

Gerador de código para calculadora em Assembly

Júlia Roberta Quoos Alves, Luana Louzada de Aguiar

Departamento de Ciências da Computação - Universidade de Santa Cruz do Sul (UNISC)

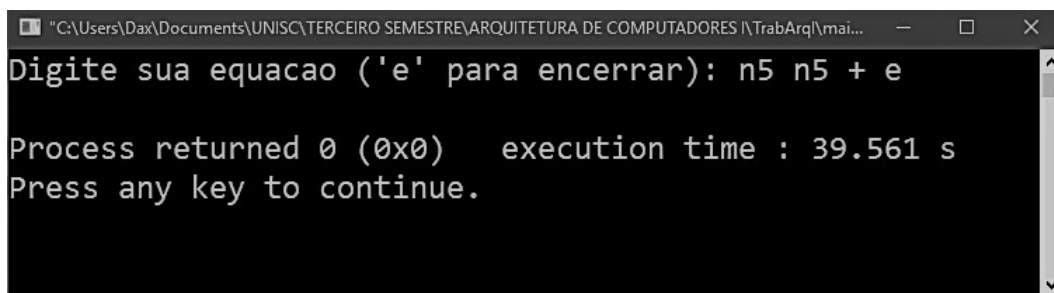
1. Introdução

O presente relatório é elaborado no âmbito da disciplina de Arquitetura de Computadores I, que visa a implementação de uma interface em linguagem de alto nível para linguagem Assembly como trabalho final da disciplina. O trabalho desenvolveu-se nas primeiras semanas de junho e conta como tema “Compilador de uma calculadora de números inteiros”. Neste período procuramos atingir alguns objetivos, tais como: entender e implementar funções recursivas, uso de pilha e fila e seus processos de armazenagem de dados no MIPS e criação de interface da linguagem C++ para linguagem de montagem.

Este relatório está dividido em 3 partes: Na primeira parte é feita uma breve introdução aos métodos utilizados, como escolha da linguagem e funções. A segunda parte tem como finalidade apresentar testes e validações do código (com imagens ilustrando exemplos). A parte final contém as conclusões e considerações finais sobre o trabalho, tais como nossas dificuldades e acertos quanto a implementação, referências e outros recursos utilizados.

2. Desenvolvimento e métodos utilizados

Nós utilizamos um único símbolo para cada operação, sendo os símbolos +, -, *, /, r, ^, ! e f, para adição, subtração, multiplicação, divisão, raiz quadrada, potenciação, fatorial e Fibonacci, respectivamente. Além disso, cada número e caractere deve ser digitado com um espaço, com exceção dos números negativos, que são lidos como na figura 1. O caractere *e* encerra o programa. Optamos por usar o caractere ‘n’ para identificar quando um número for negativo tendo em vista que não poderíamos utilizar o ‘-’, que já estava sendo utilizado para indicar a operação de subtração.



```
"C:\Users\Dax\Documents\UNISC\TERCEIRO SEMESTRE\ARQUITETURA DE COMPUTADORES I\TrabArq\mai...
Digite sua equacao ('e' para encerrar): n5 n5 + e
Process returned 0 (0x0)   execution time : 39.561 s
Press any key to continue.
```

Figura 1. Exemplo com leitura de operandos negativos para uma adição com dois operandos.

Na linguagem C++, utilizamos um comando de repetição para ler um vetor de char até que o caractere ‘e’ seja inserido, dentro desta função colocamos duas condicionais para testar se o número lido é operador (caso esteja na condição de teste realizada) ou operando (caso o caractere lido não seja nenhum dos símbolos reservados). Caso seja operador, é colocado na lista de operações para uso posterior. Caso seja operando, testamos se o número é

negativo e tratamos o caso usando uma repetição para deslocar todos os valores lidos uma posição à frente no vetor e transformamos o restante em um número inteiro com a função ‘atoi’ em seguida multiplicando o número gerado por -1 e colocando no vetor de operandos.

Após toda a equação ter sido digitada, começamos a colocar os dados no arquivo declarando o vetor utilizado e a mensagem de erro caso alguma operação não retorne um número inteiro, a lista dos operandos é percorrida, colocando no arquivo gerado um desvio para a função designada. Ao término da leitura, é colocado um salto para a impressão de resultados, representada pela função “exit” e em seguida são escritas todas as operações no arquivo.

Nosso programa aceita até dez operandos e dez operadores, podendo ser modificado no arquivo .cpp para valores maiores, sem limite específico. Este valor foi escolhido tendo em vista apenas o uso acadêmico, para validar as funções e testes de pequeno porte.

