Universidade Estadual de Maringá
Departamento de Informática
Curso: Informática
Disciplina: 5193 - Programação em Linguagem de Montagem
Professor: Ronaldo Augusto de Lara Gonçalves

Simulador de elevador em linguagem GNU Assembly para plataforma 32 bits

Equipe:

Gabriel de Oliveira Conejo	RA105483
Maicon Henrique Rodrigues do Nascimento	RA102504
Raphael Franco de Lima	RA98336

Maringá
2019

Sumário

Índice de ilustrações

1. Resumo

Este trabalho teve como objetivo desenvolver um software em linguagem GNU Assembly utilizando um conjunto de instruções para sistemas com arquitetura de 32 bits que simulasse um elevador com algumas funcionalidades reais.

O programa funciona da seguinte forma: Ao executá-lo será pedido o número de andares do prédio e uma probabilidade de acontecerem chamadas externas a cada iteração do elevador. Existe um loop infinito e a cada iteração do loop o elevador processa/executa suas determinadas chamadas externas e internas e escreve na tela as informações dos processamentos. Para o elevador prosseguir com sua execução é necessário apertar a tecla “ENTER” após cada processamento. Isso permite analisar as informações escritas na tela.

2. Divisão de tarefas

Foi descrito um pseudocódigo, sendo este dividido em 3 partes (processamento dentro de cada iteração do loop) e cada aluno se dispôs a fazer qual gostaria.

2.1. RA98336

Responsável pela escrita do pseudocódigo e sua divisão. Além disso, elaborou a rotina contida no rótulo de entrada main para leitura das variáveis e validação de entrada.

Responsável pela parte da verificação das listas (interna e externa) para verificar se existe alguma entrada ou saída de pessoas do elevador. E realizada das chamadas internas após a entrada no elevador.

Ao final das verificações mostra a quantidade de pessoas que estão saindo ou entrando no elevador.

2.2. RA102504

Responsável pela parte do processamento de chamadas externas. A cada iteração do loop o software deveria realizar o sorteio de 1 ou 2 andares e para cada andar do prédio realizar o sorteio de 1 a 3 pessoas que fizeram uma chamada externa para o elevador neste determinado andar.

Para cada andar sorteado (sorteio de 1 ou 2 andares) o programa deveria definir se as chamadas externas deveriam ocorrer ou não, baseada na probabilidade dada no início da execução do programa, serem processadas. Caso fossem processadas a posição da lista externa que representa o número de chamadas no andar sorteado deveria ser incrementada no número de pessoas sorteadas (1 a 3 pessoas) que fizeram uma chamada externa.

2.3. RA105483

Responsável pela parte de movimentação do elevador. A cada iteração do loop o programa deve realizar a chamada das funções

que verifica se existe chamadas para o elevador no sentido em que ele está indo e caso houver este se movimenta em uma unidade.

Também é verificado se o elevador já se encontra no último andar ou no primeiro andar, desta forma, caso o elevador se encontre nas extremidades, é trocado a direção do elevador e realizado a verificação novamente.

Caso não exista mais chamadas para o elevador no sentido em que ele está indo, é trocado a direção.

3. Funcionamento do código

Para explicar o funcionamento do programa pode ser utilizado o pseudocódigo desenvolvido.

```
main:
 leitura das variáveis e validações de entrada

 loop_infinito:
  verifica lista interna na posição do andar atual para ver se alguém sai
  IMPRIME X pessoas saindo do elevador
  verifica lista externa na posição do andar para ver se alguém entra
  IMPRIME X pessoas entrando do elevador
  realiza chamadas internas caso alguém entre no elevador

  IMPRIME total de pessoas no elevador

  realiza sorteio de um ou dois andares
   para cada sorteio calcula se ele ocorre ou não
    para cada andar realiza sorteio de 1 a 3 pessoas caso o sorteio
    anterior tenha acontecido

  verifica lista externa e interna e verificar se existe alguma chamada
  em andares que se encontram na direção em que o elevador está indo
   se tiver e se a direção for subindo
    andar_atual++
   se tiver e se a direção for descendo
    andar_atual--
```

```
    se não tiver
      se posição não for um limite (térreo ou terraço)
        inverte direção do elevador
      se não
        continua na direção

  IMPRIME subindo ou descendo
  DELAY (aguarda o usuário apertar a tecla Enter para executar a próxima iteração)
```

