```
 1:  /*****************************************************
 2:   * A simple example to illustrate C and assembly language *
 3:   * interface. The test function is written in assembly    *
 4:   * language (in file testex_a.asm).                       *
 5:   *****************************************************/
 6:  #include <stdio.h>
 7:  int main(void)
 8:  {
 9:        int     x=25, y=70;
10:        int     value;
11:        extern  int test(int, int, int);
12:
13:        value = test (x, y, 5);
14:        printf("result = %d\n", value);
15:        return 0;
16:  }
```

```
 1:  ;-------------------------------------------------------
 2:  ; Assembly program for the test function - called from the
 3:  ; C program in file testex_c.c
 4:  ;-------------------------------------------------------
 5:  .MODEL SMALL
 6:  .CODE
 7:  PUBLIC _test
 8:  _test   PROC
 9:          push    BP
10:          mov     BP,SP
11:          mov     AX,[BP+4] ; get argument1 x
12:          add     AX,[BP+6] ; add argument2 y
13:          sub     AX,[BP+8] ; subtract argument3 from sum
14:          pop     BP
15:          ret               ; stack cleared by C function
16:  _test   ENDP
17:          END
```

```
 1:  /*************************************************
 2:   * An example to illustrate call-by-value and    *
 3:   * call-by-reference parameter passing between C and *
 4:   * assembly language modules. The min_max function is *
 5:   * written in assembly language (in file minmax_a.asm). *
 6:   *************************************************/
 7:  #include <stdio.h>
 8:  int main(void)
 9:  {
10:        int     value1, value2, value3;
11:        int     minimum, maximum;
12:        extern  void min_max (int, int, int, int*, int*);
13:
14:        printf("Enter number 1 = ");
15:        scanf("%d", &value1);
16:        printf("Enter number 2 = ");
17:        scanf("%d", &value2);
18:        printf("Enter number 3 = ");
19:        scanf("%d", &value3);
20:
21:        min_max(value1, value2, value3, &minimum, &maximum);
22:        printf("Minimum = %d, Maximum = %d\n", minimum, maximum);
23:        return 0;
24:  }
```

```
 1:  ;-------------------------------------------------
 2:  ; Assembly program for the min_max function - called from
 3:  ; the C program in file minmax_c.c. This function finds the
 4:  ; minimum and maximum of the three integers received by it.
 5:  ;-------------------------------------------------
 6:  .MODEL SMALL
 7:  .CODE
 8:  PUBLIC _min_max
 9:  _min_max   PROC
10:          push    BP
11:          mov     BP,SP
12:          ; AX keeps minimum number and DX maximum
13:          mov     AX,[BP+4]    ; get value 1
14:          mov     DX,[BP+6]    ; get value 2
15:          cmp     AX,DX        ; value 1 < value 2?
16:          jl      skip1        ; if so, do nothing
17:          xchg    AX,DX        ; else, exchange
18:  skip1:
```

```
18:  skip1:
19:           mov     CX,[BP+8]    ; get value 3
20:           cmp     CX,AX        ; value 3 < min in AX?
21:           jl      new_min
22:           cmp     CX,DX        ; value 3 < max in DX?
23:           jl      store_result
24:           mov     DX,CX
25:           jmp     store_result
26:  new_min:
27:           mov     AX,CX
28:  store_result:
29:           mov     BX,[BP+10]   ; BX := &minimum
30:           mov     [BX],AX
31:           mov     BX,[BP+12]   ; BX := &maximum
32:           mov     [BX],DX
33:           pop     BP
34:           ret
35:  _min_max   ENDP
36:           END
```

```
 1:  /*****************************************************
 2:   * A string processing example. Demonstrates processing   *
 3:   * global variables. Calls the string_length               *
 4:   * assembly language program in file string_a.asm file.   *
 5:   *****************************************************/
 6:  #include <stdio.h>
 7:  #define LENGTH 256
 8:
 9:  char string[LENGTH];
10:  int main(void)
11:  {
12:    extern int string_length (char a[]);
13:
14:    printf("Enter string: ");
15:    scanf("%s", string);
16:    printf("string length = %d\n", string_length());
17:    return 0;
18:  }
```

```
 1:  ;-------------------------------------------------
 2:  ; String length function works on the global string
 3:  ; (defined in the C function). It returns string length.
