Implementarea unui procesor RISC-V în Verilog

Dancău Rareș-Andrei

Cuprins

Ce este RISC-V?

- RISC-V este o arhitectură de procesoare cu sursă deschisă (open-source) bazată pe un set redus de instrucțiuni (RISC).
- Este flexibilă și permite optimizarea proiectării procesoarelor, putând fi implementate de la microcontrolere până la unități de procesare din supercalculatoare. Din acest motiv este adoptată în industrie și în mediul academic.
- RV64I este un modul al arhitecturii ce definește instrucțiunile de bază pentru lucrul cu regiștrii de 64 biți, extinzând RV32I, setul de instrucțiuni pentru regiștrii de 32 de biți.

De ce RISC-V?

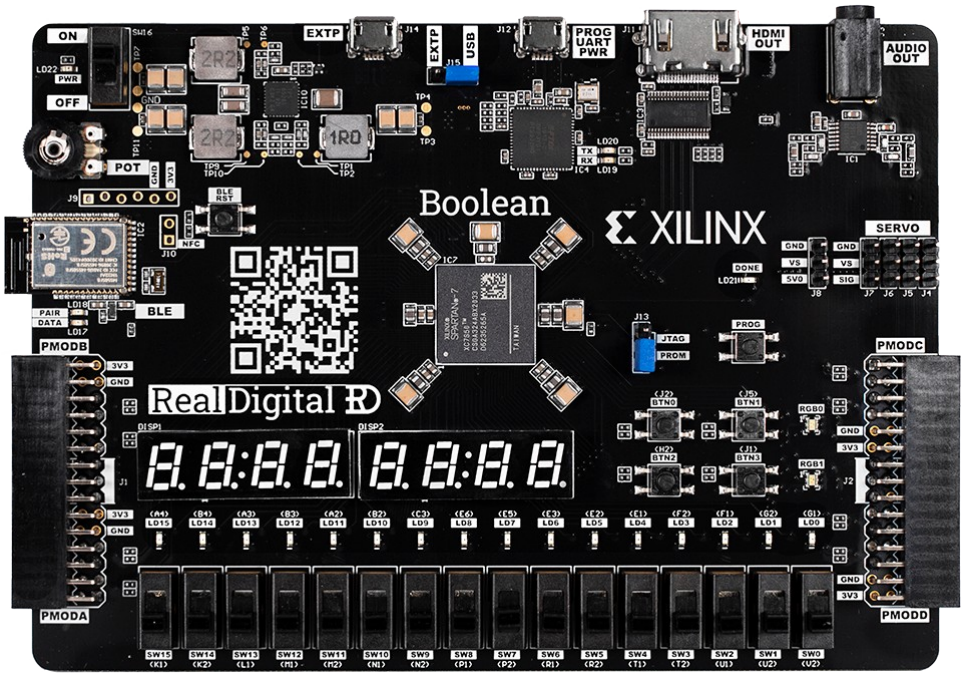
- Procesoarele bazate pe RISC-V sunt mai eficiente d.p.d.v. energetic și au o structură hardware miniaturizată comparativ cu procesoarele bazate pe arhitecturi CISC.
- Arhitectura este modulară, putând fi implementate o serie de extensii cum ar fi extensiile F și D ce permit lucrul cu numere cu virgulă.
- Permite dezvoltarea de procesoare sub o licență fără drepturi de autor (royalty-free).
- Se poate face o echivalență între sistemele de operare și producătorii de procesoare în funcție de principii.

Platforma hardware a proiectului

- Din punct de vedere al componentelor hardware, au fost folosite:
  - Placa FPGA Boolean Board;
  - Placa Raspberry Pi Pico H montată pe un breadboard împreună cu:
    - 16 leduri; 
    - 3 circuite ULN2003A; 
    - cabluri de conectare. 

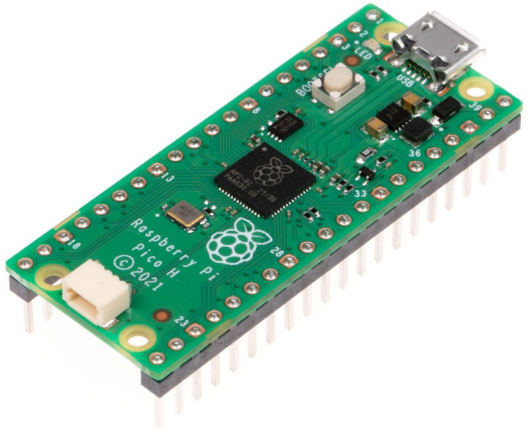
Boolean Board

- Este o platformă educațională bazată pe FPGA ce conține un număr considerabil de periferice, cum ar fi leduri, comutatoare și butoane.
- Poate fi programată folosind suita CAD Vivado de la AMD și limbajul HDL Verilog.



