

# HW6 – The **final** MIPS

**Part I – Adding new instructions**

**Part II – Adding Data Forwarding**

**Part III – Adding Branch Forwarding**

# HW6 – The final MIPS

## Part I

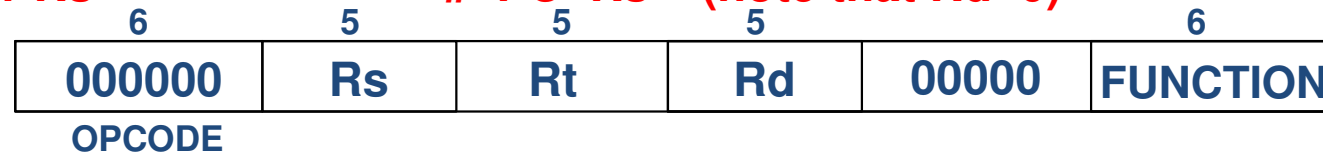
Adding **lui**, **ori**, **jr**, **jal** instructions to the CPU instruction set of Rtype, addi, beq, bne, j, lw, sw

# HW6 instruction set

(new inst. in red)

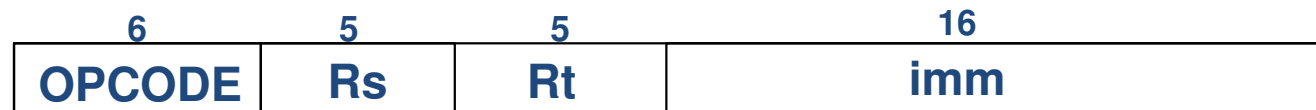
R-type

add Rd, Rs, Rt      # Rd=Rs+Rt  
 sub Rd, Rs, Rt      # Rd=Rs-Rt  
 and Rd, Rs, Rt      # Rd=Rs AND Rt  
 or Rd, Rs, Rt        # Rd=Rs OR Rt  
 xor Rd, Rs, Rt      # Rd=Rs XOR Rt  
 slt Rd, Rs, Rt      # if Rs<Rt Rd=1 else Rd=0  
 jr Rs                # PC=Rs (note that Rd=0)



I-type

addi Rt, Rs, imm      # Rt=Rs+ sext(imm)  
 lw Rt, imm(Rs)      # Rt=M[Rs + imm]  
 sw Rt, imm(Rs)      # M[Rs + imm]=Rt  
 beq Rs, Rt, label    # if Rs==Rt, PC=PC+4+ sext(imm)\*4  
                           # else            PC=PC+4  
 bne Rs, Rt, label    # same as beq with cond of Rs≠Rt  
 ori Rt, Rs, imm      # Rt=Rs OR imm (no sext)  
 lui Rt, imm          # Rt= imm<<16 (no sext)



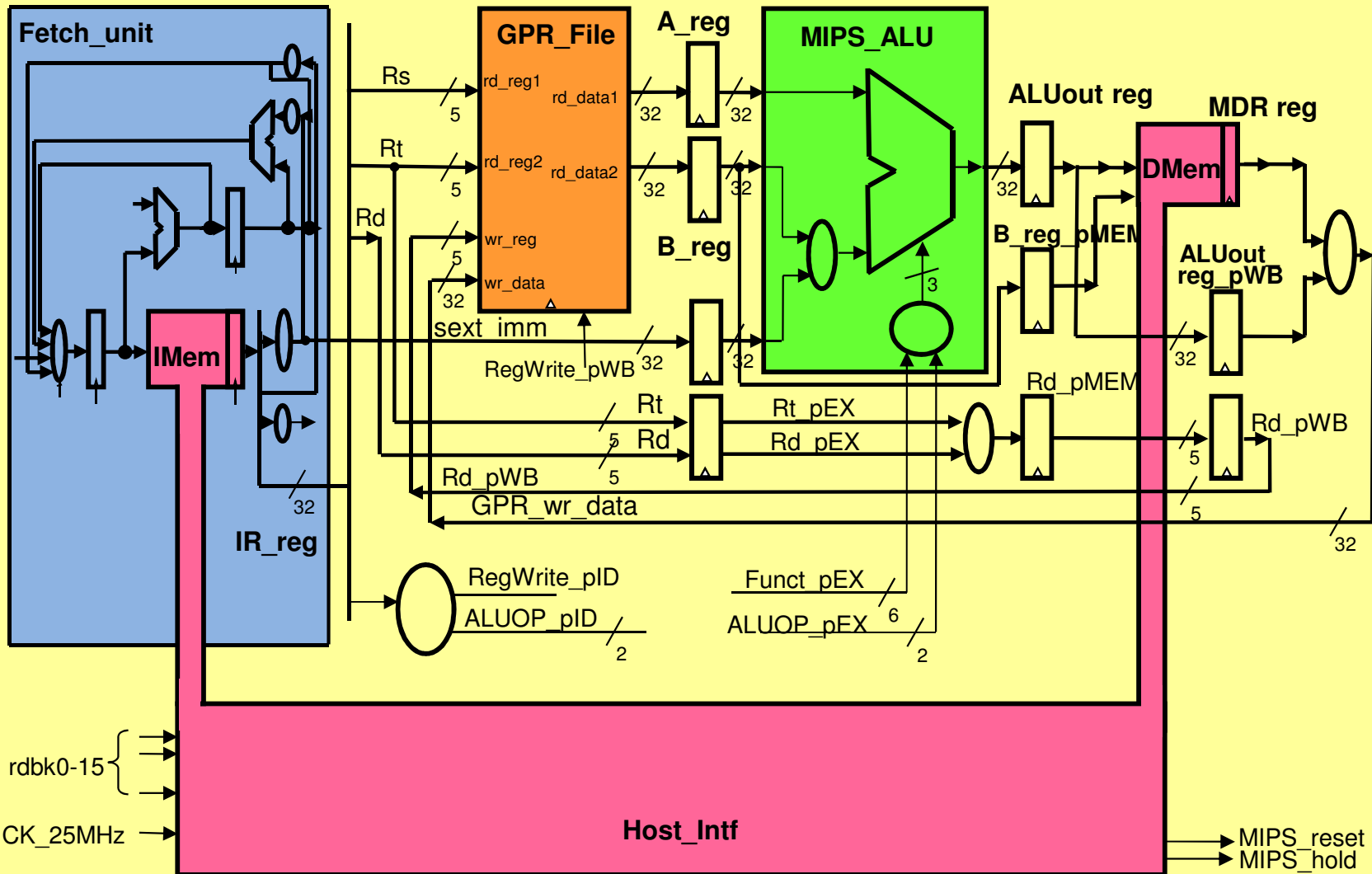
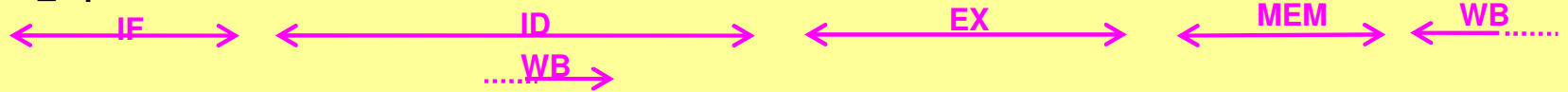
j-type

j imm                # PC= imm\*4 (no sext)  
 jal imm             # PC= imm\*4, \$31=PC+4 (no sext)

