ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DISCIPLINA: LABORATÓRIO DE PROCESSADORES- PCS3732
1° QUADRIMESTRE/2021

Aula 4

27 de Maio de 2021

GRUPO 10

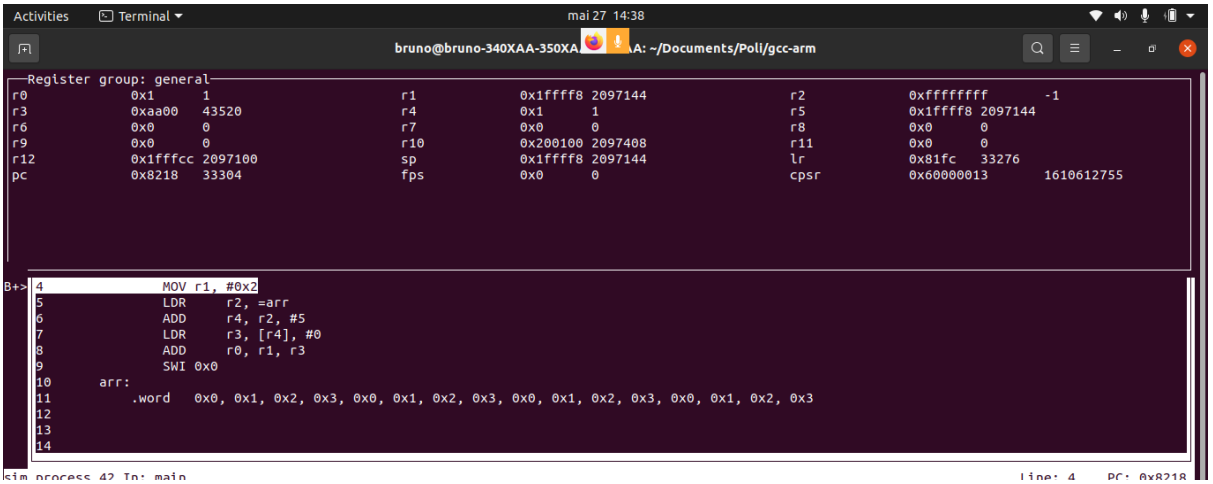
Arthur Pires da Fonseca	NUSP: 10773096
Bruno José Móvio	NUSP: 10336852
Iago Soriano Roque Monteiro	NUSP: 8572921

Sumário

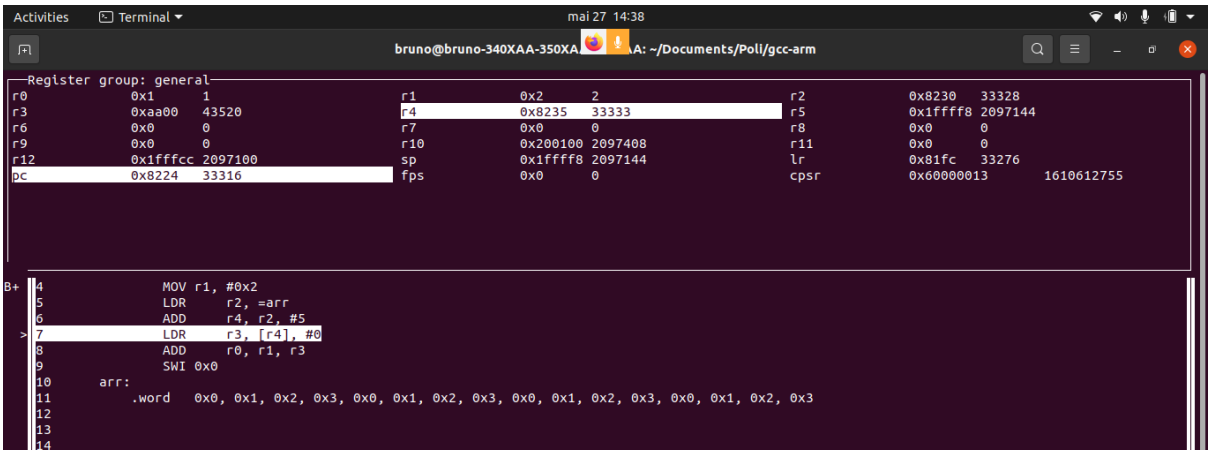
Exercício 4.5.1 - Assignments with operands in memory

Pós fixado

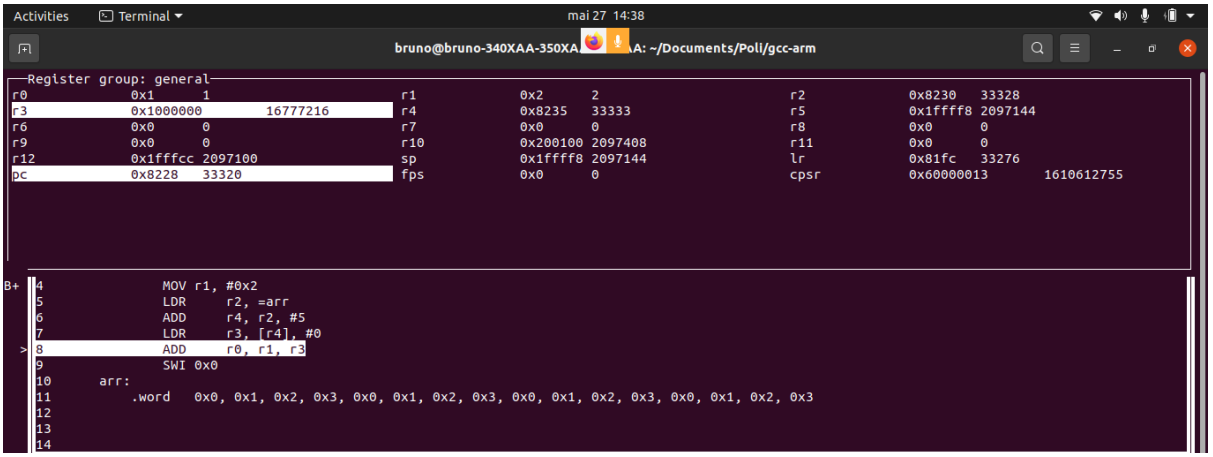
- Início do processo



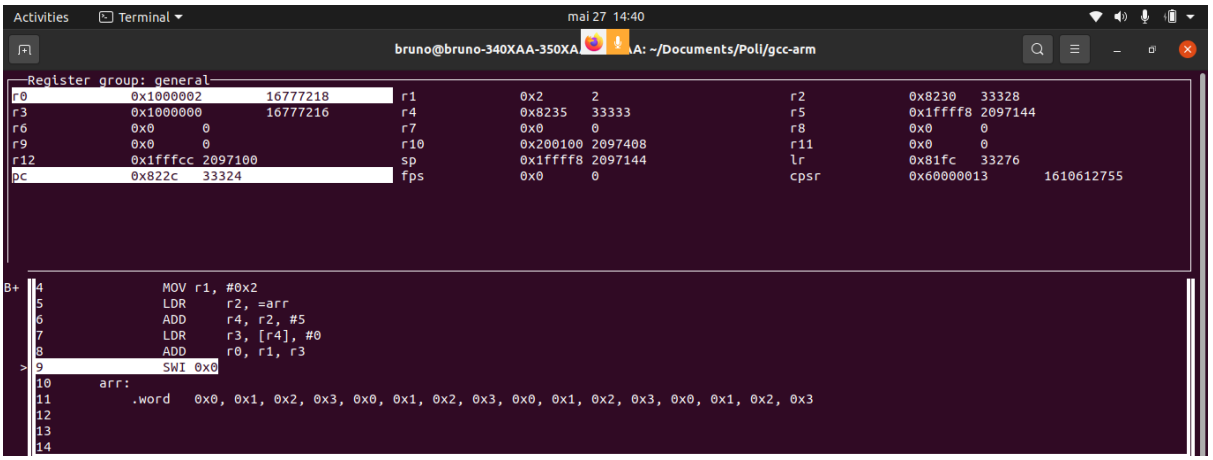
- Antes do load



- Após o load apenas o valor de r3 sofre alteração. O registrador r4 permanece o mesmo.

