

Trabalho 1 Organização e Arquitetura de Processadores

Thiago Zilberknop e Leonardo Chou da Rosa

Código em Alto Nível (c++):

```
altonivel.cpp X
Arquitetura de Processadores > Trabalho 1 > altonivel.cpp > main()
1 // Trabalho 1 Organização e Arquitetura de Processadores
2 // Thiago Zilberknop, Leonardo Chou da Rosa
3
4 #include <iostream>
5
6 using namespace std;
7
8 int funcAckermann(int m, int n) {
9     if (m == 0) {
10         return n + 1;
11     }
12     if (m > 0 && n == 0) {
13         return funcAckermann(m - 1, 1);
14     }
15     if (m > 0 && n > 0) {
16         return funcAckermann(m-1, funcAckermann(m, n-1));
17     }
18     return 0;
19 }
20
21 int main () {
22     int m = 0;
23     int n = 0;
24     cout << "Digite m: ";
25     cin >> m;
26     cout << "Digite n: ";
27     cin >> n;
28     cout << endl;
29     cout << "A(" << m << ", " << n << ")" << " = " << funcAckermann(m, n) << endl;
30 }
```

O código em alto nível executa a função Ackermann como descrito na proposta do documento:

- 1) O programa pede os valores de m e n
- 2) O programa imprime o valor recebido pela função a partir dos números recebidos

A função é feita da seguinte forma:

- Primeira comparação: se o valor de m é igual à 0, retorna o valor de n + 1
- Segunda comparação: se o valor de m é maior que 0 e o valor de n é igual à 0, chama a função recursivamente com o valor de m decrementado e o valor de n como constante 1
- Terceira comparação: se o valor de m e o valor de n são maiores que 1, chama a função recursivamente com o valor de m decrementado e outra chamada recursiva com o valor de m atual e o valor de n decrementado

Em Assembly, o código fica da seguinte forma:

```
Edit Execute
mips1.asm
1 .data
2 programa: .asciiz "Programa Ackermann\n"
3 componentes: .asciiz "Thiago Zilberknop, Leonardo Chou da Rosa\n"
4 str_1: .asciiz "Digite um valor positivo para executar o programa ou um valor negativo para encerrar.\n"
5 str_2: .asciiz "A("
6 str_3: .asciiz ", "
7 str_4: .asciiz ")"
8 str_5: .asciiz "\n"
9
10 .macro print()
11     li $v0, 4          # Comando de print
12     la $a0, str_2      # print(str_2)
13     syscall
14     li $v0, 1
15     add $a0, $zero, $t0 # print(m)
16     syscall
17     li $v0, 4
18     la $a0, str_3      # print(s
19     syscall
20     li $v0, 1
21     add $a0, $zero, $t1
22     syscall
23     li $v0, 4
24     la $a0, str_4
25     syscall
26     li $v0, 1
27     add $a0, $zero, $t3
28     syscall
29     li $v0, 4
30     la $a0, str_5
31     syscall
32 .end_macro
33
34 .text
35 .globl main
Line: 1 Column: 1 Show Line Numbers
```

```
Edit Execute
mips1.asm
34
35 .text
36 .globl main
37
38 main:    li $v0, 4          # Impressao do titulo, componentes
39         la $a0, programa
40         syscall
41         la $a0, componentes
42         syscall
43
44 inicio:  la $a0, str_1
45         syscall
46
47         li $v0, 5          # Leitura de M
48         syscall
49         blt $v0, $zero, end
50         move $t0, $v0
51         li $v0, 5          # Leitura de N
52         syscall
53         blt $v0, $zero, end
54         move $t1, $v0
55
56         addi $sp, $sp, -12 # Abre 3 posições na pilha
57         sw $t0, 0($sp)    # Guarda M na pilha
58         sw $t1, 4($sp)    # Guarda N na pilha
59         jal ackermann    # Chamada da func recursiva
60         move $t3, $v0      # Guarda resultado de ackermann em $t3
61         print()           # macro de print
62         j inicio          # reinicia o programa
63
64 end:     li $v0, 10
65         syscall
66
67 # -- Funcao Ackermann --
68
69 ackermann: sw $ra, 8($sp) # Guarda o endereço de retorno da ial
Line: 1 Column: 1 Show Line Numbers
```

```

Edit Execute
mips1.asm
67 # -- Funcao Ackermann --
68
69 ackermann:  sw    $ra, 8($sp)      # Guarda o endereço de retorno do jal
70             lw    $s0, 0($sp)     # Carrega M em $s0
71             lw    $s1, 4($sp)     # Carrega N em $s1
72             beq    $s0, $zero, m_zero  # M = 0
73             beq    $s1, $zero, n_zero  # M > 0 && N = 0
74             j      m_n             # M > 0 && N > 0
75
76
77 m_zero:     addi    $v0, $s1, 1     # retorno = n + 1
78             j      retorno
79
80 n_zero:     addi    $s0, $s0, -1     # M - 1
81             add     $s1, $zero, 1    # N = 1
82             addi    $sp, $sp, -12    # Aloca 3 espaços na pilha
83             sw     $s0, 0($sp)      # M
84             sw     $s1, 4($sp)      # N
85             jal     ackermann       # Chamada recursiva
86             j      retorno
87
88
89 m_n:        addi    $s1, $s1, -1     # N - 1
90             addi    $sp, $sp, -12    # Abre 3 espaços na pilha
91             sw     $s0, 0($sp)      # M
92             sw     $s1, 4($sp)      # N
93             jal     ackermann       # Chamada recursiva [ A(m, n-1) ]
94             lw     $s0, 0($sp)      # Resgata o valor original de M
95             addi    $s0, $s0, -1     # M - 1
96             addi    $sp, $sp, -12    # Abre 3 espaços na pilha
97             sw     $s0, 0($sp)      # M
98             sw     $v0, 4($sp)      # N ( resultado da chamada recursiva )
99             jal     ackermann       # Chamada recursiva [ A(m-1, A(m, n-1) ]
100            j      retorno
101
102
103
104
105 retorno:   lw     $ra, 8($sp)      # Resgata end. de retorno
106             addi    $sp, $sp, 12     # Desaloca a pilha
107             jr     $ra             # Retorna
108

