UNIVERSIDADE DE SÃO PAULO
Escola Politécnica

PCS3732 -- Laboratório de Processadores

Prova prática - Compilador FORTH em ARM

Professores:
Bruno Abrantes Basseto
Marco Túlio Carvalho de Andrade

Felipe Batista Arrais, N° USP: 11804268
Gabriel Stephano Santos, N° USP: 11831999
Igor Souza Lima e Silva Caixeta, N° USP: 11918564
Michel Trindade de Souza Brito, N° USP: 11257755

Introdução

A disciplina de Laboratório de processadores tem como propósito ensinar aos alunos conceitos práticos relacionados ao projeto e uso das arquiteturas de processadores, na disciplina foram abordados diversos temas relacionados a arquitetura de processadores, especificamente aplicado a arquitetura ARM, como o funcionamento de instruções, a divisão da memória, o funcionamento de I/O, comunicação entre coprocessadores e visões mais específicas sobre a arquitetura do ARM e do seu conjunto auxiliar o Thumb.

Também foram estudadas duas aplicações específicas do ARM, a placa Evaluator-7T e o Raspberry Pi, nesses momentos tivemos contato direto com compiladores para poder gerar instruções de máquina a partir de código na linguagem C e executar assim mais facilmente rotinas mais complexas.

Objetivos do projeto

Esse projeto visa a implementação de um compilador básico da linguagem FORTH para instruções de máquina na arquitetura ARM. Essa implementação tem o intuito de aprofundar os estudos nas instruções da arquitetura ARM e compreender melhor o funcionamento de compiladores simples que traduzem linguagens de alto nível para baixo nível.

A motivação desse estudo vem exatamente da busca por entender o que acontece por trás dos panos durante a compilação de um programa mais complexo, como os executados em sala de aula nas placas, para uma linguagem mais simples e por fim apreciar cada vez mais o esforço que gerou os ambientes de programação que desfrutamos atualmente.

A linguagem FORTH

FORTH é uma linguagem de programação baseada na estrutura de pilha para o armazenamento de dados e execução de comandos, utilizando a notação polonesa reversa.

A implementação de FORTH que será utilizada é uma versão simplificada, sem estrutura de dados adicional, permitindo somente a manipulação de dados numéricos inteiros com sinal e possuindo apenas alguns comandos básicos.

Entre os comandos básicos temos:

- Operações aritméticas simples, como: +, -, * e /;
- Operações condicionais, como: IF, ELSE e THEN;
- Operações de controle da pilha, como: DUP, DROP e SWAP;
- Operações lógicas, como: =, < e >.

Essa linguagem possui particularidades que permitem com que seja simples de implementar um compilador para ela, tornando-a um bom exemplo para a base de compiladores, na versão implementada da linguagem ela não opera sobre a memória diretamente, somente sobre a pilha, o que facilita o gerenciamento de memória, ela não possui estrutura de dados e ponteiros, o que deixa a implementação tanto do analisador sintático, quanto do gerenciamento de memória bem mais simples.

Além disso, as funções da linguagem são bem compatíveis com as instruções que a arquitetura ARM possui, assim sendo não é necessária grande manipulação de tipos de instruções diferentes para a execução das funções necessárias para a linguagem.

Por último é de grande relevância que a arquitetura já possui uma estrutura de pilha arquitetural, utilizando um registrador como ponteiro para o topo da pilha, assim é possível se aproveitar dessa estrutura para executar o programa FORTH, sem grandes manipulações de ponteiros e memória.

Funcionamento da linguagem

Na implementação utilizada da linguagem FORTH todas as entidades são números inteiros, quando é escrito no programa um número o mesmo é armazenado na pilha, quando o programa recebe um operador ele desempilha da pilha o número de operandos do operador, realiza a operação e coloca na pilha os resultados.

Forth também conta com a declaração de funções, na nossa implementação, no lugar em que as funções são declaradas elas também são executadas, mas podem ser referidas em outros lugares do programa, para serem utilizadas posteriormente. Outra coisa que vale ressaltar é o uso de operadores lógicos no forth, em que simplesmente 0 é considerado como falso lógico e qualquer número diferente de 0 é verdadeiro lógico, mas na nossa implementação convencionamos 1 com verdadeiro lógico devolvido pelas funções

Exemplo de programa simples:

```
: SQUARE DUP * ;
: CHECK-PYTHAGORAS-TRUE 3 SQUARE 4 SQUARE + 5 SQUARE = ;
```

