C/C++ keyword: asm

Syntax

```
asm( "instruction" );
```

The `asm` command allows you to insert assembly language commands directly into your code. The `__asm__` keyword is recognized and is equivalent to the `asm` token. Extended syntax is supported to indicate how assembly operands map to C/C++ variables.

Example

```
asm("fsinx %1,%0" : "=f"(x) : "f"(a));
// Map the output operand on "x",
// and the input operand on "a".
```

C/C++ keyword: volatile

The volatile keyword is an implementation-dependent type qualifier, used when declaring variables, which prevents the compiler from optimizing those variables. Volatile should be used with variables whose value can change in unexpected ways (i.e. through an interrupt), which could conflict with optimizations that the compiler might perform.

Processor specific keyword: __asm() (Nios II)

Syntax

```
__asm( "instruction_template"
       [ : output_param_list
       [ : input_param_list
       [ : register_save_list]]] );
```

With the `__asm()` keyword you can use assembly instructions in the C source and pass C variables as operands to the assembly code.

instruction_template	Assembly instructions that may contain parameters from the input list or output list in the form: **%***parm_nr*
%*parm_nr*[**.***regnum*]	Parameter number in the range 0 .. 9. With the optional **.***regnum* you can access an individual register from a register pair. For example, with the word register `R12`, `.0` selects register `R1`.
output_param_list	[[**"=**[**&**]*constraint_char***"(**C_expression**)],**...]
input_param_list	[[**"***constraint_char***"(**C_expression**)],**...]
&	Says that an output operand is written to before the inputs are read, so this output must not be the same register as any input.
constraint _char	Constraint character: the type of register to be used for the *C_expression*.
C_expression	Any C expression. For output parameters it must be an *lvalue*, that is, something that is legal to have on the left side of an assignment.
register_save_list	[[**"***register_name***"],**...]
register_name:q	Name of the register you want to reserve.

Constraint	Type	Operand	Remark
R	general purpose register (64 bits)	r0 .. r31	Based on the specified register, a register pair is formed (64-bit). For example r0:r1.
r	general purpose register (32 bits)	r0 .. r31	
i	immediate value	#value	
l	label	*label*	
m	memory label	*variable*	stack or memory operand, a fixed address
number	other operand	same as *%number*	Input constraint only. The *number* must refer to an output parameter. Indicates that %*number* and *number* are the same register. Use %*number*.0 and %*number*.1 to indicate the first and second half of a register pair when used in combination with R.