Esse é o oitavo artigo da série escrita pelo engenheiro Ismael Lopes da Silva, exclusivamente para o site "www.embarcados.com.br". Nessa série focarei no Microcontrolador da STMicroelectronics, o MCU STM32F103C8T6, que é um ARM Cortex-M3. Os pré-requisitos para uma boa compreensão dos artigos é ter o domínio da Linguagem C Embedded e conceitos de eletrônica.

Alguns Registradores do Núcleo (Core) do Processador ARM Cortex-M3 na Pratica

No artigo anterior vimos a teoria de todos os Registradores do núcleo do processador ARM Cortex-M3. No decorrer dessa série veremos detalhes da aplicação desses Registradores, mas, agora veremos como funciona os Registradores Program Counter (PC), Link (LR) e o Stack Pointer (SP).

Usando o STM32CubeIDE, no mesmo workspace que criamos os artigos anteriores, então, vamos copiar o projeto "03NivelAcesso" como "04RegPC_LR_SP". Na janela "Projetc Explorer", clique com o botão direito do mouse sobre o projeto "03NivelAcesso" e selecione "Copy". Novamente na janela "Projetc Explorer", clique com o botão direito do mouse sobre o projeto "03NivelAcesso" e selecione "Paste". Uma janela para renomear a aplicação será mostrada, portanto, entre como o nome "04RegPC_LR_SP". Depois clique no botão [Copy].

Fizemos uma cópia porque tudo que preparamos é mantido, então, vamos apenas editar o arquivo main.c. Segue o novo arquivo main.c, que é uma aplicação para verificarmos o comportamento dos Registradores PC, LR e SP. O foco não é explicar código em linguagem C, mas, entender o funcionamento de alguns Registradores.

```
/******************************************************************************
 * @file           : main.c
 * @author         : Auto-generated by STM32CubeIDE
 * @brief          : Main program body
 ******************************************************************************
 * @attention
 *
 * <h2><center>&copy; Copyright (c) 2019 STMicroelectronics.
 * All rights reserved.</center></h2>
 *
 * This software component is licensed by ST under BSD 3-Clause license,
 * the "License"; You may not use this file except in compliance with the
 * License. You may obtain a copy of the License at:
 *                        opensource.org/licenses/BSD-3-Clause
 *
 ******************************************************************************/

#if !defined(__SOFT_FP__) && defined(__ARM_FP)
  #warning "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."
#endif

#include<stdio.h>

void testar_LR_SP()
{
```

```
        int var1 = 10;
        var1 += 1;
}

int main(void)
{
        printf("Registrador PC\n");
        testar_LR_SP();
        __asm volatile ("NOP");
        __asm volatile ("NOP");
}
```

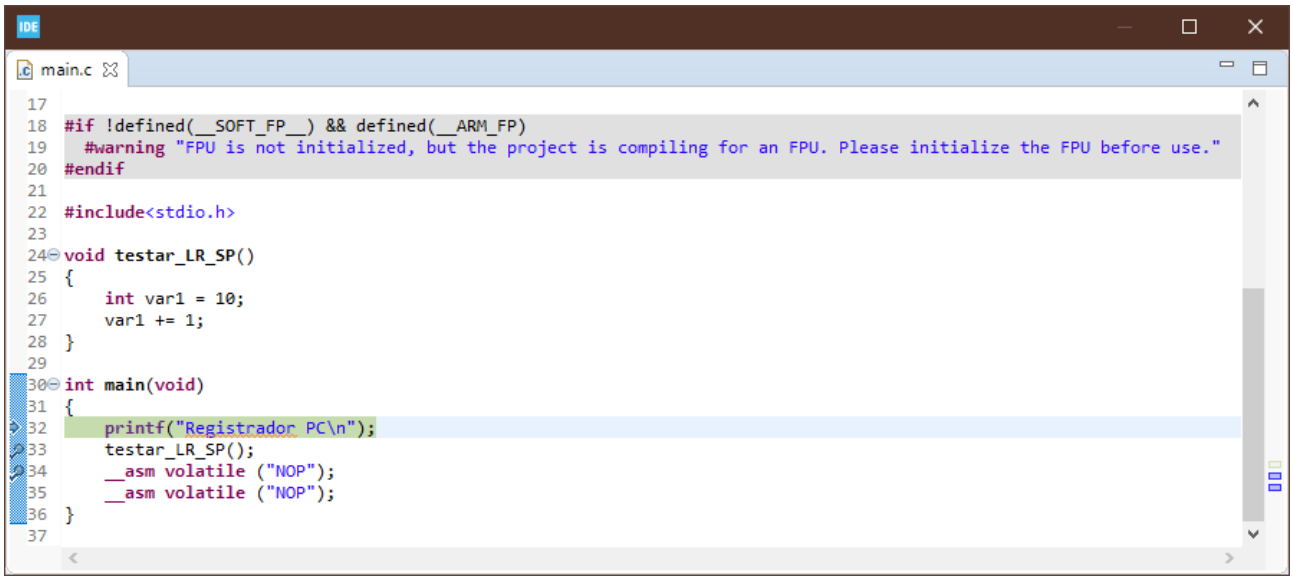

Figura 1 – Programa main.c

Como temos uma nova aplicação "04RegPC_LR_SP", então, vamos dar um "Clean Project", "Build Project" e ative a perspectiva de depuração. Vamos monitorar com uma nova janela, chamada "Disassembly". Nessa janela o código da aplicação em linguagem C é visualizado, e também o correspondente em linguagem Assembly, gerado pelo compilador. Para habilitar a janela " Disassembly", clique no menu "Window", selecione "Show View" e depois "Disassembly".

