Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report, video, demo on DE-2 board due date: May 13, 2018 by 12:00 PM

Objective:

This is a integration project of all your previous labs.
Design single cycle 32 bit CPU based on the 32 bit MIPS instruction set architecture, as it was described in the class and also described in the textbook. The addresses are 32 bits, the data elements are 32 bit integers, the instruction length is 32 bits.

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

HOW to TEST:

- **Input machine instructions you want to execute into SSRAM memory on the board using switches. You have to store the address of the first instruction in a register.**
- **Input data to SSRAM using the method using switches on the board. You have to store the address of the first data element in your data segment in a register.**
- **Load the first instruction from SSRAM to IR-Instruction Register.**
- **Start executing the code step by step, instruction after instruction.**
- **Display the final result using Seven Segment display.**

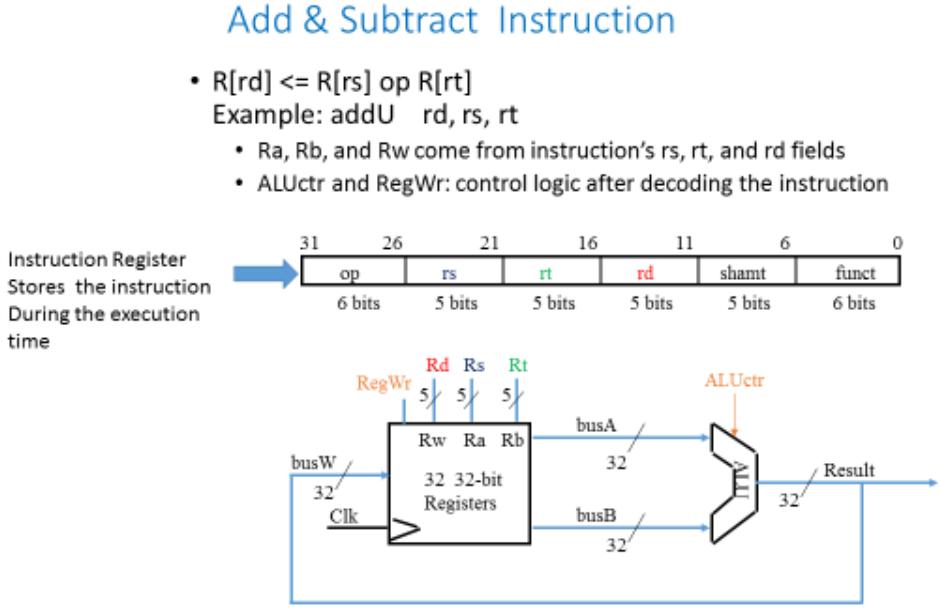
CSC 343, SPRING 2019

Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report, video, demo on DE-2 board due date: May 13, 2018 by 12:00 PM

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:



I.A Initialization:

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

I.B 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 II on DE-2 70, or CYCLONE IV E : EP4CE115FC7 on DE2- 115.**

Repeat Parts I.A, I.B reading instructions and data from SSRAM on the board..

Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report, video, demo on DE-2 board due date: May 13, 2018 by 12:00 PM

YOU HAVE TO DESIGN 3 PORTED REGISTER FILE USING LPM MODULES.

YOU CAN LOOK INTO EXAMPLE VHDL CODE for 3 PORTED register file as an example (Note: this code is from the WEB and may not work!!):

```
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;
         rdaddress_aclr_b        : STRING;
         rdaddress_reg_a     : STRING;
         rdaddress_reg_b     : STRING;
         rdcontrol_aclr_a        : STRING;
         rdcontrol_aclr_b        : STRING;
         rdcontrol_reg_a     : STRING;
```

Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report, video, demo on DE-2 board due date: May 13, 2018 by 12:00 PM

```
        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,
```

