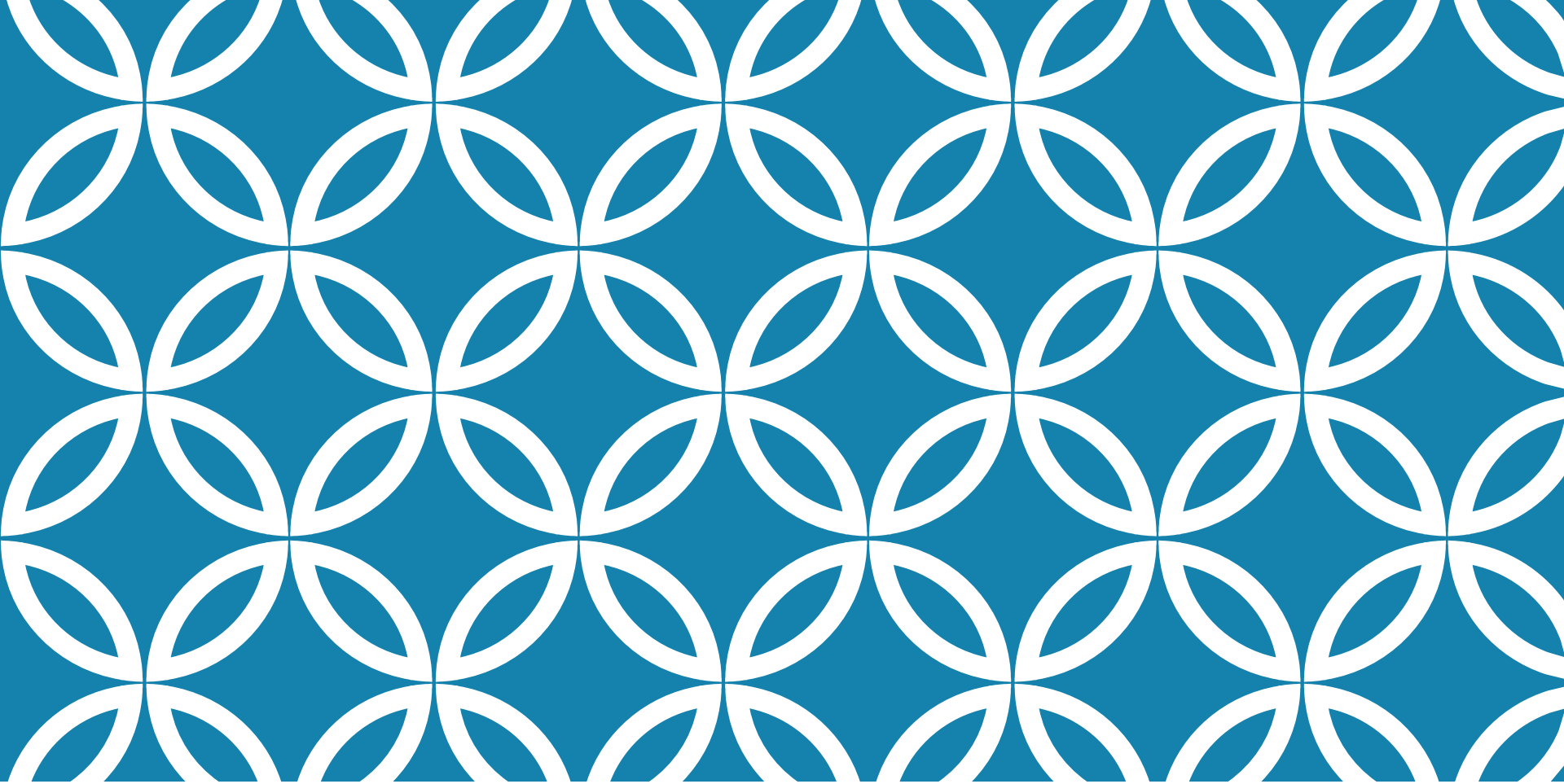


MIPS DATAPATH

FCIS Ainshams University
Spring2021

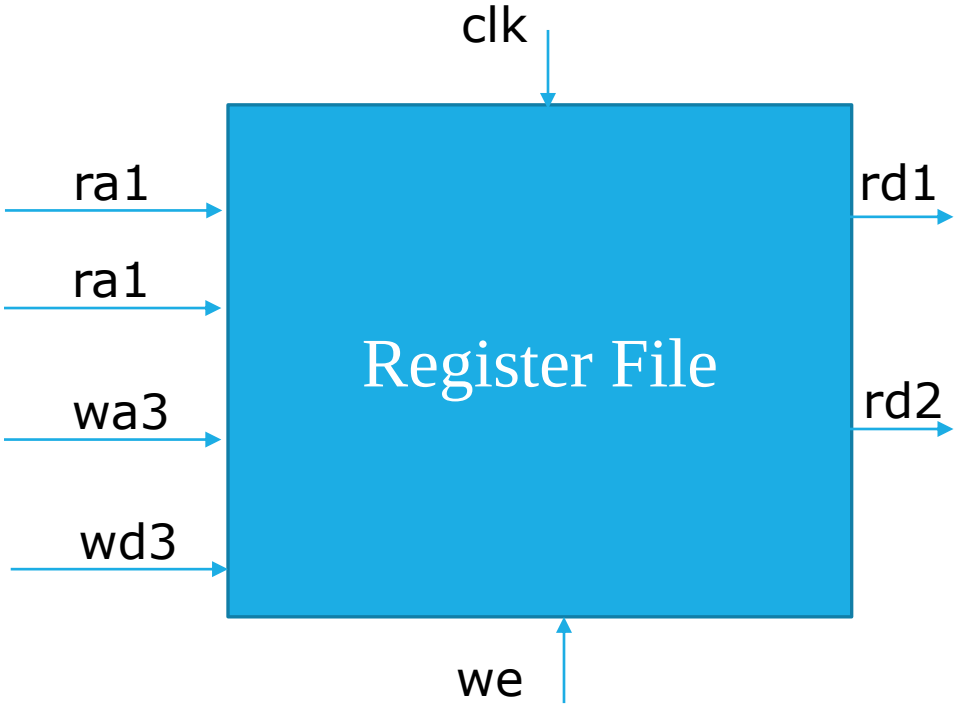
AGENDA

- Register file
- Datapath Module (R-type & I-Type instructions)
- Hands-on: Datapath Implementation

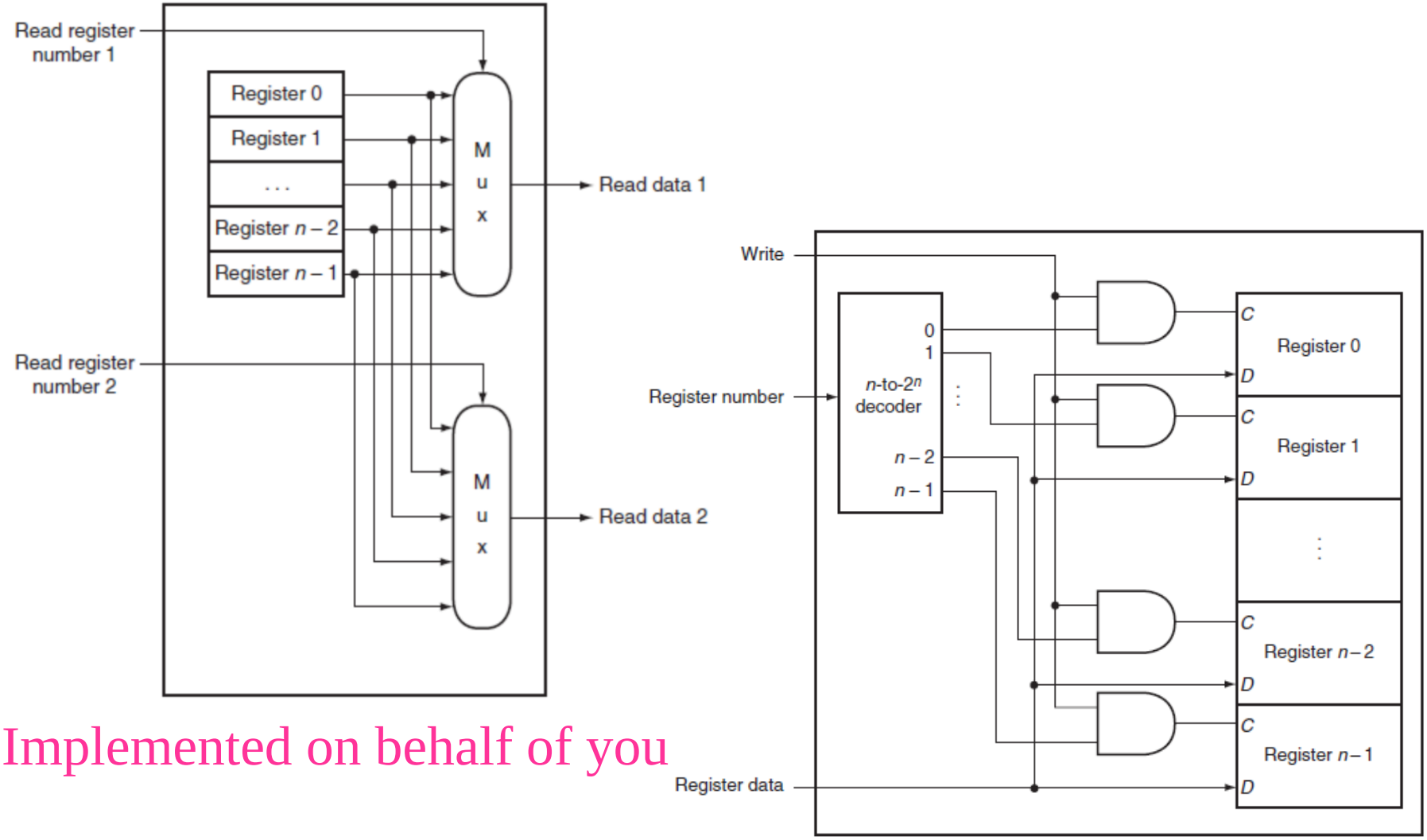
REGISTER FILE

- A register file (RF) is an array of processor registers in a central processing unit (CPU).
- In MIPS it contains 32 registers.

ra1: read register address1
ra2: read register address2
wa: write address
wd: write data
rd1: read data1
rd2: read data2
we: write enable