Como estamos no processo de depuração, a primeira linha de programa na janela "main.c" está em destaque, e também a linha equivalente na janela "Disassembly", conforme ilustrado na figura 1 e 2. Também observe que foi inserido dois 'breakpoints', na linha 33 e 34.

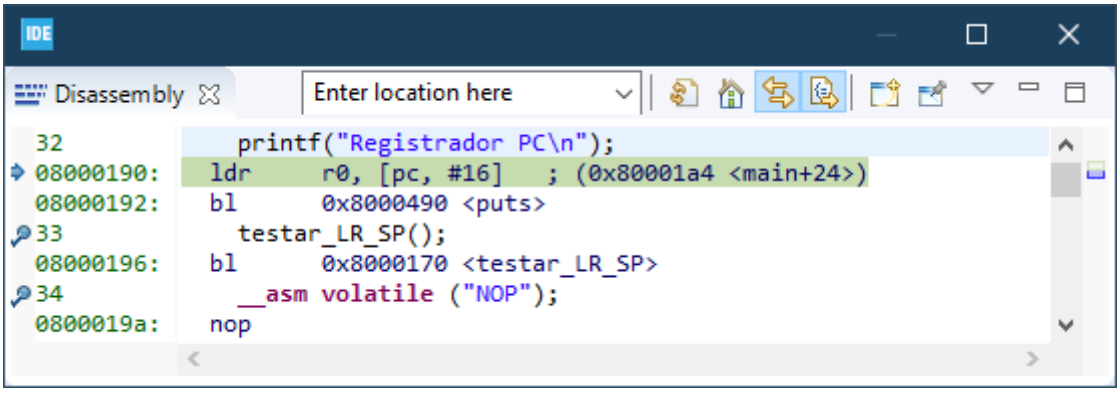

Figura 2 – Trecho de programa na janela Disassembly

Vamos habilitar a funcionalidade “Instrution Stepping Mode” do STM32CubeIDE, para que o código na janela “Disassembly” caminhe linha por linha, então, clique no botão “Instrution Stepping Mode”, que tem a letra “i” com uma seta para a direita “i→”.

Também vamos visualizar a janela “Registers” e monitorar os valores dos Registradores PC, LR e SP, em hexadecimal, conforme mostrado a seguir:

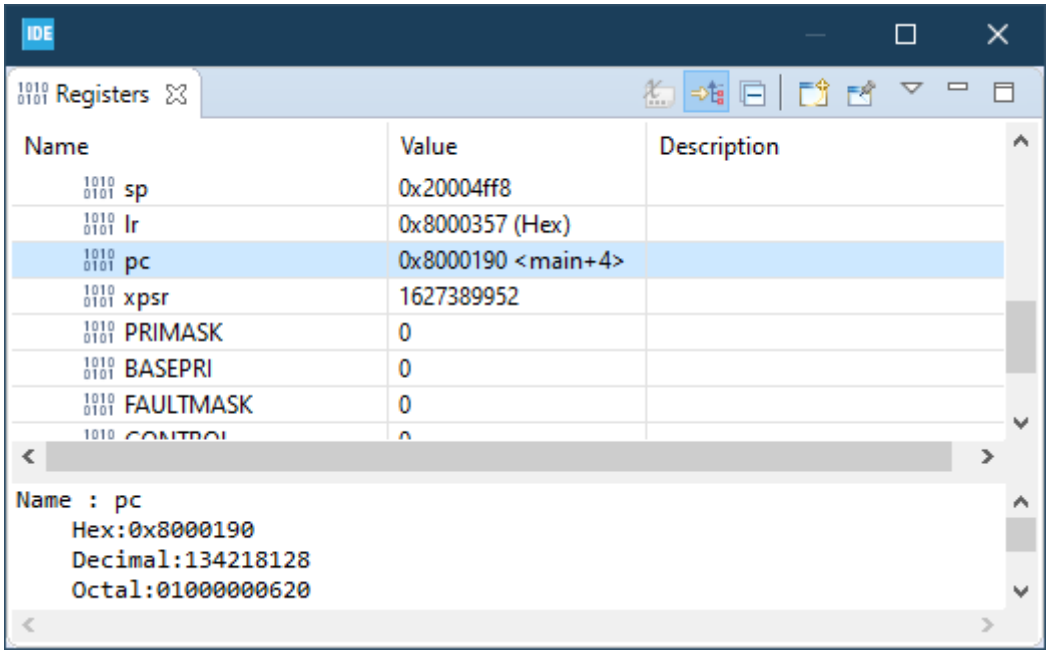

Figura 3 – Monitorar os Registradores PC, LR e SP

O Registrador Contador do Programa - Program Counter (PC)

A definição do artigo anterior foi que “O Registrador Program Counter é o Registrador R15. Ele contém o endereço na memória de código (FLASH) que aponta para a instrução atual que será executada”.

