Nome:	Fabrício Ferreira da Silva
RA:	231900
Disciplina:	MO601 (Arquitetura de Computadores II)

Projeto 2 - Um simulador simples do processador RISCV.

Nesse projeto, cada aluno deve implementar um simulador básico do processador RISC-V. A intenção é praticar conceitos de simulação de processadores, que são mais complexos que os circuitos lógicos do projeto 1, além de ter uma visão da eficiência desses algoritmos.

1 Descrição geral do projeto

O objetivo geral do projeto, é implementar um simulador do processador RISC-V RV32IM, que significa a versão de 32 bits com as instruções básicas e também as instruções de multiplicação e divisão.

Para teste o simulador deve executar todos os programas de inteiros fornecidos pelo repositório do ACStone, que é um conjunto de programas em linguagem C, e ao fim de cada um deles gerar uma linha de log para cada instrução executada.

O simulador também deve ser capaz de contar ciclos de execução, para esse projeto será considerado que cada instrução gasta um ciclo, além de encapsular todos os acessos à memória através de alguma função, para permitir que seja possível inserir temporização na memória alterando apenas essa função.

2 Descrição do seu ambiente de desenvolvimento

Esse projeto foi desenvolvido no sistema operacional Ubuntu 18.04 LTS, como linguagem de programação foi usado o Python versão 3. Não foi usada nenhuma biblioteca externa, somente as nativas, além do toolchain para RISCV disponível para instalação através do comando “sudo apt-get install -y gcc-riscv64-linux-gnu”.

3 Descrição do seu algoritmo de simulação

Primeiramente o algoritmo coleta todos os nomes dos programas dentro da pasta “test/objdump”. Em seguida, ele define as variáveis iniciais e define as funções principais do simulador, são elas:

- memory_access: responsável pelas operações de acesso a memória, tanto load quanto store, não foi adicionado nenhum tempo para essa função, mas usando o comando “time.sleep” é possível adicionar esse tempo, deixando o simulador mais próximo a um processador real.

- exec_inst: responsável pela execução de cada instrução, nela estão definidas todas as instruções necessárias para a execução dos programas do ACStone de inteiros, multiplicação e divisão.
- generate_log: responsável pela geração da linha de log para cada instrução executada, salvando-as em uma variável que será exportada mais a frente.

Em seguida, para cada programa, ele zera as variáveis iniciais, carrega as instruções do arquivo gerado pelo “objdump” e salva na memória, além de definir o PC inicial, que é o primeiro endereço da primeira instrução do bloco “start”.

Após esses passos ele inicia o ciclo de “Fetch-decode-execute”, nele durante o “fetch” é realizado o “load” da instrução presente na posição de memória do PC inicial, em seguida se inicia o “decode” onde cada campo da instrução será dividido, e a partir do “opcode”, “func3” e “func7” (dependendo da instrução que será executada), descobrir qual é essa instrução.

O último passo é o “execute” onde a instrução decodificada é executada da maneira correta chamando a função “exec_inst” e então um ciclo se passa, voltando ao início do “Fetch-decode-execute”. Esse processo continua até que a função “ebreak” seja executada. Dentro dessa função foi escolhido setar o PC para o valor “20000000”, o que indica o fim desse programa e o laço while é interrompido, seguindo para o próximo programa.

Para a implementação de cada instrução necessária foi utilizado o documento “riscv-spec-v2.2”, onde cada uma delas é explicada em detalhes tanto sobre seu funcionamento quanto suas características.

Para cada fim de execução de um programa, é gerado um arquivo com o nome do programa no formato “.log” com todas as linhas de log geradas, eles são armazenados na pasta “logs” dentro da raiz do repositório.

Ao fim da execução de todos os programas é gerado um arquivo chamado “estatísticas.txt”, nele cada programa é ordenado por sua eficiência em relação a tempo por instrução, nesse computador o mais rápido foi o “134.call” que conseguiu um tempo de 6.04 microsegundos por instrução, além de ser o maior deles, que gastou 41.386 ciclos para ser concluído.

4 Descrição de como você testou seu projeto

Para testar o projeto foram manualmente comparados os logs gerados com os resultados esperados no disassembly de cada “objdump”, além de comparar também com os programas em “C” que foram usados como entrada para o projeto. Todos os exemplos testados funcionaram corretamente.

5 Considerações gerais sobre seu aprendizado nesse projeto

Eu nunca tinha trabalhado com simuladores de processadores, achei bem interessante a experiência de ter que desenvolver uma forma própria de decodificar instruções descritas em um

documento que explica como cada uma delas funciona. Achei a documentação do RISCV bem completa o que facilitou muito o processo de implementação.

Foi muito importante poder aplicar os conhecimentos adquiridos durante as aulas sobre o funcionamento de um RISCV, já que na prática foi possível acompanhar passo a passo como funcionam os ciclos do processador, os registradores, acessos a memória, entre outros.

Isso foi uma ótima preparação para próximos projetos que visem alguma forma de simulação de processadores, já que foram aprendidas diversas ferramentas para trabalhar com os mesmos, criando uma forma de estrutura nova e adequada para os propósitos do projeto, seguindo é claro as orientações fornecidas pela documentação.

Outro aprendizado foi trabalhar com manipulação de informações em nível binário ou hexadecimal, já que normalmente trabalho com programas em mais alto nível, onde os dados são na maioria dos casos em decimal. Aprendi conceitos muito importantes como o signed extended ou unsigned, além de como manipular e converter os mesmos, o que foi de extrema importância para a correta execução das instruções.

Devido a falta de expertise em como manipular os dados e no entendimento dos conceitos necessários para a execução de cada instrução, quanto para a preparação do ambiente para funcionamento dos mesmos, acabei gastando mais tempo que o esperado e agradeço grandemente pela semana extra disponibilizada, o que proporcionou o tempo que faltava para após o processo de aprendizagem apenas implementar as funções de maneira correta e assim concluir com êxito esse projeto.