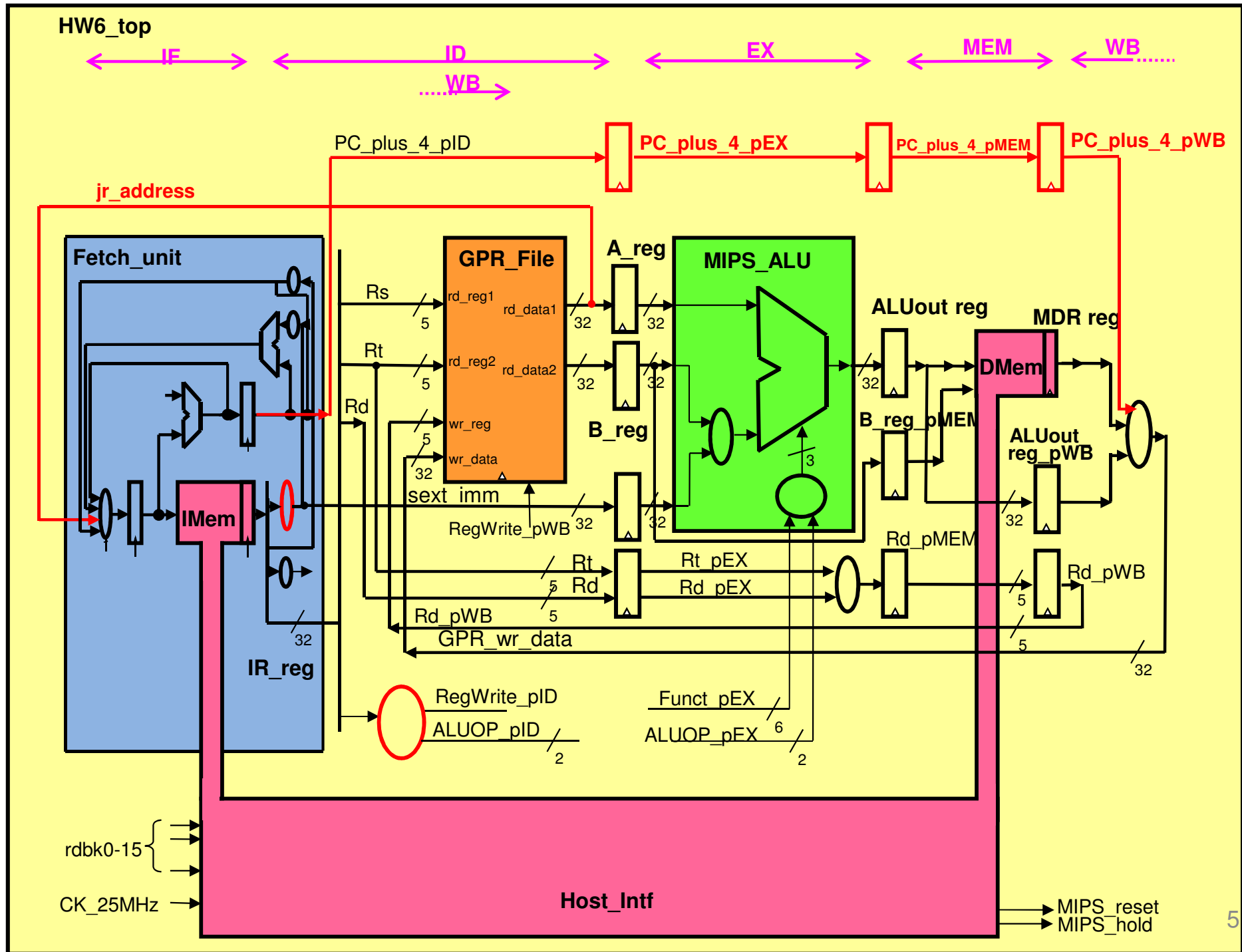


# HW5\_top

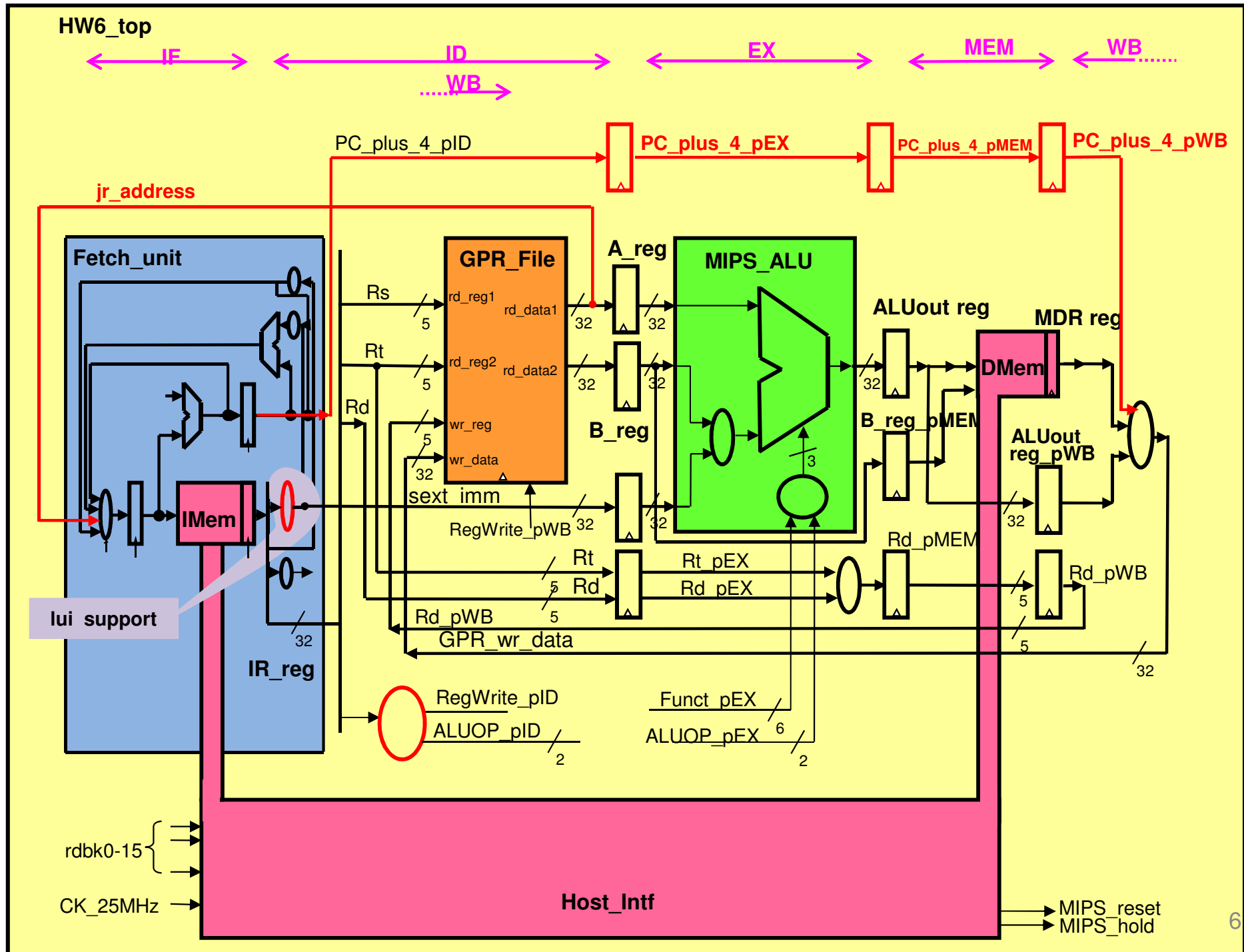
HW5\_top



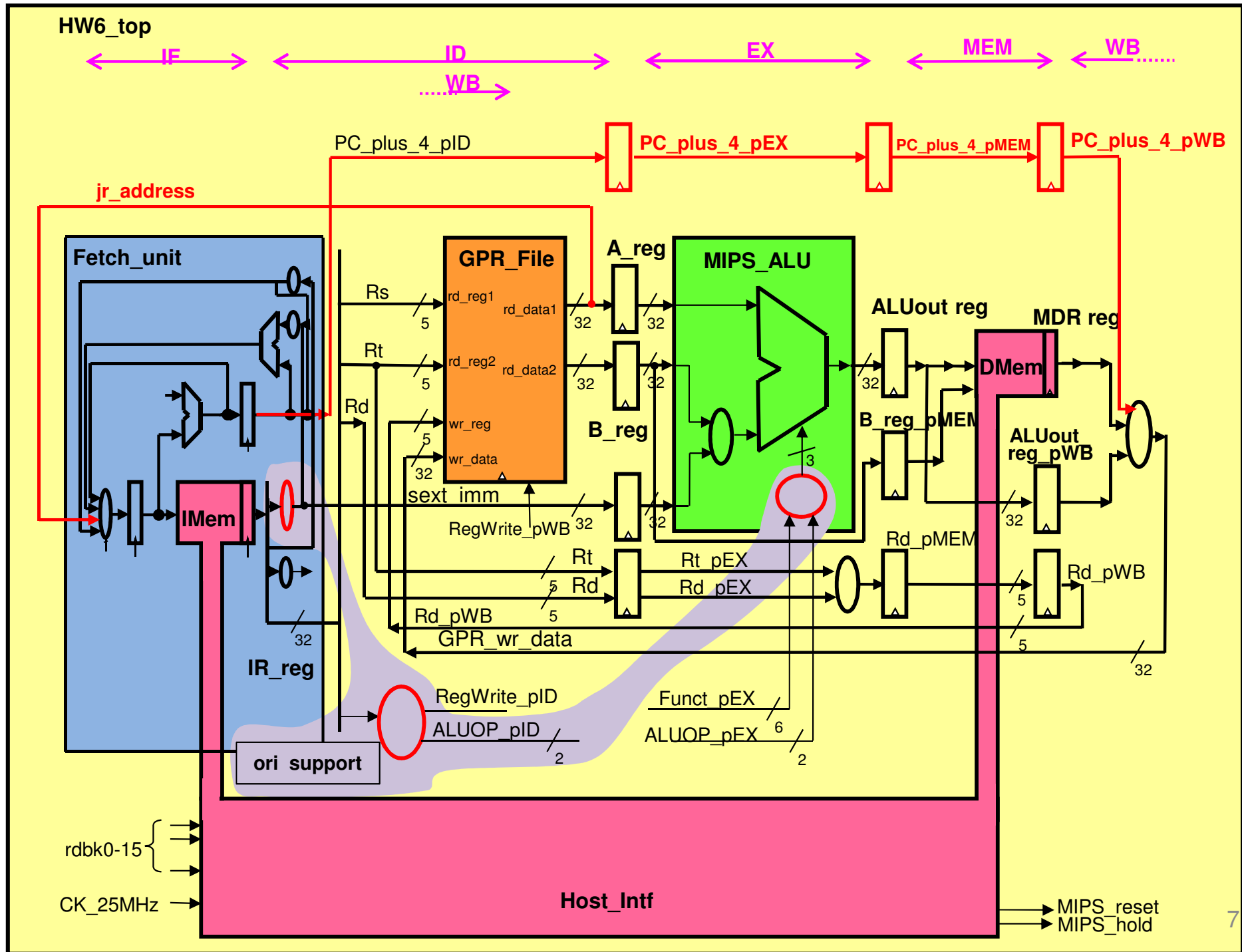
# HW6\_top



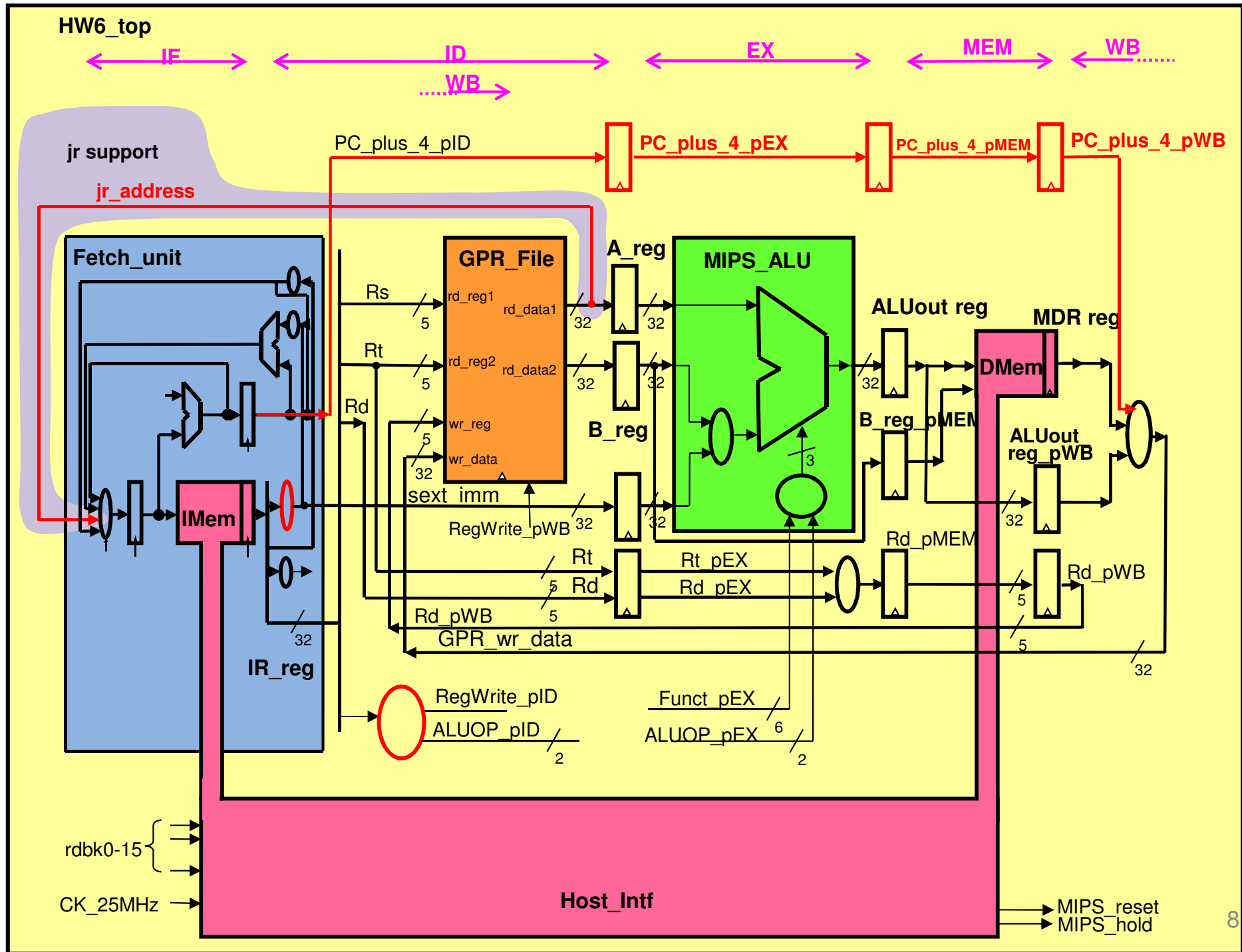
# HW6\_top



# HW6\_top

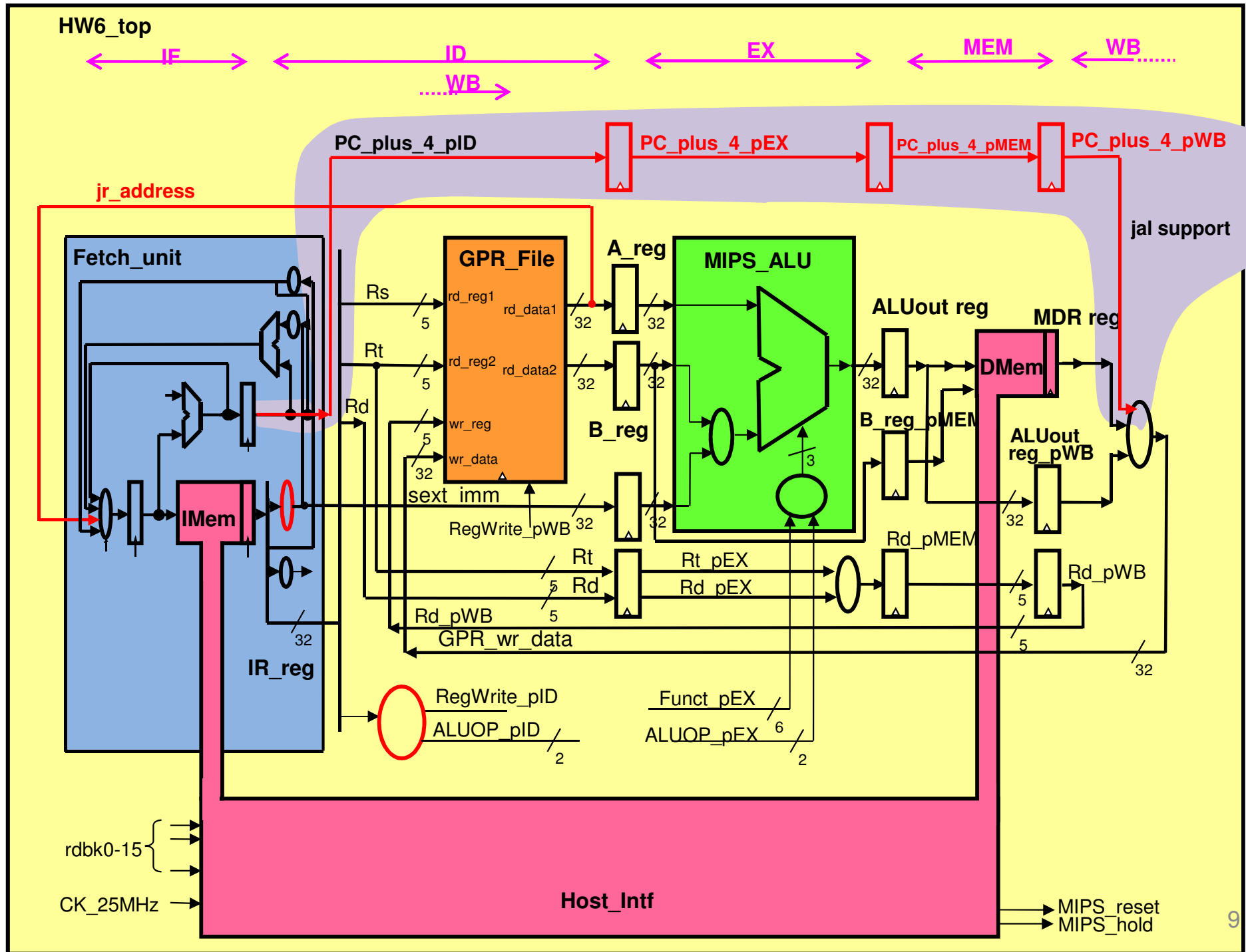


# HW6\_top





# HW6\_top



## General signals in HW6\_top

CK - The 25 MHz clock coming out of the Clock\_driver.

RESET – coming out of the Host\_Intf and is used as reset signal to all registers

HOLD –coming out of the Host\_Intf and is used to freeze writing into all FFs & registers

## IF phase signals in HW6\_top

**PC\_plus\_4\_pID** – PC\_plus\_4\_pID\_out from the Fetch Unit – for jal support

## ID phase signals in HW6\_top

IR\_reg- a 32 bit register that has the instruction we read from the IMem.

This signal is a rename of the IR\_reg\_pID signal coming out of the modified Fetch Unit

Opcode – the 6 MSBs of IR\_reg. To be decoded and produce the control signals.

Rs – IR[25:21], Rt – IR[20:16], Rd – IR[15:11], Funct – IR[5:0].

sext\_imm – renaming of sext\_imm\_pID coming out of the Fetch Unit.

GPR\_rd\_data1 – the 32 bit output of the rd\_data1 of the GPR and input to A\_reg.

GPR\_rd\_data2 – the 32 bit output of the rd\_data2 of the GPR and input to B\_reg.

Rs\_equals\_Rt – ‘1’ if GPR\_rd\_data1== GPR\_rd\_data2, and ‘0’ otherwise.

Used in branch instructions. That signal (renamed) is sent to the Fetch Unit.

**jr\_address** – connected to the new jr\_adrs\_in pin in the Fetch Unit – for jr inst.

## ID control signals in HW6\_top - These are created from decoding the opcode:

ALUsrcB – ‘1’ when sext\_imm is used (in addi instruction).

ALUOP – a 2 bit signal. “00” means add(addi inst.), “01” means subtract (not used), “10” will cause the ALU to follow the Funct field.

RegDst – ‘0’ when we WB according to Rt (addi inst.) ‘1’ -according to Rd (Rtype inst.).

RegWrite – ‘1’ when we WB (Rtype or addi inst.), ‘0’ when we don’t (j, beq & bne inst.)

MemWrite – ‘1’ in sw (writing to Dmem), MemToReg = ‘1’ in lw (reading from DMem)

