Laborator 05

Procesorul MIPS – versiune pe 16 biți, cu un ciclu de ceas pe instrucțiune

Unitatea de Instruction Fetch - IF

0. Resurse minimale necesare !

Din laboratorul 4 (terminat acasă): Memoria ROM, conținând programul definit în cod mașină de fiecare, cu comentariu pe fiecare linie cu instrucțiunea în asamblare.

Programul scris trebuie să aibă o logică de la început la sfârșit: să existe o ordine de procesare a unor regiștri, lucru cu memoria, salturi condiționate / necondiționate, toate aceste instrucțiuni consecutive fiind legate una de alta prin anumiți operanzi comuni.

Atenție, lipsa acestei memorii cu programul codificat, respectiv a temelor/activităților anterioare, se va sancționa drastic (nota 1).

1. Obiective

Studiul, proiectarea, implementarea și testarea:

- **Unității de IF pentru procesorul MIPS, pe 16 biți, un ciclu de ceas / instrucțiune (single-cycle)**

2. Descrierea procesorului MIPS, simplificat pe 16 biți - continuare

(!) Lectura obligatorie a cursurilor 3 și 4 pentru a înțelege activitățile din acest laborator.

Ciclul de execuție a unei instrucțiuni MIPS are următoarele etape / faze (curs 4):

- IF – Instruction Fetch
- ID/OF – Instruction Decode / Operand Fetch
- EX – Execute
- MEM – Memory
- WB – Write Back

Implementarea proprie a procesorului MIPS-16, pe care o începeți în acest laborator (și o veți termina în următoarele) va fi partiționată în 5 componente (entități noi). Aceste componente se vor declara și instanția în proiectul test_env, în entitatea principală (test_env, probabil…).

Partiționarea procesorului în entități asociate cu etapele de execuție nu prezintă beneficii explicite pentru procesorul MIPS 16 cu ciclu unic. Utilitatea acestei partiționări o veți înțelege în laboratoarele viitoare, când se va implementa versiunea pipeline a procesorului MIPS 16!

În acest laborator veți proiecta, descrie în VHDL, și implementa / testa unitatea de aducere a instrucțiunii Instruction Fetch – IF, pentru versiunea proprie a procesorului MIPS 16 cu ciclu unic de ceas pe instrucțiune.

Căile de date ale procesorului MIPS (versiunea 32 de biți) sunt prezentate mai jos, împreună cu unitatea de control (și semnalele aferente). Pentru a evita aglomerarea schemei, semnalele de control nu au mai fost legate explicit la destinații, dar se pot identifica ușor după nume.

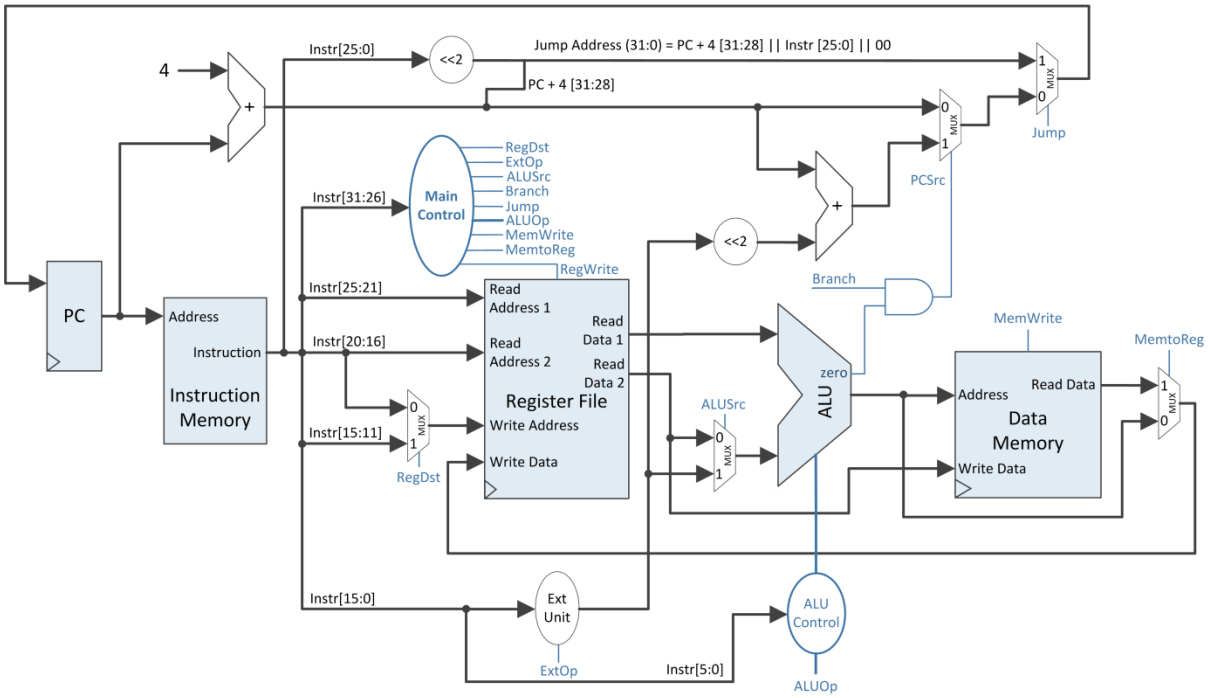

Figura 1: MIPS 32 cu ciclu unic de ceas

Mai jos sunt revizitate cele 3 formate de instrucțiuni pentru MIPS 16, descrise în laboratorul anterior:

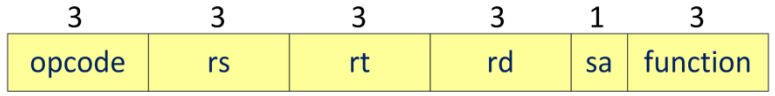

Figura 2: Instrucțiune de tip R

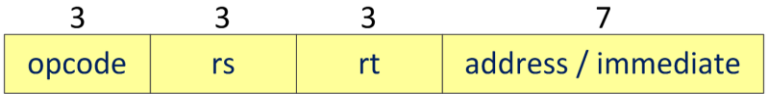

Figura 3: Instrucțiune de tip I

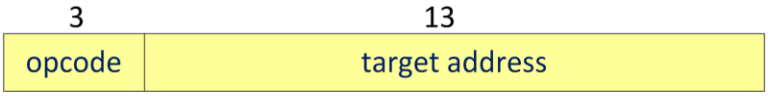

Figura 4: Instrucțiune de tip J

Unitatea IF conține următoarele elemente principale (nu se vor descrie suplimentar entități!):

- Program Counter
- Instruction Memory (ROM)
- Adder

În plus, mai există doua MUX-uri pentru stabilirea adresei următoare. Consultați laboratorul anterior pentru caracteristicile acestor componente pentru MIPS 16. Calea de date pentru unitatea IF, MIPS 32, este prezentată mai jos.

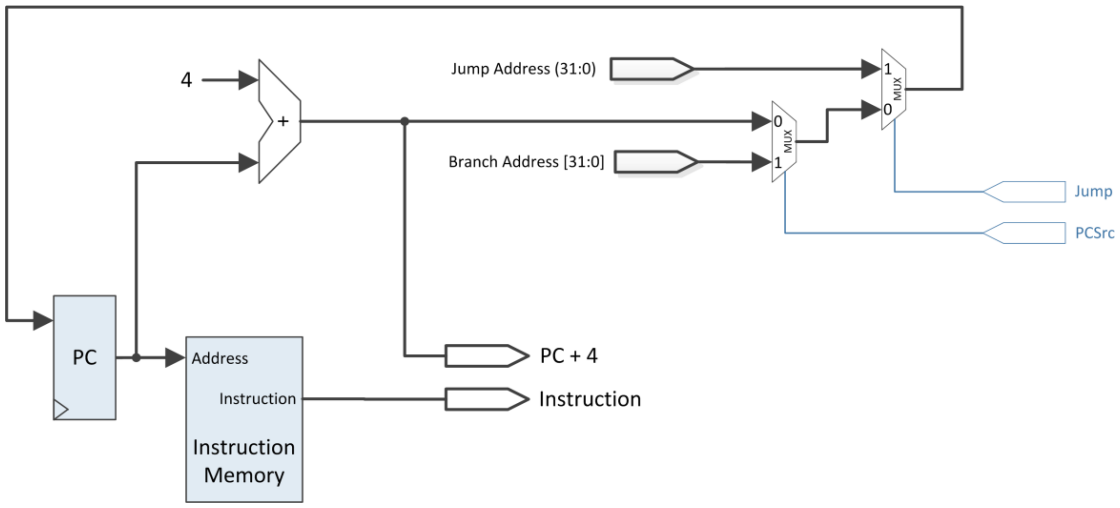

Figura 5: Calea de date pentru IF din procesorul MIPS 32

Ca ieșiri, unitatea IF furnizează instrucțiunea curentă și adresa următoarei instrucțiuni, pentru execuție secvențială. În cazul instrucțiunilor jump și branch, unitatea IF primește ca intrări adresa de salt pentru jump, respectiv adresa țintă pentru branch,

împreună cu semnalele de control care vor selecta adresa următoarei instrucțiuni care se va executa (noua valoare a PC).

Intrările unității IF sunt:

- Semnalul de ceas (pentru …. PC)
- Adresa de branch
- Adresa de jump
- Semnalul de control Jump
- Semnalul de control PCSrc (pentru branch)

Ieșirile unității IF sunt:

- Instrucțiunea de executat în ciclul de ceas curent, pe procesorul MIPS
- Adresa următoarei instrucțiuni de executat, mod secvențial (PC + 4)

Semnificația semnalelor de control:

- Jump = 1 → PC ← jump address
- Jump = 0 → PC ← (PC + 4 dacă PCSrc = 0 sau adresa de branch dacă PCSrc = 1)

3. Activități practice

Citiți fiecare activitate în întregime, înainte să o începeți practic!

