# Lista 3 - MAB353 2020-2 Remoto - 85 pontos

**Questão 1 (20 pontos)** O comando switch teve seu conteúdo apagado e a tarefa é recompor que foi apagado com base no código de montagem.

```
int ex (int x, int n) {
  switch(n) {
    ...
  }
  x=x/2;
  return x;}
```

```
ex:
 1    endbr32
 2    movl    8(%esp), %ecx
 3    movl    4(%esp), %eax
 4    leal    15(%ecx), %edx
 5    cmpl    $30, %edx
 6    ja .L8
 7    notrack jmp *.L4(,%edx,4)
.L4:
 8    .long .L7
 9    .long .L8
10    .long .L8
11    .long .L8
12    .long .L8
13    .long .L6
14    .long .L8
15    .long .L8
16    .long .L8
17    .long .L8
18    .long .L8
19    .long .L8
20    .long .L8
21    .long .L8
22    .long .L8
23    .long .L9
24    .long .L8
25    .long .L8
26    .long .L8
27    .long .L8
28    .long .L3
29    .long .L8
30    .long .L8
31    .long .L8
32    .long .L8
33    .long .L3
34    .long .L8
35    .long .L8
36    .long .L8
37    .long .L8
38    .long .L3
39    .p2align 4,,10
40    .p2align 3
```

```
.L8:
41    xorl %eax, %eax
42    ret
.L7:
43    movl %eax, %edx
44    sall $4, %edx
45    subl %edx, %eax
.L9:
46    movl %eax, %edx
47    shrl $31, %edx
48    addl %edx, %eax
49    sarl %eax
50    ret
.L3:
51    imull %eax, %eax
52    sarl %eax
53    ret
.L6:
54    imull $-10, %eax, %eax
55    sarl %eax
56    ret
```

a) (2) O que é armazenado inicialmente em ecx e em eax?

b) (2) Qual valor é carregado em edx na linha 4. Explique a razão deste valor.

c) (2) O que faz a linha 5? Explique a razão desta operação.

d) (2) Quais condições provocam o desvio para .L8? Explique a instrução ja.

e) (2) Explique a linha 7 e a necessidade do notrack.

f) (10) Escreva o código C completo da rotina ex, inclusive preenchendo o comando switch corretamente. O único return na rotina está mostrado no código C. Atenção e não insira return dentro do switch.

**Questão 2** (20 pontos)

a) (10) Fazendo engenharia reversa no código de montagem, preencha as lacunas do código C. Todas as linhas devem ser preenchidas e nenhuma acrescentada. Reticências podem conter expressões e não apenas um valor, uma variável ou um operador. Atenção aos tipos dos parâmetros de rot.

```
void rot (......... x, ............ p) {
    int i,j;
    ... (.........; ..........; ............)  // ..................................
        ... (......................)            // ........................................
           ......................;              // ........................................
        ......................;                 // ........................................
   }
```

b) (10) Faça a engenharia reversa, comentando todas as as linhas do código de montagem. É preciso associar ao C, e não indicar simplesmente o que a instrução faz, que sabemos. Justifique a presença de instrução no código de montagem se não estiver diretamente relacionada com o C.

```
rot:
    pushl   %edi            ..........................................................
    pushl   %esi            ..........................................................
```

```
    pushl   %ebx            ..........................................................
    movl    16(%esp), %edi  ..........................................................
    movl    20(%esp), %esi  ..........................................................
    movzbl  (%esi), %eax    ..........................................................
    testb   %al, %al        ..........................................................
    je  .L5                 ..........................................................
    leal    1(%esi), %edx   ..........................................................
    movl    $0, %ecx        ..........................................................
    jmp .L4                 ..........................................................
.L3:                        ..........................................................
    addl    $1, %edx        ..........................................................
    movzbl  -1(%edx), %eax  ..........................................................
    testb   %al, %al        ..........................................................
    je  .L2                 ..........................................................
.L4:                        ..........................................................
    movsbl  %al, %ebx       ..........................................................
    cmpl    %edi, %ebx      ..........................................................
    je  .L3                 ..........................................................
    movb    %al, (%esi,%ecx) .........................................................
    leal    1(%ecx), %ecx   ..........................................................
    jmp .L3                 ..........................................................
.L5:
    movl    $0, %ecx        ..........................................................
.L2:
    movb    $0, (%esi,%ecx)  .........................................................
    popl    %ebx            ..........................................................
    popl    %esi            ..........................................................
    popl    %edi            ..........................................................
    ret                     ..........................................................
```

