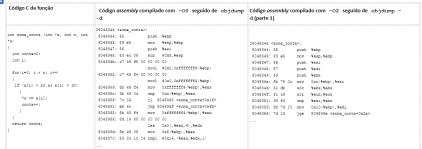
Pergunta 1



Considere as figuras com a listagem de uma função em C e dois fragmentos da compilação desse código para assembly do IA-32 sem e com otimizações



Nas duas figuras seguintes estão representadas algumas posições de memória relativas ao quadro de ativação da função (stack frame), em que cada retângulo representa 4 bytes. Preencha cada caixa descrevendo os conteúdos do quadro de ativação (para a versão -00 e para a versão -02).

Em ambos os casos considere que <u>o estado da sfack corresponde à execução do fragmento apresentado em assembly.</u>
Para facilitar a correção automática **use as seguintes designações** para descrever os conteúdos do quadro de ativação: nomes das variáveis do programa, antigo_ebp, ender_regresso, nome de registos (e.g.: %eax). Nos campos não preenchidos coloque o símbolo menos ("-").

Quadro de ativação para a versão -00			b) Quadro de ativação para a versão -O2		
%ebp-12 =>	xx xx xx xx	[sla]	%ebp-12 =>	xx xx xx xx	[s2a]
%ebp-8 =>	xx xx xx xx	[s1b]	%ebp-8 =>	xx xx xx xx	[s2b]
%ebp-4 =>	xx xx xx xx	[slc]	%ebp-4 =>	xx xx xx xx	[s2c]
%ebp =>	xx xx xx xx	[s1d]	%ebp =>	xx xx xx xx	[s2d]
%ebp+4 =>	xx xx xx xx	[sle]	%ebp+4 =>	xx xx xx xx	[s2e]
%ebp+8 =>	xx xx xx xx	[slf]	%ebp+8 =>	xx xx xx xx	[s2f]
%ebp+12 =>	xx xx xx xx	[slg]	%ebp+12 =>	xx xx xx xx	[s2g]
%ebp+16 =>	xx xx xx xx	[s1h]	%ebp+16 =>	xx xx xx xx	[s2h]

c) Com base na análise dos códigos em assembly e nos quadros de ativação (das 2 versões), indique os endereços das instruções que inicializam a zero as variáveis locais:

- variável conta : em -00 [dc0], em -02 [dc1]; - variável i : em -00 [dc2], em -02 [dc3];

49,41176 em 60 pontos Considere as figuras com a listagem da mesma função em C e um fragmento da compilação desse código para assembly do IA-32 (com-o2), imediatamente a seguir ao da questão anterior

Código C da função Assembly compilado com -02 seguido de objdump -d (parte 2)

```
| COURT | COUR
```

a) Preencha a tabela em baixo, mageando as variáveis locais e argumentos (do código C da função) aos registos. Indique também o endereço (no código apresentado acima) da primeira instrução no assembly que escreva no registo

Endereço da instrução Variável [e1] a[i\] conta

```
s = i + conta + a[i\] * 4;.
```

Codifique esta expressão numa única instrução assembly:

[b1] [b2] (%ebx, [b3]), [b4]

*s += a[i\];

c) Pretende-se modificar a estrutura do ciclo for para um do..while.

Preencha os espaços em branco do código assembly equivalente ao ciclo for do código C apresentado.

```
Código C
                        Código assembly
                              while: ...
     do{
                                [c1] %ecx
                                cmpl [c2], %esi
         i++;
                               [c3] out
     }while(i < n);</pre>
                               [c4] while
                            out: ...
```

Pergunta 3 10 em 50 pontos



Considere o estado do programa no ponto de paragem (breakpoint) indicado na figura e um fragmento da compilação do código C para assembly.

Código da função em C	Assembly compilado com -o2 seguido de	bbjdump -d Breakpoint em soma_conta (compilado com -02)
<pre>int soma_conta (int *a,</pre>	08048390 <scma_conta>: 08048396: 88 75 0c mov</scma_conta>	(gdb) info registers

a) Indique os valores armazenados nas posições da pilha indicadas por %ebp+12 [r2] e %ebp+16 [r3];

b) Indique o valor do frame pointer da função que chamou esta [r4] e o número de bytes que a função chamadora tem reservados para variáveis locais e para salvaguarda de registos (excluindo %ebp)? [r5]: c) Indique qual o endereço da última instrução executada? [r6]; (nota: existem 2 caminhos...)

Pergunta 4

10 em 30 pontos

Responda às duas alineas seguintes no espaço disponível no final das questões.

Considere as figuras com a listagem da mesma função em C e um fragmento da compilação desse código para assembly do IA-32 com algumas otimizações:

Código C da função

Assembly compilado com -o2 seguido de objdump -d (parte
1)

```
1)

"a)

(int some_conta (int *a, int n, int 08048390 <some_conta>:

"a)

(int conta=0;

int int int int 08048391 55 push telp

(int conta=0;

int int int 08048391;

8048391; 56 push telp

8048391; 56 push telp

8048391; 56 push telp

8048391; 56 push telp

8048391; 50 push telp

(int int int int 08048391; 50 push telp

8048391; 50 push telp

(int int int int 08048391; 50 push telp

(int 
             } return conta;
```

a) Introduza comentários/anotações para cada uma das instruções no código assembly que foram destacadas a **negrito** na figura.

b) Considere agora que (i) este código foi compilado para uma versão do MIPS (também com 32-bits e que (ii) o instruction set deste MIPS não tem instruções de mov (só tem load e ptore para acessos à memória) e tem o mesmo suporte a estruturas de controlo que o 14-32.

Resectoreva este código assembly para esta versão do MIPS (pode usar a sintaxe do GNU para o 14-32, substituindo apenas o nome dos registos para \$\frac{\pmathbm{\text{s}}}{\pmathbm{\text{s}}}, \text{\text{\text{unit}}} \text{\text{sinta}}, \text{\text{\text{justificando}}} todas as alterações que introduzir ao código.