Arquitetura de Computadores 2017/18

TPC 1

Prazo de entrega: 23:59 de 4 de abril de 2018

Tópicos: *Programação em C. Representação de inteiros.*

Este trabalho de casa consiste em dois exercícios de programação **individual**. Pode esclarecer dúvidas gerais com colegas, mas a solução e a escrita do código deve ser estritamente individual. Todas as resoluções serão comparadas de forma automática e os casos de plágio serão punidos de acordo com os regulamentos em vigor.

A entrega será via Mooshak do DI (`http://mooshak.di.fct.unl.pt/~mooshak/`). Os programas serão compilados com o seguinte comando:

```
cc -m32 -Wall -std=c11 -o prog  prog.c
```

Exercício 1

Pretende-se demonstrar como é possível suportar operações sobre tipos de dados com dimensão superior à palavra suportada pela arquitetura em que se executa (por exemplo, na linguagem C, **`long long`** pode representar um inteiro de 64bits numa arquitetura de 32bits, suportando o compilador as tradicionais operações aritméticas). Assim, implemente duas funções que permitam efetuar a soma e multiplicação de dois **inteiros sem sinal de 64 bits usando apenas 32 bits**. No caso da multiplicação pretende-se apenas multiplicar um número de 64bits por outro de 32.

Para representar os números inteiros sem sinal, deve agrupar duas palavras de 32bits como sendo a representação dos 64. Para tal cada número será representado num vetor de dois **`unsigned int`**, ficando os bits mais significativos do número na posição 1 do vetor.

A função para a soma terá a seguinte assinatura:

```
void soma64( unsigned n1[2], unsigned n2[2] );
```

sendo guardado em n1 o resultado da soma n1 + n2.

Note que a soma parcelar será:

n1[0] = n1[0]+n2[0] -> pode sobrar Carry

n1[1] = n1[1]+n2[1]+Carry

A função para a multiplicação terá a seguinte assinatura:

```
void mul64( unsigned n1[2], unsigned n2 );
```

sendo guardado em n1 o resultado de n1 x n2.

Note que se pode fazer a multiplicação por parcelas como na soma (mas a *carry* será difícil de calcular usando apenas 32bits) ou por outros algoritmos, mas recomenda-se para este problema que use apenas somas sucessivas usando a função anterior. Exemplo:

*n1 * 3 = n1 + n1 + n1*

Veja no anexo a este enunciado o programa que deve completar e usar para testar as suas funções (também fornecido no CLIP em tpc1.zip). Este será o programa a entregar no Mooshak.

Exercício 2

Implemente em C uma função que imprima no ecrã valores do tipo int em base 10. A conversão para caracteres tem de ser realizada por essa função. Use apenas a função putchar para escrever cada um dos caracteres e scanf para ler o inteiro. Não pode usar qualquer outra função auxiliar (p.e. printf).

```
void printInt( int val );
```

Exemplo de utilização:

```
printInt( -325 );
```

produz um efeito equivalente a:

```
printf("%d", -325 );
```

Complete o programa seguinte para testar a sua solução (também fornecido no CLIP em tpc1.zip). Este será o programa a entregar no Mooshak.

```
#include <stdio.h>

void printInt( int val ) {
  // TODO
}

int main( ) {
  int x;

  scanf( "%d", &x );
  printInt( x );
  putchar('\n');
  return 0;
}
```

Anexo - Código para teste do exercício 1

```
#include <stdio.h>
#include <limits.h>

// converter unsigned long long (64 bits s/sinal) para a
// representacao em dois unsigned int (32 bits s/sinal cada)
void ULL2myint( unsigned n[2], unsigned long long num ) {
  n[0] = num & 0xffffffff;
  n[1] = (num>>32) & 0xffffffff;
}

// converter representacao de 64bits em dois unsigned int
// para o tipo unsigned long long (64 bits s/sinal)
unsigned long long myint2ULL( unsigned n[2] )  {
  unsigned long long num = n[1];
  return (num<<32) + n[0];
}


void soma64( unsigned n1[2], unsigned n2[2] ) { // n1 = n1 + n2
 // TODO
}

void mul64( unsigned n1[2], unsigned n2 ) { // n1 = n1 * n2
 // TODO
}


int main( ) {
  unsigned long long x;
  unsigned long long y;
  unsigned z;
  unsigned n1[2];
  unsigned n2[2];

  scanf("%llu", &x);   // le tres numeros. X sera usado na soma e multiplicacao
  scanf("%llu", &y);   // y sera usado na soma
  scanf("%u", &z);     // z sera usado na multiplicacao

  printf( "soma pelo C: %llu\n", x+y );

  ULL2myint( n1, x );
  ULL2myint( n2, y );
  soma64( n1, n2 );  // n1 = n1 + n2
  unsigned long long result = myint2ULL( n1 );
  printf( "soma por soma64: %llu\n", result );

  printf( "multiplicacao pelo C: %llu\n", x*z );

  ULL2myint( n1, x );
  mul64( n1, z );
  result = myint2ULL( n1 );
  printf( "multiplicacao por mul64: %llu\n", result );

  return 0;
}
```