 4:  ;-------------------------------------------------
 5:  .MODEL SMALL
 6:  .DATA
 7:          EXTRN   _string:byte
 8:  .CODE
 9:  PUBLIC  _string_length
10:  _string_length  PROC
11:          mov     AX,0                ; AX keeps the character count
12:          mov     BX,OFFSET _string ; load BX with string address
13:  repeat:
14:          cmp     BYTE PTR[BX],0    ; compare with NULL character
15:          jz      done
16:          inc     AX                  ; increment string length
17:          inc     BX                  ; inc. BX to point to next char.
18:          jmp     repeat
19:  done:
20:          ret
21:  _string_length  ENDP
22:          END
```

```
 1:  /*************************************************
 2:   * An example to illustrate C program calling assembly    *
 3:   * procedure and assembly procedure calling a C function. *
 4:   * This program calls the assembly language procedure     *
 5:   * in file MARKS_A.ASM. The program outputs minimum,      *
 6:   * maximum, and rounded average of a set of marks.        *
 7:   *************************************************/
 8:  #include <stdio.h>
 9:
10:  #define  CLASS_SIZE  50
11:
12:  int main(void)
13:  {
14:        int     marks[CLASS_SIZE];
15:        int     minimum, maximum, average;
16:        int     class_size, i;
17:        int     find_avg(int, int);
18:        extern  void stats(int*, int, int*, int*, int*);
19:
20:        printf("Please enter class size (<50): ");
21:        scanf("%d", &class_size);
22:        printf("Please enter marks:\n");
23:        for (i=0; i<class_size; i++)
24:            scanf("%d", &marks[i]);
25:
```

```
25:
26:         stats(marks, class_size, &minimum, &maximum, &average);
27:         printf("Minimum = %d, Maximum = %d, Average = %d\n",
28:                              minimum, maximum, average);
29:         return 0;
30:   }
31:    /*************************************************
32:     * Returns the rounded average required by the assembly
33:     * procedure STATS in file MARKS_A.ASM.
34:     *************************************************/
35:   int find_avg(int total, int number)
36:   {
37:         return((int)((double)total/number + 0.5));
38:   }
```

```
 1:  ;-------------------------------------------------------
 2:  ; Assembly program example to show call to a C function.
 3:  ; This procedure receives a marks array and class size
 4:  ; and returns minimum, maximum, and rounded average marks.
 5:  ;-------------------------------------------------------
 6:  .MODEL SMALL
 7:  EXTRN   _find_avg:PROC
 8:  .CODE
 9:  PUBLIC _stats
10:  _stats  PROC
11:          push    BP
12:          mov     BP,SP
13:          push    SI
14:          push    DI
15:          ; AX keeps minimum number and DX maximum
16:          ; Marks total is maintained in SI
17:          mov     BX,[BP+4]    ; BX := marks array address
18:          mov     AX,[BX]      ; min := first element
19:          mov     DX,AX        ; max := first element
20:          xor     SI,SI        ; total := 0
21:          mov     CX,[BP+6]    ; CX := class size
```

```
22:  repeat1:
23:           mov     DI,[BX]       ; DI := current mark
24:           ; compare and update minimum
25:           cmp     DI,AX
26:           ja      skip1
27:           mov     AX,DI
28:  skip1:
29:           ; compare and update maximum
30:           cmp     DI,DX
31:           jb      skip2
32:           mov     DX,DI
33:  skip2:
34:           add     SI,DI         ; update marks total
35:           add     BX,2
36:           loop    repeat1
37:           mov     BX,[BP+8]     ; return minimum
38:           mov     [BX],AX
39:           mov     BX,[BP+10]    ; return maximum
40:           mov     [BX],DX
```

```
41:          ; now call find_avg C function to compute average
42:          push    WORD PTR[BP+6] ; push class size
43:          push    SI             ; push total marks
44:          call    _find_avg      ; returns average in AX
45:          add     SP,4           ; clear stack
46:          mov     BX,[BP+12]     ; return average
47:          mov     [BX],AX
48:          pop     DI
49:          pop     SI
50:          pop     BP
51:          ret
52:  _stats  ENDP
53:          END
```

```
 1:  ;-------------------------------------------------------
 2:  ; Assembly program for the min_max function -- called from
 3:  ; the C program in file minmax_c.c. This function finds the
 4:  ; minimum and maximum of the three integers received by it.
