Laborator 06

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

Unitatea de Instruction Decode ID / Unitatea de Control UC

0. Resurse minimale necesare !

Din laboratorul 4 (terminat acasă): Tabelul cu semnalele de control rezultat din activitatea 3.3.

Din laboratorul 5 (terminat acasă): Proiectul test_env, cu unitatea IF implementată, cu mecanismul de testare, fără erori

1. Obiective

Studiul, proiectarea, implementarea și testarea:

- **Unității de decodificare a instrucțiunii, ID, pentru procesorul MIPS, pe 16 biți, un tact de ceas / instrucțiune (ciclu unic)**
- **Unității principale de control, UC, pentru procesorul MIPS, pe 16 biți, un tact de ceas / instrucțiune (ciclu unic)**

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

(!) Lectura obligatorie a cursurilor 3 și 4 pentru a înțelege activitățile din acest laborator. Cunoașterea în detaliu a noțiunilor din laboratoarele 4 și 5!

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 continuați în acest laborator (și o veți termina în următoarele), va fi partiționată în 5 componente (entități noi) pentru calea de date. Aceste componente se vor declara și instanția în proiectul test_env, în

entitatea principală (test_env, probabil…). 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 decodificare a instrucțiunii Instruction Decode – ID, împreună cu unitatea principală de control UC, 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.

