Trabalho TDD

Problema 1: Encontrar e retornar a posição do primeiro bit igual a 1 em uma variável de 32 bits sem sinal, a partir do bit mais significativo.

Setup inicial do código:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

/* macros de testes - baseado em minUnit: www.jera.com/techinfo/jtns/jtn002.html */
#define verifica(mensagem, teste) do { if (!(teste)) return mensagem; } while (0)
#define executa_teste(teste) do { char *mensagem = teste(); testes_executados++; \
                                  if (mensagem) return mensagem; } while (0)
int testes_executados = 0;

static char * executa_testes(void);
static int firstbiteq1 (uint32_t x);

int main()
{
    char *resultado = executa_testes();
    if (resultado != 0)
    {
        printf("%s\n", resultado);
    }
    else
    {
        printf("TODOS OS TESTES PASSARAM\n");
    }
    printf("Testes executados: %d\n", testes_executados);

    return resultado != 0;
}
```

1º Teste falhando:

```
29
30  /* Função que calcula a posição do MSB */
31  /*********************************************/
32  static int msbPosition(uint32_t x)
33  {
34
35      return x;
36
37  }
38
39  static char * teste_retorna0_caso_1(void)
40  {
41      verifica("erro: deveria retornar 0", msbPosition(1) == 0);
42      return 0;
43  }
44
45  /*********************************************/
46
47  static char * executa_testes(void)
48  {
49      /* Executa testes */
50      executa_teste(teste_retorna0_caso_1);
51      return 0;
52  }
```

```
PROBLEMAS   SAÍDA   TERMINAL   CONSOLE DE DEPURAÇÃO

michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$ gcc exemplo_tdd.c -o tdd
michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$ ./tdd
erro: deveria retornar 0
Testes executados: 1
michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$
```

1º Teste passando:

```
29
30  /* Função que calcula a posição do MSB */
31  /*********************************************/
32  static int msbPosition(uint32_t x)
33  {
34                              /* 1 -> position */
35      if(x == 1) return 0; // 01 -> 0
36
37  }
38
39  static char * teste_retorna0_caso_1(void)
40  {
41      verifica("erro: deveria retornar 0", msbPosition(1) == 0);
42      return 0;
43  }
44
45  /*********************************************/
46
47  static char * executa_testes(void)
48  {
49      /* Executa testes */
50      executa_teste(teste_retorna0_caso_1);
51      return 0;
52  }
```

```
PROBLEMAS   SAÍDA   TERMINAL   CONSOLE DE DEPURAÇÃO

michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$ gcc exemplo_tdd.c -o tdd
michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$ ./tdd
TODOS OS TESTES PASSARAM
Testes executados: 1
michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$
```

2º Teste falhando:

```
29
30  /* Função que calcula a posição do MSB */
31  /*********************************************/
32  static int msbPosition(uint32_t x)
33  {
34                                  /* 1 -> position */
35      if(x == 1) return 0; // 01 -> 0
36      else return -1;
37
38  }
39
40  static char * teste_retorna0_caso_1(void)
41  {
42      verifica("erro: deveria retornar 0", msbPosition(1) == 0);
43      return 0;
44  }
45
46  static char * teste_retorna1_caso_2(void)
47  {
48      verifica("erro: deveria retornar 1", msbPosition(2) == 1);
49      return 0;
50  }
51
52  /*********************************************/
53
54  static char * executa_testes(void)
55  {
56      /* Executa testes */
57      executa_teste(teste_retorna0_caso_1);
58      executa_teste(teste_retorna1_caso_2);
59      return 0;
60  }
```

```
PROBLEMAS   SAÍDA   TERMINAL   CONSOLE DE DEPURAÇÃO

michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$ gcc exemplo_tdd.c -o tdd
michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$ ./tdd
erro: deveria retornar 1
Testes executados: 2
michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$
```

2º Teste passando:

```
29
30  /* Função que calcula a posição do MSB */
31  /*********************************************/
32  static int msbPosition(uint32_t x)
33  {
34                                  /* 1 -> position */
35      if(x == 1) return 0; // 01 -> 0
36                                  /* 2 -> position */
37      if(x == 2) return 1; // 10 -> 1
38      else return -1;
39
40  }
41
42  static char * teste_retorna0_caso_1(void)
43  {
44      verifica("erro: deveria retornar 0", msbPosition(1) == 0);
45      return 0;
46  }
47
48  static char * teste_retorna1_caso_2(void)
49  {
50      verifica("erro: deveria retornar 1", msbPosition(2) == 1);
51      return 0;
52  }
53
54  /*********************************************/
55
56  static char * executa_testes(void)
57  {
58      /* Executa testes */
59      executa_teste(teste_retorna0_caso_1);
60      executa_teste(teste_retorna1_caso_2);
61      return 0;
62  }
```