Resurse necesare (de avut la începerea laboratorului!):

- Rezultatele obținute din activitățile din laboratorul 3,
- 15 instrucțiuni definite pentru propria voastră implementare a procesorului MIPS 16 (Lab 4, activitatea 3.1),
- RTL abstract / formatul celor 15 instrucțiuni, scrise pe hârtie,
- Calea de date pentru MIPS 16 hârtie / instrument de scris (lab 4, act. 3.3, folosiți figura 1 din acest laborator ca referință, plus cursul 4 pentru detalii),
- Proiect Xilinx cu test_env, care să includă cel puțin memoria ROM (va juca rolul memoriei de instrucțiuni), inițializată la declararea semnalului cu programul vostru personalizat, scris în cod mașină (lab 4, act. 3.2).

3.1. Proiectarea / implementarea unității IF

Ținând cont de descrierea unității IF din secțiunea 2, figura 5, descrieți o nouă componentă (entitate) pentru IF în proiectul test_env, ca prima unitate implementată a procesorului MIPS 16. Toate câmpurile de date sunt pe 16 biți!

Entitatea IF va conține elementele descrise în figura 5, care nu se vor descrie cu entități suplimentare!

Memoria de instrucțiuni va fi memoria ROM din laboratorul anterior, cu programul scris în cod mașină. (!) Nu măriți dimensiunea memoriei ROM (la 2^{16} locații), ci folosiți mai puțini biți din cei 16 ai PC pentru adresarea ROM (cei mai puțin semnificativi).

Sumatorul se va descrie cu +1 în VHDL (nu +2 pentru MIPS 16), deoarece în VHDL memoria ROM ați declarat-o având cuvântul pe 16 biți (în loc de 8). De aici încolo vom referi ieșirea PC + 4 ca PC + 1 pentru varianta pe 16 biți. *Dacă afli pentru prima data informația asta…e cazul să mai dai pe la curs…unde să fii atent!*

Contorul de program, registrul PC, va fi un registru (bistabil D, 16 biți) pe front crescător.

Atenție! Pentru a avea un control al circuitului la testarea pe placă, scrierea pe front de ceas a lui PC cu noua valoare se va face doar la apăsarea unui buton de pe placa Basys. Folosiți un semnal de enable de la MPG, ca intrare în entitatea IF, pentru a valida scrierea în PC. În plus, folosiți un al doilea buton (altă instanță a MPG, a doua intrare de enable în entitatea IF) pentru a reseta registrul PC la valoarea zero (se va reveni ușor la prima instrucțiune din ROM în timpul testării).

3.2. Testarea unității IF

În entitatea test_env declarați și instanțiați unitatea IF. Conectați unitatea IF împreună cu MPG (2 instanțe / enable) și cu afișorul SSD disponibile deja în proiectul test_env.

Pentru conectarea cu SSD, se vor afișa ambele ieșiri din IF (instrucțiunea și PC + 1) folosind un mecanism de multiplexare. Folosiți switch-ul sw(7) pentru selecția multiplexorului:

- sw(7) = 0 → se afișează instrucțiunea pe SSD,
- sw(7) = 1 → se afișează următoare valoare secvențială a PC, și anume PC + 1, pe SSD.

Folosiți cele două butoane care trec prin MPG pentru a reseta, respectiv a controla scrierea în registrul PC. Veți simula astfel fluxul normal, secvențial, de execuție a instrucțiunilor.

Pentru testarea salturilor se vor simula astfel de salturi, prin maparea pe semnalele de control care intră în IF a două switch-uri:

- Folosiți sw(0) pentru semnalul de control Jump,
- Folosiți sw(1) pentru semnalul de control PCSrc.

De asemenea, deoarece elementele care calculează adresele de salt se vor descrie doar în laboratoarele viitoare, folosiți valori “hard-coded” în VHDL pe care le veți mapa pe intrările Jump address și Branch address ale componentei IF:

- Exemplu: puteți folosi x”0000” pentru jump ca un mecanism alternativ de resetare a PC (salt la prima instrucțiune),
- Folosiți o valoare intermediară pentru adresa de branch (această valoare trebuie să fie adresa unei instrucțiuni din interiorul programului existent în memoria ROM de instrucțiuni, codificat binar).

4. Referințe

- Cursurile 3 & 4 de la Arhitectura Calculatoarelor.
- MIPS® Architecture For Programmers, Volume I-A: Introduction to the MIPS32® Architecture, Document Number: MD00082, Revision 5.01, December 15, 2012
- MIPS® Architecture For Programmers Volume II-A: The MIPS32® Instruction Set Manual, Revision 6.02
- MIPS32® Architecture for Programmers Volume IV-a: The MIPS16e™ Application-Specific Extension to the MIPS32™ Architecture, Revision 2.62.
    - Chapter 3: The MIPS16e™ Application-Specific Extension to the MIPS32® Architecture.