Laborator 09

Procesorul MIPS – versiune pe 16 biți, implementare de tip pipeline

0. Resurse / cunoștințe minimale necesare !

0.1. Cunoștințe

Procesorul MIPS, versiunea pe 16 biți, implementată în laboratoarele 5, 6, și 7: schemă, principii de funcționare, modul de execuție a fiecărei instrucțiuni implementate pe procesor.

0.2. Resurse

Proiectul test_env conținând implementarea proprie a procesorului MIPS 16 cu ciclu unic.

Atenție!:

- Implementarea cu succes a versiunii pipeline depinde de baza de unde porniți: test_env cu MIPS 16 corect / complet implementat
- Dacă la predarea procesorului din laboratorul 8 nu ați obținut nota maximă, este obligatoriu ca înainte de laboratorul 9 să vă puneți la punct procesorul prin una din variantele următoare:
  1. Individual - recomandat.
  2. Folosiți o versiune corectă de la un coleg, și comparați descrierea voastră VHDL cu cea corectă pentru a depista eventualele greșeli / lipsuri, și efectuați corecțiile necesare pe proiectul personal. Dacă nu aveți o astfel de versiune corectă la îndemână, o găsiți aici începând cu Vineri 28 aprilie 2017.
  3. ~~Copiați o variantă completă / corectă de la colegi.~~

1. Obiective

Studiul, proiectarea, implementarea și testarea:

- **Procesorului MIPS 16, versiunea pipeline**

2. Transformarea procesorului MIPS 16 cu ciclu unic într-un procesor de tip pipeline

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

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.