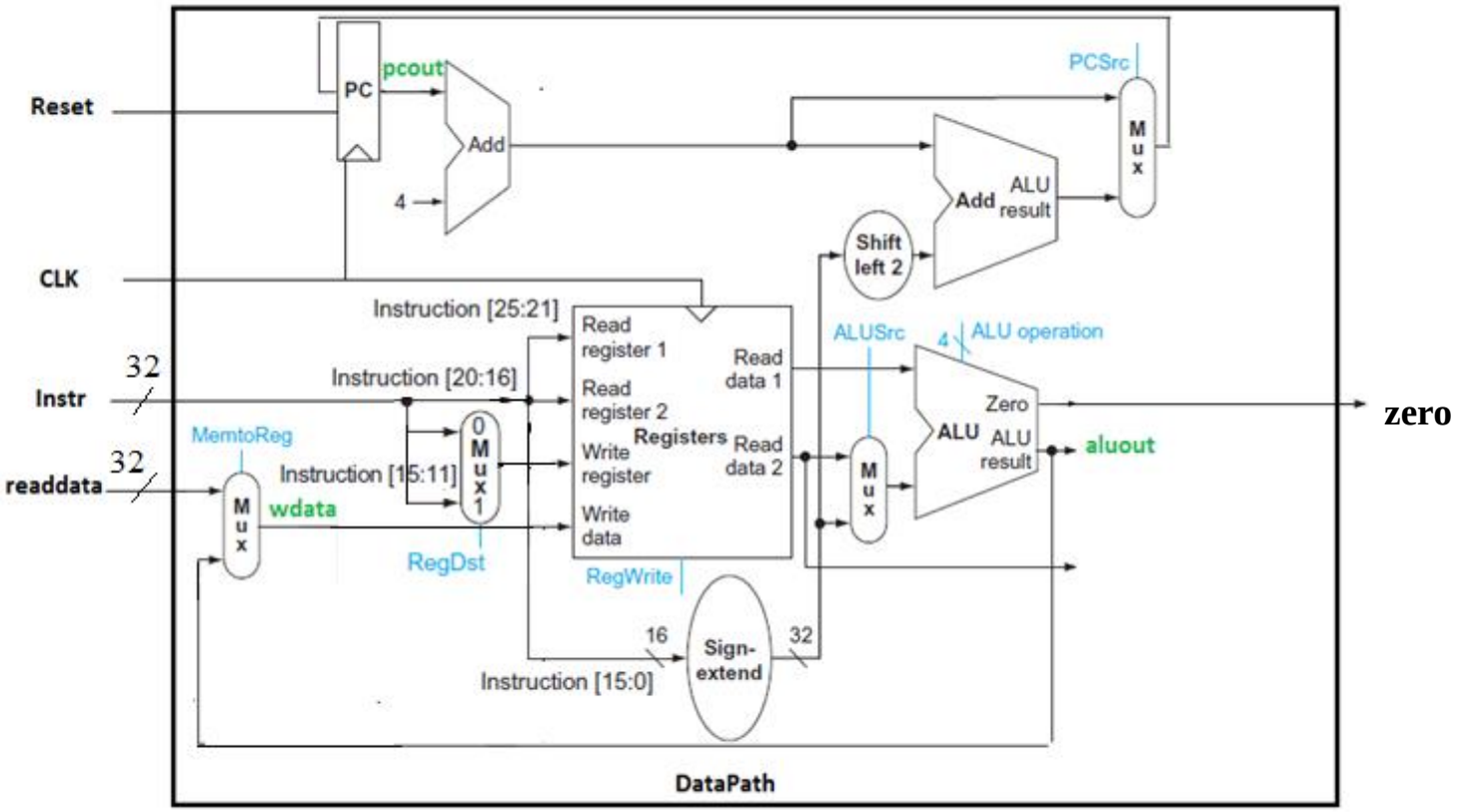


REGISTER FILE



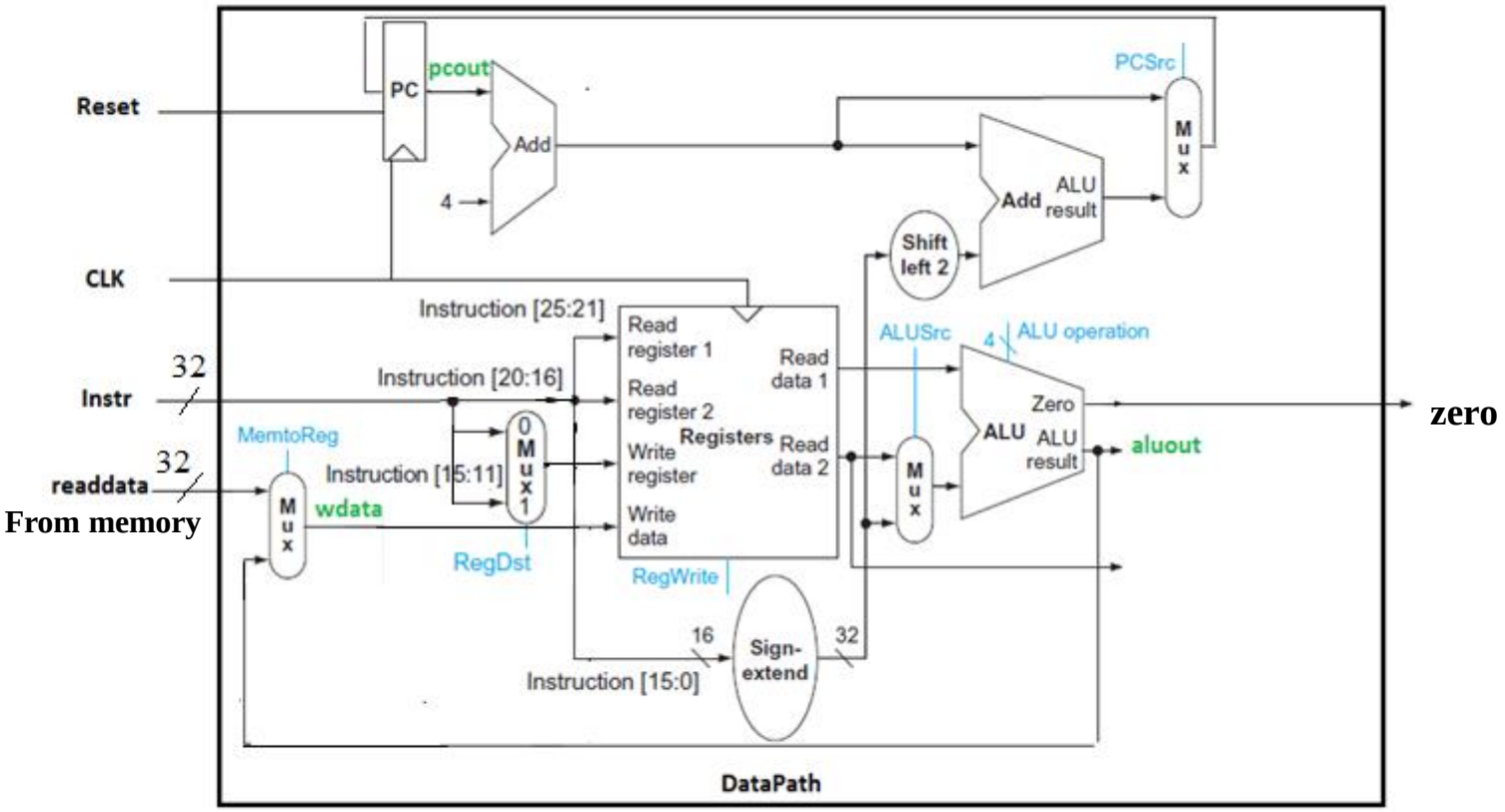
Implemented on behalf of you

DATA PATH


Reset
CLK
Instr
32
readdata
32
PC
pcout
Add
4
PCSrc
M u x
Add
ALU result
Shift left 2
Instruction [25:21]
Instruction [20:16]
Instruction [15:11]
Instruction [15:0]
MemtoReg
wdata
RegDst
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
ALUSrc
4
ALU operation