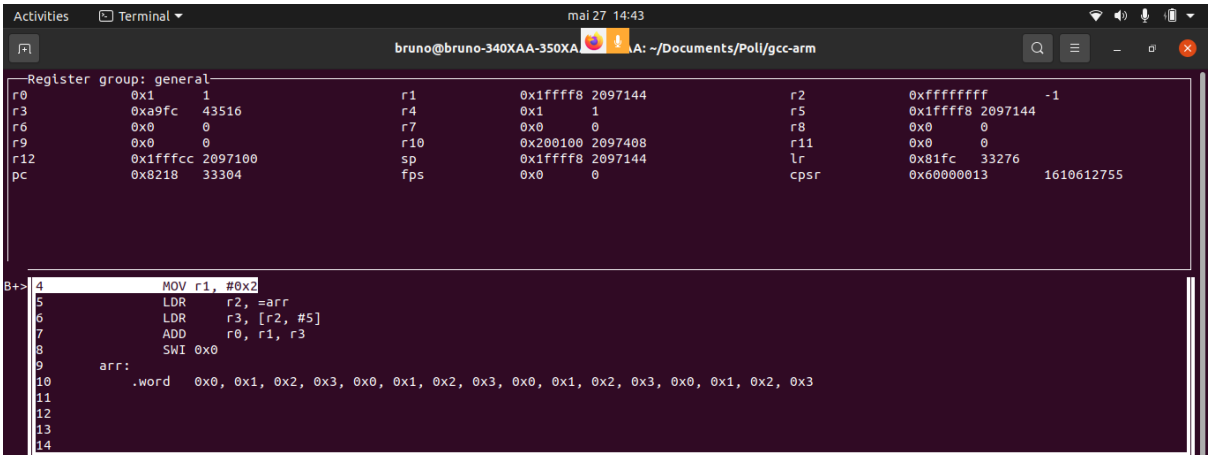


- Ao final, os registradores r1 e r2 são somados.