Na janela “Disassembly” podemos acompanhar a posição do endereço da memória de código, então, isso nos ajudará a entender o funcionamento do Registrador Program Counter (PC).

Na janela “Disassembly” a linha em destaque é:

```
08000190:  ldr     r0, [pc, #16]   ; (0x80001a4 <main+24>)
```

O conteúdo do Registrador PC é:

Name : pc

Hex:0x8000190

Veja que o Registrador PC contém o endereço da instrução que será executada. Vamos dar mais um passo na linha do programa, clicando uma vez o botão “Step Into” ou pressionando uma vez a tecla [F5], e monitorarmos o valor do Registrador PC.

```
08000192:  bl      0x8000490 <puts>
```

Name : pc

Hex:0x8000192

Conclusão: O Registrador Program Counter (PC) sempre aponta para a posição de memória de programa que contém a instrução que será executada.

O Registrador Link (LR)

A definição do artigo anterior foi que “O Registrador Link é o registro R14. Ele armazena o endereço de retorno das sub-rotinas, funções chamadas e exceções”.

Como nosso programa de aplicação está prestes a chamar uma função chamada “testar_LR_SP”, então, vamos monitorar o conteúdo do Registrador LR, quando essa função for chamada. Clique no botão ‘Resume’ ou pressione a tecla [F8], que o programa avançará até o ‘breakpoint’ da linha 33, conforme ilustrado na figura 3.

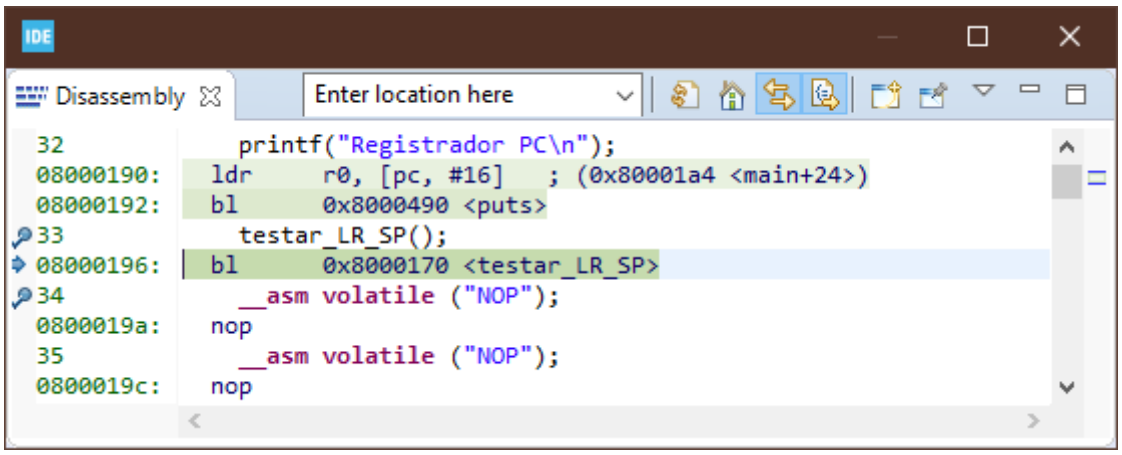

Figura 3 – Trecho do programa de aplicação

O programa está preste a saltar para a função “testar_LR_SP”. O Registrador PC contém “0x8000196” e o Registrador LR contém um valor qualquer. Observe o Registrador LR ao saltar para função “testar_LR_SP”. Vamos dar mais um passo na linha do programa, clicando uma vez o botão “Step Into” ou pressionando uma vez a tecla [F5].

O Registrador PC contém “0x8000170”, que aponta para a primeira instrução da função “testar_LR_SP”, e o Registrador LR contém “0x800019b”, conforme ilustrado na figura 4. Acredito que por um bug na ferramenta de depuração o conteúdo do Registrador é “0x800019b”, mas, deveria ser “0x800019a”.

Mesmo assim, veremos que após encerrar o processamento das instruções da função “testar_LR_SP”, o conteúdo do Registrador LR será copiado para o Registrador PC, apontando para a próxima instrução após a instrução que chamou a função “testar_LR_SP”. O código continuará sendo executado normalmente, após ter retornado da rotina “testar_LR_SP”.

