Simulação minimal/manual de um Compilador

Thiago Alexandre Nakao França

Universidade Tecnológica Federal do Paraná – UTFPR
COCIC – Coordenação do Curso de Bacharelado em Ciência da Computação
Campo Mourão, Paraná, Brasil
nakaosensei@gmail.com

Resumo

O objetivo principal deste trabalho foi produzir um codigo que "emulasse" o trabalho do compilador na passagem de um código de alto nível para assembly, depois usariamos tal arquivo para gerar as instruções em binario, para então usar essas instruções em um decodificador feito em C que interpretasse o intepretasse, retornando as instruções em assembly.

1. Introdução

O processo de compilação envolve o recebimento de um código em alto nível, que é transformado em assembly, e o assembly é quem faz a ponte para o código de máquina, neste trabalho a ideia era simular manualmente esse processo, gerando um código em Assembly com as habilidades de programação de um aluno humano, tendo-o em mãos, o próximo passo foi gerar as instruções em binário a partir do Assembly gerado, após isso, um decodificador de instruções em C deveria ser desenvolvido para que ele lesse a saida em binario e retorna-se o assembly original, ou pelo menos um assembly próximo do original.

2. O código de alto nível

Para o inicio dessa atividade, era preciso escolher um código em uma linguagem de alto nível, para um melhor aprendizado foi escolhido o algoritmo Bubble Sort clássico, pois por esse algoritmo conter dois laços de repetição e uma condicional dentre os laços, seria interessante desenvolver um Assembly manual para o algoritmo, a linguagem adotada foi C.

É importante destacar que o código dessa atividade requeria que houvesse laços de repetição, condicionais e chamadas de função, por se tratar de uma trabalho de disciplina de graduação.

3. O código Assembly

Tendo em mãos um código de Bubble Sort clássico desenvolvido na linguagem C, o segundo passou foi criar o codigo Assembly correspondente a esse código, para cada variável do programa, foi reservado um espaço na pilha do sistema, partindo da ultima posição, e também era importante não esquecer de antes de acessar qualquer variável, era preciso a acessar pela sua posição na pilha, ainda que ela ja estivesse carregado em um registrador.(Por razões didáticas e para de fato emular o que um compilador faz). Assim como a cada alteração em uma variável, o registrador que a representava deveria ser gravado na memória.

Para a visualização dos registradores e memória foi usado o simulador Mars, no final da execução, é possivel ver que os registros ficaram ordenados de acordo com o código em C proposto.

No código do bubble sort, os valores eram printados para a saida padrão, na minha implementação em assembly está presente o código para essa funcionalidade, mas por imaturidade minha na linguagem isso não funcionou.

Também, para o acesso as posições do array no assembly, eu mantinha a posição do stack frame como base e, para acessar qualquer registro, apenas movia o ponteiro para uma dada posição para fazer uma leitura ou escrita, depois da operação eu apenas retornava o ponteiro para a sua posição inicial.

O código em Assembly bubbleSortNakao.asm está todo comentado e descreve em detalhes qual foi o meu raciocinio durante o desenvolvimento, mais detalhes não serão descritos aqui por não ser uma disciplina focada em algoritmos, o desenvolvimento em assembly foi uma experiência interessante pois permite saber o que acontece com um código de alto nível quando ele é compilado em maiores detalhes, ainda que esse processo tenha sido apenas a simulação manual de um compilador já agregou valor.

4. O decodificador de instruções em C

O decodificador de instruções foi feito com base em um código deixado pelo professor Rogério Alves Gonçalves, a minha tarefa foi entender a estrutura por ele proposta, para a tarefoi foi preciso:(Entender o uso das mascaras, operador bitwise, notar a diferença entre os formatos das instruções, adicionar instruções ao decodificador, criar funções para pegar partes especificas das instruções).

Como já citado anteriormente, o decodificador recebe como entrada um arquivo binario, e printa na saida padrão o codigo Assembly do Mars correspondente, é notavel que a saida não foi identica a entrada por questão de endereçamento de memoria de labels, além de eu não ter encontrado a configuração do comando 'la'.

5. Conclusões

Após o desenvolvimento deste trabalho foi possível ter uma visualização do que acontece com o código de alto nível quando compilado, os formatos de instruções, acabei aprendendo como lidar com numeros hexadecimais em C, também foi uma boa experiência no desenvolvimento manual de um código em Assembly, pode vir a ser util em trabalhos futuros.