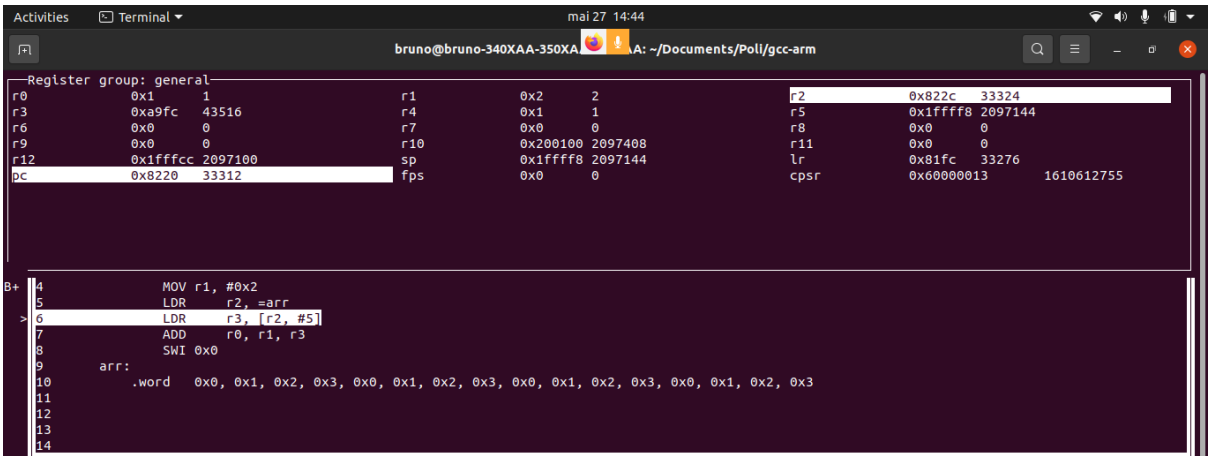


Pré fixado

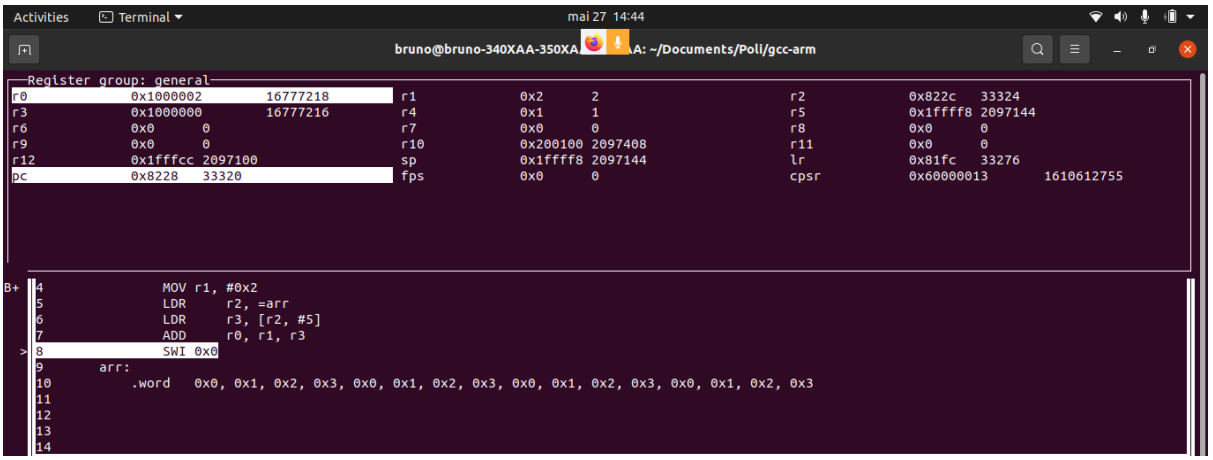
- Inîcio do processo



- Antes do load



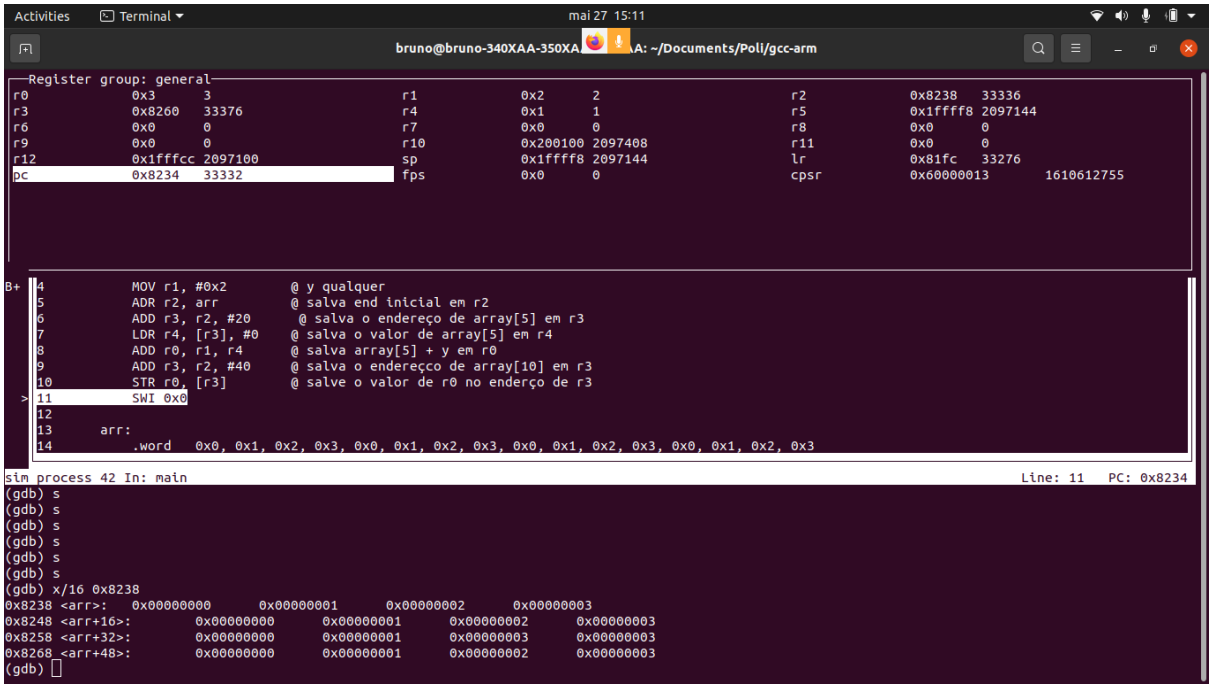
- Após o load e após a soma



Exercício 4.5.2 - Loads and stores

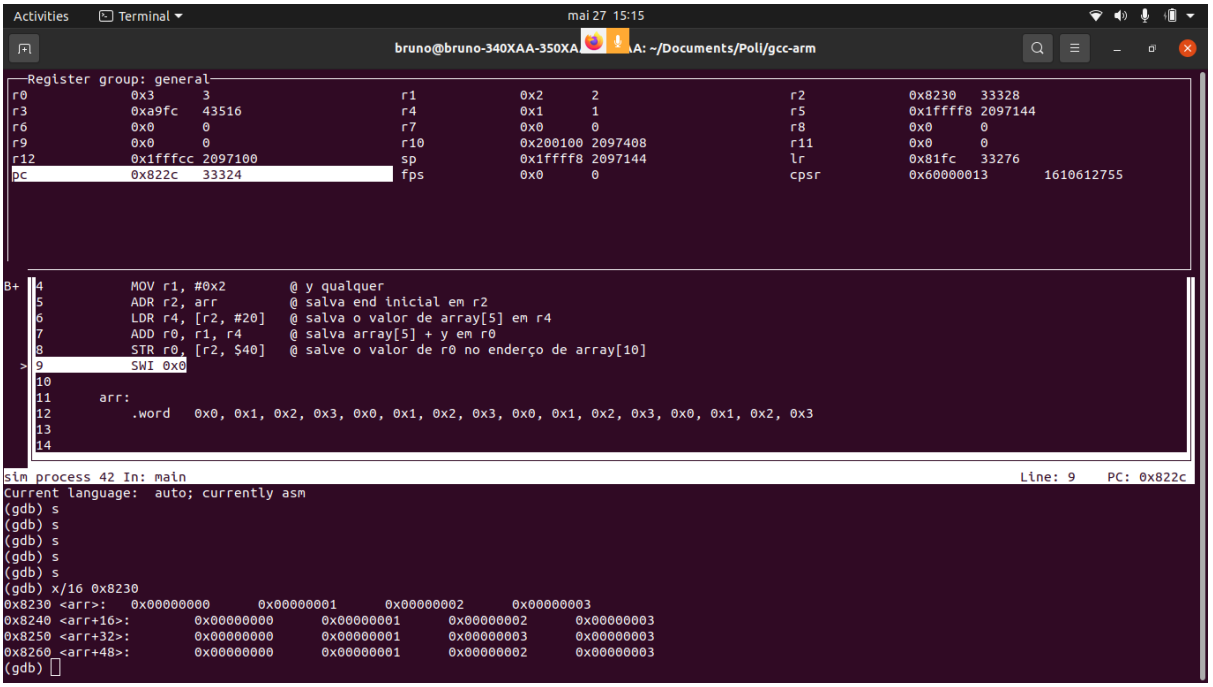
Pós fixado

- Resultado após execução, pode-se ver os valores atualizados do array na parte debaixo da imagem.



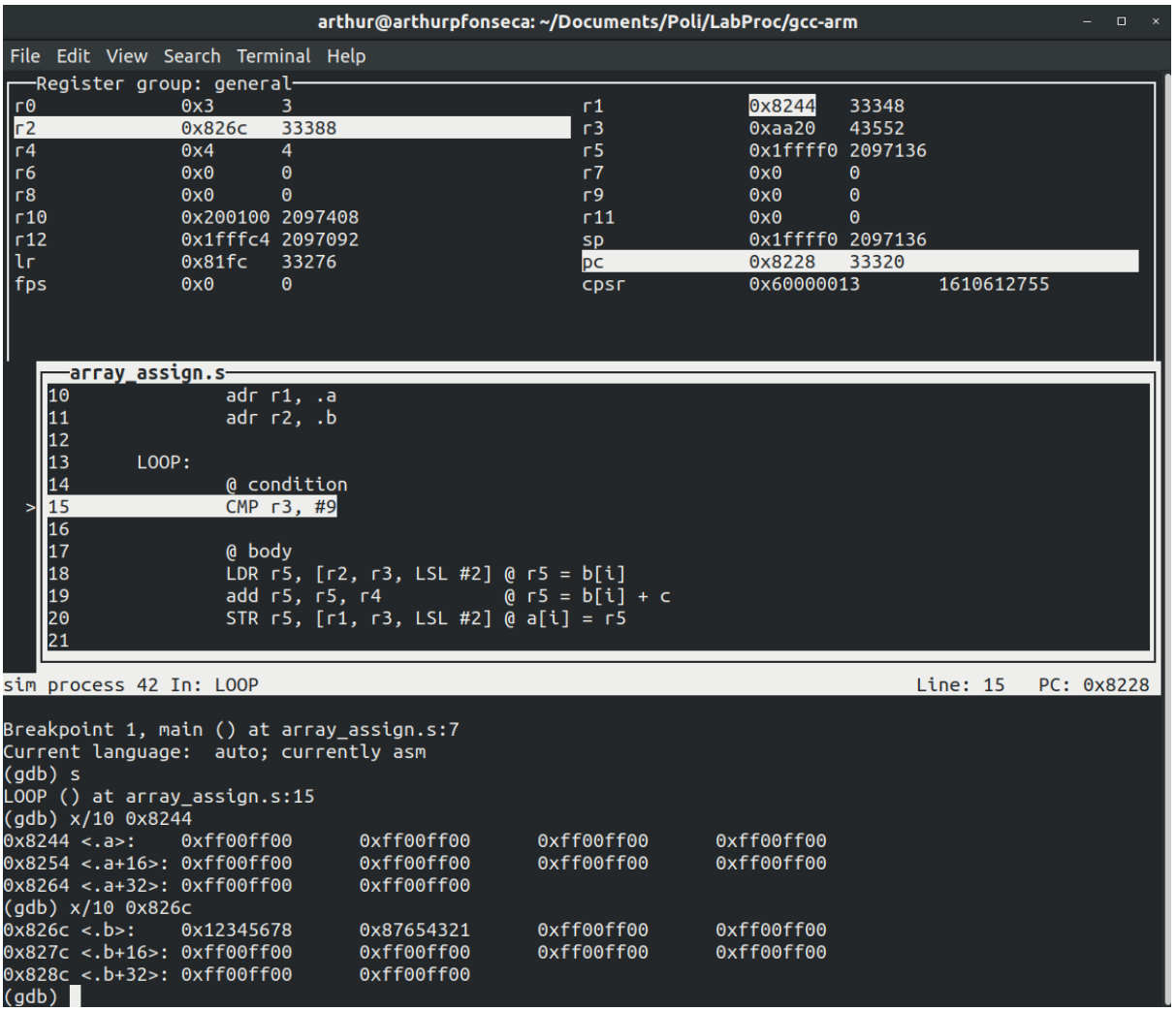
Pré fixado

- Resultado após execução, pode-se ver os valores atualizados do array na parte debaixo da imagem.

