MEMORIAL DESCRITIVO - Programando na Arquitetura MIPS

**Objetivo:** Desenvolver programas com diversos propósitos na linguagem de baixo nível: Assembly MIPS-32. Detalhando as etapas, desafios e raciocínios envolvidos nesse processo.

**Integrantes:**

Daniel Borges Gonçalves - 12311BCC005

Gustavo Costa Miranda – 12311BCC034

Luana Rodrigues Borges – 12311BCC028

Sophia Ladir Pereira Vieira de Morais – 12311BCC004

Raquel Emillene Freire Thomé - 12311BCC026

· Primeiro problema: Função isdigit(c)

**Entendendo o problema:** É necessário escrever um programa que verifica se um caractere passado é um dígito, ou seja, se é um caractere numérico que está entre '0' e '9' na tabela ASCII.

**Resolução do problema proposto:** Na solução foram utilizadas duas funções predeterminadas pelo enunciado, “getchar()” que faz a leitura do caractere e o retorna no registrador $v0 e a “print\_string()” que mostra na tela a string passada. Além dessas funções foi implementada a função “isDigit()”, que faz as comparações de acordo com os números de cada caractere na tabela ASCII, a fim de determinar se o caractere é um digito, ou seja, está entre os números 48 e 57, depois da verificação retornamos o registrador $v0 com o valor 0 se for um dígito e se não for saltamos para a etiqueta “notDigit” para retornar $v0 com o valor 1. Em seguida, carregamos a mensagem apropriada para os dois casos de retorno, se não for um dígito saltamos para a etiqueta “setStr2”, onde a string str2 ("O char digitado nao eh digito") é carregada em $a0, e se for um dígito a string str1 ("O char digitado eh digito") permanece carregada em $a0. Para finalizar o programa mostramos a string carregada em $a0 na tela usando os mecanismos de E/S.

· Segundo Problema: Converts the string to lowercase

**Entendendo o problema:** É pedido que seja feita uma função para converter os caracteres de uma string passada pelo usuário para lowercase, ou seja, colocar os caracteres na sua forma minúscula.

**Resolução do problema proposto:** Primeiramente, reservamos um espaço de memória de 100 caracteres para armazenar a string que será lida, como um vetor de nome str. Utilizamos a função print\_string() especificada no exercício anterior para pedir que o usuário digite a frase que será convertida. Implementamos a função getstr(), que realiza a leitura char por char da string passada e armazena no vetor reservado no início do programa, até encontrar um ‘\n’. Depois de lida e armazenada a string, salvamos str no registrador $a0 e chamamos a função tolower(), que itera sobre todos os caracteres do vetor str verificando se o caratere da iteração está entre ‘A’ e ‘Z’, se sim ele soma 32 ao seu valor, para que, de acordo com a tabela ASCII, eles sejam convertidos para minúsculos, mas caso não forem letras maiúsculas saltamos para a etiqueta que fará a iteração e que passará para o próximo caractere de str. Para finalizar o programa mostramos a string convertida na tela usando os mecanismos de E/S.

· Terceiro Problema: Procedimento atoi

**Entendendo o problema:** É proposto criar um programa que converte uma string de dígitos de ‘0’ a ’9’, para um número inteiro, resumidamente ASCII para int.

**Resolução do problema proposto:** Para essa solução, inicialmente reservamos o espaço de memória para a string que será lida e utilizamos as funções print\_string() e getstr() que foram implementadas nos exercícios anteriores. Após lida e armazenada a string, chamamos a função atoi(), que verifica se os caracteres são dígitos e se forem os converte para decimal e soma no registrador $v0, conforme o sistema decimal exige, por isso multiplicamos por 10 antes da soma. Para finalizar mostramos na tela que a string foi convertida com sucesso. Porém se a string contiver caracteres não numéricos, a função irá retornar -1 para indicar que um erro ocorreu.

· Quarto problema: Função floatValidate(string)

**Entendendo o problema:** Implementar um programa que verifica se a entrada do usuário é um número de ponto flutuante válido, usando as determinações do autômato passadas no enunciado.

**Resolução do problema proposto:** Primeiramente, reservamos o espaço de memória para a string que será lida e utilizamos as funções print\_string() e getstr() que foram implementadas nos exercícios anteriores.

Posteriormente, chamamos a função floatValidate(), que é dividida em estados, o registrador $a1 armazena em qual posição da string estamos durante a leitura e o $a2 armazena em qual estado o autômato está.

No estado0, saltamos para a função sinal(), que retorna no registrador $v0 o valor 0 se não for um sinal ou o valor 1 se for um sinal, ou seja se o primeiro caractere for 43(-) ou 45(+), na tabela ASCII, se for o retorno for 0 pulamos para o estado2 sem alterar o $a1, ou seja, na mesma posição da string, mas se for 1 nós incrementamos $a1 antes de ir para o estado2.

No estado2, chamamos a função isDigit() que foi implementada no primeiro exercício, e verificamos em loop até encontrar um caractere ‘.’, se encontrá-lo pulamos para o estado3, se não for nem um dígito e nem o ponto pulamos para o fim e imprimimos na tela a string falso (“Não é um float valido”).

No estado3, fazemos a leitura, após o ponto, em loop até encontrar um ‘\n’ ou os caracteres ‘e’ ou ‘E’ a fim de saltarmos para o estado5, mas se não for um dígito pulamos para o fim e imprimimos a string falso.

No estado5, é feita a verificação do sinal do expoente da potência, se não for um sinal pula para o estado7 e se for, adiciona 1 ao $a1 para depois pular para o estado7.

No estado7, se o caractere em questão for um ‘\n’ pulamos para o fim e imprimimos na tela a string falso, se não verificamos se é um digito em loop até finalizar a string. Para finalizar o programa se o registrador $v0 contiver o valor 0 então imprimimos a string falso e se contiver o valor 1 imprimimos a string afirma ("É um float valido").

· Quinto Problema: Função para o retângulo de interseção

**Entendendo o problema:** Elaborar um programa em linguagem MIPS Assembly que receba como entrada os vértices de dois retângulos no plano cartesiano e determinar se eles se interceptam ou não, caso haja interseção mostrar as coordenadas do retângulo de interseção resultante.

**Resolução do problema proposto:** Foram criadas duas strings para receberem as entradas do usuário, cada um com as coordenadas de dois pontos opostos de um retângulo; esta leitura é feita usando a função getstr. Posteriormente, cada coordenada é separada e interpretada como número - não mais como char – e armazenada no vetor de inteiros “vec”. A função check\_intersection primeiro carrega as coordenadas do primeiro retângulo, uma em cada registrador, faz a comparação entre elas, de mínimos e máximos, em seguida faz o mesmo para o retângulo 2. Em seguida, a verificação do retângulo de intersecção começa: a lógica é que para isso ocorrer é necessário que o menor X do retângulo 1 seja menor que o maior X do retângulo 2 e o maior X do retângulo 1 seja maior que o menor X do retângulo 2 o mesmo raciocínio se aplica para os Y, se essas verificações se concretizarem concluímos que há um retângulo formado pela sobreposição dos 2 anteriores. Caso não seja possível afirmar, a função mostra uma mensagem informando o erro. Ademais, a função ainda calcula as coordenadas dos pontos opostos do retângulo formado pela interseção usando da função de imprimir números.