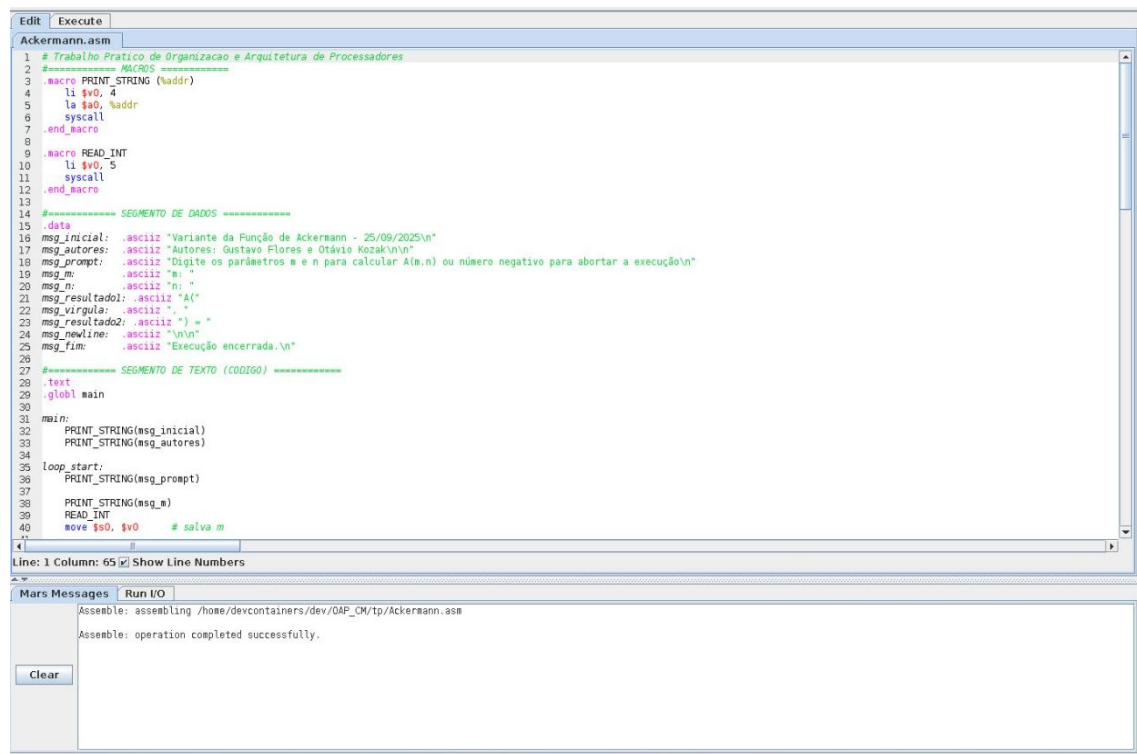


Área de código compilada (MARS):



```
1 # Trabalho Prático de Organização e Arquitetura de Processadores
2 #----- MACROS -----
3 .macro PRINT_STRING (addr)
4     li $v0, 4
5     la $a0, addr
6     syscall
7 .end_macro
8
9 .macro READ_INT
10    li $v0, 5
11    syscall
12 .end_macro
13
14 #----- SEGMENTO DE DADOS -----
15 .data
16 msg_inicial: .asciiz "Variante da Função de Ackermann - 25/09/2025\n"
17 msg_autores: .asciiz "Autores: Gustavo Flores e Otávio Kozak\n\n"
18 msg_prompt:  .asciiz "Digite os parâmetros m e n para calcular A(m,n) ou número negativo para abortar a execução\n"
19 msg_m:       .asciiz "m: "
20 msg_n:       .asciiz "n: "
21 msg_resultado: .asciiz "A("
22 msg_virgula:  .asciiz ", "
23 msg_resultado2: .asciiz ")" = "
24 msg_newline:  .asciiz "\n\n"
25 msg_fim:      .asciiz "Execução encerrada.\n"
26
27 #----- SEGMENTO DE TEXTO (CÓDIGO) -----
28 .text
29 .globl main
30
31 main:
32     PRINT_STRING(msg_inicial)
33     PRINT_STRING(msg_autores)
34
35 loop_start:
36     PRINT_STRING(msg_prompt)
37
38     PRINT_STRING(msg_m)
39     READ_INT
40     move $s0, $v0 # salva m
```