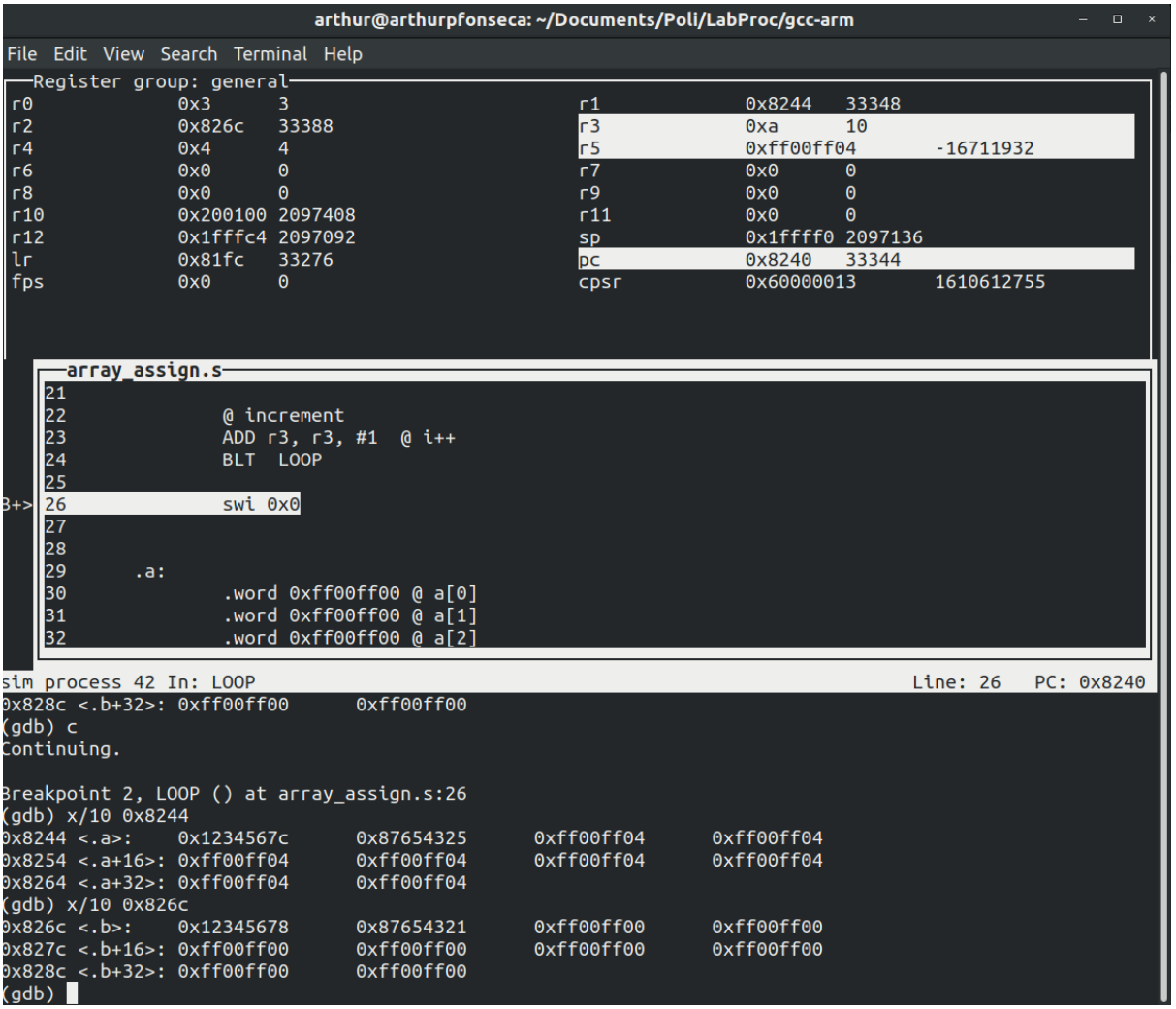


Exercício 4.5.3 - Array Assignment

a[]={0xff00ff00, … } e b[]={0x12345678, 0x87654321, 0xff00ff00, …}, em que ambas as arrays tem 10 posições. c=4.



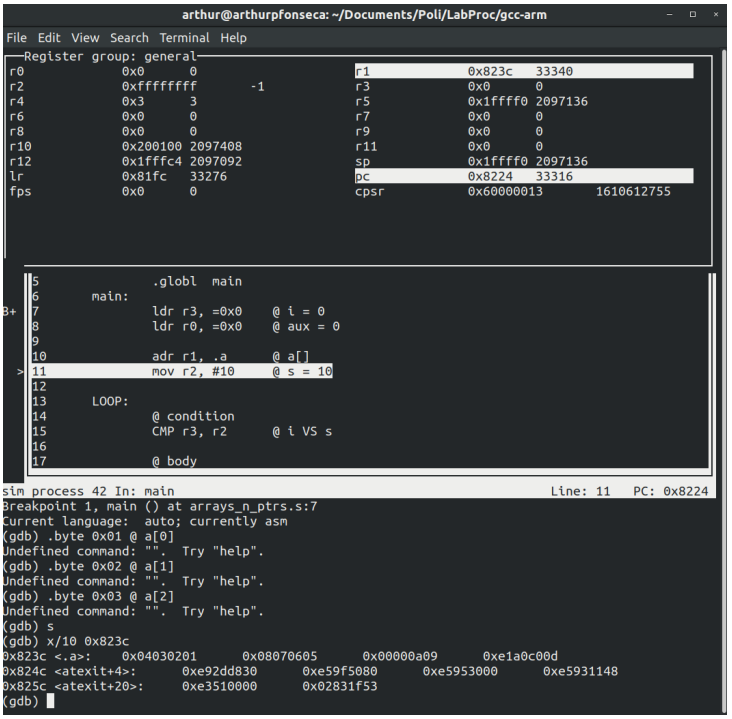
Início da execução.



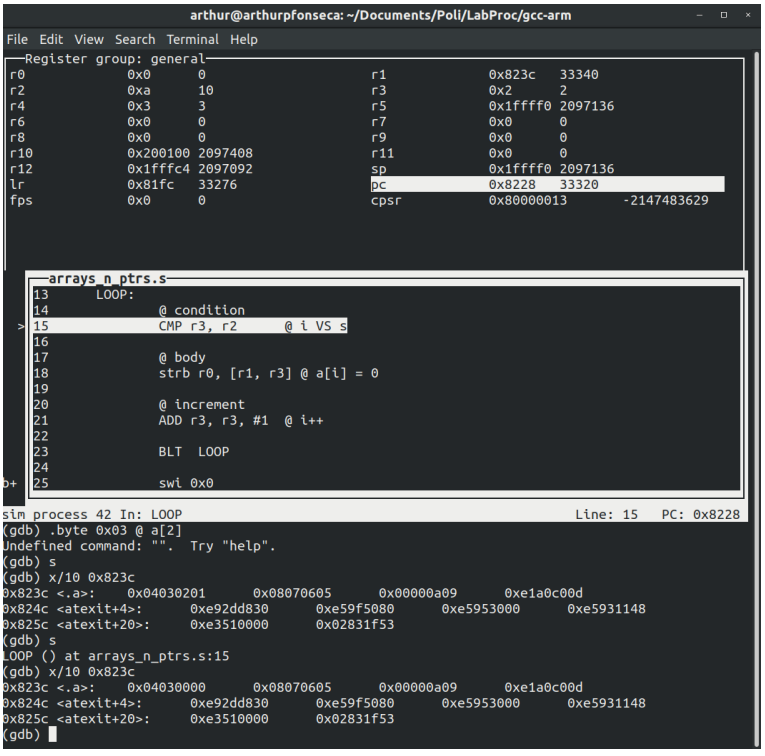
Fim da execução. a[i]=b[i]+4, para n entre 0 e 9.

