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.

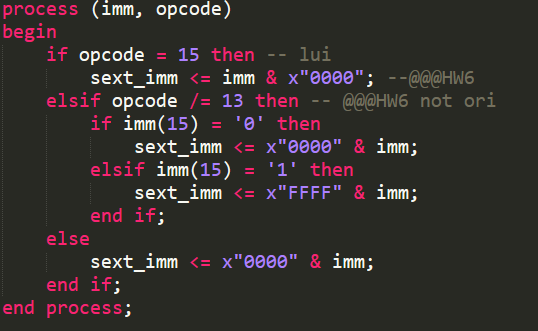
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| phase | **IF** | **ID** | **EX** | **MEM** | **WB** |
| Instruction |
| Rtype | IR=IMem[PC]  PC= PC+4 | A=GPR[Rs]  B=GPR[Rt]  Active signals:  RegDst=’1’  RegWrite=’1’  ALUOP=”10”  MemToReg=’0’ | ALUOUT = A op B  Rd is chosen:  Rd\_pMEM=Rd\_pEX | ALUOUT\_pWB=  ALUOUT  (ALUOUT is delayed 1ck) | GPR[Rd\_pWB]  = ALUOUT\_pWB |
| addi | 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 |
| 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 | 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 |
| beq | IR=IMem[PC]  PC=PC+4 | PC= jump adrs  GPR\_rd\_data1\_wt\_fwd = ALUout\_reg  Rs\_equals\_Rt = '1' | Nothing | nothing | Nothing |
| bne | IR=IMem[PC]  PC=PC+4 | PC= jump adrs  GPR\_rd\_data1\_wt\_fwd = ALUout\_reg  Rs\_equals\_Rt = '1' | Nothing | nothing | Nothing |
| lw | IR=IMem[PC]  PC=PC+4 | ALUsrcB <= '1'  MemToReg <= '1'  RegWrite <= '1' | ALUsrcB\_pEX <= ALUsrcB  Funct\_pEX <= Funct;  ALUOP\_pEX <= ALUOP  RegDst\_pEX <= RegDst  RegWrite\_pEX <= RegWrite  MemWrite\_pEX <= MemWrite  MemToReg\_pEX <= MemToReg | ALUout\_reg <= ALU\_output  RegWrite\_pMEM <= RegWrite\_pEX;  MemToReg\_pMEM <= MemToReg\_pEX;  MemWrite\_pMEM <= MemWrite\_pEX; | ALUout\_reg\_pWB <= ALUout\_reg;  GPR\_wr\_data <= MDR\_reg  RegWrite\_pWB <= RegWrite\_pMEM;  MemToReg\_pWB <= MemToReg\_pMEM; |
| sw | IR=IMem[PC]  PC=PC+4 | ALUsrcB <= '1'  MemWrite <= '1' | ALUsrcB\_pEX <= ALUsrcB  Funct\_pEX <= Funct;  ALUOP\_pEX <= ALUOP  RegDst\_pEX <= RegDst  RegWrite\_pEX <= RegWrite  MemWrite\_pEX <= MemWrite  MemToReg\_pEX <= MemToReg | ALUout\_reg <= ALU\_output  RegWrite\_pMEM <= RegWrite\_pEX;  MemToReg\_pMEM <= MemToReg\_pEX;  MemWrite\_pMEM <= MemWrite\_pEX; | ALUout\_reg\_pWB <= ALUout\_reg;  GPR\_wr\_data <= ALUout\_reg\_pWB  RegWrite\_pWB <= RegWrite\_pMEM;  MemToReg\_pWB <= MemToReg\_pMEM; |
| j | IR=IMem[PC]  PC=PC+4 | PC= jump adrs | Nothing | nothing | Nothing |
| jal | IR=IMem[PC]  PC=PC+4 | PC= jump adrs | JAL\_pEX <= JAL | JAL\_pMEM <= JAL\_pEX | JAL\_pWB <= JAL\_pMEM |
| jr | IR=IMem[PC]  PC=PC+4 | PC= jump adrs  GPR\_rd\_data1\_wt\_fwd = ALUout\_reg  jr\_address = GPR\_rd\_data1\_wt\_fwd | Nothing | nothing | Nothing |

Answer the following questions.

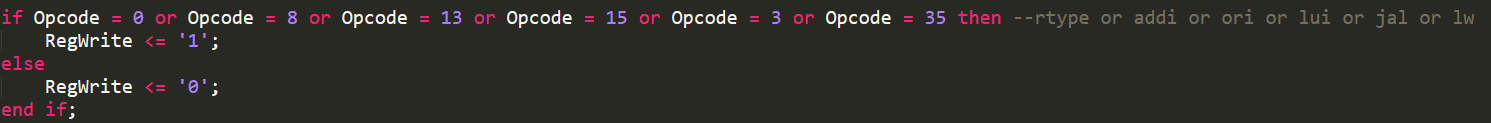
A.2) Describe the changes done in order to support the ORI instruction.

**A2 – שינויים דרושים עבור הוספת ori:**

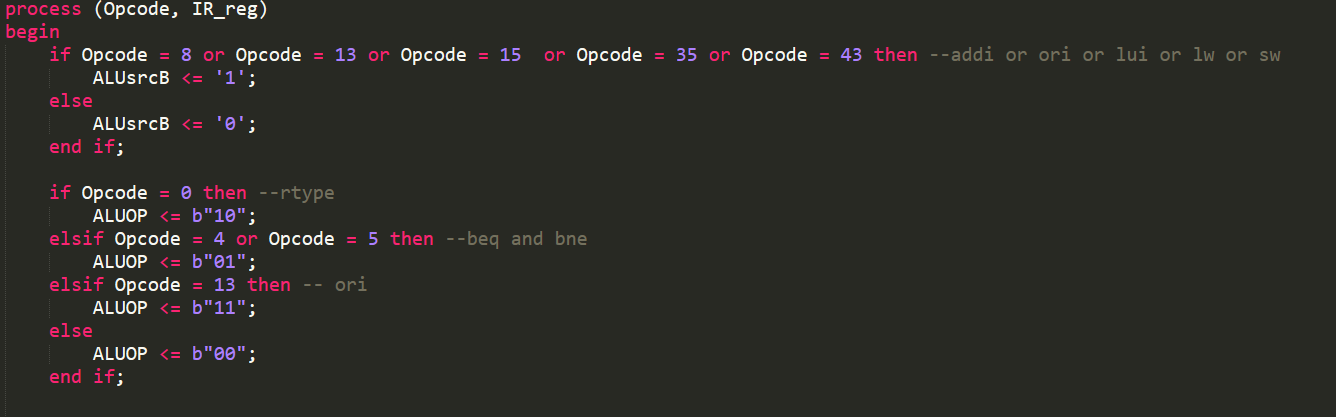
* להוספת הפקודות נדרשנו לשנות את הfetch unit של הפרוייקט וכן חלקים בtop כמובן.
* שינוי תנאי כניסה לsext imm לוודא שלא מדובר בפקודת ori כי אם כן אז sext\_imm = imm ללא שינוי.



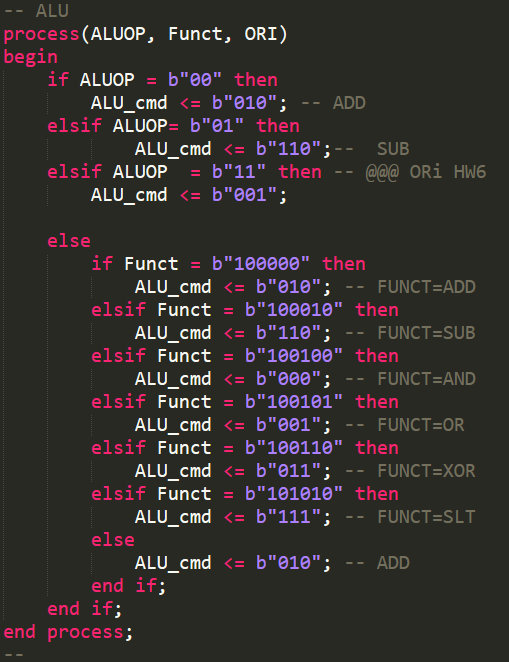
* הדלקת rewrite:



* הדלקת aluop ו-aluscrB:



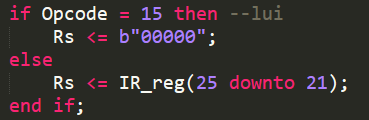
* שינויים בקובץ הalu לתמיכה בפקודה:



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

**A3 – שינויים דרושים עבור הוספת lui**

* שינוי ערך הsext imm כך שאם lui אז מבוצע באופן שונה. (קוד מופיע בהסבר של ori).
* איפוס הrs.



* הדלקת regwrite (קוד מופיע גם בori)
* הדלקת alusrcB (קוד מופיע גם בori)

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

**A4 – שינויים דרושים עבור הוספת jr:**

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

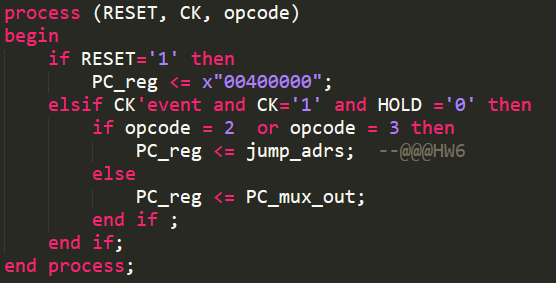
jr\_adrs <= jr\_adrs\_in; --new input for part 6

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

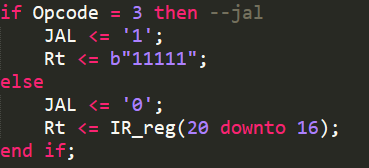
A.5) Describe the changes done in order to support the JAL instruction.

**A5 – שינויים דרושים עבור הוספת jal**

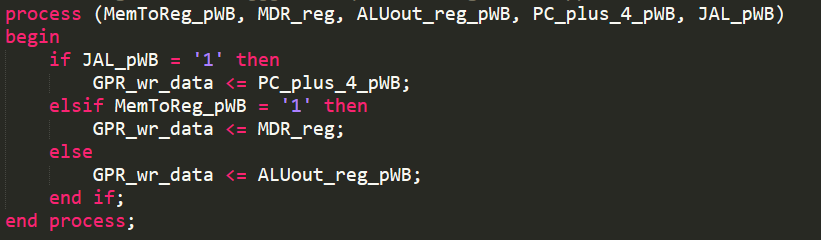
* הpc register מקבל את הכתובת הרלוונטית לקפיצה.



* העברנו החוצה מה fetch unit את pc +4 ו"שרשרנו" אותו ע"ב רגיסטרים לכל השלבים השונים.
* יצירת "דגל" בתכנית הראשית והדלקתו בopcode הרלוונטי
* "אילוץ rt " לערך 31 :



* הדלקת regwrite (קוד מופיע גם בori)
* שינוי הmux כך שיקבל את ה- pc + 4 המעודכן ויוציא את הערך הנכון:



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