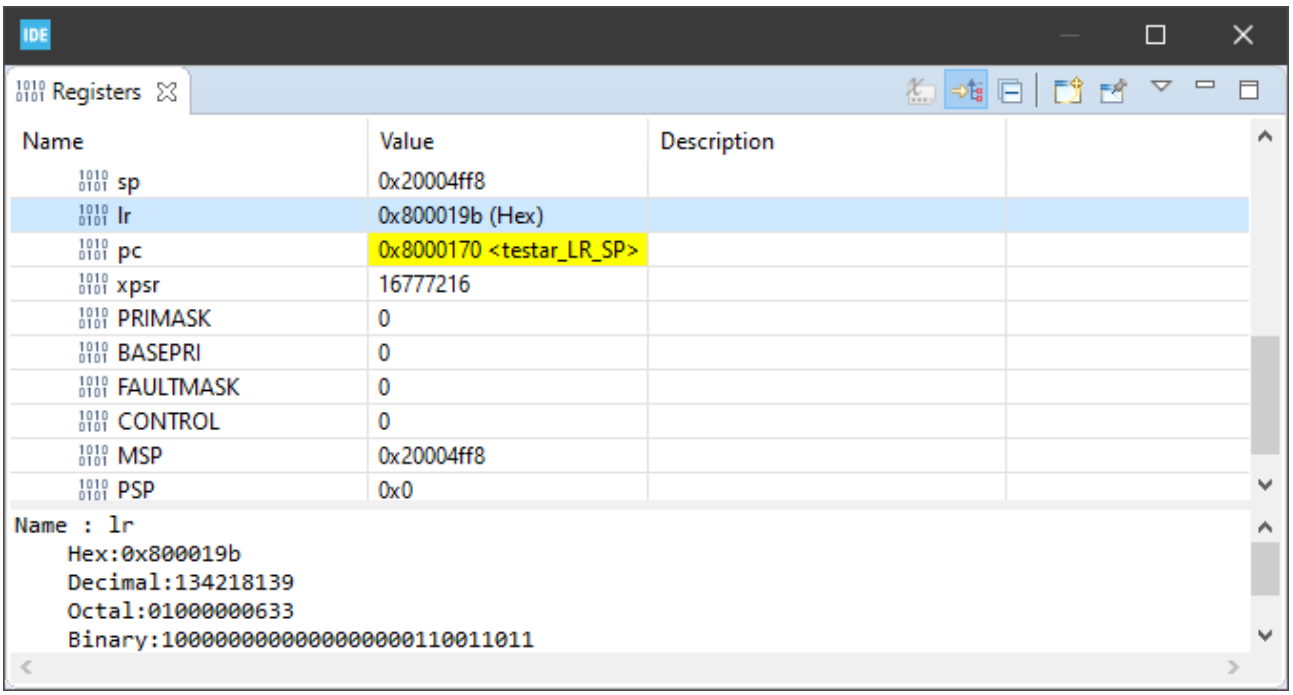

Figura 4 – Monitorando o Registrador LR

Clique no botão 'Resume' ou pressione a tecla [F8], que o programa avançará até o 'breakpoint' da linha 34, retornando da função "testar_LR_SP", conforme ilustrado a seguir.

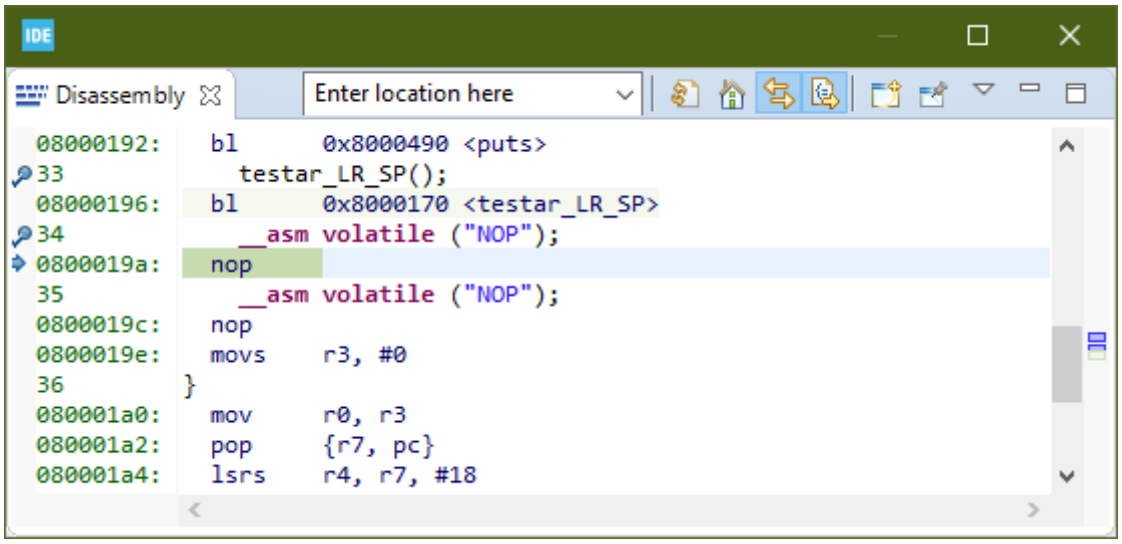

Figura 5 – Janela 'Disassembly', retorno da função "testar_LR_SP"

Na janela "Disassembly" podemos ver que o Registrador PC foi carregado com o conteúdo do Registrador LR, portanto, o Registrador PC está apontando para a posição de memória de programa "0x800019a", que é a posição de retorno da função "testar_LR_SP", conforme ilustrado na figura 5 e 6.