 5:  ; Uses ARG to simplify offset calculations of arguments.
 6:  ;-------------------------------------------------------
 7:  .MODEL SMALL
 8:  .CODE
 9:  PUBLIC _min_max
10:  _min_max   PROC
11:          ARG     v1:WORD, v2:WORD, v3:WORD,\
12:                  min_ptr:PTR WORD, max_ptr:PTR WORD
13:          push    BP
14:          mov     BP,SP
15:          ; AX keeps minimum number and DX maximum
16:          mov     AX,[v1]       ; get value 1
17:          mov     DX,[v2]       ; get value 2
18:          cmp     AX,DX         ; value 1 < value 2?
19:          jl      skip1         ; if so, do nothing
20:          xchg    AX,DX         ; else, exchange
21:  skip1:
```

```
21:  skip1:
22:          mov     CX,[v3]      ; get value 3
23:          cmp     CX,AX        ; value 3 < min in AX?
24:          jl      new_min
25:          cmp     CX,DX        ; value 3 < max in DX?
26:          jl      store_result
27:          mov     DX,CX
28:          jmp     store_result
29:  new_min:
30:          mov     AX,CX
31:  store_result:
32:          mov     BX,[min_ptr] ; BX := &minimum
33:          mov     [BX],AX
34:          mov     BX,[max_ptr] ; BX := &maximum
35:          mov     [BX],DX
36:          pop     BP
37:          ret
38:  _min_max   ENDP
39:          END
```

```
 1:  ;-----------------------------------------------------
 2:  ; Assembly program example to show call to a C function.
 3:  ; This procedure receives a marks array and class size
 4:  ; and returns minimum, maximum, and rounded average marks.
 5:  ; Uses TASM's extended procedure call instruction.
 6:  ;-----------------------------------------------------
 7:  .MODEL SMALL
 8:  EXTRN C  find_avg:PROC
 9:  .CODE
10:  PUBLIC C stats
11:  stats   PROC
12:          ARG     marks:PTR WORD, class_size:WORD, min:PTR WORD,\
13:                  max:PTR WORD, avg:PTR WORD
14:          push    BP
15:          mov     BP,SP
16:          push    SI
17:          push    DI
```

```
18:           ; AX keeps minimum number and DX maximum
19:           ; Marks total is maintained in SI
20:           mov     BX,[marks]   ; BX := marks array address
21:           mov     AX,[BX]      ; min := first element
22:           mov     DX,AX        ; max := first element
23:           xor     SI,SI        ; total := 0
24:           mov     CX,[class_size]
25:   repeat1:
26:           mov     DI,[BX]      ; DI := current mark
27:           ; compare and update minimum
28:           cmp     DI,AX
29:           ja      skip1
30:           mov     AX,DI
31:   skip1:
32:           ; compare and update maximum
33:           cmp     DI,DX
34:           jb      skip2
35:           mov     DX,DI
36:   skip2:
```

```
skip2:
        add     SI,DI           ; update marks total
        add     BX,2
        loop    repeat1
        mov     BX,[min]        ; return minimum
        mov     [BX],AX
        mov     BX,[max]        ; return maximum
        mov     [BX],DX
        ; now call find_avg C function to compute average
        ; returns the rounded average value in AX
        call    find_avg C, SI, class_size
        mov     BX,[avg]        ; return average
        mov     [BX],AX
        pop     DI
        pop     SI
        pop     BP
        ret
stats   ENDP
        END
```

```
 1:  /*************************************************************
 2:   * This program illustrates how inline assembly code can be      *
 3:   * written. It uses the interrupt service of DOS (int 21H)       *
 4:   * to get the current month information.                          *
 5:   *************************************************************/
 6:  #include        <stdio.h>
 7:
 8:  int current_month(void);
 9:
10:  int main(void)
11:  {
12:        printf ("Current month is: %d\n", current_month());
13:        return 0;
14:  }
15:  int current_month(void)
16:  {
17:        asm  mov    AH,2AH
18:        asm  int    21H
19:        asm  xor    AX,AX   /* we really want to clear AH */
20:        asm  mov    AL,DH
21:  }
```