01_a.c

Lista prática 01 > 01_a.c

```c
#include <stdio.h>
#include <stdlib.h>

int soma_quadrados(int x, int y) {
    return (x * x) + (y * y);
}

int main(){
    int a = 2;
    int b = 3;
    int resultado = 0;

    resultado = soma_quadrados(a,b);
    printf("Resultado: %d", resultado);

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'01_a.exe'
Resultado: 13
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
```

C/C++ ...

C/C++ Com...

Compilation successful.

Compile & Run

Ln 17, Col 2 Spaces: 4 UTF-8 CRLF {} C

C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\01_a.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24040002	addiu $4,$0,0x00000002	7: li $a0, 2 # a = 2
	0x00400004	0x24050003	addiu $5,$0,0x00000003	8: li $a1, 3 # b = 3
	0x00400008	0x0c10000d	jal 0x00400034	9: jal soma_quadrados
	0x0040000c	0x00024021	addu $8,$0,$2	10: move $t0, $v0 # salva resultado
	0x00400010	0x24020004	addiu $2,$0,0x00000004	12: li $v0, 4
	0x00400014	0x3c011001	lui $1,0x00001001	13: la $a0, msg
	0x00400018	0x34240000	ori $4,$1,0x00000000	
	0x0040001c	0x0000000c	syscall	14: syscall
	0x00400020	0x00082021	addu $4,$0,$8	15: move $a0, $t0
	0x00400024	0x24020001	addiu $2,$0,0x00000001	16: li $v0, 1

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x75736552	0x6461746c	0x00203a6f	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages Run I/O

Clear

```
Resultado: 13
-- program is finished running --
```

Registers Coproc 1 Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000000d
$a1	5	0x00000003
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x0000000d
$t1	9	0x00000004
$t2	10	0x00000009
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x0040000c
pc		0x00400034
hi		0x00000000
lo		0x00000009

Edit | Execute

01_a.asm

```
4         .globl main
5   main:
6         # coloca os argumentos direto
7         li   $a0, 2         # a = 2
8         li   $a1, 3         # b = 3
9         jal  soma_quadrados
10        move $t0, $v0       # salva resultado
11        # printf("Resultado: %d", resultado);
12        li   $v0, 4
13        la   $a0, msg
14        syscall
15        move $a0, $t0
16        li   $v0, 1
17        syscall
18        # return 0;
19        li   $v0, 10
20        syscall
21  soma_quadrados:
22        mul  $t1, $a0, $a0   # x*x
23        mul  $t2, $a1, $a1   # y*y
24        add  $v0, $t1, $t2   # retorno
25        jr   $ra
26
```

Line: 1 Column: 1 ☑ Show Line Numbers

Mars Messages | Run I/O

Clear

```
Resultado: 13
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000000d
$a1	5	0x00000003
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x0000000d
$t1	9	0x00000004
$t2	10	0x00000009
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x0040000c
pc		0x00400034
hi		0x00000000
lo		0x00000009

C 01_b.c

Lista prática 01 > C 01_b.c

```c
#include <stdio.h>

int media3(int a, int b, int c) {
    return (a + b + c) / 3;
}

int main(){
    int a = 1, b = 2, c = 0, resultado = 0;
    resultado = media3(a,b,c);
    printf("Resultado da media por 3: %d \n", resultado);

    return 0;
}
```

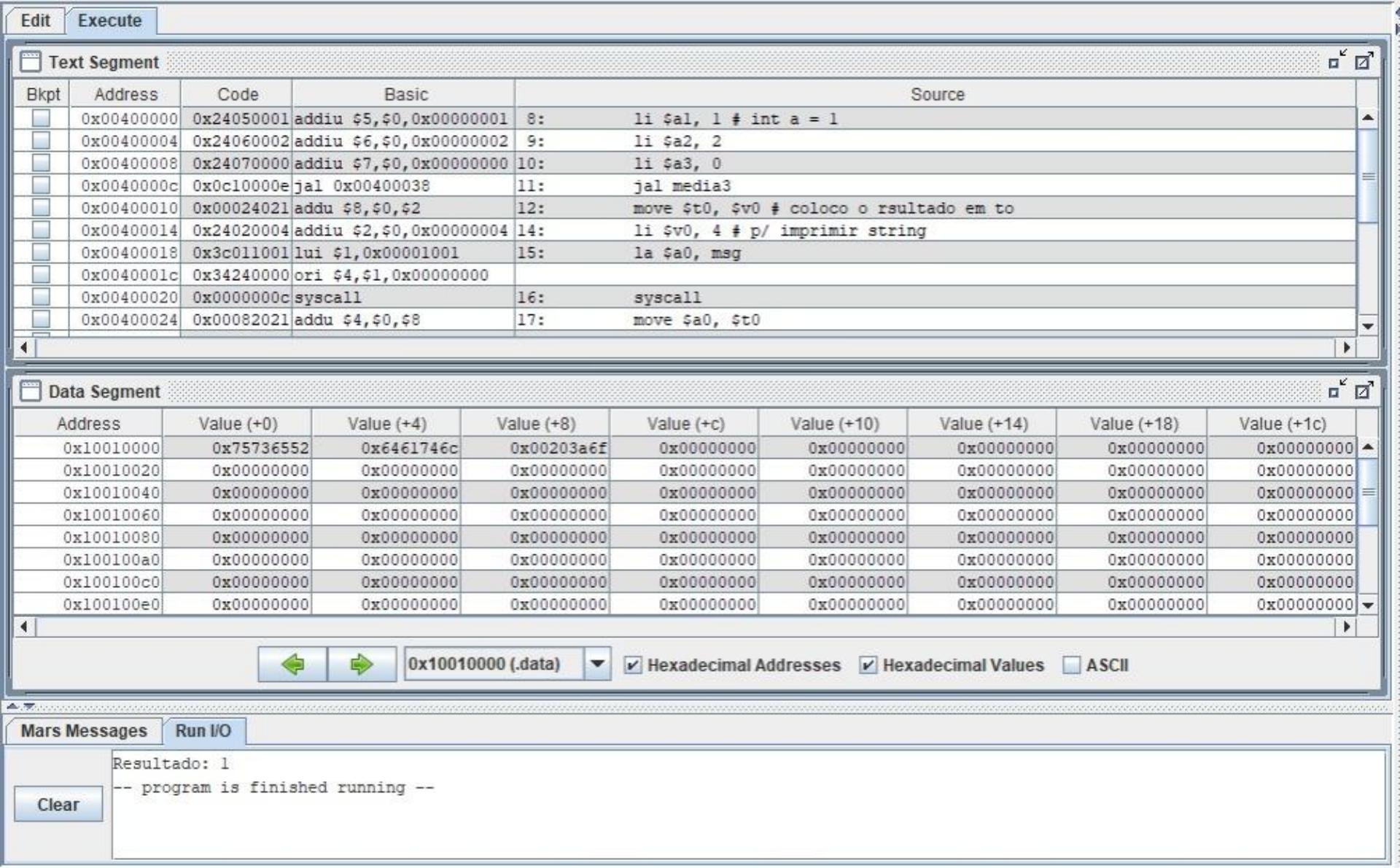
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'01_b.exe'
Resultado da media por 3: 1
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
```

