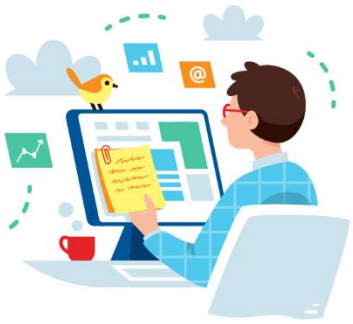
UNIVERSITETI I PRISHTINËS
Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike
Departamenti i Kompjuterikës

Lënda: Arkitektura e Kompjuterëve

Single-Cycle CPU



Profesor: Prof. Valon Raca **Grupi 13**

Maj 2020

1. Të dhënat rreth grupit

	Emri dhe mbiemri	ID
1.	Aurel Demiraj	180714100095
2.	Betim Thaçi	180716100041
3.	Durajet Mustafa	180714100091

2. Hyrje

Qëllimi i detyrës është dizajnimi i një CPU 16-bitëshe (Single-Cycle).

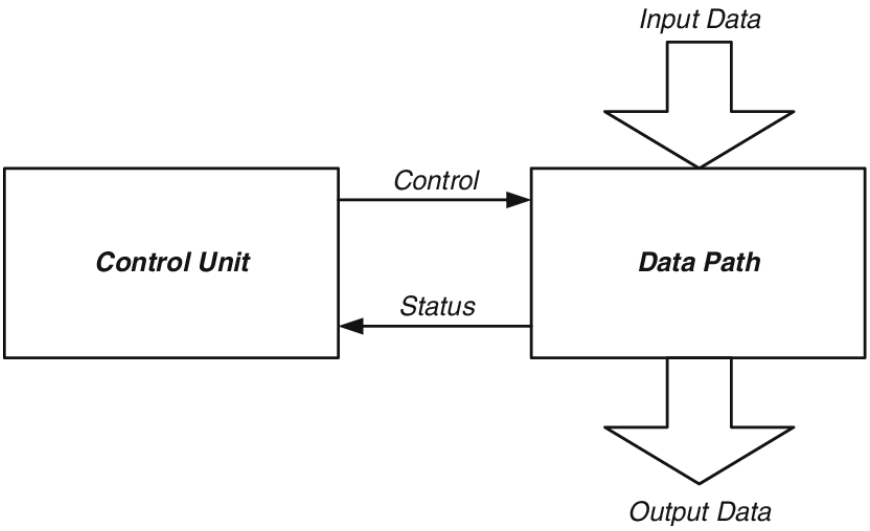
Procesi i krijimit të kësaj CPU-je duke përdorur një nënbashkësi te instruksioneve të MIPS është e ndarë në dy pjesë themi si kryesore :

• **Krijimi i Datapath**

Komponentët e procesorit që përformojnë operacione aritmetike si dhe ruajnë të dhëna.

• **Dizajnimi i një Control Unit (CU)**

Komponentët e procesorit që e komandojnë datapath-in,memorien,I/O paisjet(input,output) duke i'u përshtatur instruksioneve të memories.



Elementet që e përbëjn Datapathin tonë janë:

• *ALU 16-bitëshe (Arithmetic Logic Unit,Pikë startuese e këtij projekti)*

Pjesa e një kompjuteri që kryen të gjitha llogaritjet aritmetike, të tilla si add,sub,or,xor,and,etj.

• *ALU Control*

Specifikon se çfarë operacioni kryen ALU.

• *Instruction Memory*

Një njësi memorie për të ruajtur udhëzimet e një programi dhe udhëzime të furnizimit me të dhëna nga një adresë.

• *PC (Regjistri i adresave të rekordeve ose udhëzuesve të programit)*

Është një regjistër që mban adresën e instruksioneve të tanishme.

• *Register file (Nje grumbull i regjistrave)*

Çdo regjistër mund të lexohet ose shkruhet duke e specifikuar numrin e ati regjistri.

• *Data Memory*

Zbaton funksionalitetin për leximin dhe shkrimin e të dhënave tek/nga memoria.

• *Multiplexer (mux. 4to1 & 2to1)*

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

3. Organizimi i fajllave

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

Alu_1bit – përmban modulin alu_1bit ()

Hyrjet: a,b,cin,ainvert(vlera e saj është 0),bnegate,op.

Daljet: result,cout.

Alu_16bit – përmban modulin alu_16bit()

Tani hyrjet dhe daljet nuk janë më 1bitë-she por 16 bitë-she.

Përbëhet nga 16 Alu 1bitë-she me anë të metodës Ripple-carry kur cout i njërës alu është hyrje për alu-n e ardhshme.

Alu jonë përkrah këto operacione and,or,add,xor,sub,lw,sw.

Alu_control – përmban modulin alu_control()

Hyrjet: ALUOp,Funct.

Dalja: ALU_Cnt.

Ne Alu Control permes ALUOp përcaktojme se qfar operacione dëshirojme të kryejm, neser ALUOp është 1 behet mbledhje dhe nese është 0 kryen operacionet sipas funksionit.

Ne rastin tonë daljet nuk jane 4bitë-she siç është e cekur në detyrë, por jane 3bitë-she, biti i parë për bnegate dhe 2bitat tjerë që shërbejnë për multiplekserin të zgjedh njerin nga operacionet and, or, xor, add dhe sub.

Cu – përmban modulin cu()

Hyrja: Opcode

Daljet: RegDst,ALUSrc,MemToReg,RegWrite,MemWrite,ALUOp.

Formatet(4 raste): R,LW,SW,ADDI.

Mux_2to1 – përmban modulin mux_2to1()

Hyrjet: s,d0,d1

Dalja: z(rezultati)

Mux_4to1 – përmban modulin mux_4to1()

Hyrjet: s,d0,d1,d2,d3

Dalja: z(rezultati)

Jane multiplekserat qe na ndihmojne për të zgjedhur operacionet e kerkuara.

InstrMemory – përmban modulin InstrMemory()

Hyrjet: clk,Hyrja

Dalja: dalja.

Instruction Memory është vetëm Read-Only(vetëm për Instruksione). Hyrje 16 bitëshe nga PC për përcaktimin e adresës së instruksionit. Dalje 16 bitëshe për leximin e instruksionit.

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

Me anë të $readmemh për të inicializuar memorien nga fajlli **instrMem.mem** që përmban vlera hex dhe për të ekzekutuar kodin si në vijim:

addi $r2, $zero, 512 // 0100 0010 1000 0000 // opcode 2bit [01], rs 3 bit [$zero - 000], rt 3bit [$r2 - 010], [512hex ne bita 10000000]

add $r0, $r1, $2 // 0000 1010 0000 0100 // opcode 00, rs=$r1=001, rt=$2=010 ,rd=$r0=000, funksioni 00100 Dy bitat e pare i marrim nga opcode qe I kemi te cekura ne kerkesen e detyres kurse bitat tjere jane te rezervuar per rs, rt, rt secila nga tre bit te cilat I caktojme sipas shablonit te cekur ne detyre (psh Shablloni: ADD rd, rs, rt) dhe 5 bitat e fundit jane nga funksioni gjithashtu I gatshem ne detyre. Te gjithe keta I ndajme nga 4bit dhe I kthejme ne heksadecimal.

lw $r1, 20($r2) // 1001 0001 0001 0100 // opcode 10, rs=$r2=010, rt=$r1=001, immediate=20=00010100 Procedure e njejte eshte edhe per Lw, SW, kur dy bitat e pare I kemi nga opcode dhe 6bitat tjere, tre nga rs dhe tre nga rt qe I caktojme sipas shablonit te cekur ne detyre (Shablloni: SW rt, (immediate)rs), si dhe vlera immediate e cila kthehet ne numer binar. Te gjithe keta I ndajme nga 4bit dhe I kthejmë ne heksadecimal.

sw $r1, 0($r2) // 1101 0001 0000 0000 // opcode 11, rs=$r2=010, rt=$r1=001, immediate=0=00000000

DataMemory – përmban modulin DataMemory()

Hyrjet: AdresaneHyrje,WriteData(16bitë-sh nga regjistri),MemWrite,MemRead,clk.

Dalja: ReadData.

Data Memory është Read-Write. Hyrje 16 bitëshe për përcaktimin e adresës së fjalës 2 bajtëshe që lexohet/shkruhet. Hyrje 16 bitëshe për fjalën që shkruhet. Dalje 16 bitëshe për fjalën që lexohet.

Me anë të $readmemh për të inicializuar memorien nga fajlli **dataMem.mem** që përmban vlera hex të gjitha zero që tregojnë se është e shprazur.

RegisterFile – përmban modulin RegisterFile()

Hyrjet: RS,RT,RD,WD,WE,clk.

Daljet: RD1,RD2.

Numri i regjistrave do të jetë 8 (regjistri $zero dhe 7 të tjerë për përdorim të përgjithshëm). Regjistrat do të jenë të gjerë 16 bit. Register File ka tre hyrje tre bitëshe për përcaktimin e regjistrave RS, RT, RD. Register file ka një hyrje 16 bitëshe për të

shkruar në regjistrin RD. Register File ka dy dalje 16 bitëshe për të lexuar të dhënat nga regjistrat e përcaktuar në RS dhe RD.

WD - Te dhenat per tu shkruar ne RegisterFile.

Datapath – përmban modulin Datapath()

Hyrjet: clk,RegDst,AluSrc,MemToReg,RegWrite,MemRead,MemWrite,AluOp,PcFill.

Dalja: opcode.

Këto hyrje i fitojm nga Njësia e kontrollit(Control unit).

Datapathi nga instruksoni e mer opcode dhe i'a degron njësis së kontrollit dhe themi se këtë e ka edhe output.

PC një vlerë fillestare që na ndihmon për ta njohur se në cilën adresë me fillu.

Instancimet në Datapath:

InstMemory DPIM(clk,hyrja,dalja) – *dërgojm clokun,pc-n si adres fillestare,dhe presim instruksionin.*

Mux_2to1 M21HyrjeRF(s,RT,RD,z) – *sinjali selektues e selekton në bazë të formatit (R ose I),nëse formati s = 1(R) zgjedh RD nese s = 0 (I) e zgjedh RT.*

RegisterFile RF(RS,RT,RD,WD,RFWe,clk,RD1,RD2) – *i dërgojm si instruksione dhe clokun,fitojm read data 1 dhe read data 2 per format I dhe R.*

alu_control ACU(AluOP, funct, ALU_Cnt*)* – *Në bazë të AluOp e përcaktojm a kemi mbledhje apo operacion sipas funksionit,dhe pastaj përmes ALU_Cnt e kthejmë këtë vlerë në output për ta shfrytzuar te Alu_16bit si input.*

Alu_16bit aluRF(RD1, alu2, cin, ainvert, binvert, ALU_Cnt[1:0], result, cout) – *binvert dhe cin fitohen nëse nga muxi vlera e bitit te bnegate eshte 1 (SUB) këtë vlerë e mer si input nga AluControli.*

DataMemory DM(result,RD2,MemWrite,MemRead,clk,ReadData) – *Nëse jemi ne lë ose së ,rezultati I Alu_16bit është AdresaneHyrje për DataMemory në të dy rastet LW ose SW, nëse jo rezultati ruhet në WD në RegisterFile.*

CPU – *përmban modulin cpu()*

Hyrja: clk

Instancimet:

Datapath CPU_DP(clk, RegDst, ALUSrc, MemToReg, RegWrite, MemRead, MemWrite, ALUop, PcFill, opcode) – *Inputat i deklarojmë si wire përveç clk që e merë si parametër nga moduli i cpu.*

cu CPU_CU(opcode, RegDst, ALUSrc, MemToReg , RegWrite , MemRead , MemWrite, ALUOp) – *opcode e mer si input nda dalja e datapathit.*

4. Fajllat për testim

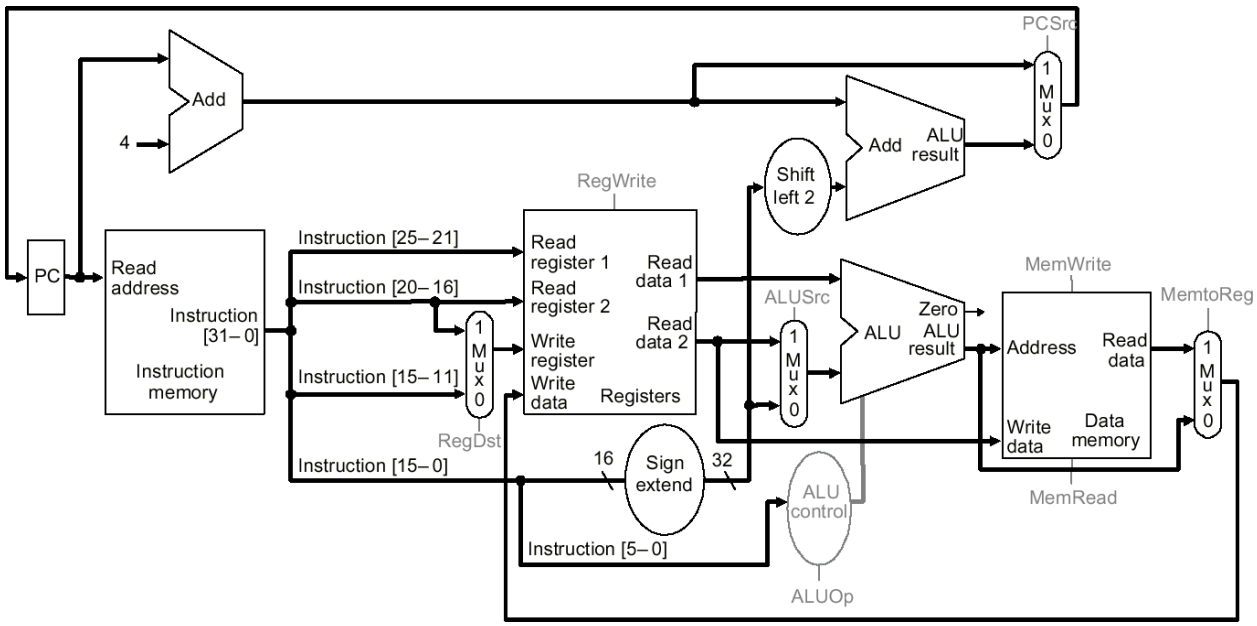
Në folderin **Testbench** kemi fajllat për testimin e ALU 16bitë-she,DataMemory,InstrMemory,RegisterFile,si dhe për fund për CPU e ruajm si regjistër clockun pastaj e instancojm fajllin e CPU në CPU_testbench(clk) si parametër e pranon at clk që e deklarojm në fillim variablën.

Clocku kur mer vleren nga 0 në 1 CPU bëhet 1 dhe bëhet tehu pozitiv instancohen Datapathi dhe Control uniti dhe e mbush PC pastaj vjen InstrMemory e pret tehun pozitiv dhe ekzekutohen njëra pas tjetrës.

4.1 Ekzekutimi

Për ekzekutim mund të përdorim platformën Vivado ose ndonjë softuer tjetër që e përkrah gjuhën Verilog.

Në Vivado fillimisht krijojmë një projekt pastaj me anë të Add Files shtojmë të gjith fajllat nga follderi Kodi dhe testbench pastaj me anë të Run Simulation mund ta bëjmë simulimin e fajllave, psh CPU_tb.



Img. EE380 Single-Cycle Design.

Falënderim për materialet e postuara nga ana juaj, si dhe për çasjen e lehtë në ato materiale me anë të platformës kompjuterika.tk.