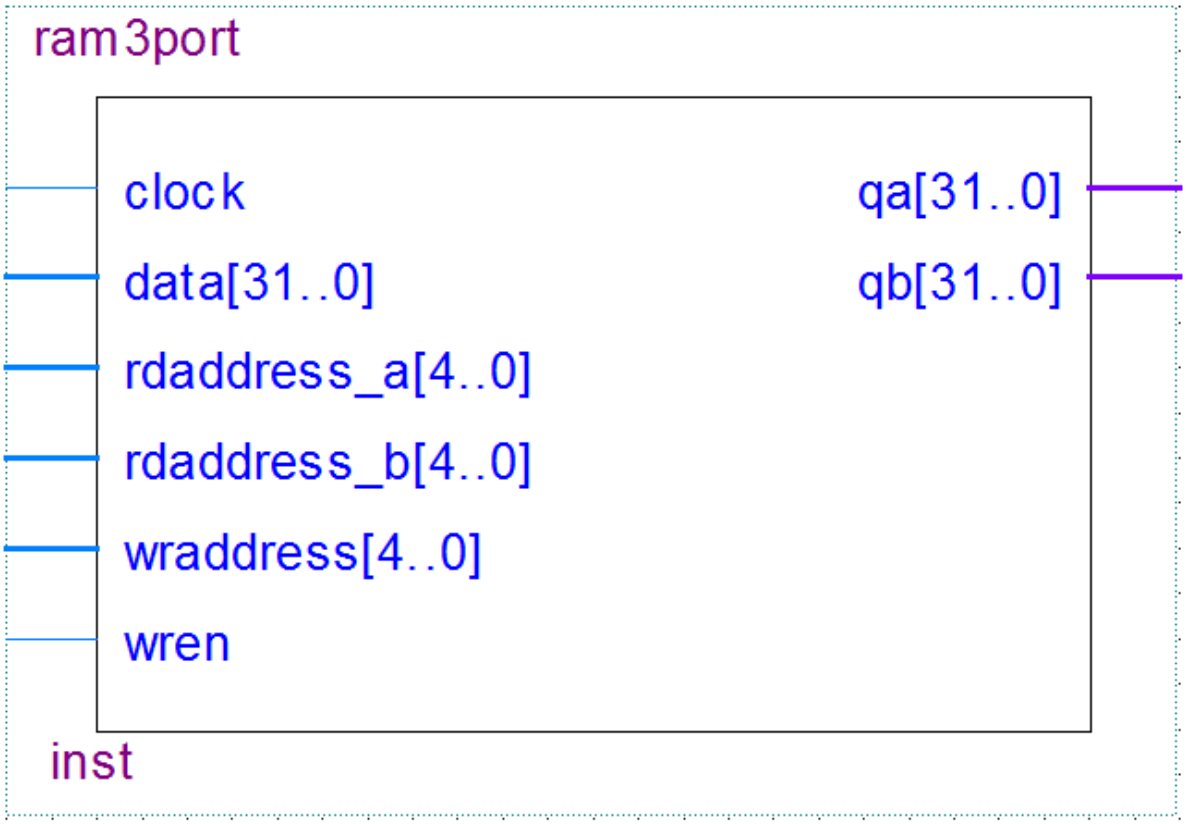
Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report, video, demo on DE-2 board due date: May 13, 2018 by 12:00 PM

```
    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

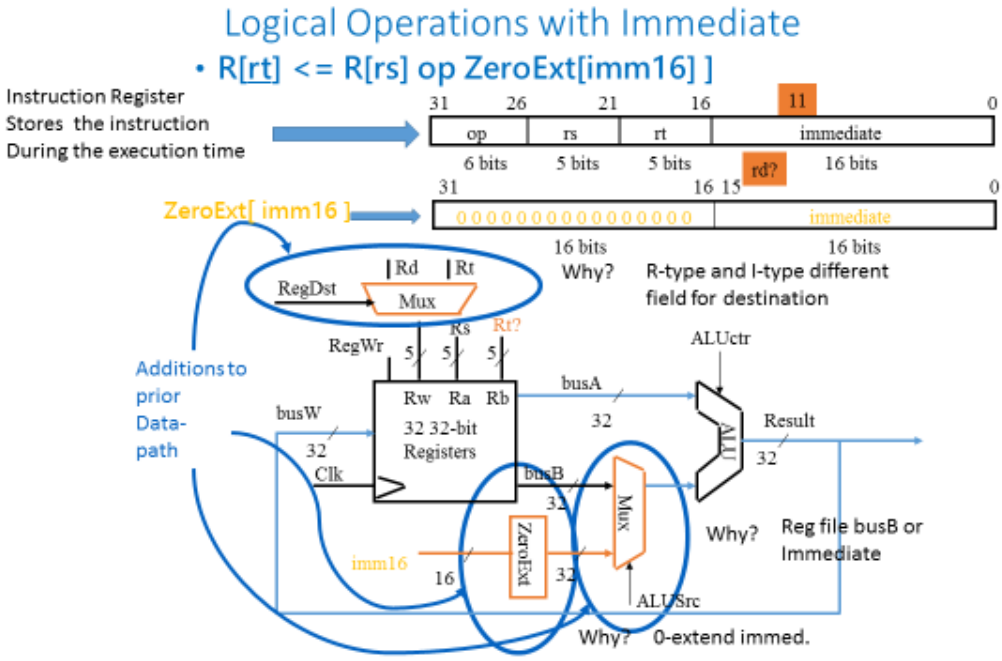
CSC 343, Fall 2018

Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report due date: December 10, 2018 by 12:00 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 Load and Store OPERATIONS, I Type format

Please refer to class notes and to the textbook for required data path.

Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report due date: December 10, 2018 by 12:00 PM

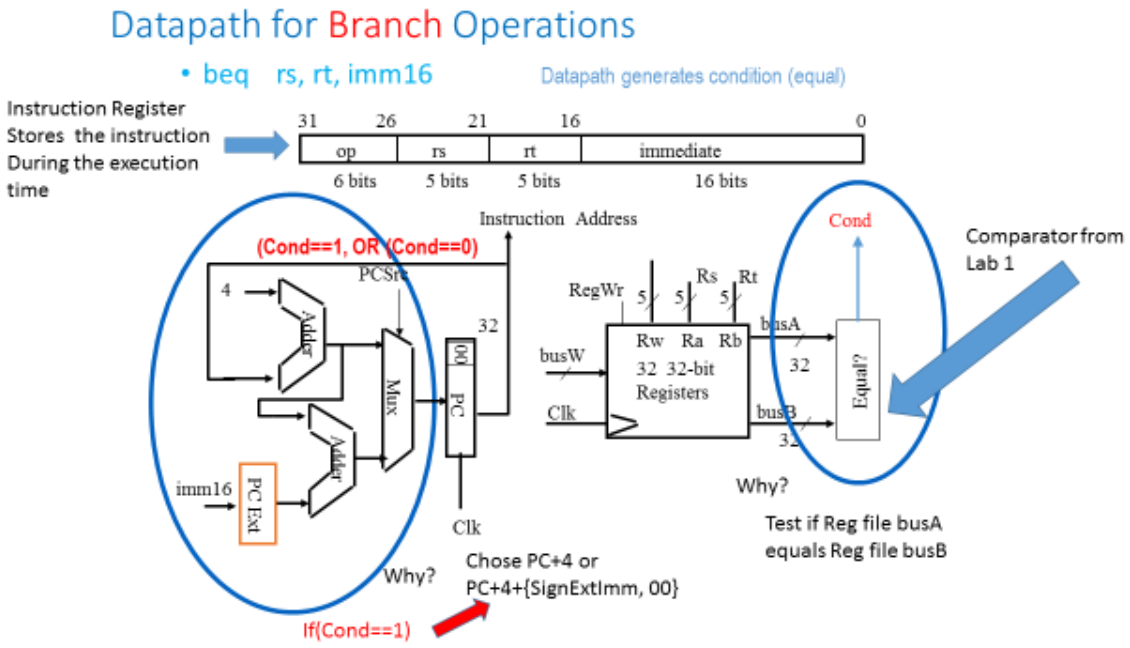
PART IV BEQ instruction, I Type format

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 }
else
PC <= PC + 4



CSC 343, Fall 2018

Lab Project: Single Cycle CPU-LITE

Instructor: Professor Izidor Gertner
Report due date: December 10, 2018 by 12:00 PM

PART V: Extend your design to include Integer multiply and divide instructions from previous lab.

How to test your design:
Write a program using machine instructions you have created and implemented to add ten integers in a loop. The program has to be stored in SSRAM and the ten numbers have to stored in SSRAM also. You can write any program you want, it has to have a LOOP.

What to Submit

1. A report with screenshots of your design, source code, simulations, explain verification in simulation, and pics of demo on a board. PLEASE DO NOT SUMBIT ANY PARTS OF MY HANDOUT IN YOUR REPORT!
3. Working code on my computer.
4. 2 min video explaining your design and demonstration on a board.