**JAL** – ‘1’ in jal . Used to control the expanded MemToReg mux

### EX phase signals in HW6 top

A\_reg – a 32 bit register receiving the GPR\_rd\_data1 signal. Its value is used in EX phase

B\_reg – a 32 bit register receiving the GPR\_rd\_data2 signal

sext\_imm\_reg – a 32 bit register receiving the sext\_imm coming from the Fetch Unit

Rt\_pEX – Rt delayed by 1 clock cycle

Rd\_pEX – Rd delayed by 1 clock cycle

ALUoutput – a 32 bit signal of the output of the ALU (renaming of ALU\_out signal coming out of the MIPS\_ALU component).

**PC\_plus\_4\_pEX** – PC\_plus\_4\_pID delayed by 1 clock cycle – for jal

### EX phase control signals in HW6 top

ALUsrcB\_pEX – ALUsrcB delayed by 1 clock cycle.

Funct\_pEX – Funct delayed by 1 clock cycle.

ALUOP\_pEX – ALUOP delayed by 1 clock cycle.

RegDst\_pEX – RegDst delayed by 1 clock cycle.

RegWrite\_pEX – RegWrite delayed by 1 clock cycle.

MemWrite\_pEX – MemWrite delayed by 1 clock cycle.

MemToReg\_pEX – MemToRegdelayed by 1 clock cycle.

**JAL\_pEX** – JAL delayed by 1 clock cycle.

### MEM phase signals in HW6\_top

B\_reg\_pMEM – a 32 bit register receiving the B\_reg signal (i.e., B\_reg delayed by 1 CK). This register has the data to be written into the DMem in sw instruction.

Rd\_pMEM – the output of RegDest mux – to which reg the CPU writes in the WB phase.

**PC\_plus\_4\_pMEM** – PC\_plus\_4\_pEX delayed by 1 clock cycle – for jal

### MEM phase control signals in HW6\_top

MemWrite\_pMEM - MemWrite\_pEX delayed by 1 clock cycle.

MemToReg\_pMEM – MemToReg\_pEX delayed by 1 clock cycle.

RegWrite\_pMEM – RegWrite\_pEX delayed by 1 clock cycle.

**JAL\_pMEM** – JAL\_pEX delayed by 1 clock cycle.

### WB phase signals in HW6\_top

MDR\_reg- a 32 bit register that has the data read from the memory. Rename of DMem\_rd\_data signal coming out of the HW5\_Host\_Intf\_4sim component.  
ALUout\_reg\_pWB - a 32 bit register that has the ALUout\_reg data delayed by 1 CK cycle.  
GPR\_wr\_data - a 32 bit signal that is the output of the MemToReg mux  
Rd\_pWB – Rd\_pMEM delayed by 1 clock cycle.  