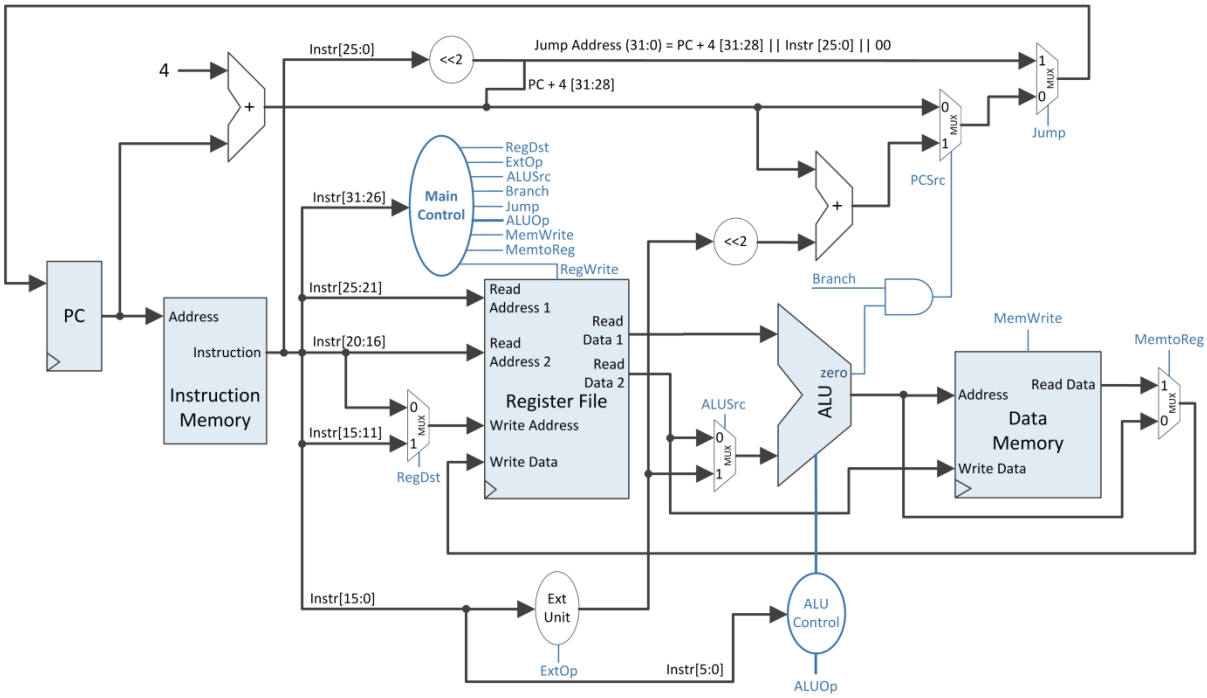

Figura 1: MIPS 32 cu ciclu unic de ceas

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

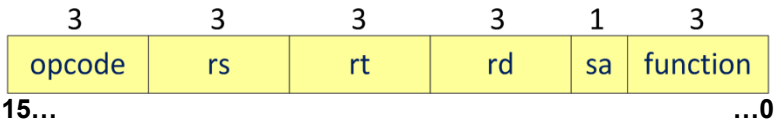

Figura 2: Instrucțiune de tip R

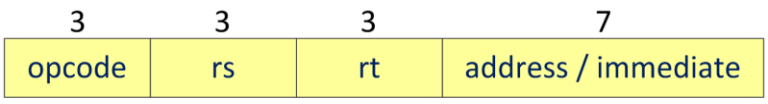

Figura 3: Instrucțiune de tip I

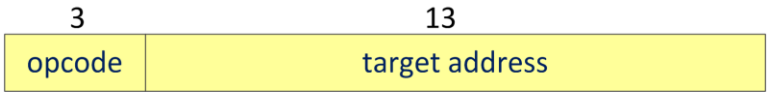

Figura 4: Instrucțiune de tip J

Unitatea ID conține următoarele elemente principale:

- Bloc de regiștrii / Register File - RF
- Multiplexor
- Unitate de extensie semn / zero

Consultați laboratorul 4 pentru caracteristicile acestor elemente pentru procesorul MIPS 16. Amintiți-vă, din laboratorul 3, despre blocul de regiștrii RF, unde citirile sunt asincrone, și doar scrierea este sincronă, pe front crescător de ceas.

Căile de date MIPS 32 pentru unitatea ID sunt prezentate în figura 5.

