1) Para resolver esta questão, considere o seguinte código fonte em linguagem C. Na coluna da esquerda, temos os endereços das instruções que estão na coluna da direita.

```
0x2200        void bar(void) {
                  short v[4];
0x2210            v[0] = 0;
              }
0x2300        void boo(void) {
                  int i[6];
0x2310            bar();
              }
0x2000        int main(void) {
0x2010            boo();
0x2020            return 0;
              }
```

Considere também os seguintes valores iniciais para os registradores imediatamente antes que a função main crie o seu Registro de Ativação:

- RSP: **0x1008**
- RIP: **0x2000**
- RBP: **0x10F0**

Considere que a função main irá retornar para o endereço 0x8752 ao terminar a sua execução (endereço da próxima instrução da função chamadora da função main).

a. Mostre o estado da pilha imediatamente antes da execução da chamada à função boo na função main no endereço 0x2010 desde do início do programa, imediatamente antes que a função main crie o seu Registro de Ativação (endereço do topo da pilha 0x2008). Para responder a esse item preencha a seguinte tabela, onde a coluna da esquerda representa o endereço de memória e a da direita, o seu conteúdo. A parte de cima possui os endereços mais baixos e a parte de baixo, os endereços de maior valor. Se precisar, inclua mais linhas. Respostas sem justificativa não serão consideradas.

Endereço	Conteúdo
0x0FF0	
0x0FF8	

0x1000	
0x1008	

b. Mostre o estado da pilha imediatamente antes da execução da chamada à função bar na função boo no endereço 0x2310 desde do início do programa, imediatamente antes que a função main crie o seu Registro de Ativação (endereço do topo da pilha 0x2008). Para responder a esse item preencha a seguinte tabela, onde a coluna da esquerda representa o endereço de memória e a da direita, o seu conteúdo. A parte de cima possui os endereços mais baixos e a parte de baixo, os endereços de maior valor. Se precisar, inclua mais linhas. Respostas sem justificativa não serão consideradas.

Endereço	Conteúdo
0x0FF0	
0x0FF8	
0x1000	
0x1008	

2) Considere a sequência de 32 bits a seguir:

   00111010111110000010000001111011

   Considerando que ela representa um número em ponto flutuante com regras semelhantes às IEEE 754, mas utilizando 1 bit para o sinal, 6 bits para expoente e o restante para a parte fracionária, expresse o valor em decimal. Explique como você chegou a esse resultado.

3) Traduza a função boofoo abaixo para assembly IA-64 **com base no assembly visto em aula restrito às instruções que estão no pdf em** http://www4.inf.puc-rio.br/~inf1018/docs/sbinst64___Instrucoes_X64_padrao_AT_T.pdf. **Não serão**

aceitas respostas com instruções fora desta lista ou que apresentem construções típicas de um compilador.

Utilize as regras usuais de alinhamento, passagem de parâmetros, salvamento de registradores e resultados em C/Linux. **Traduza o mais diretamente possível** o código de C para assembly. Comente seu código! Não se preocupe em entender a finalidade das função.

```
struct Y {
   unsigned char  ind;
   double purp;
   struct Y *prox;
};

float boofoo(struct Y *v)
{
   double w[10];
   float acc = 0.0;
   int i;

   for (i=0; i<10;i++)
      w[i] = 0.0;

   while(v)
   {
      if (v->ind < 10)
        w[v->ind] = v->purp;
      v = v->prox;
   }

   for (i=0; i<10;i++)
      acc += (float) w[i];

   return acc;
}
```