**PC\_plus\_4\_pWB – PC\_plus\_4\_pMEM delayed by 1 clock cycle – for jal**

### WB phase control signals in HW6\_top

MemToReg\_pWB – MemToReg\_pMEM delayed by 1 clock cycle  
RegWrite\_pWB – RegWrite\_pMEM delayed by 1 clock cycle.  
**JAL\_pWB – JAL\_pMEM delayed by 1 clock cycle.**

You get a **HW6\_top\_4sim.empty** file in which you have all of these signals defined . You have to add your design of the HW6\_top entity, i.e., write the equations of the top file (and rename it to **HW6\_top\_4sim.vhd**).

In this vhd file we use the **Fetch\_Unit**, **GPR**, **MIPS\_ALU**, **BYOC\_Clock\_Driver** and the **BYOC\_Host\_Intf\_4sim**.

**Fetch\_Unit & MIPS\_ALU** are in red to remind you they should be modified

# Description of the HW6\_top\_4sim project

---

1. **HW6\_top\_4sim.vhd** – This is your design of HW6. It uses the **GPR**, the updated **MIPS\_ALU**, the updated **Fetch\_Unit**, the **BYOC\_Clock\_driver\_4sim** and the **BYOC\_Host\_Intf\_4sim** components and all of the signals described in 2b.
  2. **GPR.vhd** – your GPR File design you prepared in HW3.
  3. **dual\_port\_memory.vhd** – part of the GPR File design you prepared in HW3.
  4. **MIPS\_ALU.vhd** – your MIPS\_ALU design you prepared in HW3 **and updated for HW6!!**
  5. **Fetch\_Unit.vhd** - The Fetch Unit you prepared in HW2 after the modifications done in HW4 **and the additional modification in HW6!!**
  6. **BYOC\_Clock\_driver\_4sim.vhd** – the CK divider & driver we use for simulation (also good for the Modelsim simulator)
  7. **BYOC\_Host\_Intf\_4sim.vhd** – The prepared components including the IMem and “pre-loaded” program and creating the reset & ck signals.
  8. **SIM\_HW6\_TB.vhd** - The TB vhd file prepared in advance. See the note in 9 below.
  9. **SIM\_HW6\_TB\_no\_fwdng.dat** – this is a data file prepared in advance that is read by the SIM\_HW6\_TB and used to compare the simulation results to the expected ones.
  10. **SIM\_HW6\_program.dat** - The program file for simulation.
  11. **SIM\_HW6\_filenames.vhd** - The actual path information of the two dat files.
- 