Exercício 4.5.4 - Arrays and pointers

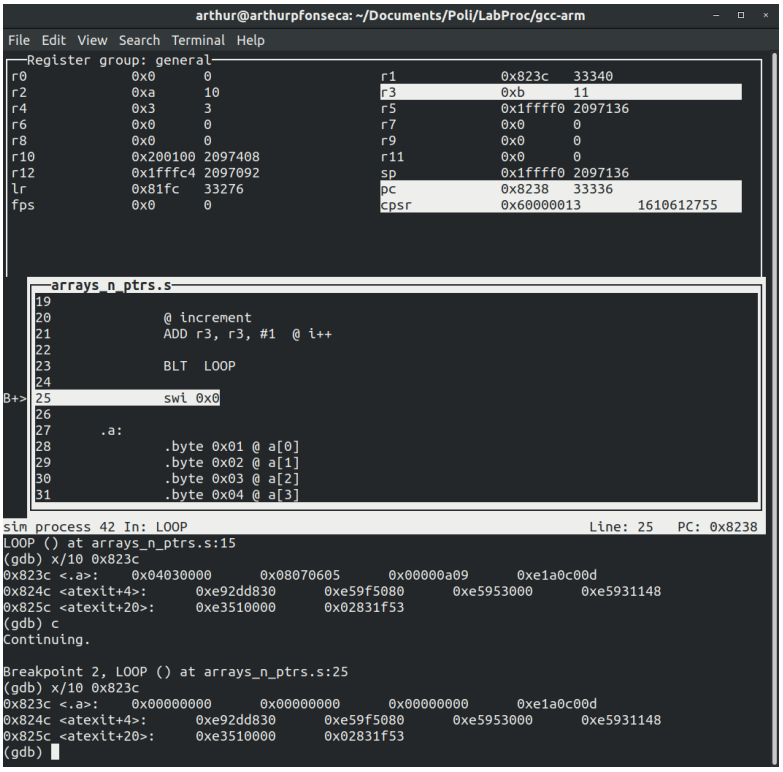
Índice



Início da execução, vê-se a array na memória.

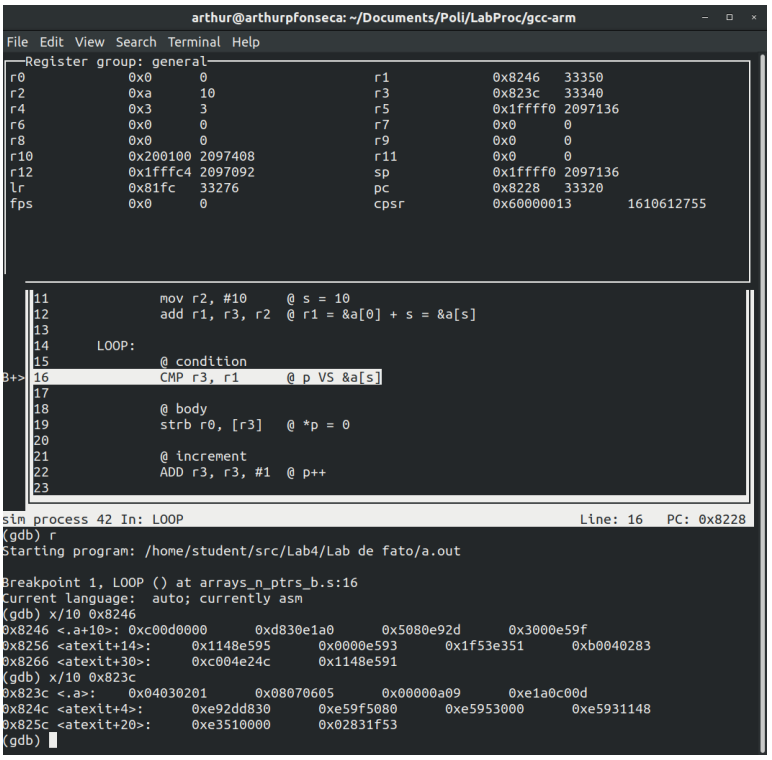


No meio da execução, alguns bytes são = 0.

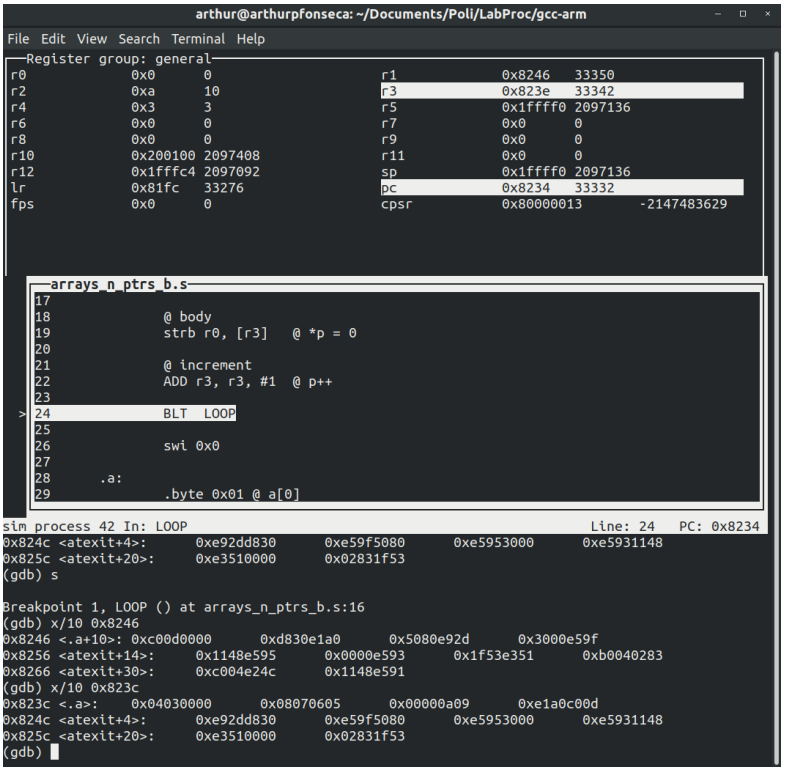


Ao final da execução, toda a array = 0.

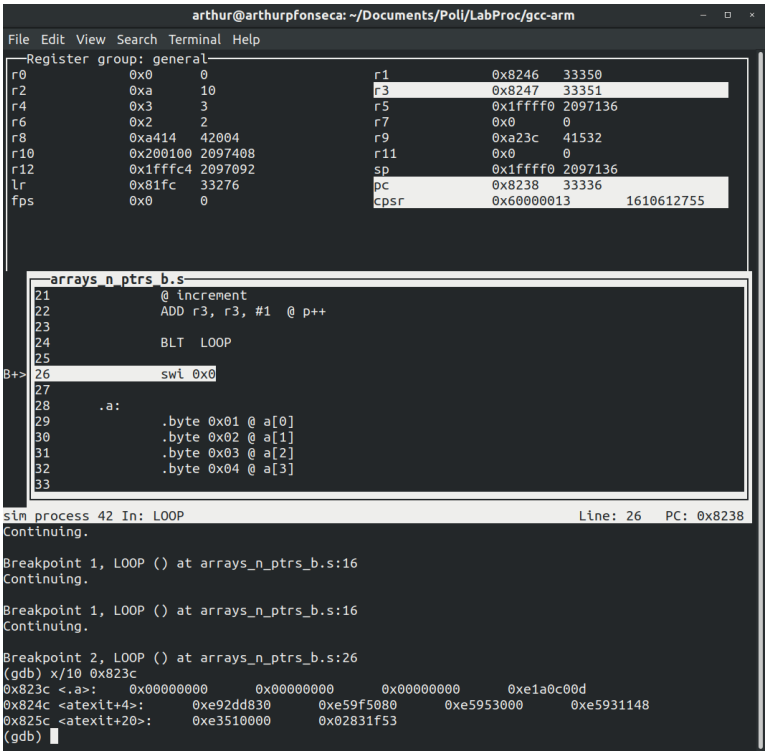
Pointers



Início da execução, vê-se a array na memória.

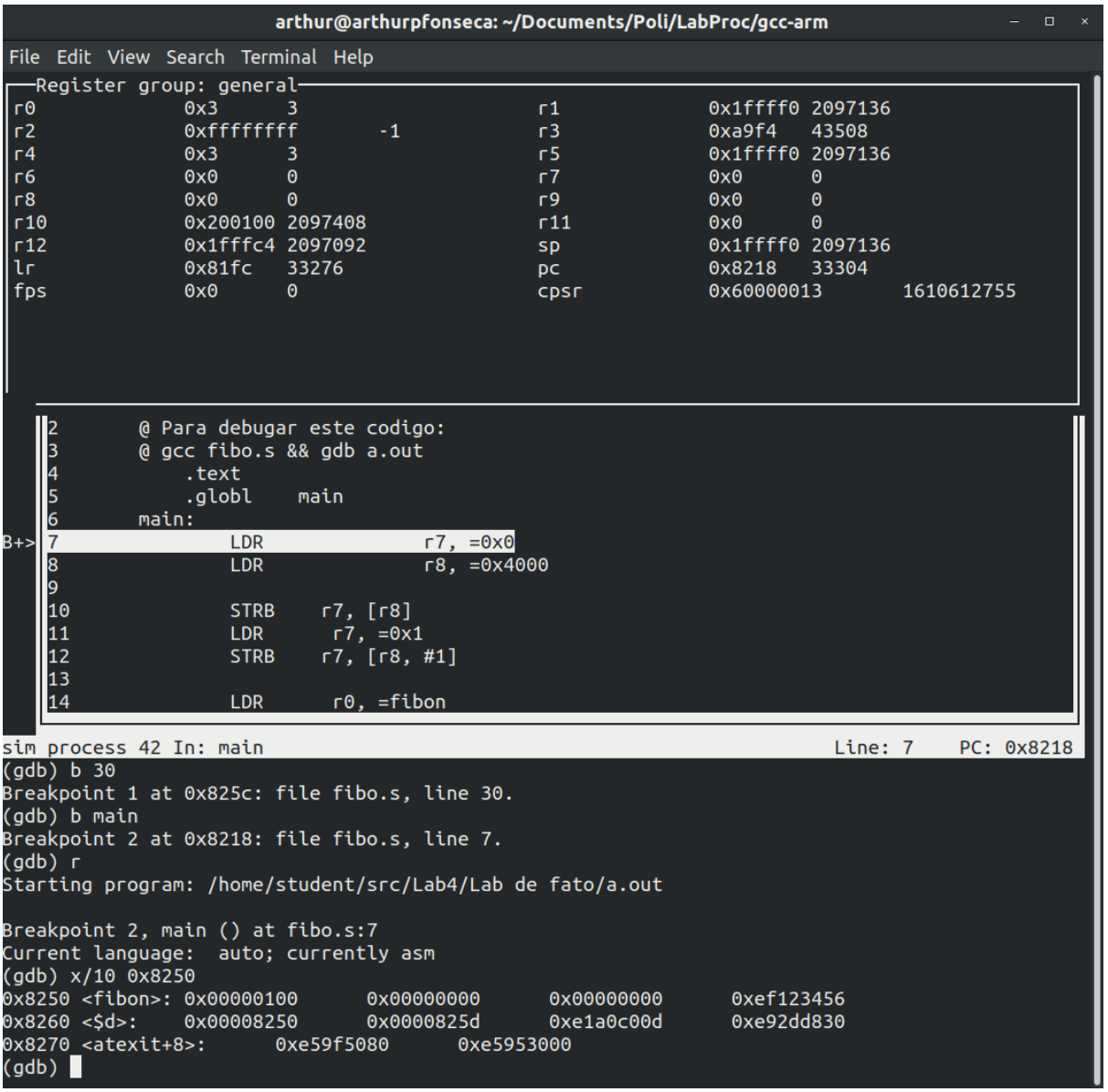


No meio da execução, alguns bytes são = 0.

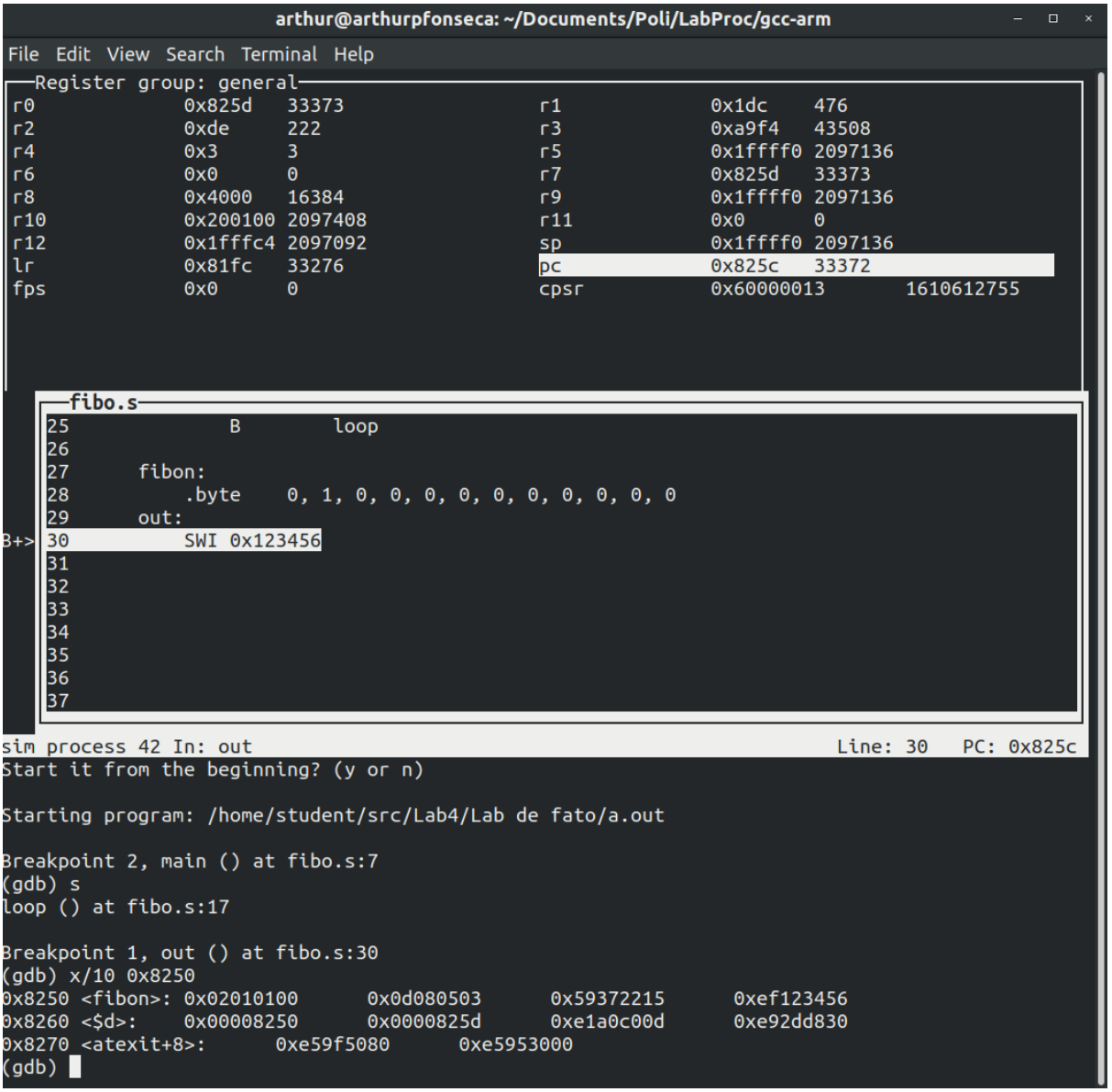


Ao final da execução, toda a array = 0.