Primeiramente, para o desenvolvimento do código são necessárias algumas variáveis para auxiliar o controle geral. As mais importantes para o funcionamento são apresentadas abaixo:

```
.section .data
qtd_andares:              .int 0
qtd_pessoas_elevador:  .int 0
direcao:                  .int 0 # SUBINDO (0) ou DESCENDO (1)
andar_atual:              .int 0

.section .bss

.lcomm lista_externa, 200
.lcomm lista_interna, 200
```

qtd_andares: representa a quantidade de andares que o elevador percorre, tal valor é informado pelo usuário.

qtd_pessoas_elevador: representa quantas pessoas estão dentro do elevador naquele momento.

direcao: representa para qual o elevador está se deslocando no momento. Sendo 0 subindo e 1 descendo.

andar_atual: representa em qual andar o elevador se encontra no momento. O andar 0 representa o térreo.

lista_externa: lista responsável por armazenar quantas chamadas externas existem em determinado andar.

lista_interna: lista responsável por armazenar quantas chamadas internas existem em determinado andar.

Dentro das interações do loop infinito que encontramos o funcionamento do sistema em geral, primeiramente é verificado na lista interna, se existe saída de pessoas do elevador, caso tenha, é imprimido a quantidade de pessoas que estão saindo do elevador. Logo após é verificado a lista interna para verificar se existe alguém entrando no elevador, se existir, é mostrado em tela a quantidade de pessoas entrando no elevador (importante realizar as chamadas nesta ordem). Após a entrada, cada pessoa gera uma chamada interna, a qual é impressa na tela.

Após a execução dessas operações, é mostrado a quantidade de pessoas total no elevador.

Feito isso, é realizado o sorteio de um ou dois andares e também o sorteio de uma a três pessoa. Para cada sorteio, é calculado a probabilidade de ocorrência do mesmo, caso o sorteio ocorra, o andar é sorteado e inserido na lista externa.

Logo após, é realizado a verificação se possui alguma chamada na lista externa ou na lista interna para andares que se encontram na direção em que o elevador está indo.

Caso exista alguma chamada em alguma das duas listas, o elevador se movimenta na respectiva direção. Se a direção do elevador for subindo, o andar do elevador é incrementado, se o elevador estiver descendo, o andar do elevador é decrementado, também é verificado se é possível incrementar ou decrementar a posição do elevador para que ele não passe das extremidades de andares. Neste último caso a direção do elevador é invertida.

Após a execução de todos os passos, é pedido para que o usuário aperte a tecla “enter” para continuar com a execução do

sistema, isto é necessário para que seja possível verificar o que está ocorrendo no sistema e o sistema volta para o início do loop.

Para o desenvolvimento do programa foram utilizadas diversas chamadas a funções, devido a isso foi adotada como prática de desenvolvimento que toda linha de código deveria ser comentada afim de facilitar o entendimento do código por todos os membros da equipe.

Além disso, foi utilizada a convenção de chamada x86 para funções. Sendo ela caracterizada pela seguinte estrutura:

```
rotulo_da_funcao:
 pushl %ebp
 movl %esp, %ebp
 .
 .
 # corpo do método
 .
 .
 popl %ebp
 ret
```

Tal código utiliza o registrado %ebp como um referencial para identificar a posição dos parâmetros passados para a função através da pilha.