**NOTE:** Inside **SIM\_HW6\_TB\_filenames.vhd**, we specified the **path** of and the **name of the dat file** to which the design is compared. You should update that according to your sim project actual path and according to the design tested (no fwd, data fwd, branch fwd). 14

## Simulation report

You should submit a single zip file for the Simulation and implementation phases. It should have **four** directories/folders. The first is called **Simulation1**, the 2<sup>nd</sup> is called **Simulation2**, the 3<sup>rd</sup> is called **Simulation 3**, the 4<sup>th</sup> is called **Implementation**. In the Simulation folders you will have 3 sub-folder of:

- **Src\_4sim** – here you put all of the \*.vhd sources for simulation
- **Sim** – here you should have the HW6 4sim project created by the simulator you used
- **Docs** – Here you put your simulation report. The first few lines in the report will have your ID numbers (names are optional). See the instructions in BYOC\_HW6.doc. **This should be a WORD file and not a PDF file – so remarks can be added when grading the report.**

**Simulation1** will have the “no forwarding” design of **part I** where you add the lui, ori, jr and jal instructions. You should answer the questions in **Appendix A** and insert these to your report of **Simulation1**.

You should use the **SIM\_HW6\_TB\_data\_no\_fwding.dat** file for the simulation.

## Simulation report (cont.)

**Simulation2** will have the “data forwarding” design of part II where you add the data forwarding to the design of **Simulation1**. You should answer the questions in **Appendix B** and insert these to your report of **Simulation2**.

You should use the [SIM\\_HW6\\_TB\\_data\\_wt\\_data\\_fwding.dat](#) file for the simulation.

**Simulation3** will have the “branch forwarding” design of part III where you add Branch Forwarding to the design of **Simulation2**. You should answer the questions in **Appendix C** and insert these to your report of **Simulation3**.

You should use the [SIM\\_HW6\\_TB\\_data\\_wt\\_data\\_and\\_branch\\_fwding.dat](#) file for the simulation.

Later, in the Implementation phase you will add 2 sub-folders to the **Implementation** folder. These will be:

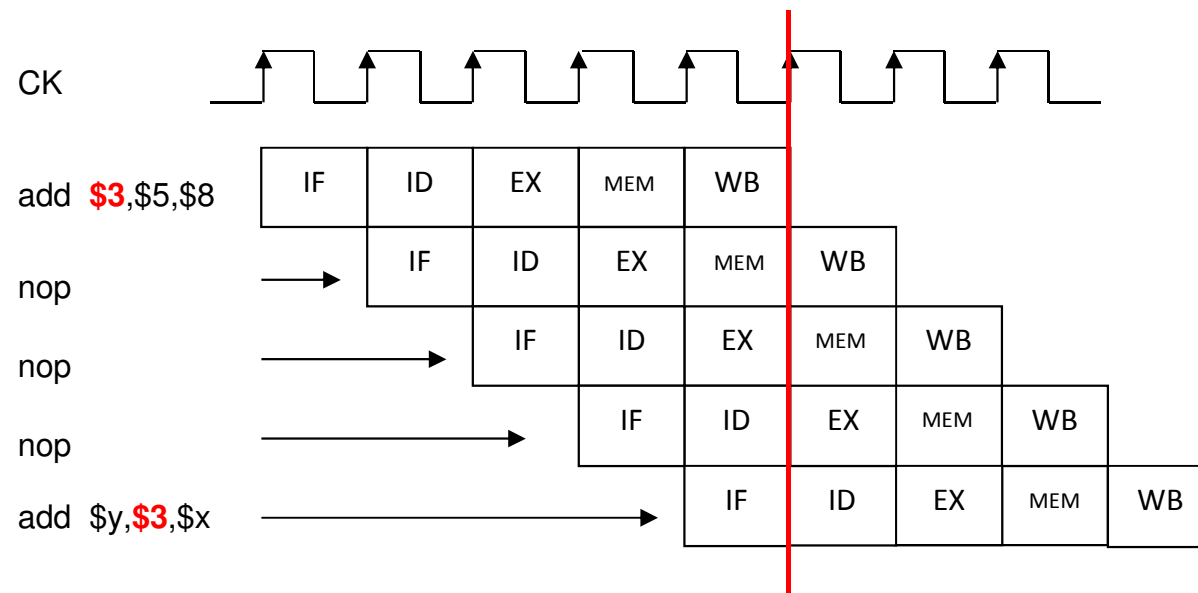
- **Src\_4ISE** – here you put all of the \*.vhd sources and the \*.ucf file (and no TB file)
- **ISE** – here you should have the HW6\_MIPS project created by the Xilinx ISE SW.



