Beginner's Guide to C Inline Assembly Programming using GCC

Introduction to Inline Assembly

Inline Assembly allows embedding assembly instructions directly within C code, providing low-level access to hardware and performance optimization.

Basic Syntax (GCC)

```
__asm__ ("assembly code");
```

or (preferred modern syntax):

```
asm ("assembly code");
```

Example:

```
asm ("movl $5, %eax");
```

Components of Inline Assembly

```
asm volatile ("assembly code"
        : output_operands
        : input_operands
        : clobbered_registers);
```

1. **asm or __asm__**: Keyword to start inline assembly.

2. **"assembly code"**: The actual assembly instructions.

3. **output_operands**: Variables modified by assembly.

4. **input_operands**: Variables used by assembly.

5. **clobbered_registers**: Registers modified unexpectedly.

Simple Examples

Example 1: Add Two Numbers

```
#include <stdio.h>
int main() {
 int a = 10;
 int b = 20;
 int sum;
 asm volatile (
   "addl %1, %0\n\t"  // Add b to a and store in a. %0 refers to sum, %1 to b
   : "=r" (sum)       // Output: sum (write-only, in a general register)
   : "r" (a), "r" (b) // Input: a and b (in general registers)
   :                  // Clobbered registers (none in this case)
 );
 printf("Sum: %d\n", sum);
 return 0;
}
```

Example 2: Swap Two Integers

```
#include <stdio.h>
int main() {
    int a = 10, b = 20;
    asm volatile("xchg %0, %1" : "=r"(a), "=r"(b) : "0"(a), "1"(b));
    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

Operand Constraints

Constraint	Meaning
"r"	General-purpose register
"m"	Memory operand
"a"	EAX register
"b"	EBX register
"c"	ECX register
"d"	EDX register

Clobber List

Used to tell the compiler which registers or memory might be modified by assembly code.

Example:

```
asm volatile ("movl $0, %%eax;" : : : "%eax");
```

NOTE

Understand AT&T syntax (used by GCC):

- Source comes before destination.
- Registers are prefixed with %.
- Constants are prefixed with $.

AT&T vs Intel Syntax

Feature	AT&T Syntax	Intel Syntax
Register	%eax	EAX
Immediate	$5	5
Memory reference	(%eax)	[EAX]
Order	Source → Dest	Dest ← Source

//Print 8 bit register

```
#include <stdio.h>
int main() {
    unsigned char value;

    // Put a value into AL and move it to our C variable
    asm volatile(
        "movb $0x5A, %%al\n\t"   // Put 0x5A into AL
        "movb %%al, %0\n\t"      // Move AL into 'value'
        : "=r"(value)            // output operand
        :                        // no input operands
        : "%al"                  // clobbered register
    );
    printf("Value in AL: %c \n", value);
    return 0;
}
```

A **clobbered register** means:
"This register's value will be changed ('clobbered') by my assembly, so GCC should not assume it still holds its old value afterward."

Syntax	Meaning	Example
`%0`, `%1`	Operand placeholders	`%0` → output variable
`%%eax`, `%%al`	Literal register names	`%%al` → `AL` register

```
asm("mov %0, %1" : "=r"(out) : "r"(in));
```