BYOC\_HW6 Simulation 1:

Tsahi Barshevsky 311334544

Kostya Lokshin 310765821

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.

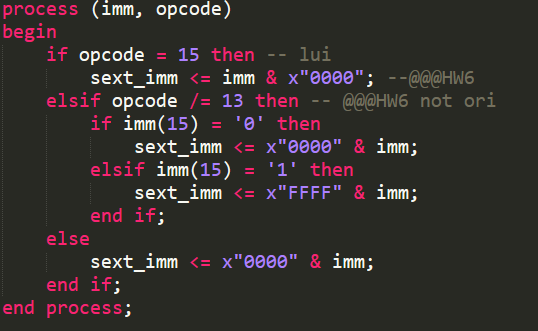
We filled in the Rtype and j instructions – as examples. We also gave the list of required registers & signals to be mentioned in the table, in the ori instruction line.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 delayed 1ck) | GPR[Rd\_pWB]  = ALUOUT\_pWB |
| addi | IR=IMem[PC]  PC= PC+4 | A=GPR[Rs]  B=imm  ALUsrcB <= '1'  RegWrite <= '1' | ALUOUT = A + B | ALUOUT\_pWB=  ALUOUT | GPR[Rd\_pWB]  = ALUOUT\_pWB |
| ori | IR=IMem[PC]  PC= PC+4. | A=GPR[Rs]  B=imm  ALUsrcB <= '1'  ALUOP <= b"11"  RegWrite <= '1' | ALUOUT = A or B  All regs that are relevant (ALUOUT, B\_reg\_pMEM, Rd\_pMEM, sext\_imm) | ALUOUT\_pWB=  ALUOUT | GPR[Rd\_pWB]  = ALUOUT\_pWB |
| lui | IR=IMem[PC]  PC= PC+4 | A=imm  B=GPR[Rt]  ALUsrcB <= '1'  RegWrite <= '1' | ALUOUT = imm<<16 | ALUOUT\_pWB=  ALUOUT | GPR[Rd\_pWB]  = ALUOUT\_pWB |
| beq | IR=IMem[PC]  PC= PC+4 | PC= branch\_adrs  ALUOP <= b"01" | nothing | nothing | nothing |
| bne | IR=IMem[PC]  PC= PC+4 | PC= branch\_adrs  ALUOP <= b"01" | nothing | nothing | nothing |
| lw | IR=IMem[PC]  PC= PC+4 | ALUsrcB <= '1'  MemToReg <= '1'  RegWrite <= '1' | Rt and Rd registes delayed  By 1 ck | MDR= DMem[adrs ] | GPR[Rd\_pWB]  = ALUOUT\_pWB |
| sw | IR=IMem[PC]  PC= PC+4 | ALUsrcB <= '1'  MemWrite <= '1' | Rt and Rd registes delayed  By 1 ck | DMem[adrs]=B\_reg\_pMEM | nothing |
| j | IR=IMem[PC]  PC=PC+4 | PC= jump adrs | nothing | nothing | nothing |
| jal | IR=IMem[PC]  PC= PC+4 | PC= jump adrs  RegWrite <= '1'  JAL <= '1' | nothing | nothing | nothing |
| jr | IR=IMem[PC]  PC= PC+4 | PC= jr\_adrs | nothing | nothing | nothing |

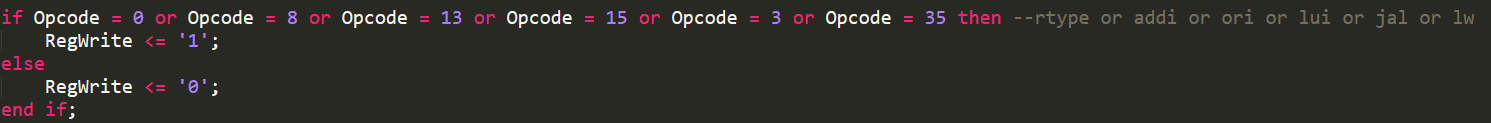
Answer the following questions.

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

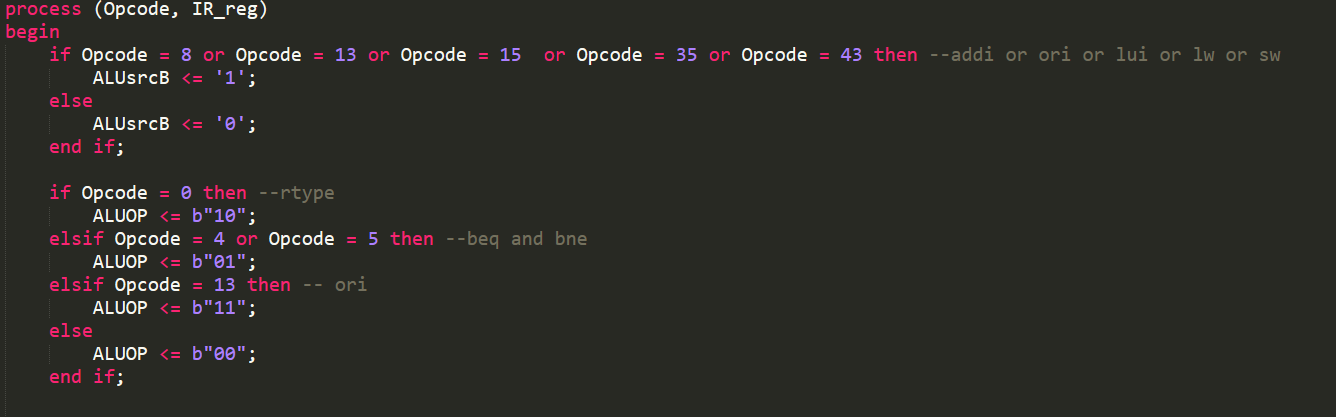
* ביטול מריחת סימן.



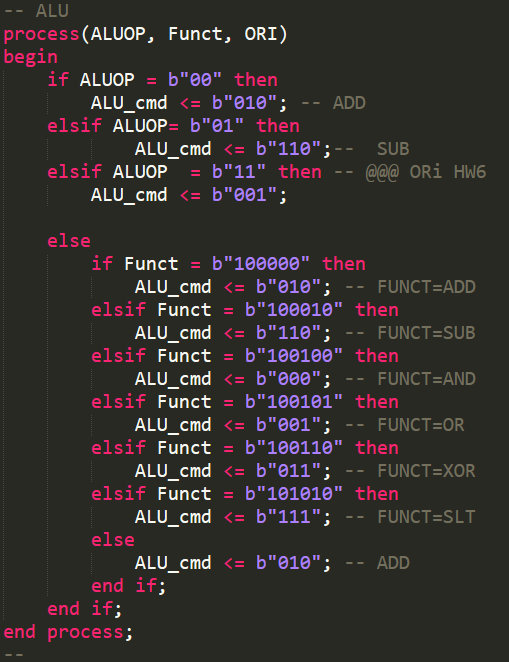
* regwrite:



* aluop ו-aluscrB ובחירת מקור האיבר השני לALU:

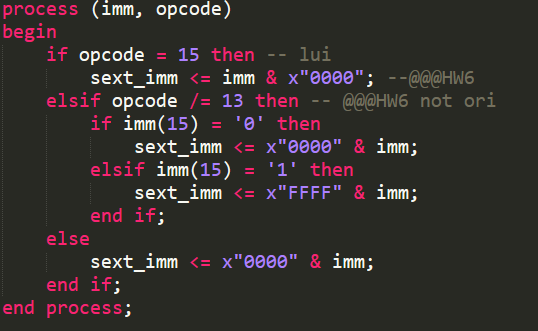


* שינויים בקובץ הalu לתמיכה בפקודה:

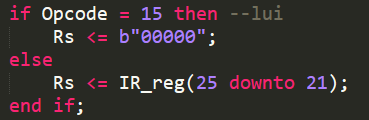


A.3) Describe the changes done in order to support the LUI instruction.

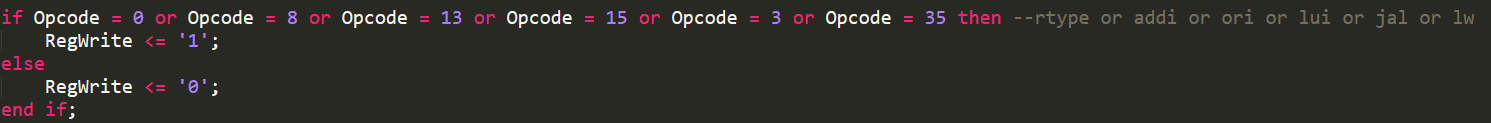
* שינוי ערך הsext imm



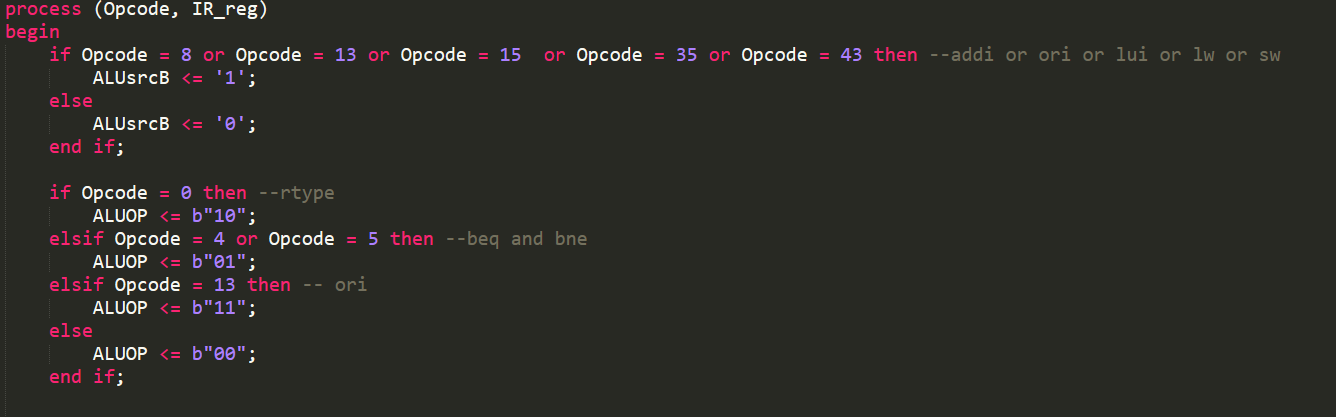
* איפוס הrs.



* regwrite



* alusrcB



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

* יצרנו משתנה חדש ככניסה לfetch unit בשם jr\_adrs\_in .

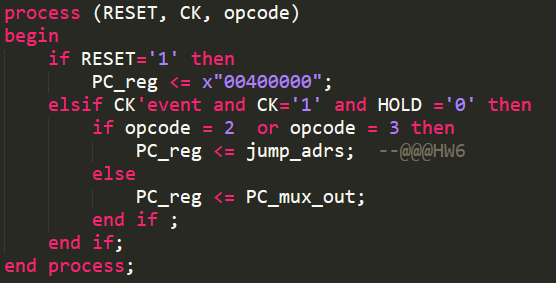


* בtop משתנה זה מקבל את הערך שלו מgpr\_read\_data1.

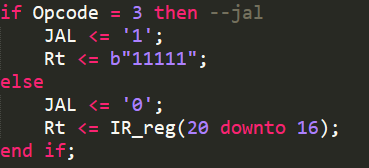


A.5) Describe the changes done in order to support the JAL instruction.

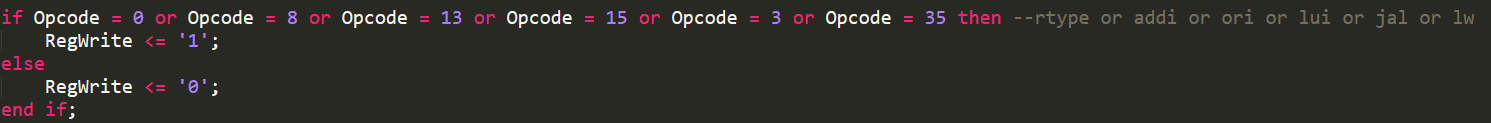
* הpc register מקבל את הכתובת הרלוונטית לקפיצה.



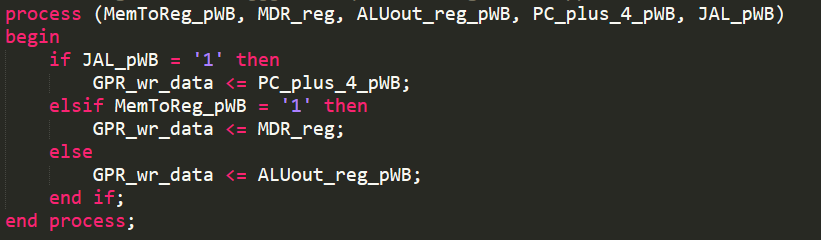
* העברנו החוצה מה fetch unit את pc +4 ו"שרשרנו" אותו ע"ב רגיסטרים לשלב הWB.
* יצירת "דגל" והדלקתו ו "אילוץ rt " לערך 31 :



* regwrite



* שינוי הmux כך שיקבל את ה- pc + 4 המעודכן:



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