Pergunta 1

9 em 60 pontos

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:

Código C da função

```
int soma_conta (int *a, int n, int *s)
{
  int conta=0;
  int i;

  for(i=0; i < n; i++)
  {
   if (a[i] > 20 && a[i] < 30)
     {
      *s += a[i];
      conta++;
     }
  }
  return conta;
}
```

Código *assembly* compilado com `-O0` seguido de `objdump -d`

```
08048344 <soma_conta>:
 8048344: 55           push   %ebp
 8048345: 89 e5        mov    %esp,%ebp
 8048347: 56           push   %esi
 8048348: 83 ec 08     sub    $0x8,%esp
 804834b: c7 45 f8 00 00 00 00
                       movl   $0x0,0xfffffff8(%ebp)
 8048352: c7 45 f4 00 00 00 00
                       movl   $0x0,0xfffffff4(%ebp)
 8048359: 8b 45 f4     mov    0xfffffff4(%ebp),%eax
 804835c: 3b 45 0c     cmp    0xc(%ebp),%eax
 804835f: 7c 02        jl   8048363 <soma_conta+0x1f>
 8048361: eb 4c        jmp 80483af <soma_conta+0x6b>
 8048363: 8b 45 f4     mov    0xfffffff4(%ebp),%eax
 8048366: 8d 14 85 00 00 00 00
                       lea    0x0(,%eax,4),%edx
 804836d: 8b 45 08     mov    0x8(%ebp),%eax
 8048370: 83 3c 10 14 cmpl  $0x14,(%eax,%edx,1)
...
```

Código *assembly* compilado com `-O2` seguido de `objdump -d` (parte 1)

```
08048344 <soma_conta>:
 8048344: 55           push   %ebp
 8048345: 89 e5        mov    %esp,%ebp
 8048347: 56           push   %esi
 8048348: 57           push   %edi
 8048349: 53           push   %ebx
 804834a: 8b 75 0c     mov    0xc(%ebp),%esi
 804834d: 31 db        xor    %ebx,%ebx
 804834f: 31 c9        xor    %ecx,%ecx
 8048351: 39 f3        cmp    %esi,%ebx
 8048353: 8b 7d 10     mov    0x10(%ebp),%edi
 8048356: 7d 16        jge    804836e <soma_conta+0x2a>
...
```

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 `-O0` e para a versão `-O2`).
Em ambos os casos considere que o estado da *stack* corresponde à execução do fragmento apresentado em *assembly*.
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 ("-").

a) Quadro de ativação para a versão -O0

%ebp-12 =>	xx xx xx xx	**[s1a]**
%ebp-8 =>	xx xx xx xx	**[s1b]**
%ebp-4 =>	xx xx xx xx	**[s1c]**
%ebp =>	xx xx xx xx	**[s1d]**
%ebp+4 =>	xx xx xx xx	**[s1e]**
%ebp+8 =>	xx xx xx xx	**[s1f]**
%ebp+12 =>	xx xx xx xx	**[s1g]**
%ebp+16 =>	xx xx xx xx	**[s1h]**

b) Quadro de ativação para a versão -O2

%ebp-12 =>	xx xx xx xx	**[s2a]**
%ebp-8 =>	xx xx xx xx	**[s2b]**
%ebp-4 =>	xx xx xx xx	**[s2c]**
%ebp =>	xx xx xx xx	**[s2d]**
%ebp+4 =>	xx xx xx xx	**[s2e]**
%ebp+8 =>	xx xx xx xx	**[s2f]**
%ebp+12 =>	xx xx xx xx	**[s2g]**
%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 -O0 **[dc0]**, em -O2 **[dc1]**;
- variável `i` : em -O0 **[dc2]**, em -O2 **[dc3]**;

Pergunta 2

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

```
int soma_conta (int *a, int n, int *s)
{
  int conta=0;
  int i;

  for(i=0; i < n; i++)
  {
   if (a[i\]>20 && a[i\]<30)
     {
      *s += a[i\];
      conta++;
     }
  }
  return conta;
}
```

Assembly compilado com `-O2` seguido de `objdump -d` (parte 2)

```
...
 80483a4:  8b 45 08  mov   0x8(%ebp),%eax
 80483a7:  8b 14 88  mov   (%eax,%ecx,4),%edx
 80483aa:  8d 42 eb  lea   0xffffffeb(%edx),%eax
 80483ad:  83 f8 08  cmp   $0x8,%eax
 80483b0:  77 03     jan   80483b5 <soma_conta+0x25>
 80483b2:  01 17     add   %edx,(%edi)
 80483b4:  43        inc   %ebx
 80483b5:  41        inc   %ecx
 80483b6:  39 f1     cmp   %esi,%ecx
 80483b8:  7c ea     jl    80483a4 <soma_conta+0x14>
...
```

a) **Preencha** a tabela em baixo, mapeando 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 correspondente a cada variável.