Estrutura do compilador

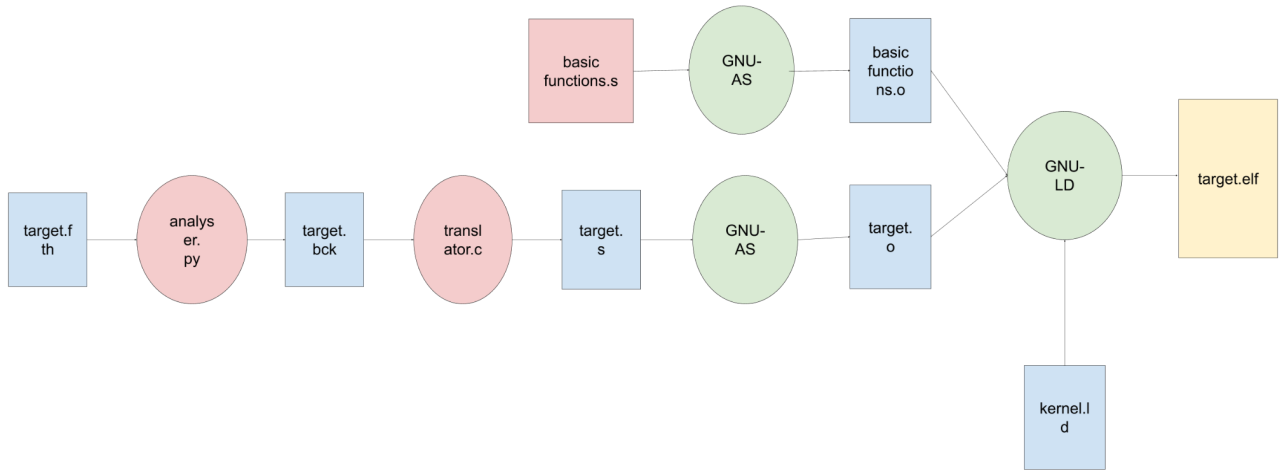
O compilador implementado é composto por três partes principais:

- O analisador léxico e sintático;
- O gerador de instruções ARM com base nas operações FORTH;
- A biblioteca de funções básicas implementadas em linguagem de máquina.

Também é conhecido que na maioria das implementações entre o analisador léxico e sintático e o tradutor são implementadas otimizações de código, porém por motivos de simplicidade esse passo não foi aplicado nessa implementação.

Por fim, para algumas funções como assembler e linker foram utilizadas as implementações do GNU, por conveniência e pelo fato de que essas etapas fugiam do escopo do projeto.

O fluxograma do compilador seria então:



Implementação

Analisador léxico e sintático:

Esse componente faz a checagem sobre a sintaxe da linguagem e, caso sua sintaxe esteja correta, gera um arquivo intermediário com tokens para que o código de máquina seja gerado.

Implementado em Python 3, o analisador utiliza essencialmente duas coleções de dados para verificar a estrutura do programa: uma tabela de símbolos, feita por meio de um dicionário Python, e uma "pilha de estados", que mantém controle do decorrer do código e contém informações sobre se o código está, por exemplo, dentro da declaração de uma subrotina, dentro de um bloco IF, e assim por diante. Dessa forma, a análise permite a identificação e indicação de erros de sintaxe.

Caso não haja erros, o programa mostra uma mensagem de sucesso relativa à escrita do arquivo de tokens target.bck, escrito também em FORTH, mas de forma que o Tradutor consiga entender de maneira mais fácil.

Gerador de código:

Esse componente recebe o arquivo simplificado pelo analisador léxico e sintático e gera o código em instrução de máquina correspondente.

Essa função foi implementada na linguagem C pelo arquivo translator.c, ela recebe uma entrada no arquivo target.bck e imprime o código no arquivo target.s, assim é possível utilizar o gnu-as para gerar o código alocável e, junto com o próximo componente, utilizar o gnu-ld para linkar os arquivos e formar o executável.

Biblioteca de funções básicas:

Essa biblioteca de funções é implementada diretamente em ASM e permite a execução de funções aritméticas, condicionais e de controle da pilha. A biblioteca deve ser compilada juntamente com o alocador de memória. Nesse projeto usamos

o alocador de memória do GNU para concatenar esse arquivo com o código principal.

Nesse módulo a utilização de pseudo instruções como pop e push foi muito conveniente para explicitar o funcionamento das rotinas, além disso foi necessário entender o funcionamento do conjunto de instruções para tomar decisões sobre qual a melhor maneira de implementar determinadas funções.

Devido à linguagem Forth funcionar utilizando pilha, ao realizarmos um pop {r0, r1} por exemplo, temos que ter em mente que o primeiro elemento da pilha está em r0 e o segundo em r1. Ademais, pela linguagem utilizar notação polonesa, devemos nos atentar na ordem dos operando, visto que “sub r0, r0, r1” é diferente de “sub r0, r1, r0”. Não apenas isto, mas os operandos foram chamados na ordem inversa da convencional, justamente pela notação em específico.

Resultados e conclusão

A construção de compiladores realmente é uma ciência complexa, além do compilador implementado ser muito simples comparativamente a outros compiladores, a linguagem escolhida também possui pequeno grau de complexidade e não foram implementadas fases de otimização de código em nível de instrução de máquina, e mesmo assim se demonstrou uma tarefa muito árdua.

Além disso, esse compilador é executado por arquivos que são compilados por outros programas, o que também demonstra a necessidade de se compilar compiladores. Ademais, ao final do projeto o grupo aprendeu sobre a operação de tradução de linguagens de alto nível para baixo nível, como estruturar essa passagem e como se aproveitar de maneira efetiva as características da arquitetura ARM para facilitar o desenvolvimento de aplicações mais complexas.