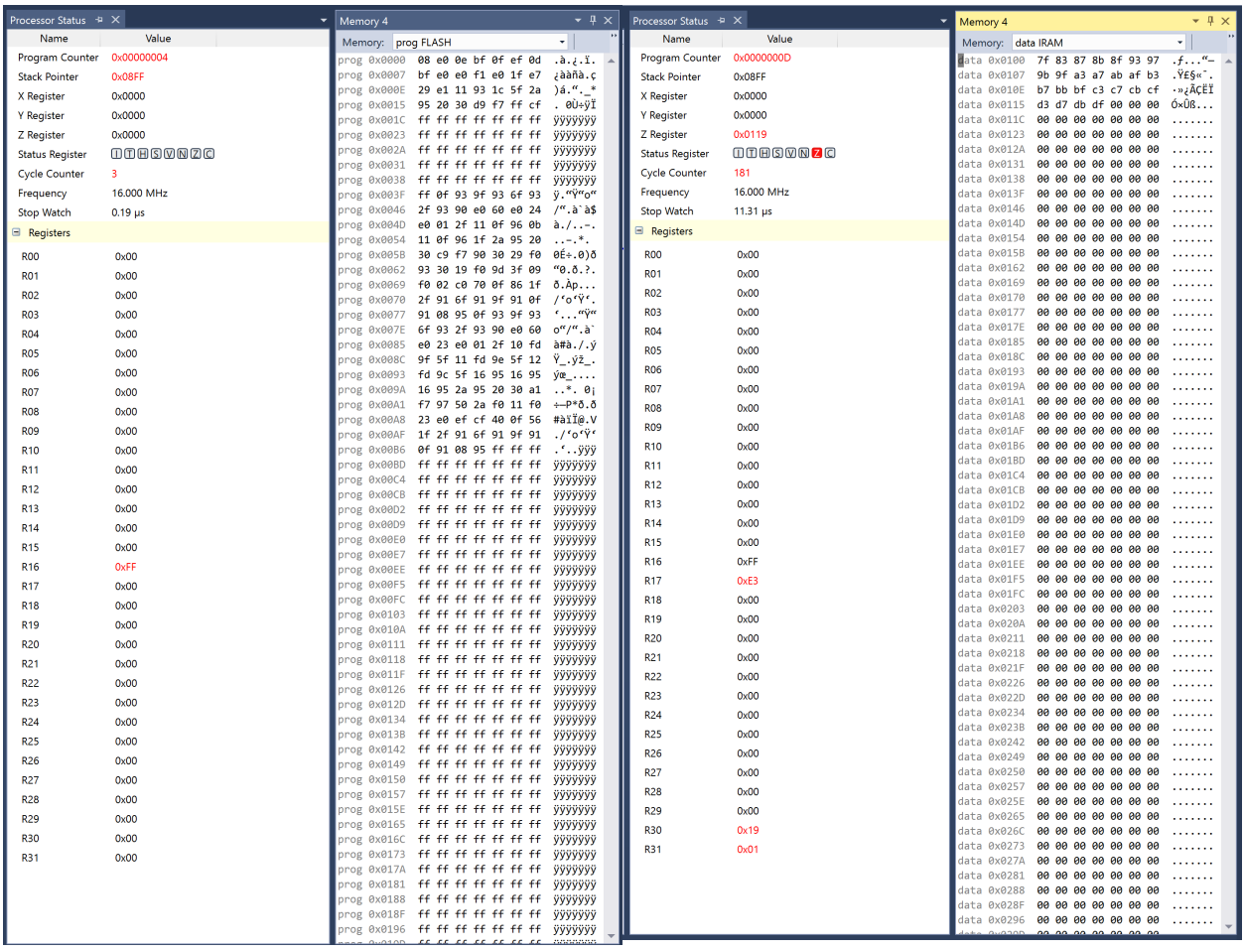
```
(Incremental Code at line 31)
   LDI    TEMP1, 0
   AND    R7, TEMP1           ;Clear R7
   CPI    R24, 0              ;Check if R24 has a value (i.e. R23 has overflowed)
   BREQ   End1                ;If R23 hasn't overflown, jump to end
   LDI    TEMP1, 8            ;Set bit 3 high
   OR     R7, TEMP1

End1:
   RJMP   End1                ;Infinite Loop
```

