Arquitetura de Computadores - Trabalho 1

Prof. Eriko Werbet - Universidade de Fortaleza

Construir um emulador

1. Construir um emulador em software para emular a arquitetura de um dispositivo computacional baseado nos seguintes componentes:
    a. [2,0 pontos] Registradores (5) [A, B, C, D, PI]
    b. [2,0 pontos] CPU
    c. [2,0 pontos] RAM
    d. [2,0 pontos] Barramento
    e. [2,0 pontos] Módulo de Entrada e Saída (E/S)

2. Parâmetros:
    a. Tamanho da palavra em bits [16, 32 ou 64];
    b. Tamanho da RAM em bytes [128, 256 ou 512];
    c. Tamanho do buffer de entrada/saída em bytes [64, 128 ou 256];
    d. Largura do barramento em bits [8, 16 ou 32];
3. Assembly a ser executado:
    a. Instrução **mov**, mover dado:
        i. “mov R, i”; onde R é um registrador qualquer e i é inteiro literal de 32 bits. Exemplo: “mov A, 2”.
        ii. “mov E, i”; onde E é um endereço de memória da largura do barramento e i é um inteiro de 32 bits. Exemplo: “mov 0x00000001, 2”;
        iii. “mov E, R”;
        iv. “mov R, E”;
    b. Instrução **add**, adição inteira:
        i. “add X, Y”; onde X pode ser um registrador ou endereço de memória e Y pode ser um registrador ou inteiro literal. Além disso, o resultado sempre é armazenado no operando X. Exemplos:
            1. “add R, i”;
            2. “add R1, R2”;
            3. “add 0x00000001, R”;
            4. “add 0x00000001, 2”;
    c. Instrução **inc**, incremento inteiro:
        i. “inc X”; onde X pode ser um registrador ou endereço de memória. Incrementa o operando em uma unidade.
    d. Instrução **imul**, multiplicação inteira:
        i. “imul X, Y, Z”; onde X pode ser um registrador ou endereço de memória e Y/Z pode ser registrador, endereço de memória ou inteiro literal. Exemplos:
            1. “imul R1, R2, R3”; //R1 = R2 * R3;

2. “imul R1, R2, 5”;
3. “imul 0x00000001, 5, R”;

4. Funcionamento:
    a. Arquivo de código assembly é parseado e as instruções são codificadas de acordo com o tamanho da palavra; cada instrução é armazenada na RAM pelo módulo de E/S. As instruções são enviadas pelo barramento e armazenadas em células da RAM. A RAM deve ser representada por uma estrutura de dados indexada (vetor, por exemplo). O barramento também deve ser representado por uma estrutura de dados e deve fazer a ligação entre a E/S e a RAM.
    b. CPU (pode ser representada por uma função ou classe) pega a primeira instrução do programa na RAM (via barramento) e depois atualiza o ponteiro de instruções (registrador PI neste emulador) com o endereço desta instrução.
    c. CPU executa a instrução usando os registradores (variáveis ou objetos) e a RAM.
    d. CPU busca uma nova instrução na RAM, atualiza o PI e executa a instrução.
    e. CPU repete 2.4. até concluir a execução do programa.
    f. O emulador deve ser capaz de mostrar um “mapa” da memória e o resultado de cada instrução sendo executada em tempo real.

5. Exemplo de código:
    a. mov A, 2 #carrega o valor 2 no registrador A
    b. mov B, 3 #carrega o valor 3 no registrador B
    c. add A, B #soma o valor em A com o valor em B, armazena o resultado em A
    d. mov 0x0001, A #carrega o valor de A no endereço 0x0001 da RAM
    e. inc 0x0001 #incrementa o valor no operando em uma unidade
    f. imul C, 0x0001, 4 # C = 0x0001 * 4
    g. mov 0x0002, C #carrega o valor de C no endereço 0x0002 da RAM