|  |  |  |
| --- | --- | --- |
|  | **Universiteti i Prishtinës**  Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike  Arkitektura e Kompjuterëve, 2022/2023  Prof. Valon Raça & Synim Selimi | Adea Lluhani, 220756100072  Agnesa Hulaj, 220756100037  Rozeta Meshi, 210756100104  Yllka Kastrati, 22078100014  Zana Ademi, 220756100008 |

# Detyra e dytë

Dizajnimi i një CPU 24-bitëshe (Single-Cycle)

**Hyrje**

Detyra e dytë përfshin dizajnimin e një CPU-je 16 bitëshe të implementuar në Single Cycle Datapath. Procesi I dizajnimit të Cpu-së përfshin këto dy pjesë kryesore: dizajnimi I DataPath (që përmban një mori komponentesh të lidhura me njëra tjetrën) dhe Control Unit që bën kontrollimin dhe drejtimin e komponenteve tjera.

Datapath pwrbwhet nga kwto komponente:

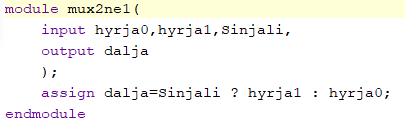
1. Program Counter - regjistër që mban adresën aktuale të instruksionit. Në të gjitha rastet mbledhesi per PC e rrit PC per 2 byte, përvec në rastin kur kemi kërcime të kushtëzuara (kushtëzimet e pakushtëzuara te formatit j nuk përkrahen nga cpu e jonë).
2. Instruction Memory – merr adresën 16 bitëshe të instruksionit nga progam counter-i dhe në dalje na jep instruksionin, të cilin e lexon nga fajlli InstructionMemory.mem.
3. Register File – I merr adresat e rs,rt,rd (nga instruksioni 16 bitësh që Instruction Memory na e jep si dalje)dhe si dalje na jep vlerat që ndodhen në ato adresa. Nga Control Unit e merr sinjalin RegWrite I cili përcakton nëse na lejohet të shkruajmë në regjistra. Poashtu e kemi hyrjen 16 bitëshe Ërite data për të shkruar në reg rd.
4. Dizajnimi I Alu Control – merr si input AluOp nga Control Unit dhe në bazë të saj zgjidhet se cili nga operacionet do kryhet. Në dalje jep 4 bit, një bit për invertimin e hyrjes B, 3 bit për zgjedhjen e operacionit që do kryhet në Alu.
5. Dizajnimi I Alu-së (Njesia Aritmetiko Logjike) 16 bitëshe, e cila do të merr si input dy hyrje 16 bitëshe nga Register File (para ALU-së e vendosim një multiplekser që të zgjedh mes rt dhe vlerës immediate të zgjedhuar) dhe në bazë të sinjalit 4 bitësh që e merr nga Alu Control do performojë njërin nga operacionet përkatëse. Për dizajnimin e Alu-së 16 bitëshe kemi dizajnuar fillimisht Alu 1bitëshe, e pastaj ato I kemi lidhur përmes 16 herë sipas ripple efektit.
6. Data Memory- nga kjo memorie mund të lexojmë dhe të shkruajmë të dhëna. Si hyrje e merr adresën e kalkuluar nga Alu, te e cila do lexojmë ose shkruajmë vlerë të caktuar. Si hyrje e ka edhe WriteData 16 bitëshe për fjalën që do shkruajmë, si dhe një dalje ReadData 16 bitëshe nëse lexojmë vlera nga memoria. Se a na lejohet lexim/shkrimi në memorie e pëcaktojnë sinjalet hyrëse që vijnë nga Control Unit: MemRead dhe MemWrite. Pas DataMemory e kemi një multiplekser që përcakton nëse në regjistra do shkruhet vlera e lexuar nga memoria ose rezultati që del nga Alu.

Ndwrsa Control Unit- nga intruksioni i merr 4 bitat e opcode dhe në bazë të tij kontrollon dhe drejton veprimin e komponenteve tjera.

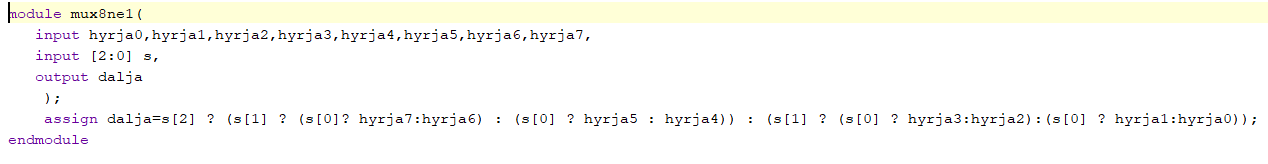
**Dizajni**

Fajllat janë të organizuar në këtë mënyrë:

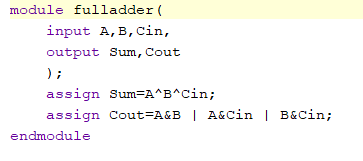
**Mux2ne1:** Multiplekser që merr dy hyrje 1 bitëshe, 1 bit për sinjal dhe zgjedh njërën dalje varësisht prej sinjalit. E kemi përdorur tek alu 1 bitëshe për zgjedhjen mes hyrjes B ose joB varësisht prej sinjalit Bnegate.



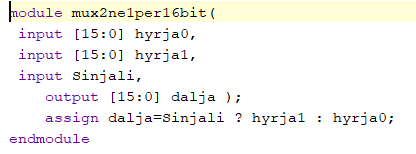
**Mux8ne1:** Multiplekser që merr 8 hyrje dhe zgjedh njërën në dalje, varësisht prej sinjalit tre bitësh. E kemi përdorur për zgjedhjen e operacionit që do kryhet në ALU.



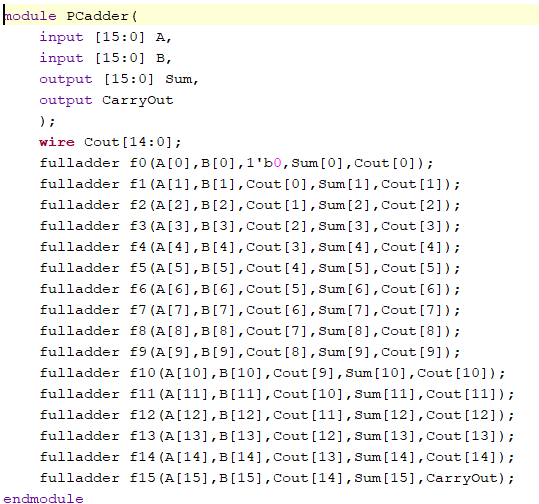
**Mbledhësi 1 bit:** Mbledhësi i plotë merr dy hyrje 1 bitëshe A dhe B, një hyrje 1 bitëshe CarryIn, dhe në dalje jep shumën e dy hyrjeve A dhe B, dhe mbetjen CarryOut. Këtë mbledhës do ta përdorim edhe për kryerjen e zbritjes, në atë rast vetëm duhet invertuar hyrjen B në Alu.



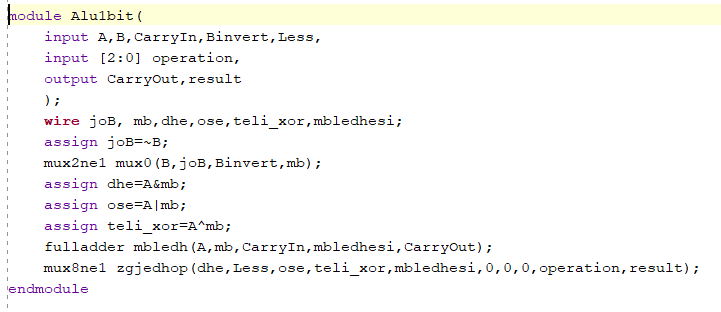
**Mux2ne1per16bit:** Multiplekser që merr dy hyrje 16 bitëshe, 1 bit për sinjal dhe zgjedh njërën dalje varësisht prej sinjalit. E kemi përdorur tek dataPath për zgjedhjen mes hyrjes rt ose vlerës immediate varësisht prej sinjalit AluSrc.

****

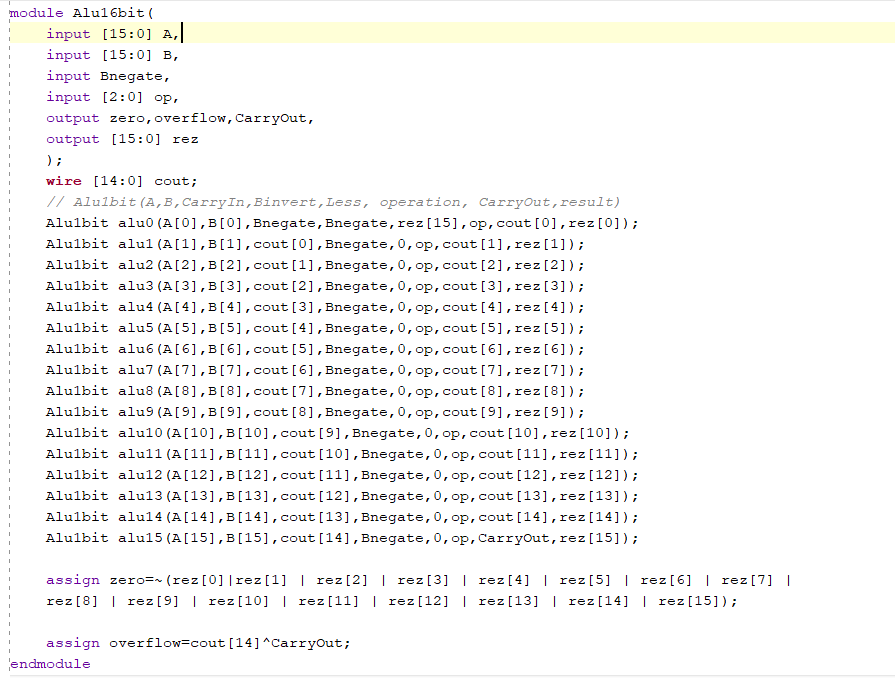
**Mbledhësi 16 bitësh për PC:** Merr dy hyrje 16 bitëshe A dhe B, dhe ka 16 bit për rezultat si dhe një bit për CarryOut. Këtë mbledhës e kemi implementuar në bazë të metodës ripple carry ku CarryOut I mbledhësi të plotë të parë bartet si CarryIn I mbledhësit të plotë 1 bitësh pasardhës.



**ALU 1 bit:** Fillimisht kemi ndërtuar Alu-në 1 bitëshe e cila merr dy hyrje 1-bitëshe A dhe B, hyrjen 1 bit CarryIn. Si sinjale Alu 1 bitëshe I ka 1 bit për Bnegate, dhe 3 bit për të zgjedhur operacionin. Për të zgjedhur se a duhet të invertohet hyrja B apo jo kemi dizajnuar një multiplekser 2 në 1, i cili zgjedh mes B dhe joB, varësisht prej vlerës së Bnegate që e merr si sinjal hyrës. Ndërsa për të zgjedhur operacionin, meqë kemi 3 bit për opearcion nënkupton që mund të kryhen gjithsej 8 operacione (2^3 permutacione), prandaj kemi dizajnuar një multiplekser 8 ne 1. Për të kryer mbledhjen (e përmes saj edhe zbritjen) kemi dizajnuar një modul të vecantë për mbledhës. Për shkak se nuk I kemi përfunduar operacionet shtesë, kur e kemi instancuar multiplekserin, për ato operacione I kemi dërguar 0.



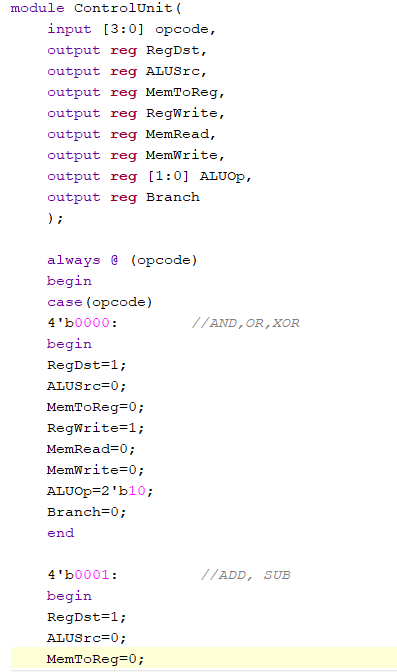
**ALU 16 bit:** Për dizajnimin e Alu-së 16 bitëshe kemi përdorur Alu 1 bitëshe; e kemi instancuar 16 herë. Ky instancim kryhet sipas metodës ripple carry ku CarryOut i alusë paraprake I dërgohet si carryIn alusë pasardhëse. Alu 16 bitëshe merr si input dy hyrje 16 bitëshe A dhe B, hyrjen 1 bitëshe për CarryIn, një dalje 16 bitëshe për rezultat, një dalje 1 bitëshe për overfloë, zero dhe carryout. Biti zero na shërben kur bëjmë krahasime, na tregon se a janë A dhe B të barabarta apo jo.



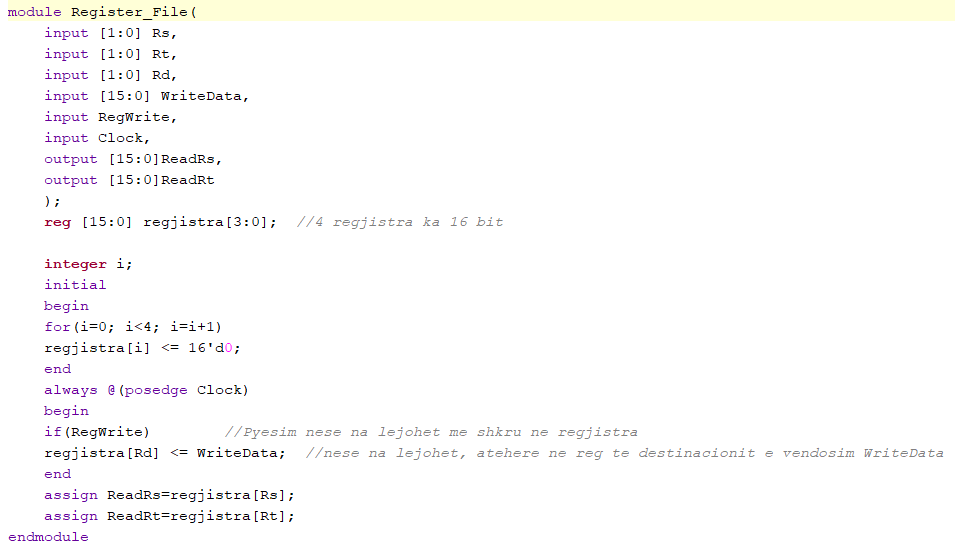
**Alu Control:** merr si input 2 bit për AluOp, 2 bit për funksion dhe 4 bit për opcode, dhe ka daljen 4 bitëshe, ku 1 bit shërben për invertimin e hyrjes B në Alu, dhe 3 bit shërbejnë për përcaktimin e operacionit në multiplekser. Hyrja 2 bitëshe AluOp përcakton se për cfarë kryhet operacioni, prandaj I kemi shqyrtuar rastet për 00 kryhet mbledhje (për kalkulimin e adresës për instruksionet lë e së), 01 kryhet zbritje (për krahasime). Kur AluOp është 01 duhet shqyrtuar sipas funksionit, në MIPS të gjitha instruksionet e formatit R e kanë opcode të njëjtë, dhe lehtë I kemi shqyrtuar sipas funksionit, por në rastin tonë jo të gjitha instruksionet e kanë opcode të njëjtë. Gjatë shqyrtimit se për cfarë do kryhet operacioni sipas funksionit nëse dy instruksione kanë 2 bitat e funksionit të njëjtë e kemi shtuar kushtin shtesë që ato instruksione të shqyrtohen edhe sipas opcode. Ndërsa kur AluOp është 11, shqyrtohen në bazë të opcode. Në këtë pjesë I kemi instruksionet e I formatit ku secila prej tyre vecohen me anë të opcode.



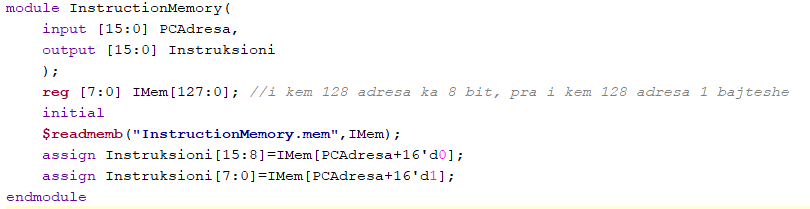
**Control Unit:** Njësia e kontrollit merr opcode 4 bitësh si input, dhe jep daljet që kontrollojnë pjesët e tjera: RegDst (përcakton nëse regjistër i destionacionit do jetë rd apo rt), AluSrc (përcakton nëse si hyrje në Alu do jetë rt apo vlera immediate e zgjeruar në 16 bit), MemToReg(përcakton nëse duhet shkruar nga memoria në regjistra), Memread (përcakton nëse duhet të lexojmë data nga memoria), MemWrite(përcakton nëse duhet të shkruajmë në memorie), AluOp( I dërgohet si hyrje alu kontrollit dhe përcakton se për cfare do kryhet operacioni), Branch(bashkë me bitin zero që del nga Alu 16 bitëshe përcakton nëse duhet të kërcejmë te ndonjë adresë).

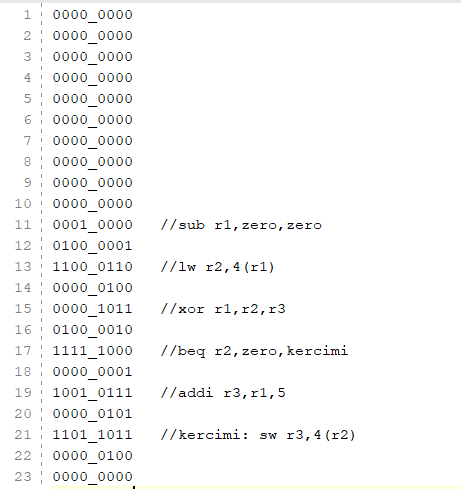


**Register File –** sipas detyrës ne kemi vetëm 4 regjistra (zero, r1, r2, r3) 16 bitësh. Register File I ka tre hyrje dy bitëshe për përcaktimin e rs,rt,rd; një hyrje 16 bitëshe për të shkruar të dhënat në rd; dy dalje 16 bitëshe për leximin e të dhënave në rs e rt. Poashtu nga Control Unit e merr si input sinjalin RegWrite që përcakton se a mund të shkruajmë në register file apo jo.



**Instruction Memory –** merr si hyrje 16 bitëshe nga PC adresën e instruksionit dhe jep në dalje 16 bit per leximin e instruksionit. Instruksionet I kemi paraqitur në një fajll InstructionMemory.mem, të cilën e kemi paraqitur në numra binary dhe e cila është 128 byte. 10 bytes të parë janë të rezervuar kurse ne intruksionet I kemi shkruar nga adresa 10.

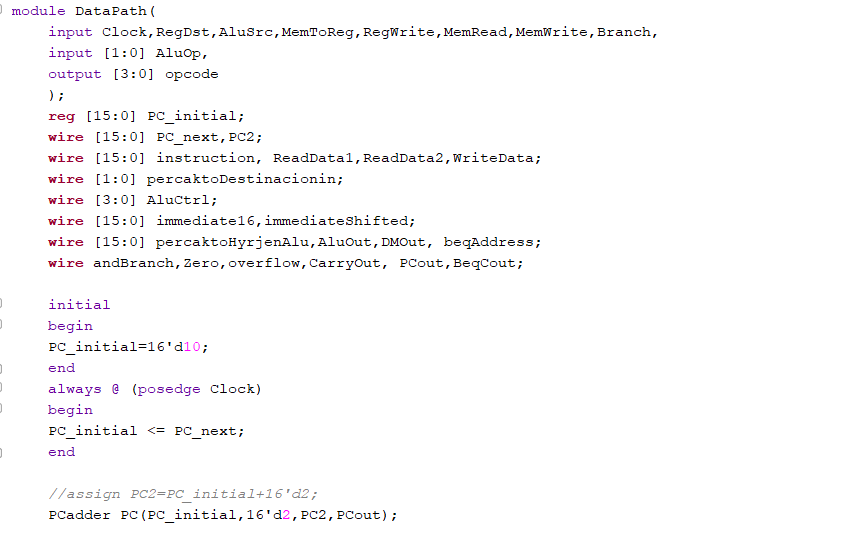


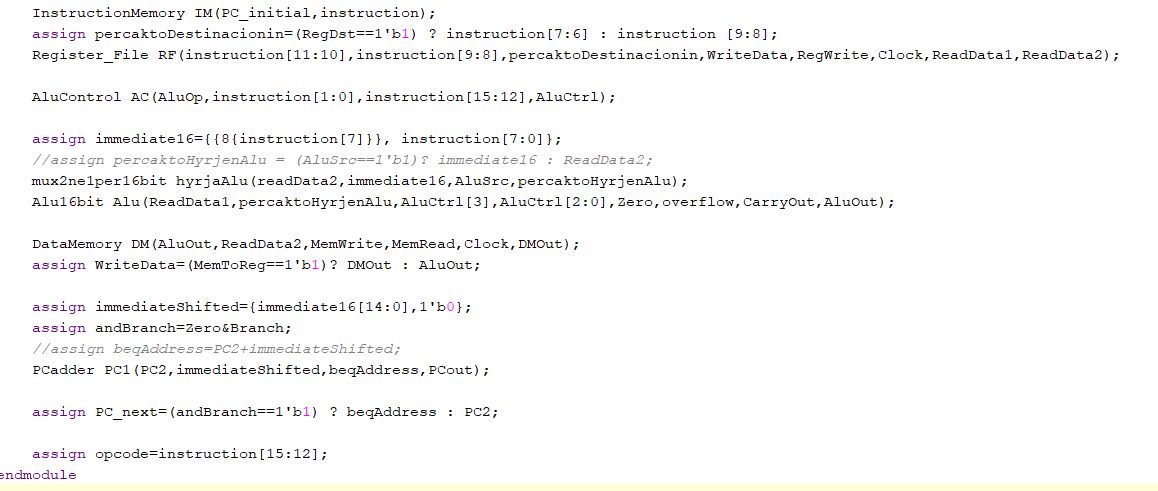


**Data Memory –** si hyrje e kemi adresën 16 bitëshe për përcaktimin e adresës, një hyrje 16 bitëshe për fjalën që shkruhet dhe një dalje 16 bitëshe për fjalën që lexohet. Nga control unit e merr si input sinjalin MemWrite që përcakton nëse lejohet të shkruajmë në memorie; sinjalin MemRead që përcakton nëse kemi nevojë të lexojmë nga memoria apo jo. Edhe këtu e kemi hartën e memories 128 byte, të cilën e kemi paraqitur me anë të numrave binarë, dhe në të cilën përvec qëë mund të lexojmë, mund edhe të shkruajmë në të.

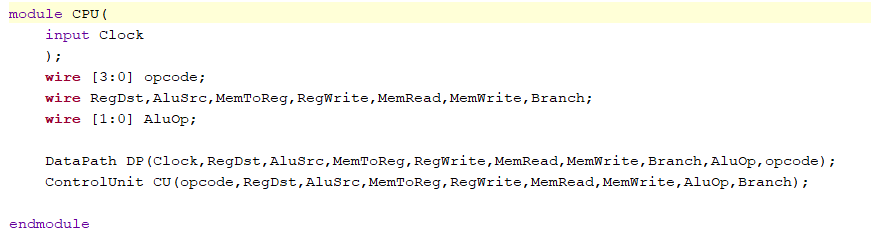


**DataPath -** Paraqet pjesën në të cilën lidhen të gjitha elementet e dizajnuara më sipër (përpos Control Unit). DataPath merr si input daljet e ControlUnit ndërsa jep në dalje 4 bita të opcode të cilat shkojnë si input në Control Unit.



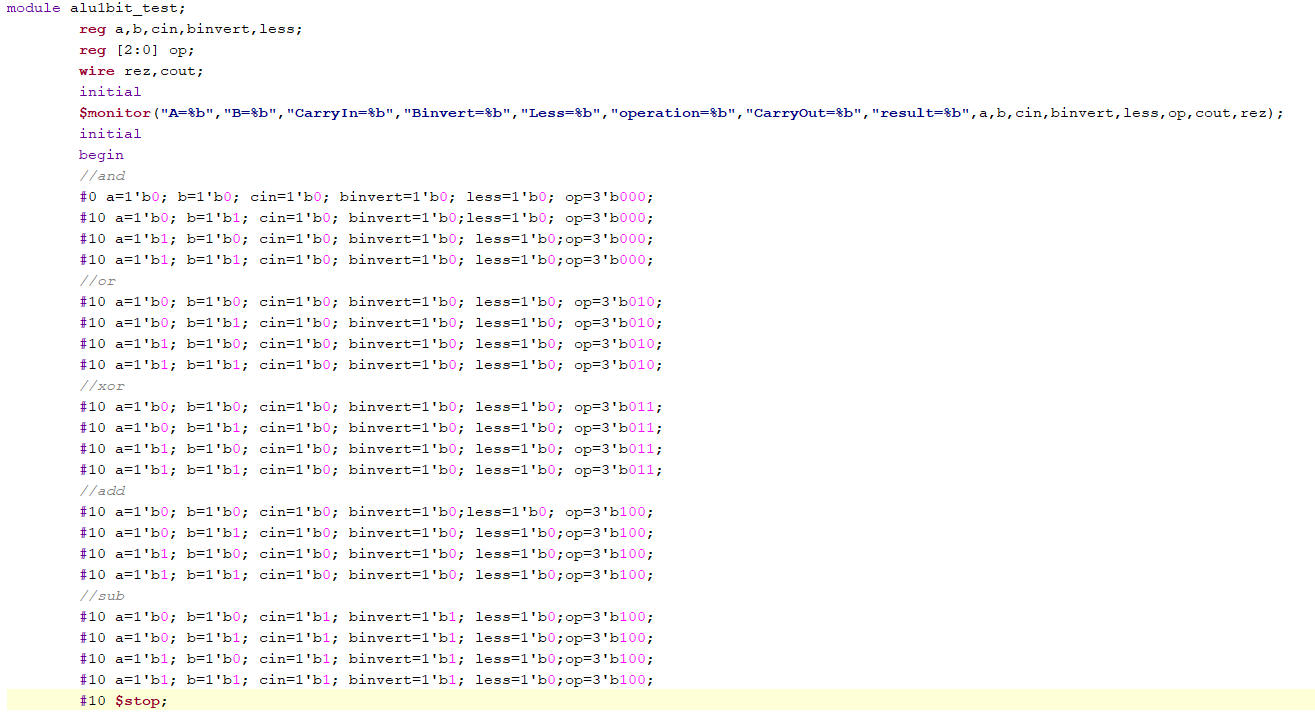


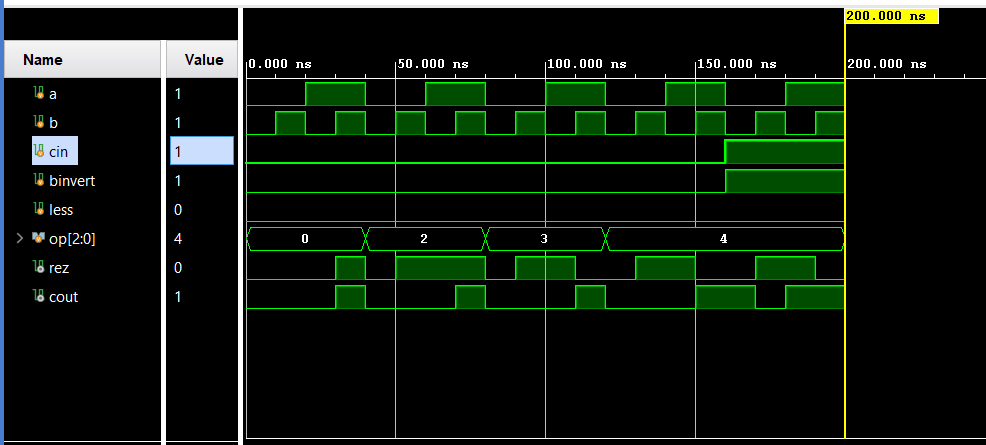
**CPU**- si input e merr vetëm clock-un. Në këtë pjesë instancohen Datapath dhe Control Unit.



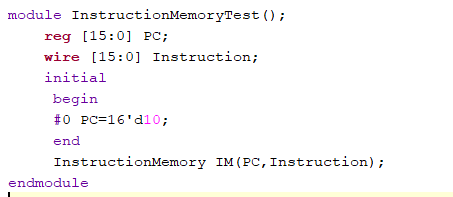
**Testimi**

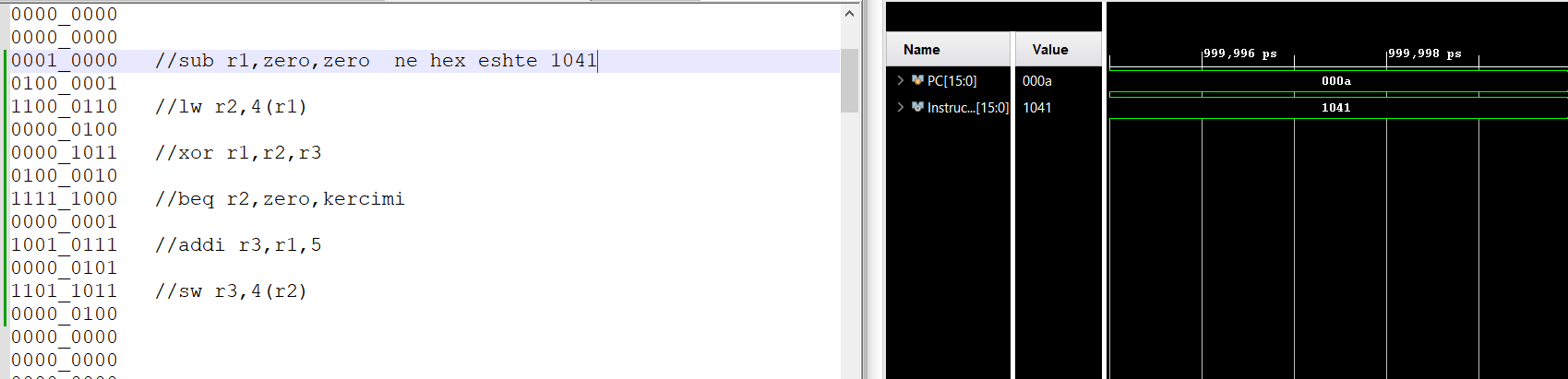
Testimi i Alu-se 1 biteshe:





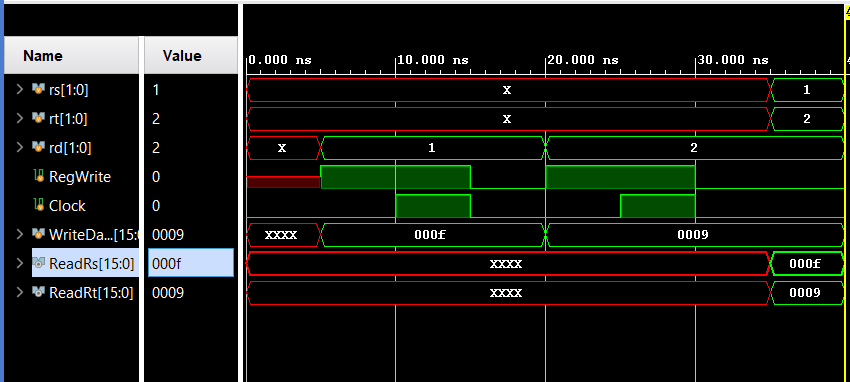
Testimi i Instruction Memory:





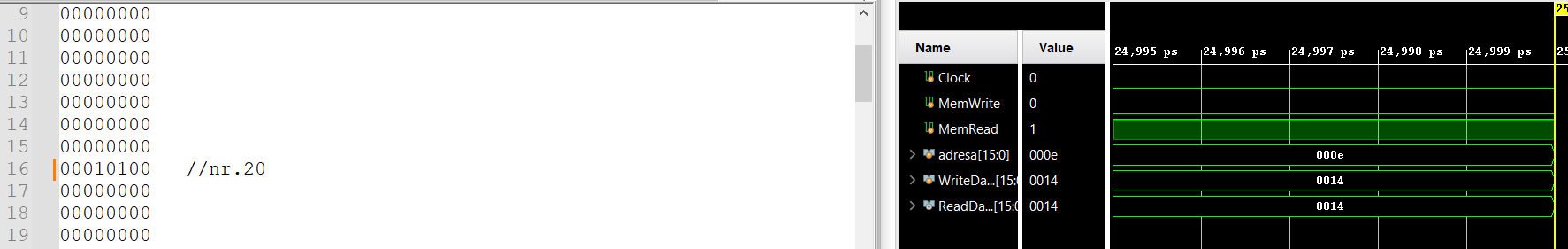
Testimi i Register File:





Testimi i Data Memory:





Testimi i CPU-së:

