CSC 343, Spring 2017

Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report due date: May 3, 2017 by 11:59 PM

<u>Objective:</u>

Design single cycle CPU based on the MIPS instruction set architecture, as it was described in the class and also described in the textbook.

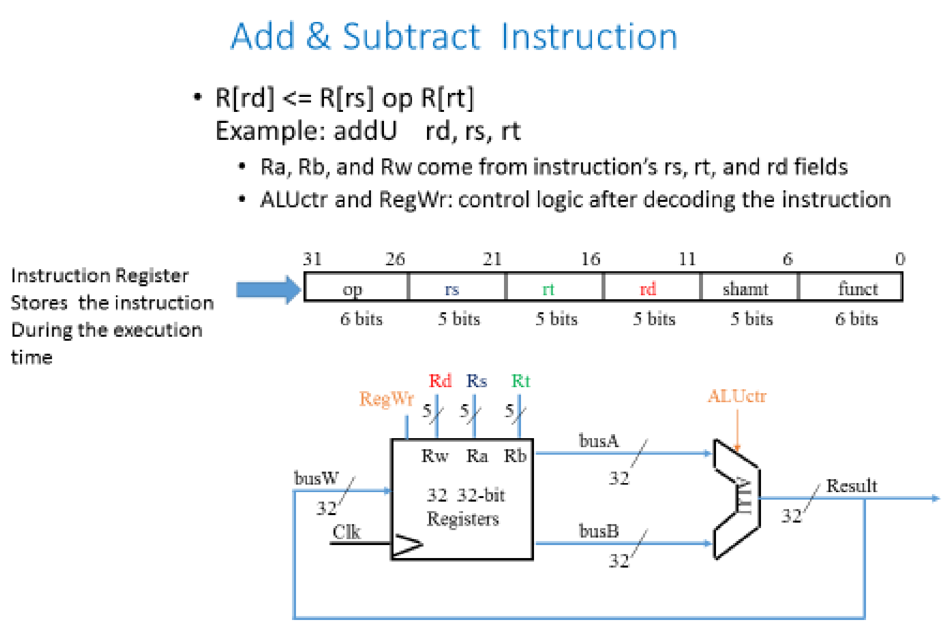
The instructions that you need to implement are of type **R** and **I type.**

How to test your design:

1. **Input, using switches on the board,**
   MACHINE INTRUCTIONS to memory on the board.
2. **Input, using switches on the board,**
   Data to memory on the board.
3. **Load the address of the first instruction to PC register.**
4. **Start execution using keys on the board as a clock. You can also use very low frequency clock (1 cycle per second).**
5. **Display the result of computation on 7 segment display.**
6. **Your program should perform some arithmetic operations on the data. E,g, loop that adds ten numbers.**

Part I ADD/SUB Unit with register file

Design, simulate, verify correctness, and implement on DE 2board the add/sub unit Shown in the figure below:



Initialization:

1. Input operand 1using switches on DE-2 board to a specified register (hint: use the same procedure as in previous lab)

Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report due date: May 3, 2017 by 11:59 PM

2. Input operand 2using switches on DE-2 board to a specified register (hint: use the same procedure as in previous lab)
3. Input 32 bit MIPS R-TYPE instruction (ADD or SUB) using switches on DE-2 board to an INSTRUCTION REGISTER (hint: use the same procedure as in previous lab)

Execution:

1. Press key to start the execution
2. Press another key to display the result on seven segment display.

For this lab you may use a new board DE-2_115 (if you have one). The FPGA device on this board is **CYCLONE IV E : EP4CE115FC7**

Two Options for 3 Ported Register File design

1. YOU CAN USE THIS VHDL **CODE for 3 PORTED register file as an example:**

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
 LIBRARY altera_mf;
 USE altera_mf.all;

 ENTITY ram3port IS
     PORT
     (
         clock       : IN STD_LOGIC ;
         data        : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
         rdaddress_a   :  IN  STD_LOGIC_VECTOR  (4  DOWNTO  0);
         rdaddress_b   :  IN  STD_LOGIC_VECTOR  (4  DOWNTO  0);
         wraddress    : IN STD_LOGIC_VECTOR (4 DOWNTO 0); wren
         : IN STD_LOGIC  := '1';
         qa      : OUT STD_LOGIC_VECTOR (31 DOWNTO 0);
         qb      : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)