```

Line: 1 Column: 1 ☒ Show Line Numbers

```

97         sw    $s0, 0($sp)      # M
98         sw    $v0, 4($sp)      # N ( resultado da chamada recursiva )
99         jal    ackermann       # Chamada recursiva [ A(m-1, A(m, n-1) ]
100        j      retorno
101
102
103
104
105 retorno:  lw    $ra, 8($sp)      # Resgata end. de retorno
106          addi   $sp, $sp, 12     # Desaloca a pilha
107          jr     $ra             # Retorna
108

```

Line: 1 Column: 1 ☒ Show Line Numbers

Linhas 1-8:

Contém todas as strings que serão utilizadas na impressão do programa

Linhas 10-32:

Macro que imprime o resultado da função Ackermann junto com as strings nomeadas str_1, str_2, etc

Linhas 38-42:

Imprime o título do projeto e integrantes do grupo

Linhas 44-54:

Faz a leitura das variáveis m (salva em $\$t0$) e n (salva em $\$t1$)

Linhas 56-62:

Abre três posições na pilha e guarda os valores de m ($\$t0$) e n ($\$t1$) nas primeiras duas posições;

Executa a função Ackermann

Linhas 69-74:

Resgata os valores de m ($\$s0$) e n ($\$s1$) da pilha e salva o endereço de retorno que está em $\$ra$ na terceira posição que foi aberta anteriormente; Executa as três comparações da função

Linhas 77-78:

Executa as condições da função Ackermann quando o valor de m for igual a 0 (retorna o valor de $n + 1$)

Linhas 80-86:

Executa as condições da função Ackermann quando o valor de n for igual a 0 (decrementa m por 1, chama a função recursivamente com os valores de $A(m-1, 1)$). Para a chamada recursiva, três posições são abertas na pilha novamente e os valores de m ($\$s0$) e n ($\$s1$) são salvos nas primeiras duas novas posições

Linhas 89-100:

Executa as condições da função Ackermann quando os valores de m e n são maiores que 0 (chama a função recursiva $A(m, n-1)$ e usando esse valor chama a função $A(m-1, A(m, n-1))$). Para a chamada recursiva, três posições são abertas na pilha novamente e os valores de m ($\$s0$) e n ($\$s1$) são salvos nas primeiras duas novas posições

Linhas 105-107:

Desaloca a pilha, resgata o endereço de retorno posto na terceira posição da pilha no início de uma execução da função e retorna para o mesmo endereço

Área de código compilada:

| Edit | Execute | | | | |
|--------------|------------|------------|--------------------|---|--|
| Text Segment | | | | | |
| Bkpt | Address | Code | Basic | Source | |
| | 0x00400000 | 0x24020004 | addiu \$2,\$0,4 | 38: main: li \$v0, 4 # Impressao do titulo, componentes | |
| | 0x00400004 | 0x3c011001 | lui \$1,4097 | 39: la \$a0, programa | |
| | 0x00400008 | 0x34240000 | ori \$4,\$1,0 | | |
| | 0x0040000c | 0x0000000c | syscall | 40: syscall | |
| | 0x00400010 | 0x3c011001 | lui \$1,4097 | 41: la \$a0, componentes | |
| | 0x00400014 | 0x34240014 | ori \$4,\$1,20 | | |
| | 0x00400018 | 0x0000000c | syscall | 42: syscall | |
| | 0x0040001c | 0x3c011001 | lui \$1,4097 | 44: inicio: la \$a0, str_1 | |
| | 0x00400020 | 0x3424002c | ori \$4,\$1,44 | | |
| | 0x00400024 | 0x0000000c | syscall | 45: syscall | |
| | 0x00400028 | 0x24020005 | addiu \$2,\$0,5 | 47: li \$v0, 5 # Leitura de M | |
| | 0x0040002c | 0x0000000c | syscall | 48: syscall | |
| | 0x00400030 | 0x0040082a | slt \$1,\$2,\$0 | 49: blt \$v0, \$zero, end | |
| | 0x00400034 | 0x14200025 | bne \$1,\$0,37 | | |
| | 0x00400038 | 0x00024021 | addu \$8,\$0,\$2 | 50: move \$t0, \$v0 | |
| | 0x0040003c | 0x24020005 | addiu \$2,\$0,5 | 51: li \$v0, 5 # Leitura de N | |
| | 0x00400040 | 0x0000000c | syscall | 52: syscall | |
| | 0x00400044 | 0x0040082a | slt \$1,\$2,\$0 | 53: blt \$v0, \$zero, end | |
| | 0x00400048 | 0x14200020 | bne \$1,\$0,32 | | |
| | 0x0040004c | 0x00024821 | addu \$9,\$0,\$2 | 54: move \$t1, \$v0 | |
| | 0x00400050 | 0x23bdfbf4 | addi \$29,\$29,-12 | 56: addi \$sp, \$sp, -12 # Abre 3 posições na pilha | |
| | 0x00400054 | 0xafaf8000 | sw \$8,0(\$29) | 57: sw \$t0, 0(\$sp) # Guarda M na pilha | |
| | 0x00400058 | 0xafaf9004 | sw \$9,4(\$29) | 58: sw \$t1, 4(\$sp) # Guarda N na pilha | |
| | 0x0040005c | 0x0c100035 | jal 0x004000d4 | 59: jal ackermann # Chamada da func recursiva | |
| | 0x00400060 | 0x00025821 | addu \$11,\$0,\$2 | 60: move \$t3, \$v0 # Guarda resultado de ackermann em \$t3 | |
| | 0x00400064 | 0x24020004 | addiu \$2,\$0,4 | 61: <11> li \$v0, 4 # Comando de print | |
| | 0x00400068 | 0x3c011001 | lui \$1,4097 | <12> la \$a0, str_2 # print(str_2) | |
| | 0x0040006c | 0x34240083 | ori \$4,\$1,131 | | |
| | 0x00400070 | 0x0000000c | syscall | <13> syscall | |
| | 0x00400074 | 0x24020001 | addiu \$2,\$0,1 | <14> li \$v0, 1 | |
| | 0x00400078 | 0x00082020 | add \$4,\$0,\$8 | <15> add \$a0, \$zero, \$t0 # print(m) | |
| | 0x0040007c | 0x0000000c | syscall | <16> syscall | |
| | 0x00400080 | 0x24020004 | addiu \$2,\$0,4 | <17> li \$v0, 4 | |
| | 0x00400084 | 0x3c011001 | lui \$1,4097 | <18> la \$a0, str_3 # print(s | |
| | 0x00400088 | 0x34240086 | ori \$4,\$1,134 | | |
| | 0x0040008c | 0x0000000c | syscall | <19> syscall | |
| | 0x00400090 | 0x24020001 | addiu \$2,\$0,1 | <20> li \$v0, 1 | |
| | 0x00400094 | 0x00092020 | add \$4,\$0,\$9 | <21> add \$a0, \$zero, \$t1 | |

