

Laboratório de Microprocessadores



Antônio Pinheiro, 9004355

Lucas Cupertino, 11257543

Otávio Freitas, 11261249

Prof. Jorge Kinoshita

São Paulo, 2022

Resumo

Este documento refere-se ao relatório de atividades da experiência de laboratório 3 do curso de Lab. de Microprocessadores da Escola Politécnica da Universidade de São Paulo. O intuito da atividade foi o contato com as operações de manipulação e processamento de dados do ARM. Ao longo do documento são mostrados resultados e comentários utilizados para a realização da experiência bem como os exercícios propostos no roteiro.

Sumário

1	Exercícios	2
1.1	Multiplicação com Sinal	2
1.2	Valor Absoluto	4
1.3	Divisão	5
1.4	Código Gray	7
1.5	Exercício Individual	9

Capítulo 1

Exercícios

Para este exercício foi preciso a consulta da seção 5.2 do livro-texto do curso. A tabela da respectiva seção é mostrada abaixo:

1.1 Multiplicação com Sinal

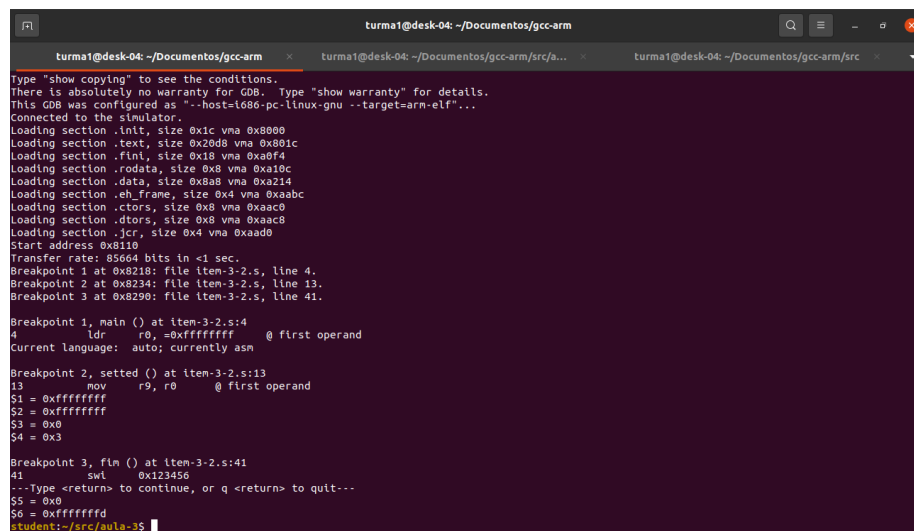
O código para multiplicação com sinal é mostrado abaixo:

```
1  .section .text, "ax"
2  .global main
3  main:
4      ldr    r0, =0xffffffff    @ first operand
5      ldr    r1, =0xffffffff
6      ldr    r2, =3             @ second operand
7      ldr    r3, =0
8      ldr    r4, =0             @ result and will also be controlling the
9      number of additions
10     ldr    r5, =0             @ result and will also have the return
11     address
12     ldr    r6, =0             @ signal indicator
13
14  setted:
15      mov    r9, r0             @ first operand
16      mov    r10, r1
17      bl     abs64
18      mov    r0, r9
19      mov    r1, r10
20      mov    r9, r2             @ second operand
21      mov    r10, r3
22      bl     abs64
23      mov    r2, r9
24      mov    r3, r10
25
26      ldr    r9, =0             @ MULTIPLICATION
27      ldr    r10, =0
28      bl     smul64
29  after:
```

```
29     mov     r4, r9
30     mov     r5, r10
31     cmp     r6, #1
32     moveq   r7, r4
33     moveq   r8, r5
34     moveq   r4, #0xffffffff
35     bleq    invt64
36     cmp     r6, #1
37     moveq   r4, r7
38     moveq   r5, r8
39
40 fim:
41     swi     0x123456
42
43
44 @
45 @  FUNCTIONS
46 @
47 smul64:
48     cmp     r4, #0
49     moveq   r5, lr
50     movs    r7, r3, lsr #1
51     bcs     calculate
52 shift:
53     mov     r7, r0      @ first operand shift
54     mov     r8, r1
55     bl      flsl64
56     mov     r0, r7
57     mov     r1, r8
58     mov     r7, r2      @ second operand shift
59     mov     r8, r3
60     bl      flsr64
61     mov     r2, r7
62     mov     r3, r8
63     add     r4, r4, #1  @ compute number of iterations
64     cmp     r4, #64
65     moveq   pc, r5      @ return to main
66     b       smul64
67 calculate:
68     adds    r10, r10, r1
69     add     r9, r9, r0
70     addcs   r9, r9, #1
71     b       shift
72
73 abs64: @ absolute value 64 bits
74     movs    r9, r9
75     rsblt   r9, r9, #0
76     sublt   r9, r9, #1
77     rsblt   r10, r10, #0
78     eorlt   r6, r6, #0b1
79     mov     pc, lr
80
81 flsl64: @ shift left 64 bits
82     movs    r8, r8, lsl #1
83     mov     r7, r7, lsl #1
84     addcss  r7, r7, #1
85     mov     pc, lr
```

```
86
87 flsr64: @ shift right 64 bits
88     mov     r8, r8, lsr #1
89     movs    r7, r7, lsr #1
90     addcss  r8, r8, #0x80000000
91     mov     pc, lr
92
93 invt64:
94     eor     r7, r7, r4
95     eor     r8, r8, r4
96     adds    r7, r7, #1
97     addcs   r8, r8, #1
98     mov     pc, lr
```

