Scalar Pipelined MIPS CPU Architecture

Laboratory five assignment

Hanan Ribo

10.06.22

Table of contents

1. Aim of the Assignment

- Design, synthesis and analysis of a simple MIPS compatible CPU.
- Scalar Pipelined MIPS CPU Architecture (from single cycle MIPS CPU Architecture)
- Understanding of CPU vs. MCU concept.
- Understanding in FPGA memory structure.

2. Definition and prior knowledge

The aim of this laboratory is to design a Scalar Pipelined MIPS CPU from a single cycle MIPS compatible CPU. The CPU core must be capable of performing instructions from a given MIPS instruction set. The design will be executed on the Altera Board. The MIPS architecture is Harvard architecture in order to increase throughput and simplify the logic. **You are required to support the Assembly Instruction Set given in the dedicated attached file**. For additional information regarding MIPS CPU, Architecture, ISA and instructions see MIPS technical documents [1].

3. Assignment definition

The architecture must include a MIPS ISA compatible CPU with data and program memory L1 Caches for hosting data and code. The block diagram of the architecture is given in Figure 1. The CPU will have a standard MIPS register file. The top level and the MIPS core must be structural. The design must be compiled and loaded to the Altera board for testing. A single clock (CLK) should be used in the design.

Note: use push-button KEY0 as a System RESET (brings PC to the first program command) and SW9 as PC and CPU register stages ENA .

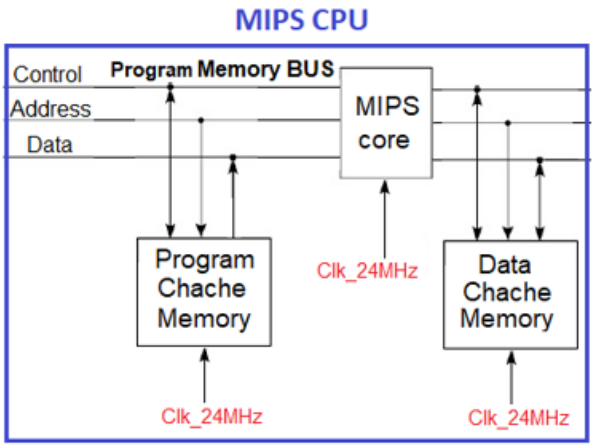
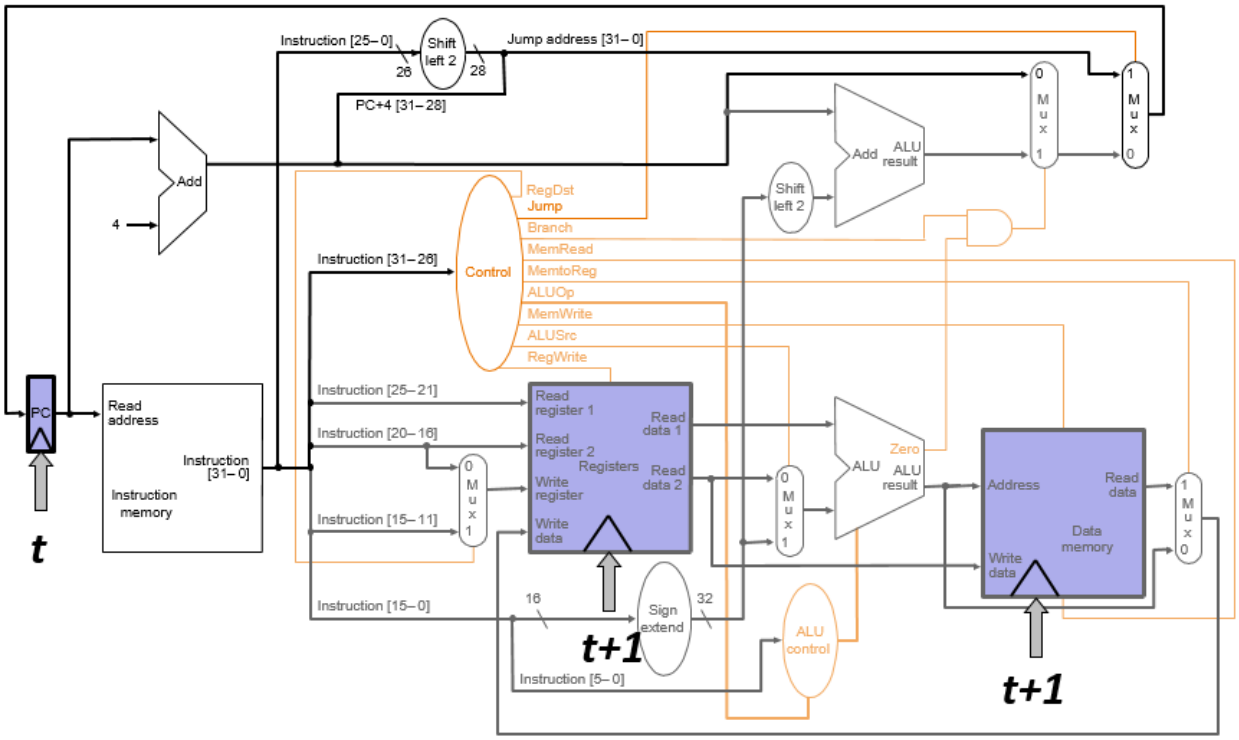


