LISTA 3 - Período 2021-1 Remoto
90 pontos

Questão 1) (20 pontos) Faça a engenharia reversa baseando-se no código de montagem e preencha as reticências colocadas na parte apagada do código C. Você não pode acrescentar linha ou deixar de preencher qualquer linha. Os caracteres não apagados devem estar presentes na sua solução.

```
1    void rot(char lista[ ], int x)
2    {
3        int i, j;
4        ...
5            ...
6                … = ... ;
7        ... = … ;
8    }
```

Compilando com "gcc -m32 -O1 -fno-PIC -S", descartadas algumas linhas de diretivas, temos:

```
rot:
1    endbr32                    ……..
2    pushl  %edi                ……..
3    pushl  %esi                ……..
4    pushl  %ebx                ……..
5    movl   16(%esp), %esi      ……..
6    movl   20(%esp), %edi      ……..
7    movzbl (%esi), %eax        ……..
8    testb  %al, %al            ……..
9    je     .L5                 ……..
10   leal   1(%esi), %edx       ……..
11   movl $0, %ecx              ……..
12   jmp    .L4                 ……..
.L8:
13   movb %al, (%esi,%ecx)      ……..
14   leal   1(%ecx), %ecx       ……..
.L3:
15   addl   $1, %edx            ……..
16   movzbl -1(%edx), %eax      ……..
17   testb  %al, %al            ……..
18   je     .L2                 ……..
.L4:
19   movsbl %al, %ebx           ……..
20   cmpl   %edi, %ebx          ……..
21   jne    .L8                 ……..
22   jmp    .L3                 ……..
.L5:
23   movl   $0, %ecx            ……..
.L2:
24   movb $0, (%esi,%ecx)       ……..
25   popl   %ebx                ……..
26   popl   %esi                ……..
27   popl   %edi                ……..
26   ret                        ……..
```

a) (10) Comente cada linha do código de montagem, procurando não dizer o que a instrução faz mas associar a instrução ao código C que foi apagado. Ao resolver este item, deixe para comentar após ter reconstruído o programa C pois fará mais sentido mostrar o significado das instruções em termos das variáveis do código C, o que é a engenharia reversa pretendida.

b) (10) Complete o código C e descreva o que a rotina faz.

Questão 2) (20 pontos) A rotina rot (int x, int n) possui um comando switch e retorna um valor inteiro. Descubra pelo código de montagem qual o conteúdo do comando switch e do código apagado.

```
int rot(int x, int n) {
    switch(n) {
        …
    }
    return … ;}
```

O código de montagem gerado pelo GCC com otimização -O1 é:

```
rot:
1       endbr32
2       movl  8(%esp), %eax
3       addl  $3, %eax
4       cmpl  $8, %eax
5       ja      .L2
6       notrack jmp *.L4(,%eax,4)
.L4:
7       .long  .L9
8       .long  .L7
9       .long  .L2
10      .long  .L2
11      .long  .L6
12      .long  .L2
13      .long  .L2
14      .long  .L5
15      .long  .L3
16      .text
.L2:
17      movl  $-1, %eax
.L8:
18      imull  %eax, %eax
19      ret
.L7:
20      movl  $1, 4(%esp)
.L6:
21      movl  4(%esp), %eax
22      addl  $1, %eax
23      jmp   .L8
.L5:
24      addl  $2, 4(%esp)
.L3:
25      movl  4(%esp), %eax
26      addl  %eax, %eax
27      jmp   .L8
.L9:
28      movl  $0, %eax
29      jmp   .L8
```

a) (4) As linhas 3 e 4 são fundamentais para descobrir os valores extremos envolvidos nos cases do switch. Indique o que cada linha está fazendo, uma por uma, e justifique o que ela está testando e mostrando como são reconhecidos os valores extremos do switch.

b) (4) Explique o que a linha 5 faz e mostre como ela completa os valores possíveis no comando switch.

c) (4) Explique o notrack na linha 6, o que a instrução faz e explique a razão do conteúdo de memória nas linhas 7 a 16. Associe cada uma das linhas 7 a 16 aos valores dos cases respectivos.

d) (8) Escreva abaixo o código C completo da rotina.

Questão 3) (25 pontos) Seja o código C abaixo, onde a dimensão das matrizes foi apagada.

```
#define N ...
typedef int matrix[N][N];
int prod (matrix A, matrix B, int i , int k){
   int j;
   int result = 0;
   for (j=0;j<N;j++)
      result += A[i][j] * B[j][k];
   return result;
}
```

a) (8) O código compilado acima com -m32 -fno-PIC -S -O2 gera o código de montagem abaixo. Comente cada linha, sem exceção, justifique sua existência e associe ao código C. Identifique o uso de ponteiros para acesso facilitado aos elementos das matrizes.

```
prod:
1       endbr32
```

```
2     pushl %esi
3     pushl %ebx
4     movl 20(%esp), %eax
5     xorl  %ebx, %ebx
6     movl 12(%esp), %edx
7     movl 24(%esp), %esi
8     leal  (%eax,%eax,4), %eax
9     leal  (%edx,%eax,4), %eax
10    leal  0(,%esi,4), %ecx
11    addl  16(%esp), %ecx
12    leal  20(%eax), %esi
.L2:
13    movl (%eax), %edx
14    imull (%ecx), %edx
15    addl  $4, %eax
16    addl  $20, %ecx
17    addl  %edx, %ebx
18    cmpl %esi, %eax
19    jne   .L2
20    movl %ebx, %eax
21    popl  %ebx
22    popl  %esi
23    ret
```

b) (2) Qual a dimensão das matrizes?