C/C++ ...
C/C++ Com...

Compilation successful.

pile ▷ Compile & Run Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} C


Edit
Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x24050001 addiu $5,$0,0x00000001 8: li $a1, 1 # int a = 1
0x00400004 0x24060002 addiu $6,$0,0x00000002 9: li $a2, 2
0x00400008 0x24070000 addiu $7,$0,0x00000000 10: li $a3, 0
0x0040000c 0x0c10000e jal 0x00400038 11: jal media3
0x00400010 0x00024021 addu $8,$0,$2 12: move $t0, $v0 # coloco o rsultado em to
0x00400014 0x24020004 addiu $2,$0,0x00000004 14: li $v0, 4 # p/ imprimir string
0x00400018 0x3c011001 lui $1,0x00001001 15: la $a0, msg
0x0040001c 0x34240000 ori $4,$1,0x00000000
0x00400020 0x0000000c syscall 16: syscall
0x00400024 0x00082021 addu $4,$0,$8 17: move $a0, $t0
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x75736552 0x6461746c 0x00203a6f 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010060 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data)
Hexadecimal Addresses
Hexadecimal Values
ASCII
Mars Messages
Run I/O
Clear
Resultado: 1
-- program is finished running --

Edit | Execute

01_a.asm | mips1.asm | 01_b.asm

```
6  main:
7          # iniicalizar as variáveis
8          li $a1, 1 # int a = 1
9          li $a2, 2
10         li $a3, 0
11         jal media3
12         move $t0, $v0 # coloco o rsultado em to
13         # printf ("resultado: ");
14         li $v0, 4 # p/ imprimir string
15         la $a0, msg
16         syscall
17         move $a0, $t0
18         li $v0, 1 # p/ imprimir int
19         syscall
20         li $v0, 10
21         syscall
22 media3:
23         add $t1, $a1, $a2
24         add $t1, $t1, $a3
25         li $t2, 3
26         div $t1, $t2 #t1/3
27         mflo $v0 #resultado em r0
28         jr  $ra
```

Line: 28 Column: 9 ☑ Show Line Numbers

Mars Messages | Run I/O

Clear

```
Resultado: 1
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000001
$a1	5	0x00000001
$a2	6	0x00000002
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x00000003
$t2	10	0x00000003
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x00400038
hi		0x00000000
lo		0x00000001

01_c.c

Lista prática 01 > 01_c.c

```c
#include <stdio.h>

int dobro_sub(int a, int b) {
    return 2 * (a - b);
}

int main (){
    int a = 0, b = 5, resultado;
    resultado = dobro_sub(a,b);
    printf("Resultado: %d", resultado);
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'01_c.exe'
Resultado: -10
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
```

C/C++ ...

C/C++ Com...

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x24040000	addiu $4,$0,0x00000000	6:	li $a0, 0 # carrego a0 com 0
	0x00400004	0x24050005	addiu $5,$0,0x00000005	7:	li $a1, 5 # a1 com 5
	0x00400008	0x0c100008	jal 0x00400020	8:	jal dobro_sub # chamo a função
	0x0040000c	0x00022021	addu $4,$0,$2	9:	move $a0, $v0# p/ imprimir tem que estar em A e não em V
	0x00400010	0x24020001	addiu $2,$0,0x00000001	10:	li $v0, 1 #print inteiro
	0x00400014	0x0000000c	syscall	11:	syscall
	0x00400018	0x2402000a	addiu $2,$0,0x0000000a	12:	li $v0, 10 #encerrar
	0x0040001c	0x0000000c	syscall	13:	syscall
	0x00400020	0x00851022	sub $2,$4,$5	15:	sub $v0, $a0, $a1
	0x00400024	0x00021040	sll $2,$2,0x00000001	16:	sll $v0, $v0, 1

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages | Run I/O

Clear

```
-10
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0xfffffff6
$a1	5	0x00000005
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x0040000c
pc		0x00400020
hi		0x00000000
lo		0x00000000

Edit | Execute

01_a.asm | mips1.asm | 01_b.asm | 01_c.asm*

```
1         .data
2         .text
3         .globl main
4
5   main:
6         li $a0, 0 # carrego a0 com 0
7         li $a1, 5 # a1 com 5
8         jal dobro_sub # chamo a função
9         move $a0, $v0# p/ imprimir tem que estar em A e não em V
10        li $v0, 1 #print inteiro
11        syscall
12        li $v0, 10 #encerrar
13        syscall
14  dobro_sub:
15        sub $v0, $a0, $a1
16        sll $v0, $v0, 1
17        jr $ra
18
19
20
```

Line: 8 Column: 32 ☑ Show Line Numbers

Mars Messages | Run I/O

Clear

```
-10
-- program is finished running --
```

Lista prática 01 > C 02_a.c

```c
#include <stdio.h>

int soma_dupla(int v[]) {
    return v[0] + v[1];
}

int main(){
    int vetor[2] = {1,2};
    int resultado;

    resultado = soma_dupla(vetor);
    printf("Resultado: %d", resultado);

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'02_a.exe'
Resultado: 3
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
```

C/C++ ... ✓
C/C++ Com...