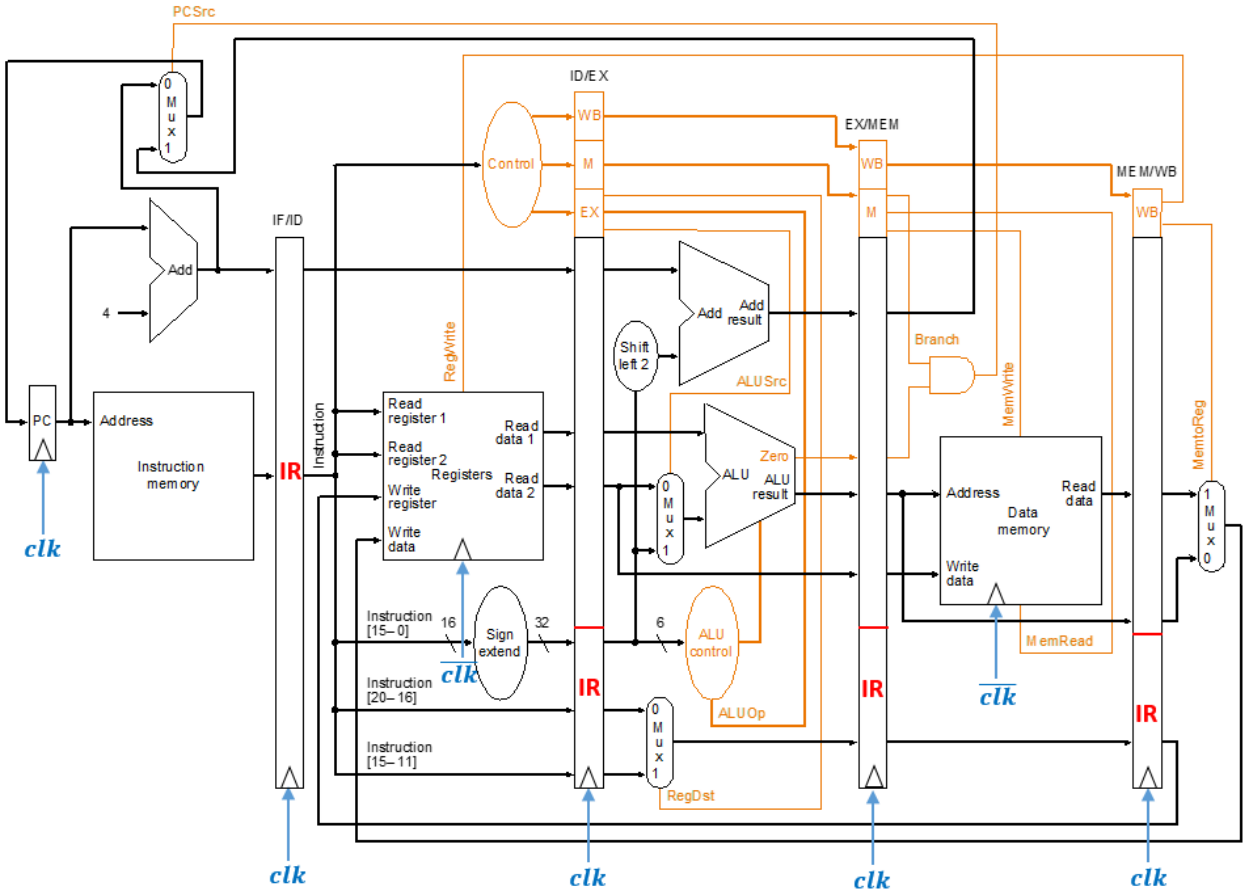
Figure 1 : System architecture

4. Single cycle and Pipelined MIPS architecture diagrams

Single cycle MIPS diagram – figure 2:



Pipelined MIPS diagram (without interlocking) – figure 3:



5. Pipelined MIPS architecture requirements

The single cycle MIPS architecture you have given is need to be elevated to a pipelined MIPS architecture from one of two options of your choice (the second option qualifies bonus points of 25%). Option 1 is pipelined MIPS architecture with Stall detection Logic based on Combinational Check approach as Data Hazard detection unit (figure 4), this option is the basis mandatory. Option 2 is pipelined MIPS architecture with Data Hazard detection unit (as in option 1) and Data Forwarding Unit when branches are resolved in single delay slot (figures 5, 6).