# **HW6 – The final MIPS**

## **Part II**

### **Adding Data Forwarding**



**Fig. 2 – The pipelined MIPS latency**

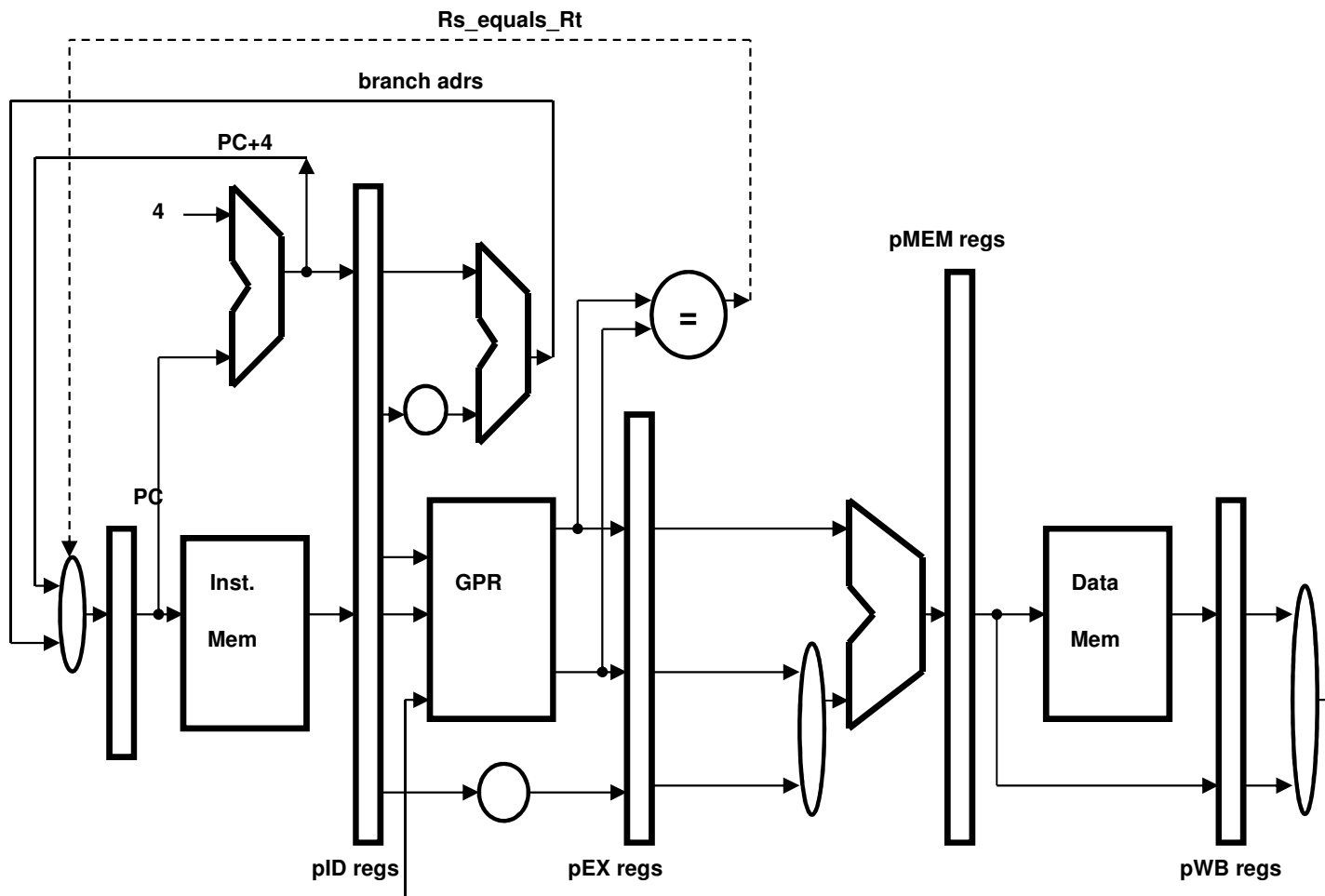


Fig. 4 – MIPS data path (part) with no forwarding

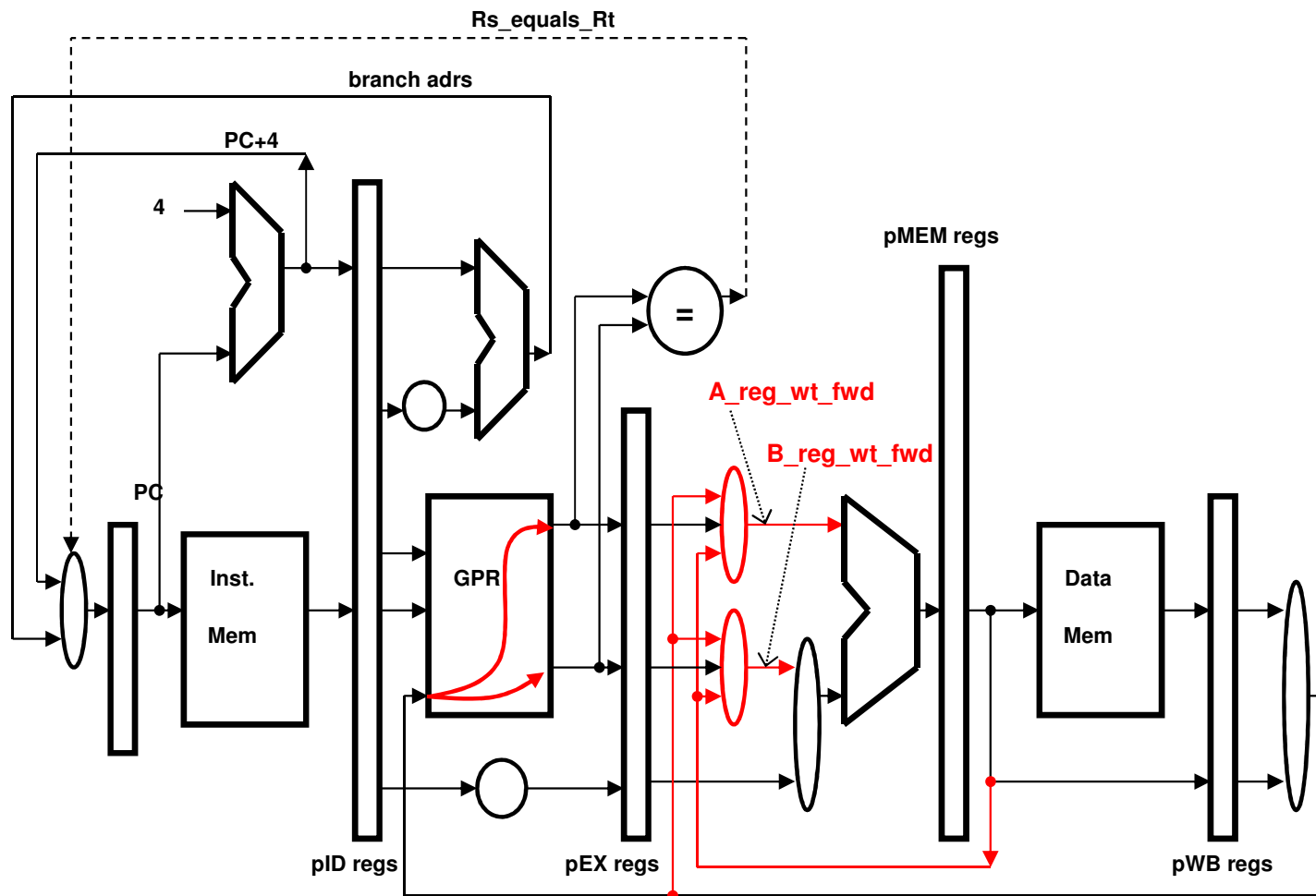


Fig. 5 – MIPS Data Path with Data Forwarding

You need to add the **red** signals

You need to handle also the B\_reg\_pMEM. To which signal should we connect it to?

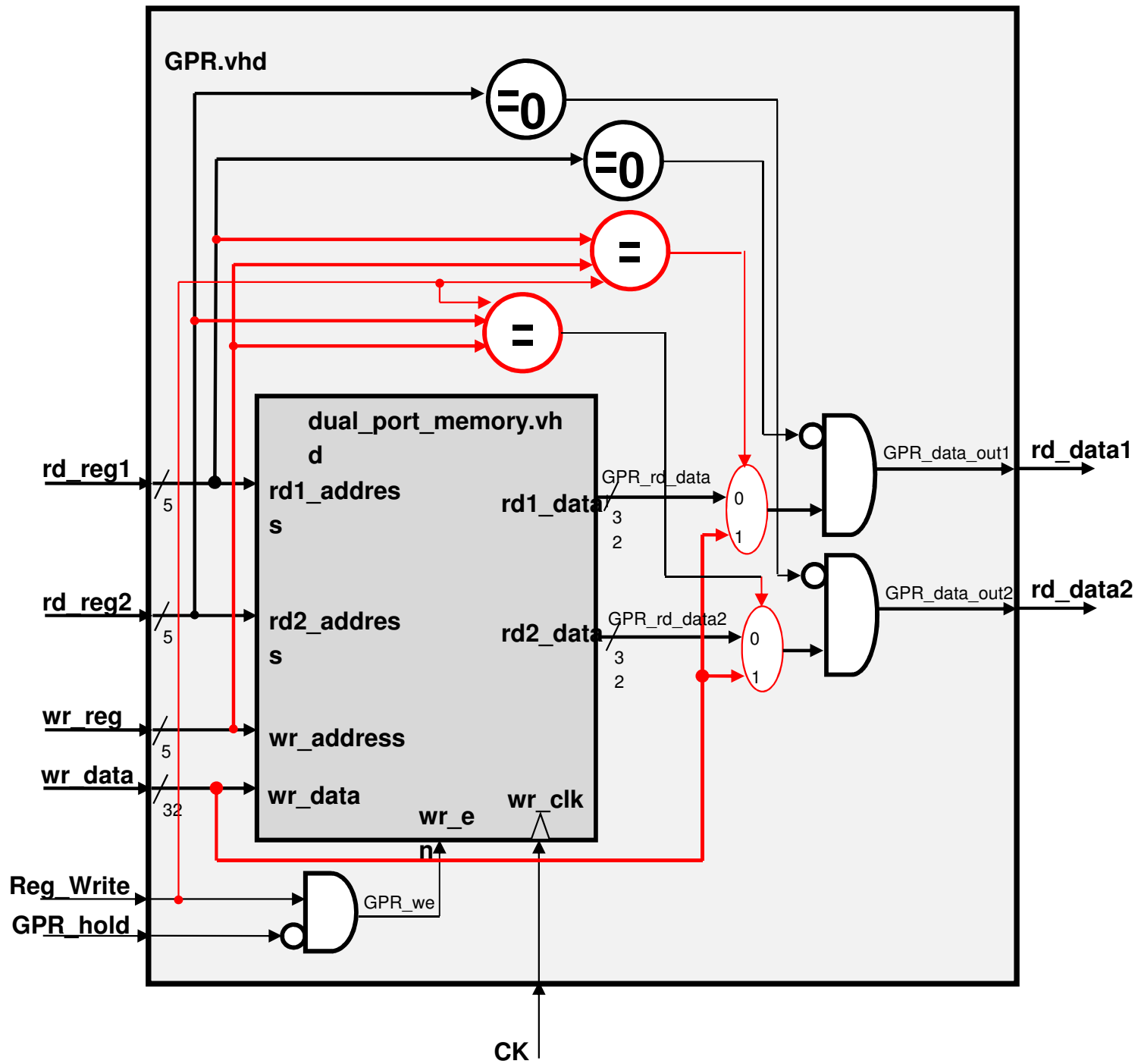
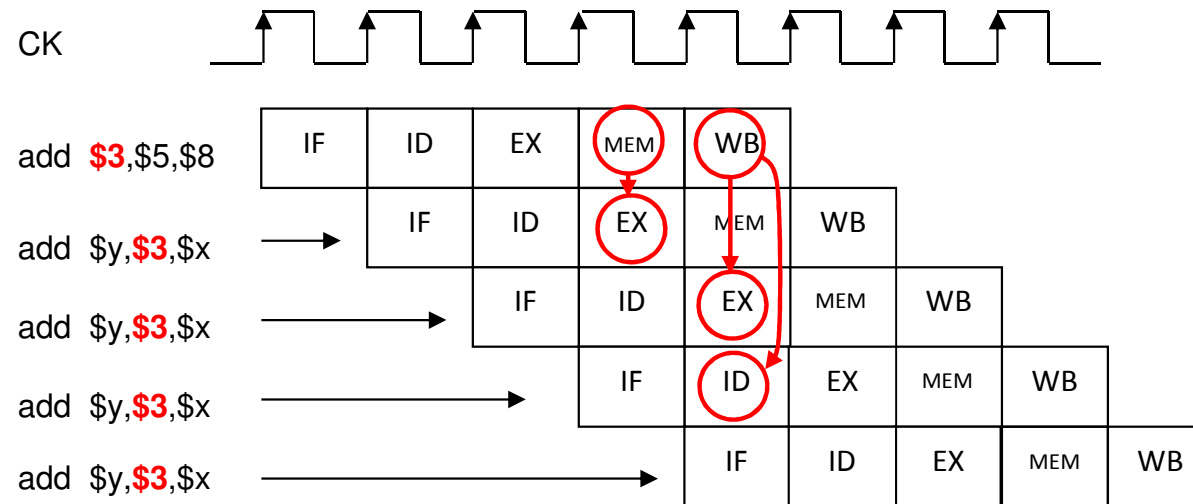
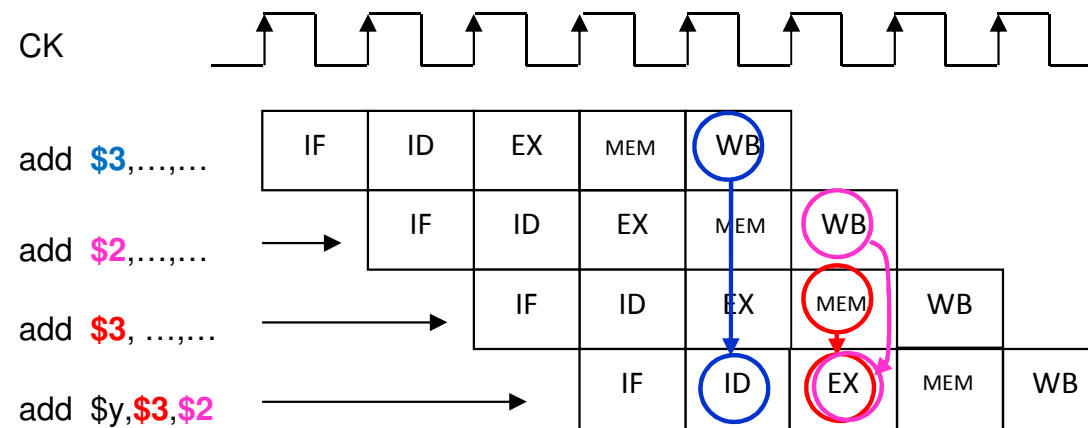


