Lista 4 - MAB353 2020-2 Remoto - 120 pontos - v1

Entregue apenas no formato pdf: lista4-nome1-nome2.pdf

Questão 1 (25 pontos) Considere o seguinte programa C:

```
main(){
 short int array[] = {0,1,2,3,4};
 #define TOTAL_ELEMENTS (sizeof(array) / sizeof(array[0]))
 printf("%d\n",sizeof(TOTAL_ELEMENTS));
 printf("%d\n", sizeof(array)/sizeof(array[0]));
 }
```

a) (5) Compile o código com a opção -m32 e gere um executável. Procure explicar a saída impressa. Seja detalhado, justificando cada valor impresso.

b) (5) Compile, sem includes como dado, usando as opções -E -m32 e liste como resposta o que foi obtido, verificando se houve alguma substituição de sizeof no pré-processamento.

c) (10) O código de montagem correspondente é mostrado abaixo. Justifique cada linha do código gerado, associando ao código C mostrado ou a procedimento explicado.

```
main:
01   endbr32                 .....................................................
02   leal 4(%esp), %ecx     .....................................................
03   andl $-16, %esp        .....................................................
04   pushl -4(%ecx)          .....................................................
05   pushl %ebp              .....................................................
06   movl %esp, %ebp        .....................................................
07   pushl %ecx              .....................................................
08   subl $20, %esp         .....................................................
09   movl %gs:20, %eax      .....................................................
10   movl %eax, -12(%ebp)   .....................................................
11   xorl %eax, %eax        .....................................................
12   movw $0, -22(%ebp)     .....................................................
13   movw $1, -20(%ebp)     .....................................................
14   movw $2, -18(%ebp)     .....................................................
15   movw $3, -16(%ebp)     .....................................................
16   movw $4, -14(%ebp)     .....................................................
17   subl $8, %esp          .....................................................
18   pushl $4                .....................................................
19   pushl $.LC0             .....................................................
20   call printf            .....................................................
21   addl $16, %esp         .....................................................
22   subl $8, %esp          .....................................................
23   pushl $5                .....................................................
24   pushl $.LC0             .....................................................
25   call printf            .....................................................
26   addl $16, %esp         .....................................................
27   movl $0, %eax          .....................................................
30   movl -12(%ebp), %edx   .....................................................
31   xorl %gs:20, %edx      .....................................................
32   je .L3                  .....................................................
33   call __stack_chk_fail .....................................................
.L3:
```

```
34   movl -4(%ebp), %ecx   ....................................................
35   leave                   ..................................................
36   leal -4(%ecx), %esp   ....................................................
37   ret                     ..................................................
```

d) (5) Descubra o que seria sizeof, pois não é uma função pelo código acima. Descubra como sizeof é processado.

Questão 2 (45 pontos) O código abaixo tem uma função `soma` que recebe estruturas como argumentos e retorna uma estrutura, além da função `produto` que chama `soma`.

```
typedef struct { int a; int *p;} st1;
typedef struct { int soma; int dif;} st2;

st2 soma(st1 s1) {
  st2 resultado;
  resultado.soma = s1.a + *s1.p;
  resultado.dif = s1.a - *s1.p;
  return resultado;}

int produto (int x, int y) {
  st1 s1;
  st2 s2;
  s1.a = x;
  s1.p = &y;
  s2 = soma(s1);
  return s2.soma * s2.dif;}

soma:
01  endbr32
02  pushl %ebp
03  movl %esp, %ebp
04  subl $16, %esp
05  movl 12(%ebp), %edx
06  movl 16(%ebp), %eax
07  movl (%eax), %eax
08  addl %edx, %eax
09  movl %eax, -8(%ebp)
10  movl 12(%ebp), %edx
11  movl 16(%ebp), %eax
12  movl (%eax), %eax
13  subl %eax, %edx
14  movl %edx, %eax
15  movl %eax, -4(%ebp)
16  movl 8(%ebp), %ecx
17  movl -8(%ebp), %eax
18  movl -4(%ebp), %edx
19  movl %eax, (%ecx)
20  movl %edx, 4(%ecx)
21  movl 8(%ebp), %eax
22  leave
23  ret $4
produto:
24  endbr32
25  pushl %ebp
26  movl %esp, %ebp
27  subl $40, %esp
```

```
28  movl %gs:20, %eax
29  movl %eax, -12(%ebp)
30  xorl %eax, %eax
31  movl 8(%ebp), %eax
32  movl %eax, -28(%ebp)
33  leal 12(%ebp), %eax
34  movl %eax, -24(%ebp)
35  leal -20(%ebp), %eax
36  pushl -24(%ebp)
37  pushl -28(%ebp)
38  pushl %eax
39  call soma
40  addl $8, %esp
41  movl -20(%ebp), %edx
42  movl -16(%ebp), %eax
43  imull %edx, %eax
44  movl -12(%ebp), %ecx
45  xorl %gs:20, %ecx
46  je .L5
47  call __stack_chk_fail
.L5:
48  leave
49  ret
```

a) (10) Comente cada linha do código acima, associando ao código C. Identifique os elementos das estruturas que estão sendo acessados e/ou manipulados. Justifique a ocorrência das instruções.

b) (5) Faça um desenho inicial da pilha no estado imediatamente antes de entrar na execução da função `produto`, mostrando os parâmetros que foram passados para a função, bem como o RIP de quem chamou produto. Mostre que você conhece a passagem de parâmetros para a rotina `produto`. Mostre o alinhamento em múltiplo de 16, sabendo que existe um padrão para isso. Todos os endereços que forem x16 devem ser identificados.