4.	CALCULATION OF EXECUTION TIME @ 16MHZ		

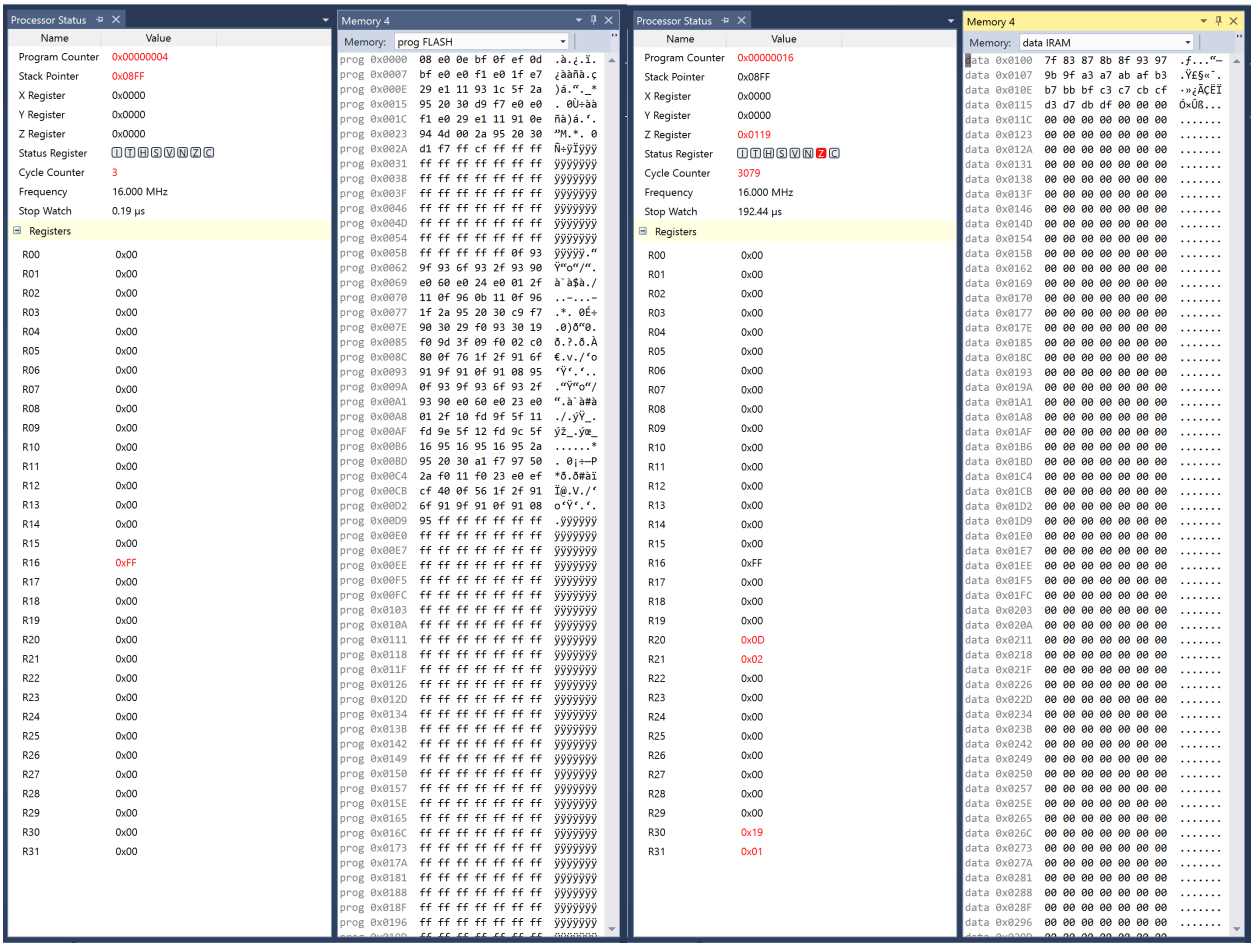
Simulation shows an execution time of 299.38 μs at 16 Mhz using 4790 clock cycles.

