Lista 4 - MAB353 2021-1 Remoto - 85 pontos

Nome: ———————————————————

Questão 1 (35 pontos) No código C abaixo, `opera` recebe uma estrutura como argumento e retorna outra estrutura. A função `calcula` chama `opera` e retorna um inteiro.

```
typedef struct { short int a1; short int a2;} st1;
typedef struct { int soma; int dif;} st2;

st2 opera (st1 s1) {
  st2 guarda;
  guarda.soma = s1.a1 + s1.a2;
  guarda.dif = s1.a1 - s1.a2;
  return guarda;}

int calcula (short int x, short int y) {
  st1 s1;
  st2 s2;
  s1.a1 = x;
  s1.a2 = y;
  s2 = opera(s1);
  return s2.soma * s2.dif;}
```

O código de montagem eliminadas as diretivas para carregador:

```
opera:
1 endbr32
2 pushl %ebp
3 movl %esp, %ebp
4 subl $16, %esp
5 movzwl 12(%ebp), %eax
6 movswl %ax, %edx
7 movzwl 14(%ebp), %eax
8 cwtl
9 addl %edx, %eax
10 movl %eax, -8(%ebp)
11 movzwl 12(%ebp), %eax
12 movswl %ax, %edx
13 movzwl 14(%ebp), %eax
14 cwtl
15 subl %eax, %edx
16 movl %edx, %eax
17 movl %eax, -4(%ebp)
18 movl 8(%ebp), %ecx
19 movl -8(%ebp), %eax
20 movl -4(%ebp), %edx
21 movl %eax, (%ecx)
22 movl %edx, 4(%ecx)
23 movl 8(%ebp), %eax
24 leave
25 ret $4

calcula:
```

```
26 endbr32
27 pushl %ebp
28 movl %esp, %ebp
29 subl $40, %esp
30 movl 8(%ebp), %edx
31 movl 12(%ebp), %eax
32 movw %dx, -28(%ebp)
33 movw %ax, -32(%ebp)
34 movl %gs:20, %eax
35 movl %eax, -12(%ebp)
36 xorl %eax, %eax
37 movzwl -28(%ebp), %eax
38 movw %ax, -24(%ebp)
39 movzwl -32(%ebp), %eax
40 movw %ax, -22(%ebp)
41 leal -20(%ebp), %eax
42 pushl -24(%ebp)
43 pushl %eax
44 call opera
45 addl $4, %esp
46 movl -20(%ebp), %edx
47 movl -16(%ebp), %eax
48 imull %edx, %eax
49 movl -12(%ebp), %ecx
50 xorl %gs:20, %ecx
51 je .L5
52 call __stack_chk_fail
.L5:
53 leave
54 ret
```

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

b) (5) Assuma que main chama `calcula`. Preencha o desenho da pilha imediatamente antes de executar `calcula`, mostrando os argumentos, a base de `main`, e o RIP para retorno a `main`. Indique o endereço alinhado em x16, pelo padrão do Linux. A pilha cresce de cima para baixo em direção a endereços menores. Use mais linhas se julgar necessário.

endereço (em relação a %esp)	PILHA (4 bytes)	descrição

c) (5) Agora complemente o desenho da pilha, a partir do início de execução da função `calcula` até imediatamente após a execução da linha 44, quando o controle é transferido para a função `opera`. Na coluna endereço, use a referência a %ebp(calcula), a base do registro de ativação da função. Marque todos os endereços x16. Na coluna descrição, use "Após L?"para indicar qual instrução causou a alteração respectiva na pilha. Nesta coluna deve estar indicado o endereço das estruturas s1 e s2.

endereço (em relação a %ebp(calcula)	PILHA (4 bytes)	descrição

Você está desenhando a memória e este desenho continua de onde o desenho anterior parou. Apenas ressalto a referência na coluna endereço que muda ao entrar em nova função.

