To write an assembly language module in a C program we need to know how to link the two programs together and how to pass parameters between the two. Linking any two modules together, whether they are both in C, both in assembly, or mixed is done by using the extern and public directives.

*Public and Extern directives*

Public – This directive is used to make a symbol in one module available in another module. For example, if a function written in assembly code is to be used by a C program then the assembly code module name or entry symbol should be declared public.

Example:

```
public assem
public bcdmodule, asciimodule
```

Extern – This directive is used to indicate that a symbol being used in one module is not defined in that module. For example, a C program may call an assembly language module. In this case, the assembly language module name must be declared extern in the C-code.

Example:

```
extern assem
```

It is the job of the linker to match up the public and extern directives at compile time. Every extern declaration must have a corresponding public declaration in another module.

*Parameter passage*

In general, C passes parameters to functions by way of the stack but this may differ from compiler to compiler. Since the 8051 has only a small amount of read/write memory and stack operation are slow, the Keil C-compiler uses a mix of registers and the stack to pass parameters. For a Keil C program the first three parameters are passed in registers and all parameters beyond the first three are pushed onto the stack. All parameters are passed as 16-bit numbers even though they may be declared as 8-bit quantities.

The first parameter goes into registers R6-R7, the second in R4-R5, and the third in R2-R3. (R3, R5, and R7 are the least significant bytes.) Functions which return a value place the value in R6-R7.

The program example on the following page is a main program in C which calls an external assembly language program. The assembly language program calls a C-function. The main C-code passes one parameter to the assembly function and gets one value back. The C-function returns an unsigned char to the assembly code. You can use the simulator to verify that the parameter passage takes place in R6-R7.
This is the main C code – it calls an assembly function which, in turn, calls a C function.

/*MixedTst.c
   This is the main C code.
   This program is a test of parameter passage between C and assembly
   language. It is meant to be run on the simulator only.
   Mixed.c cfunc.c and Assem.A51 are in a common project.

   Note that the c-compiler issues a warning for an uncalled
   segment generated by the c_func
*/

//Assem is an external assembly language program.
extern int Assem(int, unsigned char);
//Needs a prototype for the cfunction
unsigned char cfunc(void);

void main (void)
{
    unsigned char i;
    int KInt, PInt;
    i = 0x080;
    PInt = 0x6789;
    // Call the assembly language program and pass it an int and an unsigned char
    // PInt will go into R6:R7 and i will go in R4:R5 with R4 = 0.
    KInt = Assem(PInt, i);
    while(1);
}

This is the C function which is called by the assembly code
//This is the C function called only by the assembly code
// It returns an unsigned char
unsigned char cfunc()
{
    unsigned char a;
    a = 5;
    a = a + 1;
    return a;
}
This is the assembly module which is called by the main program. It calls the c function.

;Assem.A51
; This program is in assembly language and is activated by a function call
; in C. It accepts and int and an unsigned char as parameters and
; returns an int.
; At the end of the assembly code there is a long call to a c-function
; which returns an unsigned char.
; Since this routine is called by a C program it's name must begin with
; an underscore.
public _Assem
; Define a relocatable code segment named Assemb
Assemb SEGMENT CODE
;Declare the c function as external
EXTRN CODE (cfunc)
;
; Select Assemb as the active segment
RSEG Assemb
*********************************************************
;NOTE:
; The SEGMENT and RSEG directives are used to create a relocatable
; set of assembly statements for general use. These can be replaced
; with a single directive that says
;            CSEG AT 8000h
; which locates the code at 8000h as an absolute segment. If this is
; done, you must assure that the original C code does not over lap this
; space. To locate the C code choose Options - Banking Linker -
; Size/Location and set the code to 8100h.
**********************************************************

_Assem is the entry point for the C language call.
_Assem: mov A, R7 ;R7 has the LSB of Pint
       add A, R5 ;R5 has the LSB of i
       mov R7, A ;Store LSB of return value in R7
       mov A, R6 ;Get the MSB of PInt
       addc A, #0 ;MSB of i is 0 so add that with the carry
       mov R6, A ;Put the MSB of the result in R6 as a return value
using 0 ;Make sure we are in RB0
       lcall cfunc ;Call the c function
       mov P2, R7; ;The return data comes back in R7
       ret
end