

Universidade Federal de Ouro Preto
Instituto de Ciência Exatas e Aplicadas
Departamento de Engenharia Elétrica
Laboratório de Microprocessadores e Microcontroladores – CEA580

Prática 1

Nome

Graziele de Cássia Rodrigues 21.1.8120

Objetivos

Compreender o princípio de funcionamento do assembler e do simulador Arm

Utilizar instruções para transferência de dados entre memória e registradores

Utilizar instruções de soma e subtração

Analisar o comportamento dos flags armazenados no registrador CPSR.

Referências

Mazidi and Naimi “The STM32F103 Arm Microcontroller and Embedded Systems” , Cap. 2 e 3.

Installing the Keil for STM32F10x step by step tutorial

STM32 Assembly Programming in Keil step by step tutorial

Lista de Materiais

Keil IDE, disponível em <http://www.keil.com/>

Keil.STM32F1xx_DFP.2.4.0.pack, disponível em <https://www.keil.com/dd2/pack/>

Atividades

1 – Criar um projeto no Keil e carregar o algoritmo abaixo:

```
EXPORT __main
AREA OUR_PROG, CODE, READONLY
__main
    MOV    R1, #0x12      ; R1 = 0x12
    MOV    R2, #0x25      ; R2 = 0x25
    ADD    R3, R2, R1      ; R3 = R2 + R1
    SUBS   R3, R3, #0x7    ; R3 = R3 - 0x07

HERE    B      HERE      ; stay here forever
END
```

A – Clicar em build para gerar o código assembly.

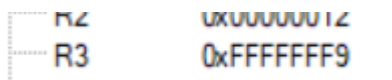
B – Examinar o arquivo map gerado e encontrar o endereço inicial utilizado para armazenar a função __main

O endereço inicial da função main é 0x08000009

__main	0x08000009	Thumb Code	16	main.o(MY_PROG)
--------	------------	------------	----	-----------------

C – Clicar em debug. Examinar o arquivo disassembly gerado e encontrar o código de máquina utilizado para representar a instrução SUBS R3, R3, #0x7

O código de máquina para o R3 é 1FDB

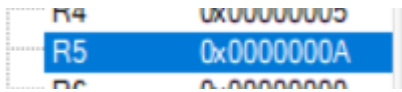


R2	0x00000012
R3	0xFFFFFFFF9

2 – Escrever um programa para carregar os registradores R2, R3 e R4 com 2, 3 e 5, respectivamente. Em seguida, calcular R2+R3+R4.

```
__main PROC
    MOV R2, #0x02
    MOV R3, #0x03
    MOV R4, #0x05
    ADD R5, R2, R3
    ADD R5, R5, R4
HERE B HERE
ENDP
```

END



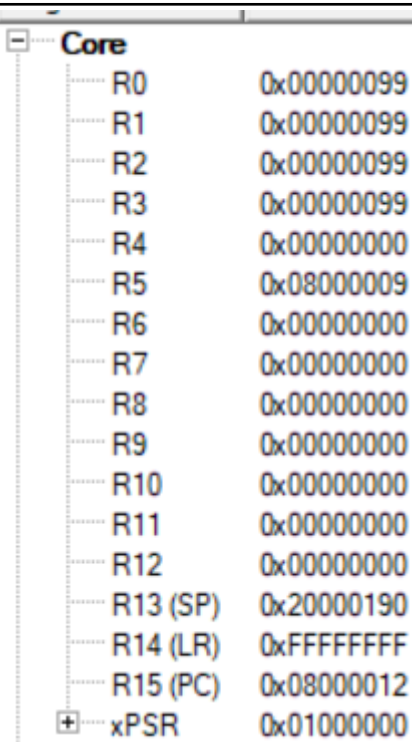
R4	0x00000000
R5	0x0000000A
R6	0x00000000

3 – Escrever um programa para carregar o registrador R2 com o valor 0x99. Em seguida, mover o conteúdo do registrador R2 para os registradores R0, R1 e R3. Utilizar a função simulador e o recurso single-step para examinar as informações dos registradores.

```
__main PROC
    MOV R2,#0x99
    MOV R0, R2
    MOV R1, R2
    MOV R3, R2

HERE B HERE
    ENDP

    END
```



Core	
R0	0x00000099
R1	0x00000099
R2	0x00000099
R3	0x00000099
R4	0x00000000
R5	0x08000009
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000190
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x08000012
xPSR	0x01000000

4 – Escrever um algoritmo assembly para somar os números abaixo.

0x33322292, 0x55566623, 0x9998884B, 0xFF, 0xDDDEEE01

Executar o algoritmo utilizando a opção simulador single-step e analisar o comportamento dos flags C e Z após a execução de cada operação.

```

__main PROC
    LDR R0,=0x33322292
    LDR R1,=0x55566623
    LDR R2,=0x9998884B
    LDR R3,=0xFF
    LDR R4,=0xDDDEEE01
    ADD R5, R0, R1
    ADD R5, R5, R2
    ADD R5, R5, R3
    ADD R5, R5, R4
HERE B HERE

```

1° soma

Core	
R0	0x33322292
R1	0x55566623
R2	0x9998884B
R3	0x000000FF
R4	0xDDDEEE01
R5	0x08000009

2° soma

Register	Value
Core	
R0	0x33322292
R1	0x55566623
R2	0x9998884B
R3	0x000000FF
R4	0xDDDEEE01
R5	0x888888B5

3° soma

Core	
R0	0x33322292
R1	0x55566623
R2	0x9998884B
R3	0x000000FF
R4	0xDDDEEE01
R5	0x22211100

4° soma

Core	
R0	0x33322292
R1	0x55566623
R2	0x9998884B
R3	0x000000FF
R4	0xDDDEEE01
R5	0x222111FF

fim

Core	
R0	0x33322292
R1	0x55566623
R2	0x9998884B
R3	0x000000FF
R4	0xDDDEEE01
R5	0x00000000

5 – Criar um projeto no Keil e carregar o algoritmo abaixo

```

EXPORT __main
AREA MY_PROG, CODE, READONLY
__main
    MOV    R1,#27
    MOV    R2,#15
    SUBS   R3, R1, R2

    MOV    R1,#20
    MOV    R2,#15
    SUBS   R3, R1, R2

    MOV    R1,#95
    MOV    R2,#95
    SUBS   R3, R1, R2

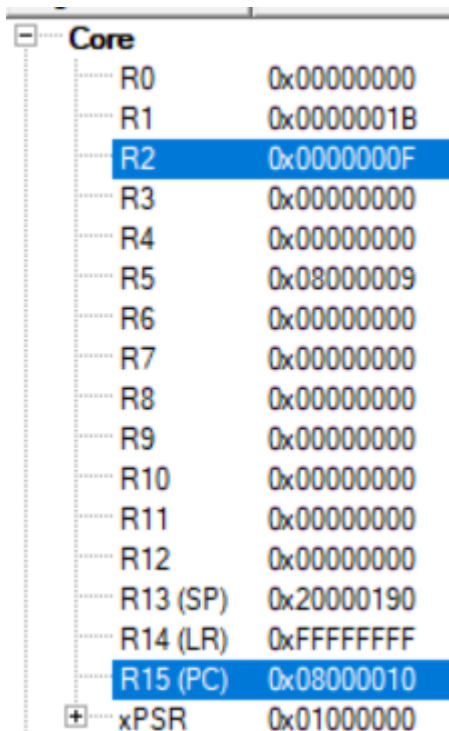
    MOV    R1,#50
    MOV    R2,#70
    SUBS   R3, R1, R2

    H      B      H      ;stay here forever

```

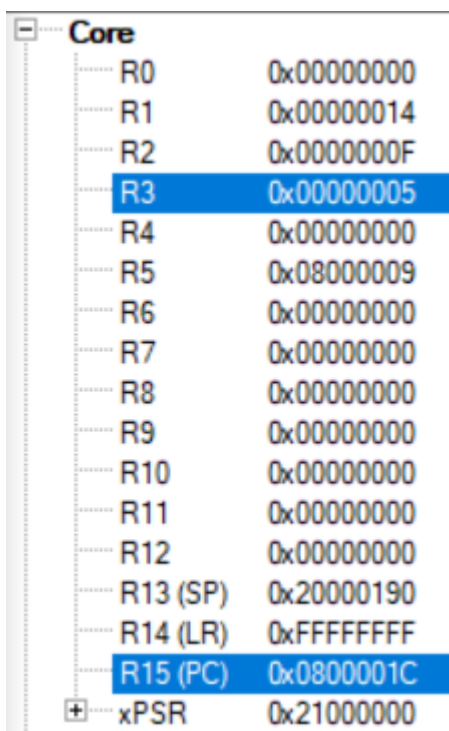
Executar o algoritmo utilizando a opção simulador single-step. Analisar o conteúdo dos registradores e o comportamento dos flags C e Z após a execução de cada operação.

Conteúdo R3 após primeira sub



Core	
R0	0x00000000
R1	0x0000001B
R2	0x0000000F
R3	0x00000000
R4	0x00000000
R5	0x08000009
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000190
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x08000010
xPSR	0x01000000

Conteúdo R3 após segundo sub



Core	
R0	0x00000000
R1	0x00000014
R2	0x0000000F
R3	0x00000005
R4	0x00000000
R5	0x08000009
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000190
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x0800001C
xPSR	0x21000000

Conteúdo após terceiro sub

Core	
R0	0x00000000
R1	0x0000005F
R2	0x0000005F
R3	0x00000000
R4	0x00000000
R5	0x08000009
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000190
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x08000026
xPSR	0x61000000

Conteúdo após quarto sub

Register	Value
Core	
R0	0x00000000
R1	0x00000032
R2	0x00000046
R3	0xFFFFFEC
R4	0x00000000
R5	0x08000009
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000190
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x08000030
xPSR	0x81000000

As flags estão mantidas em 0, por não ter sido habilitado.