ALU
Zero
ALU result
aluout
zero
16
Sign-extend
32
DataPath

Field bit positions	31-26	25-21	20-16	15-11	10-6	5-0
No. of bits	6	5	5	5	5	6
R-type	op	rs	rt	rd	shamt	funct
I-type	op	rs	rt	16- bit immediate/address		

DATA PATH



All blue controls are input

DATAPATH IMPLEMENTATION

1. Create a main module for the datapath (DataPath Entity)

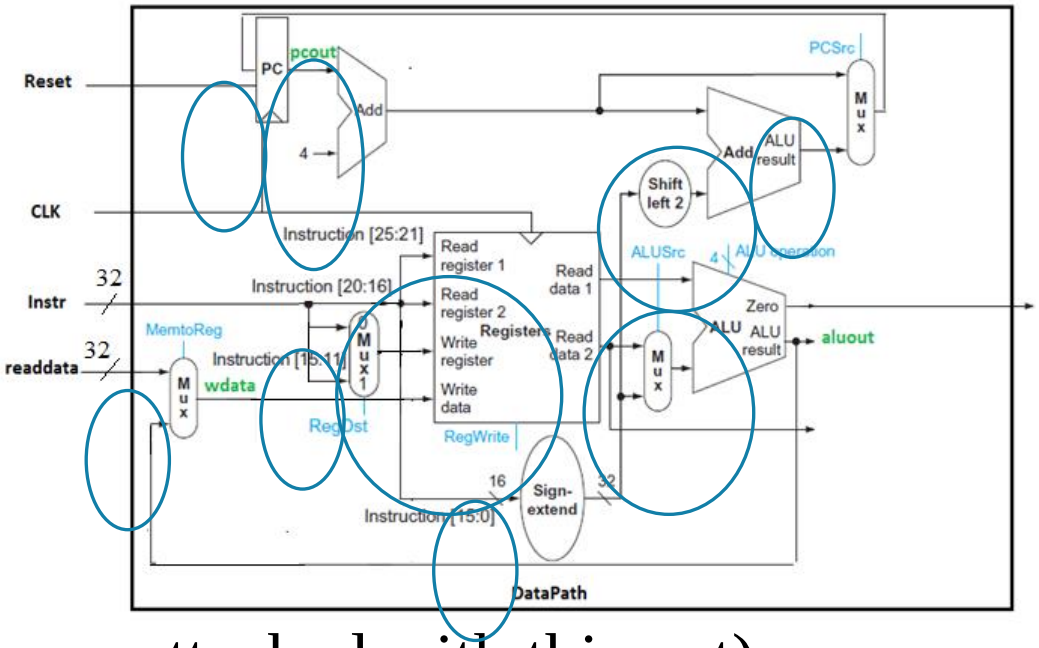
```
entity datapath is -- MIPS datapath
    port(clk, reset: in STD_LOGIC;
    readdata: in STD_LOGIC_VECTOR(31 downto 0);
    instr: in STD_LOGIC_VECTOR(31 downto 0);
    memtoreg, pcsrc,alusrc,regwrite, regdst: in STD_LOGIC;
    aluoperation: in STD_LOGIC_VECTOR(2 downto 0);
    zero: out STD_LOGIC;
    pc: buffer STD_LOGIC_VECTOR(31 downto 0);
  aluout, writedata: buffer STD_LOGIC_VECTOR(31 downto 0));
end;
```