5.	SCREENSHOTS OF EACH TASK OUTPUT		

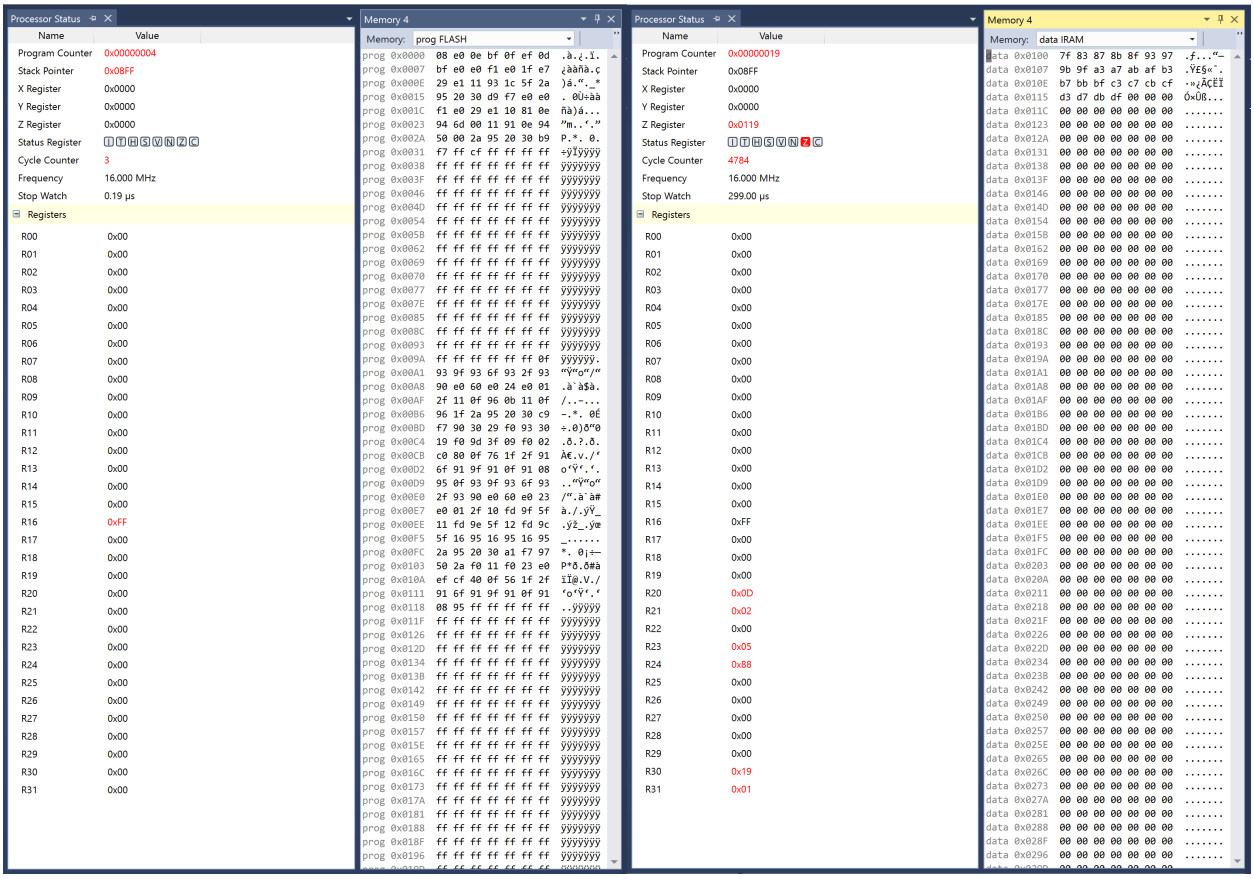
TASK A:



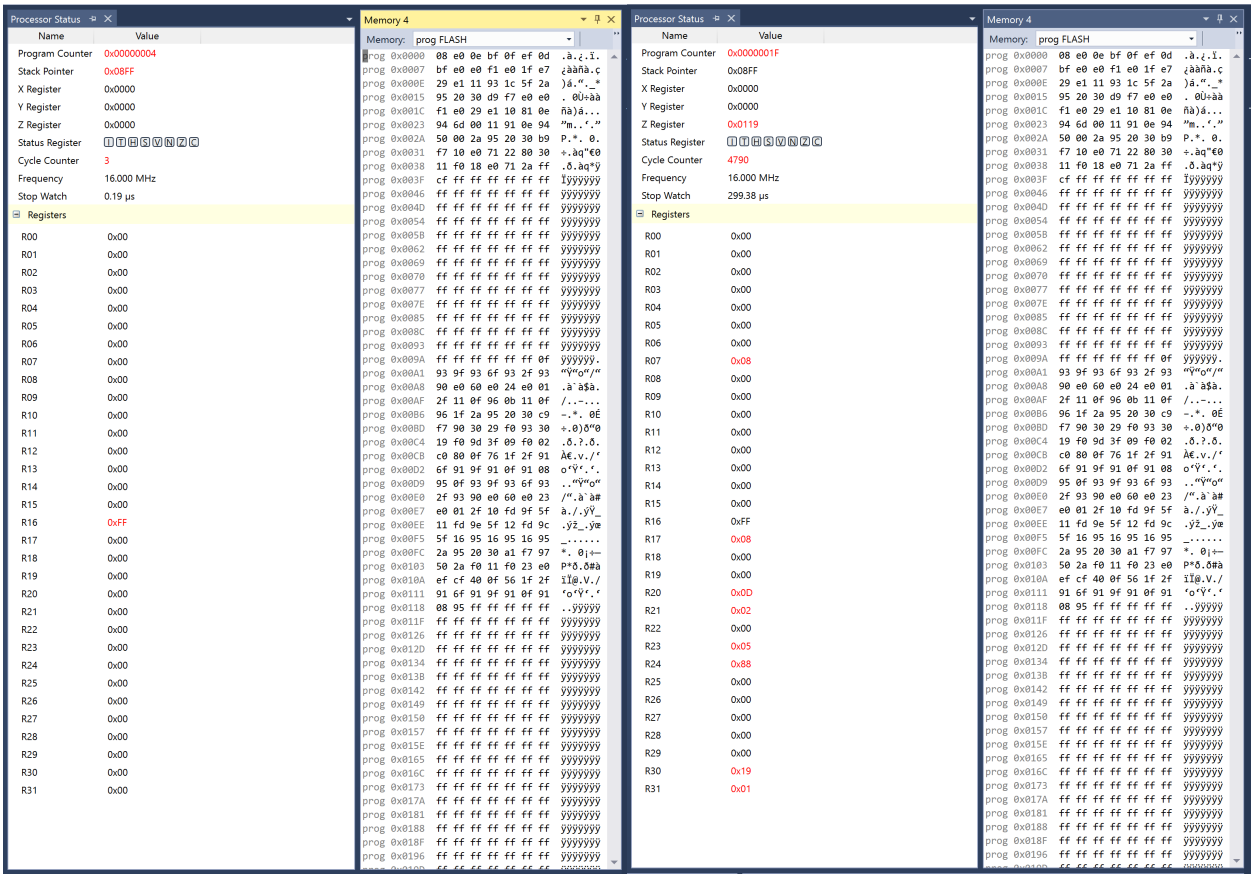
TASK B:


Processor Status
Name
Value
Program Counter
0x00000004
Stack Pointer
0x08FF
X Register
0x0000
Y Register
0x0000
Z Register
0x0000
Status Register
I T H S V N Z C
Cycle Counter
3
Frequency
16.000 MHz
Stop Watch
0.19 µs
Registers
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 0xFF
R17 0x00
R18 0x00
R19 0x00
R20 0x00
R21 0x00
R22 0x00
R23 0x00
R24 0x00
R25 0x00
R26 0x00
R27 0x00
R28 0x00
R29 0x00
R30 0x00
R31 0x00
Memory 4
Memory: prog FLASH
prog 0x0000 08 e0 0e bf 0f ef 0d .à.¿.ï.
prog 0x0007 bf e0 e0 f1 e0 1f e7 ¿ààñà.ç
prog 0x000E 29 e1 11 93 1c 5f 2a)á.“._*
prog 0x0015 95 20 30 d9 f7 e0 e0 . 0Ù÷àà
prog 0x001C f1 e0 29 e1 11 91 0e ñà)á.‘.
prog 0x0023 94 4d 00 2a 95 20 30 ”M.*. 0
prog 0x002A d1 f7 ff cf ff ff ff Ñ÷ÿÏÿÿÿ
prog 0x0031 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0038 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x003F ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0046 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x004D ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0054 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x005B ff ff ff ff ff 0f 93 ÿÿÿÿÿ.“
prog 0x0062 9f 93 6f 93 2f 93 90 Ÿ“o“/“.
prog 0x0069 e0 60 e0 24 e0 01 2f à`à$à./
prog 0x0070 11 0f 96 0b 11 0f 96 ..–...–
prog 0x0077 1f 2a 95 20 30 c9 f7 .*. 0É÷
prog 0x007E 90 30 29 f0 93 30 19 .0)ð“0.
prog 0x0085 f0 9d 3f 09 f0 02 c0 ð.?.ð.À
prog 0x008C 80 0f 76 1f 2f 91 6f €.v./‘o
prog 0x0093 91 9f 91 0f 91 08 95 ‘Ÿ‘.‘..
prog 0x009A 0f 93 9f 93 6f 93 2f .“Ÿ“o“/
prog 0x00A1 93 90 e0 60 e0 23 e0 “.à`à#à
prog 0x00A8 01 2f 10 fd 9f 5f 11 ./.ýŸ_.
prog 0x00AF fd 9e 5f 12 fd 9c 5f ýž_.ýœ_
prog 0x00B6 16 95 16 95 16 95 2a .....*
prog 0x00BD 95 20 30 a1 f7 97 50 . 0¡÷–P
prog 0x00C4 2a f0 11 f0 23 e0 ef *ð.ð#àï
prog 0x00CB cf 40 0f 56 1f 2f 91 Ï@.V./‘
prog 0x00D2 6f 91 9f 91 0f 91 08 o‘Ÿ‘.‘.
prog 0x00D9 95 ff ff ff ff ff ff .ÿÿÿÿÿÿ
prog 0x00E0 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x00E7 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x00EE ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x00F5 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x00FC ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0103 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x010A ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0111 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0118 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x011F ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0126 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x012D ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0134 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x013B ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0142 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0149 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0150 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0157 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x015E ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0165 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x016C ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0173 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x017A ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0181 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0188 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x018F ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
prog 0x0196 ff ff ff ff ff ff ff ÿÿÿÿÿÿÿ