**Questão 3** (20 pontos) Seja dada a seguinte rotina:

```
int n;
int calc(int x){
if (x>10)
n=4*x;
else
n=++x;
return n;
}
```

a) (10) Compile sem otimização, mas com opções -fno-PIC -m32 -march=i686, e gere o código de montagem correspondente. Elimine as diretivas do compilador, as linhas iniciadas por ponto. Enumere as linhas , exceto os endereços .Lx:. Considere um custo m para instruções que acessam a memória e um custo n para as instruções que não acessam.

a1) Comente as linhas do código, associando ao código C.

a2) Indique o custo ? m + ? n desta compilação. Comente sobre as ineficiências que observar.

b) (10 pontos) Compile com otimização -O2 -m32 -fno-PIC -march=i686 e gere o código de montagem correspondente.

b1) Comente as linhas do código, associando ao código C.

b2) Indique o custo ? m + ? n desta compilação. Compare com o anterior sem otimização. Cite as principais alterações observadas que favorecem o desempenho.

**Questão 4 (25 pontos)** Seja o código C:

```
#define N 6
typedef int fix_matrix[N][N];
int fix_prod_ele (fix_matrix A, fix_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) (10) O código compilado acima com -m32 -fno-PIC -S -O2 gera o código de montagem abaixo. Comente cada linha, justifique sua existência e associe ao código C. Identifique o uso de ponteiros para acesso facilitado aos elementos das matrizes.

```
fix_prod_ele:
    endbr32
    pushl %esi
    pushl %ebx
    movl 20(%esp), %eax
    xorl %ebx, %ebx
    movl 12(%esp), %edx
    movl 24(%esp), %esi
    leal (%eax,%eax,2), %eax
    leal (%edx,%eax,8), %eax
    leal 0(,%esi,4), %ecx
    addl 16(%esp), %ecx
    leal 24(%eax), %esi
.L2:
    movl (%eax), %edx
    imull (%ecx), %edx
    addl $4, %eax
    addl $24, %ecx
    addl %edx, %ebx
    cmpl %esi, %eax
    jne .L2
    movl %ebx, %eax
    popl %ebx
    popl %esi
    ret
```

b) (10) Agora foi feita a compilação com otimização -O3. Comente cada linha, justifique sua existência e associe ao código C.

```
fix_prod_ele:
    endbr32
    pushl %edi
    pushl %esi
    pushl %ebx
    movl 24(%esp), %eax
    movl 28(%esp), %edi
    movl 20(%esp), %ecx
```

```
leal (%eax,%eax,2), %edx
movl 16(%esp), %eax
leal 0(,%edi,4), %esi
leal (%eax,%edx,8), %edx
movl 24(%ecx,%esi), %eax
imull 4(%edx), %eax
movl %eax, %ebx
movl (%ecx,%edi,4), %eax
imull (%edx), %eax
addl %ebx, %eax
movl 48(%ecx,%esi), %ebx
imull 8(%edx), %ebx
addl %ebx, %eax
movl 72(%ecx,%esi), %ebx
imull 12(%edx), %ebx
addl %eax, %ebx
movl 96(%ecx,%esi), %eax
imull 16(%edx), %eax
addl %eax, %ebx
movl 120(%ecx,%esi), %eax
imull 20(%edx), %eax
addl %ebx, %eax
popl %ebx
popl %esi
popl %edi
ret
```

c) (5) Tente justificar a otimização feita pelo GCC, pensando na eficiência da execução. Analise e conclua as razões deste novo código ser mais eficiente do que o anterior. Compare os custos dos dois códigos (em função do custo definido na questão 1).