DATAPATH IMPLEMENTATION

2. **add the needed vhdl files:**
   - Adder
   - Sl2
   - Signext
   - flopr
   - mux2
   - Alu
   - regfile

DATAPATH IMPLEMENTATION

2. **Add the needed vhdl files you've previously implemented:**
   - Adder
   - Sl2
   - Signext
   - flopr
   - mux2
   - Alu
   - Regfile (given to you- attached with this ppt)



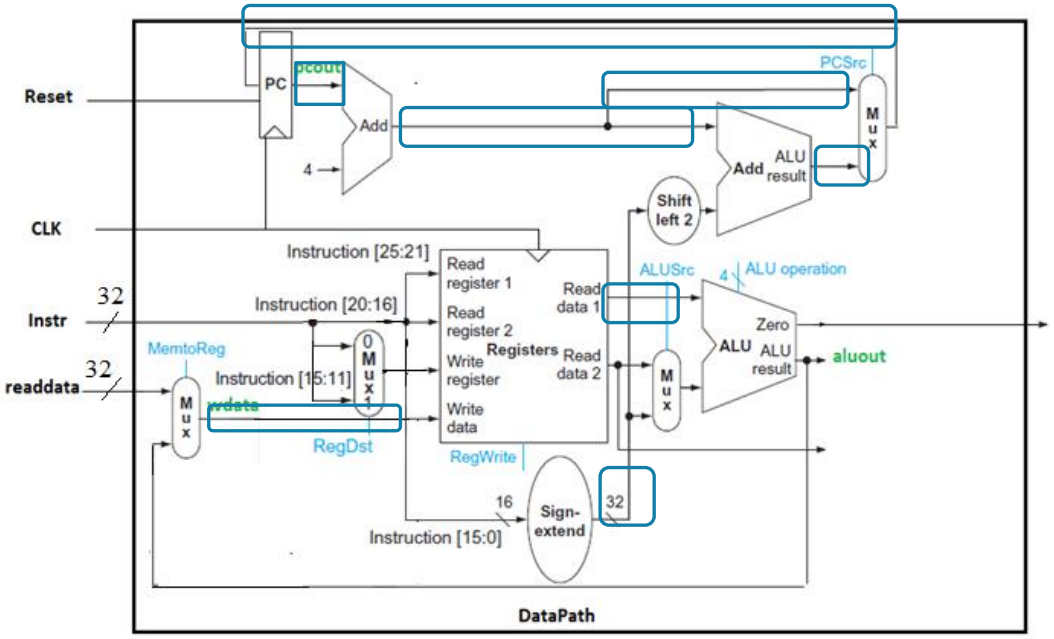
DATAPATH IMPLEMENTATION

3. Add the MIPS package created last lab: “MyPackage”. to be used while building the MIPS processor
   - Don’t forget to create a component for the RegisterFile

```
component regfile
        port(clk: in STD_LOGIC;
        we: in STD_LOGIC;
        ra1, ra2, wa: in STD_LOGIC_VECTOR(4 downto 0);
        wd: in STD_LOGIC_VECTOR(31 downto 0);
        rd1, rd2: out STD_LOGIC_VECTOR(31 downto 0));
end component;
```