Exercício 4.5.5 - The Fibonacci sequence

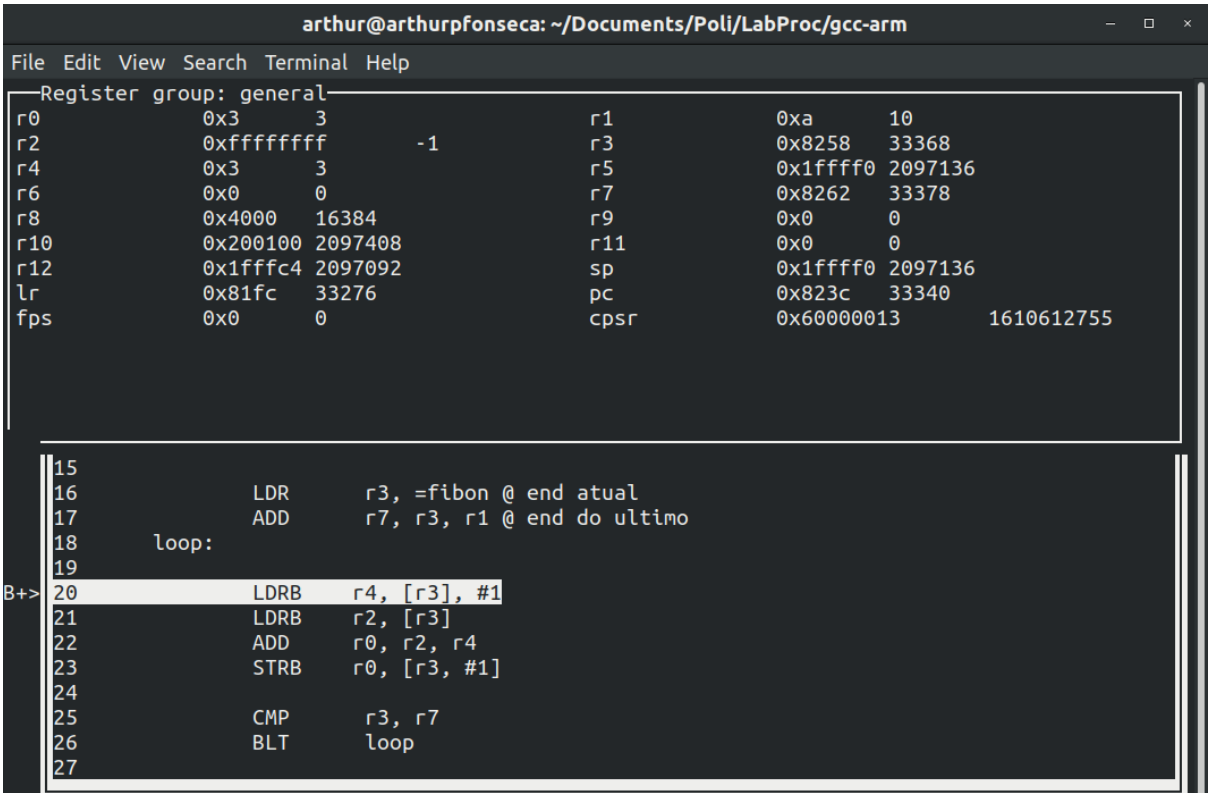


Início da execução

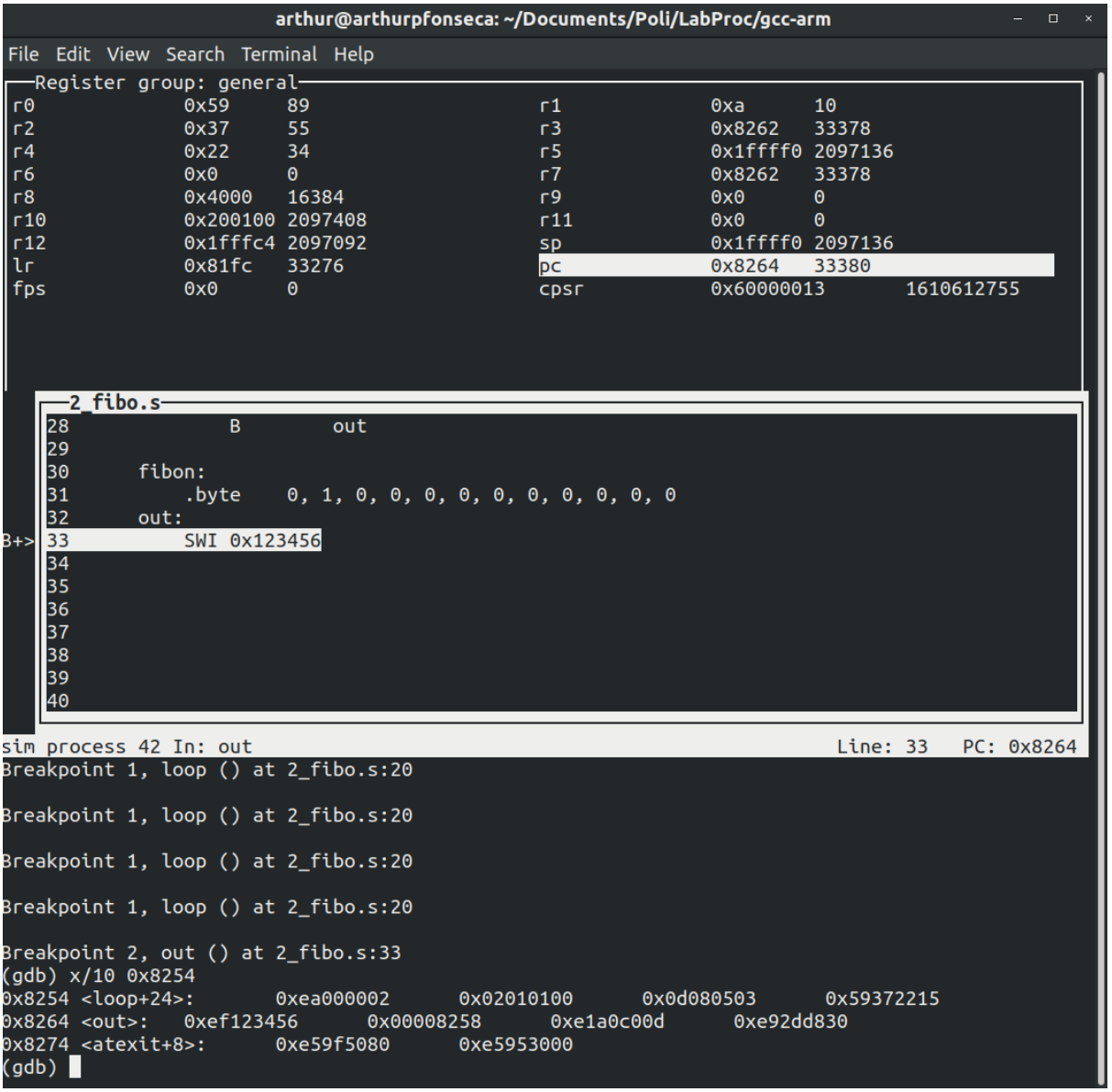


Após a execução, o array armazena os 12 primeiros elementos da sequência de Fibonnacci.

Exercício 4.5.6 - The nth Fibonacci number



Início da execução



Após a execução, vê-se os resultados a partir da segunda palavra (na memória) da imagem acima.

