Exam 2

1. Short Answer [10 pts]

a. [2 pts] Briefly describe what each of the following instructions do so that it is clear what the differences between them are:
   
   STAA -2, X

   STAA 2, -X

   STAA 2, X-

b. [2 pts] Briefly describe the differences between the following instructions:

   ADDB #1
   INCB

   [2 pts] If the X register contains $0900, describe all of the memory accesses that take place when the instruction executes:

   INC [$12, X]

   [2 pts] How many cycles does it take for “LDAA 4, X” to execute?

   [2 pts] What instruction can be used to SET all of the condition code bits simultaneously?
2. [10 pts] Write a subroutine `pmsgx`. This function will print a message to the terminal pointed to by the X register. Print all the characters until a NULL is reached. You may use any of the basic IO routines we used in lecture and lab (they are provided at the end of the exam). Do not affect any registers except X and the CC (condition codes).

```assembly
;*********************************************************
;pmsgx -- Print string pointed to by X register until it
;reaches a NULL character. You may only affect
;the X and CC registers (protect or do not use
;the others)
;
; Input:  X register points to the string to be printed
;        (place your code below)

$include "basicio.asm"

pmsgx
```
3. [10 pts] Write a function “grXY” that will take two unsigned numbers in X and Y and will place the greater number in X and the lesser in Y.

;***********************************************************************
; grXY – Compare the contents of registers X and Y and
; place the greater unsigned number in X and the
; other in Y.
;
; Input: X and Y register contain values to compare
; Output: X and Y
; (place your code below)

grXY
4. [10 pts] The following shows data stored in a 68HC12’s memory, starting at location 0800h through 080Fh. The initial contents of the registers are also given. Assume the five instructions listed (stored elsewhere in memory) are executed in the order listed (each instruction will use the state of the machine as it was after the previous instruction executes). Fill in the blanks for A and the C,N, and Z flags after each instruction executes.

Initial Values: (A) = 00, (CC) = 91, (X) = 0804, (Y) = 0808

(a) ADCA -2,X

(A) = _____h   CF = ___   NF=___   ZF = ___

(b) SBCA [6,Y]

(A) = _____h   CF = ___   NF=___   ZF = ___

(c) LDAA 3,X

(A) = _____h   CF = ___   NF=___   ZF = ___

(d) EORA [0,Y]

(A) = _____h   CF = ___   NF=___   ZF = ___

(e) ANDA 1,+X

(A) = _____h   CF = ___   NF=___   ZF = ___
5. [10 pts] Write a subroutine “onecnt” counts the number of 1’s in the value in the A register. The subroutine should not change any registers other than B (you either have to protect them or not use them). The result should be passed in the B register. For example:
   If \((A) = 65h (= 01100101)\), then B should be set to 04h because A contained 4 ones.

;*******************************************************
; onecnt – Count the number of 1’s in A and put the answer
;          in B.
;
; Input:  A
; Output: B
; (place your code below)

org $0800

onecnt
Declarations
Serial Communications Interface (SCI)
rxdrf equ 20h ; receive buffer full mask pattern
txdre equ 80h ; transmit buffer empty mask pattern
scisr equ 00c4h ; SCI control/status register
scidr equ 00c7h ; SCI transmit/receive data register

Name: inchar
Description: inputs ASCII character from SCI serial port
and returns it in the A register
Returns: ASCII character in A register
Modifies: A register

inchar:
pshc
cil: ldaa #rxdrf ; load receive data register full mask
bita scisr ; check for incoming character
beq cil ; wait if no character received
ldaa scidr ; return ASCII character in A register
pulc
rts

Name: outchar
Description: outputs ASCII character passed in the A register
to the SCI serial port

outchar:
pshc
pshb
ldab #txdre ; load transmit data register empty mask
col: bitb scisr ; check if ready to transmit
beq col ; wait if not ready
staa scidr ; output ASCII character to SCI
pulb
pulc
rts

Name: getbyte
Description: inputs two ASCII characters from the HC12 SCI
and converts them to two hexadecimal digits packed into a single byte
Returns: hex byte equivalent of ASCII characters in A register
Modifies: A register
Calls: inchar, outchar, atoh

getbyte:
pshc
jsr inchar ; get first ASCII character
jsr outchar ; echo character
jsr atoh ; convert ASCII character to hex
bcs errhex1 ; if not hex, go to error routine
asla ; shift converted hex digit
asla
asla
psha ; save on stack temporarily

get2:
jsr inchar ; get second ASCII character
jsr outchar ; echo to screen
; Name: atoh
; Description: converts an ASCII character to a hexadecimal digit
; Inputs: ASCII character passed via A register
; Returns: converted hexadecimal digit returned in A register,
; CF = 0, result OK; CF = 1, error occurred (invalid input)
; Modifies: A register and CC register
;**********************************************************************
atoh:
    pshb
    pshx
    pshy
    suba #30h ; subtract "bias" to get ASCII equivalent
    blt outhex
    cmpa #0ah
    bge cont1
    quithx:
        clc                 ; return with CF = 0 to indicate result OK
        puly
        pulx
        pulb
        rts
    cont1:
        suba #07h
        cmpa #09h
        blt outhex
        cmpa #10h
        blt quithx
        suba #20h
        cmpa #09h
        blt outhex
        cmpa #10h
        blt quithx
    outhex:
        sec                 ; set CF <- 1 to indicate error
        puly
        pulx
        pulb
        rts
;**********************************************************************
; Name: getword
; Description: get 4 ASCII characters and put into a hex word in the
; D register
; Inputs: 4 ASCII characters typed on the keyboard
; Returns: equivalent hex word in the D register
; Reg. Mod.: D,CC
; Calls: getbyte
;**********************************************************************
getword:
    jsr getbyte          ; get first byte of the data entered
    bcs badval           ; is there an error in the first byte
    tfr a,b              ; save MSB in B
    jsr getbyte          ; get second byte of data entered
    bcs badval           ; is there an error in the second byte
    exg a,b              ; put MSB in A and LSB in B
    andcc #$FE           ; no errors, clear Z flag
    rts
badval:
    orcc #$01            ; error, set Z flag
    rts