# Proiect: Implementarea Procesorului MIPS32

## 1. Introducere

Acest proiect are ca scop implementarea și testarea unui procesor MIPS32 folosind VHDL. Proiectul acoperă dezvoltarea și testarea diferitelor etape ale unui procesor cu ciclu unic, inclusiv extragerea instrucțiunilor (IFetch), decodificarea instrucțiunilor (ID), execuția instrucțiunilor (EX), accesul la memorie (MEM) și scrierea rezultatelor în registre (WB).

## 2. Obiective

1. Extragerea Instrucțiunii (IFetch):  
- Implementarea unui contor de program (PC) care să gestioneze secvențialitatea instrucțiunilor.  
- Implementarea unei memorii ROM care conține instrucțiunile de test.  
  
2. Decodificarea Instrucțiunii (ID):  
- Implementarea unui bloc de registre (RegFile) pentru stocarea temporară a datelor.  
- Extinderea imediatului (Ext\_Imm) la 32 de biți în funcție de semnul său.  
  
3. Execuția Instrucțiunii (EX):  
- Implementarea unei unități aritmetice și logice (ALU) capabile să efectueze operații de bază.  
- Generarea adreselor pentru salturi condiționate și necondiționate.  
  
4. Accesul la Memorie (MEM):  
- Implementarea unei memorii RAM pentru stocarea și recuperarea datelor.  
- Gestionarea accesului asincron la date.  
  
5. Scrierea Rezultatelor (WB):  
- Implementarea unui multiplexor care selectează sursa datelor pentru scrierea în registre.

## 3. Structura Generală

### 3.1 Extragerea Instrucțiunii (IFetch)

Unitatea IFetch se ocupă cu extragerea instrucțiunilor din memoria ROM. Aceasta include un contor de program (PC) care se incrementează pentru a parcurge instrucțiunile secvențial.  
  
- Componenta PC: Un registru pe 32 de biți care se incrementează la fiecare ciclu de ceas sau se resetează la apăsarea unui buton.  
- Memoria ROM: Conține instrucțiunile de test în format binar.

### 3.2 Decodificarea Instrucțiunii (ID)

Unitatea ID decodifică instrucțiunea curentă și extrage operanzii necesari din blocul de registre (RegFile).  
  
- RegFile: Un set de 32 de registre pe 32 de biți fiecare.  
- Ext\_Imm: Extinderea imediatului pe 16 biți la 32 de biți, cu semn sau fără semn, în funcție de instrucțiune.

### 3.3 Execuția Instrucțiunii (EX)

Unitatea EX efectuează operațiile aritmetice și logice necesare instrucțiunii.  
  
- ALU: Capabilă să efectueze operații precum adunare, scădere, deplasare la stânga și comparare.  
- Generarea Adreselor: Calcularea adreselor pentru salturile condiționate și necondiționate.

### 3.4 Accesul la Memorie (MEM)

Unitatea MEM gestionează accesul la memoria de date (RAM).  
  
- Memoria RAM: Stocarea datelor temporare necesare programului.  
- Acces Asincron: Citirea datelor din memorie fără a aștepta un semnal de ceas.

### 3.5 Scrierea Rezultatelor (WB)

Unitatea WB scrie rezultatele finale în blocul de registre.  
  
- Multiplexor: Selectează între datele provenite din ALU și datele din memorie pentru a fi scrise în registre.

## 4. Procedura de Testare

1. Inițializare: Configurați memoria ROM cu instrucțiunile de test.  
2. Extragerea Instrucțiunilor: Asigurați-vă că PC-ul extrage corect instrucțiunile din ROM.  
3. Decodificare și Execuție: Verificați corectitudinea decodificării instrucțiunilor și execuția operațiilor ALU.  
4. Accesul la Memorie: Testați citirea și scrierea datelor în RAM.  
5. Scrierea Rezultatelor: Asigurați-vă că rezultatele corecte sunt scrise în registrele corespunzătoare.

## 5. Elemente Funcționale și Ne-funcționale

Elemente Funcționale:  
- Implementarea corectă a contorului de program și a memoriei ROM pentru extragerea instrucțiunilor.  
- Decodificarea precisă a instrucțiunilor și manipularea registrelor.  
- Execuția corectă a operațiilor aritmetice și logice.  
- Gestionarea eficientă a accesului la memorie și scrierea rezultatelor.  
  
Elemente Ne-funcționale:  
- Optimizarea performanței procesorului pentru execuție rapidă.  
- Minimizarea consumului de energie.  
- Scalabilitatea arhitecturii pentru a suporta instrucțiuni suplimentare.

## 6. Probleme Întâmpinate

În timpul implementării procesorului MIPS32, au fost întâmpinate următoarele probleme:  
  
- Sincronizarea corectă a componentelor pentru a asigura execuția instrucțiunilor fără conflicte de date.  
- Gestionarea corectă a semnalelor de control pentru multiplexoare și unități aritmetice.  
- Depanarea erorilor de cod VHDL pentru a asigura funcționarea corectă a tuturor componentelor.

## 7. Testarea pe Placă și/sau Simulator

Procesorul MIPS32 a fost testat utilizând un simulator de VHDL pentru a verifica corectitudinea execuției instrucțiunilor. Nu a fost testat pe hardware fizic din lipsa resurselor necesare, dar simulările au arătat că procesorul funcționează conform specificațiilor.

## 8. Modificări Particulare pentru Instrucțiuni Suplimentare

Pentru a implementa cele 4 instrucțiuni suplimentare, au fost necesare următoarele modificări:  
  
- Adăugarea unor cazuri suplimentare în unitatea de control pentru a recunoaște noile opcode-uri.  
- Extinderea unității ALU pentru a suporta noile operații specifice instrucțiunilor adăugate.  
- Actualizarea logicii de decodificare pentru a extrage corect operanzii necesari pentru noile instrucțiuni.  
- Modificarea unității de acces la memorie pentru a gestiona noile tipuri de acces cerute de noile instrucțiuni.