CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS –
CEFET/MG

Diogo Emanuel Antunes Santos (20213002091)
Mateus Ribeiro Ferraz (20213001530)

AULA 4: IMPLEMENTAÇÃO DE FUNÇÕES EM ASSEMBLY DO MIPS

Laboratório de Arquitetura e Organização de Computadores

BELO HORIZONTE
2022

1) **Explique o quê o código da função “f” faz:**

Edit Execute

pratica4.asm pratica_1*

```
15  .globl main
16
17  main:
18          addi $a0, $0, 32 # result = f(32)
19          jal funcao    # pega retorno da funcao
20          j encerraPrograma
21
22  funcao:
23          subi $sp, $sp, 136   #liberando espaço na memoria pro stack pointer
24          addi $t0, $0, -1     # Atrbuindo -1 a $t0
25          sw $t0, 0($sp)       # Atribuindo $t0 a duas poisções da pilha
26          sw $t0, 4($sp)
27          addi $s0, $0, 2      #Valor inicial de i
28          addi $s1, $0, 8      #end base
29          addi $t6, $a0, 1     #Somando 1 a $t6 para usar na condicional do loop
30
31          loop:
32                  add $t5, $sp, $s1    # t5 é registrador de a[i]
33                  lw $t3, -4 ($t5)     # a [i-1]
34                  lw $t4, -8 ($t5)     # a [i-2]
35                  add $s2, $t4, $t3    #a[i] = a[i-1] + a[i-2]
36                  sw $s2, 0 ($t5)      # Salvo o valor da soma na posição a[i]
37                  addi $s1, $s1, 4     #incrementa o índice do end base
38                  addi $s0, $s0, 1     # i++
39                  slt $t1, $s0, $t6    # Teste condicional do loop
40                  beq $t1, $zero, fim  # Se i>y, então sai do loop
41                  j loop               # Continuação do loop
42
43          fim:
44
45                  addi $a0, $sp, 128          # salvo o ultimo a[i] em a0
46                  lw $v1, 0($a0)              #salvo o ultimo a[i] no retorno
47                  addi $sp, $sp, 136          #recomponhe a memoria
48                  jr $ra                      #Volta pra main com o valor de a[i]
49
50  encerraPrograma:
51          li $v0,10 #comando para finalizar o programa
52          syscall
53
```

A função “f” começa liberando espaço na pilha para a execução do seu código. Com isso, o registrador “$t0” recebe o valor de -1, que é usado na atribuição das duas primeiras posições do vetor “a”. Em seguida, o registrador “$t6” recebe o valor de $a0(y) + 1, sendo usado na condicional do loop, e o registrador “s0” recebe o valor inicial de “i” (2) e $s1 recebe o valor para base de do vetor "a". Com essas atribuições feitas, o código entra no loop. No loop, "$t5" recebe o valor de a[i] é carregado da memória as posições de a[i-1] e de a[i-2]. Com isso, a soma dos valores dessas duas posições do vetor é atribuída a “$a0”. Para verificar se o código deve sair ou não do loop, há o teste condicional, verificando se $s1(i) é menor $t6(y+1). Caso seja menor que $t6, o loop continua, encerrando-o quando for igual a $t6. Com isso, é salvo o último valor de a[i] em $a0, que é carregado em “$v1”, e a memória

usada da pilha é recomposta e a leitura do código é retornada a “main”, depois é encaminhada para uma função que encerra o programa.

2) **O que acontece se o valor de y for 32? Qual valor é retornado nesse caso? Aponte os valores intermediários assumidos por “a[i]” e por “i” durante uma chamada com esse valor (f(32)).**

a[i]	i	a[i]	i
-2	2	-2584	17
-3	3	-4181	18
-5	4	-6765	19
-8	5	-10946	20
-13	6	-17711	21
-21	7	-28657	22
-34	8	-46368	23
-55	9	-75025	24
-89	10	-121393	25
-144	11	-196418	26
-233	12	-317811	27
-377	13	-514229	28
-610	14	-832040	29
-987	15	-1346269	30
-1597	16	-2178309	31

Valor final: a[i] = -3524578 e i = 32

Observa-se que se o y = 32, o valor final de a[i] será muito pequeno, já que o valor de a[i] reduz a cada iteração de “i” a partir do seus dois valores antecessores. Logo, se o valor de y for 32, o valor final de a[i] é -3524578.