Processor Status
Name
Value
Program Counter
0x00000016
Stack Pointer
0x08FF
X Register
0x0000
Y Register
0x0000
Z Register
0x0119
Status Register
I T H S V N Z C
Cycle Counter
3079
Frequency
16.000 MHz
Stop Watch
192.44 µs
Registers
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 0xFF
R17 0x00
R18 0x00
R19 0x00
R20 0x0D
R21 0x02
R22 0x00
R23 0x00
R24 0x00
R25 0x00
R26 0x00
R27 0x00
R28 0x00
R29 0x00
R30 0x19
R31 0x01
Memory 4
Memory: data IRAM
data 0x0100 7f 83 87 8b 8f 93 97 .ƒ...“–
data 0x0107 9b 9f a3 a7 ab af b3 .Ÿ£§«¯.
data 0x010E b7 bb bf c3 c7 cb cf ·»¿ÃÇËÏ
data 0x0115 d3 d7 db df 00 00 00 Ó×Ûß...
data 0x011C 00 00 00 00 00 00 00 ....
data 0x0123 00 00 00 00 00 00 00 ....
data 0x012A 00 00 00 00 00 00 00 ....
data 0x0131 00 00 00 00 00 00 00 ....
data 0x0138 00 00 00 00 00 00 00 ....
data 0x013F 00 00 00 00 00 00 00 ....
data 0x0146 00 00 00 00 00 00 00 ....
data 0x014D 00 00 00 00 00 00 00 ....
data 0x0154 00 00 00 00 00 00 00 ....
data 0x015B 00 00 00 00 00 00 00 ....
data 0x0162 00 00 00 00 00 00 00 ....
data 0x0169 00 00 00 00 00 00 00 ....
data 0x0170 00 00 00 00 00 00 00 ....
data 0x0177 00 00 00 00 00 00 00 ....
data 0x017E 00 00 00 00 00 00 00 ....
data 0x0185 00 00 00 00 00 00 00 ....
data 0x018C 00 00 00 00 00 00 00 ....
data 0x0193 00 00 00 00 00 00 00 ....
data 0x019A 00 00 00 00 00 00 00 ....
data 0x01A1 00 00 00 00 00 00 00 ....
data 0x01A8 00 00 00 00 00 00 00 ....
data 0x01AF 00 00 00 00 00 00 00 ....
data 0x01B6 00 00 00 00 00 00 00 ....
data 0x01BD 00 00 00 00 00 00 00 ....
data 0x01C4 00 00 00 00 00 00 00 ....
data 0x01CB 00 00 00 00 00 00 00 ....
data 0x01D2 00 00 00 00 00 00 00 ....
data 0x01D9 00 00 00 00 00 00 00 ....
data 0x01E0 00 00 00 00 00 00 00 ....
data 0x01E7 00 00 00 00 00 00 00 ....
data 0x01EE 00 00 00 00 00 00 00 ....
data 0x01F5 00 00 00 00 00 00 00 ....
data 0x01FC 00 00 00 00 00 00 00 ....
data 0x0203 00 00 00 00 00 00 00 ....
data 0x020A 00 00 00 00 00 00 00 ....
data 0x0211 00 00 00 00 00 00 00 ....
data 0x0218 00 00 00 00 00 00 00 ....
data 0x021F 00 00 00 00 00 00 00 ....
data 0x0226 00 00 00 00 00 00 00 ....
data 0x022D 00 00 00 00 00 00 00 ....
data 0x0234 00 00 00 00 00 00 00 ....
data 0x023B 00 00 00 00 00 00 00 ....
data 0x0242 00 00 00 00 00 00 00 ....
data 0x0249 00 00 00 00 00 00 00 ....
data 0x0250 00 00 00 00 00 00 00 ....
data 0x0257 00 00 00 00 00 00 00 ....
data 0x025E 00 00 00 00 00 00 00 ....
data 0x0265 00 00 00 00 00 00 00 ....
data 0x026C 00 00 00 00 00 00 00 ....
data 0x0273 00 00 00 00 00 00 00 ....
data 0x027A 00 00 00 00 00 00 00 ....
data 0x0281 00 00 00 00 00 00 00 ....
data 0x0288 00 00 00 00 00 00 00 ....
data 0x028F 00 00 00 00 00 00 00 ....
data 0x0296 00 00 00 00 00 00 00 ....

TASK C:


Processor Status
Name
Value
Program Counter
0x00000004
Stack Pointer
0x08FF
X Register
0x0000
Y Register
0x0000
Z Register
0x0000
Status Register
Cycle Counter
3
Frequency
16.000 MHz
Stop Watch
0.19 µs
Registers
Memory 4
Memory:
prog FLASH
Processor Status
Name
Value
Program Counter
0x00000019
Stack Pointer
0x08FF
X Register
0x0000
Y Register
0x0000
Z Register
0x0119
Status Register
Cycle Counter
4784
Frequency
16.000 MHz
Stop Watch
299.00 µs
Registers
Memory 4
Memory:
data IRAM

TASK D:



6.	GITHUB LINK OF THE DA		
https://github.com/jmsikorski/UNLVCpE301Sp16/tree/master/DA1			

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

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