Solution to Assignment 2
Assembly Language Programming

Question 1

The best way to write assembly-language programs that are more than a few lines long is to start with a high-level version of the program. It is much easier to write, debug and optimize a high-level description of the code.

The 'C' code for a solution to this problem is as follows:

```
#include <stdio.h>

main()
{
  char c, s[10] ;
  int i, swap ;

  /* read in 10 characters */

  for ( i=0 ; i<10 ; i++ ) {
    s[i] = getc ( stdin ) ;
  }

  /* repeatedly iterate over and swap out-of-order
     characters in the array */

  do {

    swap = 0 ;

    for ( i=0 ; i<10 ; i++ ) {

      if ( s[i] > s[i+1] ) {
        c = s[i] ;
        s[i] = s[i+1] ;
        s[i+1] = c ;
        swap = 1 ;
      }

    }

  } while ( swap ) ;

  /* print sorted string */

  for ( i=0 ; i<10 ; i++ ) {
    putc ( s[i], stdout ) ;
  }

}
```

Most C compilers can generate assembly language code instead of object code. Most compilers also do a good job of optimizing the resulting code. Therefore the most efficient way to write assembly language is usually to simplify and/or optimize critical portions of a C compiler's assembly code.

In our case we are restricted to a small subset of the instruction set and it was easier to manually convert the logic of the C program into assembly language:

```
;
; ELEC 379 1998/99 Term 2
; Solution to Assignment 2
; Ed Casas, February 16, 1999
;
; program to read/sort/print 10 characters
;

code segment public
        assume  cs:code,ds:code
        org     100h

start:

; read 10 characters

        mov     bx,offset s     ; start of string
        mov     cx,10   ; number of characters to read
readc:
        mov     ah,1    ; use DOS to read a character
        int     21H
        mov     [bx],al ; store character in string
        inc     bx      ; point to next character in string
        dec     cx      ; decrement characters left to read
        jnz     readc   ; repeat until all read

; bubble sort loop

sort:   mov     al,0    ; clear 'values swaped' flag
        mov     swap,al

        mov     bx,offset s     ; point to first character
        mov     cx,9    ; set count to compare 9 pairs

next2:  mov     al,[bx] ; get character pair into al, ah
        inc     bx      ; and point to next character
        mov     ah,[bx]
        cmp     ah,al   ; compare the pair
        jnc     noswap  ; skip the swap if first is <= second
        mov     [bx],al ; swap
        dec     bx
        mov     [bx],ah
        inc     bx

        mov     al,1    ; set "swapped" flag
        mov     swap,al

noswap: dec     cx      ; compare next pair if not done
        jnz     next2

        mov     al,swap ; do another pass if any swaps
        cmp     al,0
        jnz     sort

; print results

        mov     bx,offset s     ; start of string
        mov     cx,10   ; number of characters to read
printc:
        mov     ah,2    ; use DOS to print a character
        mov     dl,[bx] ;
        int     21H
        inc     bx      ; next character in string
        dec     cx      ; decrement characters to print
        jnz     printc  ; repeat until all printed

; terminate program and return to dos

        int     20H

; variables

s       db      10 dup (?)      ; string to read/sort/print
swap    db      1 dup (?)       ; "swapped" flag

code ends
        end     start
```