Fig. 6 – MIPS Data Path with Data Forwarding



**Fig. 3 – Data Forwarding timing diagram**  
 (from the 1<sup>st</sup> instruction to future instructions)



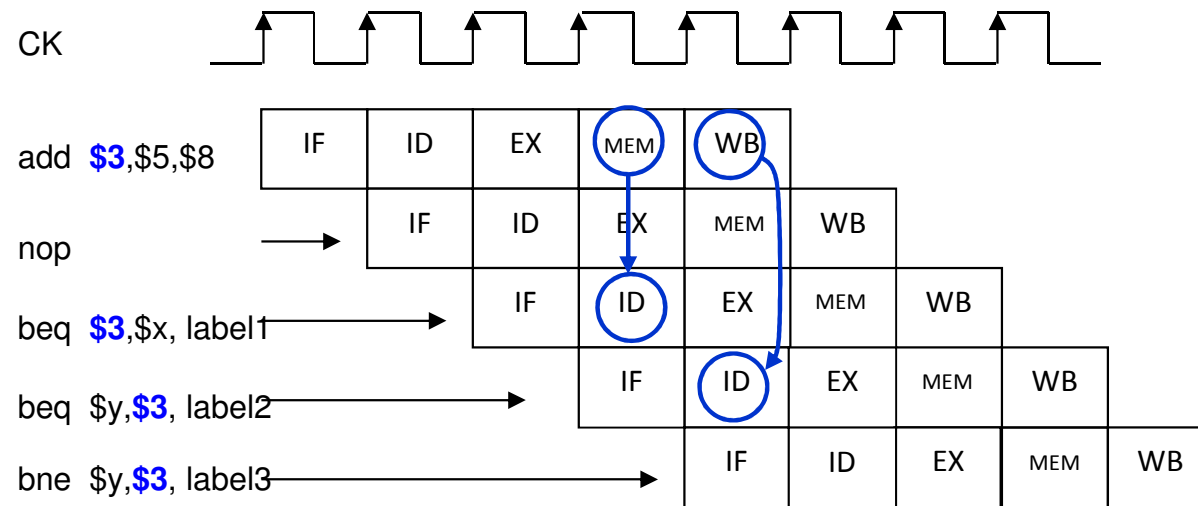
**Fig. 3B – The 3 Data Forwarding options to an instruction  
(to the 4<sup>th</sup> instruction from previous instructions)**

