UNIVERSIDADE DE SÃO PAULO

BEATRIZ ALVES DOS SANTOS
JHONATAN BARBOZA DA SILVA

RELATÓRIO
TRABALHO 1 - ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

SÃO CARLOS, SP
13 de abril de 2025

OBJETIVO

O objetivo deste relatório é apresentar e discutir a implementação do primeiro trabalho da disciplina de Organização e Arquitetura de Computadores: implementar uma calculadora sequencial em Assembly RISC-V. Para isso, será abordada a nossa linha de raciocínio juntamente com as dificuldades e decisões tomadas no decorrer do desenvolvimento do projeto.

PROBLEMA

Neste trabalho, é requerido a elaboração de uma calculadora sequencial capaz de realizar quatro operações básicas: soma (+), subtração (-), multiplicação (*) e divisão por inteiro (/) e mais duas operações especiais: finalizar o programa (f) e desfazer a última operação (u). Para isso, a primeira entrada fornecida pelo usuário deve consistir de um número, um caractere de operação e mais um número. A partir disso, é calculado o primeiro resultado e ele passa a ser o primeiro operando da operação seguinte, logo, o usuário passa a fornecer apenas o operador e, se necessário, o segundo operando das operações seguintes.

IMPLEMENTAÇÃO

- **Ideia Geral**

Para implementar o que foi proposto, decidimos usar um tipo de lista encadeada a pilha encadeada para armazenar os resultados das operações. Assim, cada elemento (nó) da pilha corresponde ao resultado de uma operação, sendo o elemento do topo da pilha correspondente à última operação. Durante a execução do programa, um novo nó é inserido na pilha sempre que um resultado válido é obtido. Nesse momento, os parâmetros da pilha (como o ponteiro para o topo e o valor do nó) são atualizados. Dessa forma, é possível armazenar todos os resultados e voltar a eles como ponto de partida, se necessário.

- **Aplicação**

Para representar a pilha, utilizamos o registrador s10 como ponteiro para o nó do topo e o s0 para guardar o valor do topo. A função principal do programa (main) é responsável por inicializar a lista encadeada, ou seja, atribuir zero (nulo) ao registrador s10. Além disso, a main se encarrega de fazer a leitura do primeiro número da primeira operação e depois iniciar o loop principal responsável por fazer a leitura do operador e redirecionar o código para diferentes labels a depender da operação desejada.

Para cada operação matemática implementada, o código segue uma sequência lógica bem definida. Primeiro, ele lê o segundo operando e verifica possíveis erros.

Em seguida, executa o cálculo adequado e armazena o resultado no registrador s0. Com o valor calculado, o fluxo do programa é direcionado para a rotina 'insere_no', que cuida da gestão dinâmica da memória.

Esta rotina aloca um novo nó da pilha com 8 bytes: os primeiros 4 bytes armazenam o valor do resultado, enquanto os 4 bytes seguintes guardam o ponteiro para o elemento anterior. Além disso, ela atualiza os registradores (s0 e s10) que mantêm o topo atual da pilha. Finalizado esse processo, o controle é transferido para a seção 'resultado', que se encarrega de exibir o valor obtido na tela antes de retornar ao loop principal do programa, pronto para processar a próxima operação

Na operação especial 'undo' (u), quando possível (ver Tratamento de Erros), o sistema atualiza os registradores s10 (com o ponteiro do nó anterior) e s0 (com o valor desse nó). Assim, o elemento do topo é atualizado com o anterior e a última operação é desfeita. Após isso, é escrito na tela uma mensagem indicando o resultado anterior e o programa volta para o loop principal. Por fim, quando a operação especial de finalizar (f) é detectada, é realizada uma chamada ao sistema para encerrar o programa.

- **Tratamento de Erros**

Durante o desenvolvimento do projeto, se fez necessário atentarmos para alguns cenários que poderiam acarretar na quebra do programa caso não fossem tratados da forma correta.

O primeiro é quando o caractere de operação dado pelo usuário não é válido, pois, nesse caso, o programa não seria direcionado para nenhuma operação e o fluxo do código não aconteceria como previsto. Para resolver esse problema, quando o operador não é identificado, é exibida uma mensagem de erro ao usuário e o programa volta para o loop principal.

O segundo é quando há uma tentativa de divisão por zero, o que não apresenta um resultado matemático definido e, consequentemente, resultaria em erro na execução. Para isso, foi adicionado uma verificação logo após a leitura do segundo elemento na operação de divisão, fazendo com que, caso seja zero, seja exibida uma mensagem de erro e a operação seja cancelada, voltando para o loop principal.

O terceiro é quando a operação undo é solicitada mas não é possível realizá-la por não ter resultados anteriores. Para resolver isso, é necessário verificar duas coisas: primeiro, se a pilha não está vazia e, depois, se o resultado anterior ao topo não é nulo. Se algum desses casos for verdadeiro, uma mensagem de erro é escrita na tela e o código volta para o início (main).

- **Execução**

A seguir, é mostrado o programa em execução com exemplos de utilização de todos os operadores e das mensagens de tratamento de erro.

Operação de soma, subtração, multiplicação e divisão:

```
Digite o número: 10
Digite o operador: +
Digite o número: 8
O resultado é: 18
Digite o operador: -
Digite o número: 5
O resultado é: 13
Digite o operador: *
Digite o número: 2
O resultado é: 26
Digite o operador: /
Digite o número: 5
O resultado é: 5
Digite o operador:
```

Operação undo:

```
O resultado é: 5
Digite o operador: u
O resultado anterior é: 26
Digite o operador: u
O resultado anterior é: 13
Digite o operador: u
O resultado anterior é: 18
Digite o operador: u

 *** Não há resultado anterior ***

Digite o número:
```

Operação de finalização:

```
Digite o número: 1
Digite o operador: f

-- program is finished running (0) --
```

Divisão por zero:

```
Digite o número: 5
Digite o operador: /
Digite o número: 0

 *** Não é possível dividir por zero ***

Digite o operador:
```

Operador inválido:

```
Digite o número: 5
Digite o operador: a

 *** Operador inválido ***

Digite o operador:
```

CONCLUSÃO

O projeto foi implementado com sucesso conforme as especificações, utilizando uma estrutura de lista encadeada com alocação dinâmica e manipulação eficiente de registradores. Os resultados obtidos demonstraram pleno funcionamento em todas as operações previstas, atendendo integralmente aos requisitos funcionais.

O sistema incorporou um robusto tratamento de erros, notificando os usuários através de mensagens claras e específicas sempre que situações excepcionais foram detectadas. Essa abordagem preventiva garantiu a estabilidade do programa e evitou comportamentos inesperados.