Variável	Registo	Endereço da instrução
a	**[r1]**	**[e1]**
a[i]	**[r2]**	**[e2]**
conta	**[r3]**	**[e3]**
i	**[r4]**	**[e4]**
n	%esi	-
s	**[r5]**	-

b) Pretende-se modificar a expressão no corpo do ciclo `for` no código fonte

```
*s += a[i\];
```

para

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

Codifique esta expressão numa única instrução *assembly*:

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

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

```
do{
  ...
      i++;
}while(i < n);
```

Código *assembly*

```
          while:  ...
            [c1] %ecx
            cmpl [c2], %esi
            [c3] out
            [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 `objdump -d`	*Breakpoint* em `soma_conta` (compilado com `-O2`)

```
int soma_conta (int *a,
        int n, int *s)
{
  int conta=0;
  int i;

  for(i=0; i < n; i++)
  {
   if (a[i]>20 &&
      a[i]<30)
     {
      *s += a[i];
      conta++;
     }
  }

  return conta;
}
```

```
08048390 <soma_conta>:
...
 8048396:  8b 75 0c  mov   0xc(%ebp),%esi
 8048399:  31 db     xor   %ebx,%ebx
 804839b:  31 c9     xor   %ecx,%ecx
 804839d:  39 f3     cmp   %esi,%ebx
 804839f:  8b 7d 10  mov   0x10(%ebp),%edi
 80483a2:  7d 16     jge   80483ba
 80483a4:  8b 45 08  mov   0x8(%ebp),%eax
 80483a7:  8b 14 88  mov   (%eax,%ecx,4),%edx
 80483aa:  8d 42 eb  lea   0xffffffeb(%edx),%eax
 80483ad:  83 f8 08  cmp   $0x8,%eax
 80483b0:  77 03     ja    80483b5
 80483b2:  01 17     add   %edx,(%edi)
 80483b4:  43        inc   %ebx
 80483b5:  41        inc   %ecx
 80483b6:  39 f1     cmp   %esi,%ecx
 80483b8:  7c ea     jl    80483a4
...
```

```
(gdb) info registers

eax    0xbfffd720
ecx    0x1
edx    0xc
ebx    0x1
esp    0xbfffd700
ebp    0xbfffd70a
esi    0x4
edi    0xbfffd730
eip    0x80483b5

(gdb) x/8xw $esp
0xbfffd700:
 0x0016d51a  0x00241f60 0x00000005  0xbfffd738
0xbfffd710:
 0x0804841e  ?????????? ??????????  ??????????
```

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 alíneas 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)

```
int soma_conta (int *a, int n, int
*s)
{
  int conta=0;
  int i;

  for(i=0; i < n; i++)
  {
   if (a[i]>20 && a[i]<30)
     {
      *s += a[i];
      conta++;
     }
  }
  return conta;
}
```

```
08048390 <soma_conta>:
 8048390: 55        push  %ebp
 8048391: 89 e5     mov   %esp,%ebp
 8048393: 57        push  %edi
 8048394: 56        push  %esi
 8048395: 53        push  %ebx
 8048396: 8b 75 0c  mov   0xc(%ebp),%esi
 8048399: 31 db     xor   %ebx,%ebx
 804839b: 31 c9     xor   %ecx,%ecx
 804839d: 39 f3     cmp   %esi,%ebx
 804839f: 8b 7d 10  mov   0x10(%ebp),%edi
 80483a2: 7d 16     jge   80483ba <soma_conta+0x2a>
...
```

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 `store` para acessos à memória) **e** tem o mesmo suporte a estruturas de controlo que o IA-32.
Reescreva este código *assembly* para esta versão do MIPS (pode usar a sintaxe do GNU para o IA-32, substituindo apenas o nome dos registos para `$r0`, `$r1`, ...), **justificando** todas as alterações que introduzir ao código.