| Edit | Execute | | | |
|--------------------------|------------|------------|--------------------|---|
| Text Segment | | | | |
| Bkpt | Address | Code | Basic | Source |
| <input type="checkbox"/> | 0x00400090 | 0x24020001 | addiu \$2,\$0,1 | <20> li \$v0, 1 |
| <input type="checkbox"/> | 0x00400094 | 0x00092020 | add \$4,\$0,\$9 | <21> add \$a0, \$zero, \$t1 |
| <input type="checkbox"/> | 0x00400098 | 0x0000000c | syscall | <22> syscall |
| <input type="checkbox"/> | 0x0040009c | 0x24020004 | addiu \$2,\$0,4 | <23> li \$v0, 4 |
| <input type="checkbox"/> | 0x004000a0 | 0x3c011001 | lui \$1,4097 | <24> la \$a0, str_4 |
| <input type="checkbox"/> | 0x004000a4 | 0x34240089 | ori \$4,\$1,137 | |
| <input type="checkbox"/> | 0x004000a8 | 0x0000000c | syscall | <25> syscall |
| <input type="checkbox"/> | 0x004000ac | 0x24020001 | addiu \$2,\$0,1 | <26> li \$v0, 1 |
| <input type="checkbox"/> | 0x004000b0 | 0x000b2020 | add \$4,\$0,\$11 | <27> add \$a0, \$zero, \$t3 |
| <input type="checkbox"/> | 0x004000b4 | 0x0000000c | syscall | <28> syscall |
| <input type="checkbox"/> | 0x004000b8 | 0x24020004 | addiu \$2,\$0,4 | <29> li \$v0, 4 |
| <input type="checkbox"/> | 0x004000bc | 0x3c011001 | lui \$1,4097 | <30> la \$a0, str_5 |
| <input type="checkbox"/> | 0x004000c0 | 0x3424008e | ori \$4,\$1,142 | |
| <input type="checkbox"/> | 0x004000c4 | 0x0000000c | syscall | <31> syscall |
| <input type="checkbox"/> | 0x004000c8 | 0x08100007 | j 0x0040001c | 62: j inicio # reinicia o programa |
| <input type="checkbox"/> | 0x004000cc | 0x2402000a | addiu \$2,\$0,10 | 64: end: li \$v0, 10 |
| <input type="checkbox"/> | 0x004000d0 | 0x0000000c | syscall | 65: syscall |
| <input type="checkbox"/> | 0x004000d4 | 0xafbf0008 | sw \$31,8(\$29) | 69: ackermann: sw \$ra, 8(\$sp) # Guarda o endereço de retorno do jal |
| <input type="checkbox"/> | 0x004000d8 | 0x8fb00000 | lw \$16,0(\$29) | 70: lw \$s0, 0(\$sp) # Carrega M em \$s0 |
| <input type="checkbox"/> | 0x004000dc | 0x8fb10004 | lw \$17,4(\$29) | 71: lw \$s1, 4(\$sp) # Carrega N em \$s1 |
| <input type="checkbox"/> | 0x004000e0 | 0x12000002 | beq \$16,\$0,2 | 72: beq \$s0, \$zero, m_zero # M = 0 |
| <input type="checkbox"/> | 0x004000e4 | 0x12200003 | beq \$17,\$0,3 | 73: beq \$s1, \$zero, n_zero # M > 0 && N = 0 |
| <input type="checkbox"/> | 0x004000e8 | 0x08100044 | j 0x00400110 | 74: j m_n # M > 0 && N > 0 |
| <input type="checkbox"/> | 0x004000ec | 0x22220001 | addi \$2,\$17,1 | 77: m_zero: addi \$v0, \$s0, 1 # retorno = n + 1 |
| <input type="checkbox"/> | 0x004000f0 | 0x08100050 | j 0x00400140 | 78: j retorno |
| <input type="checkbox"/> | 0x004000f4 | 0x2210ffff | addi \$16,\$16,-1 | 80: n_zero: addi \$s0, \$s0, -1 # M - 1 |
| <input type="checkbox"/> | 0x004000f8 | 0x20110001 | addi \$17,\$0,1 | 81: add \$s1, \$zero, 1 # N = 1 |
| <input type="checkbox"/> | 0x004000fc | 0x23bdfbf4 | addi \$29,\$29,-12 | 82: addi \$sp, \$sp, -12 # Aloca 3 espaços na pilha |
| <input type="checkbox"/> | 0x00400100 | 0xafbf0000 | sw \$16,0(\$29) | 83: sw \$s0, 0(\$sp) # M |
| <input type="checkbox"/> | 0x00400104 | 0xafbf1004 | sw \$17,4(\$29) | 84: sw \$s1, 4(\$sp) # N |
| <input type="checkbox"/> | 0x00400108 | 0x0c100035 | jal 0x004000d4 | 85: jal ackermann # Chamada recursiva |
| <input type="checkbox"/> | 0x0040010c | 0x08100050 | j 0x00400140 | 86: j retorno |
| <input type="checkbox"/> | 0x00400110 | 0x2231ffff | addi \$17,\$17,-1 | 89: m_n: addi \$s1, \$s1, -1 # N - 1 |
| <input type="checkbox"/> | 0x00400114 | 0x23bdfbf4 | addi \$29,\$29,-12 | 90: addi \$sp, \$sp, -12 # Abre 3 espaços na pilha |
| <input type="checkbox"/> | 0x00400118 | 0xafbf0000 | sw \$16,0(\$29) | 91: sw \$s0, 0(\$sp) # M |
| <input type="checkbox"/> | 0x0040011c | 0xafbf1004 | sw \$17,4(\$29) | 92: sw \$s1, 4(\$sp) # N |
| <input type="checkbox"/> | 0x00400120 | 0x0c100035 | jal 0x004000d4 | 93: jal ackermann # Chamada recursiva [A(m, n-1)] |
| <input type="checkbox"/> | 0x00400124 | 0x8fb00000 | lw \$16,0(\$29) | 94: lw \$s0, 0(\$sp) # Resgata o valor original de M |

