Simulation Report

MIPS CPU

**Appendix A**

|  |  |
| --- | --- |
| Students: | Otabek Sherman -326868775 |
|  | Lidor Cohen - 305782732 |

A.1) Fill up the following table describing what happens in each CK cycle in all instructions. You should specify the specific operations that are required for the execution of the instruction.

We filled in the Rtype and j instructions – as examples. We also gave the list of required registers & signals to be mentioned in the table, in the ori instruction line.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| phase | **IF** | **ID** | **EX** | **MEM** | **WB** |
| Instruction |
| Rtype | IR=IMem[PC]  PC= PC+4 | A=GPR[Rs]  B=GPR[Rt]  Active signals:  RegDst=’1’  RegWrite=’1’  ALUOP=”10”  MemToReg=’0’ | ALUOUT = A op B  Rd is chosen:  Rd\_pMEM=Rd\_pEX | ALUOUT\_pWB=  ALUOUT  (ALUOUT is d  elayed 1ck) | GPR[Rd\_pWB]  = ALUOUT\_pWB |
| addi | IR=IMem[PC]  PC=PC+4 | A=GPR[Rs]  Sext\_imm  Active signals:  RegWrite=’1’  RegDst=’0’  ALUSrcB=’1’  MemToReg=’0’  ALUOP=’00’  MemWrite=’0’;  Jal=’0’ | ALUOUT=sext\_imm\_reg;  Sext\_imm\_reg=sext\_imm;  Rd\_pMEM=Rd+pEX; | ALUOUT\_reg=ALUOUT\_output; | GPR\_wr\_data=ALUOUT\_reg\_pWB;  Rd\_pWB=Rd+pMEM=Rt;  GPR[Rt]=ALUOUT\_reg\_pWB; |
| ori | Need to tell what is loaded to IR & PC – the relevant regs. | Again, all regs that are relevant (A, B, sext\_imm, PC in j & branch)  Also – all **active** signals created at the ID phase | All regs that are relevant (ALUOUT, B\_reg\_pMEM, Rd\_pMEM, sext\_imm) | All regs that are relevant (ALUOUT\_reg\_bWB, Rd\_pWB, MDR)  MDR= DMem[adrs ] or  DMem[adrs]=B\_reg\_pMEM | GPR[Rd\_pWB]  = ALUOUT\_pWB |
| lui | IR=IMEM[PC];  PC=PC+4;  Rs=”00000”; | A\_reg=0x00000000;  Sext\_imm<=imm(15 downto 0) & “0000”;  Active Signals:  RegWrite=’1’;  RegDst=’0’;  ALUsrcB=’1’;  MemToReg=’0’;  MemWrite=’0’;  JAL=’0’;  ALUOP=’00’; | Sext\_imm\_reg=sext\_imm  ALUOUT=A+sext\_imm\_reg;  Rd\_pMEM=Rt\_pEX; | ALUOUT\_reg=ALU\_output; | GPR[Rd\_pWB]=ALUOUT\_reg\_pWB;  ALUOUT\_reg\_pWB=ALUOUT\_reg;  Rd\_pWB=Rd\_pMEM; |
| beq | PC=PC+4;  IR=IMem[PC]; | Active Signals:  ALUOP=”01”;  If Rs\_equals\_rt=’1’;then PC\_Source=”01”;  Else PC\_Source=”00’ |  |  |  |
| bne | PC=PC+4;  IR=IMem[PC]; | ALUOP=”01”;  If PC\_source=”00”;  Then  PC\_Plus\_4  Else branch\_adrs |  |  |  |
| lw | PC=PC+4;  IR=IMem[PC]; | A=GPR[Rs];  Sext\_imm=imm;  Active signals:  ALUOp=”00”;  RegWrite=’1’;  MemWrite=’0’;  MemToReg=’1’;  RegDest=’0’;  JAL=’0’; | Sext\_imm\_reg=sext\_imm;  ALUOut=sext\_imm\_reg + A;  Rd\_pMEM=Rt\_pEX; | MDR=DMem[ALUout]; | GPR\_wr\_data=MDR\_reg;  Rd\_pWB=Rd\_pMEM; |
| sw | PC=PC+4;  IR=IMem[PC]; | A=GPR[Rs];  B=GPR[Rt];  Sext\_imm=imm;  Active Signals:  ALUOp=”00”;  RegWrite=”00”;  MemWrite=’1’;  MemToReg=0’;  ALUSrcB=’1’;  JAL=’0’; | Sext\_imm\_reg=sext\_imm;  ALUOut=A+sext\_imm\_reg;  B\_reg\_pMEM=B; | DMEM[ALUOut]=B; |  |
| j | IR=IMem[PC]  PC=PC+4 | PC= jump adrs | nothing | nothing | nothing |
| jal | IR=IMEM[PC];  PC=PC+4; | PC=jump adrs;  B=GPR[Rt];  Active Signals:  ALUOp=”00”;  RegWrite=’1’;  RegDst=’0’;  ALUsrcB=’0’;  MemToReg=’0’;  MemWrite=’0’;  JAL=’1’; | PC\_plus\_4\_pEx=PC\_plus\_4\_pID;  Rd\_pMEM=Rt\_pEX;  JAL\_pEX=JAL; | JAL\_pMEM=JAL\_pEX;  PC\_plus\_4\_pMEM=PC\_plus\_4\_pEX; | JAL\_pWB=JAL\_pMEM;  Rd\_pWB=RD\_pMEM;  If JAL\_pWB=’1’ then GPR\_wr\_data=PC\_plus\_4\_pWB;  GPR[Rd\_pWB]=PC\_plus\_4\_pWB; |
| jr | IR=IMEM[PC];  PC=PC+4; | PC=Rs;  Rd=$0; |  |  |  |

Answer the following questions.

A.2) Describe the changes done in order to support the ORI instruction.

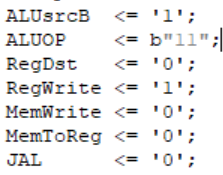
Changes we make in ALU MIPS file to support the ORI instruction:



Changes we make in Fetch Unit file to support the ORI instruction:



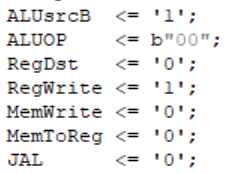
Also we did changes in the active control signals of the Instruction Decode step:



A.3) Describe the changes done in order to support the LUI instruction.

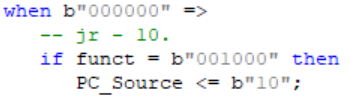


Active Signals:



A.4) Describe the changes done in order to support the JR instruction.

We changed the the Fetch unit. The PC source decoder detecting the Jr instruction by addressing the bits of funct:



In the Fetch unit file the jr\_adrs gets the value jr\_adrs\_in:

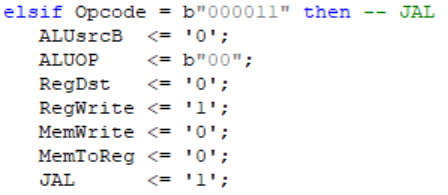


In Top file the jr\_addr gets the value GPR\_rd\_data1;

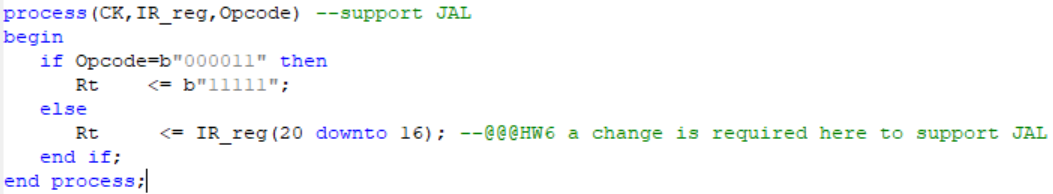


A.5) Describe the changes done in order to support the JAL instruction.

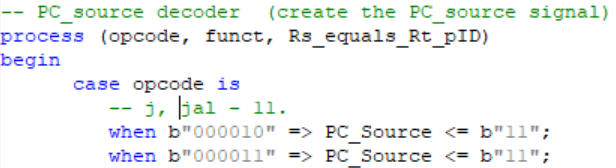
We changed the Instruction Decode active control signals to support the JAL instruction:



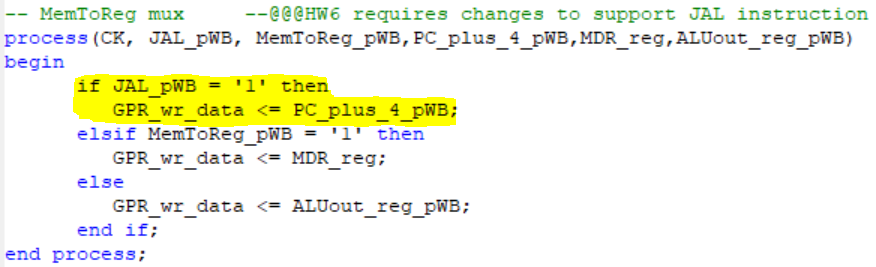
In Top file we set the Rt value to “11111” to be sure that we write PC\_plus\_4 to $31:



In the Fetch unit file the PC\_Source gets the value “11” to use the jump address as input to the PC Source:



In the Top file we inserting the PC\_plus\_4 to the GPR\_wr\_data to allow the inserting of the “PC+4” into the GPR by GPR\_wr\_data



In your answers, besides stating the reasoning in detail, show the relevant VHDL code sections to better explain your answers.