Raspberry Pi Pico H

- Este o placă bazată pe microcontrolerul RP2040 ce poate fi folosită în diverse proiecte DIY.
- Programarea acesteia se face folosind C sau MicroPython.



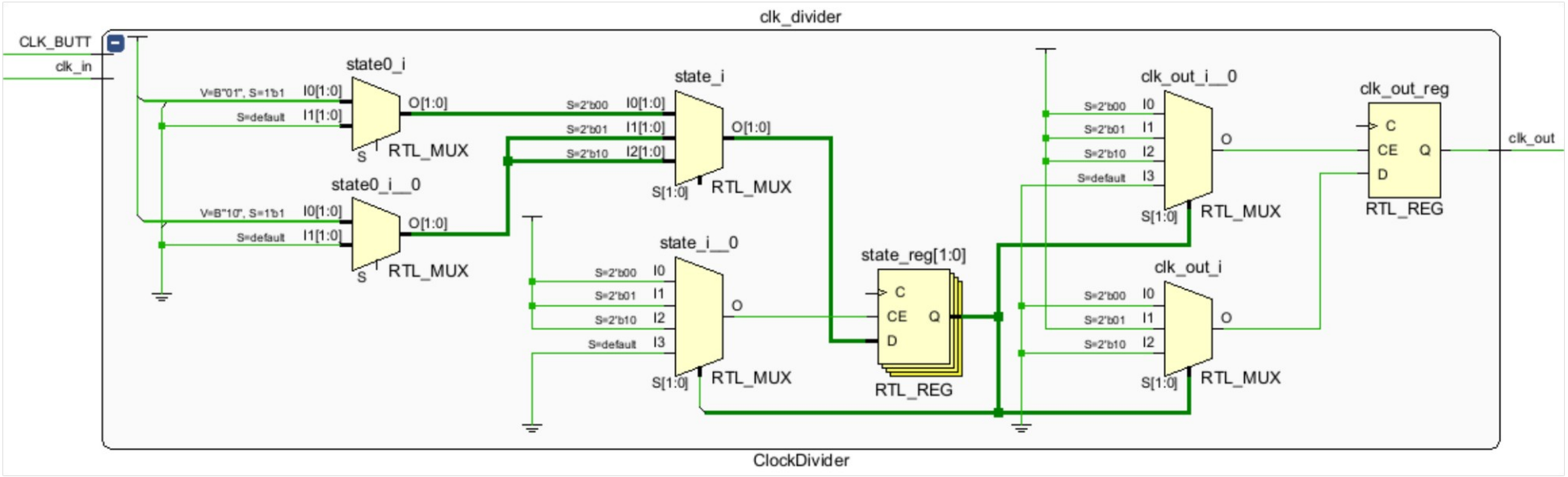
Platforma software a proiectului

- Din punct de vedere al componentelor software, acestea au fost dezvoltate utilizând următoarele limbaje de programare:
  - C# cu framework-ul WinForms;
  - Python cu framework-ul Qt;
  - MicroPython.

Implementarea procesorului

- Procesorul a fost implementat folosind Verilog și arhitectura sa se bazează pe două module principale.
- Modulul ClockDivider are rolul de a gestiona semnalul clock folosit de procesor. Pentru a face acest lucru în cadrul lui se folosește un semnal controlat de utilizator și un registru de stare.
- Modulul RISC-VProcessor gestionează modul de executare al instrucțiunilor și salvarea rezultatelor în regiștrii și memoriile aferente.
- Procesorul a fost simulat și testat folosind EDA Playground, iar programarea pe Boolean Board a fost făcută cu Vivado 2023.1.

Modulul ClockDivider



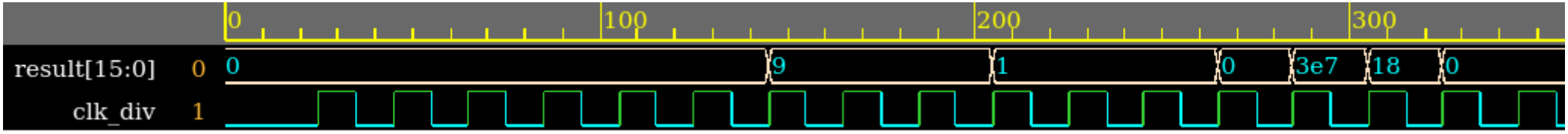
Modulul RISC-VProcessor

- Are o structură complexă și în cadrul acestuia au loc mai multe etape:
  - Etapa de inițializare a regiștrilor și memoriilor;
  - Instanțierea modulului ClockDivider;
  - Efectuarea următoarelor operații la apariția unui front pozitiv pentru semnalul ceas de referință:
    - Verificarea valorii semnalului reset și reinițializarea memoriei RAM și a regiștrilor dacă semnalul este activ;
    - Dacă nu este activ, se trece în etapa de pipeline cu cinci sub-etape: fetch, decode, execute, memory și write-back.