Line: 1 Column: 65 ☒ Show Line Numbers

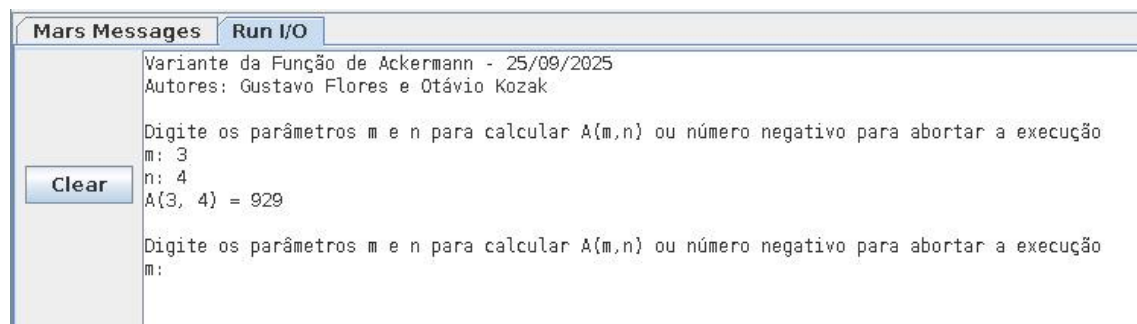
Mars Messages Run I/O

Assemble: assembling /home/devcontainers/dev/OAP_CM/tp/Ackermann.asm

Assemble: operation completed successfully.

Clear

Execução + Estado dos Registradores:



Mars Messages Run I/O

Variante da Função de Ackermann - 25/09/2025
Autores: Gustavo Flores e Otávio Kozak

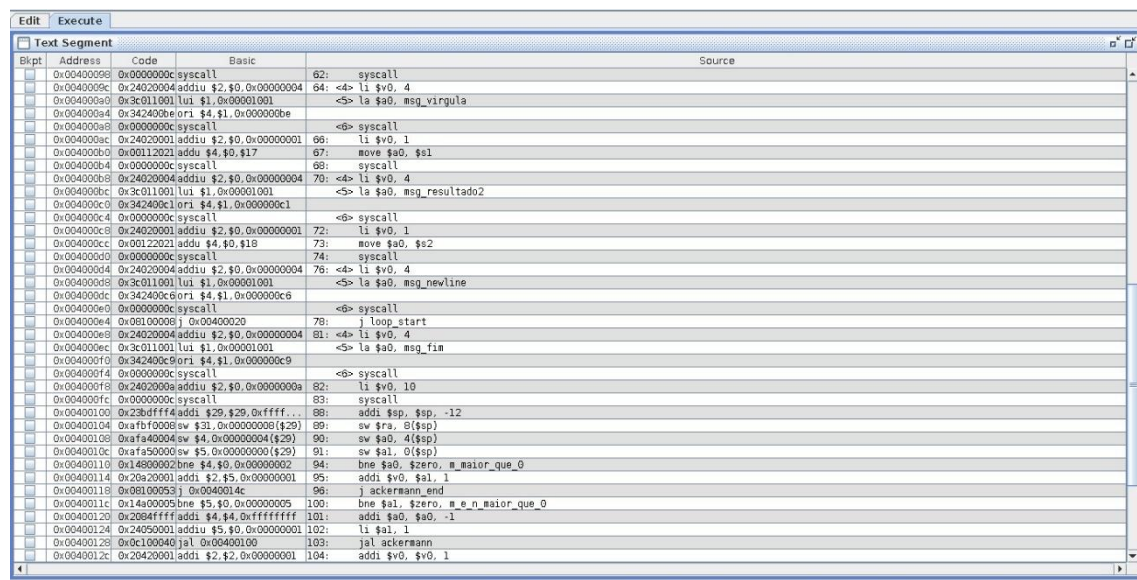
Digite os parâmetros m e n para calcular A(m,n) ou número negativo para abortar a execução

m: 3
n: 4
A(3, 4) = 929

Digite os parâmetros m e n para calcular A(m,n) ou número negativo para abortar a execução

m:

Clear



Bkpt	Address	Code	Basic	Source
	0x00400096	0x0000000c	syscall	62: syscall
	0x0040009c	0x24020004	addiu \$2,\$0,0x00000004	64: <<> li \$v0, 4
	0x0040009e	0x3c011001	lui \$1,0x000001001	<<> la \$a0, msg_virgula
	0x004000a4	0x342400be	ori \$4,\$1,0x000000be	
	0x004000a8	0x0000000c	syscall	<<> syscall
	0x004000ac	0x24020001	addiu \$2,\$0,0x00000001	66: li \$v0, 1
	0x004000b0	0x00112021	addiu \$4,\$0,\$17	67: move \$a0, \$s1
	0x004000b4	0x0000000c	syscall	68: syscall
	0x004000b8	0x24020004	addiu \$2,\$0,0x00000004	70: <<> li \$v0, 4
	0x004000bc	0x3c011001	lui \$1,0x000001001	<<> la \$a0, msg_resultado2
	0x004000c0	0x342400c1	ori \$4,\$1,0x000000c1	
	0x004000c4	0x0000000c	syscall	<<> syscall
	0x004000c8	0x24020001	addiu \$2,\$0,0x00000001	72: li \$v0, 1
	0x004000cc	0x00122021	addiu \$4,\$0,\$18	73: move \$a0, \$s2
	0x004000d0	0x0000000c	syscall	74: syscall
	0x004000d4	0x24020004	addiu \$2,\$0,0x00000004	76: <<> li \$v0, 4
	0x004000d8	0x3c011001	lui \$1,0x000001001	<<> la \$a0, msg_newline
	0x004000dc	0x342400c6	ori \$4,\$1,0x000000c6	
	0x004000e0	0x0000000c	syscall	<<> syscall
	0x004000e4	0x00100008	j 0x00400020	78: j loop_start
	0x004000e8	0x24020004	addiu \$2,\$0,0x00000004	81: <<> li \$v0, 4
	0x004000ec	0x3c011001	lui \$1,0x000001001	<<> la \$a0, msg_fim
	0x004000f0	0x342400c9	ori \$4,\$1,0x000000c9	
	0x004000f4	0x0000000c	syscall	<<> syscall
	0x004000f8	0x2402000a	addiu \$2,\$0,0x0000000a	82: li \$v0, 10
	0x004000fc	0x0000000c	syscall	83: syscall
	0x00400100	0x2b24ffff	addi \$29,\$29,0xffff...	88: addi \$sp, \$sp, -12
	0x00400104	0xaf100004	sw \$31,0x00000004(\$29)	89: sw \$ra, 0(\$sp)
	0x00400108	0xaf140004	sw \$4,0x00000004(\$29)	90: sw \$a0, 4(\$sp)
	0x0040010c	0xaf150000	sw \$5,0x00000000(\$29)	91: sw \$a1, 0(\$sp)
	0x00400110	0x14800002	bne \$4,\$0,0x00000002	94: bne \$a0, \$zero, m_maior_que_0
	0x00400114	0x20a20001	addi \$2,\$5,0x00000001	95: addi \$v0, \$a1, 1
	0x00400118	0x00100003	j 0x00400020	96: j ackermann_end
	0x0040011c	0x14a00005	bne \$5,\$0,0x00000005	100: bne \$a1, \$zero, m_e_n_maior_que_0
	0x00400120	0x20a4ffff	addi \$4,\$4,0xffff...	101: addi \$a0, \$a0, -1
	0x00400124	0x24050001	addiu \$5,\$0,0x00000001	102: li \$a1, 1
	0x00400128	0xc100940	jal 0x00400100	103: jal ackermann
	0x0040012c	0x20420001	addi \$2,\$2,0x00000001	104: addi \$v0, \$v0, 1