Compilation successful.

npile ▷ Compile & Run Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} C

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x3c011001	lui $1,0x00001001	7:	la $a2, vetor
	0x00400004	0x34260000	ori $6,$1,0x00000000		
	0x00400008	0x0c100009	jal 0x00400024	8:	jal soma_dupla
	0x0040000c	0x00024021	addu $8,$0,$2	9:	move $t0, $v0
	0x00400010	0x00082021	addu $4,$0,$8	10:	move $a0, $t0
	0x00400014	0x24020001	addiu $2,$0,0x00000001	11:	li $v0, 1
	0x00400018	0x0000000c	syscall	12:	syscall
	0x0040001c	0x2402000a	addiu $2,$0,0x0000000a	13:	li $v0, 10
	0x00400020	0x0000000c	syscall	14:	syscall
	0x00400024	0x8cc90000	lw $9,0x00000000($6)	17:	lw $t1, 0($a2)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000001	0x00000002	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages | Run I/O

Clear

```
3
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000003
$a1	5	0x00000000
$a2	6	0x10010000
$a3	7	0x00000000
$t0	8	0x00000003
$t1	9	0x00000001
$t2	10	0x00000002
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x0040000c
pc		0x00400024
hi		0x00000000
lo		0x00000000

Edit | Execute

01_a.asm | mips1.asm | 01_b.asm | 01_c.asm | 02_a.asm*

```
1         .data
2   vetor: .word 1,2
3         .text
4         .globl main
5
6   main:
7         la $a2, vetor # carrego o vetor em a2
8         jal soma_dupla # chamo a função
9         move $a0, $v0 # resultado que retornou em v0 eu coloquei em a0
10        li $v0, 1 # instrução que imprime inteiro
11        syscall
12        li $v0, 10 # encerrar
13        syscall
14
15  soma_dupla:
16        lw $t1, 0($a2) # coloco em t1 o elemento de posição [0] do vetor
17        lw $t2, 4($a2)  #em t2 o de posição [1]
18        add $v0, $t1, $t2 # somo ambos
19        jr $ra # retorno
20
21
```

Line: 6 Column: 6 ☑ Show Line Numbers

Mars Messages | Run I/O

Clear

```
3
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeff
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

02_b.c

Lista prática 01 > 02_b.c

```c
#include <stdio.h>

int multiplica_espacado(int v[]) {
    return v[0] * v[4];
}

int main(){
    int vetor[5] = {1,2,3,4,5};
    int resultado;

    resultado = multiplica_espacado(vetor);

    printf("Resultado: %d \n", resultado);

    return 0;
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'02_b.exe'
Resultado: 5
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
```

C/C++ ...

C/C++ Com...

Compilation successful.

pile Compile & Run Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} C

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui $1,0x00001001	7: la $a2, vetor
	0x00400004	0x34260000	ori $6,$1,0x00000000	
	0x00400008	0x0c100008	jal 0x00400020	8: jal multiplica_espacado
	0x0040000c	0x00022021	addu $4,$0,$2	9: move $a0, $v0
	0x00400010	0x24020001	addiu $2,$0,0x00000001	10: li $v0, 1
	0x00400014	0x0000000c	syscall	11: syscall
	0x00400018	0x2402000a	addiu $2,$0,0x0000000a	12: li $v0, 10
	0x0040001c	0x0000000c	syscall	13: syscall
	0x00400020	0x8cc90000	lw $9,0x00000000($6)	16: lw $t1, 0($a2)
	0x00400024	0x8cca0010	lw $10,0x00000010($6)	17: lw $t2, 16($a2)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000001	0x00000002	0x00000003	0x00000004	0x00000005	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages | Run I/O

Clear

```
5
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000005
$a1	5	0x00000000
$a2	6	0x10010000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000001
$t2	10	0x00000005
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x0040000c
pc		0x00400020
hi		0x00000000
lo		0x00000005

Edit | Execute

01_a.asm | mips1.asm | 01_b.asm | 01_c.asm | 02_a.asm* | 02_b.asm*

```
1         .data
2  vetor: .word 1,2,3,4,5
3         .text
4         .globl main
5
6  main:
7         la $a2, vetor # carrego o vetor em a2
8         jal multiplica_espacado # chamo a função
9         move $a0, $v0 # coloco o retorno em a0
10        li $v0, 1 # print int
11        syscall
12        li $v0, 10 # encerro
13        syscall
14
15 multiplica_espacado:
16        lw $t1, 0($a2) # carrego a posição 1 do vetor (valor 1)
17        lw $t2, 16($a2) # carrego a posição 5 do vetor valor 5)
18        mul $v0, $t1, $t2 # 1 * 5
19        jr $ra # retorno
20
21
```

Line: 19 Column: 18 ☑ Show Line Numbers

Mars Messages | Run I/O

Clear

```
5
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

02_c.c

Lista prática 01 > 02_c.c

```c
#include <stdio.h>

int soma_n(int v[], int n) {
int soma = 0;
for (int i = 0; i < n; i++)
    soma += v[i];
return soma;
}

int main (){
    int vetor[3] = {1,2,3};
    int soma = 0, n = 2;

    soma = soma_n(vetor, n);
    printf("Resultado da soma: %d", soma);
    return 0;

}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'02_c.exe'
Resultado da soma: 3
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
```

C/C++ ...

C/C++ Com...

Compilation successful.