Para a execução, obtemos os seguintes resultados:



```
turma1@desk-04: ~/Documentos/gcc-arm
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "--host=686-pc-linux-gnu --target=arm-elf"...
Connected to the simulator.
Loading section .init, size 0x1c vma 0x0000
Loading section .text, size 0x20d8 vma 0x001c
Loading section .fini, size 0x18 vma 0xa0f4
Loading section .rodata, size 0x8 vma 0xa10c
Loading section .data, size 0x8a8 vma 0xa214
Loading section .eh_frame, size 0x4 vma 0xaabc
Loading section .ctors, size 0x8 vma 0xaac0
Loading section .dtors, size 0x8 vma 0xaac8
Loading section .jcr, size 0x4 vma 0xaad0
Start address 0x8110
Transfer rate: 85064 bits in ~1 sec.
Breakpoint 1 at 0x8218: file item-3-2.s, line 4.
Breakpoint 2 at 0x8234: file item-3-2.s, line 13.
Breakpoint 3 at 0x8290: file item-3-2.s, line 41.
Breakpoint 1, main () at item-3-2.s:4
4     ldr     r0, #0xffffffff @ first operand
Current language: auto; currently asm
Breakpoint 2, setted () at item-3-2.s:13
13     mov     r9, r0 @ first operand
$1 = 0xffffffff
$2 = 0xffffffff
$3 = 0x0
$4 = 0x3
Breakpoint 3, fim () at item-3-2.s:41
41     swi     0x123456
---Type <return> to continue, or q <return> to quit---
$5 = 0x0
$6 = 0xffffffff
student:~/src/aula-3$
```

Figura 1.1: Saída do exercício de multiplicação com sinal.