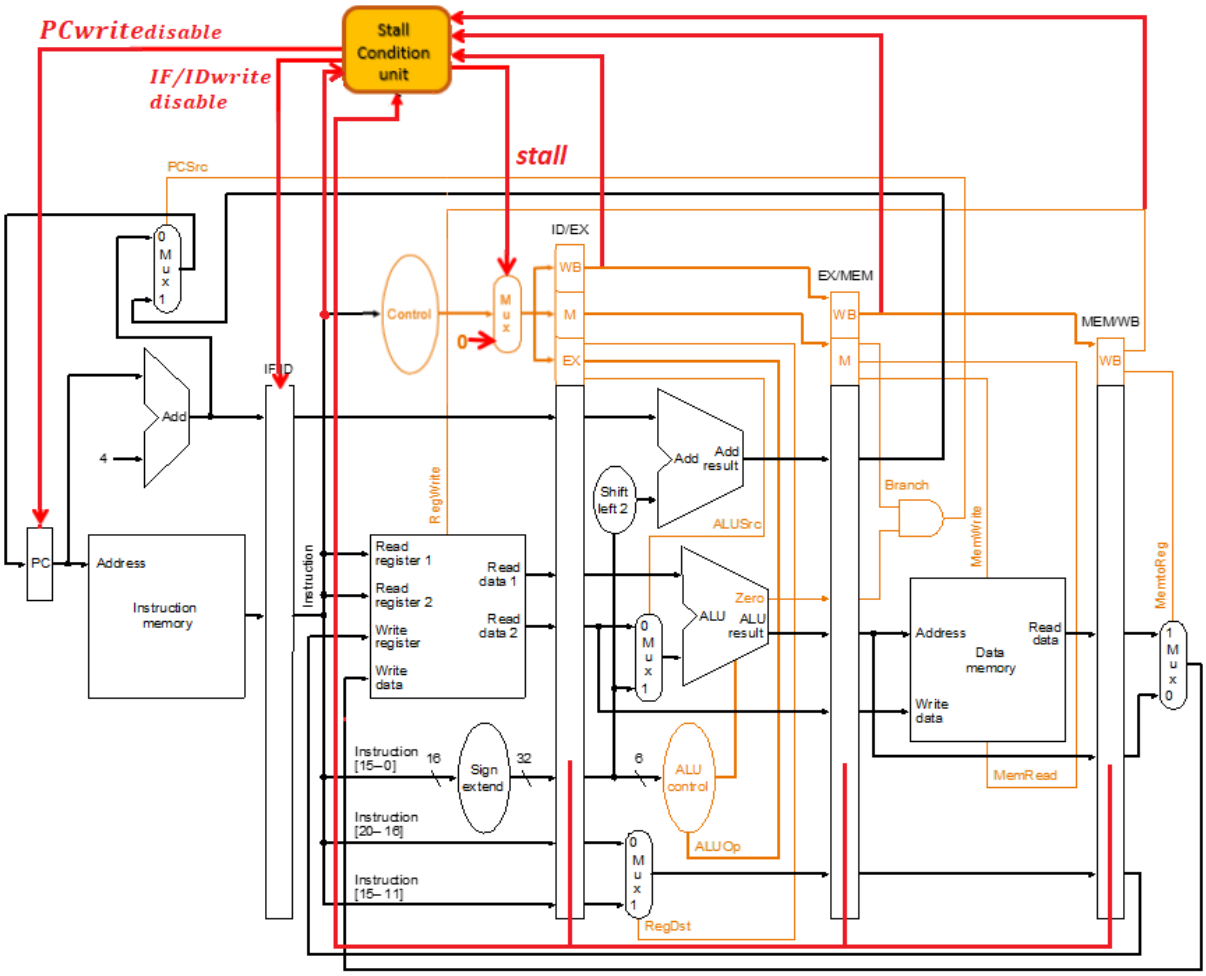


Figure 4: Option 1 is pipelined MIPS architecture

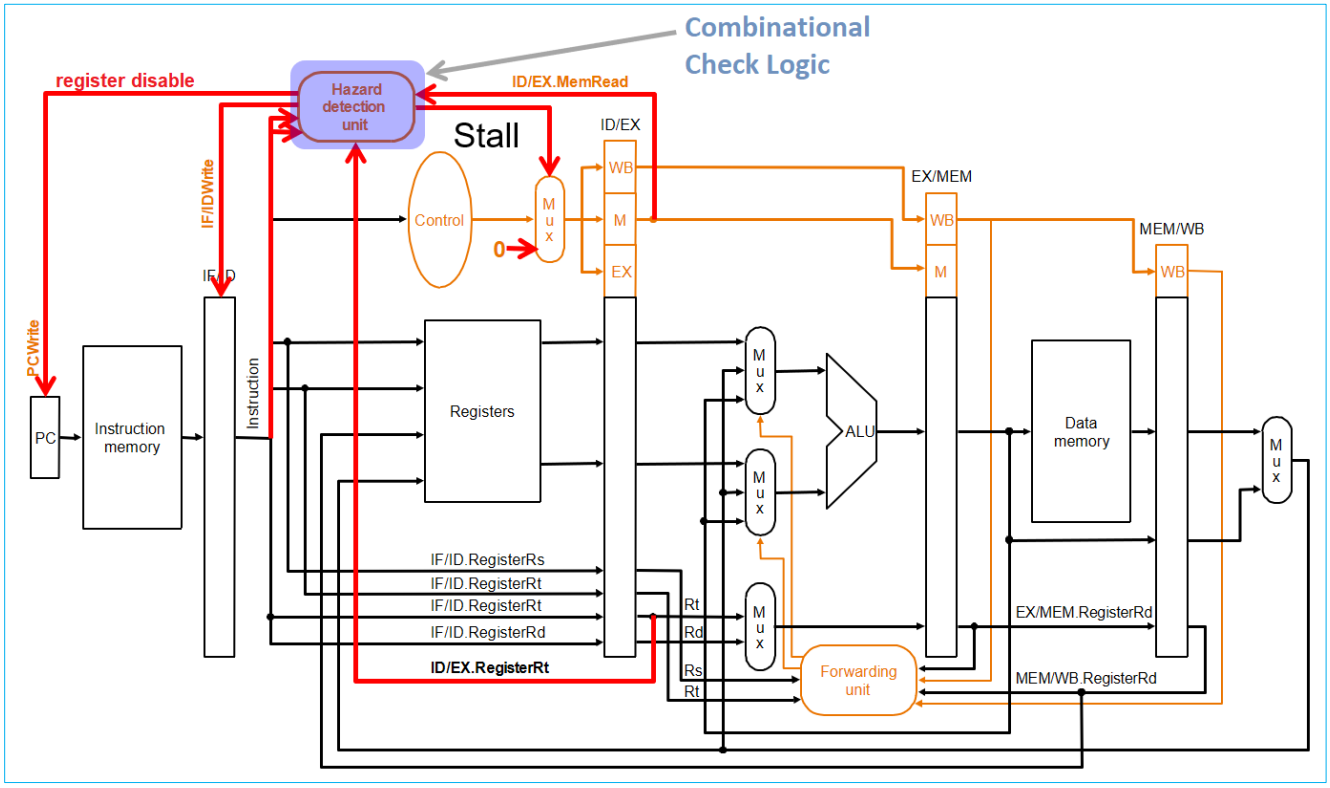


Figure 5: Option 2 pipelined MIPS architecture

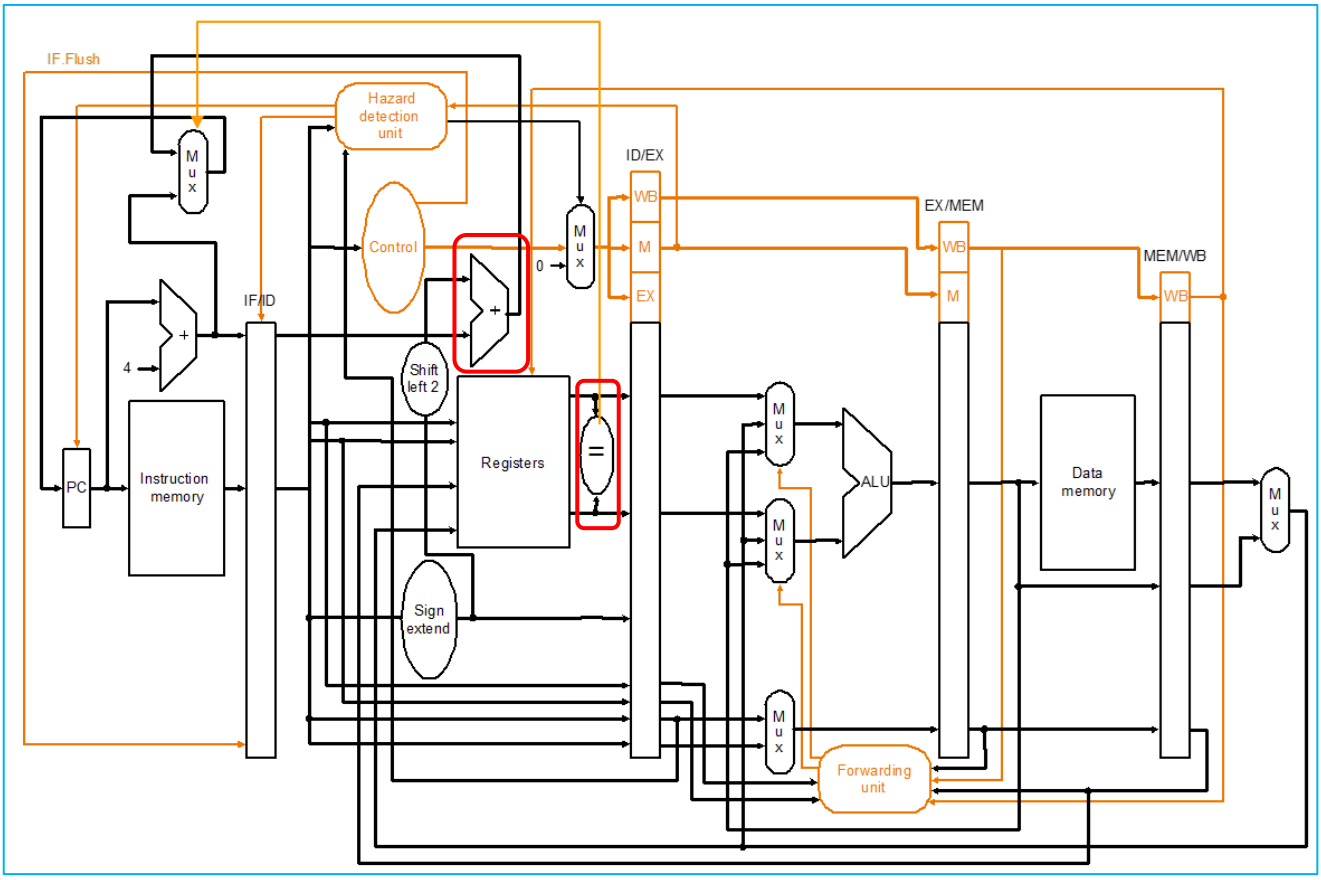


Figure 6: Option 2 pipelined MIPS architecture with single delay slot

6. System design flow

The development design flow is contained from functional verification using ModelSim (start phase) with the ability to discern each line in the design and electrical verification using signal tap (final phase) with limited ability to discern only few