d) (5) Agora continue o desenho da pilha, a partir do início de execução da função `opera` até imediatamente após a execução da linha 25, quando o controle retorna para a função `calcula`. Na coluna endereço, use a referência a %ebp(opera), a base do registro de ativação da função. Marque todos os endereços x16. Na coluna descrição, use "Após L?"para indicar qual instrução causou a alteração respectiva na pilha. Ao alterar o topo da pilha para retornar espaços, mantenha o conteúdo da memória (que ocorre na prática) e indique para onde está o topo apontando após a linha 25 ser executada.

e) (5) Expresse a instrução na linha 25 em termos de uma sequência equivalente de código de montagem com operação na pilha (pushl/popl) e instrução sobre os registradores %eip e %esp.

f) (5) Analise como a função `opera` monta a estrutura `guarda` na pilha da função `calcula`. Indique o que a função `opera` retorna em %eax. Descreva a estratégia para passar uma estrutura como argumento para uma função e como receber uma estrutura de retorno.

Questão 2 (25 pontos) Procura-se resgatar as declarações perdidas de `struct_a` e a definição de D, a partir do código de montagem referente ao fragmento de código C abaixo. Sabe-se que `struct_a` contém apenas os elementos idx e x[?].

```
typedef struct {
   int left;
   a_struct a[D];
   int right ;
   } b_struct;

void test(int i, b_struct *bp) {
   int n = bp->left + bp->right;
   a_struct *ap = &bp->a[i];
   ap->x[ap->idx] = n;}
```

Código de montagem:

```
test:
1 endbr32
2 pushl %ebx
3 movl 8(%esp), %ecx
4 movl 12(%esp), %edx
5 imull $14, %ecx, %eax
6 movswl 16(%edx,%eax), %ebx
7 leal 0(,%ecx,8), %eax
8 subl %ecx, %eax
9 addl %ebx, %eax
10 movl 60(%edx), %ecx
11 addl (%edx), %ecx
12 movw %cx, 4(%edx,%eax,2)
13 popl %ebx
14 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 suas possíveis dimensões com justificativas claras.

e) (5) Identifique as possíveis declarações da estrutura `struct_a`, 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 `struct_a`.

Questão 3 (25 pontos) Sabendo que o programador fez sua escolha de registradores, escreva o comando `asm` apagado que originou o código de montagem correspondente, usando a sintaxe (nominal ou posicional) que preferir:

```
int main() {
   int x=5, z=3, y=1;
   asm(...
        );
printf("x =%d, y = %d\n", x,y);
return z;
}
```

Código de montagem:

```
main:
1 endbr32
2 leal 4(%esp), %ecx
3 andl $-16, %esp
4 pushl -4(%ecx)
5 pushl %ebp
6 movl %esp, %ebp
7 pushl %edi
8 pushl %esi
9 pushl %ebx
10 pushl %ecx
11 subl $24, %esp
12 movl $5, -36(%ebp)
13 movl $3, -32(%ebp)
14 movl $1, -28(%ebp)
15 movl -36(%ebp), %eax
16 movl -32(%ebp), %edx
17 movl %eax, %esi
18 movl %edx, %edi
#APP
19 leal 5(%esi), %edx
20 addl %edi, %edi
21 movl %edi, %ecx
#NO_APP
22 movl %edi, %eax
23 movl %esi, %ebx
24 movl %edx, -28(%ebp)
25 movl %ebx, -36(%ebp)
26 movl %eax, -32(%ebp)
27 subl $4, %esp
28 pushl -28(%ebp)
29 pushl -36(%ebp)
30 pushl $.LC0
31 call printf
32 addl $16, %esp
33 movl -32(%ebp), %eax
34 leal -16(%ebp), %esp
35 popl %ecx
36 popl %ebx
37 popl %esi
```

```
38 popl %edi
39 popl %ebp
40 leal -4(%ecx), %esp
41 ret
```

a) (5) Quais os registradores que GCC escolheu inicialmente para as variáveis x, y 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.