Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática
6894 - Programação para Interfaceamento de Hardware e Software
Professor Ronaldo Augusto de Lara Gonçalves

# Segundo Trabalho: Relatório

Alunos:

| | |
|---|---|
| Chen Po Hsiang | 83473 |
| Rafael Cortez Sanches | 82357 |

## 1 Principais Módulos do Trabalho

Esse trabalho implementa uma lista ligada de registros em linguagem IA-32 GNU Assembly. Cada registro representa um funcionário, o qual possui os seguintes atributos:

- Nome (string de no máximo 44 bytes, um deles reservado para marcar fim da string)
- Data de nascimento (12 bytes no total, 3 inteiros de 4 bytes cada)
- Sexo (um caractére armazenado em espaço de 4 bytes para facilitar o alinhamento)
- Salário (um ponto flutuante de 4 bytes)

### 1.1 Arquivo Principal

No arquivo *listaligada.s* se encontram, além do rótulo global *_main*, diversas rotinas para manipulacao da lista de registros. As áreas de código dessas operações são referenciadas pelos rótulos:

- inserir
- remover
- buscar
- mostratudo
- alterar
- mudar_catalogacao

- alterar_salario

Elas são acessadas através da interace de menu, começando no rótulo "menuop".

Além delas, o arquivo principal contém a função "buscarreg", que recebe como parâmetro uma string por pilha, retornando no registrador %edi um ponteiro para o registro procurado, que pode ser NULL caso o registro não seja encontrado na lista. Essa função também passa um valor de retorno no registrador %esi, contendo um ponteiro para o elemento anterior ao procurado (ou NULL, caso o elemento procurado não exista ou seja o primeiro da lista). Esse segundo retorno é útil para a função de remover elementos, que precisa alterar o ponteiro *prox* do registro anterior ao removido.

A seguir, uma descrição breve de cada uma das opções do menu principal:

1. INSERIR REGISTRO: Pede ao usuário para que digite cada um dos campos de um novo registro. Ao final da enquete, o programa insere ordenadamente o novo registro na lista, de acordo com a diretiva atual de ordenação (ordem de nome ou ordem por data de nascimento).

2. REMOVER REGISTRO: Pede que o usuário informe o nome do registro que deve ser apagado, fazendo-o em seguida.

3. BUSCAR REGISTRO: Busca um registro na lista com o respectivo nome informado pelo usuário. Mostra todos os campos desse registro.

4. LISTAR TODOS: Mostra todos registros da lista

5. ALTERAR REGISTRO: Dado o nome de um registro, informado pelo usuário, é pedido para que o usuário digite novos valores para todos os campos desse registro, com excessão do nome.

6. ALTERAR ORDENACAO: Dá a opção para o usuário mudar o atributo usado na ordenação, deixando os registros na lista com uma disposição diferente. As opções de ordenação disponíveis são: ordenação por nome, ordenação por data de nascimento.

7. ALTERAR SALARIO: Pede ao usuário o nome de um registro, do qual o campo "salário"pretende-se alterar. São disponibilizadas diversas opções de alterações: por soma, subtração, aumento em taxa percentual, diminuição em taxa percentual, multiplicação e divisão.

8. SAIR: Termina a aplicação.

## 1.2 Comparador de Registros por Strings

No arquivo *comaparastring.s* há uma função de rótulo "comparastring". Ela recebe por parâmetro (empilhados) os ponteiros de dois registros, retornando em %edi os seguintes valores:

- 0 caso os nomes de ambos registros sejam iguais
- 1 caso o primeiro registro empilhado venha depois do segundo, em ordem alfabética de nomes
- 2 caso contrário

Essa função se utiliza de outra, fornecida pela biblioteca C, chamada "strcmp". A função comparadora de strings é utilizada para organizar os registros em ordem alfabética por nome de funcionário.

### 1.3 Comparador de Registros por Data

Muito semelhante à função "comparastring", no arquivo *comaparastring.s* há uma função chamada "comparadata". Essa recebe como parâmetros os ponteiros de dois registros por pilha, retornando em %edi os seguintes valores:

- 0 caso as datas de nascimento sejam iguais em ambos registros
- 1 caso o primeiro registro empilhado venha depois do segundo, em ordem de data de nascimento
- 2 caso contrário

O arquivo principal possui uma opção que permite alternar a forma como os registros são ordenados na lista, permitindo que eles sejam disponibilizados em ordem de data de nascimento. Essa função é usada para ordenar e para inserir novos elementos na lista quando o atributo de ordenação escolhido é a data de nascimento.

### 1.4 Função de Ordenação

No arquivo *sortallreg.s* está implementada a função "sortallreg", que aplica um algoritmo de ordenação por inserção na lista ligada. Os parâmetros passados por pilha são: ponteiro para a lista e um inteiro indicando o atributo que deve ser usado na ordenação: 0 para ordenação por nome, 1 para ordenação por data de nascimento.

Caso a raiz da lista seja mudada durante a reordenação, o ponteiro da raiz antiga, passado por pilha ao se chamar a função, é alterado para a nova raiz. Por esse motivo, no arquivo principal, a raiz deve ser alterada para o valor que está empilhado ao término da função.

## 2 Dificuldades Encontradas

A principal dificuldade no começo envolveu a manipulação de ponteiros para percorrer o registro. Algumas depurações foram necessárias para se perceber que o ponteiro *prox* estava sendo lido a partir de um local de memória inadequado, por exemplo.

Também foram encontrados problemas nas funções implementadas, devido ao empilhamento do endereço de retorno, executado pela instrução "call". Ao se desempilhar os parâmetros, esquecia-se de desempilhar e guardar o endereço de retorno previamente, gerando um erro no processamento da função. O problema foi resolvido através do desempilhamento do endereço de retorno, guardando-o em uma variável e o empilhando novamente antes de se chamar a instrução "ret".

Novamente problemas com ponteiros apareceram ao se construir a função de ordenação. O *insertion sort* implementado requeria que uma subrotina *swap* fosse implementada, para trocar o maior elemento encontrado com o elemento da primeira posição do subproblema (o problema de ordenação vai diminuindo conforme os elementos corretos são inseridos nas primeiras posições). Essa subrotina apresentou diversos problemas de funcionamento, até finalmente ser corrigida.

# 3 Condições de Funcionamento

O programa foi testado em ambientes Linux Ubuntu, seu funcionamento se dá sem falha aparente. Para montar a aplicação, disponibilizou-se um *makefile* e instruções de execução no arquivo *readme.txt*.

# 4 Convenções e Limitações

Na inserção de novos registros, a data de nascimento é pedida na seguinte ordem: ano, mês, dia. Essa ordem de escrita facilita a leitura do registro na rotina "le_dados". Os nomes devem ter no máximo 43 caractéres, e a profissão tem limite de 23 caractéres.

Ao se alterar a data de nascimento de um registro, caso se esteja ordenando os registros por ordem de data de nascimento, o registro alterado pode ficar em uma posição incorreta dentro da lista. Para corrigir isso, pode-se chamar a opção *ALTERAR ORDENACAO* no menu principal e selecionar novamente a ordenação por data de nascimento. Esse procedimento reordenará todos os elementos da lista por data de nascimento, resolvendo o problema.