Rezultatele obținute pentru program_paritate.s

- În urma executării pe procesor a programului, se afișează pe leduri valorile:

```
addi a0, zero, 9
andi t0, a0, 1
beq t0, zero, 12
addi t0, zero, 999
jal t1, 8
addi t0, zero, 888
jalr ra, 0(zero)
```

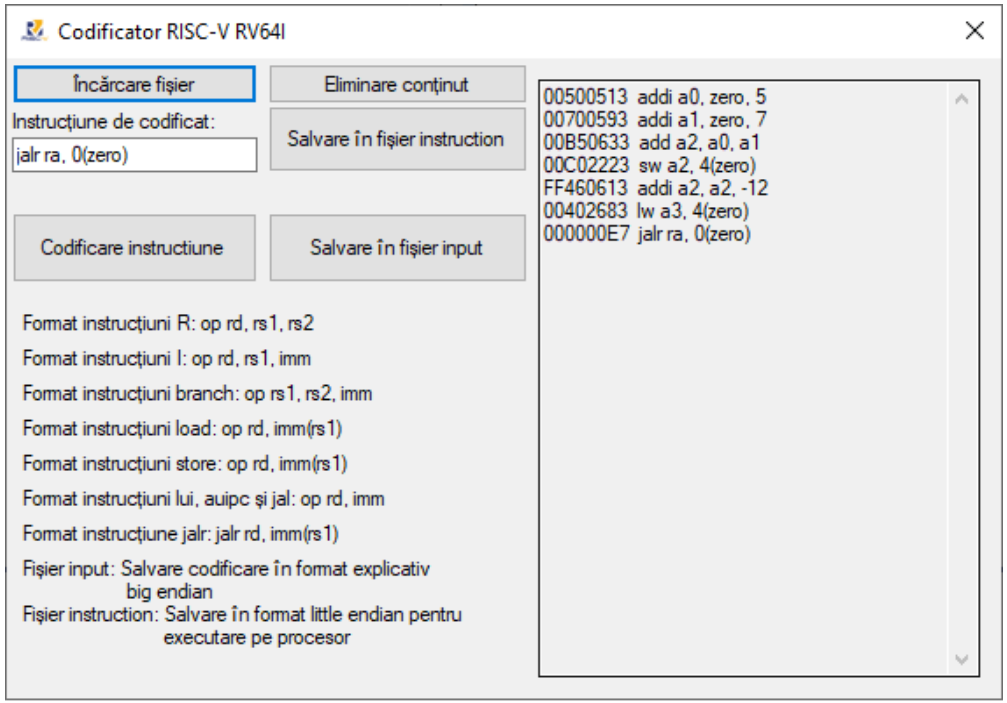


Dezvoltarea codificatoarelor

- În cadrul procesului de dezvoltare și testare a procesorului a fost necesară implementarea codificatoarelor pentru a transforma fișiere cu instrucțiuni în asamblare în forma lor executabilă pe procesor.
- Pentru a codifica instrucțiunile au fost create două codificatoare:
  - Codificatorul scris în C# a fost primul dezvoltat și folosit pentru codificarea instrucțiunilor.
  - Codificatorul scris în Python cu Qt a fost implementat pentru a permite codificarea pe mai multe sisteme de operare.