     );

 END ram3port;

 ARCHITECTURE SYN OF ram3port IS
     SIGNAL sub_wire0    : STD_LOGIC_VECTOR (31 DOWNTO 0);
     SIGNAL sub_wire1    : STD_LOGIC_VECTOR (31 DOWNTO 0);
     COMPONENT alt3pram

     GENERIC (
         indata_aclr     : STRING;
         indata_reg      : STRING;
         intended_device_family      : STRING;
         lpm_type        : STRING;
         outdata_aclr_a      : STRING;
         outdata_aclr_b      : STRING;
         outdata_reg_a       : STRING;
         outdata_reg_b       : STRING;
         rdaddress_aclr_a        : STRING;
```

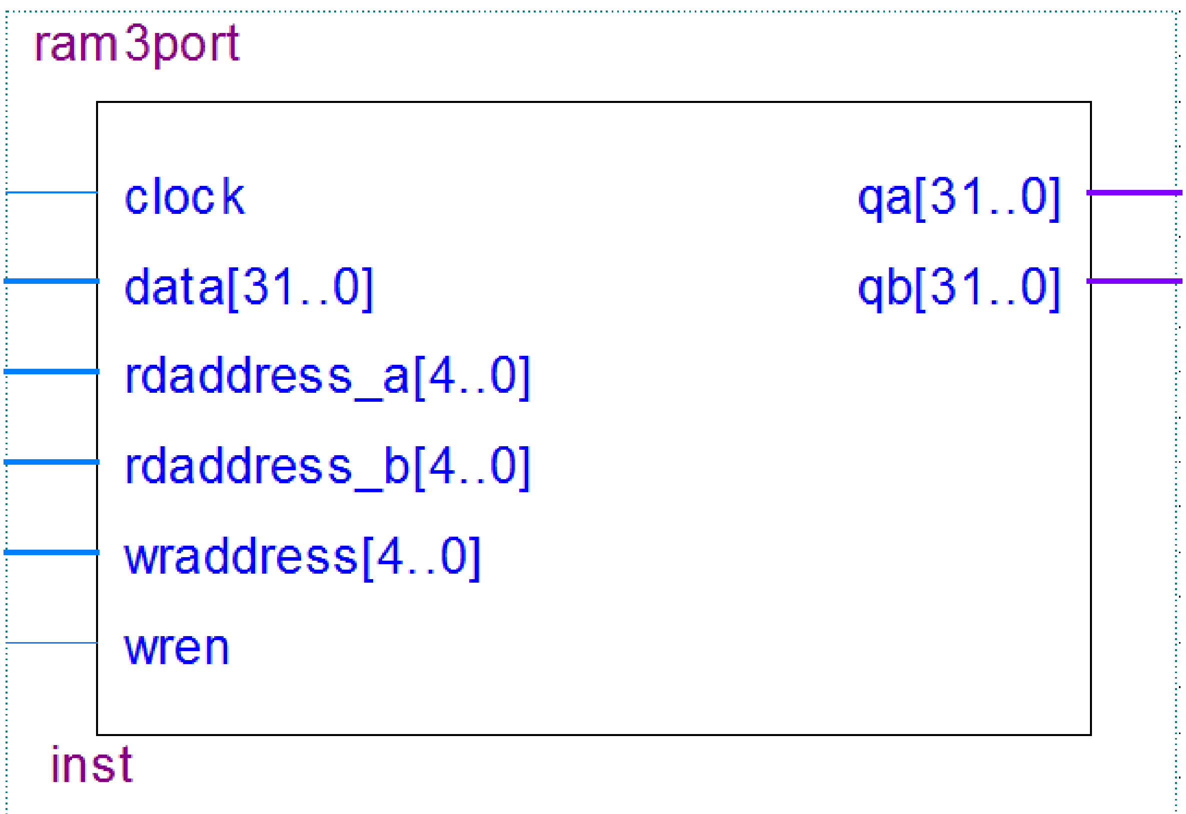
Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report due date: May 3, 2017 by 11:59 PM