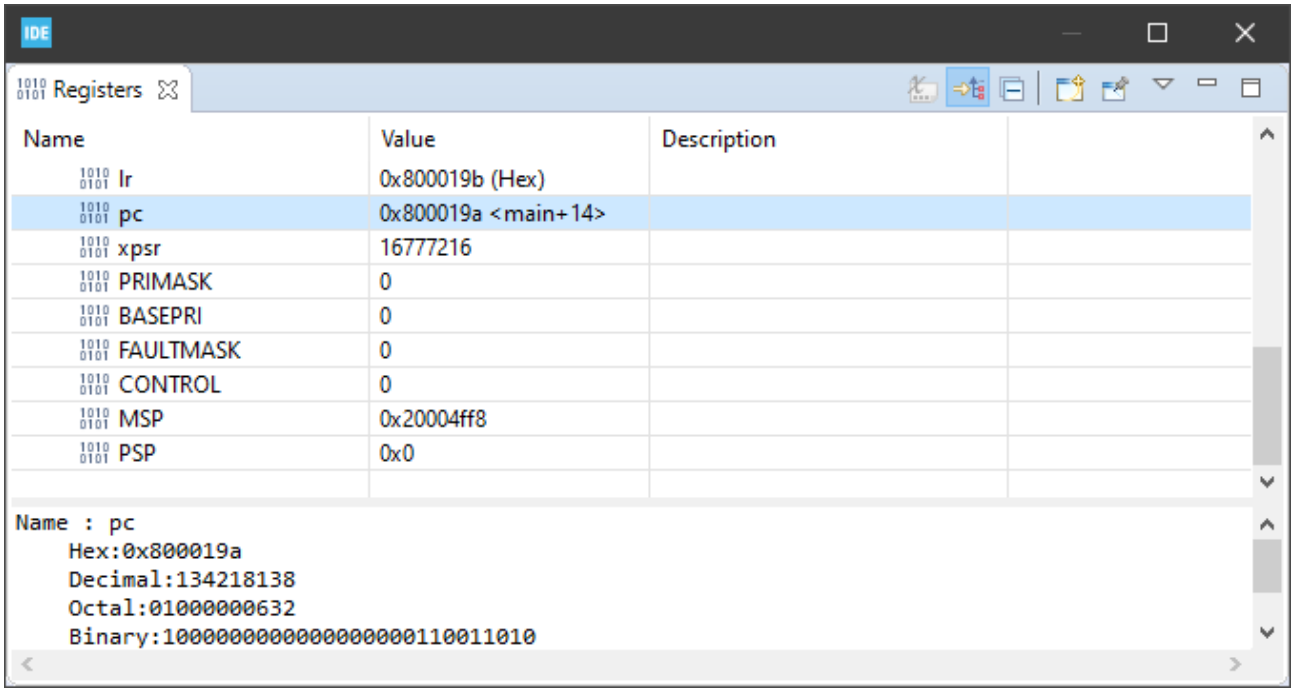

Figura 6 – Registrador PC apontando para a posição de retorno da função chamada

O Registrador Ponteiro da Pilha - Stack Pointer (SP)

O Registrador Stack Pointer é o Registrador R13. Esse Registrador é usado para acessar a seção stack da memória de dados SRAM.

Vamos reiniciar o processo de depuração, que inicialmente ficará conforme ilustrado na figura 1. A janela "Disassembly". Habilite a funcionalidade "Instrution Stepping Mode" do STM32CubeIDE, para que o código na janela "Disassembly" caminhe linha por linha, então, clique no botão "Instrution Stepping Mode", botão "i→".

Clique no botão 'Resume' ou pressione a tecla [F8], que o programa avançará até o 'breakpoint' da linha 33, que é a chamada para a função "testar_LR_SP".

