Simulador RISC-V

Por: Guilherme Da Hora Andrade Fontoura,
Amanda Lemos Ribas,
Maria Eduarda D’Angelo Quitete Vianna,
Lais Ferreira Nazareth

Resumo

O projeto consiste em um Simulador/Assembler da arquitetura RISC-V com implementação de pipeline.

Ele é capaz de receber um arquivo .txt (código binário gerado pelo RARS) ou .asm e executá-lo, mostrando as instruções em seus respectivos estágios no pipeline em uma interface gráfica.

Funcionalidades

- Receber arquivo .asm ou .txt e executar as instruções
- Implementação de pipeline
- Tratamento de hazards
- Interface gráfica que permite analisar a execução do arquivo “step by step”

Tecnologias Usadas

Estrutura de Diretório

- **Assembler Risc-V**
  - **arquivosexemplos**
  - **interface**
  - **pipeline**
    - InstructionFetch.py
    - InstructionDecode.py
    - Execute.py
    - MemoryAccess.py
    - WriteBack.py
    - RunPipeline.py
    - DeteeccaoHazards.py
  - **riscv_data**
    - registers.py
    - instructions.py
  - main.py

Implementação Pipeline

Instruction Fetch (para .asm)

- Quebra a instrução com split
- Usa o mnemônico para buscar informações da instrução em instructions.py
- Define rd, rs1, rs2 e immediate dependendo do tipo da instrução e mnemônico

Implementação Pipeline

Instruction Fetch (para .txt)

- Busca opcode para determinar tipo
- Analisa funct3 e funct7 para determinar instrução
- Lê rd, rs1, rs2 e immediate

Implementação Pipeline

Instruction Decode

- Recebe a instrução do Instruction Fetch
- Analisa o tipo da instrução
- Busca registradores utilizados no banco de registradores
- Caso use imediato, carrega valor para o campo correto

Implementação Pipeline

Execute

- Recebe lista contendo dados da instrução e dos registradores de Instruction Decode
- Executa a instrução
- Retorna lista com tipo da instrução, mnemônico, address(rd) e operandos da instrução

Implementação Pipeline

Memory Access

- Recebe lista do Execute
- Se for lw, retorna o valor que está na posição da memória
- Se for sw, salva o valor de rd na posição da memória

Implementação Pipeline

Write Back

- Guarda o sinal de controle memToReg
- Verifica o tipo da instrução e, se for válido, guarda o valor a escrever no registrador de destino