| | | | | | | | |
|--------------------------|------------|------------|--------------------|---------------|------|-----------------|--|
| <input type="checkbox"/> | 0x00400120 | 0x0c100035 | jal 0x004000d4 | 93: | jal | ackermann | # Chamada recursiva [A(m, n-1)] |
| <input type="checkbox"/> | 0x00400124 | 0x8fb00000 | lw \$l6,0(\$29) | 94: | lw | \$s0, 0(\$sp) | # Resgata o valor original de M |
| <input type="checkbox"/> | 0x00400128 | 0x2210ffff | addi \$l6,\$l6,-1 | 95: | addi | \$s0, \$s0, -1 | # M - 1 |
| <input type="checkbox"/> | 0x0040012c | 0x23bdfff4 | addi \$29,\$29,-12 | 96: | addi | \$sp, \$sp, -12 | # Abre 3 espaços na pilha |
| <input type="checkbox"/> | 0x00400130 | 0xafb00000 | sw \$l6,0(\$29) | 97: | sw | \$s0, 0(\$sp) | # M |
| <input type="checkbox"/> | 0x00400134 | 0xaf200004 | sw \$2,4(\$29) | 98: | sw | \$v0, 4(\$sp) | # N (resultado da chamada recursiva) |
| <input type="checkbox"/> | 0x00400138 | 0x0c100035 | jal 0x004000d4 | 99: | jal | ackermann | # Chamada recursiva [A(m-1, A(m, n-1)] |
| <input type="checkbox"/> | 0x0040013c | 0x08100050 | j 0x00400140 | 100: | j | retorno | |
| <input type="checkbox"/> | 0x00400140 | 0x8fbf0008 | lw \$31,8(\$29) | 105: retorno: | lw | \$ra, 8(\$sp) | # Resgata end. de retorno |
| <input type="checkbox"/> | 0x00400144 | 0x23bd000c | addi \$29,\$29,12 | 106: | addi | \$sp, \$sp, 12 | # Desaloca a pilha |
| <input type="checkbox"/> | 0x00400148 | 0x03e00008 | jr \$31 | 107: | jr | \$ra | # Retorna |

