UNIVERSIDADE DE BRASÍLIA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
116394 ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

Trabalho 2: Simulador RISC-V

João Pedro de Oliveira Silva 190057807

1. Objetivos e descrição

Este trabalho consiste na implementação de um simulador da arquitetura RV32I em linguagem de alto nível (C++). As funções básicas de busca e decodificação de instruções são fornecidas. Deve-se implementar a função de execução (execute()) das instruções para o subconjunto de instruções indicado. O programa binário a ser executado deve ser gerado a partir do montador RARS, juntamente com os respectivos dados. O simulador deve ler arquivos binários contendo o segmento de código e o segmento de dados para sua memória e executá-lo.

2. Materiais e métodos

2.1. Materiais

- **C++/G++:** Linguagem e compilador, versão g++ (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
- **Windows 10/WSL2:** WSL de Ubuntu 20.04 no Windows 10.
- **Visual Studio Code:** editor de texto utilizado.
- **RARS:** Montador Assembler que roda em JAVA.

2.2. Métodos

Necessário obter casos de exemplo para o uso do simulador, este fornecido pelo professor, o qual para isso foi feito um dump de memória de um código assembly qualquer em binário no RARS. Esse arquivo binário já codificado é o que será usado no simulador do trabalho.
Parte do trabalho foi disponibilizado pelo professor. Juntei os arquivos **globals.h** e **riscv.h** em um único arquivo, no **globals.h**, também foi acrescido algumas funções e variáveis extras. O código **riscv.cpp** foi quase todo retirado do código do professor, o qual este código foi mostrado em aula.

Com a leitura das linhas em binário e suas decodificações, então bastou criar as funções pedidas nas especificações do trabalho, levando em consideração as variáveis e métodos gerados pelo código do professor.
Essas são as instruções requisitadas:

• add	• beq	• bltu	• lbu	• ori	• srli
•addi	• bne	• jal	• lw	• sb	• sub
•and	• bge	• jalr	• lui	• slli	• sw
•andi	•bgeu	• lb	• nop	• slt	• xor
•auipc	•blt	• or	• sltu	• srai	• ecall

ecall é uma função que faz diversas coisas com o sistema, como printar bytes ou receber input do io. Para esse trabalho, era simplesmente preciso implementar ECALL com as funções de printar um inteiro referente ao teste e printar duas possíveis frases:

Teste %d OK
Teste %d FAIL

3. Testes e Resultados

O trabalho foi executado com o teste.asm fornecido pelo professor, imagem apenas de pequena parte do código.

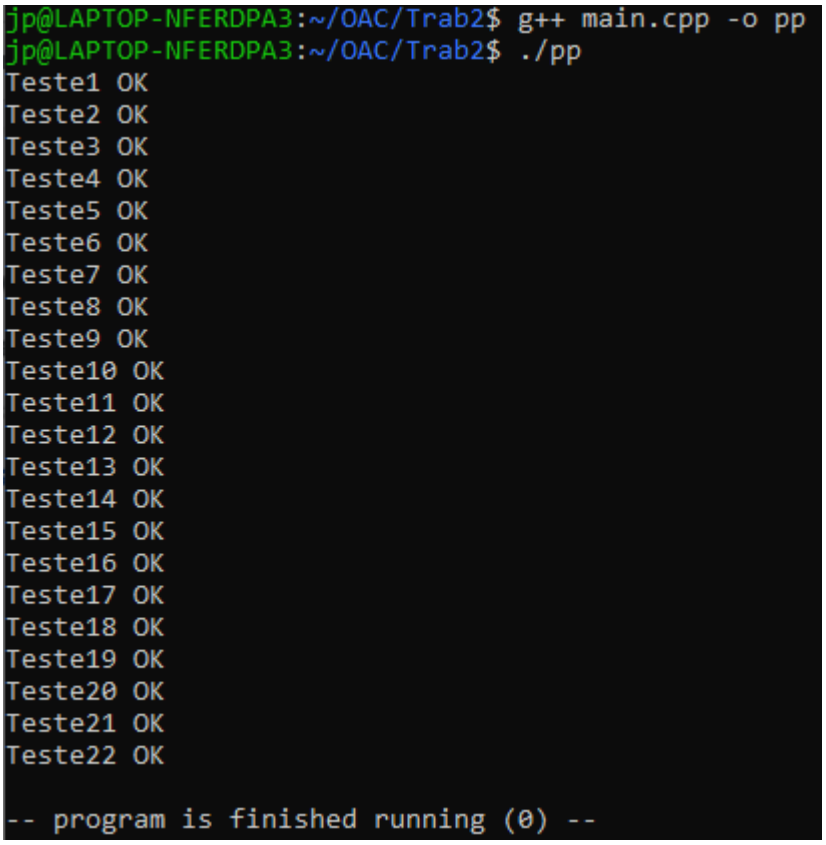
```
1   .data
2   word:    .word 0xFAFEF1F0
3
4   TAB:     .asciz "\t"
5   NL:      .asciz  "\n"
6   Label:   .asciz "Teste"
7   l_ok:    .asciz  " OK"
8   l_fail:  .asciz  " FAIL"
9
10  .text
11
12  teste1:
13          li t1, -2       # Testa ADD
14          li t2, 3
15          add t3, t1, t2
16          li a1, 1
17          li t6, 1
18          beq t3, t6, t1_ok
19          jal FAIL
20          j teste2
21  t1_ok:
22          jal OK
23
24  teste2:
25          li t1, -2048    # Testa ADDi
```

Resultado:
Teste1 OK
Teste2 OK
Teste3 OK
Teste4 OK
Teste5 OK
Teste6 OK
Teste7 OK
Teste8 OK
Teste9 OK
Teste10 OK
Teste11 OK
Teste12 OK
Teste13 OK
Teste14 OK
Teste15 OK
Teste16 OK
Teste17 OK
Teste18 OK
Teste19 OK
Teste20 OK
Teste21 OK

Teste22 OK

-- program is finished running (0) --

Resultado do Simulador criado:



4. Conclusão

O trabalho consistia em implementar diversas funções de um processador de modelo RISCV e executar um arquivo contendo instruções binárias, respeitando suas estruturas.

O trabalho foi finalizado com sucesso, passando em cada um dos casos de teste. Vale notar a facilidade com a qual a linguagem C++ permite manipular os bits de seus argumentos inteiros.