Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas

Organização e Arquitetura de Computadores

Grupo:
Alan Pedro Silva Pessoa 18.1.8001
Renan Saldanha Linhares 18.1.5908

Professor: Carlos Henrique Gomes Ferreira

2019
João Monlevade

Universidade Federal De Ouro Preto

Trabalho Prático:

Documentação referente ao trabalho prático de Organização e Arquitetura de Computadores proposto aos alunos como forma de obtenção parcial dos créditos necessários para aprovação na disciplina.

Professor: Carlos Henrique Gomes Ferreira

Junho
2019

Introdução:

O trabalho exige que um programa escrito em linguagem C leia um arquivo na extensão ".a", pegue seus dados, interprete e os converta para um código em binário, cada item citado será abordado com mais detalhes no futuro. O programa montador tem a função de executar esta tradução, e, para isto, utiliza do método de conversão apresentado pelo livro de arquitetura de computadores de Willian Stallings e em conteúdo digital disponibilizado pela UNICAMP. Na primeira leitura do arquivo, o programa captura os rótulos e na segunda traduz em código binario de máquina apropriado.

Main e Estrutura:

A função principal recebe diretamente do usuário o nome do arquivo que deseja realizar a tradução. Com a ajuda de um laço infinito, o arquivo é buscado e verificado sua existência, em caso positivo, o laço é quebrado, mas se nome do arquivo citado não existir, então é oferecida a opção de buscar novamente ou sair do programa, assim reiniciando o ciclo sempre que é optado pela busca e, porém, não é bem-sucedida. O usuário pode quebrar o laço a qualquer momento se optar por não buscar novamente o nome do arquivo, ressaltando que o arquivo deve estar na mesma pasta do programa para que ele seja encontrado.

Para auxiliar durante o registro e organização do montador, uma estrutura é criada com o objetivo de receber de cada rótulo seu endereço e seu valor correspondente em binário. Para isto um vetor de caracteres foi declarado para cada item, e a estrutura foi definida com a palavra "Endereço". É esta estrutura que acompanhará todo o processo de montagem, onde será escrito e buscado dados durante o processo.

Funções de Conversão:

Há duas funções que são básicas neste tópico, uma apenas para a conversão de um valor decimal para binário e outra para a conversão de decimal para hexadecimal. Ambas são feitas pelo método de divisões consecutivas pelo valor de sua base e adaptando os restos, no caso da conversão em hexadecimal, substituindo os valores quando necessários. Duas outras funções ajudam a identificar o registrador necessário, ou os registradores necessários, para a

execução, enquanto uma terceira encontra o endereço de memória e o guarda em uma string para uso futuro.

Uma função de conversão serve como uma espécie de banco de dados, ela recebe uma matriz de rótulos, a estrutura criada no início do arquivo, o nome do arquivo e a linha da ser posicionado. Após a interpretação do comando, uma próxima função executa o registro no arquivo destino, utilizando antes as funções de conversão citadas para escrever o número da linha em hexadecimal. Com estas informações o código interpreta o comando e escreve no arquivo com o nome passado. Deve-se ressaltar que este conjunto de funções interpreta e escreve a instrução no arquivo .mif durante o segundo passo da execução.

Por último, um recurso é criado para adaptar números negativos ao código, portanto, utilizando o método do complemento de dois, os números são convertidos quando necessário. A lógica para a função é simples, troca-se os valores de cada bit na string passada como parâmetro para função. Para finalizar a conversão, o número convertido é somado então a seu valor positivo, e em caso de overflow o bit mais significativo é ignorado.

Funções Principais:

Durante o primeiro passo, o número de linhas do código a ser traduzido é contado, é especificado um valor máximo de 128 linhas, e caso seja excedido o programa emitira um aviso de overflow e encerrará a aplicação. Ao capturar cada rótulo, é feita algumas separações e especificações para melhor distinguir cada um, por exemplo, para classificar uma instrução como label, utilizamos a ideia de que estas sempre estarão entre um caractere underline (_) e um caractere dois pontos (:). A seguir o programa recebe a instrução e neste estágio, se fez necessário um tratamento especial para as pseudo instruções do tipo “.data”, por isto um teste de comparação com a string capturada nos informará nos casos especiais, bem como se seu endereço não for um valor divisível por 2. Algumas condições indicam se um rótulo chegou ao fim, se o próximo caractere capturado for um espaço, então aquele comando chegou ao fim e seu endereço é registrado. Os caracteres de quebra de linha (\n) e de inicio de comentário (;) também indicam que a captura chegou ao fim. Tudo isto pertence a um ciclo while, que segue capturando as instruções e registrando seu endereço até o final do arquivo.

