Laborator 08

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

Evaluarea/testare a procesorului

0. Resurse minimale necesare !

Din laboratorul 7: Proiectul test_env, cu toate unitățile procesorului complete, legate, cu mecanismul de testare, fără erori de sintaxă. Verificați pe RTL schematic în VIVADO corectitudinea schemei.

Din laboratoarele anterioare: toate materialele scrise pe foaie.

De făcut acasă (ușurează mult testarea pe placă): trasarea programului pe hârtie, dacă există un ciclu repetitiv – cel puțin o iterație. Pentru fiecare instrucțiune trasată se notează pe hârtie operanzii sursă, rezultatul, valorile semnalelor interne relevante (RD1, RD2, ALURes, Branch Address, etc.) care se pot afișa pe SSD.

1. Testarea propriului procesor MIPS 16

În acest moment trebuie să aveți procesorul MIPS 16 implementat complet în proiectul test_env.

Pentru afișarea pe SSD a semnalelor relevante din căile de date, de la toate unitățile, aveți la dispoziție din laboratorul anterior:

- sw(7:5) = 000 → se afișează instrucțiunea pe SSD
- sw(7:5) = 001 → se afișează următoare valoare secvențială a PC (PC + 1), pe SSD
- sw(7:5) = 010 → se afișează RD1 pe SSD
- sw(7:5) = 011 → se afișează RD2 pe SSD
- sw(7:5) = 100 → se afișează Ext_Imm pe SSD
- sw(7:5) = 101 → se afișează ALURes pe SSD
- sw(7:5) = 110 → se afișează MemData pe SSD
- sw(7:5) = 111 → se afișează WD pe SSD
- dacă este necesar pentru depanarea procesorului MIPS (= trasarea atentă pas cu pas a programului scris în ROM) pe placa de dezvoltare, se pot afișa și alte semnale pe SSD / LED-uri: Branch Address, Jump Address, ALUCtrl, etc. De exemplu dacă nu funcționează BEQ afișati suplimentar imediatul extins cu semn și Branch Address pentru a depista mai ușor unde este eroarea.

Pe ledurile plăcii se vor afișa semnalele de control generate de UC. Există 8 semnale de 1 bit (cel puțin), plus ALUOp. Pe leduri se afișează toate semnalele de control definite pentru instrucțiunile implementate.

Acum, încărcați procesorul pe placa de dezvoltare, și executați pas cu pas (de la buton prin MPG) programul din memoria ROM. Verificați cu atenție la fiecare instrucțiune valorile prezente în semnalele relevante din căile de date, afișate pe SSD / LED-uri, pentru a stabili dacă instrucțiunea se execută corect.

Varianta 1 (the easy way out): executați până la sfârșit programul, și la ultimele instrucțiuni vedeți rezultatul corect (ex. suma șirului, sau calculul unei formule, etc.) caz în care procesorul funcționează corect.

Varianta 2 (no easy way out): rezultatul final nu e cel așteptat, deci se verifică pas cu pas, fiecare instrucțiune, inclusiv semnalele de control. Vezi mai jos indicii.

Se verifică succesiunea corectă a instrucțiunilor, inclusiv funcționarea instrucțiunilor de salt: în principal urmărind cum se modifică valoarea lui PC.

La instrucțiuni de tip R sau I se verifică propagarea corectă a datelor plecând de la codul mașină al instrucțiunii, ieșirile din blocul de regiștri, imediatul extins, ieșirea ALURes, WD, etc.

Pentru lw/sw cu memoria se procedează la fel.

Pentru a testa că rezultatele instrucțiunilor se scriu înapoi în registrul destinație sau locația de memorie se folosește următoarea strategie: se urmărește valoarea locațiilor destinație din instrucțiunea curentă în instrucțiunile următoare unde ele apar ca operanzi sursă (le vedeți valorile din căile de date, pe SSD). Exemplu:

addi $2, $0, 5 -- în $2 ar trebui să se încarce valoare 5
….
add $3, $2, $4 -- în acest moment, pe câmpul RD1 la ieșirea RF ar trebui să fie 5 (vizualizare pe SSD)

În cazul în care ceva nu coincide cu ce ați trasat pe foaie, localizați eroarea (unde e în procesor) și reveniți la codul VHDL pentru a încerca să o corectați.

2. Criterii de notare

Nota primită în acest laborator se va duplica și în laboratorul 7. Termenul de predare este fix (laboratorul 8), nu se admit extensii.

Contează următoarele pentru stabilirea notei finale (notă mai mare sau mai mică în funcție de criteriile de mai jos):

- Proiectul test_env să fie complet, cu procesorul implementat, fără erori de sintaxă, conform specificațiilor din laboratoarele anterioare (setul de 15 instrucțiuni implementate în procesor, program cu toate sau majoritatea instrucțiunilor – obligatoriu tip R, tip I, cu memoria, salt BEQ sau echivalent, jump).
- Cunoașterea principiilor de funcționare pentru procesor.
- Materialele scrise pe foaie în laboratoarele anterioare.
- Tot codul VHDL prezentat să fie scris individual, de către student – răspuns la întrebări pe cod, și capacitatea de a rescrie anumite porțiuni de cod pe loc.
- Trasarea/testarea pe placă: poate merge perfect sau parțial (vezi variantele 1/2 de mai sus).

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.