Estados dos registradores ao término de uma execução (A(1, 2)):

| Registers | Coproc 1 | Coproc 0 |
|-----------|----------|------------|
| Name | Number | Value |
| \$zero | 0 | 0 |
| \$at | 1 | 268500992 |
| \$v0 | 2 | 4 |
| \$v1 | 3 | 0 |
| \$a0 | 4 | 268501134 |
| \$a1 | 5 | 0 |
| \$a2 | 6 | 0 |
| \$a3 | 7 | 0 |
| \$t0 | 8 | 1 |
| \$t1 | 9 | 2 |
| \$t2 | 10 | 0 |
| \$t3 | 11 | 4 |
| \$t4 | 12 | 0 |
| \$t5 | 13 | 0 |
| \$t6 | 14 | 0 |
| \$t7 | 15 | 0 |
| \$s0 | 16 | 0 |
| \$s1 | 17 | 3 |
| \$s2 | 18 | 0 |
| \$s3 | 19 | 0 |
| \$s4 | 20 | 0 |
| \$s5 | 21 | 0 |
| \$s6 | 22 | 0 |
| \$s7 | 23 | 0 |
| \$t8 | 24 | 0 |
| \$t9 | 25 | 0 |
| \$k0 | 26 | 0 |
| \$k1 | 27 | 0 |
| \$gp | 28 | 268468224 |
| \$sp | 29 | 2147479548 |
| \$fp | 30 | 0 |
| \$ra | 31 | 4194400 |
| pc | | 4194500 |
| hi | | 0 |
| lo | | 0 |

| Registers | Coproc 1 | Coproc 0 |
|-----------|----------|-------------|
| Name | Number | Value |
| \$zero | 0 | 0x00000000 |
| \$at | 1 | 0x10010000 |
| \$v0 | 2 | 0x00000004 |
| \$v1 | 3 | 0x00000000 |
| \$a0 | 4 | 0x1001008e |
| \$a1 | 5 | 0x00000000 |
| \$a2 | 6 | 0x00000000 |
| \$a3 | 7 | 0x00000000 |
| \$t0 | 8 | 0x00000001 |
| \$t1 | 9 | 0x00000002 |
| \$t2 | 10 | 0x00000000 |
| \$t3 | 11 | 0x00000004 |
| \$t4 | 12 | 0x00000000 |
| \$t5 | 13 | 0x00000000 |
| \$t6 | 14 | 0x00000000 |
| \$t7 | 15 | 0x00000000 |
| \$s0 | 16 | 0x00000000 |
| \$s1 | 17 | 0x00000003 |
| \$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 | 0x7ffffeffc |
| \$fp | 30 | 0x00000000 |
| \$ra | 31 | 0x00400060 |
| pc | | 0x004000c8 |
| hi | | 0x00000000 |
| lo | | 0x00000000 |

Registradores antes do programa reiniciar

Área de pilha utilizada para a recursividade:

Entrada A(1, 2):

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|-------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| 0x7ffffefc0 | 0 | 0 | 0 | 0 | 1 | 4194572 | 0 | 2 |
| 0x7ffffefe0 | 4194596 | 1 | 1 | 4194596 | 1 | 2 | 4194400 | 0 |

Área de memória após todas as recursões

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) |
|-------------|------------|------------|------------|------------|-------------|-------------|-------------|
| 0x7ffffefe0 | 4194620 | 0 | 3 | 4194620 | 1 | 2 | 4194400 |

Área de memória antes do programa reiniciar

Exemplo de execução do programa:

| Text Segment | | | | | |
|--------------------------|------------|------------|--------------------|-------------|---|
| Bkpt | Address | Code | Basic | Source | |
| <input type="checkbox"/> | 0x00400018 | 0x0000000c | syscall | 42: | syscall |
| <input type="checkbox"/> | 0x0040001c | 0x3c011001 | lui \$1,4097 | 44: inicio: | la \$a0, str_1 |
| <input type="checkbox"/> | 0x00400020 | 0x3424002c | ori \$4,\$1,44 | | |
| <input type="checkbox"/> | 0x00400024 | 0x0000000c | syscall | 45: | syscall |
| <input type="checkbox"/> | 0x00400028 | 0x24020005 | addiu \$2,\$0,5 | 47: | li \$v0, 5 # Leitura de M |
| <input type="checkbox"/> | 0x0040002c | 0x0000000c | syscall | 48: | syscall |
| <input type="checkbox"/> | 0x00400030 | 0x0040082a | slt \$1,\$2,\$0 | 49: | blt \$v0, \$zero, end |
| <input type="checkbox"/> | 0x00400034 | 0x14200025 | bne \$1,\$0,37 | | |
| <input type="checkbox"/> | 0x00400038 | 0x00024021 | addu \$8,\$0,\$2 | 50: | move \$t0, \$v0 |
| <input type="checkbox"/> | 0x0040003c | 0x24020005 | addiu \$2,\$0,5 | 51: | li \$v0, 5 # Leitura de N |
| <input type="checkbox"/> | 0x00400040 | 0x0000000c | syscall | 52: | syscall |
| <input type="checkbox"/> | 0x00400044 | 0x0040082a | slt \$1,\$2,\$0 | 53: | blt \$v0, \$zero, end |
| <input type="checkbox"/> | 0x00400048 | 0x14200020 | bne \$1,\$0,32 | | |
| <input type="checkbox"/> | 0x0040004c | 0x00024821 | addu \$9,\$0,\$2 | 54: | move \$t1, \$v0 |
| <input type="checkbox"/> | 0x00400050 | 0x23bffff4 | addi \$29,\$29,-12 | 56: | addi \$sp, \$sp, -12 # Abre 3 posições na pilha |
| <input type="checkbox"/> | 0x00400054 | 0xafa80000 | sw \$8,0(\$29) | 57: | sw \$t0, 0(\$sp) # Guarda M na pilha |
| <input type="checkbox"/> | 0x00400058 | 0xafa90004 | sw \$9,4(\$29) | 58: | sw \$t1, 4(\$sp) # Guarda N na pilha |

O valor de M é lido (no caso da execução, o valor inserido é 1).

| Text Segment | | | | | |
|--------------------------|------------|------------|--------------------|----------|---|
| Bkpt | Address | Code | Basic | Source | |
| <input type="checkbox"/> | 0x00400028 | 0x24020005 | addiu \$2,\$0,5 | 47: | li \$v0, 5 # Leitura de M |
| <input type="checkbox"/> | 0x0040002c | 0x0000000c | syscall | 48: | syscall |
| <input type="checkbox"/> | 0x00400030 | 0x0040082a | slt \$1,\$2,\$0 | 49: | blt \$v0, \$zero, end |
| <input type="checkbox"/> | 0x00400034 | 0x14200025 | bne \$1,\$0,37 | | |
| <input type="checkbox"/> | 0x00400038 | 0x00024021 | addu \$8,\$0,\$2 | 50: | move \$t0, \$v0 |
| <input type="checkbox"/> | 0x0040003c | 0x24020005 | addiu \$2,\$0,5 | 51: | li \$v0, 5 # Leitura de N |
| <input type="checkbox"/> | 0x00400040 | 0x0000000c | syscall | 52: | syscall |
| <input type="checkbox"/> | 0x00400044 | 0x0040082a | slt \$1,\$2,\$0 | 53: | blt \$v0, \$zero, end |
| <input type="checkbox"/> | 0x00400048 | 0x14200020 | bne \$1,\$0,32 | | |
| <input type="checkbox"/> | 0x0040004c | 0x00024821 | addu \$9,\$0,\$2 | 54: | move \$t1, \$v0 |
| <input type="checkbox"/> | 0x00400050 | 0x23bffff4 | addi \$29,\$29,-12 | 56: | addi \$sp, \$sp, -12 # Abre 3 posições na pilha |
| <input type="checkbox"/> | 0x00400054 | 0xafa80000 | sw \$8,0(\$29) | 57: | sw \$t0, 0(\$sp) # Guarda M na pilha |
| <input type="checkbox"/> | 0x00400058 | 0xafa90004 | sw \$9,4(\$29) | 58: | sw \$t1, 4(\$sp) # Guarda N na pilha |
| <input type="checkbox"/> | 0x0040005c | 0x0c100035 | jal 0x004000d4 | 59: | jal ackermann # Chamada da func recursiva |
| <input type="checkbox"/> | 0x00400060 | 0x00025821 | addu \$11,\$0,\$2 | 60: | move \$t3, \$v0 # Guarda resultado de ackermann em \$t3 |
| <input type="checkbox"/> | 0x00400064 | 0x24020004 | addiu \$2,\$0,4 | 61: <11> | li \$v0, 4 # Comando de print |
| <input type="checkbox"/> | 0x00400068 | 0x3c011001 | lui \$1,4097 | <12> | la \$a0, str_2 # print(str_2) |

O valor de N é lido (no caso da execução, o valor inserido é 1).

Após leitura, são abertas três posições na pilha, onde os valores de M e N são guardados nas posições 0(\$sp) e 4(\$sp), respectivamente.

| | | | | | |
|--------------------------|------------|------------|--------------------|-----|---|
| <input type="checkbox"/> | 0x0040004c | 0x00024821 | addu \$9,\$0,\$2 | 54: | move \$t1, \$v0 |
| <input type="checkbox"/> | 0x00400050 | 0x23bffff4 | addi \$29,\$29,-12 | 56: | addi \$sp, \$sp, -12 # Abre 3 posições na pilha |
| <input type="checkbox"/> | 0x00400054 | 0xafa80000 | sw \$8,0(\$29) | 57: | sw \$t0, 0(\$sp) # Guarda M na pilha |
| <input type="checkbox"/> | 0x00400058 | 0xafa90004 | sw \$9,4(\$29) | 58: | sw \$t1, 4(\$sp) # Guarda N na pilha |
| <input type="checkbox"/> | 0x0040005c | 0x0c100035 | jal 0x004000d4 | 59: | jal ackermann # Chamada da func recursiva |

| Data Segment | | | | | | | |
|--------------|------------|------------|------------|------------|-------------|-------------|-------------|
| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) |
| 0x7ffffefe0 | 0 | 0 | 0 | 0 | 1 | 1 | 4194400 |

O programa então chama a função ackermann (recursiva)

A função guarda o endereço de retorno na pilha, na posição 8(\$sp)

| Data Segment | | | | | | | |
|--------------|------------|------------|------------|------------|-------------|-------------|-------------|
| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) |
| 0x7ffffefe0 | 0 | 0 | 0 | 0 | 1 | 1 | 4194400 |

Os valores lidos, que estavam guardados na pilha, são carregados nos registradores \$s0 e \$s1

| Registers | | | Coproc 1 | Coproc 0 |
|-----------|--------|-----------|----------|----------|
| Name | Number | Value | | |
| \$zero | 0 | 0 | | |
| \$at | 1 | 0 | | |
| \$v0 | 2 | 1 | | |
| \$v1 | 3 | 0 | | |
| \$a0 | 4 | 268501036 | | |
| \$a1 | 5 | 0 | | |
| \$a2 | 6 | 0 | | |
| \$a3 | 7 | 0 | | |
| \$t0 | 8 | 1 | | |
| \$t1 | 9 | 1 | | |
| \$t2 | 10 | 0 | | |
| \$t3 | 11 | 0 | | |
| \$t4 | 12 | 0 | | |
| \$t5 | 13 | 0 | | |
| \$t6 | 14 | 0 | | |
| \$t7 | 15 | 0 | | |
| \$s0 | 16 | 1 | | |
| \$s1 | 17 | 1 | | |