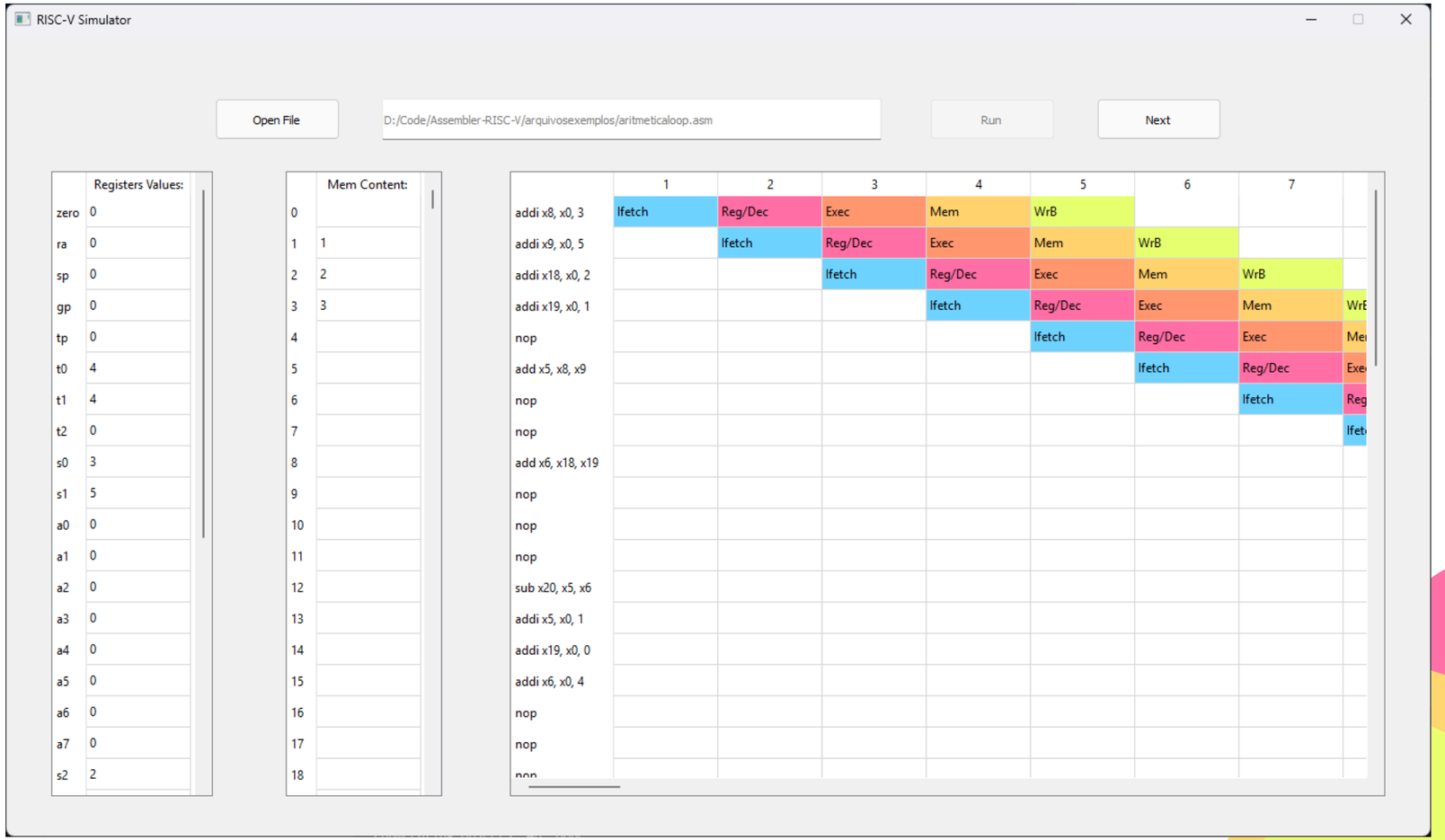
Registradores e Instruções

- São armazenados na pasta riscv_data
- Classe registradores e classe instruções
- Dicionários para cada classe contendo valores dos registradores e informações das instruções

