USP/ ICMC/SSC
SSC0610 Organização de Computadores Digitais I
1º Trabalho – disponibilizado em 01º /09/2014

Implementação em Assembly MIPS do algoritmo para fazer percorrimentos em árvore binária ordenada

O objetivo deste trabalho é implementar o algoritmo do *Bucket Sort* em A*ssembly* MIPS para ordenar vetores de *strings* (uma lista de nomes). O *Bucket Sort* é um algoritmo de ordenação de vetores. Basicamente seu funcionamento consiste em dividir o vetor em partes (baldes) e então ordenar cada parte do vetor individualmente através de outro algoritmo de ordenação (ou do próprio algoritmo do *Bucket Sort* recursivamente). Por fim o vetor final é remontado trazendo como resultado os dados ordenados parcialmente. Ao remontar o vetor com o resultado de cada balde, o resultado final estará ordenado. No exemplo da Figura 1, com números inteiros entre 1 e 100, percebe-se que cada balde recebe apenas os valores que estão dentro da faixa esperada, i.e., 1-20, 21-40, 41-60, 61-80 e 81-100. Com isso cada ordenação parcial poderá ser concatenada para formar um vetor ordenado final novamente.

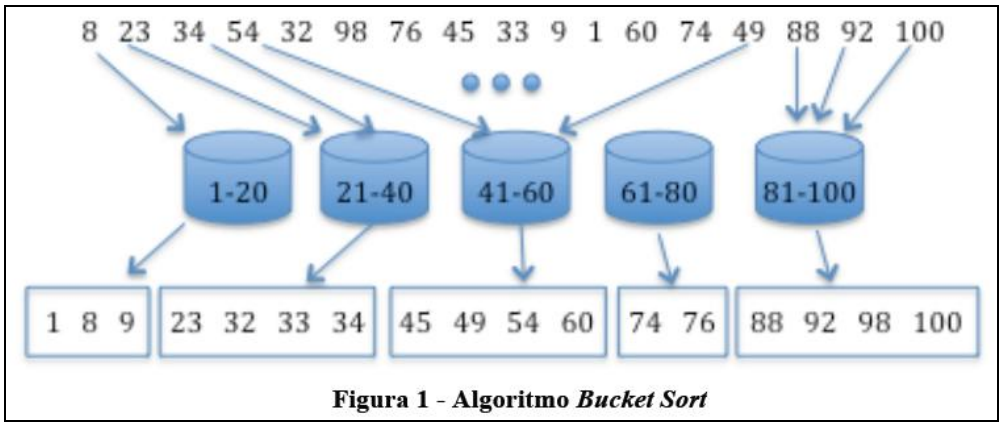

Figura 1 - Algoritmo *Bucket Sort*

Neste trabalho podem ser lidos até 100 nomes e cada nome pode ter até 25 caracteres. Utilize 26 baldes, um para cada letra do alfabeto. O método de ordenação a ser utilizado internamente ao balde é o *Bubble Sort*. O programa deve ler uma lista de nomes de um arquivo do tipo "txt" contendo um nome por linha. Exemplo:

João Silva
José Maria
Maria José

O nome do arquivo "txt" será digitado pelo usuário. O programa deve abrir e ler esse arquivo "txt", ordenar a lista de nomes e imprimir o resultado, com um nome por linha, em um arquivo chamado "saida.txt". No site da disciplina vocês encontram dois arquivos "txt" contendo linhas com nomes a serem usados como testes. Um dos arquivos não está ordenado e o outro está (contém os mesmos nomes do primeiro arquivo, porém ordenados alfabeticamente). Utilizem esses arquivos como primeiros casos de teste. No entanto, não se limitem a estes casos. Após fazer uma ordenação o programa deve retornar ao começo, solicitar um novo nome de arquivo e iniciar uma nova ordenação. O critério de parada é um nome de arquivo igual a ***getout***.

A correção levará em conta a execução correta do algoritmo e a qualidade do código fonte feito, conforme explicado nas aulas. Não altere o padrão de entrada e nem a saída dos dados na aplicação. Não imprima outras saídas na tela ou no arquivo, para não dificultar a correção. Caso haja uma entrada inválida, o algoritmo deve dar uma saída correta, em qualquer situação. No caso de haver essa entrada inválida, o algoritmo deve informar tal situação e voltar para a execução normal, sempre que possível. O aluno deverá procurar o professor responsável pela disciplina e/ou o aluno PAE se houver dúvida no desenvolvimento do trabalho ou se forem encontrados eventuais problemas/omissões nesta especificação.

O trabalho deve ser feito em grupo, o qual já foi determinado no início do semestre letivo. Envie apenas um arquivo por grupo contendo o código fonte. Este arquivo **deve** ter o nome: G***XX-nnnn*** (***XX*** indica o nr do grupo e ***nnnn*** indica o nome de um dos integrantes do grupo). Forneça, obrigatoriamente, como comentário no corpo do arquivo submetido o número do grupo e o nome de **todos** os integrantes do grupo que efetivamente participaram do desenvolvimento. Quem não estiver nesta lista terá nota zero.

A data de entrega do trabalho já foi determinada em sala de aula e está na página da disciplina no Moodle/STOA/USP. A correção considerará os simuladores PCSPIM ou QTSPIM. Utilize um deles para o desenvolvimento.

Siga rigorosamente toda a especificação deste texto a fim de validar a sua nota. Essa validação será feita por um fator de multiplicação à nota final do trabalho: 0 (zero) por não seguir a especificação ou 1 (um) por seguir.