A pseudo instrução .data recebeu uma função exclusiva para seu tratamento. Como o seu número de linhas é variável, uma desambiguação foi feita para seu tratamento. Primeiramente um é valor captado em caractere, este valor corresponde ao número de bytes que cada pseudo instrução deste tipo irá necessitar, por esta razão, o número mínimo de bits definido é oito e como o byte da palavra possui quatro bits, este valor deve sempre ser par, após sua captura é convertido em um número inteiro, e este é convertido para um valor binário que será alocado em uma string que será registrada.

Para o segundo passo, os dados obtidos até o momento devem ser convertidos para linguagem de máquina, logo, o arquivo com a extensão ".a" deve gerar um novo arquivo com o mesmo nome, porém com a extensão ".mif". Portanto a primeira ação executada neste processo é receber o nome do arquivo e guarda-lo com sua extensão alterada. É então criado um novo arquivo e registrado as definições do montador, como o número máximo de linhas (altura) e o número de bits por instrução. Para evitar qualquer erro, o arquivo é então fechado e reaberto para escrita em binário, e assim começamos de fato o segundo passo.

Para a montagem em binário, a escrita ignora os rótulos "label" e captura somente as outras informações. Uma metodologia utilizando matriz foi escolhida para esta etapa. Como forma de controle, caso um comentário seja identificado, um salto será feito para a próxima linha, tomando o cuidado para quando atingir o final do arquivo a leitura se encerre. Além deste, outros tratamentos especiais foram planejados para o código, sempre que encontrar uma quebra de página, comentário, espaços seguidos ou final do arquivo o vetor de caracteres é encerrado, substituindo a última posição válida por "\0".

Com a ajuda de algumas funções auxiliares já descritas, os comandos registrados são convertidos para linguagem de máquina, os rótulos ".data" requerem uma adaptação para serem finalizadas. O arquivo destino (.mif) é então reaberto, e se o número de linhas for menor que 128, a quantidade restante é então resumida na forma: "[XX..7F]: 00000000;", em que XX é o número da próxima linha após a última linha escrita em hexadecimal.

Outros Exemplos:

Como forma de autenticar o programa montador, outros dois programas são requisitados para exemplo de teste. Isto posto, um código para calcular fatoriais e outro para calcular potencia foram definidos. Ambos os arquivos ".a" e ".mif" estão dispostos no trabalho.

Para o programa que calcula o fatorial a seguinte lógica foi implementada: é lido um valor e este é passado para o topo da pilha, a função de FATORIAL é então iniciada. A constante um é carregada para um primeiro registrador que servirá como parâmetro para a instrução de subtração, decrementando nosso valor em uma unidade a cada loop, este valor é copiado em um segundo registrador auxiliar para que possa servir como o parâmetro da multiplicação, sem alterar o registrador original. Caso o registrador auxiliar possua um valor zero ou menor que zero, o ciclo é então quebrado indicando que o fatorial está pronto. Caso contrario nosso registrador original recebe sua multiplicação pelo registrador auxiliar (que começa com uma unidade inferior) e o loop é seguido, como a cada ciclo este registrador auxiliar é decrementado, o intuito de reproduzir uma função fatorial é concluído com sucesso.

A função de potencial requer dois parâmetros, um valor e a qual potencia este deve ser elevado, portanto, dois registradores são lidos e empilhados. Como este programa não executa potencias negativas, um erro é emitido para caso o valor da potencia seja negativa, e caso a potencia seja zero, a constante um é carregada no registrador. Um valor constante de uma unidade negativa é carregado a um registrador de constante, e o registrador original é copiado para um registrador auxiliar. Um loop é então iniciado e o registrador original é multiplicado por si mesmo a cada ciclo, no final do loop, o registrador auxiliar é somado com o constante, ou seja, decrementado em uma unidade, e o loop reinicia. Quando o registrador auxiliar adquiri o valor zero, quer dizer que a potencia foi executada o número previsto de vezes, realizando, assim, a operação.

Referências:

Ivan L. M. Ricarte 14-02-2003. http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node90.html último acesso em 08-06-2019, 12:21

Apêndice – Willian Stallings. Arquitetura e Organização de Computadores. Pearson Education, 8° ed., 4° reimpressão, Novembro de 2013.