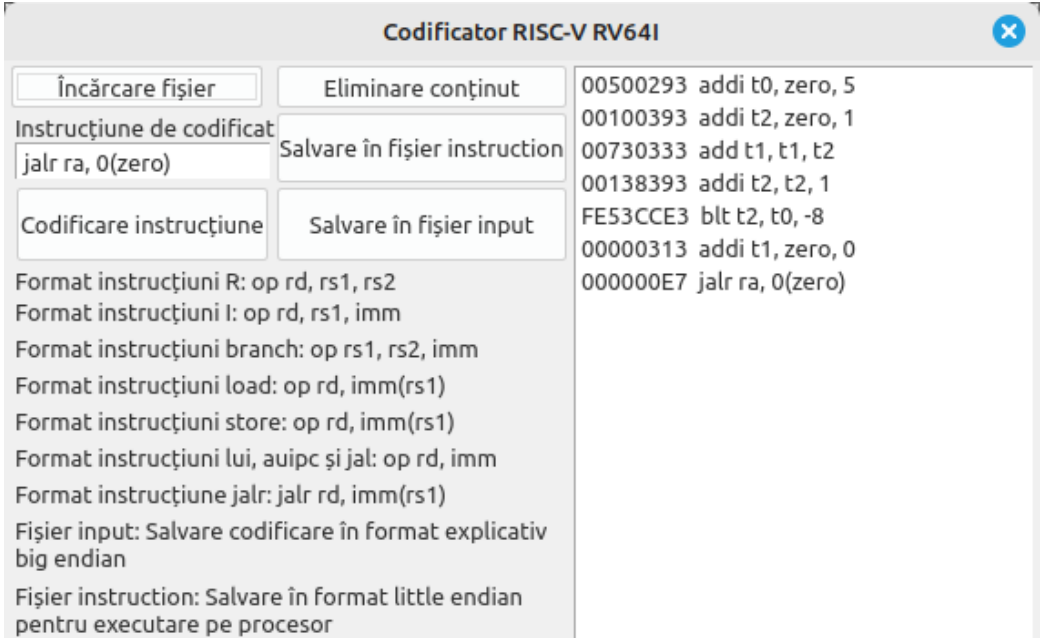
Codificatorul C#

- A fost implementat folosind .NET Framework și permite citirea fișierelor cu instrucțiuni în asamblare, dar și salvarea instrucțiunilor în două formate: fișier de tip instruction ce poate fi executat direct pe procesor și fișier input în care se salvează conținutul casetei de text.



Codificatorul Python

- A fost dezvoltat folosind framework-ul Qt pentru interfața grafică și are caracteristici similare cu programul implementat în C#, dar poate fi executat pe mai multe sisteme de operare.

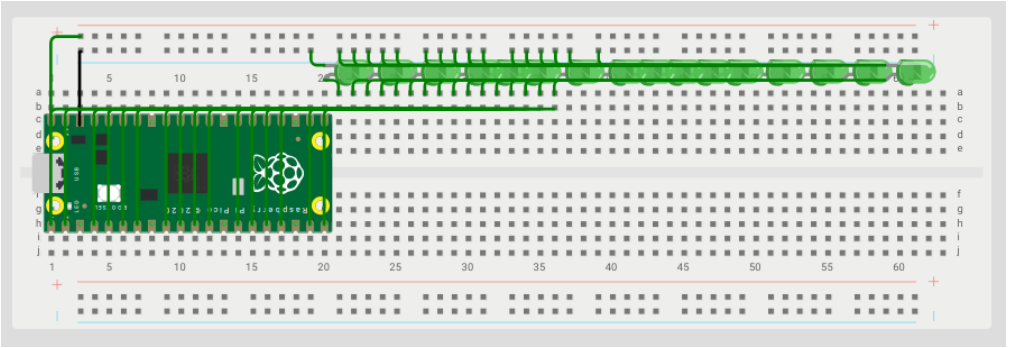


Rezultatele codificării pentru program_counter.s

program_counter.s	input_counter.txt	instructions(counter).mem
addi t1, zero, 10	00A00313 addi t1, zero, 10	13
addi t2, zero, 1	00100393 addi t2, zero, 1	03
sub t1, t1, t2	40730333 t1, t1, t2	A0
bne t1, zero, -4	FE031EE3 bne t1, zero, -4	00
jalr ra, 0(zero)	000000E7 jalr ra, 0(zero)	93
		03
		10
		00
		33
		[...]

Emulatorul procesorului pe RPi Pico

- Pentru a executa programele într-un mod diferit de implementarea pe FPGA a procesorului s-a dezvoltat un emulator folosind un Raspberry Pi Pico și limbajul MicroPython.
- Acest emulator se comportă într-un mod asemănător procesorului, dar execuția instrucțiunilor nu se face în pipeline.



- În schemă se prezintă simplificat modul de conectare al componentelor, fără a mai include circuitele ULN2003A. Pinii GPIO0-GPIO15 sunt conectați la leduri.