3) **O que poderia acontecer se o prólogo da função “f” guardasse o seu endereço de retorno na pilha (para poder efetuar uma chamada a outro procedimento, por exemplo), no primeiro espaço disponível no “frame” da função?**

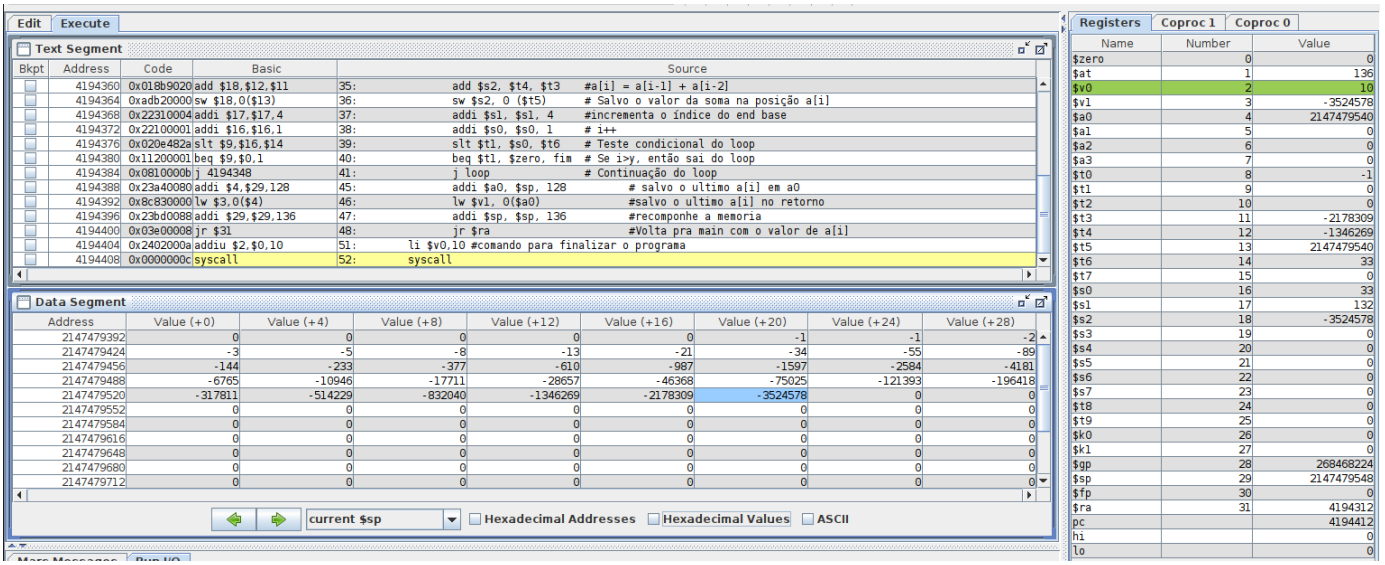
Neste caso o que poderia acontecer é que ao prólogo da função “f” guardar o endereço de retorno no frame da função não há como prever se quando retornasse para a chamadora o procedimento chamado teria um endereço da posição inicial correto, é provável que $ra vá sobrescrever endereço base. Então o procedimento não retorna para a posição inicial.

