

INF1018 - Software Básico (2019.2)

Primeiro Trabalho

Inteiros Grandes

O objetivo do trabalho é implementar, na linguagem C, uma biblioteca que permita representar valores inteiros *signed* de 128 bits, e ofereça algumas operações aritméticas básicas sobre esses valores (soma, subtração, multiplicação, negação) e operações de deslocamento de bits.

Instruções Gerais

- Leia com atenção o enunciado do trabalho e as instruções para a entrega. [Em caso de dúvidas, não invente. Pergunte!](#)
 - O trabalho deve ser entregue até as 23h do dia 07/10.
 - Trabalhos entregues após o prazo perderão um ponto por dia de atraso.
 - Trabalhos que não compilem (i.e., que não produzam um executável) [não serão considerados](#), ou seja, receberão grau zero.
 - Os trabalhos podem ser feitos em grupos de [no máximo](#) dois alunos.
 - Alguns grupos poderão ser chamados para apresentar seus trabalhos.
-

Representação para Inteiros Grandes

Para representar um valor inteiro de 128 bits, a biblioteca deverá usar a seguinte definição:

```
#define NUM_BITS 128
typedef unsigned char BigInt[NUM_BITS/8];
```

Ou seja, um valor do tipo `BigInt` deve ser representado por um *array de bytes*, interpretado como um único inteiro de 128 bits, em complemento a 2 e seguindo a ordenação **little-endian**.

Por exemplo, o array

```
{0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}
```

representa o valor inteiro 1.

O array a seguir

```
{0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}
```

representa o inteiro `0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE`, que corresponde ao valor -2 em complemento a 2.

Operações da Biblioteca

A implementação das funções disponibilizadas pela biblioteca deve obedecer os protótipos fornecidos abaixo:

```

/* Atribuição (com extensão) */
void big_val (BigInt res, long val);

/* Operações Aritméticas */

/* res = -a */
void big_comp2(BigInt res, BigInt a);

/* res = a + b */
void big_sum(BigInt res, BigInt a, BigInt b);

/* res = a - b */
void big_sub(BigInt res, BigInt a, BigInt b);

/* res = a * b */
void big_mul(BigInt res, BigInt a, BigInt b);

/* Operações de Deslocamento */

/* res = a << n */
void big_shl(BigInt res, BigInt a, int n);

/* res = a >> n (lógico) */
void big_shr(BigInt res, BigInt a, int n);

/* res = a >> n (aritmético) */
void big_sar(BigInt res, BigInt a, int n);

```

- A função **big_val** atribui a `res` o valor fornecido por `val` (um *signed long*), corretamente estendido para 128 bits.
- A função **big_comp2** atribui a `res` o valor "negado" (complemento a 2) de `a`.
- As funções **big_sum**, **big_sub** e **big_mul** realizam, respectivamente, as operações de soma, subtração e multiplicação de dois inteiros de 128 bits (`a` e `b`), armazenando o resultado em `res`.
- A função **big_shl** realiza um deslocamento para a esquerda (*shift left*) de um inteiro de 128 bits (`a`), armazenando em `res` o resultado da operação. Sua implementação pode assumir que o valor de `n` é um inteiro no intervalo `[0,127]`.
- A função **big_shr** realiza um deslocamento **lógico** para a direita (*shift right*) de um inteiro de 128 bits (`a`), armazenando em `res` o resultado da operação. Sua implementação pode assumir que o valor de `n` é um inteiro no intervalo `[0,127]`.

Implementação e Execução

Você deve criar um arquivo fonte chamado **bigint.c**, contendo a implementação das funções descritas acima e funções auxiliares, se for o caso. Esse arquivo **não deve conter uma função main!**

O arquivo **bigint.c** deverá incluir o arquivo de cabeçalho **bigint.h**, fornecido [aqui](#). Você **não deve** alterar este arquivo. Caso necessite de definições auxiliares, faça-as dentro de seu arquivo fonte **bigint.c**.

Para testar seu programa, crie um outro arquivo, por exemplo, **teste.c**, contendo uma função **main**. Este arquivo também deverá incluir o arquivo de cabeçalho **bigint.h**.

Crie seu programa executável (por exemplo, **teste**) com a linha:

```
gcc -Wall -o teste bigint.c teste.c
```

Importante! Se, por acaso, você não implementar alguma função, crie uma função *dummy* para que o programa de teste usado para a correção do trabalho possa ser gerado sem problemas.

Essa função deverá ter o nome da função não implementada e o corpo

```
{ return; }
```

Dicas

- **Implemente seu trabalho por partes, testando cada parte implementada antes de prosseguir.**

Por exemplo, você pode implementar primeiro a operação de atribuição, e para testá-la usar alguma função auxiliar que realize um *dump* do array usado para armazenar o inteiro de 128 bits. Em seguida, vá implementando **e testando**, uma a uma, as outras funções.

- Repare que algumas funções podem ser usadas pela implementação de outras (por exemplo, a operação de subtração pode usar soma e complemento a 2!)
 - Não se esqueça de testar bem seu programa antes de entregá-lo! Para cada operação implementada, pense nos diferentes casos relevantes (números positivos, negativos, muito grandes, zero, etc...).
 - Algumas operações possuem uma implementação evidente, enquanto outras requerem um pouco mais de investigação. Por exemplo, a multiplicação pode usar somas sucessivas, mas esta seria uma implementação ineficiente...
-

Entrega

Devem ser entregues **via Moodle** dois arquivos:

1. o arquivo fonte **bigint.c**

Coloque no início do arquivo fonte, como comentário, os nomes dos integrantes do grupo, da seguinte forma:

```
/* Nome_do_Aluno1 Matricula Turma */  
/* Nome_do_Aluno2 Matricula Turma */
```

Lembre-se que este arquivo não deve conter a função `main`!

2. um arquivo texto, chamado **relatorio.txt**, relatando o que está funcionando e, eventualmente, o que não está funcionando. Mostre alguns exemplos de valores testados (casos de sucesso e casos de insucesso, se estes existirem).

Você não deve explicar a sua implementação neste relatório. Seu programa deve ser suficientemente claro e bem comentado.

Coloque também no relatório o nome dos integrantes do grupo.

Coloque na área de texto da tarefa do Moodle os nomes e turmas dos integrantes do grupo.

Para grupos de alunos da mesma turma, apenas uma entrega é necessária (usando o *login* de um dos integrantes do grupo).

Se o grupo for composto por alunos de turmas diferentes, os dois alunos deverão realizar a entrega, e avisar aos professores!

