UNIVERSIDADE FEDERAL DE SANTA MARIA

CENTRO DE TECNOLOGIA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

PROGRAMA DESMONTADOR PARA PROCESSADOR MIPS DESENVOLVIDO EM ASSEMBLY

Nikolas Machado Corrêa

SANTA MARIA – RS

2017

INTRODUÇÃO

Existem dezenas de linguagens de programação que podem ser escolhidas quando se decide montar um programa, e alguns fatores, como a eficiência e a implementação que se quer fazer influenciam nessa escolha. No entanto, o que muda de uma para outra basicamente é a representação, pois elas são adaptadas para que o ser humano possa compreendê-las melhor, algo que um computador não consegue processar da mesma forma.

Sendo assim, ao executar um programa em uma linguagem qualquer, subprogramas efetuam algumas “traduções” do código para outras linguagens, de tal forma que se torne mais simples para que as máquinas entendam. Uma dessas linguagens é conhecida como Assembly, também chamada de linguagem de montagem, que, ainda assim, os dispositivos computacionais não entendem, mas que facilita o processo do assembler ou montador, um programa capaz de gerar comandos que a máquina entende a partir de um código em Assembly.

Neste trabalho, o programa desenvolvido foi baseado nesta noção, porém, ao inverso. Isto é, foi projetado um desassembler, ou desmontador, que difere do assembler por fazer o contrário, ou seja, transformar um código entendido pela máquina em uma instrução em Assembly, compreensível pelos humanos.

OBJETIVOS

Demonstrar a partir de um arquivo texto como entrada, que o programa desenvolvido em Assembly para simular um desmontador é capaz de transformar códigos escritos em linguagem de máquina em instruções Assembly. Para isso, vale ressaltar que o processador utilizado é o MIPS (**M**icroprocessor without **i**nterlocked **p**ipeline **s**tages).

METODOLOGIA

O editor de texto e também montador do programa utilizado foi o software MARS para o processador MIPS. Quanto a organização do código, foram utilizados recursos da linguagem que facilitam a legibilidade do mesmo, como, por exemplo, as macros e os procedimentos (funções).

Para que o programa funcione, é fundamental que haja a compreensão sobre como as instruções são representadas em um processador MIPS e qual é a sequência lógica de decodificação desses códigos em palavras. Em primeiro lugar, a cada 8 dígitos de um código de máquina é representada uma instrução. Esse 8 dígitos, escritos em hexadecimal, são traduzidos para binário, gerando uma sequência de 32 bits. Logo após, verifica-se os 6 primeiros bits e a partir disso já é possível determinar o tipo da instrução, facilitando a desmontagem.

De acordo com o tipo de instrução, os outros 26 bits são reagrupados e separados em diferentes campos, sendo que cada um representa uma informação sobre a instrução e a soma dos bits de todos os campos resulte em 32. A partir disso, o processo de decodificação consiste em comparações dos dados de cada campo com a palavra (string) equivalente em Assembly.

Desta forma, o código efetua os processos de decodificação descritos acima até que o arquivo de onde o texto está sendo lido chegue ao final. Para simplificar o entendimento do processo, foi anexado a este trabalho um arquivo denominado “Representação das instruções”, em que consta mais detalhadamente e com exemplos o funcionamento da decodificação dos códigos juntamente com o arquivo “entrada”, utilizado como exemplo para a tradução.

REFERÊNCIAS BIBLIOGRÁFICAS

Patterson, David A., Organização e projeto de computadores: a interface hardware/software / 4. ed. Rio de Janeiro, RJ : Elsevier, 2014.