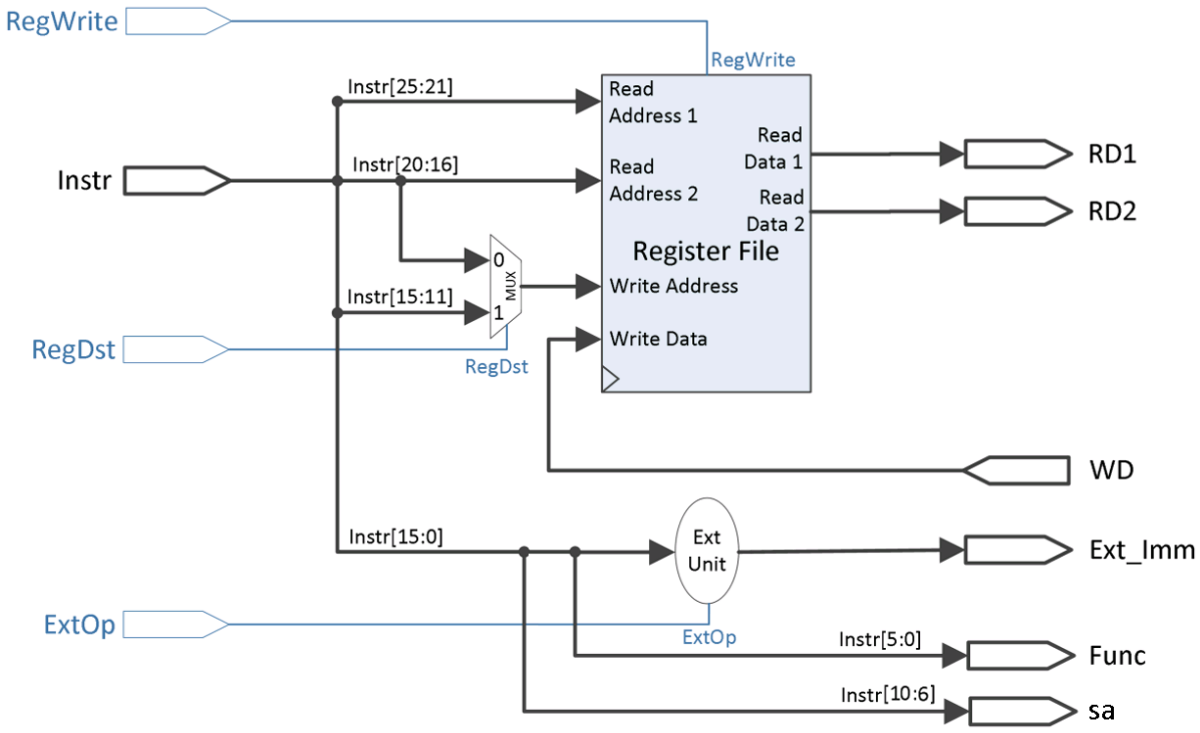

Figura 5: Unitatea ID, pentru MIPS 32, căile de date

Unitatea ID are, ca ieșiri, câmpurile de date (Read Data 1, Read Data 2 și Extended Immediate) folosite de procesor în fazele următoare de execuție ale instrucțiunii. În plus, câmpul function este furnizat mai departe către unitatea locală de control ALU.

Intrările unității ID (MIPS 32) sunt:

- Semnalul de ceas, folosit pentru scriere în RF
- Instrucțiunea Instr pe 32 de biți
- Datele care se scriu in RF, 32 de biți, pe WD
- Semnale de control:
  - RegWrite – activarea scrierii în RF
  - RegDst – selectează adresa de scriere în RF
  - ExtOp – selectează tipul de extensie pentru câmpul immediat: cu zero sau cu semn

Ieșirile unității ID sunt:

- Registru de la adresa rs, 32-biți, RD1
- Registru de la adresa rt, 32-biți, RD2
- Imediatul extins la 32-biți, Ext_Imm
- Câmpul func, pe 6 biți
- Câmpul sa, pe 5 biți

Semnificația semnalelor de control este:

- RegDst = 1 → pe Write Address din RF ajunge câmpul rd al instrucțiunii (Instr[15:11])
- RegDst = 0 → pe Write Address din RF ajunge câmpul rt al instrucțiunii (Instr[20:16])

- RegWrite = 1 → se activează scrierea (front crescător de ceas!) valorii pe 32 de biți de pe Write Data în registrul dat de Write Address, în RF

- ExtOp = 0 → imediatul pe 16 biți este extins cu zero
- ExtOp = 1 → imediatul pe 16 biți este extins cu semn

Semnalele de control ale unității principale de control sunt prezentate în figura 6. Consultați cursul 4 pentru descrierea completă a acestor semnale pentru procesorul MIPS.

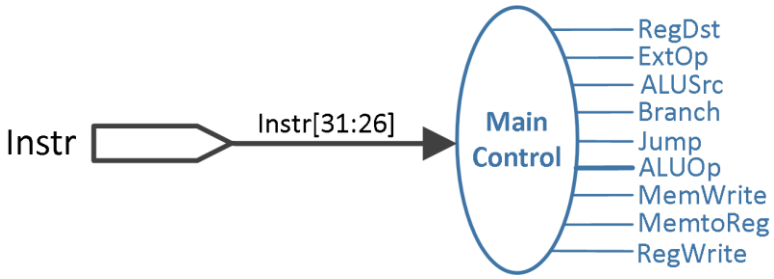

Figura 6: Unitatea principală de control UC pentru MIPS 32, cu ciclu unic

Intrarea în UC constă din cei 6 biți ai codului de operație (câmpul opcode), iar ieșirile sunt reprezentate de semnalele de control care merg în căile de date (mai puțin

semnalul ALUCtrl). Există 8 semnale de 1 bit, iar semnalul ALUOp este de 2 sau mai mulți biți, în funcție de necesar (pentru cele 15 instrucțiuni pe care le-ați ales). Din acest motiv, pe figura de mai sus, ALUOp este desenat cu linie îngroșată.

3. Activități practice

Citiți fiecare activitate în întregime, înainte să o începeți!

Resurse necesare (de avut la începerea laboratorului!):

- Rezultatele obținute din activitățile din laboratoarele 3 și 4,
- Proiectul test_env, cu unitatea IF implementată, fără erori

3.1. Proiectarea / implementarea unității ID

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

Entitatea ID va conține elementele prezentate în figura 5, care nu se vor descrie cu entități suplimentare, cu excepția RF (vezi laboratorul 3)!

