ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Graduação em Engenharia Computação

PCS3732 - Laboratório de Processadores

Professor Jorge Kinoshita

Grupo 10 - Planejamento E6

Arthur Pires da Fonseca NUSP: 10773096

Sumário

6.1.2 B. Responda

1- O que há de errado com as seguintes instruções?

a) STMIA r5!, {r5, r4, r9}

Os registradores entre chaves não estão em ordem crescente.

O registrador de writeback está na lista, portanto tem que ser o de menor índice entre as chaves, o que não é o caso.

b) LDMDA r2, {}

Não há nenhum registrador especificado, onde se deva carregar o que será tirado da memória quando a instrução for executada.

c) STMDB r15!, [r0-r3, r4, lr}

O registrador r15 (program counter) não pode servir como registrador-base.

2- Se o registrador r6 possui 0x8000 (como ponteiro para a memória); após executar "LDMIA r6,{r7,r4,r0,lr}" o que fica em r0, r4, r7 e em lr?

Os registradores ficam com os seguintes valores:

r0 = palavra guardada em 0x8000

r4 = palavra guardada em 0x8004

r7 = palavra guardada em 0x8008

lr = palavra guardada em 0x800c

3- Assuma que a memória e registradores estejam:

0x8010	**0x1**
0x800C	**0xfeeddeaf**
0x8008	**0x00008888**
0x8004	**0x12340000**
0x8000	**0xbabe0000**

r0 = 0x13; r1 = 0xffffffff; r2 = 0xeeeeeeee; r3 = 0x8000

Descreva a memória e conteúdos dos registradores após a instrução:

LDMIA r3!, {r0,r1,r2}

A memória permanecerá como estava antes da instrução e os registradores estarão assim:

- r0 = 0xbabe0000
- r1 = 0x12340000
- r2 = 0x00008888
- r3 = 0x800c

4- Suponha que a pilha esteja como o diagrama abaixo. Que instrução seria necessária para sair do estado original e ir para o estado a), depois b) e depois c)?

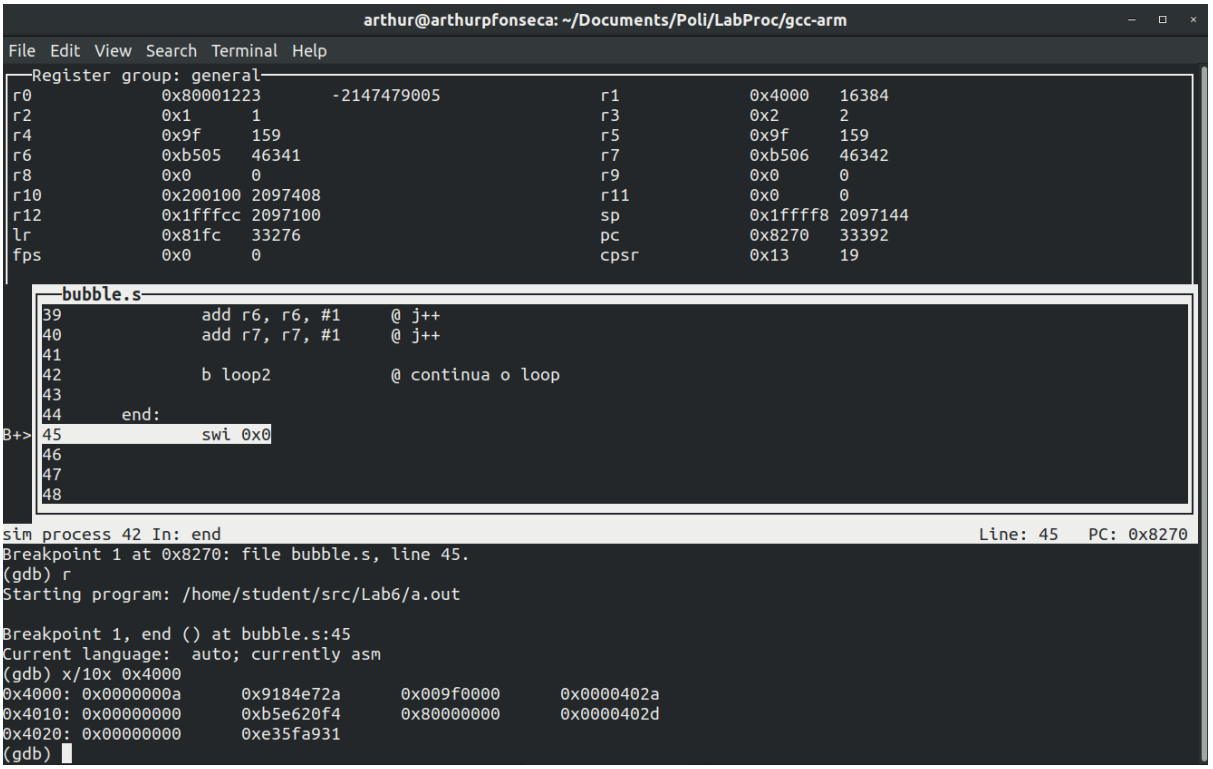
Endereço	Original	A	B	C
0x8010	0x1	0x1	0x1	0x1
0x800C	0xfeeddeaf	0xfeeddeaf	0xfeeddeaf	0xfeeddeaf
0x8008		0xbabe2222	0xbabe22222	
0x8004			0x12340000	
0x8000				

Supondo, por exemplo, que r0 = 0xbabe2222, r1 = 0x12340000 e r13 = 0x800c:

```
STMFA r13!, {r0}
STMFA r13!, {r1}
LDMFA r13!, {r0, r1}
```

6.1.3 C. Apresente o código Assembly rodando com printscreen de: 6.5.2 Bubble sorting

```
@ Exercicio 6.5.2 do livro
@ Para debugar este codigo:
@ gcc bubble.s && gdb a.out
        .text
        .globl   main
main:
        ldr r0, =0xa       @ len = 10
        ldr r1, =0x4000  @ address list = *a

        str r0, [r1]         @ a[-1] = len

        mov r2, #0         @ indice do bubble = i
        add r3, r2, #1    @ i + 1

loop1:
        cmp r3, r0
        bge end                      @ se i >= n : cheogu ao fim do vetor

        ldrb r4, [r0], r2  @ operando1 = a[i]
        ldrb r5, [r0], r3  @ operando2 = a[i + 1]

        mov r6, r3           @ j = i
        add r7, r6, #1      @ j + 1


        add r2, r2, #1      @ i++
        add r3, r3, #1      @ i++

loop2:
        cmp r7, r0
        bge loop1           @ se j >= len, volta para o loop maior

        cmp r4, r5
        blt loop2            @ se op1 < op2 => não precisa fazer mais nada

        strb r5, [r0], r6  @ a[j] = operando2
        strb r4, [r0], r7  @ a[j + 1] = operando1

        add r6, r6, #1      @ j++
        add r7, r7, #1      @ j++

        b loop2                   @ continua o loop

end:
        swi 0x0
```



Print do fim da execução do código acima.