Há uma comparação para conferir se $M = 0$

Se M não for igual a 0, há uma comparação para conferir se $N = 0$

Já que os dois são maiores que 0, pula para o rótulo `m_n`

| | | | | | | |
|--------------------------|------------|------------|----------------|-----|--------------------------|------------------|
| <input type="checkbox"/> | 0x004000e0 | 0x12000002 | beq \$16,\$0,2 | 72: | beq \$s0, \$zero, m_zero | # M = 0 |
| <input type="checkbox"/> | 0x004000e4 | 0x12200003 | beq \$17,\$0,3 | 73: | beq \$s1, \$zero, n_zero | # M > 0 && N = 0 |
| <input type="checkbox"/> | 0x004000e8 | 0x08100044 | j 0x00400110 | 74: | j m_n | # M > 0 && N > 0 |

Diminui o valor de N por 1

| | | |
|------|----|---|
| \$s0 | 16 | 1 |
| \$s1 | 17 | 0 |

Abre três novos espaços na pilha

Guarda os valores de M e N na pilha

Chama a função `ackermann` novamente (Neste caso agora a função `A(1, 0)` é chamada

| | | | | | | |
|--------------------------|------------|------------|--------------------|----------|----------------------|-----------------------------------|
| <input type="checkbox"/> | 0x00400110 | 0x2231ffff | addi \$17,\$17,-1 | 89: m_n: | addi \$s1, \$s1, -1 | # N - 1 |
| <input type="checkbox"/> | 0x00400114 | 0x23bdfbf4 | addi \$29,\$29,-12 | 90: | addi \$sp, \$sp, -12 | # Abre 3 espaços na pilha |
| <input type="checkbox"/> | 0x00400118 | 0xafb00000 | sw \$16,0(\$29) | 91: | sw \$s0, 0(\$sp) | # M |
| <input type="checkbox"/> | 0x0040011c | 0xafb10004 | sw \$17,4(\$29) | 92: | sw \$s1, 4(\$sp) | # N |
| <input type="checkbox"/> | 0x00400120 | 0x0c100035 | jal 0x004000d4 | 93: | jal ackermann | # Chamada recursiva [A(m, n-1)] |

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+c) | Value (+10) | Value (+14) | Value (+18) | Value (+1c) |
|------------|------------|------------|------------|------------|-------------|-------------|-------------|-------------|
| 0x7ffffef0 | 0 | 1 | 0 | 0 | 1 | 1 | 4194400 | 0 |

Novamente há uma checagem para saber se M é igual a 0

M continua sendo 1, mas N agora é 0, portanto ocorre um salto para o rótulo `n_zero`

| | | | |
|-----------------|----------------|--------------------------|---------------------------------------|
| sw \$31,8(\$29) | 69: ackermann: | sw \$ra, 8(\$sp) | # Guarda o endereço de retorno do jal |
| lw \$16,0(\$29) | 70: | lw \$s0, 0(\$sp) | # Carrega M em \$s0 |
| lw \$17,4(\$29) | 71: | lw \$s1, 4(\$sp) | # Carrega N em \$s1 |
| beq \$16,\$0,2 | 72: | beq \$s0, \$zero, m_zero | # M = 0 |
| beq \$17,\$0,3 | 73: | beq \$s1, \$zero, n_zero | # M > 0 && N = 0 |

| | | |
|------|----|---|
| \$s1 | 17 | 0 |
|------|----|---|

Diminui o valor de M por 1

Incrementa o valor de N por 1

Aloca três novos espaços na pilha, guardando os valores de M e N nela

retorna para a função Ackermann

| | |
|--------------------|---|
| addi \$l6,\$l6,-1 | 80: n_zero: addi \$s0, \$s0, -1 # M - 1 |
| addi \$l7,\$0,1 | 81: add \$s1, \$zero, 1 # N = 1 |
| addi \$29,\$29,-12 | 82: addi \$sp, \$sp, -12 # Aloca 3 espaços na pilha |
| sw \$l6,0(\$29) | 83: sw \$s0, 0(\$sp) # M |
| sw \$l7,4(\$29) | 84: sw \$s1, 4(\$sp) # N |
| jal 0x004000d4 | 85: jal ackermann # Chamada recursiva |

Os valores de M e N são carregados, e desta vez M é igual a 0, portanto ocorre um salto para o rótulo m_zero

| | |
|-----------------|---|
| sw \$31,8(\$29) | 69: ackermann: sw \$ra, 8(\$sp) # Guarda o endereço de retorno do jal |
| lw \$l6,0(\$29) | 70: lw \$s0, 0(\$sp) # Carrega M em \$s0 |
| lw \$l7,4(\$29) | 71: lw \$s1, 4(\$sp) # Carrega N em \$s1 |
| beq \$l6,\$0,2 | 72: beq \$s0, \$zero, m_zero # M = 0 |

| | | |
|------|----|---|
| \$s0 | 16 | 0 |
| \$s1 | 17 | 1 |

Retorna o valor de n+1 (inserido no registrador \$v0)

| | |
|-----------------|--|
| addi \$2,\$l7,1 | 77: m_zero: addi \$v0, \$s1, 1 # retorno = n + 1 |
| j 0x00400140 | 78: j retorno |