DATAPATH IMPLEMENTATION

4. Behavior of the module
   I. Will we need some signals



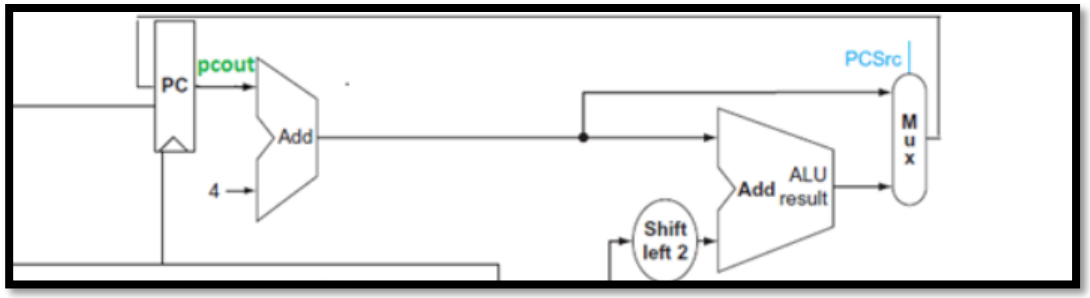
DATAPATH IMPLEMENTATION

4. Behavior of the module
   I. Will we need some signals

```
signal pcnext, pcplus4,pcbranch: STD_LOGIC_VECTOR(31 downto 0);
signal signimm, signimmsh: STD_LOGIC_VECTOR(31 downto 0);
signal srca, srcb, result: STD_LOGIC_VECTOR(31 downto 0);
signal wdata: STD_LOGIC_VECTOR(31 downto 0);
signal writereg: STD_LOGIC_VECTOR(4 downto 0);
```

DATAPATH IMPLEMENTATION

4. Behavior of the module
   II. next PC logic

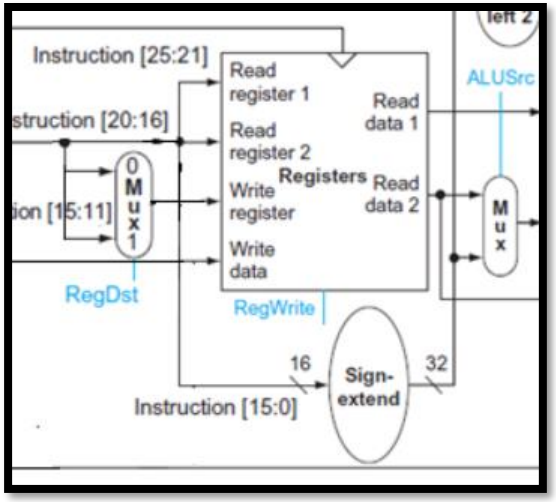



```
begin
-- next PC logic
pcreg: flopr generic map(32) port map(clk, reset, pcnext, pc);
pcadd1: adder port map(pc, X"00000004", pcplus4);
immsh: sl2 port map(signimm, signimmsh);
pcadd2: adder port map(pcplus4, signimmsh, pcbranch);
pcbrmux: mux2 generic map(32) port map(pcplus4, pcbranch, pcsrc, pcnext);
```

DATAPATH IMPLEMENTATION

4. Behavior of the module
   III. register file logic

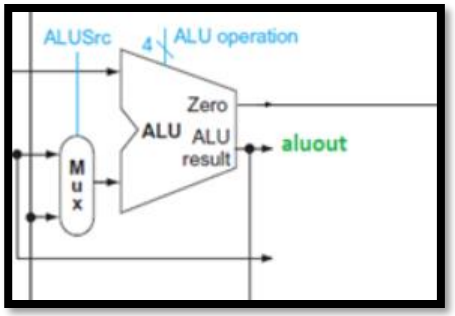
Your turn?