# **HW6 – The final MIPS**

## **Part III**

### **Adding Branch Forwarding**





**Fig. 7 – Branch Forwarding timing diagram**

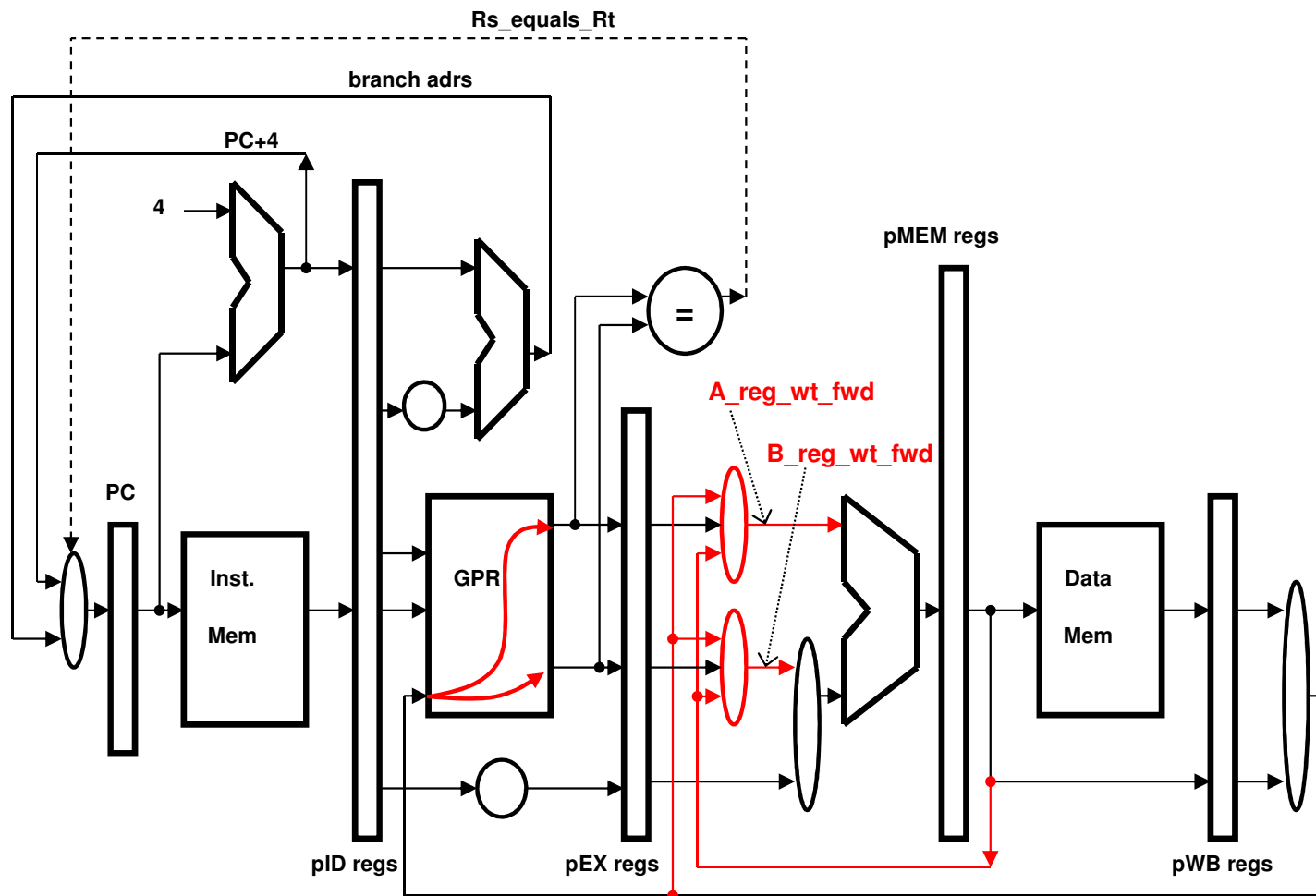


Fig. 5 – MIPS Data Path with Data Forwarding

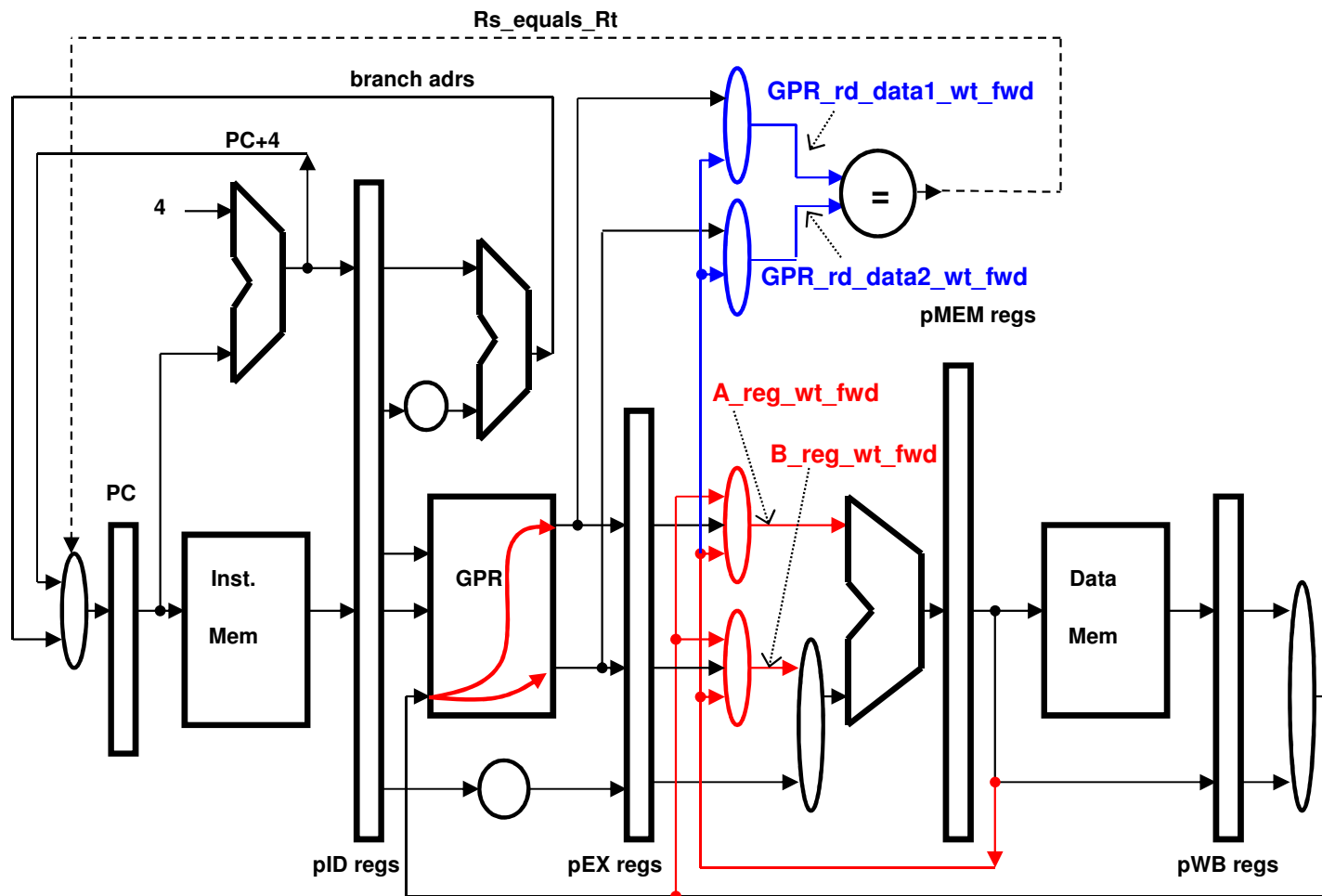


Fig. 8 – MIPS Data Path with Data and Branch Forwarding

You need to add the **blue** signals

You need to handle also the `jr_address`. To which signal should we connect it to?

All the new signals are listed in page 11 of the BYOC\_HW6 doc.

Also in the “empty” vhd file they are already there and marked with “-- @@@HW6”

**Now let's talk on the  
Implementation phase**

# HW6 MIPS - implementation

1. Take the **HW6\_top\_4sim.vhd** and remove of all TB signals. Then rename it to **HW6\_top.vhd**. Another way is to take the **HW6\_top.empty** and integrate into it all the contents you made on the **HW6\_top\_4sim.empty** file in the simulation phase.
2. You should replace the **BYOC\_Host\_Intf\_4sim.vhd** with a component that looks the same, the **BYOC\_Host\_Intf.ngc**, which has the infra-structure that allows the PC to load data into the IMem via the RS232 by the **BYOCInterface** SW.
3. The files we will use to implement the design on the Nexys2 board are:
  - **BYOC.ucf** - The file listing which signal are connected to which FPGA pins in the Nexys2 board.
  - **HW6\_top.vhd** – This is your design of HW6
  - **Fetch\_Unit.vhd** - The Fetch Unit you prepared in HW2 after modifications of HW4 & HW6
  - **GPR.vhd** – your GPR File design you prepared in HW3 with the changes of HW6.
  - **dual\_port\_memory.vhd** – part of the GPR File design you prepared in HW3.
  - **MIPS\_ALU.vhd** – your MIPS\_ALU design you prepared in HW3 with HW6 modifications
  - **BYOC\_Clock\_driver.vhd** – the CK divider & driver we use for implementation as of HW2.
  - **BYOC\_Host\_Intf.ngc** - The actual infrastructure interfacing the PC.
4. Now run the Xilinx ISE SW, create a **HW6\_top.bit** file and test it by running **Pong1\_v32.txt** program.

## HW6 top – testing the implemented design

We'll run that the **BYOCInterface** SW and load the IMem. Then run the circuit. If we have issues, we will run the circuit in a single ck mode and check that the reading we see at the points we "hooked" to the rdbk signals are as what we expect.

The file we want to load into the IMem is called "**Pong1\_v32.txt**". The file itself includes all the information required in order to load it into the IMem and switch to a single ck mode. Following the loading, we can run it in ck on mode. This is a simple Pong game and we should be able to control it via the right & left arrows on the keyboard.

For that we should connect a VGA screen to the Nexys2 board, and a PS2 keyboard

- What happens when you press the RUN button?

## HW5 top – implementation report

You should submit a single zip file for the Simulation and implementation phases. It should have **four** directories/folders. The first is called **Simulation1**, the 2<sup>nd</sup> is called **Simulation2**, the 3<sup>rd</sup> is called **Simulation 3**, the 4<sup>th</sup> is called **Implementation**.

In the **Implementation** directory you should have 2 sub-directories:

- **Src\_4ISE** – here you put all of the \*.vhd sources and the \*.ucf file (and no TB file)
- **ISE** – here you should have the HW6 project created by the Xilinx ISE SW.

**As part of completing this part of the course you will have to show me how you run the design on the Nexys2 board in the lab. And maybe answer some questions.**

**If the game works,  
CONGRATULATIONS!!**

**You actually  
built your own computer!!!**



**Enjoy the assignment!**

**Thanks for  
listening!**