To help you out, the extended `asm` format provides *placeholders* that can be used to reference input and output values within the inline assembly code. This enables you to declare input and output values in any register or memory location that is convenient for the compiler.

The placeholders are numbers, preceded by a percent sign. Each input and output value listed in the inline assembly code is assigned a number based on its location in the listing, starting with zero. The placeholders can then be used in the assembly code to represent the values.

For example, the following inline code:

```
asm ("assembly code"
     : "=r"(result)
     : "r"(data1), "r"(data2));
```

will produce the following placeholders:

- `%0` will represent the register containing the `result` variable value.
- `%1` will represent the register containing the `data1` variable value.
- `%2` will represent the register containing the `data2` variable value.

Notice that the placeholders provide a method for utilizing both registers and memory locations within the inline assembly code. The placeholders are used in the assembly code just as the original data types would be:

```
imull %1, %2
movl %2, %0
```

Remember that you must declare the input and output values as the proper storage elements (registers or memory) required by the assembly instructions in the inline code. In this example, both of the input values were required to be loaded into registers for the ***IMULL*** *instruction.*

To demonstrate using placeholders, the `regtest2.c` program performs the same function as the `regtest1.c` program, but enables the compiler to choose which registers to use:

```
/* regtest2.c - An example of using placeholders */
#include <stdio.h>

int main()
{
   int data1 = 10;
   int data2 = 20;
   int result;

   asm ("imull %1, %2\n\t"
        "movl %2, %0"
        : "=r"(result)
        : "r"(data1), "r"(data2));

   printf("The result is %d\n", result);
   return 0;
}
```