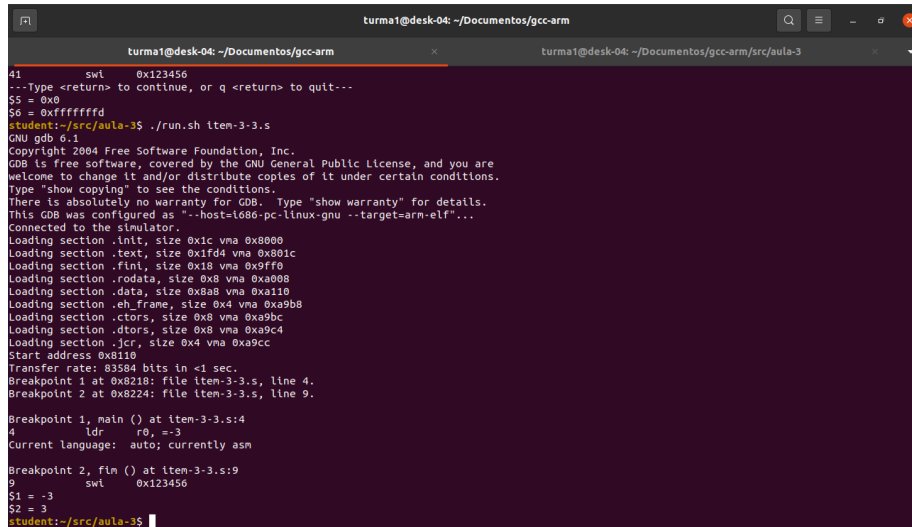
1.2 Valor Absoluto

Para este exercício, desejamos implementar a função $f(x) = |x|$, $x \in \mathbb{R}$. O código é mostrado abaixo:

```
1     .section .text, "ax"
2     .global main
3 main:
4     ldr     r0, #-3
5     ldr     r1, =0
6     bl     abs
7
8 fim:
9     swi     0x123456
10
```

```
11 abs:
12     movs    r2, r0
13     rsblt   r1, r0, #0
14     mov     pc, lr
```

A saída para o programa é dada por:



```
turma1@desk-04: ~/Documentos/gcc-arm
turma1@desk-04: ~/Documentos/gcc-arm
turma1@desk-04: ~/Documentos/gcc-arm/src/aula-3
41     swi     0x123456
--Type <return> to continue, or q <return> to quit--
$5 = 0x0
$6 = 0xffffffff
student1@src/aula-3$ ./run.sh item-3-3.s
GNU gdb 6.1
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "--host=i686-pc-linux-gnu --target=arm-elf"...
Connected to the simulator.
Loading section .init, size 0xc vma 0x0000
Loading section .text, size 0x1fd4 vma 0x001c
Loading section .fini, size 0x18 vma 0x9ff0
Loading section .rodata, size 0x8 vma 0xa008
Loading section .data, size 0x8a8 vma 0xa110
Loading section .eh_frame, size 0x4 vma 0xa9b8
Loading section .ctors, size 0x8 vma 0xa9bc
Loading section .dtors, size 0x8 vma 0xa9c4
Loading section .jcr, size 0x4 vma 0xa9cc
Start address 0x8110
Transfer rate: 83384 bits in <1 sec.
Breakpoint 1 at 0x8218: file item-3-3.s, line 4.
Breakpoint 2 at 0x8224: file item-3-3.s, line 9.
Breakpoint 1, main () at item-3-3.s:4
4     ldr     r0, =3
Current language: auto; currently asm
Breakpoint 2, fin () at item-3-3.s:9
9     swi     0x123456
$1 = -3
$2 = 3
student1@src/aula-3$
```

Figura 1.2: Saída do exercício de valor absoluto.

1.3 Divisão

O algoritmo de divisão é mostrado abaixo segundo a abordagem do algoritmo de subtrações sucessivas:

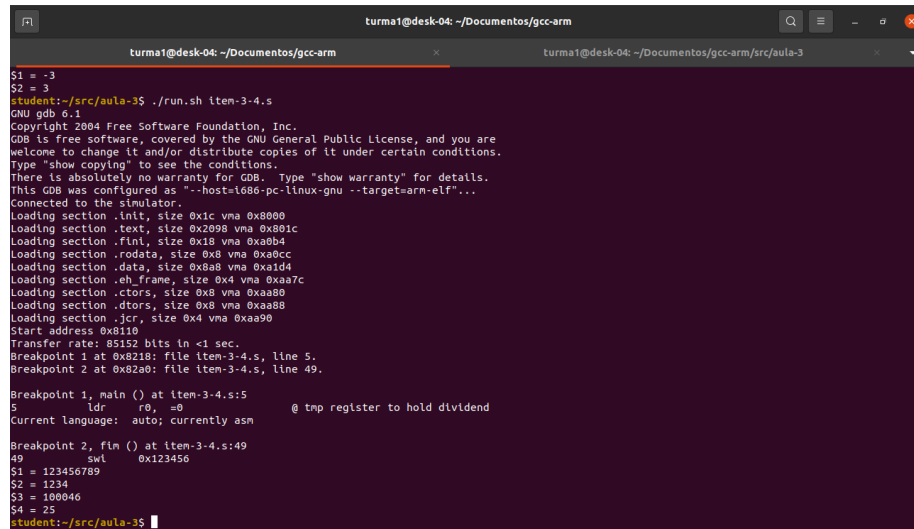
```
1     .section .text, "ax"
2     .global main
3 main:
4     @ variables
5     ldr     r0, =0                @ tmp register to hold dividend
6     ldr     r1, =123456789        @ dividend
7     ldr     r2, =1234             @ divisor
8     ldr     r3, =0                @ quotient
9     ldr     r4, =0                @ bits to be shifted
10    ldr     r5, =0                @ remainder
11    ldr     r6, =0                @ tmp register used in division
12    ldr     r7, =0                @ most significant bit from
13    ldr     r8, =0                @ most significant bit from
14    ldr     r9, =0                @ argument of msb (number to
15    ldr     r10, =0               @ argument of msb (number to
    find msb)
```

```
16     ldr    r11, =0                @ tmp register to hold dividend
17
18     @ saving dividend and divisor
19     mov    r0, r1
20     mov    r11, r2
21
22     @ find msb
23     mov    r10, r1                @ passing arguments
24     ldr    r9, =0
25     cmp    r2, #0                @ if divisor is 0, jump to fim
26     beq    fim
27     cmp    r2, #1                @ if divisor is 1, quotient is
dividend and jump to fim
28     moveq  r3, r1
29     beq    fim
30     bl     msb
31     mov    r7, r9
32     mov    r10, r2                @ passing arguments
33     ldr    r9, =0
34     bl     msb
35     mov    r8, r9
36
37     @ division
38     sub    r4, r7, r8
39     mov    r2, r2, lsl r4        @ shift divisor to msb
40     add    r4, r4, #1
41     bl     division
42     mov    r5, r1
43
44     @ restoring dividend and divisor
45     mov    r1, r0
46     mov    r2, r11
47
48 fim:
49     swi    0x123456
50
51 msb:
52     movs   r10, r10, lsr #1
53     addne  r9, r9, #1
54     moveq  pc, lr
55     b     msb
56
57 division:
58     subs   r6, r1, r2
59     bcc    ndivide
60     bcs    divide
61
62 divide:
63     mov    r3, r3, lsl #1
64     add    r3, r3, #1
65     sub    r1, r1, r2
66     mov    r2, r2, lsr #1
67     subs   r4, r4, #1
68     moveq  pc, lr
69     b     division
70
71 ndivide:
72     mov    r3, r3, lsl #1
73     mov    r2, r2, lsr #1
```



```
72     subs    r4, r4, #1
73     moveq   pc, lr
74     b       division
```

A saída para o programa é dada por:



```
turma1@desk-04: ~/Documentos/gcc-arm
turma1@desk-04: ~/Documentos/gcc-arm/src/aula-3
$1 = -3
$2 = 3
student:~/src/aula-3$ ./run.sh item-3-4.s
GNU gdb 6.1
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "--host=i686-pc-linux-gnu --target=arm-elf"...
Connected to the simulator.
Loading section .init, size 0xc1c vma 0x8000
Loading section .text, size 0x2098 vma 0x801c
Loading section .finit, size 0x18 vma 0xa0b4
Loading section .rodata, size 0x8 vma 0xa0cc
Loading section .data, size 0x8a8 vma 0xa1d4
Loading section .eh_frame, size 0x4 vma 0xaa7c
Loading section .ctors, size 0x8 vma 0xaa80
Loading section .dtors, size 0x8 vma 0xaa88
Loading section .jcr, size 0x4 vma 0xaa90
Start address 0xb110
Transfer rate: 89152 bits in <1 sec.
Breakpoint 1 at 0xb218: file item-3-4.s, line 5.
Breakpoint 2 at 0xb2a0: file item-3-4.s, line 49.
Breakpoint 1, main () at item-3-4.s:5
5     ldr     r0, =0          @ tmp register to hold dividend
Current language: auto; currently asm
Breakpoint 2, fin () at item-3-4.s:49
49     swi    0x123456
$1 = 123456789
$2 = 1234
$3 = 100046
$4 = 25
student:~/src/aula-3$
```