DATAPATH IMPLEMENTATION

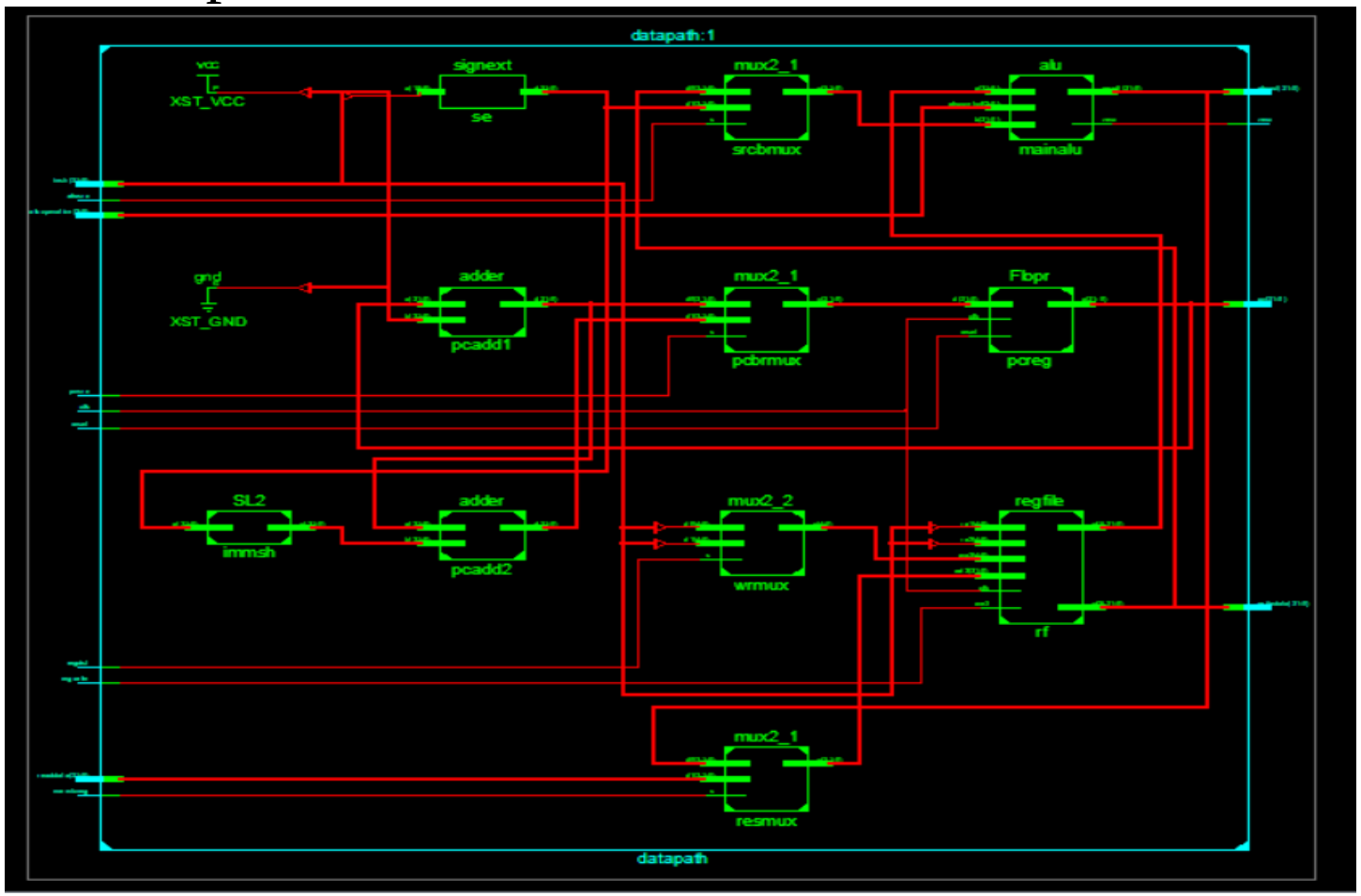
4. Behavior of the module
   IV. ALU logic

Your turn?



DATAPATH IMPLEMENTATION

RTL of Datapath



Thanks