Extended ASM

The basic `asm` format provides an easy way to create assembly code, but it has its limitations. For one, all input and output values have to use global variables from the C program. In addition, you have to be extremely careful not to change the values of any registers within the inline assembly code.

The GNU compiler provides an extended format for the `asm` section that helps solve these problems. The extended format provides additional options that enable you to more precisely control how the inline assembly language code is generated within the C or C++ language program. This section describes the extended `asm` format.

Extended ASM format

Because the extended `asm` format provides additional features to use, they must be included in the new format. The format of the extended version of `asm` looks like this:

```
asm ("assembly code" : output locations : input operands : changed registers);
```

This format consists of four parts, each separated by a colon:

- **Assembly code:** The inline assembly code using the same syntax used for the basic `asm` format
- **Output locations:** A list of registers and memory locations that will contain the output values from the inline assembly code
- **Input operands:** A list of registers and memory locations that contain input values for the inline assembly code
- **Changed registers:** A list of any additional registers that are changed by the inline code

Not all of the sections are required to be present in the extended `asm` format. If no output values are associated with the assembly code, the section must be blank, but two colons must still separate the assembly code from the input operands. If no registers are changed by the inline assembly code, the last colon may be omitted.

The following sections describe how to use the extended `asm` format.

Specifying input and output values

In the basic `asm` format, input and output values are incorporated using the C global variable name within the assembly code. Things are a little different when using the extended format.

In the extended format, you can assign input and output values from both registers and memory locations. The format of the input and output values list is

```
"constraint"(variable)
```

where `variable` is a C variable declared within the program. In the extended `asm` format, both local and global variables can be used. The `constraint` defines where the variable is placed (for input values) or