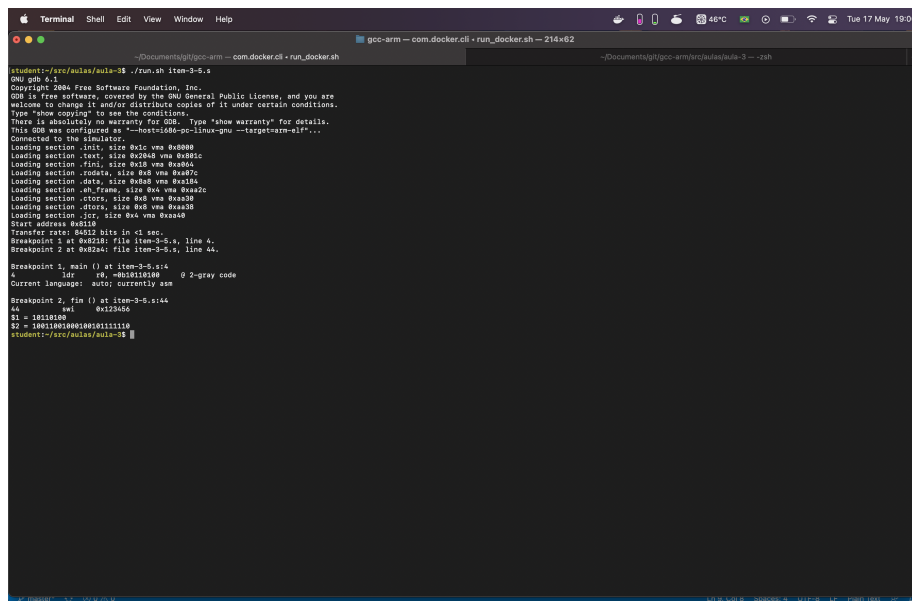
Figura 1.3: Saída do exercício de divisão.

1.4 Código Gray

Para o código Gray com a transição do caso $n = 2$ para $n = 3$, temos o seguinte código:

```
1     .section .text, "ax"
2     .global main
3 main:
4     ldr     r0, =0b10110100    @ 2-gray code
5     ldr     r1, =0             @ 3-gray code
6     ldr     r2, =0             @ 10
7     ldr     r3, =0             @ 11
8     ldr     r4, =0             @ 01
9     ldr     r5, =0             @ 00
10    ldr     r6, =0
11
12    mov     r2, r0, lsr #6
13    mov     r3, r0, lsr #4
14    mov     r3, r3, lsl #30
15    mov     r3, r3, lsr #30
16    mov     r4, r0, lsr #2
17    mov     r4, r4, lsl #30
18    mov     r4, r4, lsr #30
19    mov     r5, r5, lsl #30
20    mov     r5, r5, lsr #30
```

```
21
22     mov     r6, r2, lsl #21
23     add     r1, r1, r6
24     mov     r6, r3, lsl #18
25     add     r1, r1, r6
26     mov     r6, r4, lsl #15
27     add     r1, r1, r6
28     mov     r6, r5, lsl #12
29     add     r1, r1, r6
30
31     add     r6, r5, #0b100
32     mov     r6, r6, lsl #9
33     add     r1, r1, r6
34     add     r6, r4, #0b100
35     mov     r6, r6, lsl #6
36     add     r1, r1, r6
37     add     r6, r3, #0b100
38     mov     r6, r6, lsl #3
39     add     r1, r1, r6
40     add     r6, r2, #0b100
41     add     r1, r1, r6
42
43 fim:
44     swi     0x123456
```

