

Trabalho Prático 1

SSC0902 - Organização e Arquitetura de Computadores

Integrantes:

Murilo Leandro Garcia (15480943)

Vitor Daniel Resende (15554396)

Renan Banci Catarin (14658181)

Glauco Fleury Corrêa de Moraes (15456302)

1 Estrutura e Implementação

O projeto foi implementado utilizando código em Assembly RISC-V, e testado em um simulador dessa ISA (Rars). Há diversos comentários no programa que buscam facilitar o entendimento das linhas de código. Ele é dividido em 2 files: 'util.asm' e 'main.asm'.

1.1 util.asm

Contém código auxiliar para a implementação da calculadora, na forma de definições úteis (exemplo: igualando falso a 0 e verdadeiro a 1 com a diretiva '.eqv') e de muitas macros. As definições auxiliam na lógica de programação e legibilidade do código, enquanto as macros permitem uma implementação mais enxuta e compreensível.

1.2 main.asm

Possui a lógica da calculadora em si. Pode ser dividido em funções auxiliares e funções principais, sendo que para cada uma há, em diversos pontos, labels criadas para cada erro possível durante o funcionamento.

Auxiliares:

- `cria_node`: cria um nó da lista encadeada
- `strncpy`: copia N caracteres de uma string em um determinado endereço de memória
- `atoi`: transforma uma string de dígitos no número correspondente
- `rstrip`: retira os whitespaces de uma string dada
- `eh_operador` & `eh_digito`: verificam se o caracter dado é, respectivamente, um operador ou um dígito.

Principais:

- `parse_input`: dada uma string de entrada, retira seus whitespaces, e verifica se ela possui uma estrutura passível de análise pelo programa. No caso, há apenas 3 tipos de inputs passíveis de interpretação: "número, não-número, número" OU "não-número" OU "não-número, número"; qualquer outro tipo de entrada resultará em um erro, acarretando uma `exit`.
- `calcula`: recebe o output do parsing e efetua a operação determinada pelo usuário, incrementando ou reduzindo a linked list conforme necessário. É aqui que erros de lógica do input em si serão tratados, como, por exemplo, o uso de um operador inexistente.

- main: responsável pelo loop principal da calculadora (receber input → avaliá-lo → (se não deu erro ou foi requisitado o fim) repetir)

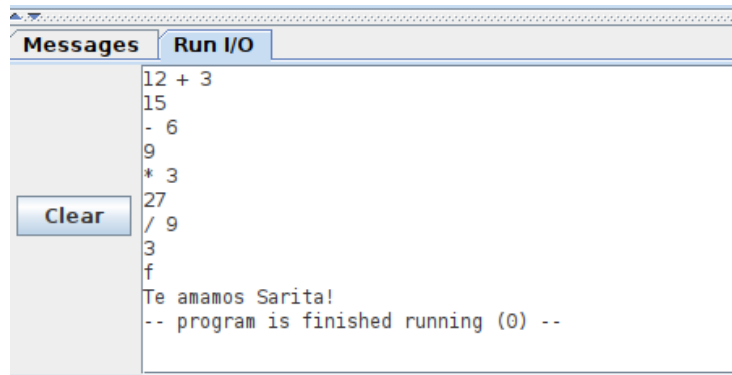
O programa inicia com uma flag sendo setada para TRUE, indicando que a string lida a seguir será a primeira. Após o 1º input, essa flag torna-se permanentemente FALSE. Repete-se um ciclo em 'main_loop': a entrada do usuário é lida e armazenada como string, essa string passa pelo processo de parsing, e o comando passado é então interpretado e executado, resultando em outputs na saída padrão. Nós são criados ou removidos da stack de linked lists dependendo do que o usuário desejar. Esse loop só pode ser quebrado pelo usuário se ele digitar um input inválido ou requisitar o término do programa com 'f'.

A linked list foi implementada da seguinte forma: cada nó contém um valor (4 bytes) e um ponteiro para o próximo nó (4 bytes), justapostos na memória. Para escolher qual deles acessar, basta ajustar o byte offset do registrador com o ponteiro para o nó atual. A linked list foi usada para implementar uma stack: há push quando um input altera o resultador armazenado no topo, e pop, caso 'u' seja usado. Por padrão, decidiu-se armazenar no primeiro nó o operando1 da entrada inicial "operando1, operador, operando2".