```
        rdaddress_aclr_b        : STRING;
        rdaddress_reg_a     : STRING;
        rdaddress_reg_b     : STRING;
        rdcontrol_aclr_a        : STRING;
        rdcontrol_aclr_b        : STRING;
        rdcontrol_reg_a     : STRING;
        rdcontrol_reg_b     : STRING;
        width       : NATURAL;
        widthad     : NATURAL;
        write_aclr      : STRING;
        write_reg       : STRING

); PORT (
qa  : OUT STD_LOGIC_VECTOR (31 DOWNTO 0);
outclock    : IN STD_LOGIC ;
qb  : OUT STD_LOGIC_VECTOR (31 DOWNTO 0); wren    : IN STD_LOGIC ;
inclock : IN STD_LOGIC ;


                : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
        rdaddress_a : IN STD_LOGIC_VECTOR (4 DOWNTO 0); wraddress
        : IN STD_LOGIC_VECTOR (4 DOWNTO 0);
        rdaddress_b : IN STD_LOGIC_VECTOR (4 DOWNTO 0)

);
END COMPONENT;
 BEGIN
     qa    <= sub_wire0(31 DOWNTO 0);
     qb    <= sub_wire1(31 DOWNTO 0);
alt3pram_component : alt3pram

GENERIC MAP (
    indata_aclr => "OFF", indata_reg
    => "INCLOCK",
    intended_device_family => "Stratix II", lpm_type =>
    "alt3pram",
    outdata_aclr_a => "OFF", outdata_aclr_b
    => "OFF", outdata_reg_a => "OUTCLOCK",
    outdata_reg_b => "OUTCLOCK",
    rdaddress_aclr_a => "OFF",
    rdaddress_aclr_b => "OFF",
    rdaddress_reg_a => "INCLOCK",
    rdaddress_reg_b => "INCLOCK",
    rdcontrol_aclr_a => "OFF",
    rdcontrol_aclr_b => "OFF",
    rdcontrol_reg_a => "UNREGISTERED",
    rdcontrol_reg_b => "UNREGISTERED", width
    => 32,
    widthad => 5, write_aclr =>
    "OFF", write_reg =>
    "INCLOCK"
)
PORT MAP (
```

```
outclock => clock, wren
=> wren, inclock =>
clock, data => data,
rdaddress_a => rdaddress_a,
wraddress => wraddress,
rdaddress_b => rdaddress_b, qa =>
sub_wire0,
qb => sub_wire1

);
END SYN;
```

REMARK: MAKE SURE that the file name is ***ram3port (the same as entity).***
Create a symbol for your use.



END REGISTER FILE EXAMPLE

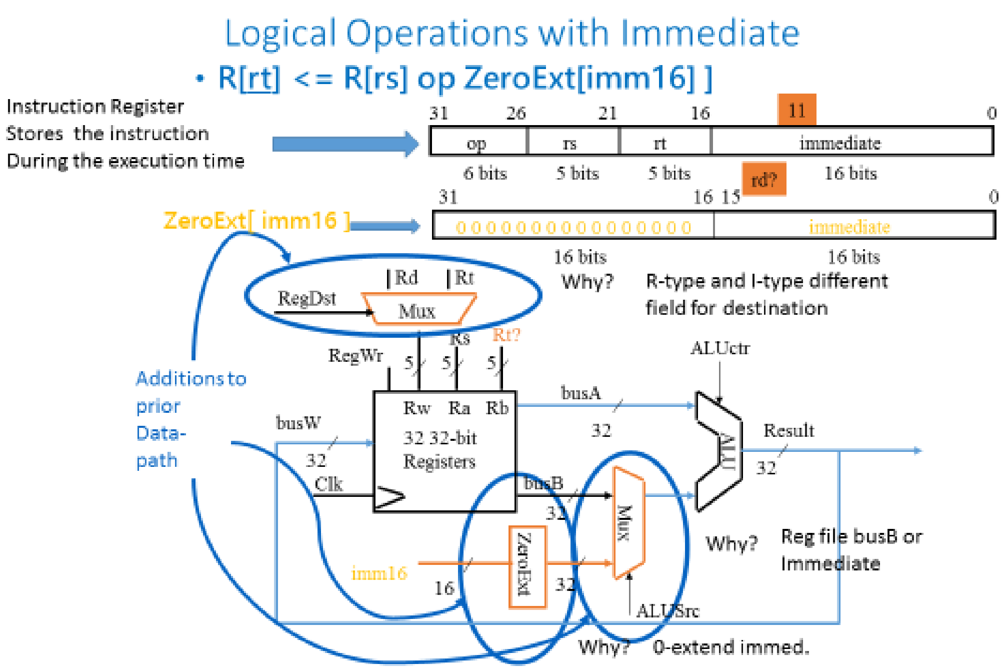
2. Alternatively, you can create your own 3 ported register file using two LPM modules , as you have done in a previous lab.

Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report due date: November 23, 2016 by 11:59 PM

PART II BITWISE OPERATIONS

Extend your design in PART I to include **ORI** instruction, and then all bitwise operations from your previous lab.



Initialization steps are the same as in Part I.
You may extend this design by adding BITWISE OPERATIONS your previous lab.

PART III BEQ instruction, I Type format

