DCC006: Organização de computadores I **Data de entrega: 25/05/2022**

Trabalho Prático #1

Professor: Omar Paranaiba Vilela Neto
Monitor: Luís Fernando Miki

Antes de começar seu trabalho, leia todas as instruções abaixo.

- O trabalho deve ser feito individualmente. Cópias de trabalho acarretarão em devida penalização às partes envolvidas.
- Entregas após o prazo serão aceitas, porém haverá uma penalização. Quanto maior o atraso maior a penalização.
- Submeta apenas um arquivo .zip contendo as suas soluções e um arquivo .txt com seu nome e matrícula. Nomeie os arquivos de acordo com a numeração do problema a que se refere. Por exemplo, o arquivo contendo a solução para o problema 1 deve ser nomeado 1.s. Se for solicitado mais de uma implementação para o mesmo problema nomeie 1a.s, 1b.s e assim por diante.
- O objetivo do trabalho é praticar as suas habilidades na linguagem assembly. Para isso, você utilizará o ***Venus Simulator*** (https://www.kvakil.me/venus/). Venus é um simulador de ciclo único que te permite enxergar o valor armazenado em cada registrador e seguir a execução do seu código linha a linha. O simulador foi desenvolvido por Morten Petersen e possui a ISA do RISC-V, embora apresente algumas alterações. Você pode utilizar o seguinte link: https://github.com/mortbopet/Ripes/blob/master/docs/introduction.md para verificar as modificações da sintaxe ISA utilizada pelo simulador. Note que no livro e material da disciplina os registradres são de 64 bits, mas o simulador utilizada registradores de apenas 32 bits. Para utilizar o simulador basta você digitar seu código aba ***Editor*** e para executá-lo basta utilizar a aba ***Simulator***

Problema 1: Capacitor (2 pontos)

Em circuitos elétricos, a capacitância ou capacidade elétrica é a grandeza escalar que mede a capacidade de armazenamento de energia em equipamentos e dispositivos elétricos. O dispositivo mais usual para armazenar carga é o capacitor, presente nos circuitos que compõem os computadores pessoais. A capacitância C é calculada pela relação entre a diferença de potencial V (ou tensão elétrica) existente entre as placas do capacitor e a carga elétrica nele armazenada Q:

$$\mathrm{C} = \frac{Q}{V}$$

Escreva um programa *assembly* que, dadas duas dessas grandezas quaisquer, o programa possa calcular e retornar o valor da terceira grandeza. Considere x10, x11 e x12 como e capacitância (C), carga (Q) e a diferença de potencial (V) respectivamente e a variável com valor zero é a que deve ser calculada. Caso mais de uma váriável tenha valor zero, seu programa deve retornar também zero.

Problema 2: Fibonacci (4 pontos)

A sequência de Fibonacci é uma sequência matemática que aparece em diversas situações na natureza inclusive possuindo sua utilidade no mercado de ações, na ciência da computação e até em jogos digitais. A sequência começa pelo 0 seguido do 1, os termos susbsequentes são a soma dos dois anteriores:

0,1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, ...

Abaixo está indicado um algoritmo recursivo (com sua implementação em C) para calcular o n-ésimo termo da sequência:

```c
int fib(int n){
    if (n==0)
     return 0;
    else if (n == 1)
     return 1;
    else
     return fib( n 1 ) + fib( n 2 );
}
```

Implemente o algoritmo acima em assembly, considere que o número n está no registrador x10.
Dica: você pode usar o registrador x2 *(stack pointer)* para armazenar os valores dos parâmetros de cada chamada recursiva na memória.

Problema 3: Ordenação (4 pontos)

Imagine que você recebeu uma tarefa de ordernar uma fileira de pessoas de forma crescente da esquerda para a direita de acordo com a altura de cada uma. Todavia, você consegue apenas comparar a altura de duas pessoas por vez e então, começa a ordernar a partir das duas pessoas mais à esquerda da fileira, trocando a posição das duas caso a segunda pessoa seja menor. Em seguida, faz o mesmo processo, agora com a atual segunda pessoa da fila e a terceira. Você repete o processo até deixar a pessoa mais alta da fileira na extrema direita. Em seguida, você volta para o começo da fila e começa tudo de novo, porém agora, até a penúltima pessoa da direita e assim por diante até obter a fileira ordernada.

Esta é a ideia por trás do algoritmo de ordenação *bubble sort*. Abaixo, há uma possıvel implementação do *bubble sort* em C:

```c
void bubble_sort(int a[], int n) {
    int i = 0, j = 0, tmp;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (a[j] > a[j + 1]) {
                tmp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = tmp;
            }
        }
    }
}
int main() {

  int arr[] = {27 , 16 , 20 , 49 , 39 , 48 , 38 , 7 , 5 , 19};
  int n = sizeof(arr) / sizeof(arr[0]);

  bubble_sort(arr, n);
  return 0;
}
```

Implemente o *bubble sort* em assembly, considere que o começo do array está em x10 e o tamanho do array está em x11. Para esta atividade você pode utilizar arrays ou ponteiros.

Dicas e sugestões

- Não deixe o trabalho para o último dia. Não viva perigosamente!
- Comente seu código sempre que possıvel. Isso será visto com bons olhos.
- O monitor está disponıvel para atender a quaisquer dúvidas sobre esse trabalho. Segue o email para contato com o monitor: **luisfmiki@gmail.com**. Insira a tag [DCC006] no assunto do email.