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x3c011001	lui $1,0x00001001	7:	la $a0, vetor #carrego o vetor
	0x00400004	0x34240000	ori $4,$1,0x00000000		
	0x00400008	0x24050002	addiu $5,$0,0x00000002	8:	li $a1, 2 # a1 tem 2
	0x0040000c	0x0c100009	jal 0x00400024	9:	jal soma_n # chamo função
	0x00400010	0x00022021	addu $4,$0,$2	10:	move $a0, $v0 # movo o resultado da função para a0
	0x00400014	0x24020001	addiu $2,$0,0x00000001	11:	li $v0, 1
	0x00400018	0x0000000c	syscall	12:	syscall
	0x0040001c	0x2402000a	addiu $2,$0,0x0000000a	13:	li $v0, 10
	0x00400020	0x0000000c	syscall	14:	syscall
	0x00400024	0x00001021	addu $2,$0,$0	17:	move $v0, $zero # corresponde a soma = 0 no programa em c

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000001	0x00000002	0x00000003	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages | Run I/O

Clear

```
3
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000003
$a1	5	0x00000002
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000002
$t1	9	0x00000000
$t2	10	0x10010004
$t3	11	0x00000002
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x00400024
hi		0x00000000
lo		0x00000000

Edit | Execute

01_a.asm | mips1.asm | 01_b.asm | 01_c.asm | 02_a.asm* | 02_b.asm* | 03_c.asm | 02_c.asm | 03_a.asm*

C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\01_a.asm

```
main:
        la $a0, vetor #carreço o vetor
        li $a1, 2 # a1 tem 2
        jal soma_n # chamo função
        move $a0, $v0 # movo o resultado da função para a0
        li $v0, 1
        syscall
        li $v0, 10
        syscall

soma_n:
        move $v0, $zero # corresponde a soma = 0 no programa em c
        move $t0, $zero # aqui é i = 0
loop:
        slt $t1, $t0, $a1 # n < i, cond que não quero
        beq $t1, $zero, exit # se t1 == 0, saio do loop
        sll $t2, $t0, 2 # desloco 4 posições (i * 4)
        add $t2, $t2, $a0 # nova posição "livre" do vetor
        lw $t3, 0($t2)
        add $v0, $v0, $t3
        addi $t0, $t0, 1 # i++
        j loop # volto pto inicio do loop
exit:
        jr $ra # retorno da função
```

Line: 5 Column: 2 ☑ Show Line Numbers

Mars Messages | Run I/O

Clear

```
3
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000003
$a1	5	0x00000002
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000002
$t1	9	0x00000000
$t2	10	0x10010004
$t3	11	0x00000002
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x00400024
hi		0x00000000
lo		0x00000000

ARQUITETURA

03_a.c

Lista prática 01 > 03_a.c

```c
#include <stdio.h>

int abs(int x) {
if (x < 0)
    return -x;
return x;
}

int main(){
    int a = -2, soma;

    soma = abs(a);
    printf("Resultado: %d", soma);
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'03_a.exe'
Resultado: 2
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
```

C/C++ ...

C/C++ Com...

Compilation successful.

Compile & Run

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} C

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x2404fffe	addiu $4,$0,0xfffffffe	6:	li $a0, -2 # carrego a0 com -2
	0x00400004	0x0c100007	jal 0x0040001c	7:	jal abs_ # chamo a função
	0x00400008	0x00022021	addu $4,$0,$2	8:	move $a0, $v0 # movo o retorno da função para a0
	0x0040000c	0x24020001	addiu $2,$0,0x00000001	9:	li $v0, 1 # imprimi int
	0x00400010	0x0000000c	syscall	10:	syscall
	0x00400014	0x2402000a	addiu $2,$0,0x0000000a	11:	li $v0, 10 # encerra
	0x00400018	0x0000000c	syscall	12:	syscall
	0x0040001c	0x28890000	slti $9,$4,0x00000000	15:	slti $t1, $a0, 0 # x < 0 (t1 = 1 se x < 0 ou recebe 0 se x > 0)
	0x00400020	0x11200003	beq $9,$0,0x00000003	16:	beq $t1, $zero, else # se t1 == 1, pulo para o else. O beq verifica s...
	0x00400024	0x240affff	addiu $10,$0,0xffff...	17:	li $t2, -1 # em t2 tem -2

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages | Run I/O

Clear

```
2
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000002
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000001
$t2	10	0xffffffff
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400008
pc		0x0040001c
hi		0x00000000
lo		0x00000002

Edit | Execute

01_a.asm | mips1.asm | 01_b.asm | 01_c.asm | 02_a.asm* | 02_b.asm* | 03_c.asm | 02_c.asm | 03_a.asm