Obs.: Pilha é uma tabela com 3 colunas. Evite fazer a mão e fotografar. Faça no editor, que facilita muito a todos.

c) (10) Complete o desenho da pilha, usando o modelo ex12-editado, a partir do início de execução da função `produto`, passando pela execução da função `soma`, mostrando todo o conteúdo da pilha até o retorno de `soma`, após execução da L23. Ao alterar o topo da pilha para retornar espaços, mantenha o conteúdo da memória e apenas atualize o ponteiro para o topo da pilha.

O desenho da pilha deverá apresentar simultaneamente os registros de ativação de produto e o de soma. Identifique no campo descrição a base de cada um dos registros de forma clara (e.g., %ebp de prod, %ebp de soma). Não deixe de explicar.

d) (5) Quais os endereços das estruturas s1 e s2, manuseadas por produto? Indique isso no desenho da pilha e justifique.

e) (5) O que a instrução na linha 23 faz? Pesquise e seja preciso.

f) (10) Existe uma estratégia geral para o retorno de estruturas por uma função. Analise como a função `soma` sabe onde montar a estrutura resultado na pilha da função `produto`, o que a função `soma` retorna em %eax e entenda e descreva a estratégia. Dê argumentos sólidos baseados no exemplo acima.

Questão 3 (25 pontos) Procura-se resgatar as declarações perdidas de struct1 e a definição de D, a partir do código de montagem referente ao fragmento de código C abaixo. Sabe-se que struct1 contém apenas idx e x[].

```
typedef struct {
   int left;
   struct1 a[D];
   int right;
} struct2;

void test(int i, struct2 *p) {
   int n = p->left + p->right;
   struct1 *q = &p->a[i];
   q->x[p->idx] = n;
}
```

Código de montagem:

```
test:
1    movl    4(%esp), %eax        ................................................
2    movl    8(%esp), %edx        ................................................
3    movl    %eax, %ecx           ................................................
4    sall    $4, %ecx             ................................................
5    sall    $3, %eax             ................................................
6    addl    16(%edx,%ecx), %eax  ................................................
7    movl    52(%edx), %ecx       ................................................
8    addl    (%edx), %ecx         ................................................
9    movw    %cx, 4(%edx,%eax,2)  ................................................
10   ret                          ................................................
```

a) (5) Faça a engenharia reversa, associando as linhas acima ao código C. Identifique o que está sendo calculado, quais variáveis e ponteiros estão sendo manipulados.

b) (5) Encontre o valor de D, apresentando argumentos sólidos para a dimensão do vetor a.

c) (5) Determine sizeof(idx) e sizeof(a) com justificativas claras.

d) (5) Determine o tipo do vetor x e sua dimensão com justificativas claras.

e) (5) Identifique as possíveis declarações da estrutura `struct1`, sabendo que os únicos campos nesta estrutura são `idx` e o vetor `x`. Você tem que justificar os tipos das variáveis e a dimensão dos vetores de forma clara. Ao final, indique as declarações viáveis para `struct1`..

Questão 4 (25 pontos) Sabendo que o programador fez sua escolha de registradores, escreva o comando ASM que originou o seguinte código de montagem, usando a sintaxe (nominal ou posicional) que preferir:

```
int main(){int x=7, z=3, y; asm (...);printf("%d, %d\n",x,y);return z;}

main:
01    leal    4(%esp), %ecx    ..............................................
02    andl    $-16, %esp       ..............................................
03    pushl   -4(%ecx)         ..............................................
04    pushl   %ebp             ..............................................
05    movl    %esp, %ebp       ..............................................
06    pushl   %esi             ..............................................
07    pushl   %ebx             ..............................................
08    pushl   %ecx             ..............................................
09    subl    $16, %esp        ..............................................
10    movl    $7, %esi         ..............................................
11    movl    $3, %ebx         ..............................................
#APP
12    leal    5(%esi), %edx    ..............................................
13    addl    %ebx, %esi       ..............................................
#NO_APP
14    pushl   %edx             ..............................................
15    pushl   %esi             ..............................................
16    pushl   $.LC0            ..............................................
17    call    printf           ..............................................
18    addl    $16, %esp        ..............................................
19    movl    $3, %eax         ..............................................
20    leal    -12(%ebp), %esp  ..............................................
21    popl    %ecx             ..............................................
22    popl    %ebx             ..............................................
23    popl    %esi             ..............................................
24    popl    %ebp             ..............................................
25    leal    -4(%ecx), %esp   ..............................................
26    ret                      ..............................................
```

a) (5) Quais os registradores que GCC escolheu inicialmente para as variáveis x e z? Justifique.

b) (5) Quais os registradores escolhidos pelo programador no comando ASM para armazenar as variáveis x, y e z? Tem que justificar a dedução com argumentos sólidos.

c) (5) Dê argumentos sólidos para configurar a lista de saída do comando ASM.

d) (5) Dê argumentos sólidos para configurar a lista de entrada do comando ASM.

e) (5) Escreva o comando ASM apagado, usando tanto a notação posicional como a nominal.