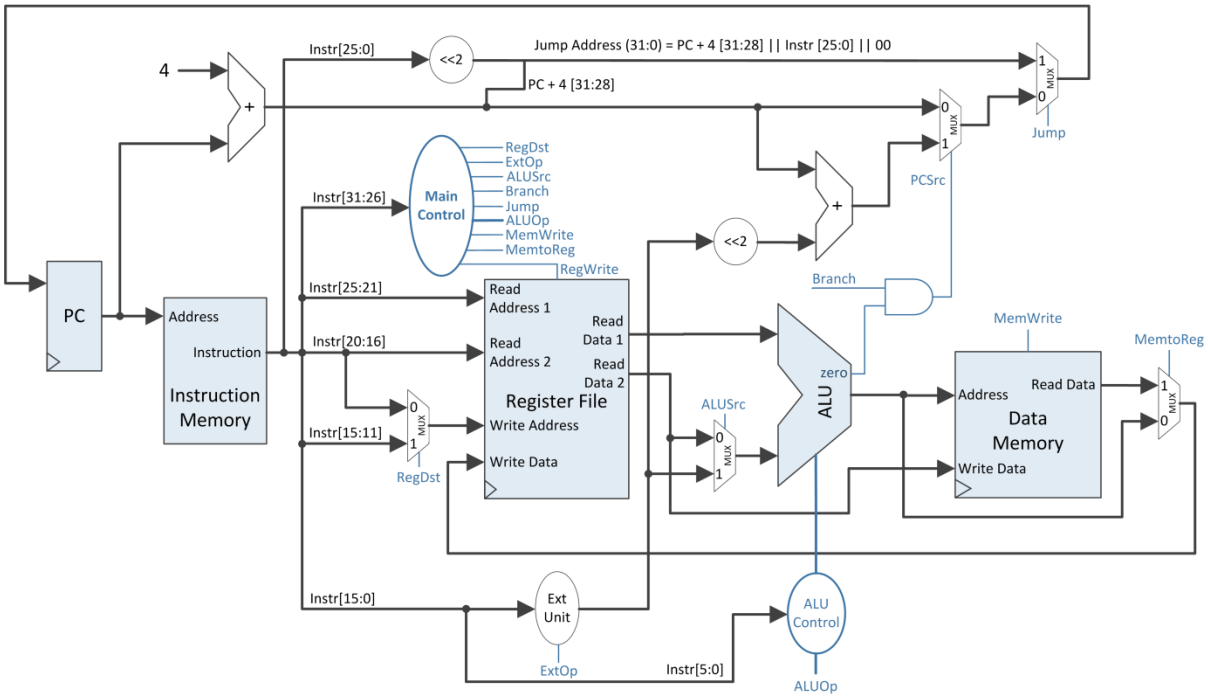

Figura 1: MIPS 32 cu ciclu unic de ceas

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

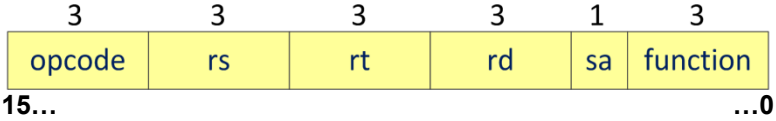

Figura 2: Instrucțiune de tip R

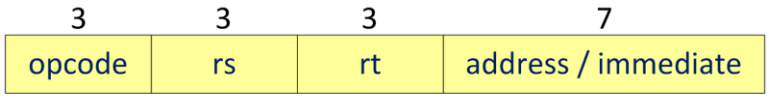

Figura 3: Instrucțiune de tip I

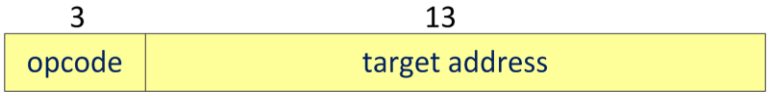

Figura 4: Instrucțiune de tip J

Principala problemă a procesorului MIPS cu ciclu unic este lungimea căii critice, pentru instrucțiunea load word (vezi curs 4). Timpul necesar pentru transmiterea (combinațională) și stabilizarea nivelelor logice de-a lungul căii critice trebuie să fie acoperit de durata ciclului de ceas. Astfel rezultă un ciclu lung de ceas.

Pentru a reduce durata ciclului de ceas, soluția este secționarea căii critice cu elemente de stocare sincrone (regiștri). Acești regiștri se interpun practic între unitățile procesorului MIPS 32, care coincid și cu fazele de execuție ale instrucțiunii: IF, ID, EX, MEM și WB. Astfel, se pot executa până la 5 instrucțiuni consecutive din program, fiecare aflându-se într-una din cele 5 faze de execuție, în funcție de ordinea de succesiune. Unitățile din varianta pipeline vor fi referite în continuare ca **etaje.**

Căile de date, și partea de control pentru procesorul MIPS 32 în varianta pipeline sunt prezentate în figura 5.

Fiecare registru intermediar va fi referit în funcție de poziția lui între etaje. Registrul dintre etajul IF și etajul ID este IF/ID, dintre ID și EX este ID/EX, etc.

Rolul regiștrilor intermediari este de a păstra rezultatele intermediare ale instrucțiunilor, pentru a furniza aceste rezultate intermediare etajului următor, în următorul ciclu de ceas.

În plus (!) execuția în căile de date depinde de valorile semnalelor de control, valori care sunt specifice fiecărei instrucțiuni. Astfel, prin intermediul regiștrilor intermediari (începând cu ID/EX) se transmit și valorile semnalelor de control, pentru fiecare etaj următor. Semnalele de control sunt grupate simbolic după numele etajului unde trebuie să ajungă.

Practic, semnalele de control sunt transmise în tandem cu rezultatele intermediare ale instrucțiunii, pentru fiecare etaj care urmează.

La curs (vezi site) va avea (a avut) loc o discuție pe larg legată de acest subiect, detaliile descrise până acum sunt minimul necesar pentru a începe transformarea procesorului din ciclu unic în pipeline.

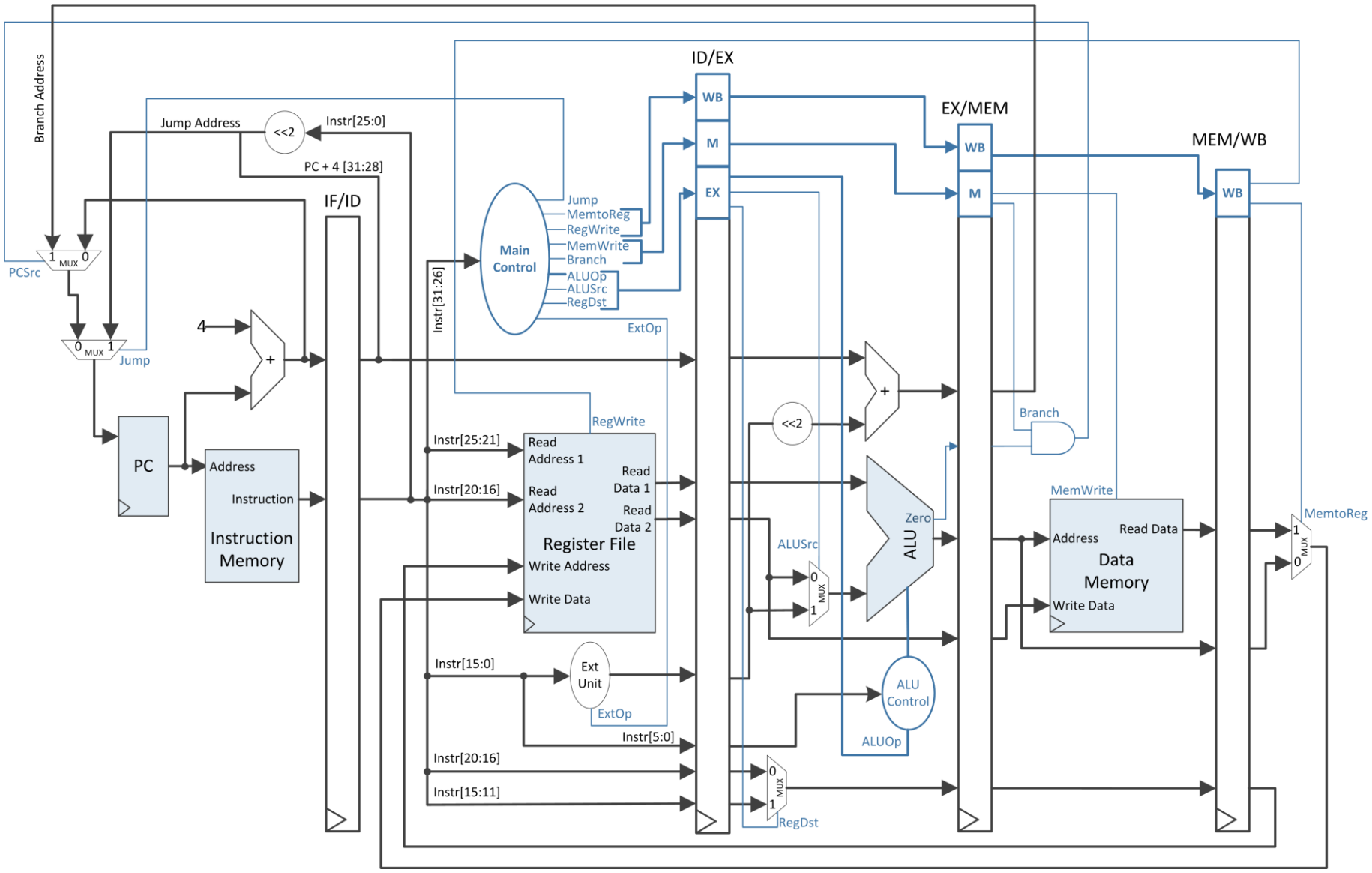

Figura 5: Procesorul MIPS 32, implementare de tip pipeline, obținut prin secționarea căilor de date MIP 32 cu ciclu unic, din figura 1

O diferență notabilă pe schemă, față de gruparea în unități pe care ați făcut-o deja pentru MIPS 16 cu ciclu unic, este că multiplexorul care selectează adresa registrului destinație este plasat în EX, nu în ID cum îl aveți în varianta proprie. Există 2 soluții posibile:

1. Îl lăsați în ID, în acest caz semnalul de control RegDst nu se mai transmite prin ID/EX, ci se ia direct de la UC. UC și ID se află în același etaj de pipeline.
2. Îl mutați în EX, modificând porturile de intrare / ieșire ale unităților ID și EX, și transmiteți RegDst conform schemei.

În rest, semnalul de control MemRead se va ignora, la fel ca la MIPS 16 cu ciclu unic, pentru a evita modificări suplimentare în căile de date.

Un alt semnal care nu apare explicit pe schema, dar a fost folosit și la MIPS 16 cu ciclu unic este semnalul de 1 bit sa. Acest semnal trebuie transmis prin registrul intermediar ID/EX către unitatea EX (unde e legat deja la ALU).

3. Activități practice

Citiți fiecare activitate în întregime, înainte să o începeți!

Resurse necesare (de avut la începerea laboratorului!):

- Proiect Xilinx cu test_env, care să conțină implementarea proprie a procesorului MIPS 16 cu ciclu unic, corectă și completă

3.0. Verificarea procesorului MIPS 16 cu ciclu unic

În cazul în care în laboratorul 8 nu ați reușit testarea completă a procesorului, și ați reușit acasă să îl corectați / completați, se poate încerca testarea la începutul laboratorului nouă (nu se recomandă mai mult de 15 minute pentru această activitate).

Scopul este de a vă asigura ca procesorul de la care porniți este funcțional.

3.1. Proiectarea regiștrilor intermediari (hârtie / instrument de scris)

Pentru fiecare registru intermediar, enumerați ce câmpuri trebuie să stocheze, în funcție de particularitățile versiunii proprii a procesorului MIPS 16.

Folosiți schema din figura 5, dar (!) țineți cont de particularitățile procesorului vostru: e pe 16, NU pe 32 de biți, și în funcție de setul de instrucțiuni alese, căile de date s-ar putea să conțină elemente adiționale.

De exemplu, pentru primul registru intermediar, IF/ID, trebuie memorate două câmpuri (le denumim generic cu numele etajului din care provin, în față):

- IF.PC+4 pe 16 biți
- IF.Instruction pe 16 biți

Rezultă un necesar de 32 de biți pentru IF/ID.

Descrieți în mod similar regiștrii ID/EX, EX/MEM, MEM/WB.

Pentru fiecare registru intermediar, când faceți descrierea pe biți a câmpurilor de intrare, vizualizați unitățile asociate (cea de intrare, respectiv destinația, ex. pentru IF/ID unitățile IF, respectiv ID) în laboratoarele 5, 6, respectiv 7. Identificați aceste câmpuri între porturile de intrare/ieșire ale unităților. Acest pas va fi foarte util pentru activitatea următoare.

3.2. Descrierea regiștrilor intermediari în VHDL

Pentru această activitate se va lucra în entitatea principală test_env (unde sunt instanțiate și legate unitățile principale).

Atenție: pe măsură ce avansați cu procesul de secționare, vor apărea mici modificări necesare pentru unitățile existente. Veți remarca aceste modificări dacă studiați cu atenție, în paralel, schema secționată din figura 5, si schemele particulare ale unităților descrise în laboratoarele 5, 6, 7. De exemplu, unitatea ID necesită adăugarea unui port de intrare pentru adresa de scriere în blocul de regiștri, adresă care se va întoarce (via port map) din ultimul registru intermediar MEM/WB.

Nu se vor declara entități noi pentru regiștrii intermediari. Pentru fiecare registru se va declara un semnal de dimensiunea potrivită, conform rezultatelor primei activități, iar comportamentul se va descrie printr-un proces, unde are loc un transfer sincron pe front crescător de ceas. Pentru a ușura munca de testare, fiecare registru va fi controlat și de un semnal de la buton (același monopuls folosit pentru validarea scrierii în PC).

Realizați secționarea propriu-zisă, prin folosirea câmpurilor din regiștrii intermediari ca intrări în etajul (unitatea) care urmează (vezi figura 5), sau transmiterea lor mai departe spre registrul intermediar următor. Unitatea care „urmează” nu e neapărat cea situată la dreapta registrului, e vorba de următoarea în fluxul de date/semnale de control. De exemplu, valoarea semnalului de control RegWrite transmisă până în registrul intermediar MEM/WB trebuie legată pe intrare aferentă din unitatea ID.

Exemplu (se poate folosi și concatenare dacă se dorește): pentru IF/ID se declară semnalul RegIF_ID, de 32 de biți, iar comportamentul se descrie într-un proces unde:

```
pe front crescător de ceas
        RegIF_ID(31..16) <= PC+4
        RegIF_ID(15..0)  <= Instruction
```

Unde PC+4 și Instruction sunt porturile de ieșire din unitatea IF.

Pozițiile din RegIF_ID corespunzătoare celor două câmpuri se transmit mai departe:

- Poziția corespunzătoare PC+4 merge ca intrare în registrul intermediar următor ID_EX,
- Poziția corespunzătoare Instruction este transmisă spre unitatea ID (modificați port map).

3.3. Participarea la cursul 9

Este obligatorie participarea la cursul 9, unde se vor oferi explicații în detaliu legate de transformarea procesorului MIPS cu ciclu unic, în MIPS pipeline.

4. Referințe

- Cursurile 3 & 4, și cursul MIPS pipeline 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.