```
1       .data
2       .text
3       .globl main
4
5   main:
6       li $a0, -2 # carrego a0 com -2
7       jal abs_ # chamo a função
8       move $a0, $v0 # movo o retorno da função para a0
9       li $v0, 1 # imprimi int
10      syscall
11      li $v0, 10 # encerra
12      syscall
13
14  abs_:
15      slti $t1, $a0, 0 # x < 0 ( t1 = 1 se x < 0 ou recebe 0 se x > 0)
16      beq $t1, $zero, else # se t1  == 1, pulo para o else. O beq verifica se t1 != 0, oque significa t1 == 1
17      li $t2, -1 # em t2 tem -2
18      mul $v0, $a0, $t2 # multiplico por -1
19      jr $ra # retorno o valor
20  else:
21      move $v0, $a0 # caso t1 == 1, apenas movo o valor do paramêtro para a variavel de retorno
22      jr $ra # retorno
23
```

Line: 22 Column: 18 ☑ Show Line Numbers

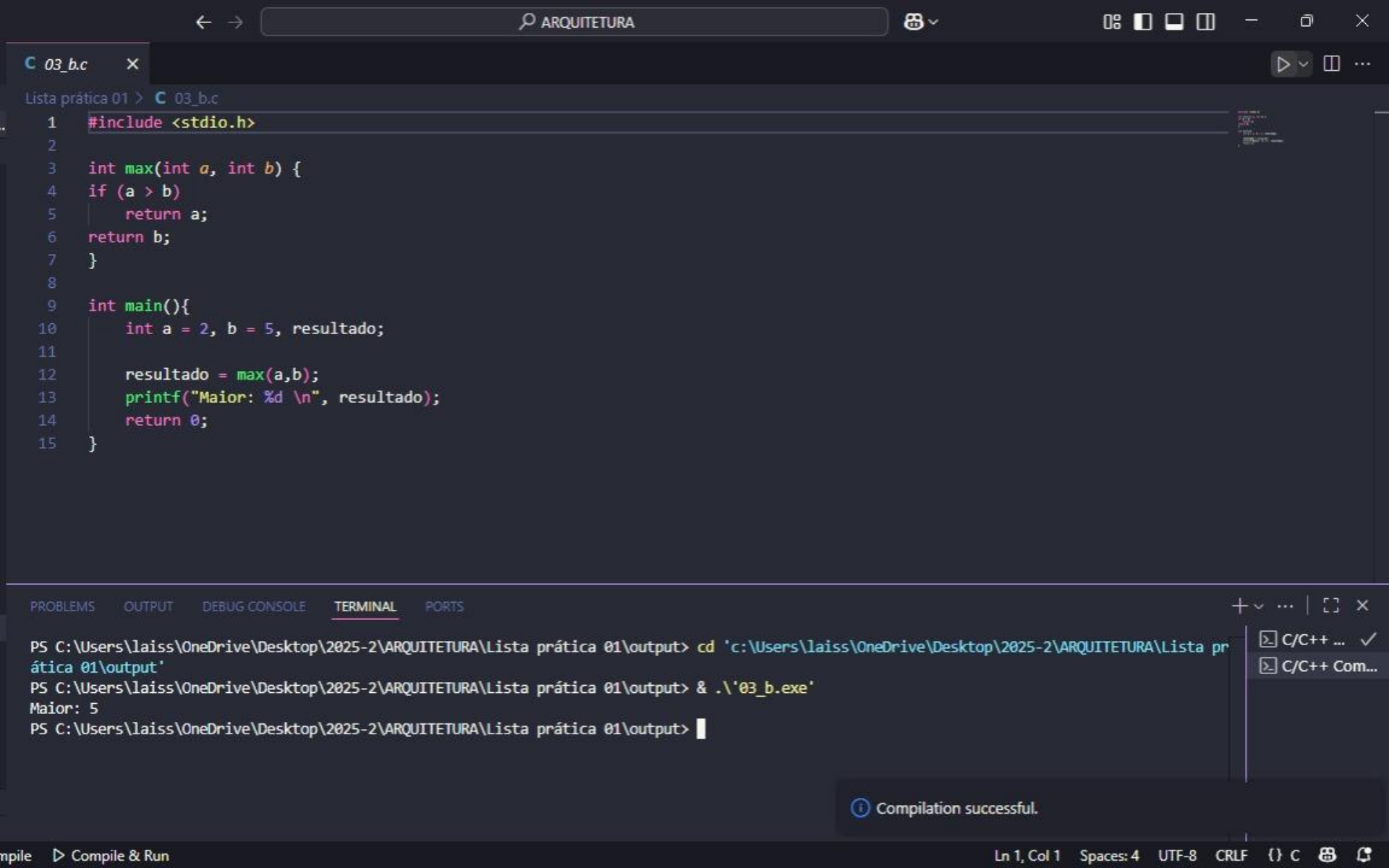
Mars Messages | Run I/O

Clear

```
2
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000002
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000001
$t2	10	0xffffffff
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400008
pc		0x0040001c
hi		0x00000000
lo		0x00000002


ARQUITETURA
03_b.c
Lista prática 01 > 03_b.c
#include <stdio.h>
int max(int a, int b) {
if (a > b)
return a;
return b;
}
int main(){
int a = 2, b = 5, resultado;
resultado = max(a,b);
printf("Maior: %d \n", resultado);
return 0;
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'03_b.exe'
Maior: 5
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
C/C++ ...
C/C++ Com...
Compilation successful.
Compile & Run
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
C

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24040002	addiu $4,$0,0x00000002	6: li $a0, 2 # carrego 2 em a0
	0x00400004	0x24050005	addiu $5,$0,0x00000005	7: li $a1 , 5 # e em a1
	0x00400008	0x0c100008	jal 0x00400020	8: jal max # chamo a função
	0x0040000c	0x00022021	addu $4,$0,$2	9: move $a0, $v0 # coloco o resultado em a0
	0x00400010	0x24020001	addiu $2,$0,0x00000001	10: li $v0, 1 # imprimo o resultado
	0x00400014	0x0000000c	syscall	11: syscall
	0x00400018	0x2402000a	addiu $2,$0,0x0000000a	12: li $v0, 10 # encerro
	0x0040001c	0x0000000c	syscall	13: syscall
	0x00400020	0x0085482a	slt $9,$4,$5	15: max: slt $t1, $a0, $a1 # t1 = 1 se a0 < a1
	0x00400024	0x15200002	bne $9,$0,0x00000002	16: bne $t1, $zero, else # se t1 != 0, pulo p/ o else, ou seja, executo a ...

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages | Run I/O

Clear

```
5
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000005
$a1	5	0x00000005
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000001
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x0040000c
pc		0x00400020
hi		0x00000000
lo		0x00000000

Edit | Execute

01_a.asm | mips1.asm | 01_b.asm | 01_c.asm | 02_a.asm* | 02_b.asm* | 03_c.asm | 02_c.asm | 03_a.asm | 03_b.asm