```
PROBLEMAS   SAÍDA   TERMINAL   CONSOLE DE DEPURAÇÃO

michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$ gcc exemplo_tdd.c -o tdd
michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$ ./tdd
TODOS OS TESTES PASSARAM
Testes executados: 2
michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$
```

3º Teste falhando:

```
30  /* Função que calcula a posição do MSB */
31  /*********************************************/
32  static int msbPosition(uint32_t x)
33  {
34                                  /* 1 -> position */
35      if(x == 1) return 0;       // 01 -> 0
36                                  /* 2 -> position */
37      if(x == 2) return 1;       // 10 -> 1
38      else return -1;             /* 3 -> position */
39                                 // 11 -> 1
40
41  }
```

3º Teste passando:

```
30  /* Função que calcula a posição do MSB */
31  /*********************************************/
32  static int msbPosition(uint32_t x)
33  {
34                                      /* 1 -> position */
35      if(x == 1) return 0;           // 01 -> 0
36                                      /* 2 -> position */
37      if(x == 2 || x == 3) return 1; // 10 -> 1
38      else return -1;                 /* 3 -> position */
39                                     // 11 -> 1
40
41  }
```

4º Teste falhando:

```
29
30  /* Função que calcula a posição do MSB */
31  /*********************************************/
32  static int msbPosition(uint32_t x)
33  {
34                                      /* 1 -> position */
35      if(x == 1) return 0;           // 01 -> 0
36                                      /* 2 -> position */
37      if(x == 2 || x == 3) return 1; // 10 -> 1
38                                      /* 3 -> position */
39                                     // 11 -> 1
40                                      /* 4 -> position */
41                                    // 100 -> 2
42      else return -1;
43
44  }
```

4º Teste passando:

```
29
30  /* Função que calcula a posição do MSB */
31  /*********************************************/
32  static int msbPosition(uint32_t x)
33  {
34                                      /* 1 -> position */
35      if(x == 1) return 0;           // 01 -> 0
36                                      /* 2 -> position */
37      if(x == 2 || x == 3) return 1; // 10 -> 1
38                                      /* 3 -> position */
39                                     // 11 -> 1
40                                      /* 4 -> position */
41      if(x == 4) return 2;           // 100 -> 2
42      else return -1;
43
44  }
```

Refatoração do código:
Vamos observar os valores que testamos até o momento:

Decimal	Binário	Posição MSB
4	100	2
3	011	1
2	010	1
1	001	0

Podemos observar que do valor maior ocorre um shift no bit mais significativo, da esquerda para a direita.

Dessa forma, podemos tentar refatorar o código, para que, ao invés de comparar o número recebido com o valor correspondente, buscar uma ferramenta na linguagem que realize o shift do MSB, onde possamos contar esse deslocamento, dessa forma vamos poder descobrir a sua posição.

Ao pesquisar na documentação da linguagem, foi possível observar alguns recursos para trabalhar com números binários, diretamente. Um dos recursos chama-se left shift (<<) ou right shift (>>), justamente o recurso que estamos buscando.

Podemos utilizar uma variável pré declarada com valor zero para utilizar como contador e declarar um laço que vai se repetir até o valor se tornar igual a zero. Por fim, retornamos à variável utilizada para contar e subtraímos o valor 1, visto que a posição do MSB inicia por zero.

O código fica dessa forma:

```
30  /* Função que calcula a posição do MSB */
31  /*********************************************/
32  static int msbPosition(uint32_t x)
33  {
34      //                                /* 1 -> position */
35      //if(x == 1) return 0;            // 01 -> 0
36      //                                /* 2 -> position */
37      //if(x == 2 || x == 3) return 1;  // 10 -> 1
38      //                                /* 3 -> position */
39      //                               // 11 -> 1
40      //                                /* 4 -> position */
41      //if(x == 4) return 2;           // 100 -> 2
42      //else return -1;
43
44      uint32_t count = 0;
45
46      while (x)
47      {
48          count++;
49          x >>= 1;
50      }
51
52      return count - 1;
53
54  }
```

Realizando mais alguns testes notamos que o algoritmo está funcionando bem:

```
 97
 98  /*********************************************/
 99
100  static char * executa_testes(void)
101  {
102      /* Executa testes */
103      executa_teste(teste_retorna0_caso_1);
104      executa_teste(teste_retorna1_caso_2);
105      executa_teste(teste_retorna1_caso_3);
106      executa_teste(teste_retorna2_caso_4);
107      executa_teste(teste_retorna2_caso_5);
108      executa_teste(teste_retorna3_caso_8);
109      executa_teste(teste_retorna4_caso_16);
110      return 0;
111  }
```

```
PROBLEMAS   SAÍDA   TERMINAL   CONSOLE DE DEPURAÇÃO

michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$ gcc exemplo_tdd.c -o tdd
michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$ ./tdd
TODOS OS TESTES PASSARAM
Testes executados: 7
michael@pop-os:~/Documentos/faculdade/Embarcados/trab-01$
```