Question 1 [15]

a) What is the function of Computer Operating Properly (COP) Watchdog? [1]

b) Outline the two ways of preventing the watchdog timer, with the sequence of instructions to be executed, from resetting the microprocessor in the middle of a program. [2]

c) What is the function of a reset vector? [2]

d) Write the sequence of instructions that will change the content of the Program Counter (PC) to the target address of the start of program if the RESET Button is pressed while the GT16A is at WAIT State. [2]

e) What is Stack. How can you use Stack Pointer and Index Register in GT16A programming. [2]

f) What is the difference between Stack Pointer and Index Register? [2]

g) Explain with an example how the Stack and Stack Pointer are used for saving and retrieving data automatically during the execution of a pair of instructions JSR and RTS. [2+2]

Question 2 [15]

Here is the basic structure of running a typical GT16A program. Analyze the execution of the sequences of instructions as in Questions 2(a) and 2(b) and explain what happens if these are written in the Main_Program_Section and the program is run in GT16A Microcontroller.

```
;An Example of Basic Structure platform for GT16A Program.

;This project provides a common structure for testing the output

;of a sequence of codes as asked.

;The results can be checked with the contents of registers and memories used.


        ABSENTRY _Startup
```

```
; Define and Name Specific Useful  Locations

RomStart            EQU $C000           ; Start of program memory

Z_RamStart          EQU $0080           ; Start of Direct Page RAM

Z_RamEnd            EQU $00FF           ; End of Direct Page RAM

RamStart            EQU $0100           ; Start of RAM

RamEnd              EQU $087F           ; End of RAM

PortA               EQU $0000           ; Port A

PTADD               EQU $0003           ; Data direction register for Port A

PortB               EQU $0004           ; Port B

PTBDD               EQU $0007           ; Data direction register for Port B

SRSRegister         EQU $1800           ; System Reset Status Register Location

SOPTRegister        EQU $1802           ; System Options Register Location

ResetVector         EQU $FFFE           ; Reset Vector Location

; Constant definition section

        ORG $0100 ; DataArray01=$0100

DataArray01:  DC.B   $70, $51, $62, $6C, $4F, $5D, $6E, $76, $6F, $9C, $7A, $5B, $2C, $3D, $2E,
$2F, $59, $3E, $4F, $75

        ORG $0200  ; DataArray02=$0200

DataArray02:  DC.B   $00, $01, $02, $03, $04, $05, $06, $07, $08, $09, $0A, $0B, $0C, $0D, $0E,
$0F, $10, $11, $12, $13

        ORG $0300  ; Constant=$0300, (Constant)=$20

 ConstantB00:     DC.B $20

       ORG $0400  ; DataArray04=$0400

 DataArray04:  DCB.B 32, 0

      ORG $90

 ArrayCount01:   DC.B   $08

ArrayCount02:   DC.B   $01
```

```
RegisterB:    DS.B 1     ;RegisterB=$91

RegisterC:    DS.B 1     ;RegisterC=$92

RegisterD:    DS.B 1     ;RegisterD=$93

RegisterE:    DS.B 1     ;RegisterE=$94

ConstantB01:   DC.B $AD

ConstantB02:   DS.B 1


ConstantW01:   DC.W $ABCD

ConstantW02:   DS.W 1

; code section

              ORG RomStart  ; code starts in flash memory $C000

; Initialize the Stack Pointer

_Startup:

; Disable the COP Watchdog  to prevent its timer to reset the program at any point

              LDA #$73   ; MSb of SOPT is set to 0 to disable COPE

              STA SOPTRegister

; Initialize Stack Pointer

              LDHX   #DataArray01

              TXS       ; (SP) = #DataArray01 - 1

; Initialize the Index Register

              LDHX   #DataArray04

; Write your Sequence of Codes here in the Main_Program_Section and run and analyze the
program by testing the output.

Main_Program_Section:

MainLoop:


           BRN MainLoop
```

```
 ; Halt the Processor here to see the output using Wait Statement

          WAIT

;Re-direct the program to the location _Startup ($C000) once RESET Button is pressed while the
GT16A is at WAIT State.

           ORG   $FFFE

          DC.W  _Startup         ; Reset
```

a) What happens if you run this program in GT16A? [7]

```
MainLoop:

              MOV #20, ArrayCount01

              MOV #$80, RegisterB

              MOV #$7F, RegisterC

Repeat:

              PULA

Check_Maximum:

              CMPA RegisterB

              BGT Store_Maximum

Check_Minimum:

              CMPA RegisterC

              BLT Store_Minimum

              DBNZ ArrayCount01, Repeat

              BRA End_Loop

Store_Maximum:

              STA RegisterB

              BRA Check_Minimum
```

```
Store_Minimum:
                STA RegisterC
                DBNZ ArrayCount01, Repeat
 End_Loop:
                BRN MainLoop
```

b) What happens if you run this program in GT16A? [8]

```
MainLoop:
                MOV #$08, ArrayCount01
                LDX ConstantW01
                CLRA
Repeat:
                LSLX
                ADC #00
                DBNZ ArrayCount01, Repeat
                DBNZ ArrayCount02, Skip_Loop
                STA RegisterB
                LDX ConstantW01+1
                MOV #$08, ArrayCount01
                CLRA
                BRA Repeat
Skip_Loop:
```

```
STA RegisterC

ADD RegisterB

STA RegisterD

LDA #$10

SUB RegisterD

STA RegisterE

BRN MainLoop
```

Question 3 [20]

a) **Write a GT16A program to multiply a block of 20 data starting at $0100 by a multiplier from direct address $90 and store the double byte products in a block of 40 data starting at $0200. [8]**

b) **Write a GT16A program to display the pattern of the Nybbles of the hexadecimal byte $AD in the GT16A board LEDs alternately at a gap of approximately 3 seconds. [12]**