```
1         .data
2         .text
3         .globl main
4
5   main:
6         li $a0, 2 # carrego 2 em a0
7         li $a1 , 5 # e em a1
8         jal max # chamo a função
9         move $a0, $v0 # coloco o resultado em a0
10        li $v0, 1 # imprimo o resultado
11        syscall
12        li $v0, 10 # encerro
13        syscall
14
15  max:  slt $t1, $a0, $a1 # t1 = 1 se a0 < a1
16        bne $t1, $zero, else # se t1 != 0, pulo p/ o else, ou seja, executo a proxima parte apenas se a0 > a1
17        move $v0, $a0 # preparo o retorno da função
18        jr $ra # retorno
19  else:
20        move $v0, $a1
21        jr $ra
22
```

Line: 20 Column: 16 ☑ Show Line Numbers

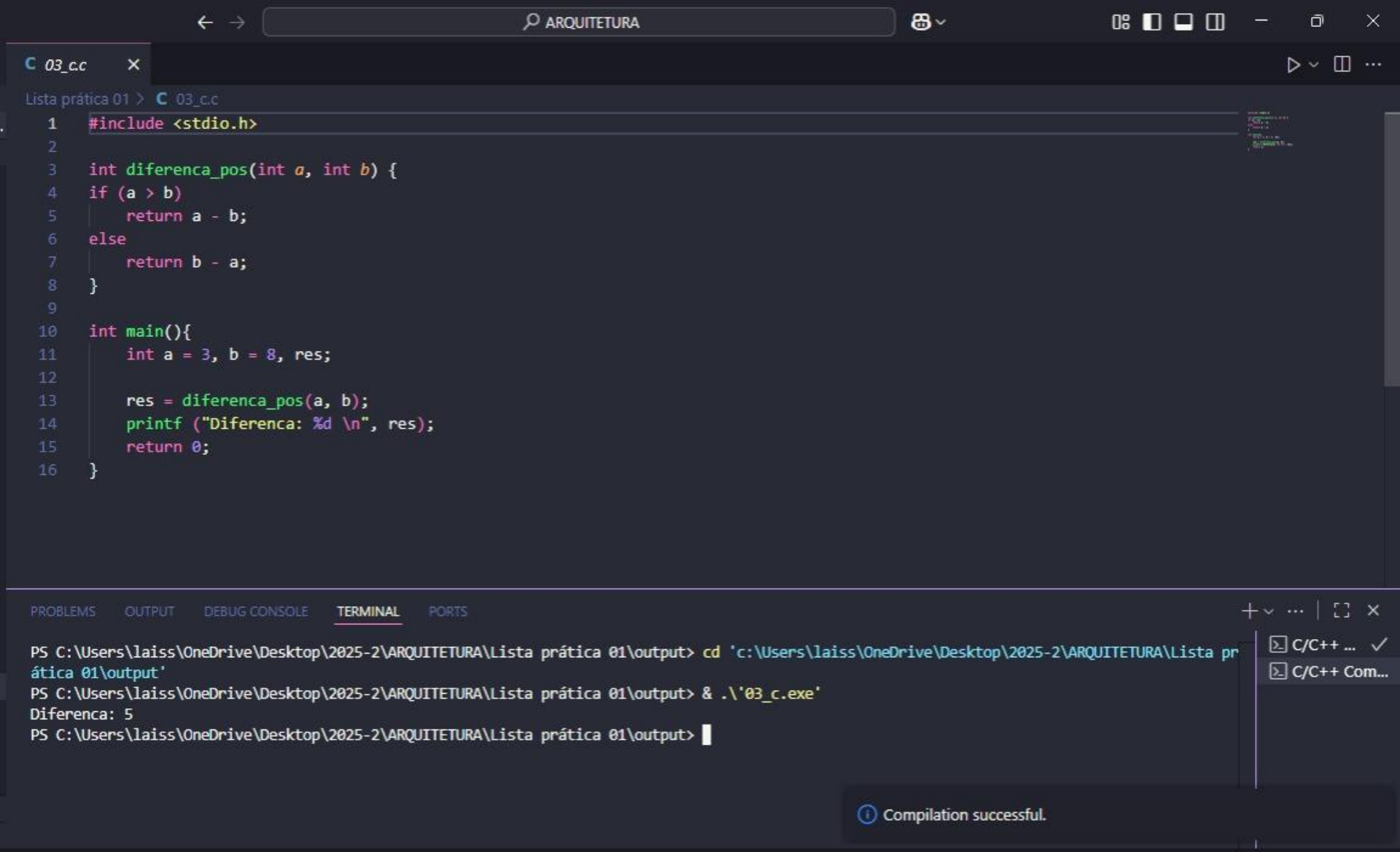
Mars Messages | Run I/O

Clear

```
5
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000005
$a1	5	0x00000005
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000001
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x0040000c
pc		0x00400020
hi		0x00000000
lo		0x00000000


ARQUITETURA
03_c.c
Lista prática 01 > 03_c.c
#include <stdio.h>
int diferenca_pos(int a, int b) {
if (a > b)
return a - b;
else
return b - a;
}
int main(){
int a = 3, b = 8, res;
res = diferenca_pos(a, b);
printf ("Diferenca: %d \n", res);
return 0;
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'03_c.exe'
Diferenca: 5
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
C/C++ ...
C/C++ Com...
Compilation successful.

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x24040003	addiu $4,$0,0x00000003	6:	li $a0, 3 # carrego 3 em a0
	0x00400004	0x24050008	addiu $5,$0,0x00000008	7:	li $a1 , 8 # carrego 8 em a1
	0x00400008	0x0c100008	jal 0x00400020	8:	jal max # cchamo a função
	0x0040000c	0x00022021	addu $4,$0,$2	9:	move $a0, $v0
	0x00400010	0x24020001	addiu $2,$0,0x00000001	10:	li $v0, 1
	0x00400014	0x0000000c	syscall	11:	syscall
	0x00400018	0x2402000a	addiu $2,$0,0x0000000a	12:	li $v0, 10
	0x0040001c	0x0000000c	syscall	13:	syscall
	0x00400020	0x0085482a	slt $9,$4,$5	16:	slt $t1, $a0, $a1 # t1 = 1 se a0 < a1
	0x00400024	0x15200002	bne $9,$0,0x00000002	17:	bne $t1, $zero, else # se t1 != 0, executo o else, ou seja, se a0 > a1...

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages | Run I/O

Clear

```
5
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000005
$a1	5	0x00000008
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000001
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x0040000c
pc		0x00400020
hi		0x00000000
lo		0x00000000

Edit | Execute

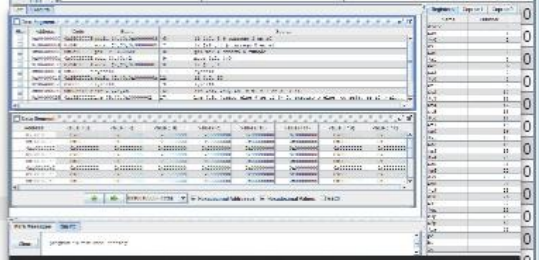
01_a.asm | mips1.asm | 01_b.asm | 01_c.asm | 02_a.asm* | 02_b.asm* | 03_c.asm | 02_c.asm | 03_a.asm | 03_b.asm

```
1        .data
2        .text
3        .globl main
4
5   main:
6        li $a0, 3 # carrego 3 em a0
7        li $a1 , 8 # carrego 8 em a1
8        jal max # cchamo a função
9        move $a0, $v0
10       li $v0, 1
11       syscall
12       li $v0, 10
13       syscall
14
15  max:
16       slt $t1, $a0, $a1 # t1 = 1 se a0 < a1
17       bne $t1, $zero, else # se t1 != 0, executo o else, ou seja, se a0 > a1 executo o else
18       sub $v0, $a0, $a1 # realizo a subtração
19       jr $ra # retorno
20  else:
21       sub $v0, $a1, $a0 # subtração
22       jr $ra # retorno
23
```

Line: 22 Column: 18 ☑ Show Line Numbers

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000005
$a1	5	0x00000008
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000001
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000



Mars Messages | Run I/O

Clear

```
5
-- program is finished running --
```

Ferramenta de Captura

O texto foi copiado para a área de transferência.
Salvo automaticamente na pasta de capturas de tela.

Marcar e compartilhar

03_d.c

Lista prática 01 > 03_d.c

```c
#include <stdio.h>

int valida_primeiro(int v[]) {
    if (v[0] > 10)
        return 1;
    return 0;
}

int main(){
    int vetor[3] = {11,12,1};
    int validade = valida_primeiro(vetor);
    printf("%d", validade);

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'03_d.exe'
1
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
```

C/C++ ...

C/C++ Com...

Compilation successful.

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x3c011001	lui $1,0x00001001	6:	la $a0, vetor # carrego o vetor em a0
	0x00400004	0x34240000	ori $4,$1,0x00000000		
	0x00400008	0x0c100008	jal 0x00400020	7:	jal valida_primeiro # chamo a função
	0x0040000c	0x00022021	addu $4,$0,$2	8:	move $a0, $v0 # coloco o retorno da função em a0
	0x00400010	0x24020001	addiu $2,$0,0x00000001	9:	li $v0, 1 # imprimo o resultado
	0x00400014	0x0000000c	syscall	10:	syscall
	0x00400018	0x2402000a	addiu $2,$0,0x0000000a	11:	li $v0, 10 # encerro
	0x0040001c	0x0000000c	syscall	12:	syscall
	0x00400020	0x8c8a0000	lw $10,0x00000000($4)	14:	lw $t2, 0($a0) # carrego a posição 0 do vetor
	0x00400024	0x2949000a	slti $9,$10,0x0000000a	15:	slti $t1, $t2, 10 # verifico de t2 < 10, se sim, t1 = 1 e caso contrár...

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x0000000b	0x0000000c	0x00000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages | Run I/O

Clear

```
1
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000001
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x0000000b
$t3	11	0x00000001
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x0040000c
pc		0x00400020
hi		0x00000000
lo		0x00000000

Edit | Execute

01_a.asm | mips1.asm | 01_b.asm | 01_c.asm | 02_a.asm* | 02_b.asm* | 03_c.asm | 02_c.asm | 03_a.asm | 03_b.asm | 03_d.asm

```
1         .data
2   vetor: .word 11, 12, 1
3         .text
4         .globl main
5   main:
6         la $a0, vetor # carrego o vetor em a0
7         jal valida_primeiro # chamo a função
8         move $a0, $v0 # coloco o retorno da função em a0
9         li $v0, 1 # imprimo o resultado
10        syscall
11        li $v0, 10 # encerro
12        syscall
13  valida_primeiro:
14        lw $t2, 0($a0) # carrego a posição 0 do vetor
15        slti $t1, $t2, 10 # verifico de t2 < 10, se sim, t1 = 1 e caso contrário. t1 = 0
16        bne $t1, $zero, else # se t1 != 0, else
17        li $t3, 1
18        move $v0, $t3 # coloco 1 no retorno
19        jr $ra # retorno 1, pois t2 > 10
20  else:
21        move $v0, $zero
22        jr $ra # retorno 0 se t2 < 10
23
```

Line: 22 Column: 31 ☑ Show Line Numbers

Mars Messages | Run I/O

Clear

```
1
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000001
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x0000000b
$t3	11	0x00000001
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x0040000c
pc		0x00400020
hi		0x00000000
lo		0x00000000

ARQUITETURA

03_e.c

Lista prática 01 > 03_e.c

```c
#include <stdio.h>

int conta_pares(int v[], int n) {
    int cont = 0;
    for (int i = 0; i < n; i++)
        if (v[i] % 2 == 0)
            cont++;
    return cont;
}

int main(){
    int vetor[3] = {2,4,7}, n = 3;
    int resultado = conta_pares(vetor, n);
    printf("contagem: %d", resultado);
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'03_e.exe'
contagem: 2
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
```

C/C++ ...

C/C++ Com...

pile Compile & Run Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} C

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui $1,0x00001001	6: la $a0, vetor # carrego o vetor em a0
	0x00400004	0x34240000	ori $4,$1,0x00000000	
	0x00400008	0x24050003	addiu $5,$0,0x00000003	7: li $a1, 3 # carrego 3 em a1
	0x0040000c	0x0c100009	jal 0x00400024	8: jal conta_pares #executo a função
	0x00400010	0x00022021	addu $4,$0,$2	9: move $a0, $v0 # coloco o resultado em a0
	0x00400014	0x24020001	addiu $2,$0,0x00000001	10: li $v0, 1 # imprimmo o resultado
	0x00400018	0x0000000c	syscall	11: syscall
	0x0040001c	0x2402000a	addiu $2,$0,0x0000000a	12: li $v0, 10 # encerro
	0x00400020	0x0000000c	syscall	13: syscall
	0x00400024	0x24090000	addiu $9,$0,0x00000000	15: li $t1, 0 # corresponde a cont = 0 em c

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000002	0x00000004	0x00000007	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages | Run I/O

Clear

```
2
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000002
$a1	5	0x00000003
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000003
$t1	9	0x00000002
$t2	10	0x00000000
$t3	11	0x00000008
$t4	12	0x10010008
$t5	13	0x00000007
$t6	14	0x00000001
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x00400024
hi		0x00000000
lo		0x00000000

Edit | Execute

01_a.asm | mips1.asm | 01_b.asm | 01_c.asm | 02_a.asm* | 02_b.asm* | 03_c.asm | 02_c.asm | 03_a.asm | 03_b.asm | 03_d.asm | 03_e.asm

```
5  main:
6      la $a0, vetor  # carrego o vetor em a0
7      li $a1, 3      # carrego 3 em a1
8      jal conta_pares  #executo a função
9      move $a0, $v0 # coloco o resultado em a0
10     li $v0, 1 # imprimmo o resultado
11     syscall
12     li $v0, 10 # encerro
13     syscall
14 conta_pares:  li $t1, 0      # corresponde a cont = 0 em c
15         li $t0, 0 # i = 0
16 loop: slt $t2, $t0, $a1   # verifico a condição: t2 = 1 se i < a1
17     beq $t2, $zero, fim  # se i > a1, vou para o fim
18     sll $t3, $t0, 2 # desloco o tamano de 1 int (4)
19     add $t4, $a0, $t3  # calculo a "nova posição"
20     lw  $t5, 0($t4) #carrego a nova posição
21     andi $t6, $t5, 1
22     bne  $t6, $zero, next
23     addi $t1, $t1, 1  # cont++
24 next:
25     addi $t0, $t0, 1 #i++
26     j loop #continuo o loop
27 fim:
28     move $v0, $t1
29     jr $ra # retorno
```

Line: 5 Column: 6 ☑ Show Line Numbers

Mars Messages | Run I/O

Clear

```
2
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000002
$a1	5	0x00000003
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000003
$t1	9	0x00000002
$t2	10	0x00000000
$t3	11	0x00000008
$t4	12	0x10010008
$t5	13	0x00000007
$t6	14	0x00000001
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x00400024
hi		0x00000000
lo		0x00000000

ARQUITETURA

03_f.c

Lista prática 01 > 03_f.c

```c
#include <stdio.h>

int multiplica_ate_n(int v[], int n) {
    int res = 1, i = 0;
    while (i < n) {
        res *= v[i];
        i++;
}
return res;
}

int main(){
    int vet[5] = {1,2,3,5,6}, n =4;
    int res = multiplica_ate_n(vet, n);
    printf("%d", res);

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> cd 'c:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output'
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output> & .\'03_f.exe'
30
PS C:\Users\laiss\OneDrive\Desktop\2025-2\ARQUITETURA\Lista prática 01\output>
```

C/C++ ...

C/C++ Com...

Compilation successful.

ile Compile & Run Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} C

Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x3c011001	lui $1,0x00001001	6:	la $a0, vet # carrego o vetor em a0
	0x00400004	0x34240000	ori $4,$1,0x00000000		
	0x00400008	0x24050004	addiu $5,$0,0x00000004	7:	li $al, 4 # 4 me al
	0x0040000c	0x0c100009	jal 0x00400024	8:	jal multiplica_ate_n # função
	0x00400010	0x00022021	addu $4,$0,$2	9:	move $a0, $v0 # colooc o retorno da função em a0
	0x00400014	0x24020001	addiu $2,$0,0x00000001	10:	li $v0, 1 #para imprimir esse retono
	0x00400018	0x0000000c	syscall	11:	syscall
	0x0040001c	0x2402000a	addiu $2,$0,0x0000000a	12:	li $v0, 10 # para encerrar
	0x00400020	0x0000000c	syscall	13:	syscall
	0x00400024	0x24080000	addiu $8,$0,0x00000000	15:	li $t0, 0 # t0 = 0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000001	0x00000002	0x00000003	0x00000005	0x00000006	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Mars Messages | Run I/O

Clear

```
30
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000001e
$a1	5	0x00000004
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000004
$t1	9	0x0000001e
$t2	10	0x0000000c
$t3	11	0x1001000c
$t4	12	0x00000005
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x00400024
hi		0x00000000
lo		0x0000001e

Edit | Execute

02_a.asm* | 02_b.asm* | 03_c.asm | 02_c.asm | 03_a.asm | 03_b.asm | 03_d.asm | 03_e.asm | 03_f.asm

01_a.asm | mips1.asm | 01_b.asm | 01_c.asm

```
5  main:
6      la $a0, vet # carrego o vetor em a0
7      li $a1, 4 # 4 me a1
8      jal multiplica_ate_n # função
9      move $a0, $v0 # colooc o retorno da função em a0
10     li $v0, 1 #para imprimir esse retono
11     syscall
12     li $v0, 10 # para encerrar
13     syscall
14 multiplica_ate_n: # execução da função
15     li $t0, 0 # t0 = 0
16     li $t1, 1 # t1 = 1
17 loop:
18     bge $t0, $a1, fim
19     sll $t2, $t0, 2 # desloco 2^2 posições
20     add $t3, $a0, $t2
21     lw $t4, 0($t3) #nova posição
22     mul $t1, $t1, $t4 # realizo a multiplicação
23     addi $t0, $t0, 1 # i++
24     j loop
25 fim:
26     move $v0, $t1
27     jr $ra
28
```

Line: 27 Column: 11 ☑ Show Line Numbers

Mars Messages | Run I/O

Clear

```
30
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000001e
$a1	5	0x00000004
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000004
$t1	9	0x0000001e
$t2	10	0x0000000c
$t3	11	0x1001000c
$t4	12	0x00000005
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x00400024
hi		0x00000000
lo		0x0000001e