Organização e Arquitetura de Computadores 1/2015
Prof Dr. Ricardo Pezzuol Jacobi
Turma C

Simulador MIPS

Aluno Matheus Yamamoto Matsumoto
10/0017967

Descrição do problema:

Neste trabalho implementamos um simulador da arquitetura MIPS em C, as entradas do programa são as intruções de 32 bits já em formato binário, e não em codigo assembly, foi utilizado um array de inteiros com 8192 posições, para simular a memória, tanto de dados quanto de código.

Foram implementadas trinta e sete instruções, e funções básicas de um simulador para montar a memória, fazer a busca da instrução, decodificar a instrução e executa-la, e também funções para mostrar o estado dos registradores e da memória.

Descricao sucinta das funcoes implementadas:

As funções prinicipais foram declaradas em funcoes.h e implementadas em funcoes.c, e sao as seguinte:

uint32_t fetch(uint32_t pc);
-Le instrucao da memoria e incrementa PC

void decode();
-Quebra a instrucao lida em varios pedacos como: opcode, rd,rs,rt e outros.

void execute();
-Pega as variaveis criadas no decode(), e as executa, lendo o opcode, e direcionando executando funcoes declaradas em isa.h e implementadas em isa.c

void step();
-Chama as funcoes fetch(), depois decode(), e então execute(), e avisa se programa foi terminado.

void run();
-Chama step(), até perceber que programa foi terminado.

void dump_mem(int start, int end, char format);

-Imprime um pedaco da memoria na tela, especificado pelo usuario, e tambem o formato se em base 10 ou base 16.

void dump_reg(char format);

-Quase a mesma coisa que dump_mem(), mas imprime o banco de registradores inteiro, no formato especificado pelo usuario.

void mem_mount(char* filename_text, char* filename_data);

-Le arquivos de entrada code.bin e data.bin e escreve na memoria.

Testes e resultados:

Para realizar os testes foram criados dois arquivos, test_isa.c e test_funcoes.c, as funcoes foram todas testadas em test_funcoes.c, com entradas e saídas controladas, os testes das instruções da isa foram realizados em test_isa.c mas também em tempo de execução com programas de testes, utilizando printf's e getchar's para acompanhar a execução das funcões.

A execução dos programas fornecidos foram executados sem erros, com as saídas esperadas comparando com as saídas do simulador MARS.