3. Testes e análises de resultado

Foram testadas todas as funções utilizadas em conjunto, sendo lidos números inteiros em notação polonesa reversa e exibindo um resultado também inteiro. Tratamos algumas funções que poderiam dar números imaginários e ou fracionários identificando a situação e deslocando para o label error, que exibe a mensagem da figura 2. O único caso não tratado foi o da divisão, que arredonda o número para menos.

```
Não foi possível calcular
-- program is finished running (dropped off bottom) --
```

Figura 2. Mensagem de erro.

Na figura 3 é ilustrada a leitura dos dados com o programa em C++ e uma parte do código gerado pela equação. Na parte superior podemos ver o vetor num declarado com os valores imediatos recebidos, logo acima do main com os respectivos saltos para as funções.

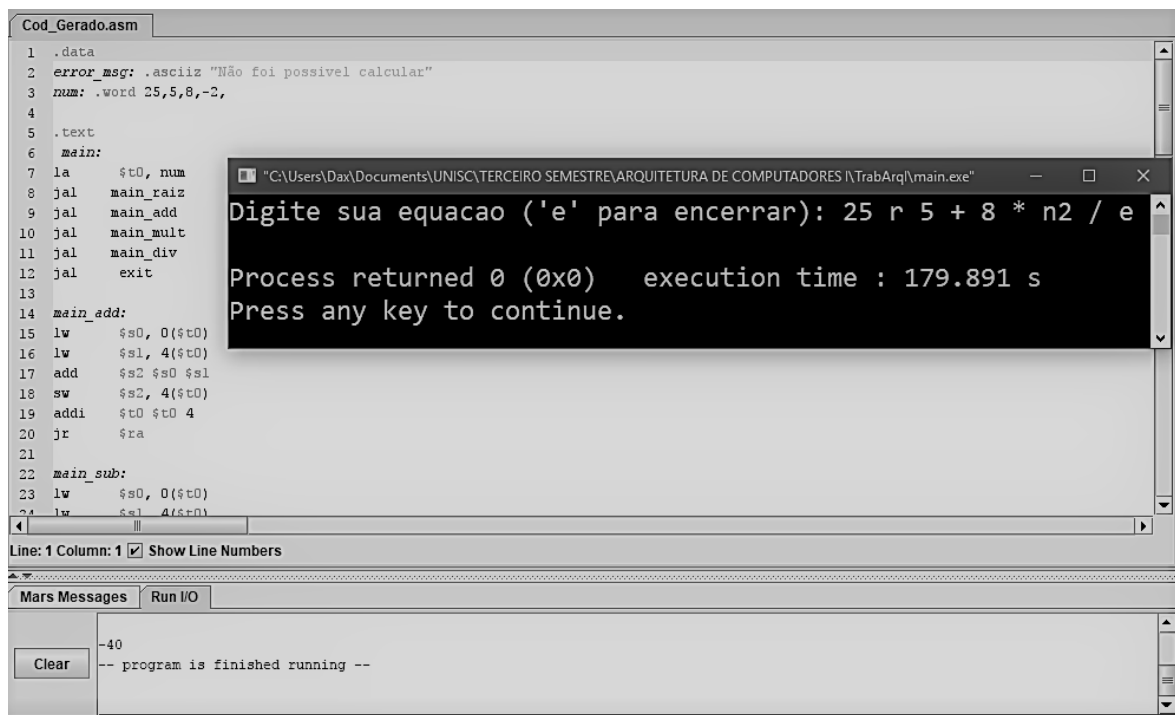


Figura 3. Leitura da operação $(\sqrt{5} + 5) * 8) / (-2)$, com visão do vetor gerado, do código comum a todos os programas e resultado na parte inferior da tela.

4. Considerações finais

Ao longo do trabalho identificamos que há pouco material disponível sobre Assembly em português na internet, quase não havendo sites com explicação detalhada sobre cada operação e como manipular as informações na linguagem. As operações de lista que fizemos foi com base em exemplos de manipulação de vetores, mas com algumas simplificações: decidimos não usar o stack pointer e apenas incrementamos o endereço do elemento inicial do vetor quando são utilizados 2 valores da lista. Caso contrário o primeiro elemento é substituído pelo resultado da operação. Sendo assim, consideramos as pesquisas feitas na internet fundamentais para a conclusão e desenvolvimento do trabalho, tendo em vista que não achamos livros específicos sobre programação em linguagem de máquina.

Referências

- Hennessy, John L.; Patterson, David A. (2000) "Organização e projeto de computadores: a interface hardware/software.", p. 53-70.
- Moura, Rudá. (2018) "RUDA #33 - Sobre a Notação Polonesa Inversa ou Reverse Polish Notation (RPN)", <https://www.youtube.com/watch?v=V5SO9V7RzJA>.
- Santana, Richard C. (2012) "Calculadora condicional (Arquitetura Mips)", <https://gist.github.com/RichardCSantana-zz/4025650>