Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Departamento de Engenharia de Computação e Automação

Segunda Tarefa
Programação *Assembly* para Arquitetura MIPS

Disciplina: DCA0104 – Arquitetura de Computadores
Turmas: 01 (35T34) e 02 (46M34), 2022.1
Professor: Diogo Pinheiro Fernandes Pedrosa

Para as duas questões enunciadas a seguir, elabore um programa em *assembly* para arquitetura MIPS de forma que possam resolver os problemas apresentados e, assim, também possam ser testados com os simuladores apresentados na turma virtual no SIGAA.

Questão 1 [50%]: A listagem a seguir consiste em uma sequência de referências feitas a blocos de uma memória principal em um computador hipotético. Considerando que, nesse computador, há uma cache de 16 linhas com mapeamento direto e que inicialmente está com conteúdo inválido. Após cada referência ser realizada, há a contagem de acertos ou falhas de cache. Quantos erros e quantos acertos têm-se no final?

Referências aos blocos: 2, 3, 11, 16, 21, 13, 64, 48, 19, 11, 3, 22, 4, 27, 6 e 11

Observações específicas para a questão 1:

- O código *assembly* para resolver esse problema faz manipulação, apenas, de registradores inteiros;
- A listagem de referências (número dos blocos) pode ser alocada em um *array* de *words* inteiras;
- O *array* de 16 posições que representará a cache pode ser alocada com valores iniciais negativos, representando dados inválidos, por exemplo. Não é necessário fazer um *array* multidimensional (para armazenar TAG e *word*, por exemplo);
- Nessa "cache", a identificação do número da linha a ser gravada pode ser facilmente obtida através da seguinte operação:
  número da linha = (número do bloco) resto da divisão *(16 linhas)*
- Uma dica importante para saber se o programa *assembly* está correto é resolver manualmente esse problema e, aí, comparar com a saída do programa; e
- A resposta desta questão deve ser o arquivo .ASM com o código comentado (explicando o que foi feito) e com a identificação (nome e matrícula).

Questão 2 [50%]: Em vários problemas de ciências e engenharia, um que se mostra bastante comum de ser tratado é o de se encontrar as raízes de uma equação. Este problema consiste em determinar a *raiz*, ou solução, de uma equação na forma $f(x) = 0$, para uma dada função f. Essa raiz é, normalmente, chamada de *zero* da função. Dependendo da função f, essa raiz pode ser encontrada por métodos analíticos. Porém, em diversas situações, é necessário aplicar métodos numéricos para tal objetivo.

Um dos métodos numéricos mais comumente empregados para determinar raízes de equações é conhecido como *método da bisseção*. Neste procedimento, considere que tem-se uma função $f(x)$ contínua e definida em um intervalo $[a, b]$, com $f(a)$ e $f(b)$ com sinais opostos (ou seja, $f(a) < 0$ e $f(b) > 0$ ou $f(a) > 0$ e $f(b) < 0$). Assim sendo, pelo teorema do valor intermediário, existe um número p no intervalo (a, b) de tal forma que $f(p) = 0$, ou seja, é raiz da função f.

De forma resumida, o método da bisseção, que é usado para encontrar uma raiz, consiste em reduzir gradativamente o intervalo $[a, b]$, até uma determinada tolerância, e testar o valor intermediário do

novo intervalo para saber se ele está próximo da raiz desejada. O algoritmo do método da bisseção é apresentado a seguir:

Entrada: pontos extremos *a* e *b;* tolerância *TOL*; número máximo de iterações *N*.
Saída: solução aproximada *p* ou mensagem de erro

Defina i = 1;

Defina FA = f(a);

Enquanto i ≤ N, faça:
    p = a + (b – a)/2;

    FP = f(p);

    Se FP = 0 ou (b – a)/2 < TOL, então:
        Exibir p
        Parar
    Fim-Se

    i = i + 1;

    Se FA · FP > 0, então:
        a = p;
        FA = FP;
    Senão:
        b = p;
    Fim-Se
Fim-Enquanto

Exibir saída: "Raiz não encontrada!"

Assim sendo, encontre a raiz positiva da equação $f(x) = x^3 - 10$ pelo método da bisseção, com *TOL = 0,1*, máximo de 10 iterações e com intervalo de busca igual a [*2.0*, *3.0*].

Observações específicas para a questão 2:

- Esse programa fará manipulação com dados em ponto flutuante. É suficiente trabalhar com ponto flutuante de precisão simples (ou seja, não precisa usar *double*);
- Embora seja um programa que envolverá operações com ponto flutuante, também será necessário lidar com instruções e registradores de valores inteiros;
- A resposta deve ser o arquivo .ASM com o código comentado (explicando o que foi feito) e com a identificação (nome e matrícula). Esse programa É DIFERENTE do programa da questão 1 (são arquivos diferentes);
- Percebam que não é necessário lidar com *arrays* ou estruturas similares. Mas será necessário trabalhar com chamadas do sistema;
- Uma dica é resolver manualmente este problema e comparar esse resultado com retornado pelo programa desenvolvido.
- Os valores iniciais podem ser passados diretamente pelo programa (ou seja, NÃO É NECESSÁRIO fazer a interface com usuário para ele digitar os valores do intervalo, tolerância nem o número máximo de iterações).

Observações Gerais:

- Este exercício é individual;
- A entrega será pelo SIGAA (Tarefas), até a data especificada;

- Perceba que são dois arquivos .ASM diferentes (um para cada problema). Gere um arquivo compactado (.ZIP) com os dois códigos *assembly* e o envie pelo SIGAA;
- Lembre-se que os códigos devem estar com comentários que expliquem o programa. Teste-os antes de enviar (se o programa não executar por erro de sintaxe ou de digitação de alguma instrução, isso será prejudicial para a questão);
- Quaisquer outras questões: diogo.pedrosa@ufrn.br

Referências

- *Cálculo Numérico (com aplicações)*; 2ª edição; Leônicas Conceição Barroso e outros; Editora Harbra; 1987.
- *Análise Numérica*; Richard L. Burden e J. Douglas Faires; Editora Thomson; 2003.
- *Organização e Projeto de Computadores – A Interface Hardware/Software*; 3ª edição; David A. Patterson e John L. Hennessy; Editora Campus; 2005.