LISTA 2 - Período 2020-2 Remoto v1
95 pontos

Questão 1) (25 pontos) Na lista 1 foi dado o seguinte programa C:

```
int main (){
    float x = 5.26, y;
    double z = 3.1;
    y = x - z;
    printf ("y = 5.26 - 3.1 = %10.13f \n", y);
    }
```

Compilando com “gcc -m32 -O2 -fno-PIC -S”, descartadas algumas linhas de diretivas, temos:

```
.LC1:
        .string     "y = 5.26 - 3.1 = %10.13f \n"
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       %ecx
8       subl        $4, %esp
9       pushl       $1073825710         // 0x400147AE
10      pushl       $1073741824         // 0x40000000
11      pushl       $.LC1
12      pushl       $1
13      call        __printf_chk
14      movl        -4(%ebp), %ecx
15      addl        $16, %esp
16      xorl        %eax, %eax
17      leave                           ……
18      leal        -4(%ecx), %esp      ……
19      ret                             ……
```

Quando se entra em main, o topo da pilha contém o endereço de retorno para o SO.

endereço	**Conteúdo da Pilha <4 bytes>**	***Comentários***
%esp	RIP SO	*End para retorno ao SO*
%esp - 4		*Fora da pilha*

a) (5) A linha 7 salva o valor de %ecx na pilha e a linha 14 restaura este valor. Por que é necessário salvar este registrador?

b) (5) Desenhe o estado da pilha antes da linha 13 e justifique a existência da linha 8.

c) (5) Você acha que a linha 15 poderia ser suprimida ou ela é essencial? Justifique.

d) (5) O que faz a linha 16? Qual a justificativa para ela?

e) (5) Comente as linhas 17, 18 e 19, mostrando o que cada uma delas causa e, em especial, a existência da linha 18 e sua finalidade explícita.

Questão 2) (30 pontos) Na lista 1 foi dado o seguinte programa:

```
int main (){
    float x = 5.26, y;
    double z = 3.1;
    y = x - z;
    printf ("y = 5.26 - 3.1 = %10.13f \n", y);
    }
```

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

```
.LC2:
        .string     "y = 5.26 - 3.1 = %10.13f \n"
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       %ecx
8       subl        $20, %esp
9       flds  .LC0                  ……...
10      fstps       -24(%ebp)       ……...
11      fldl  .LC1                  ……...
12      fstpl       -16(%ebp)       ……...
13      flds        -24(%ebp)       ……...
14      fsubl       -16(%ebp)       ……...
15      fstps       -20(%ebp)       ……...
16      flds        -20(%ebp)       ……...
17      subl        $4, %esp
18      leal        -8(%esp), %esp
19      fstpl       (%esp)          ……...
20      pushl       $.LC2
21      call        printf
22      addl        $16, %esp
23      movl        $0, %eax
24      movl        -4(%ebp), %ecx
25      leave
26      leal        -4(%ecx), %esp
27      ret
        .section    .rodata
28      .align 4
.LC0:
29      .long       1084772844    //0x40A851EC = 5,26 em precisão simples (da lista 1)
30      .align 8
.LC1:
31      .long       3435973837    //0xCCCCCCCD = parte baixa de 3,1 (da lista 1)
32      .long       1074318540    //0x4008CCCC = parte alta de 3,1 (da lista 1)
        .ident      "GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0"
```

Quando se entra em main, o topo da pilha contém o endereço de retorno para o SO.

endereço	**Conteúdo da Pilha <4 bytes>**	***Comentários***
%esp	RIP SO	*End para retorno ao SO*
%esp - 4		*Fora da pilha*

a) (5) Desenhe a pilha após serem executadas as linhas 1 a 8. Marque os endereços alinhados e a base de main, seguindo o modelo apresentado em ex12-editado.pdf, Semana 4 no classroom.

b) (5) Comente cada uma das linhas de 9 a 13, indicando o que elas causam. Após executar a linha 13, indique o conteúdo de ST(0) e ST(1) na pilha x87.

c) (5) Comente cada uma das linhas de 14 a 16, indicando o que elas causam. Após executar a linha 16, indique o conteúdo de ST(0) e ST(1) na pilha x87.

d) (5) Indique a finalidade e a necessidade das linhas 17 e 18.

e) (5) Mostre o conteúdo completo da pilha antes da execução da linha 21. Indique os endereços alinhados bem como os conteúdos respectivos. Use o campo comentários para relacionar a alteração da pilha com a linha do código. Dica: Veja a questão 1 da lista como subsídio para esta resposta.

f) (5) Justifique as diretivas nas linhas 28 e 30.

Questão 3) (20 pontos) Um valor inteiro foi apagado na rotina C abaixo.

```
int foo(unsigned int n) {unsigned int x; x = n/… ;return x;}
```

O código de montagem gerado pelo GCC com otimização -O2 é:

```
foo:
1    endbr32                          ……
2    movl    4(%esp), %ecx            ……
3    movl    $613566757, %edx         ……
4    movl    %ecx, %eax               ……
5    mull    %edx                     ……
6    movl    %ecx, %eax               ……
7    subl    %edx, %eax               ……
8    shrl    %eax                     ……
9    addl    %edx, %eax               ……
10   shrl    $2, %eax                 ……
11   ret                              ……
```

Comente cada linha, sob o ponto de vista de engenharia reversa, para descobrir o valor apagado. A justificativa tem que ser clara e todas as linhas têm que ser comentadas. Conversões para hexa ou binário podem ser apresentadas sem justificativa. Dica: 613566757= 0x24924925 .

Questão 4) (20 pontos)

É dada uma rotina que recebe um valor inteiro com sinal e imprime o resultado da divisão, que pode ser negativo ou positivo. Determine o valor do divisor apagado na rotina C abaixo:

int foo (int n) {int x; x = n/… ; return x;}

O código de montagem gerado pelo GCC com otimização -O2 é:

```
foo:
1    endbr32                          …….
2    movl    4(%esp), %ecx            …….
3    movl    $-1840700269, %edx       …….
4    movl    %ecx, %eax               …….
5    imull   %edx                     …….
6    leal    (%edx,%ecx), %eax        …….
7    sarl    $31, %ecx                …….
8    sarl    $2, %eax                 …….
9    subl    %ecx, %eax               …….
10   ret                              …….
```

Comente cada linha, sob o ponto de vista de engenharia reversa, para descobrir o valor apagado. A justificativa tem que ser clara e todas as linhas têm que ser comentadas. Conversões para hexa ou binário podem ser apresentadas sem justificativa. Dica: -1840700269 = 0x92492493.