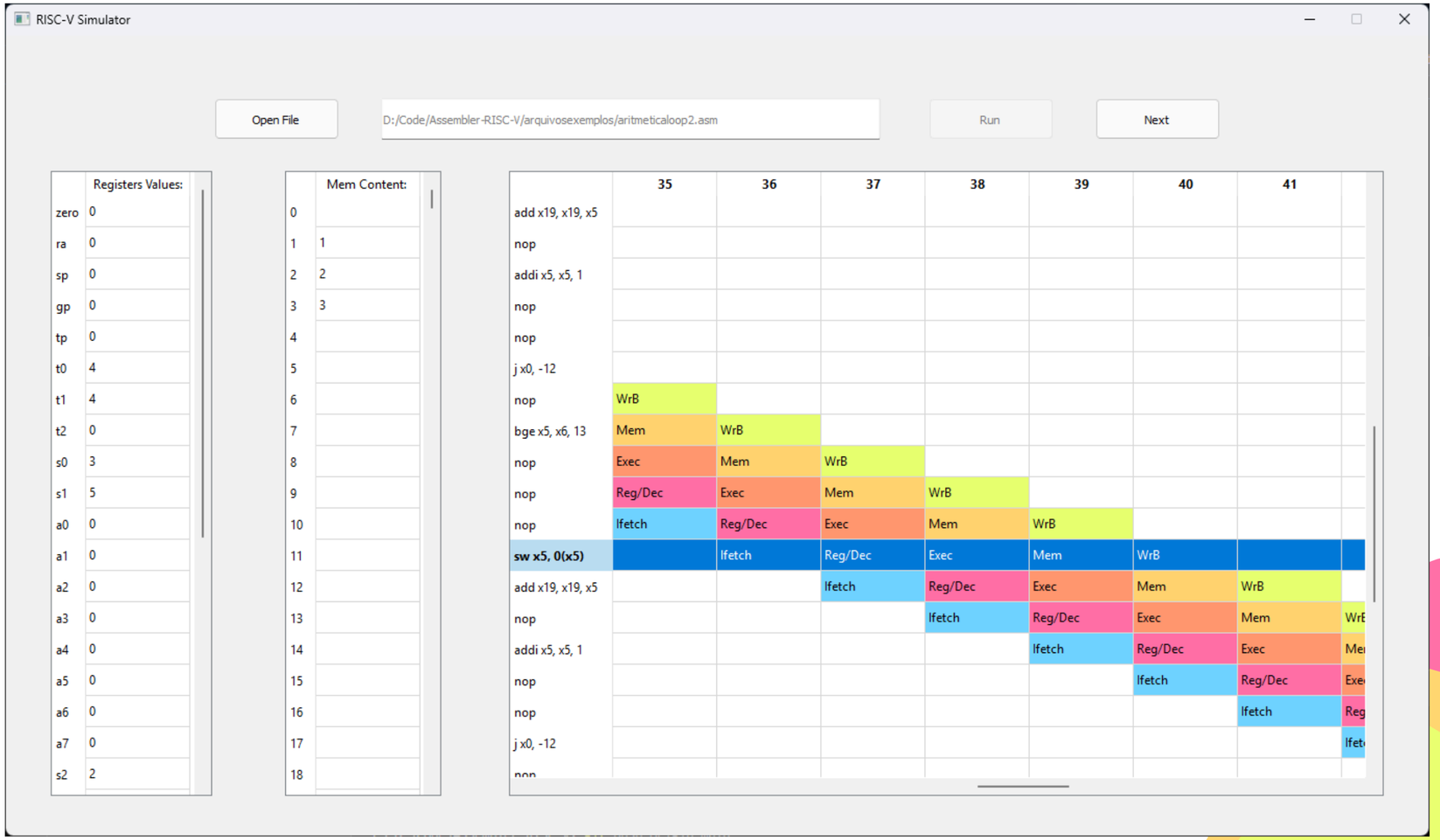
Detecção de Hazards

- Procura identificar hazards para fazer o tratamento posteriormente, entretanto não foi possível concluir a implementação. O rascunho da implementação pode ser encontrado em DeteccaoHazards.py

Interface


RISC-V Simulator
Open File
D:/Code/Assembler-RISC-V/arquivosexemplos/aritmeticaloop.asm
Run
Next
Registers Values:
zero 0
ra 0
sp 0
gp 0
tp 0
t0 4
t1 4
t2 0
s0 3
s1 5
a0 0
a1 0
a2 0
a3 0
a4 0
a5 0
a6 0
a7 0
s2 2
Mem Content:
0
1 1
2 2
3 3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
1 2 3 4 5 6 7
addi x8, x0, 3 Ifetch Reg/Dec Exec Mem WrB
addi x9, x0, 5 Ifetch Reg/Dec Exec Mem WrB
addi x18, x0, 2 Ifetch Reg/Dec Exec Mem WrB
addi x19, x0, 1 Ifetch Reg/Dec Exec Mem
nop Ifetch Reg/Dec Exec
add x5, x8, x9 Ifetch Reg/Dec
nop Ifetch
nop
add x6, x18, x19
nop
nop
nop
sub x20, x5, x6
addi x5, x0, 1
addi x19, x0, 0
addi x6, x0, 4
nop
nop

Interface


RISC-V Simulator
Open File
D:/Code/Assembler-RISC-V/arquivosexemplos/aritmeticaloop2.asm
Run
Next
Registers Values:
zero 0
ra 0
sp 0
gp 0
tp 0
t0 4
t1 4
t2 0
s0 3
s1 5
a0 0
a1 0
a2 0
a3 0
a4 0
a5 0
a6 0
a7 0
s2 2
Mem Content:
0
1 1
2 2
3 3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
35 36 37 38 39 40 41
add x19, x19, x5
nop
addi x5, x5, 1
nop
nop
j x0, -12
nop WrB
bge x5, x6, 13 Mem WrB
nop Exec Mem WrB
nop Reg/Dec Exec Mem WrB
nop Ifetch Reg/Dec Exec Mem WrB
sw x5, 0(x5) Ifetch Reg/Dec Exec Mem WrB
add x19, x19, x5 Ifetch Reg/Dec Exec Mem WrB
nop Ifetch Reg/Dec Exec Mem
addi x5, x5, 1 Ifetch Reg/Dec Exec
nop Ifetch Reg/Dec
nop Ifetch
j x0, -12
nop

Obrigado!