Algumas das funções mais importantes para a execução do sistema são descritas abaixo.

incrementa_andar_na_lista: recebe como parâmetro uma das listas e um andar. Caminha na lista até aquele andar e incrementa o valor contido nele em uma unidade, representando assim uma nova chamada.

verifica_lista_externa: verifica se no andar atual existe alguém querendo entrar, caso haja, incrementa o número de pessoas

no elevador e realiza um loop (baseado no número de pessoas que entraram) de chamadas internas.

verifica_lista_interna: verifica se alguém quer sair do elevador naquele andar, caso haja, atualiza a quantidade de pessoas no elevador e zera as chamadas internas naquele andar.

existem_chamadas_direcao: recebe uma lista por parâmetro e dado o andar atual verifica se existem chamadas naquela direção. Retorna, em `%eax`, 0 caso não haja chamadas e 1 caso haja.

Abaixo seguem algumas imagens do programa em execução.

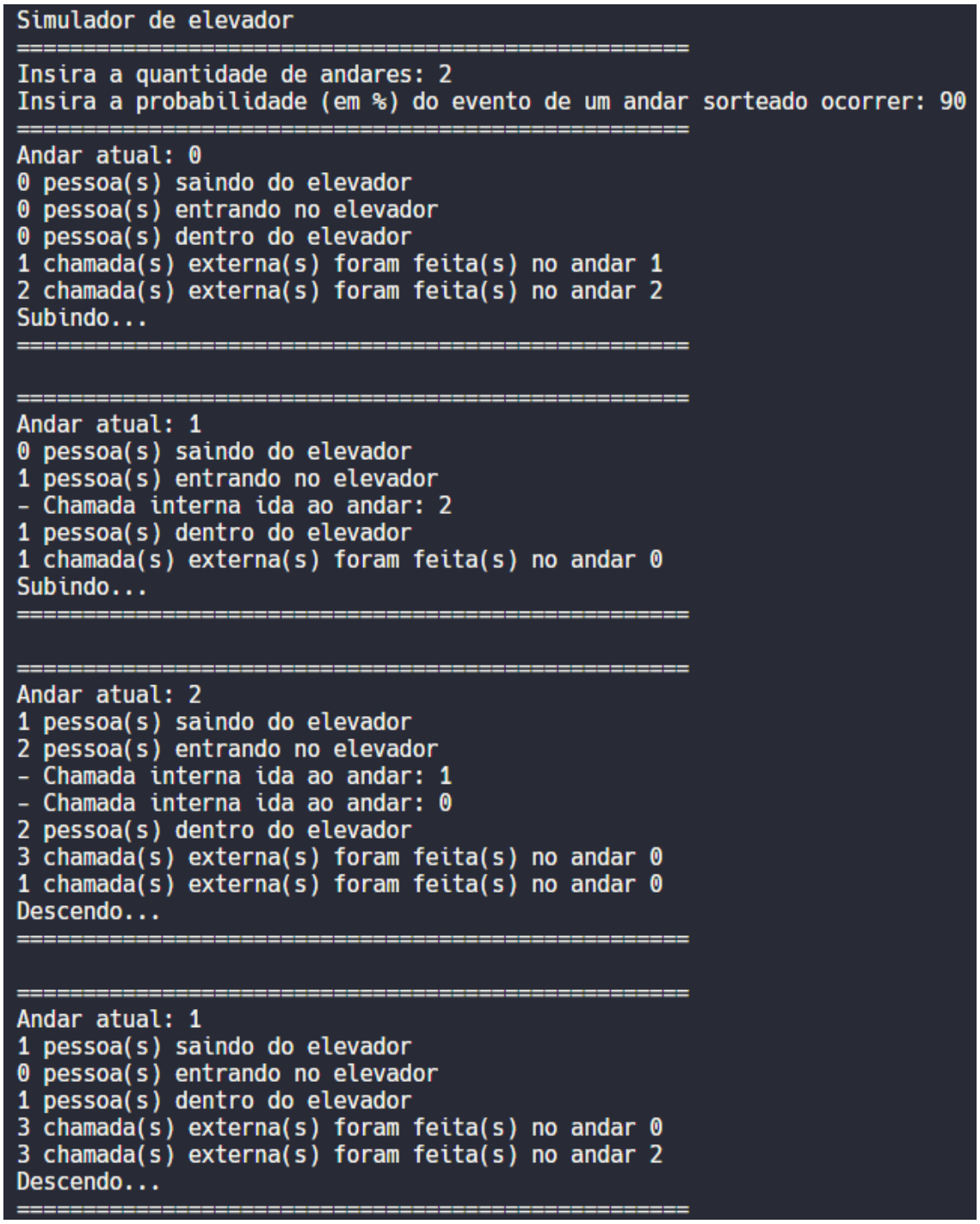

Figura 1 Programa em execução

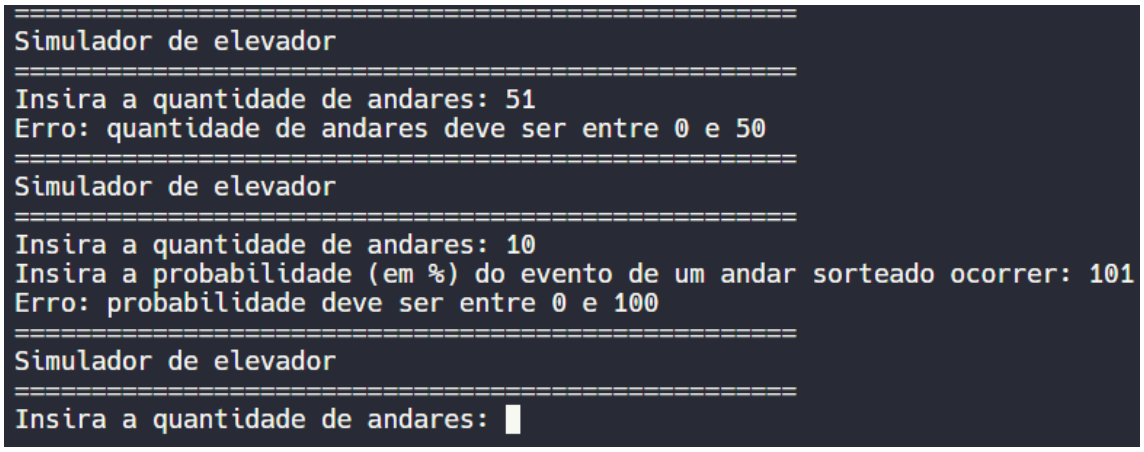

Figura 2 Validação da entrada de dados

Para execução do programa foi utilizado o compilador `gcc`, sendo que a sequência de comandos a serem realizadas a seguinte:

- `gcc -c -m32 elevador.s -o elevador.o`
- `gcc -m32 elevador.o -o elevador`
- `./elevador`

Com o intuito de simplificar a execução foi utilizada a ferramenta de automação `make` a qual possibilita reunir todos os comandos acima em apenas um. Sendo assim, para executar o programa, alternativamente pode-se utilizar o seguinte comando (desde que o arquivo `Makefile` esteja presente e o programa `make` instalado):

- `make run`

4. Instruções diferentes utilizadas

Não houve o uso de instruções fora do conjunto apresentado em sala de aula.

5. Restrições, limitações e problemas

O programa desenvolvido não faz o uso de dois recursos: alocação dinâmica e números reais. Devido a isso existem algumas limitações a serem consideradas. Primeiramente, o número de andares que o elevador pode percorrer é definido como

no máximo 50. Já a probabilidade de um sorteio ocorrer em certa iteração deve ser um número natural entre 0 e 100.

6. Bibliografia

University of Virginia. (25 de Novembro de 2019). *x86 Assembly Guide*. Fonte: Site da University of Virginia: https://www.cs.virginia.edu/~evans/cs216/guides/x86.html