Universidade Federal de Mato Grosso do Sul
Alunos: Alberto Y. Hisano Higuti RGA: 2021.1905.016-0
Eduardo Lopes de Lemos RGA: 2021.1905.034-9
Disciplina: Arquitetura de Computadores II
Professor: Renan Albuquerque Marks

Relatório do Trabalho Prático

O emulador foi escrito em C++. Nele foram implementadas todas as instruções solicitadas na Descrição do Trabalho e para melhorar a organização, as instruções do MIPS foram implementadas em diferentes arquivos dependendo de sua função. Por exemplo, as instruções Add, Sub foram feitas no arquivo arithmetic_instructions.cpp, e assim por diante.
Além disso, foram implementados também todos os Syscalls solicitados e a principal estrutura de dados utilizado foi uma Struct. De recursos foram utilizados o Visual Studio Code com as extensões disponíveis para C++, o editor hexadecimal Okteta para analisar os arquivos .bin gerados do Simulador MARS e o KDiff3 para comparar e verificar se a formatação da saída está correta.
O código do trabalho funciona da seguinte forma:

- É dividido em 2 partes principais, a main e o emulador, a main possui a função de carregar os arquivos de data e text, enquanto o emulador é responsável por todo o resto;
- Inicialmente se constroem os registradores, e a memória (dividida em campos de texto, dados e pilha);
- Após isso se carregam os valores para a memória usando o fread do C++;
- Com os dados na memória, chama-se a função run_program do emulador que, basicamente, percorre a memória inteira, de forma que o registrador PC recebe o valor zero e soma-se a cada instrução lida um valor de 4 palavras ou 32 bits no registrador;
- A função run_program encerra seu funcionamento em dois casos, a instrução Syscall 10 é lida pelo código ou a memória acessada não existe;
- Com exceção das duas situações descritas acima, o funcionamento da função se baseia em:
    - Buscar instrução da memória (onde é nescessario pegar cada byte e organizá-los);
    - Converter a instrução em binário para o formato local do nosso programa chamado de “Mips_instruction”, que consiste em uma Struct;
    - As “Mips_instructions” possuem todos os campos possíveis de qualquer instrução do MIPS, porém, dependendo do Tipo da Instrução (R, I, J), seus campos são preenchidos de maneira diferente. As instruções são convertidas pela função “bin_to_MIPS”, que reconhece o tipo de instrução e preenche os campos corretos, atribuindo um id único que é reconhecido por um “dicionário” interno do programa.
- Após isso, o programa executa a instrução, ao chamar a funcão que tem como key o id de cada instrução e passa a “Mips_Instruction”, o banco de registradores e o banco de memória como argumentos da função;
- Finalmente, depois da execução de tudo o programa chama a função “print_registers” e o “print_memory” para que a saída gerada seja igual a solicitada no Trabalho.

As principais dificuldades encontradas durante o desenvolvimento do trabalho foram:

- Conversão dos binários para o formato de instrução local;
- Instruções de Branch;
- Rodar os comandos para gerar os arquivos necessários do Simulador MARS.

Para o Software de Demonstração foi escolhido uma aplicação simples e inicial na área de Inteligência Artificial, que consiste nas “hidden layers”. As “hidden layers” são, resumidamente, uma implementação concreta de funções afins “stackadas” entre si, onde o resultado de uma das funções afim chamadas de nodos (nós) pode ser usado como entrada para outra função afim, assim como a soma de vários resultados também pode ser usado como entrada. Essa configuração das

“hidden layers” é o que permite que redes neurais aprendam características mais complexas sobre sua tarefa, ainda que para tarefas muito complexas seja necessário usar outras estruturas, todas elas tomam como base o princípio de propagação de informação que as “hidden layers” utilizam, ou seja, sendo possível implementar uma dessa estrutura é possível implementar qualquer rede neural moderna em seu dispositivo com arquitetura MIPS.