Folosiți o abordare simplă pentru descrierea unității de extensie, ca în laboratorul 2 (when / else, case, sau if…)

Atenție! Aveți grijă la transformarea câmpurilor de date din figura 5 (MIPS 32) în versiunea proprie de implementare a MIPS pe 16 biți.

3.2. Proiectarea / implementarea unității UC

Primul pas este identificarea valorilor pentru fiecare semnal de control, in funcție de instrucțiune. Vezi tabelul completat pentru activitatea 3.3 din laboratorul 4 ! În cazul în care nu aveți la dispoziție acest tabel care trebuia făcut la tema pentru laboratorul anterior, consultați cursul 4 și stabiliți valorile pentru semnalele de control ale celor 15 instrucțiuni (pentru a reuși testarea pe placă, în timp util, scrieți aceste semnale pentru 4-6 instrucțiuni, și terminați acasă).

Unitatea UC se poate implementa fie ca o nouă entitate în proiectul test_env, sau ca un simplu proces de decodificare în arhitectura test_env. Ca o sugestie suplimentară, semnalele de control se recomandă a fi declarate individual, pentru o bună lizibilitate a descrierii VHDL. Pentru descriere, se recomandă folosirea unui proces, cu o structura case (după opcode). Pentru a evita, pe o ramură „when” a case-ului, enumerarea și atribuirea tuturor semnalelor de control, acestea se vor inițializa la valoarea 0 în același proces dar înainte de case, urmând ca pe fiecare ramură să fie puse pe 1 doar semnalele necesare (restul rămân implicit pe 0 dacă nu se specifică altfel).

3.3. Testarea unităților ID și UC

Lucrați în proiectul test_env, finalizat din laboratorul anterior!

În entitatea test_env declarați și instanțiați unitatea ID, și unitatea UC (dacă ați declarat-o ca entitate).

Conectați unitatea ID împreună cu unitatea existentă IF. Ieșirea unității IF, cei 16 biți ai instrucțiunii, reprezintă o intrare în ID.

Conectați semnalele necesare generate de UC la unitățile IF și ID.

Următoarele elemente din fluxul de execuție se vor implementa abia în următoarele laboratoare. Din acest motiv, pentru a verifica în acest laborator (!) scrierea în RF folosiți sumatorul din laboratorul 3 pentru a aduna ieșirile din ID RD1 și RD2, iar ieșirea din sumator se leagă pe intrarea WD a ID. **Atenție**: practic acum RegWrite este controlat de unitatea de control, deci scrierea sumei în RF se va face doar pentru acele instrucțiuni din programul vostru unde se face scriere in registrul destinație din RF.

Atenție! Pentru scrierea în blocul de regiștrii RF, este necesar ca scrierea să fie controlată de la unul din butoane, la fel cum este controlată scrierea în registrul PC. Practic, în RF trebuie folosită ieșirea enable a aceluiași MPG folosit pentru activarea scrierii în PC (din unitatea IF), ca o condiție suplimentară pe lângă testarea frontului de ceas și a lui RegWrite.

Pentru conectarea cu SSD a semnalelor prezente în căile de date, de la IF și ID, trebuie să extindeți multiplexorul cu două intrări din laboratorul anterior:

- sw(7:5) = 000 → se afișează instrucțiunea pe SSD
- sw(7:5) = 001 → se afișează următoare valoare secvențială a PC, și anume 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 WD pe SSD
- …

Pe ledurile plăcii se vor afișa semnalele de control ale UC. Există 8 semnale de 1 bit, plus ALUOp. Pentru plăci de dezvoltare care au doar 8 leduri, folosiți un alt switch pentru a controla ce se afișează pe leduri:

- sw(0) = 0 → Pe leduri se afișează cele 8 semnale de 1 bit, în ordinea dorită.
- sw(0) = 1 → Pe leduri se afișează cei n biți ai ALUOp.

La fel ca în laboratorul anterior, folosiți cele două butoane care trec prin MPG pentru a reseta PC, respectiv a controla scrierea în registrul PC și în RF. Veți simula astfel fluxul normal, secvențial, de execuție a instrucțiunilor.

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 definit anterior, 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.