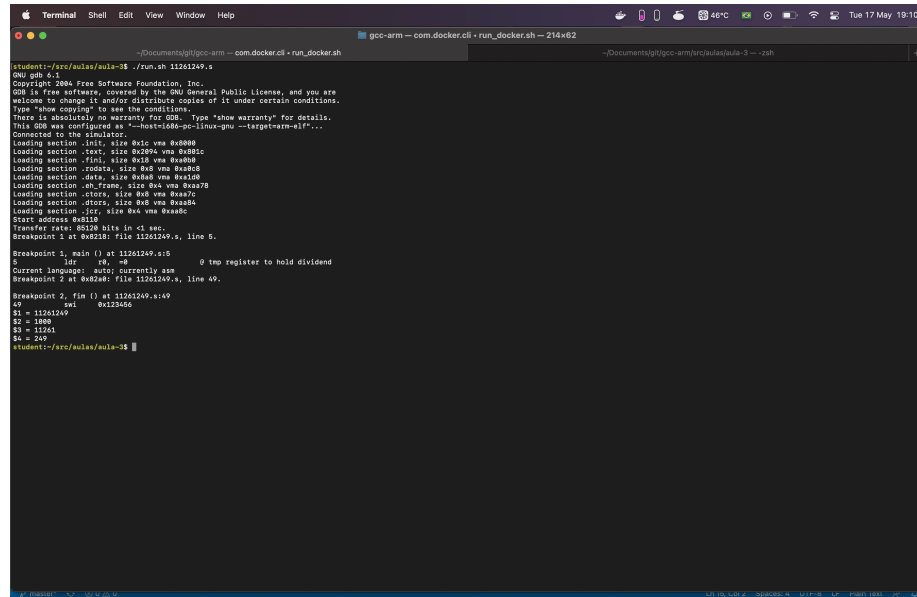


```
Terminal Shell Edit View Window Help
gcc-arm - com.docker.cli - run_docker.sh - 214x62
~/Documents/gt/gcc-arm/riscv/aulas/aula-3 - ssh
student@riscv/aulas/aula-3:~$ ./run.sh item-3-5.s
GNU gdb 6.1
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type 'show copying' to see the conditions.
There is absolutely no warranty for GDB.  Type 'show warranty' for details.
This GDB was configured as "x86_64-pc-linux-gnu --target=riscv64".
Connected to the simulator.
Loading section .init, size 0x1c vma 0x0000
Loading section .text, size 0x20d8 vma 0x001c
Loading section .fini, size 0x1b vma 0x00d6
Loading section .rodata, size 0x8 vma 0x007c
Loading section .data, size 0x40d vma 0x0136
Loading section .eh_frame, size 0x4 vma 0x0a2c
Loading section .ctors, size 0x8 vma 0x0a38
Loading section .dctors, size 0x4 vma 0x0a3c
Start address 0x0110
Transfer rate 0x0110 bits in-cl/sec
Breakpoint 1 at 0x0218: file item-3-5.s, line 4.
Breakpoint 2 at 0x0204: file item-3-5.s, line 44.
Breakpoint 1, main () at item-3-5.s:4
4      lwr     r4, whos116000      @ 2-gray code
Current language: auto; currently asm
Breakpoint 2, fin () at item-3-5.s:44
44     swi     0x123456
$1 = 10110100
$2 = 1001100100100101111111
student@riscv/aulas/aula-3:~$
```

Figura 1.4: Saída do exercício de código *Gray*.

1.5 Exercício Individual

Este exercício utiliza o mesmo código 1.3 apresentado no algoritmo da divisão, a diferença são as entradas.



```
Terminal Shell Edit View Window Help
gcc-arm - com.docker.cli - run_docker.sh - 214x62
~/Documents/git/gcc-arm - com.docker.cli - run_docker.sh
~/Documents/git/gcc-arm/riscv/aulas/aula-3 - -zsh

student@riscv/aulas/aula-3:~$ ./run.sh 11261249.s
GNU gdb 8.1
Copyright 2018 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu --target=riscv-elf"...
Connected to the simulator.
Loading section .init, size 0x1c vma 0x0000
Loading section .text, size 0x2094 vma 0x000c
Loading section .fini, size 0x18 vma 0x0008
Loading section .rodata, size 0x8 vma 0x000d
Loading section .data, size 0x008 vma 0x0010
Loading section .eh_frame, size 0x4 vma 0x0078
Loading section .core, size 0x0 vma 0x007c
Loading section .store, size 0x0 vma 0x007d
Loading section .jcr, size 0x4 vma 0x008c
Start address 0x0000
Transfer rate: 85128 bits in <1 sec.
Breakpoint 3 at 0x0200: file 11261249.s, line 5.
Breakpoint 1, main () at 11261249.s:5
5       ldr    r0, =0          @ two register to hold dividend
Current language: auto; currently asm
Breakpoint 2 at 0x0200: file 11261249.s, line 49.
Breakpoint 2, fin () at 11261249.s:49
49      ldr    r1,    0x120000
r1 = 11261249
r2 = 1000
r3 = 11261
r4 = 249
student@riscv/aulas/aula-3:~$
```

Figura 1.5: Exemplo de saída do exercício exercício individual com o NUSP do aluno Otávio Freitas.