
EXAMPLE 1

Description:
Code referencing to one of the MCU registers.

Notes:
Register names can be included in inline assembler in GCC C code. GCC compiler usually takes care of the proper register assignment in the code surrounding inline assmbler.

IAR C requires to use inline function and explicitly move a required register to register R0. Functions are inlined only with the highest optimization levels It is recommended NOT to use function inlining, i.e. your code should include a function call not inline code. This is because the compiler does not know about register assignment. If function inlining is used it is possible to generate a code where a required register is moved to R0, but R0 is overwritten later.

Danger warning for IAR code: **HIGH**

GCC code example:

---

register char *stack_ptr asm ("sp");

```
char Test_SP_Code(void)
{
          char *Ptr;
          Ptr = stack_ptr;
          return *Ptr;
}

char * Test_SP_Code_Ptr(void)
{
          char *Ptr;
          Ptr = stack_ptr;
          return Ptr;
}
```

GCC assembler output:

```
          .global    Test_SP_Code
          .type      Test_SP_Code,function
Test_SP_Code:
          @ args = 0, pretend = 0, frame = 0
          @ frame_needed = 0, current_function_anonymous_args = 0
          @ link register save eliminated.
          ldrb       r0, [sp, #0]          @ zero_extendqisi2
          mov        pc, lr
.Lfe1:
          .size      Test_SP_Code,.Lfe1-Test_SP_Code
          .align     2
          .global    Test_SP_Code_Ptr
          .type      Test_SP_Code_Ptr,function
Test_SP_Code_Ptr:
          @ args = 0, pretend = 0, frame = 0
          @ frame_needed = 0, current_function_anonymous_args = 0
          @ link register save eliminated.
          mov        r0, sp
          mov        pc, lr
.Lfe2:
          .size      Test_SP_Code_Ptr,.Lfe2-Test_SP_Code_Ptr
```

Acknowledgment: Ivan Paulik @ PASAT, Bratislava

Equivalent IAR C code:

```
#pragma diag_suppress=Pe940
#pragma inline = forced // Optimization at HIGH
inline char * get_SP( void )
{
  /* On function exit,
  function return value should be present in R0 */
  asm( "MOV R0, SP" );
}

char Test_SP_Code(void)
{
           char *Ptr;
           Ptr = get_SP();
           return *Ptr;
}

char * Test_SP_Code_Ptr(void)
{
           char *Ptr;
           Ptr = get_SP();
           return Ptr;
}
```

IAR assembler output:

```
      7          #pragma diag_suppress=Pe940
      8          // Optimization at HIGH !!!!
      9          #pragma inline = forced
     10          inline char * get_SP( void )
     11          {
     12            /* On function exit,
     13            function return value should be present in R0 */
     14            asm( "MOV R0, SP" );
     15          }
     16

   \                                In section .text, align 2, keep-with-next
     17          char Test_SP_Code(void)
     18          {
     19            char *Ptr;
     20
     21            Ptr = get_SP();
   \                        Test_SP_Code:
   \   00000000   6846               MOV R0, SP
     22            return *Ptr;
   \   00000002   0078               LDRB     R0,[R0, #+0]
   \   00000004   7047               BX       LR               ;; return
     23          }
     24

   \                                In section .text, align 2, keep-with-next
     25          char * Test_SP_Code_Ptr(void)
     26          {
     27               char *Ptr;
     28
     29               Ptr = get_SP();
   \                        Test_SP_Code_Ptr:
   \   00000000   6846               MOV R0, SP
     30
     31               return Ptr;
   \   00000002   7047               BX       LR               ;; return
     32          }
```

Acknowledgment: Ivan Paulik @ PASAT, Bratislava