MEMORIAL DESCRITIVO
LABORATÓRIO DE PROGRAMAÇÃO ASSEMBLY 2

AMANDA DUARTE GARCIA - 12221BCC031
CAIKE CESAR MOTA DE ARAUJO - 12221BCC030
FELIPE SANTOS SILVA - 12221BCC042
MATHEUS FIOD SALIBA - 12221BCC024

P1. CALCULANDO TAMANHO DE UMA STRING

Este código em MIPS Assembly foi desenvolvido com o propósito de calcular o comprimento de uma string fornecida pelo usuário. Ele compreende tanto a função principal **main**, que interage com o usuário, quanto a função **string_lenght**, responsável por calcular o comprimento da string.

No segmento .data, é reservado um espaço de 100 bytes para o buffer que armazenará a entrada do usuário, e é definido um prompt para solicitar a palavra do usuário. A partir disso, no segmento .text, temos as funções:

1) **main**
   Imprime o prompt para usuário, solicitando a entrada de uma palavra. Em seguida, lê a entrada do usuário e a armazena no buffer. Depois, a função **main** chama a função **string_lenght** para calcular o comprimento da string. Por fim, a função imprime o comprimento da string e finaliza o programa;
2) **string_lenght**
   Recebe o endereço inicial da string e utiliza três registradores para iterar pela string para contar seu comprimento. Com o contador, inicializado como zero, sendo incrementado a cada iteração, a iteração acontece byte a byte até que o caractere nulo (\0) seja encontrado. Por fim, retorna o comprimento da string, desconsiderando o caractere nulo.

P2. CONVERTENDO STRING PARA LOWERCASE

Este código converte os caracteres de uma string para lowercase. A partir de uma **string** definida no segmento .data do código, o programa analisa cada caractere e faz a devida alteração por meio das seguintes funções:

1) **main**
   Define um registrador para a **string** que será convertida, além de um **contador**;
2) **loop**
   Carrega um caractere e analisa se é nulo ou não. Se for nulo, o processo é encerrado, senão, é conferido se esse caractere está entre o intervalo de ‘A’ a ‘Z’ na tabela ASCII. Se estiver, o caractere é convertido para um caractere minúsculo (de ‘a’ a ‘z’);
3) **minu**
   Função responsável por incrementar o **contador** e permitir que o programa acesse todos os caracteres da **string**;
4) **fim**
   Imprime na tela a **string** modificada, totalmente em caracteres minúsculos (lowercase) e finaliza o programa.

P3. FUNÇÃO SENO

Para realizar o cálculo do seno de um dado ângulo, foram declaradas a variável **x** e as constantes **pi** (3.141592), **divisor** (180.0), **precisão** (0.00001) e **um** (1.0), além de uma string **resultado** para imprimir o resultado no final do programa. Além disso, foram estabelecidas as funções:

1) **main**
   Converte o ângulo **x** (em graus) para radianos, a partir do cálculo **x * pi / 180.0**;
2) **senox**
   Prepara o programa para o cálculo do seno de **x**, definindo **resultado** e **precisão** em seus respectivos registradores, além de inicializar o fator $\mathbf{x^3}$ e o **sinal** (alternância);
3) **loop**
   Faz o cálculo do seno de **x** a partir de várias iterações, que se encerram quando o termo atual for menor que a precisão;
4) **fimLoop**
   Imprime o **resultado** na tela do usuário.

P4. FUNÇÃO COSSENO

Para realizar o cálculo do cosseno de um dado ângulo, este programa, no segmento .data, define um **x**, as constantes **pi** (3.141592), **divisor** (180.0), **precisão** (0.00001) e **um** (1.0), além de uma string **result_msg** para imprimir o resultado no final do programa. Além disso, foram estabelecidas as funções:

5) **main**
   Converte o ângulo **x** (em graus) para radianos, a partir do cálculo **x * pi / 180.0**, e chama a função **cosex**;
6) **cosex**
   Prepara o programa para o cálculo propriamente dito do cosseno de **x**, definindo **resultado** e **precisão** em seus respectivos registradores, além de inicializar o fator $\mathbf{x^2}$ e o **sinal** (alternância);
7) **loop**
   Faz o cálculo do cosseno de **x** a partir de várias iterações, que se encerram quando o termo atual for menor que a precisão;
8) **end_loop**
   Imprime o **resultado** na tela do usuário.

P5. QUICKSORT

Para a implementação do algoritmo do QuickSort em MIPS Assembly, foram declarados uma pilha **stack**, um vetor **vetor** e uma string vazia **espaco** para formatação do resultado. A partir disso, no segmento .text, os registradores são inicializados e o programa prepara saltos para as demais funções, sendo elas:

1) **loop_arrayj_maior_pivot**
   Encontra elementos maiores que o pivô na lista;
2) **primeiro_if**
   Compara dois elementos e realiza a troca se necessário;
3) **segundo_if**
   Verifica se o índice atual ultrapassou o índice superior, indicando que não há mais elementos à esquerda do pivô para processar;
4) **loop_i_menorigual_j**
   Controla a iteração sobre a lista enquanto o índice inferior não ultrapassa o índice superior. Ele chama outras funções para encontrar os elementos menores e maiores que o pivô;
5) **loop_arrayi_menor_pivot**
   Encontra elementos menores que o pivô na lista;
6) **quick_sort**
   Seleciona um elemento como pivô e rearranja os elementos de tal forma que todos os elementos menores que o pivô estejam à sua esquerda e todos os elementos maiores estejam à sua direita;
7) **terceiro_if**
   Verifica se o índice atual ultrapassou o índice superior, indicando que não há mais elementos à direita do pivô para processar;
8) **fim**
   Finaliza o programa;
9) **print_vetor**
   Imprime os elementos do vetor após a ordenação;
10) **return**
    Retorna para a instrução chamada.