c) (10) Agora foi feita a compilação com otimização -O3. Comente cada linha, sem exceção, mostrando o que ela está de fato calculando em relação ao produto matricial. Justifique a existência de cada linha, associando ao código C.

```
prod:
1     endbr32
2     pushl  %edi
3     pushl  %esi
4     pushl  %ebx
5     movl  24(%esp), %eax
6     movl  28(%esp), %ecx
7     movl  20(%esp), %ebx
8     leal   (%eax,%eax,4), %edx
9     movl  16(%esp), %eax
10    leal   0(,%ecx,4), %esi
11    movl  (%ebx,%ecx,4), %ecx
12    leal   (%eax,%edx,4), %edx
13    movl  20(%ebx,%esi), %eax
14    imull  (%edx), %ecx
15    imull  4(%edx), %eax
16    leal   (%eax,%ecx), %eax
17    movl  40(%ebx,%esi), %ecx
18    imull  8(%edx), %ecx
19    addl   %eax, %ecx
20    movl  60(%ebx,%esi), %eax
21    imull  12(%edx), %eax
22    addl   %eax, %ecx
23    movl  80(%ebx,%esi), %eax
24    imull  16(%edx), %eax
25    popl   %ebx
26    popl   %esi
27    popl   %edi
28    addl   %ecx, %eax
29    ret
```

d) (5) Vamos calcular os custos dos dois códigos de montagem, usando m para o custo de execução de uma instrução que acessa a memória e por n o custo de execução de uma instrução que não acessa a memória. Para o código da otimização -O2, indique os tipos de cada uma das linhas. Considerando que algumas linhas serão executadas mais de uma vez, calcule o custo total de execução do código. Para o código da otimização -O3, indique os tipos de cada uma das linhas. Calcule o custo total de execução do código. Tente justificar a otimização feita pelo GCC, pensando na eficiência da execução. Analise e conclua a razão da utilização -O3 ser mais eficiente, considerando todos os argumentos possíveis.

Questão 4) (25 pontos) Seja dado o seguinte programa C que chama uma rotina recursiva,

```
void rotina (int n){
        if (n < 0) {putchar('-'); n = -n; }
        if (n/10) rotina (n/10);
        putchar (n%10 + '0');
        }
int main () {rotina (…);}
```

a) (5) Explique e justifique a saída desta rotina recursiva, assumindo $|n| = b_k 10^k + b_{k-1}\, 10^{k-1} + \ldots + b_1 10 + b_0$. Mostre que instância da rotina imprime o quê.

b) (10) A compilação sem otimização gerou o código de montagem que se segue. Comente cada linha deste código e procure então associá-lo ao código C dado. Não descreva o que a instrução faz, pois isso é sabido. Tente entender o que ela representa para o código C e faça a engenharia reversa. Um conjunto de instruções de montagem podem ser necessárias para realizar uma operação em C.

```
rotina:
1     endbr32                  ……..
2     pushl %ebp               ……..
3     movl  %esp, %ebp         ……..
4     subl  $8, %esp           ……..
5     cmpl  $0, 8(%ebp)        ……..
6     jns   .L2                ……..
7     subl  $12, %esp          ……..
8     pushl $45                ……..
9     call  putchar            ……..
10    addl  $16, %esp          ……..
11    negl  8(%ebp)            ……..
.L2:
12    movl  8(%ebp), %eax      ……..
13    addl  $9, %eax           ……..
14    cmpl  $18, %eax          ……..
15    jbe   .L3                ……..
16    movl  8(%ebp), %ecx      ……..
17    movl  $1717986919, %edx  ……..
18    movl  %ecx, %eax         ……..
19    imull %edx               ……..
20    sarl  $2, %edx           ……..
21    movl  %ecx, %eax         ……..
22    sarl  $31, %eax          ……..
23    subl  %eax, %edx         ……..
24    movl  %edx, %eax         ……..
25    subl  $12, %esp          ……..
26    pushl %eax               ……..
27    call  rotina             ……..
28    addl  $16, %esp          ……..
.L3:
29    movl  8(%ebp), %ecx      ……..
30    movl  $1717986919, %edx  ……..
31    movl  %ecx, %eax         ……..
32    imull %edx               ……..
33    sarl  $2, %edx           ……..
34    movl  %ecx, %eax         ……..
35    sarl  $31, %eax          ……..
36    subl  %eax, %edx         ……..
37    movl  %edx, %eax         ……..
38    sall  $2, %eax           ……..
39    addl  %edx, %eax         ……..
40    addl  %eax, %eax         ……..
41    subl  %eax, %ecx         ……..
42    movl  %ecx, %edx         ……..
43    leal  48(%edx), %eax     ……..
44    subl  $12, %esp          ……..
45    pushl %eax               ……..
46    call  putchar            ……..
47    addl  $16, %esp          ……..
48    nop                      ……..
49    leave                    ……..
50    ret                      ……..
```

c) (10) Compilando e executando main () {rotina (0x80000000) é impresso --214748364(, que está claramente errado. Analise e encontre o que está errado no código de montagem e que precisa ser alterado. Apresente a justificativa de sua análise. Indique exatamente as modificações que devem ser feitas para que o programa rode corretamente e apresente a saída correta esperada que é ***-2147483648.*** Não delete ou acrescente linhas ao código. Altere apenas as instruções que foram necessárias, o mínimo possível.