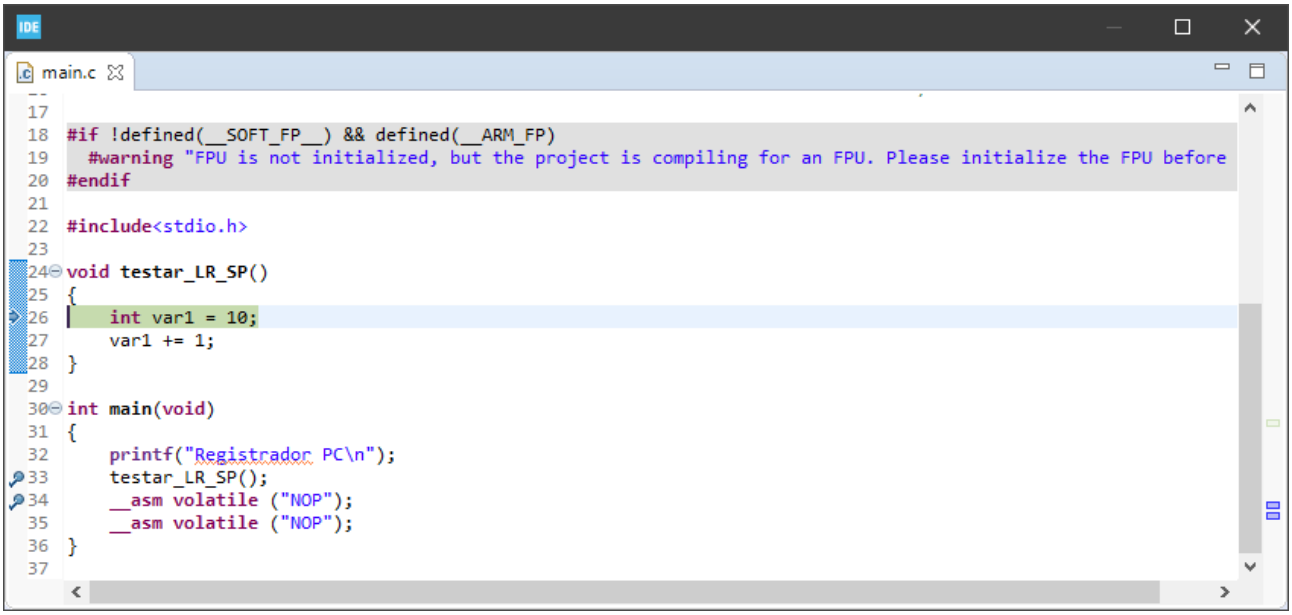

Figura 7 – Definindo um valor a variável var1

Dê um passo na linha do programa, clicando uma vez no botão “Step Into” ou pressionando uma vez a tecla [F5], conforme ilustrado da figura 8.

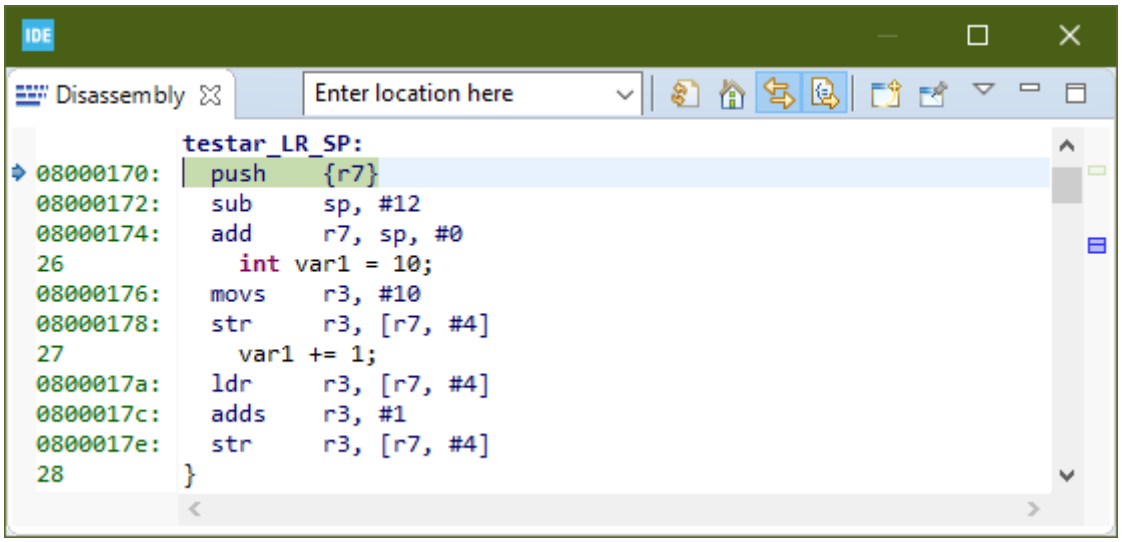

Figura 8 – Carregando o dado imediato para R3

Na janela “Registers” monitore os valores dos Registradores, conforme ilustrado na figura 9.

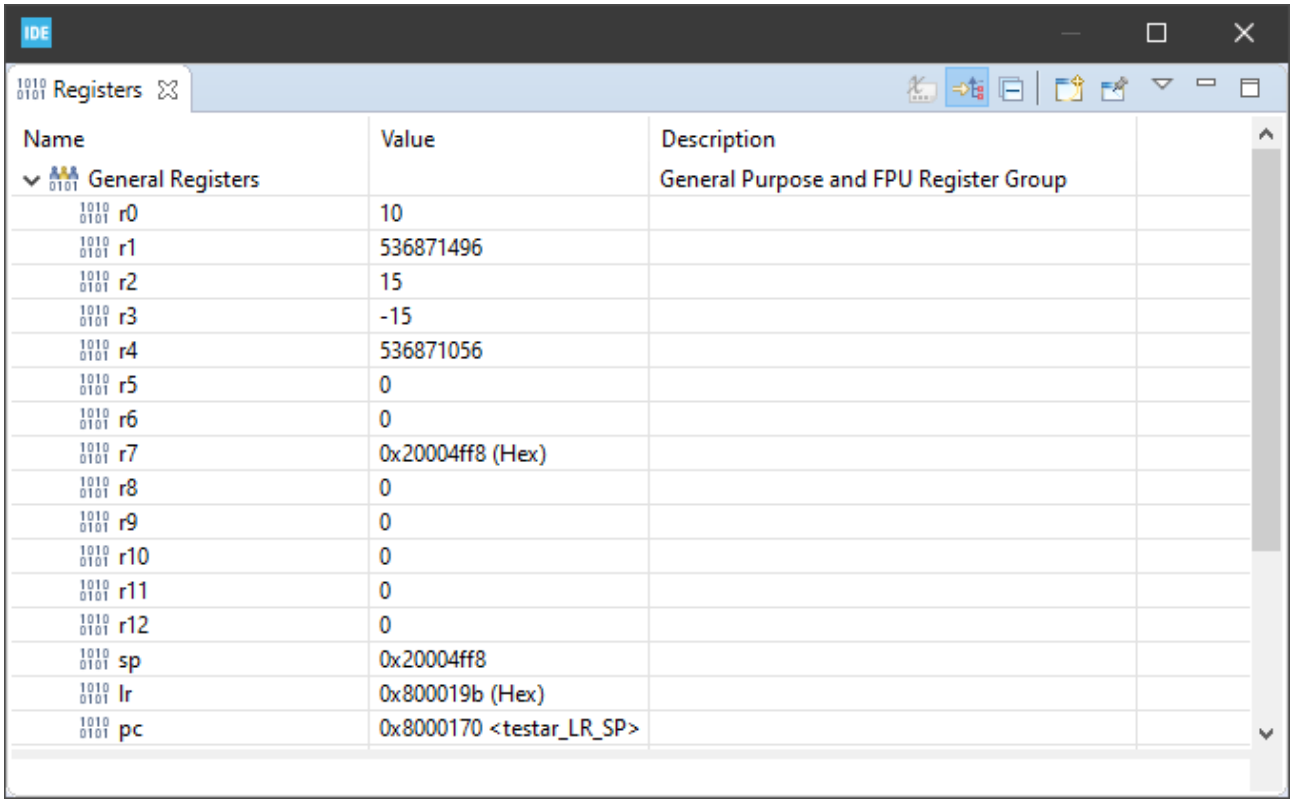

Figura 9 – Monitorando os Registradores

Na janela “Disassembly” a posição de memória de código ‘0x8000170’ está em destaque, ver figura 8, e será a linha a ser executada, apontada pelo Registrador PC, ver figura 9. No início da função “testar_LR_SP” será criado uma variável local, portanto, variáveis locais são alocadas na seção de ‘stack’ da memória de dados SRAM. Esse conceito será detalhado em outro artigo, mas, nesse artigo veremos que realmente o Registrador SP aponta para posições na seção ‘stack’ da SRAM.

Nas três primeiras instruções apontadas pelos endereços ‘0x8000170’, ‘0x8000172’ e ‘0x8000174’ o compilador aloca uma posição na memória de dados SRAM, seção ‘stack’. Nesse momento o Registrador SP contém um valor qualquer. Dê dois passos na linha do programa, clicando duas vezes no botão “Step Into” ou pressionando duas vezes a tecla [F5], conforme ilustrado na figura 10.

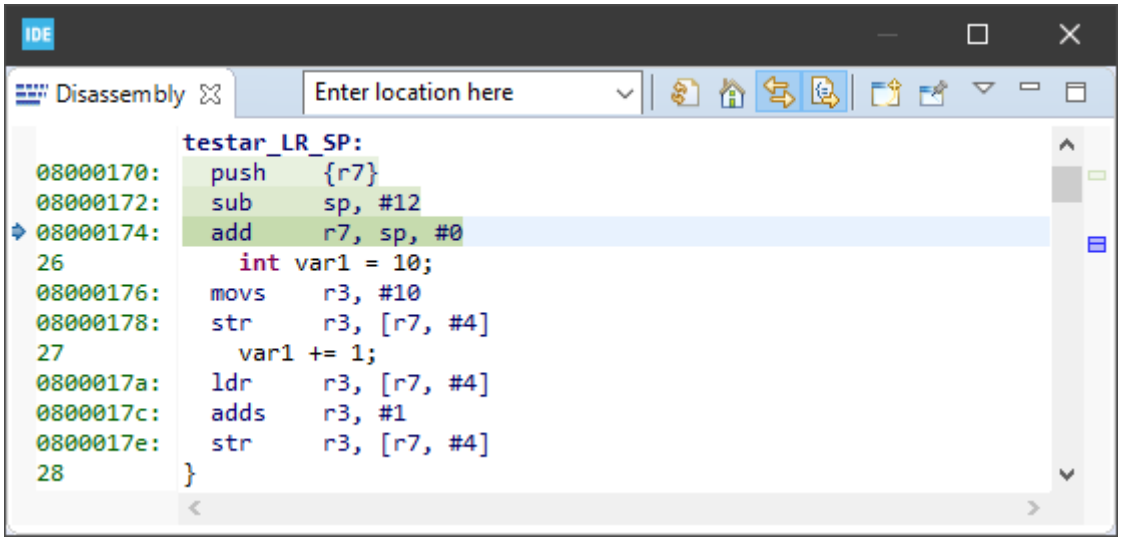

Figura 10 – Variável local na seção ‘stack’ da memória SRAM

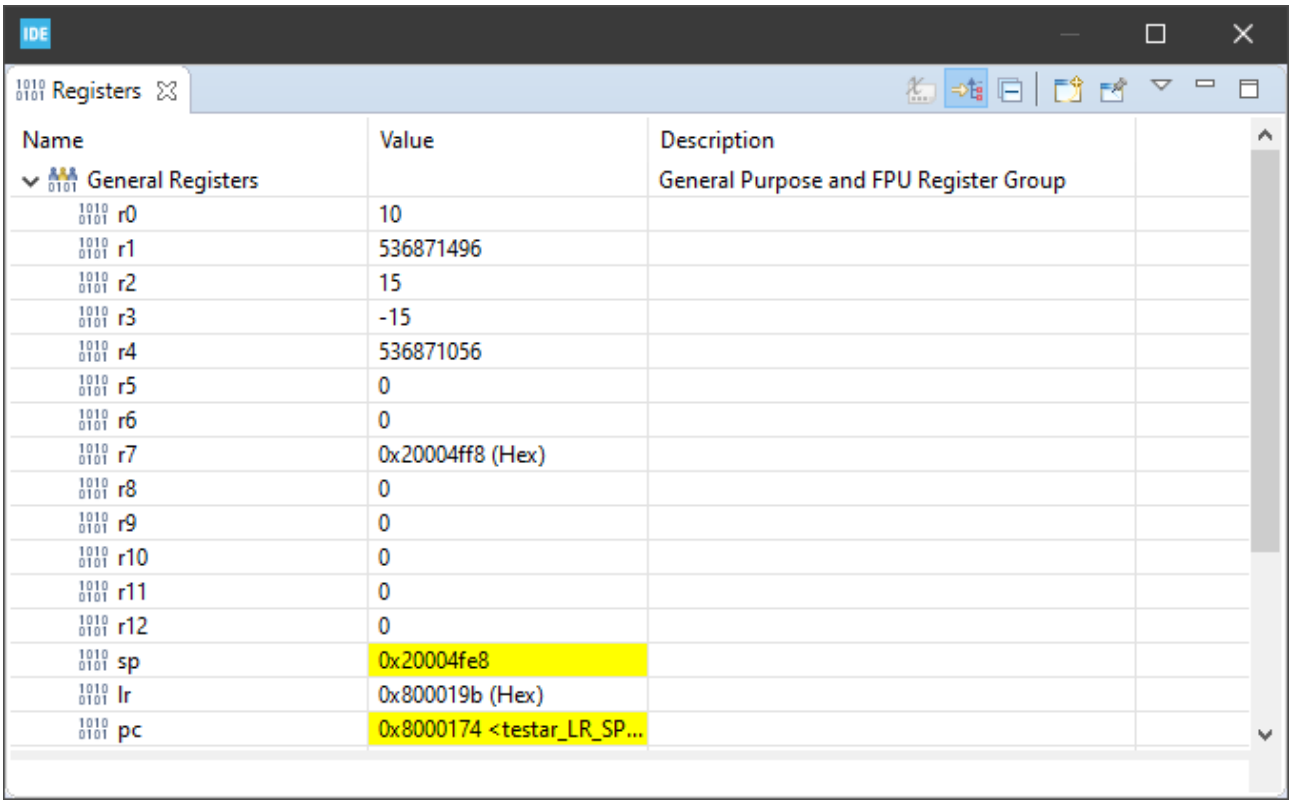

Figura 11 – Registrador SP aponta para variável local

Na figura 11 podemos confirmar que o Registrador SP aponta para uma posição na memória de dados SRAM, conforme já destacado nesse artigo. Para finalizar interrompa o processo de depuração, clicando no botão ‘terminate’ ou pressionando as teclas [CTRL+F2].