

(1) Qual o número do registrador \$sp no conjunto de registradores do MIPS e qual o seu valor inicial (atribuído pelo simulador MARS)? O número do registrador é o 29 e o valor atribuído é o endereço na memória de dados 0x7ffffeffc

(2) Qual é o primeiro valor escrito na pilha, e qual o significado do mesmo? O primeiro valor escrito na pilha é o endereço contido no registrador \$ra durante a chamada da função.

(3) Mostre o conteúdo da pilha ao entrar na terceira chamada aninhada de alguma recursão (use a opção File→Dump Memory do simulador MARS).

Text Segment									
Bkpt	Address	Code	Basic						Source
	0x00400100	0x00000000	jr \$ra	154:	jr	\$ra			
	0x00400100	0x00000000	lw \$a0, 0(\$sp)	160:	lw	\$a0, 0(\$sp)			# Carrega o endereço do RA
	0x00400104	0x12000000	beq \$16, \$0, 0x0000000c (\$29)	161:	beq	\$a0, \$zero, cap_ra_ins			# Caso RA não tenha sido capturado salta para capturar
	0x00400108	0x00000000	lw \$17, 0x00000004 (\$29)	165:	lw	\$a1, 4(\$sp)			# Carrega o endereço do apontador da lista
	0x0040010c	0x00000000	lw \$18, 0x00000000 (\$17)	166:	lw	\$a2, 0(\$a1)			# Carrega o dado na posição
	0x00400110	0x12400000	beq \$18, \$0, 0x00000003 (\$29)	167:	beq	\$a2, \$zero, ins_el			# Verifica se existe dado na posição : salta para inserir
	0x00400114	0x26310004	addiu \$17, \$17, 0x00000004 (\$29)	168:	addiu	\$a1, \$a1, 4			# Incrementa o apontador para o próximo item
	0x00400118	0x00000000	sw \$17, 0x00000004 (\$29)	169:	sw	\$a1, 4(\$sp)			# Atualiza o endereço do próximo elemento na pilha
	0x0040011c	0x00000000	jal 0x00400100	171:	jal	ins_ult_el			# Chama a função novamente
	0x00400120	0x00000000	lw \$20, 0x00000008 (\$29)	175:	lw	\$a4, 8(\$sp)			# Carrega o dado a ser inserido
	0x00400124	0x00000000	sw \$20, 0x00000000 (\$17)	176:	sw	\$a4, 0(\$a1)			# Adiciona o dado na lista
	0x00400128	0x26310004	addiu \$17, \$17, 0x00000004 (\$29)	177:	addiu	\$a1, \$a1, 4			# Endereço para o próximo elemento
	0x0040012c	0x26310004	addiu \$18, \$17, 0x00000004 (\$29)	178:	addiu	\$a2, \$a1, 4			# Gera o endereço do próximo elemento
	0x00400130	0x00000000	sw \$18, 0x00000000 (\$17)	179:	sw	\$a2, 0(\$a1)			# Salva o endereço para próximo elemento
	0x00400134	0x00000000	end ins	180:					# Salta finalizar a função
	0x00400138	0x00000000	sw \$2a, 0(\$sp)	184:	sw	\$2a, 0(\$sp)			# Salva o RA para retornar
	0x0040013c	0x00000000	main_ins	185:					# Retorna para a função

Data Segment									
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)	
0x7ffffeffc	0x00000000	0x00000000	0x00000000	0x00000000	0x00400090	0x10040008	0x00000001e	0x000000000	
0x7ffffeff0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffeff4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffeff8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffeffc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefa0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefa4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefa8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefac	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefb0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefb4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefb8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefbc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefc4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefc8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefcc	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefd0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefd4	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefd8	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	
0x7ffffefd0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	

(4) Qual o conteúdo do registrador \$sp neste momento? Para entender o conteúdo do \$sp (Stack Pointer) é necessário mencionar que para execução das funções é armazenado o endereço de retorno, o endereço do primeiro elemento e o elemento a ser inserido ou o somatório dos elementos. A chamada capturada na imagem acima o registrador \$sp tem armazenado neste momento o endereço de retorno da função, o endereço para o próximo elemento da lista e o elemento a ser inserido ao final da lista.

(5) Isto implica quanto espaço alocado na pilha? Para armazenar esses dados foi alocado 12 bytes na pilha equivalente a 3 .word.

(6) Observar o retorno do procedimento recursivo. O valor do registrador \$sp volta ao valor original? (Se isto não ocorrer seu programa está incorreto, pois sua execução deixa lixo na pilha). O retorno da função recursiva somente desaloca (move) o registrador \$sp ao endereço inicial. Porém os dados utilizados na chamada continuam “salvos” nas posições alocadas

(7) Em qual linha de código este valor é re-estabelecido? O registrador \$sp (Stack Pointer) é retornado ao endereço original ao finalizar a função e recuperar o retorno (caso haja) na pilha. Assim na chamada descrita acima a linha em que este valor é re-estabelecido é a 54.

addiu \$sp, \$sp, 12 # Desaloca espaço na pilha