Rezultatele emulatorului pentru program_paritate.s

addi a0, zero, 9 -> 0x009

andi t0, a0, 1 -> 0x001

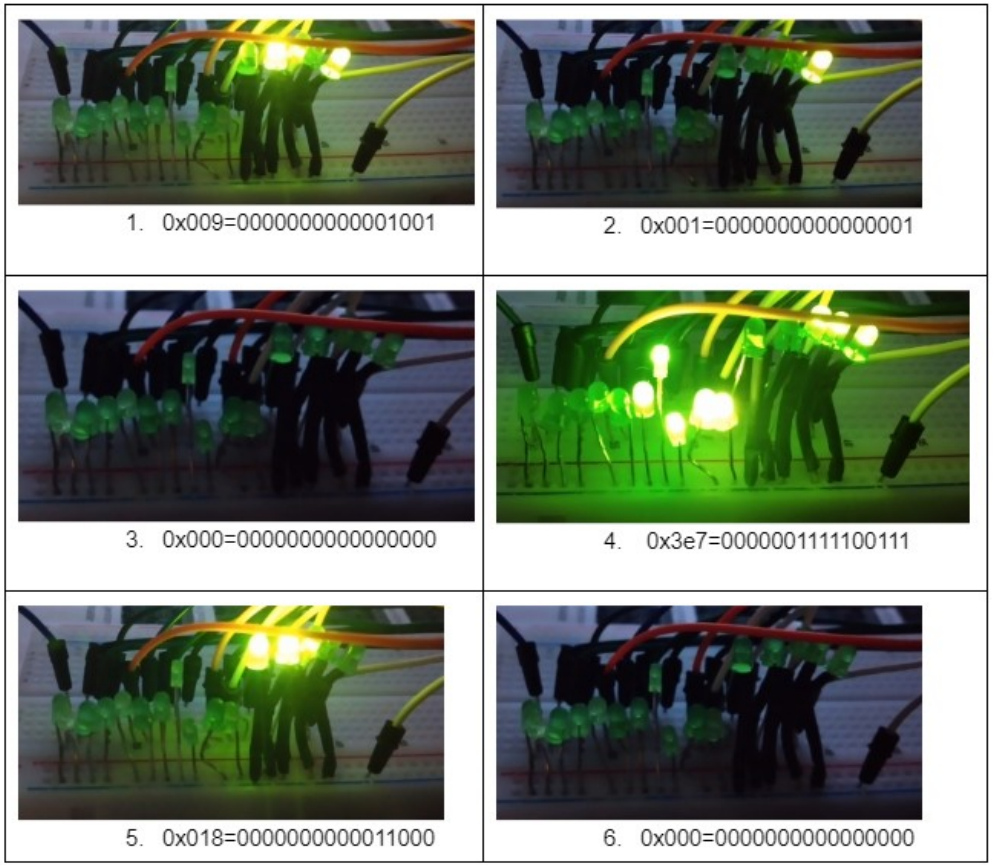
beq t0, zero, 12 -> 0x000

addi t0, zero, 999 -> 0x3e7

jal t1, 8 -> 0x018

addi t0, zero, 888

jalr ra, 0(zero) -> 0x000



1. 0x009=0000000000001001
2. 0x001=0000000000000001
3. 0x000=0000000000000000
4. 0x3e7=0000001111100111
5. 0x018=0000000000011000
6. 0x000=0000000000000000

Concluzii

- Prin prezentul proiect s-a demonstrat posibilitatea implementării unui procesor RISC-V pe Boolean Board.
- Aplicațiile implementate în cadrul lucrării pot constitui o bază robustă pentru viitoare proiecte din domeniul arhitecturii calculatoarelor.
- Sunt posibile multiple direcții de dezvoltare pentru fiecare componentă a proiectului.

Posibile direcții de dezvoltare

- Câteva idei viitoare de dezvoltare a procesorului includ:
  - Adăugarea implementării altor instrucțiuni din setul de instrucțiuni RV64I, dar și din alte extensii ale arhitecturii RISC-V.
  - Îmbunătățirea modului de execuție al instrucțiunilor în pipeline.
  - Dezvoltarea unui sistem de operare ce poate fi executat pe procesor.
- Pentru codificatoare se poate adăuga posibilitatea codificării altor instrucțiuni.
- Pentru emulator se poate efectua execuția în pipeline a instrucțiunilor.

Vă mulțumesc!

Bibliografie imagini

- Logo RISC-V: https://commons.wikimedia.org/wiki/File:RISC-V-logo-square.svg
- Led verde: https://ro.farnell.com/productimages/standard/en_GB/1581138-40.jpg
- ULN2003A: https://zutech.ro/8716-medium_default/uln2003apg.jpg
- Cabluri conectare: https://zutech.ro/2277-medium_default/set-65-cabluri-conexiune-breadboard.jpg
- Boolean Board: https://www.realdigital.org/img/1bfa8a56e075cdc92c0b03204c6c5618.png
- Raspberry Pi Pico H: https://www.kiwi-electronics.com/image/cache/catalog/product/83habfak/KW-3018-1-1920x1280.jpg