Edit		Execute		
Text Segment				
Bkpt	Address	Code	Basic	Source
	0x04000090	0x0000000c	syscall	62: syscall
	0x0400009c	0x24020004	addiu \$2,\$0,0x00000004	64: << li \$v0, 4
	0x040000a0	0x3c011001	lui \$1,0x00001001	<> la \$a0, msg_virgula
	0x040000a4	0x324200e0	ori \$4,\$1,0x000000be	
	0x040000a8	0x0000000c	syscall	<> syscall
	0x040000ac	0x24020001	addiu \$2,\$0,0x00000001	66: li \$v0, 1
	0x040000b0	0x00112021	addu \$4,\$0,\$17	67: move \$a0, \$a1
	0x040000b4	0x0000000c	syscall	68: syscall
	0x040000b8	0x24020004	addiu \$2,\$0,0x00000004	70: << li \$v0, 4
	0x040000bc	0x3c011001	lui \$1,0x00001001	<> la \$a0, msg_resultado2
	0x040000c0	0x324200e0	ori \$4,\$1,0x000000c1	
	0x040000c4	0x0000000c	syscall	<> syscall
	0x040000c8	0x24020001	addiu \$2,\$0,0x00000001	72: li \$v0, 1
	0x040000cc	0x00122021	addu \$4,\$0,\$18	73: move \$a0, \$a2
	0x040000d0	0x0000000c	syscall	74: syscall
	0x040000d4	0x24020004	addiu \$2,\$0,0x00000004	76: << li \$v0, 4
	0x040000d8	0x3c011001	lui \$1,0x00001001	<> la \$a0, msg_newline
	0x040000dc	0x324200e0	ori \$4,\$1,0x000000c6	
	0x040000e0	0x0000000c	syscall	<> syscall
	0x040000e4	0x00100050	j 0x04000020	78: j loop_start
	0x040000e8	0x24020004	addiu \$2,\$0,0x00000004	81: << li \$v0, 4
	0x040000ec	0x3c011001	lui \$1,0x00001001	<> la \$a0, msg_fim
	0x040000f0	0x324200e0	ori \$4,\$1,0x000000c9	
	0x040000f4	0x0000000c	syscall	<> syscall
	0x040000f8	0x2402000a	addiu \$2,\$0,0x0000000a	82: li \$v0, 10
	0x040000fc	0x0000000c	syscall	83: syscall
	0x04000100	0x23dffff4	addi \$29,\$29,0xffff...	88: addi \$sp, \$sp, -12
	0x04000104	0xafbf0008	sw \$31,0x00000008(\$29)	89: sw \$ra, 0(\$sp)
	0x04000108	0xafaf0004	sw \$4,0x00000004(\$29)	90: sw \$a0, 4(\$sp)
	0x0400010c	0xaf550000	sw \$5,0x00000000(\$29)	91: sw \$a1, 0(\$sp)
	0x04000110	0x14500002	bne \$a0,\$zero,m_maior_que_0	94: bne \$a0, \$zero, m_maior_que_0
	0x04000114	0x20a20001	addi \$2,\$5,0x00000001	95: addi \$v0, \$a1, 1
	0x04000118	0x00100053	j 0x0400014c	96: j ackermann_end
	0x0400011c	0x14a00005	bne \$5,\$0,0x00000005	100: bne \$a1, \$zero, m_e_n_maior_que_0
	0x04000120	0x208af0ff	addi \$4,\$4,0xfffffff	101: addi \$a0, \$a0, -1
	0x04000124	0x24050001	addiu \$5,\$0,0x00000001	102: li \$a1, 1
	0x04000128	0x0c100940	jal 0x04000100	103: jal ackermann
	0x0400012c	0x20420001	addi \$2,\$2,0x00000001	104: addi \$v0, \$v0, 1

0x04000098	ori \$4,\$1,0x000000c4	105:	j ackermann_end	
0x0400009c	addi \$a1,\$a1,-1	109:	addi \$a1, \$a1, -1	
0x040000a0	jal ackermann	110:	jal ackermann	
0x040000a4	addiu \$5,\$0,\$2	112:	move \$a1, \$v0 # resultado vira novo n	
0x040000a8	lw \$4,0(\$29)	113:	lw \$a0, 4(\$sp) # recupera n original	
0x040000ac	addi \$a0,\$a0,-1	114:	addi \$a0, \$a0, -1	
0x040000b0	jal ackermann	116:	jal ackermann	
0x040000b4	lw \$5,0(\$29)	120:	lw \$a1, 0(\$sp)	
0x040000b8	lw \$4,0(\$29)	121:	lw \$a0, 4(\$sp)	
0x040000bc	lw \$31,0(\$29)	122:	lw \$ra, 0(\$sp)	
0x040000c0	addi \$sp,\$sp,12	123:	addi \$sp, \$sp, 12	
0x040000c4	jr \$31	129:	jr \$ra	

Registers		Coproc 1	Coproc 0
Name	Number	Value	
\$zero	0	0x00000000	
\$at	1	0x10010000	
\$v0	2	0x0000000a	
\$v1	3	0x00000000	
\$a0	4	0x100100c9	
\$a1	5	0x00000004	
\$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	0xffffffff	
\$s1	17	0xfffffffffe	
\$s2	18	0x000003a1	
\$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	0x7ffefffc	
\$fp	30	0x00000000	
\$ra	31	0x0040007c	
pc		0x00400100	
hi		0x00000000	
lo		0x00000000	

Algoritmo + execução em Java:

```
public static long ackermann(long m, long n) {  
    if (m == 0) {  
        return n + INCREMENTO_BASE;  
    }  
    if (m > 0 && n == 0) {  
        return ackermann(m - 1, VALOR_INICIAL) + 1;  
    }  
    if (m > 0 && n > 0) {  
        return ackermann(m - 1, ackermann(m, n - 1));  
    }  
    return 0;  
}
```

Variante da Função de Ackermann - 25/09/2025

Autores: Gustavo Flores e Otávio Kozak

- Digite os parâmetros m e n para calcular A(m, n).
- Insira um número negativo para abortar a execução.

Insira m:

3

Insira n:

4

Função: A(3, 4)

A(3, 4) = 929

- Digite os parâmetros m e n para calcular A(m, n).
- Insira um número negativo para abortar a execução.

Insira m: