UNIVERSITETI I PRISHTINËS

FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE

DEPARTAMENTI I KOMPJUTERIKËS

Lënda: Arkitektura e kompjuterëve

Implementimi I Single Cycle CPU 16 bëshe

Profesor: Valon Raça **Grupi: 12**

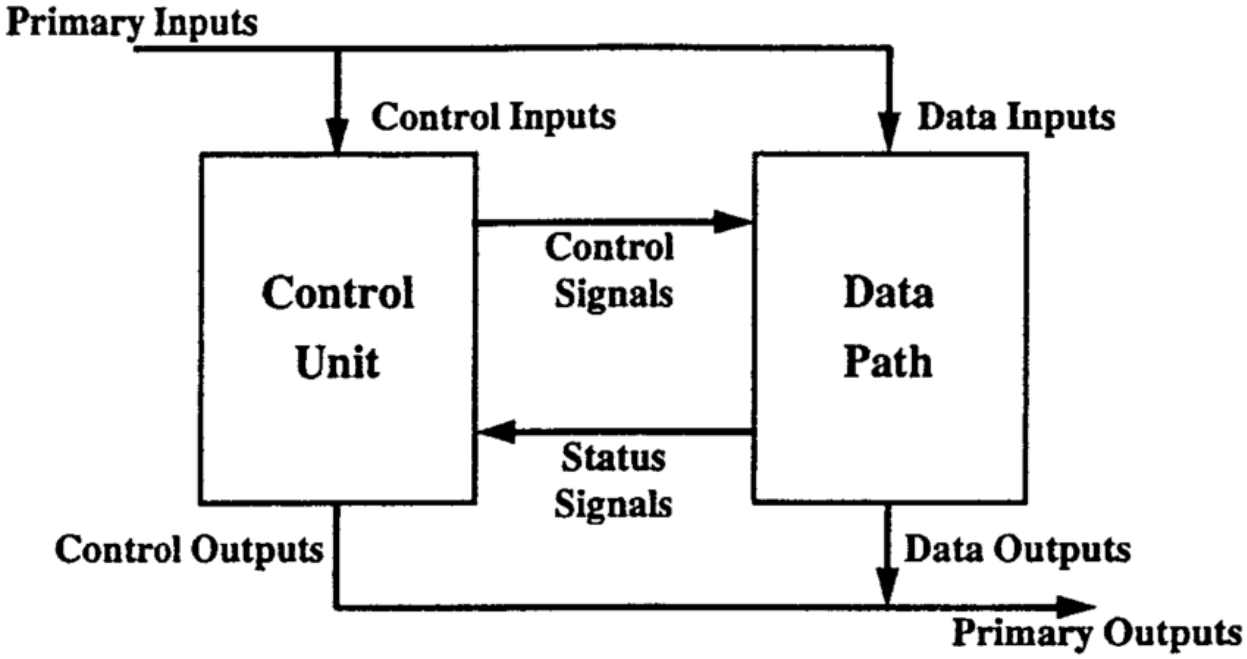
Student	ID
Erna Hulaj	200714100110
Shefket Bylygbashi	200714100079
Ofertë Ramadani	200714100098
Rina Bekolli	200714100111
Vanesa Aliu	200714100099

HYRJE

Qëllimi i detyrës është dizajnimi i një CPU 16-bitëshe e cila është e implementuar në Single-Cycle Datapath me anë të veglës Verilog.

Procesi i krijimit të CPU-së ndahet ne dy pjesë:

1. Krijimi I DataPath
2. Dizajnimi I Control Unit



Krijimi I datapath perfshin lidhjen e komponentave te procesorit që kryejnë operacione aritmetike si dhe ruajtjen e te dhënave.

Elementet përbërëse të Datapath janë:

- ALU 16 bitëshe

Një ALU është pjesa themelore e një CPU-je ku kryhen veprimet aritmetike dhe logjike mbi operandët në bazë të udhëzimeve kompjuterike. Përkrah numra me gjatësi 16 bitëshe. Fillimisht është krijuar ALU 1 bitëshe ku pastaj janë lidhur në seri 16 ALU 1 bitëshe përmes metodës RippleCarry.

- ALU Controler

Në Alu Control përmes ALUOp përcaktojmë se çfarë operacione dëshirojmë të kryejmë.

- Instruction memory

Instruction Memory shërben për të ruajtur udhëzimet e një programi dhe udhëzimet të furnizimit me të dhëna nga një adresë. Instruksionet të cilat i përkrah CPU-ja, sipas kërkesave të detyrës, janë instruksionet e formatit I dhe formatit R. Instruksionet e formatit R e kanë op code-in e specifikuar, kurse veprimet që kryhen pastaj variojnë në bazë të pjesës së fundit, pjesës së funct e cila është 4 bitëshe.

Formati i intruksionit të R-formatit: `opcode-000 rs-xxx rt-xxx rd-xxx funct –xxxx`

- PC

PC (Program Counter) është një regjistër që mban adresën e instruksioneve të tanishme.

- Register file

Për të mbështetur udhëzimet e formatit R, do të na duhet të shtojmë një element memorik të quajtur register file, i cili është një koleksion regjistrave i lexueshëm/shkrueshëm. Gjithsej ka 8 regjistra. Ka të implementuar dy multiplekserë për zgjedhjen e rs dhe rt-së dhe një dekoder i cili vendos në cilin regjistër te shkruajmë, pastaj në atë regjistër vendosen të dhënat që vijnë në writedata.

- Data memory

Elementi data memory ka funksionalitet leximin dhe shkrimin e të dhënave në / nga kujtesa. Ka dy hyrje. Një për adresën e vendndodhjes së memories hyrëse, tjetra për të dhënat që shkruhen në memorie nëse është e mundshme. Dalje janë të dhënat e lexuara nga memoria sipas vendndodhjes që po i adresohemi, nëse është e çasshme. Leximet dhe shkrimet sinjalizohen (kontrollohen) nga MemRead dhe MemWrite, përkatësisht, për të cilën duhet të pohohet veprimi përkatës që do të kryhet.

- Multiplexer

Është një pajisje që zgjedh midis disa sinjaleve hyrëse analoge ose digjitale dhe e përcjell atë në një linjë të vetme daljeje.

Organizimi I fajllave:

Projekti ndahet në folderin Kodi i cili përmban të gjitha fajllat që përmbajn kod dhe në folderin Test të gjitha fajllat për testim.

1.Komponentat që i kemi përdorur për dizajnimin e Single-Cycle-CPU:

- Mux2ne1

Multiplekser I thjeshtë, ka dy hyrje dhe nje bit selektues, te gjitha janë një bitëshe, ka një output e cila varet nga biti selektues, kur është zero zgjedh inputin e parë, përndryshe zgjedh inputin e dytë:

```
assign Dalja = S ? Hyrja1 : Hyrja0;
```

- Mux6ne1

Permban 6 inputs dhe nje output I cili varet nga 3 bita kontrollues permes te cilit zgjedhet operacioni perkates :

```
assign Dalja = S[2] ? (S[1] ? (S[0] ? Hyrja4 : Hyrja5) : (S[0] ? Hyrja2 : Hyrja2)) : (S[1] ? (S[0] ? Hyrja3 : Hyrja1): (S[0] ? Hyrja2: Hyrja0));
```

- Mbledhesi_1bit

Mbledhësi i plotë 1 bitësh merr si hyrje dy bita A dhe B, poashtu edhe bartjen, në dalje vendos shumën dhe vendos mbetjen e cila do shërbejë tek mbledhësi me ripple carry si bartje për mbledhësin tjetër.

Qarku për shumën është xor mes A, B dhe bartjes (CIN) kurse mbetja është

```
A*B+A*CIN+B*CIN  (A&B  | CIN&A | CIN&B)
```

- Mbledhesi_16bit

Ka dy hyrje 16bit-ësh A,B një hyrje 1 bit-ësh per Carryout dhe rezultatin 16 bit-ësh.

Formohet nga 16 mbledhësa 1 bit permes metodes RippleCarry.

- ALU_1b

  Ky si hyrje ka A,B,CIN,BINVERT,LESS,ALUCtrl dhe output rezultatin dhe cout, kjo ALU sherben per ndërtimin e ALU 16 bit-ëshe.

Veprimet që kryen dhe shoqërimi I daljes:

```
assign teliDHE = A & mB;

assign teliOSE = A | m;

assign teliXOR = A ^ mB;

SLTI slti(A, mB, LESS);

mbledhesi_1b mbledhesi(A, mB, CIN, COUT, teliMBL);
```

- ALU_16b

Si input përmban telat hyrës A, B 16-bitesh, BNegate, ALUOp dhe si output ka zero, overfloë, carryout dhe rezultatin 16bitësh.

Perbëhet nga lidhja e 16 ALU 1bitëshe sipas metodës RippleCarry ku COUT i ALU-së paraprake paraqet CIN per ALU-në e rradhës. Dhe gjithashtu kemi inputin zero ku permes saj kuptojmë se a jane inputet (A,B) te barabarta apo jo, dhe permes overfloë kuptojmë se a e ka rezultati shenjën e ndryshme me telat hyres A, B.

- ALUControl

Si input ka ALUOp 2-bitësh, Funct 4-bitësh, OPCode 3-bitësh dhe output ALUCtrl 4-bitësh. Permes ALUOp përcaktojmë se çfarë operacioni deshirojmë të kryejmë ku edhe sipas kerkesave ALUCtrl merr vlera specifike te komandave te ndryshme sikurse mbledhje, zbritje etj.

- CONTROL UNIT

Merr si hyrje 3 bita të opcode, që i merr nga instruction memory, pastaj përmes: `alëays @(OPCODE)` . Shqyrton kur kemi instruksione R apo I format dhe në varësi të instruksionit vendos bitat dalës perkates.

- INSTRUCTION MEMORY

Merr një hyrje 16 bitëshe nga PC, e cila paraqet adresën e instruksionit, dhe një output16 bitësh që paraqet instruksionin e caktuar, këtu kemi lexuar prej fajllit të jashtëm ku kemi vendosur instruksionet:

```
initial $readmemb("instMemory.mem", iMem);

assign Instruction = {iMem[PC], iMem[PC+1]};
```

Rreshti i fundit vlen pasi që adresat i kemi 8 bitëshe, kurse dalja 16 bit, i bie se duhet të marr dy rreshta të adresës nga memoria.

InstMemory është në numra binar dhe është 128bajt, ku dhjetë bajtat e parë janë të rezervuar kurse nga bajti i 10 kemi shkruar instuksionet per ekzekutim nga memoria.

Tek fajlli instMemory.mem gjenden instruksionet:

```
add $r1, $zero, $zero 0000_0000_0001_0000
lw $r2, 4($r1)    1010_0101_0000_0100
xor $r1, $r2, $r3  0000_1001_1001_1101
bne $r2, $zero, kercimi  1110_1000_0000_0001
andi $r3, $r1, 5  0010_0101_1000_0101
kercimi: sw $r3, 4($r2)  1100_1001_1000_0100
addi $r4, $r3, 4     0110_1110_0000_0100
slti $r5, $r4, 2     1001_0010_1000_0010
mod $r6, $r5, $r4    0001_0110_0110_001
```

- Register File

-3 hyrje 3 bitëshe, që mbajnë adresat e regjistrave RS,RT,RD, 2 hyrje 2 bitëshe qe paraqesin RegWrite dhe clock, në dalje 3 dalje 16 bitëshe, WriteData, ReadRS, ReadRT Ku ReadRS/ReadRT sherbejnë për të lexuar të dhënat nga regjistrat e përcaktuar ne RS dhe RD ndërsa WriteData paraqet te dhënën e cila do të shkruhet ndërsa clock dhe RegWrite percaktojnë se a mundemi të shënojmë në register file apo jo. Numri i regjistrave do të jetë 8 dhe regjistrat do të jenë të gjatësisë 16 bit.

- DataMemory

Si hyrje kemi: Adresa(16bite-sh qe vje nga ALU),WriteData(16bite-sh nga regjistri),MemWrite,MemRead,clock dhe outputin 16bitesh ReadData.

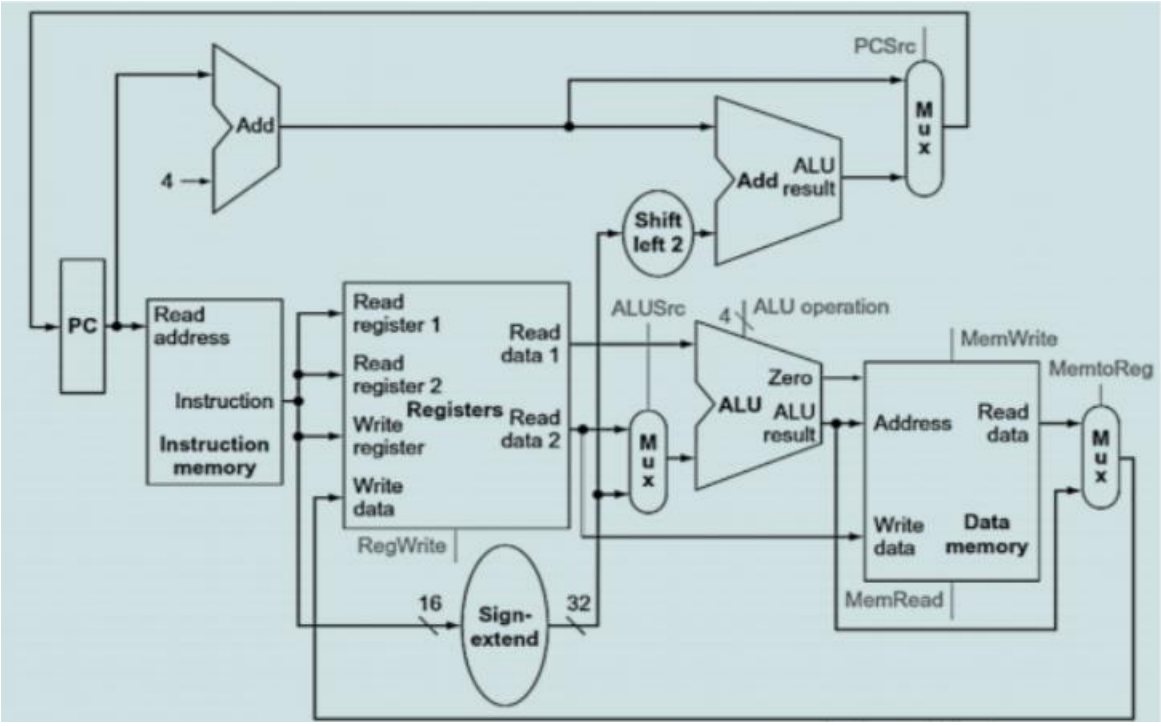
Sikurse instruction memory dhe kjo lexon nga nje fajll I jashtem por per dallim nga I.M është se këtu mundemi edhe të shkruajmë të dhëna pra:

```
$Writememh("dataMemory.mem”,dataMem);
dataMem[Adresa] <= WriteData[15:8];
dataMem[Adresa+1] <= WriteData[7:0];
```

-dataMem[] paraqesin adresat në memorie që janë 8 bitëshe, pra nëse shënojmë të dhëna 16 bitëshe ajo duhet të ruhet në dy rreshta të memories, e njëjta logjikë vlen edhe per daljen:

```
assign ReadData = {dataMem[Adresa],dataMem[Adresa+1]};
```

- DataPath



Paraqet pjesën më të rëndësishme ku bëhet ndërlidhja e të gjitha komponentave te lartcekura. Secili element (tel) që nuk I takon ndonjë komponenti apo nuk është hyrje duhet të deklarohet në DataPath. Përmes DataPath na mundësohet komunikimi mes të gjitha pjesëve të dizajnuara deri më tash, të cilat në tërësi formojnë CPU-në.

- CPU

Pjesa e fundit në të cilën ndërlidhen te gjita elementet. Të vetmin input ka Clockun. Fillimisht vendosim vlerën e PC-së në 10, ngase adresat prej 0-9 janë të rezervuara.

```
assign PC_Initial = 16'd10;
```

Bartim parametrat tek instruction memory, datapath dhe UI control, me këto pjesë kompletohet CPU.

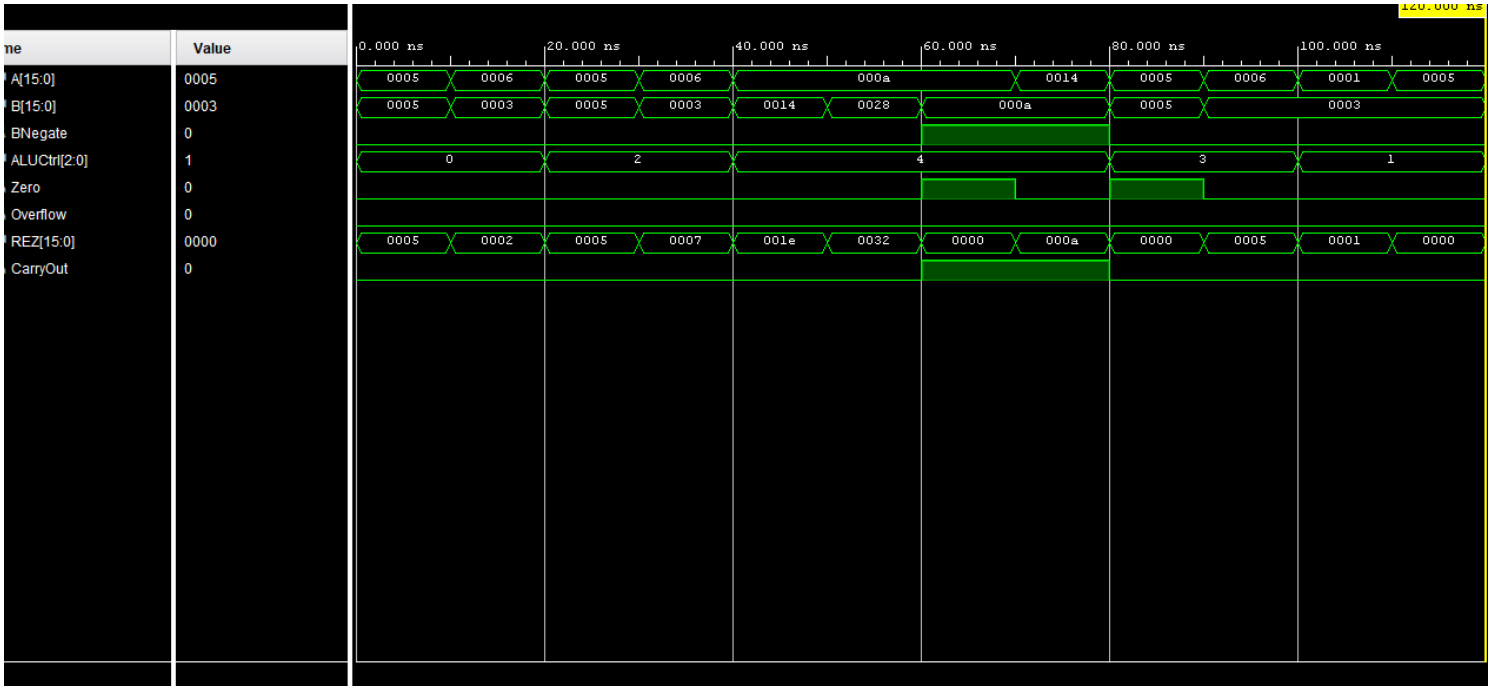
Pamje nga Testimi I ALU-së Kryesore

```
//AND
#0 A=16'd5; B=16'd5;ALUCtrl=3'b000; BNegate=1'b0;
#10 A=16'd6; B=16'd3;ALUCtrl=3'b000;BNegate=1'b0;
//OR
#10 A=16'd5; B=16'd5;ALUCtrl=3'b010;BNegate=1'b0;
#10 A=16'd6; B=16'd3;ALUCtrl=3'b010;BNegate=1'b0;
//ADD
#10 A=16'd10; B=16'd20;ALUCtrl=3'b100;BNegate=1'b0;
#10 A=16'd10; B=16'd40;ALUCtrl=3'b100;BNegate=1'b0;
//SUB
#10 A=16'd10; B=16'd10;ALUCtrl=3'b100;BNegate=1'b1;
#10 A=16'd20; B=16'd10;ALUCtrl=3'b100;BNegate=1'b1;
//XOR
#10 A=16'd5; B=16'd5;ALUCtrl=3'b011;BNegate=1'b0;
#10 A=16'd6; B=16'd3;ALUCtrl=3'b011;BNegate=1'b0;

//SLTI
#10 A=16'd1; B=16'd3; ALUCtrl=3'b001;BNegate=1'b0;
#10 A=16'd5; B=16'd3; ALUCtrl=3'b001;BNegate=1'b0;

#10 $stop;
end

ALU_Extra ALUExtraTest(A,B,BNegate,ALUCtrl,Zero,Overflow,REZ,CarryOut);
```



Përfundim

Punimi I kësaj detyre, që ishte ndoshta më e detajuar sesa menduam fillimisht, konsiderojmë se na ka shërbyer që në mënyrë më të thellë të kuptojmë mënyrën e funksionimit të një CPU-je, ndonëse shumë më e thjeshtë se CPU-të që përdoren në ditët e sotme.

Burimet e siguruara nga profesori i lëndës, duke veçuar ushtrimet e mbajtura, fajllat e CPU-së 32 bitëshe dhe video ligjeratat përkatëse ishin pikënisje shumë e mirë për ne dhe na ndihmuan dukshëm gjatë punës, duke na dhënë një vullnet të shtuar për të përfunduar detyrën. Prandaj, jemi shumë falenderues për korrektësinë e treguar.