Optou-se pelo uso das funções 'parse_input' e 'calcula' devido à necessidade de tornar mais fácil o tratamento de erro durante o programa. Quando tentamos implementar ambas as funções em uma só, o código se tornou tão extenso e complexo que se fez necessário dividi-lo.

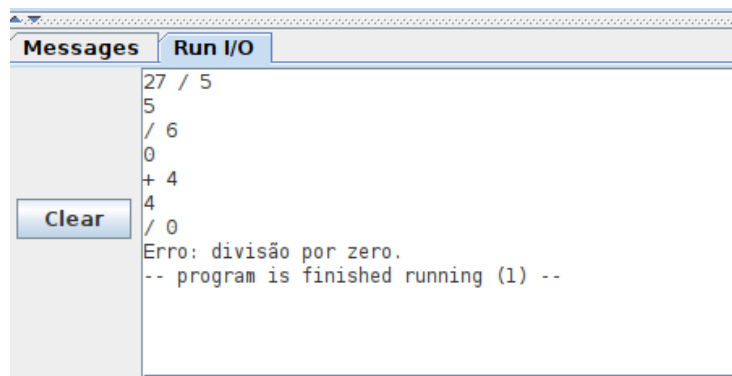
2 Imagens da execução do código

Seguem imagens que exemplificam o funcionamento do programa. A execução dele se deu no simulador Rars.



```
Messages Run I/O
12 + 3
15
- 6
9
* 3
27
/ 9
3
f
Te amamos Sarita!
-- program is finished running (0) --
```

Figure 1: Exemplo de uso padrão adequado da calculadora.



```
Messages Run I/O
27 / 5
5
/ 6
0
+ 4
4
/ 0
Erro: divisão por zero.
-- program is finished running (1) --
```

Figure 2: Exemplo de tratamento de erro: divisão por 0.

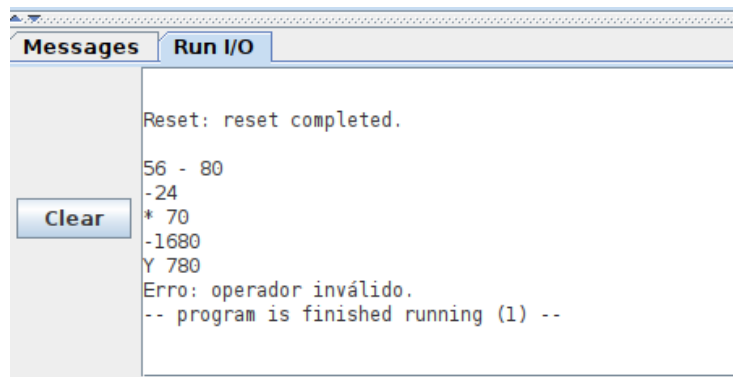


Figure 3: Exemplo de tratamento de erro: operador inadequado, detectado durante a etapa de análise da operação requisitada pela função 'calcula'.

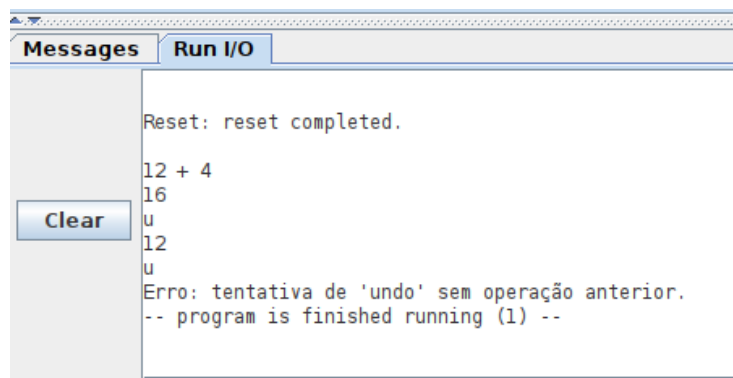


Figure 4: Exemplo de tratamento de erro: o usuário utiliza o comando 'undo', que funciona adequadamente enquanto a linked list ainda possui nós.