Apêndice:

```
.globl main

main:
	addi $a0, $0, 32 # result = f(32)
	jal funcao	# pega retorno da funcao
	j encerraPrograma

funcao:
	subi $sp, $sp, 136	#liberando espaço na memoria pro stack pointer
	addi $t0, $0, -1	# Atrbuindo -1 a $t0
	sw $t0, 0($sp) # Atribuindo $t0 a duas poisções da pilha
	sw $t0, 4($sp)
	addi $s0, $0, 2 #Valor inicial de i
	addi $s1, $0, 8	#end base
	addi $t6, $a0, 1	#Somando 1 a $t6 para usar na condicional do loop

	loop:
		add $t5, $sp, $s1	# t5 é registrador de a[i]
		lw $t3, -4 ($t5)	# a [i-1]
		lw $t4, -8 ($t5)	# a [i-2]
		add $s2, $t4, $t3	#a[i] = a[i-1] + a[i-2]
		sw $s2, 0 ($t5)	# Salvo o valor da soma na posição a[i]
		addi $s1, $s1, 4	#incrementa o índice do end base
		addi $s0, $s0, 1	# i++
		slt $t1, $s0, $t6	# Teste condicional do loop
		beq $t1, $zero, fim	# Se i>y, então sai do loop
		j loop	# Continuação do loop

	fim:

		addi $a0, $sp, 128		# salvo o ultimo a[i] em a0
		lw $v1, 0($a0)	#salvo o ultimo a[i] no retorno
		addi $sp, $sp, 136		#recomponhe a memoria
		jr $ra	#Volta pra main com o valor de a[i]

encerraPrograma:
```

```
li $v0,10 #comando para finalizar o programa
syscall
```

Teste de execução:



Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source
	4194360	0x018b9020	add $18,$12,$11	35:	add $s2, $t4, $t3 #a[i] = a[i-1] + a[i-2]
	4194364	0xadb20000	sw $18,0($13)	36:	sw $s2, 0 ($t5) # Salvo o valor da soma na posição a[i]
	4194368	0x22310004	addi $17,$17,4	37:	addi $s1, $s1, 4 #incrementa o índice do end base
	4194372	0x22100001	addi $16,$16,1	38:	addi $s0, $s0, 1 # i++
	4194376	0x020e482a	slt $9,$16,$14	39:	slt $t1, $s0, $t6 # Teste condicional do loop
	4194380	0x11200001	beq $9,$0,1	40:	beq $t1, $zero, fim # Se i>y, então sai do loop
	4194384	0x0810000b	j 4194348	41:	j loop # Continuação do loop
	4194388	0x23a40080	addi $4,$29,128	45:	addi $a0, $sp, 128 #salvo o ultimo a[i] em a0
	4194392	0x8c830000	lw $3,0($4)	46:	lw $v1, 0($a0) #salvo o ultimo a[i] no retorno
	4194396	0x23bd0088	addi $29,$29,136	47:	addi $sp, $sp, 136 #recomponhe a memoria
	4194400	0x03e00008	jr $31	48:	jr $ra #Volta pra main com o valor de a[i]
	4194404	0x2402000a	addiu $2,$0,10	51:	li $v0,10 #comando para finalizar o programa
	4194408	0x0000000c	syscall	52:	syscall

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
2147479392	0	0	0	0	0	-1	-1	-2
2147479424	-3	-5	-8	-13	-21	-34	-55	-89
2147479456	-144	-233	-377	-610	-987	-1597	-2584	-4181
2147479488	-6765	-10946	-17711	-28657	-46368	-75025	-121393	-196418
2147479520	-317811	-514229	-832040	-1346269	-2178309	-3524578	0	0
2147479552	0	0	0	0	0	0	0	0
2147479584	0	0	0	0	0	0	0	0
2147479616	0	0	0	0	0	0	0	0
2147479648	0	0	0	0	0	0	0	0
2147479680	0	0	0	0	0	0	0	0
2147479712	0	0	0	0	0	0	0	0

current $sp | Hexadecimal Addresses | Hexadecimal Values | ASCII

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	136
$v0	2	10
$v1	3	-3524578
$a0	4	2147479540
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	-1
$t1	9	0
$t2	10	0
$t3	11	-2178309
$t4	12	-1346269
$t5	13	2147479540
$t6	14	33
$t7	15	0
$s0	16	33
$s1	17	132
$s2	18	-3524578
$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	4194312
pc		4194412
hi		0
lo		0