Recebe de volta o endereço de retorno

Desaloca a pilha

Retorna para a posição após a chamada recursiva no n_zero

| | |
|-------------------|--|
| lw \$31,8(\$29) | 105: retorno: lw \$ra, 8(\$sp) # Resgata end. de retorno |
| addi \$29,\$29,12 | 106: addi \$sp, \$sp, 12 # Desaloca a pilha |
| jr \$31 | 107: jr \$ra # Retorna |

| | | |
|------------|--------------------|---|
| 0x2210ffff | addi \$l6,\$l6,-1 | 80: n_zero: addi \$s0, \$s0, -1 # M - 1 |
| 0x20110001 | addi \$l7,\$0,1 | 81: add \$s1, \$zero, 1 # N = 1 |
| 0x23bdfff4 | addi \$29,\$29,-12 | 82: addi \$sp, \$sp, -12 # Aloca 3 espaços na pilha |
| 0xafb00000 | sw \$l6,0(\$29) | 83: sw \$s0, 0(\$sp) # M |
| 0xafb10004 | sw \$l7,4(\$29) | 84: sw \$s1, 4(\$sp) # N |
| 0x0c100035 | jal 0x004000d4 | 85: jal ackermann # Chamada recursiva |
| 0x08100050 | j 0x00400140 | 86: j retorno |

Volta para a função retorno e retorna para a posição após a recursão quando m e n > 1

| | | |
|------------|-------------------|--|
| 0x8fbf0008 | lw \$31,8(\$29) | 105: retorno: lw \$ra, 8(\$sp) # Resgata end. de retorno |
| 0x23bd000c | addi \$29,\$29,12 | 106: addi \$sp, \$sp, 12 # Desaloca a pilha |
| 0x03e00008 | jr \$31 | 107: jr \$ra # Retorna |

| | |
|--------------------|---|
| addi \$17,\$17,-1 | 89: m_n: addi \$s1, \$s1, -1 # N - 1 |
| addi \$29,\$29,-12 | 90: addi \$sp, \$sp, -12 # Abre 3 espaços na pilha |
| sw \$16,0(\$29) | 91: sw \$s0, 0(\$sp) # M |
| sw \$17,4(\$29) | 92: sw \$s1, 4(\$sp) # N |
| jal 0x004000d4 | 93: jal ackermann # Chamada recursiva [A(m, n-1)] |
| lw \$16,0(\$29) | 94: lw \$s0, 0(\$sp) # Resgata o valor original de M |
| addi \$16,\$16,-1 | 95: addi \$s0, \$s0, -1 # M - 1 |
| addi \$29,\$29,-12 | 96: addi \$sp, \$sp, -12 # Abre 3 espaços na pilha |
| sw \$16,0(\$29) | 97: sw \$s0, 0(\$sp) # M |
| sw \$2,4(\$29) | 98: sw \$v0, 4(\$sp) # N (resultado da chamada recursiva) |
| jal 0x004000d4 | 99: jal ackermann # Chamada recursiva [A(m-1, A(m, n-1)] |
| j 0x00400140 | 100: j retorno |

Decrementa o valor de m e chama a função recursivamente com os valores adquiridos (A(m-1, A(m,n-1))

| | |
|----------------------------|---|
| 0xafbf0008 sw \$31,8(\$29) | 69: ackermann: sw \$ra, 8(\$sp) # Guarda o endereço de retorno do jal |
| 0x8fb00000 lw \$16,0(\$29) | 70: lw \$s0, 0(\$sp) # Carrega M em \$s0 |
| 0x8fb10004 lw \$17,4(\$29) | 71: lw \$s1, 4(\$sp) # Carrega N em \$s1 |
| 0x12000002 beq \$16,\$0,2 | 72: beq \$s0, \$zero, m_zero # M = 0 |
| 0x12200003 beq \$17,\$0,3 | 73: beq \$s1, \$zero, n_zero # M > 0 && N = 0 |
| 0x08100044 j 0x00400110 | 74: j m_n # M > 0 && N > 0 |

| | | |
|------|----|---|
| \$s0 | 16 | 0 |
| \$s1 | 17 | 2 |

Já que o valor de m é zero, ocorre um salto para o rótulo m_zero

O valor de n+1 é novamente retornado

| | |
|----------------------------|--|
| 0x22220001 addi \$2,\$17,1 | 77: m_zero: addi \$v0, \$s1, 1 # retorno = n + 1 |
| 0x08100050 j 0x00400140 | 78: j retorno |

| | | |
|------|---|---|
| \$v0 | 2 | 3 |
|------|---|---|

A função retorna para o início e guarda o resultado em \$t3

| | |
|---|---|
| 0x0040005c 0x0c100035 jal 0x004000d4 | 59: jal ackermann # Chamada da func recursiva |
| 0x00400060 0x00025821 addu \$11,\$0,\$2 | 60: move \$t3, \$v0 # Guarda resultado de ackermann em \$t3 |

O macro print é chamado, que imprime os resultados

| | |
|---------------------------------------|--|
| 0x00400064 0x24020004 addiu \$2,\$0,4 | 61: <11> li \$v0, 4 # Comando de print |
| 0x00400068 0x3c011001 lui \$1,4097 | <12> la \$a0, str_2 # print(str_2) |

O programa é reiniciado

| | |
|-------------------------|------------------------------------|
| 0x0000000c syscall | <31> syscall |
| 0x08100007 j 0x0040001c | 62: j inicio # reinicia o programa |

Mars Messages Run I/O

Programa Ackermann
Thiago Zilberknop, Leonardo Chou da Rosa
Digite um valor positivo para executar o programa ou um valor negativo para encerrar.
1
1
A(1, 1) = 3
Digite um valor positivo para executar o programa ou um valor negativo para encerrar.

Clear