```
Extend your design to include
    beq   rs, rt, imm16
```

You will need to add another register-Instruction Pointer **PC**

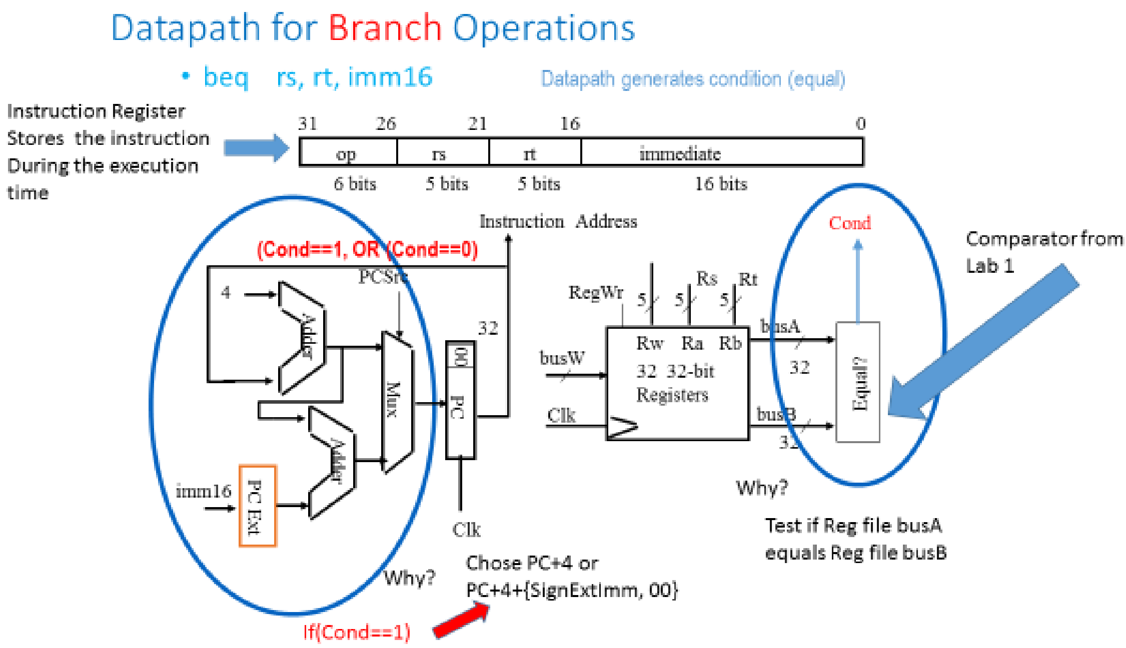
Equal <= (R[rs] == R[rt]) Calculate the branch condition
if (Equal) Calculate the next instruction's address
PC <= PC + 4 + { SignExt(imm16) , 2b00 }

CSC 343, Spring 2016

Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report due date: November 23, 2016 by 11:59 PM

else
PC <= PC + 4



PART IV: Extend your design to include Integer multiply and divide instructions.