Projeto de Arquiteturas de Computadores II

Professor:

Gabriel Dias Scarpioni

Equipe:

Cícero Henrique Maciel Penha – 1121 – GEC

Evandro Pijanowski Andrade – 1210 – GEC

Introdução

Este software tem por objetivo simular máquina virtual a partir de um interpretador de instruções. O grupo decidiu utilizar uma arquitetura de 16 bits e os comandos add e sub são representados por 0000 e 0001, respectivamente. Para isso o software foi desenvolvido na linguagem de programação de alto nível Python.

Descrição de funcionamento

Neste projeto a lógica principal do software está no arquivo “main.py”. O software inicia abrindo o arquivo de texto "instruction,txt”, onde ficam as instruções que o programa deverá executar, dando início a um loop que percorrerá todas as instruções armazenadas. Dentro do loop o software lê uma palavra e em seguida aciona a “get_inst” que irá decodificar esta palavra. Para isso a função separa a palavra em suas respectivas variáveis correspondentes por meio das funções “get_op” e “get_reg”, que segue o seguinte funcionamento:

- Entre o dígito 0 ao 3: correspondem a operação a ser realizada;
- Entre o dígito 4 ao 6: correspondem ao primeiro registrador;
- Entre o dígito 7 ao 10: correspondem ao segundo registrador;
- Entre o dígito 11 ao 14: correspondem ao terceiro registrador;

A função “get_op” retorna qual a operação deve ser executada e a função “get_reg” retorna quais serão os três registradores que a operação irá utilizar. Após separar o bloco em variáveis, a função “get_inst” chama a função “confirm_reg”, esta função é responsável por buscar o registrador na memória cache do sistema, caso ela não o encontre a função acessa a memória principal, atualiza a memória cache, evitando assim possíveis erros, e busca novamente o registrador na memória cache.

Em seguida a função “get_inst” aciona a função “load” para retornar os respectivos valores a serem operados. A função “load” tem por objetivo buscar na memória os respectivos valores que estão guardados nos registradores. Ela recebe e trabalha com os valores convertidos em decimal, para isso existe a função “convert” que faz essa conversão de números binários para decimais.

Depois de já deixar tudo preparado para a execução a função “get_inst” então aciona a função “execute” para realizar as operações. A função “execute” tem como parâmetros o valor que indica a operação desejada e os três registradores que irão ser utilizados. A execução funciona assim:

1- Verificação de qual operação será executada;
2- Invocação da função “save”;
3- Invocação da função “add” ou “sub”;
4- A função de operação matemática recebe os valores que irão ser operados, faz o cálculo e retorna para “save” o valor final.
5- A função “save” então vai na memória atualiza o novo valor do registrador alterado.

Assim o programa encerra a função “save”, retorna para a função “execute” que também é encerrada, terminando a função “get_inst” e finalizando uma instrução. O software permanece neste loop até que todas as instruções sejam executadas.

Especificações da memória cache

Os três primeiros dígitos encontrados na cache correspondem a TAG, indicando quais são as posições de memória no arquivo de dados, a TAG está na numeração binária. O quarto dígito corresponde ao dado a qual aquela TAG pertence, sendo esse em numeração decimal.

Mesmo não tendo sido implementado o bit de validação será sempre 1 pois a cache estará sempre atualizada.

O tamanho do bloco é implementado é: TAG + dados = 4 bits

Capacidade de palavras: 4

Link do Github

https://github.com/EvandroPijanowski/EC-208/tree/master