Apêndice

1. Exercício 4.5.1

Pré indexado

```
@ Exercicio 4.5.1 do livro
@ Para debugar este codigo:
@ gcc assig_pre.s && gdb a.out

   .text
   .globl         main
main:
   MOV r1, #0x2
   LDR    r2, =arr
   LDR r3, [r2, #5]
   ADD    r0, r1, r3
   SWI 0x0
arr:
         .word   0x0, 0x1, 0x2, 0x3, 0x0, 0x1, 0x2, 0x3, 0x0, 0x1, 0x2, 0x3, 0x0,
0x1, 0x2, 0x3, 0x0, 0x1, 0x2, 0x3, 0x0, 0x1, 0x2, 0x3


Pós fixado
```

Pós indexado

```
@ Exercicio 4.5.1 do livro
@ Para debugar este codigo:
@ gcc assig_pos.s && gdb a.out

   .text
   .globl         main
main:
   MOV r1, #0x2
   LDR    r2, =arr
   ADD    r4, r2, #5
   LDR    r3, [r4], #0
   ADD    r0, r1, r3
   SWI 0x0
arr:
         .word   0x0, 0x1, 0x2, 0x3, 0x0, 0x1, 0x2, 0x3, 0x0, 0x1, 0x2, 0x3, 0x0,
0x1, 0x2, 0x3
```

2. Exercício 4.5.2

Pós indexado

```
@ Exercicio 4.5.1 do livro
@ Para debugar este codigo:
@ gcc load_store_pos.s && gdb a.out

    .text
   .globl        main
main:
        MOV r1, #0x2          @ y qualquer
        ADR r2, arr           @ salva end inicial em r2
        ADD r3, r2, #20       @ salva o endereço de array[5] em r3
        LDR r4, [r3], #0      @ salva o valor de array[5] em r4
        ADD r0, r1, r4        @ salva array[5] + y em r0
        ADD r3, r2, #40       @ salva o endereçco de array[10] em r3
        STR r0, [r3]    @ salve o valor de r0 no enderço de r3
        SWI 0x0

arr:
        .word   0x0, 0x1, 0x2, 0x3, 0x0, 0x1, 0x2, 0x3, 0x0, 0x1, 0x2, 0x3, 0x0,
0x1, 0x2, 0x3
```

Pré indexado

```
@ Exercicio 4.5.1 do livro
@ Para debugar este codigo:
@ gcc load_store_pre.s && gdb a.out

   .text
   .globl        main
main:
        MOV r1, #0x2          @ y qualquer
        ADR r2, arr           @ salva end inicial em r2
        LDR r4, [r2, #20]   @ salva o valor de array[5] em r4
        ADD r0, r1, r4        @ salva array[5] + y em r0
        STR r0, [r2, $40]   @ salve o valor de r0 no enderço de array[10]
        SWI 0x0

arr:
        .word   0x0, 0x1, 0x2, 0x3, 0x0, 0x1, 0x2, 0x3, 0x0, 0x1, 0x2, 0x3, 0x0,
0x1, 0x2, 0x3
```

3. Exercício 4.5.3

```
@ Exercicio 4.5.3 do livro
@ Para debugar este codigo:
@ gcc array_assign.s && gdb a.out

        .text
        .globl  main
main:
        ldr r3, =0x0 @ i = 0
        ldr r4, =0x4 @ c = 4

        adr r1, .a
        adr r2, .b

LOOP:
        @ condition
        CMP r3, #10

        @ body
        LDR r5, [r2, r3, LSL #2] @ r5 = b[i]
        add r5, r5, r4              @ r5 = b[i] + c
        STR r5, [r1, r3, LSL #2] @ a[i] = r5

        @ increment
        ADD r3, r3, #1          @ i++

        BLT  LOOP

        swi 0x0


.a:
        .word 0xff00ff00 @ a[0]
        .word 0xff00ff00 @ a[1]
        .word 0xff00ff00 @ a[2]
        .word 0xff00ff00 @ a[3]
        .word 0xff00ff00 @ a[4]
        .word 0xff00ff00 @ a[5]
        .word 0xff00ff00 @ a[6]
        .word 0xff00ff00 @ a[7]
        .word 0xff00ff00 @ a[8]
        .word 0xff00ff00 @ a[9]

.b:
        .word 0x12345678 @ b[0]
        .word 0x87654321 @ b[1]
        .word 0xff00ff00 @ b[2]
        .word 0xff00ff00 @ b[3]
        .word 0xff00ff00 @ b[4]
        .word 0xff00ff00 @ b[5]
        .word 0xff00ff00 @ b[6]
        .word 0xff00ff00 @ b[7]
        .word 0xff00ff00 @ b[8]
        .word 0xff00ff00 @ b[9]
```

4. Exercício 4.5.4

4.5.4.A

```
@ Exercicio 4.5.4.A do livro
@ Para debugar este codigo:
@ gcc arrays_n_ptrs.s && gdb a.out

        .text
        .globl  main
main:
        ldr r3, =0x0    @ i = 0
        ldr r0, =0x0    @ aux = 0

        adr r1, .a      @ a[]
        mov r2, #10     @ s = 10

LOOP:
        @ condition
        CMP r3, r2      @ i VS s

        @ body
        strb r0, [r1, r3] @ a[i] = 0

        @ increment
        ADD r3, r3, #1          @ i++

        BLT  LOOP

        swi 0x0

.a:
        .byte 0x01 @ a[0]
        .byte 0x02 @ a[1]
        .byte 0x03 @ a[2]
        .byte 0x04 @ a[3]

        .byte 0x05 @ a[4]
        .byte 0x06 @ a[5]
        .byte 0x07 @ a[6]
        .byte 0x08 @ a[7]

        .byte 0x09 @ a[8]
        .byte 0x0a @ a[9]

        .align 1
```

4.5.4.B

```
@ Exercicio 4.5.4.B do livro
```

```
@ Para debugar este codigo:
@ gcc arrays_n_ptrs_b.s && gdb a.out

        .text
        .globl  main
main:
        adr r3, .a      @ p = &a[0]
        ldr r0, =0x0    @ aux = 0

        @adr r1, .a     @ a[]
        mov r2, #10     @ s = 10
        add r1, r3, r2  @ r1 = &a[0] + s = &a[s]

LOOP:
        @ condition
        CMP r3, r1      @ p VS &a[s]

        @ body
        strb r0, [r3]   @ *p = 0

        @ increment
        ADD r3, r3, #1          @ p++

        BLT  LOOP

        swi 0x0

.a:
        .byte 0x01 @ a[0]
        .byte 0x02 @ a[1]
        .byte 0x03 @ a[2]
        .byte 0x04 @ a[3]

        .byte 0x05 @ a[4]
        .byte 0x06 @ a[5]
        .byte 0x07 @ a[6]
        .byte 0x08 @ a[7]

        .byte 0x09 @ a[8]
        .byte 0x0a @ a[9]

        .align 1
```

5. Exercício 4.5.5

```
@ Exercicio 4.5.5 do livro
@ Para debugar este codigo:
@ gcc fibo.s && gdb a.out

   .text
   .globl    main
```

```
main:
        LDR             r7, =0x0
        LDR             r8, =0x4000

        STRB    r7, [r8]
        LDR     r7, =0x1
        STRB    r7, [r8, #1]

        LDR     r0, =fibon
        LDR     r7, =fibon + 13
loop:
        CMP     r7, r0
        BLE     out

        LDRB  r1, [r0], #1
        LDRB  r2, [r0]
        ADD   r1, r2, r1
        STRB  r1, [r0, #1]

        B       loop
out:
   SWI 0x123456

fibon:
   .byte    0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

6. Exercício 4.5.6

```
@ Exercicio 4.5.6 do livro
@ Para debugar este codigo:
@ gcc 2_fibo.s && gdb a.out

   .text
   .globl    main
main:
        LDR r7, =0x0
        LDR r8, =0x4000
        MOV r1, #11
        SUB r1, r1, #1 @ Tira 1 pro indice ficar certo

        STRB    r7, [r8]
        LDR    r7, =0x1
        STRB    r7, [r8, #1]

        LDR     r3, =fibon @ end atual
        ADD     r7, r3, r1 @ end do ultimo
loop:

        LDRB  r4, [r3], #1
        LDRB  r2, [r3]
        ADD   r0, r2, r4
```

```
        STRB  r0, [r3, #1]

        CMP     r3, r7
        BLT     loop

        B       out
out:
   SWI 0x123456

fibon:
   .byte    0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```