lines. In order discern important lines in pipelined MIPS CPU using signal tap, there is need to add these desirable lines to the top entity (in comparison to single cycle top entity you are given – figure 7).

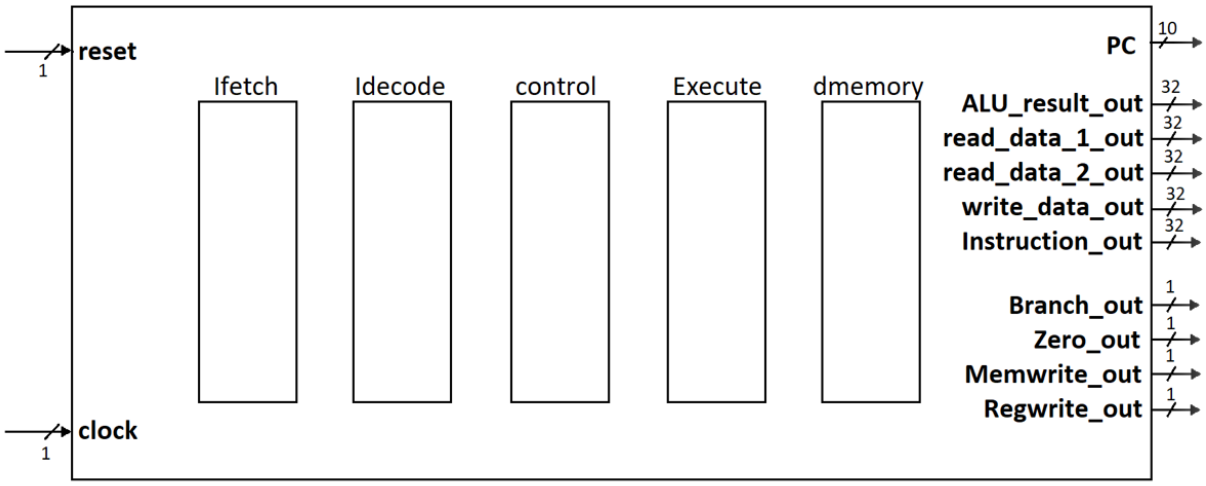


Figure 7: Single cycle MIPS architecture top entity

In addition to desirable lines that you add to the pipelined MIPS top level entity, you are required to add the next three important registers to support CPI/IPC calculation and break point debug ability.

i. **CLKCNT** (clock counter register) 16 bit register - **CLKCNT** cleared on reset:

This register is a clock cycles counter counts on clock rising edge.

ii. **STCNT** (stall counter register) 8 bit register - **STCNT** cleared on reset:

This register is a core stalls counter counts on clock rising edge when a stall occurred.

iii. **FHCNT** (flush counter register) 8 bit register - **FHCNT** cleared on reset:

This register is a core flush counter counts on clock rising edge when a flush occurred.

iv. **BPADD** (break point address) 10 bit register:

This register uses as break point address in order to feed logic that issues signal tap trigger event. This signal tap trigger enables to track CPU core lines from any PC address on until the signal tap buffer is full. **BPADD** cleared on reset. Using Auto run button in signal tap window we can debug the CPU core as long as we wish in only one signal tap session. **BPADD** register is fed by SW7-SW0 located on Altera board and padded with two zeroes. Signal tap trigger equation is:

$$STtrigger = (PC == BPADD)$$

Performance: $IPC = \frac{\mathbf{CLKCNT} - (STCNT + 4 + \mathbf{FHCNT} * \text{dept})}{\mathbf{CLKCNT}}$ where: IPC=1/CPI

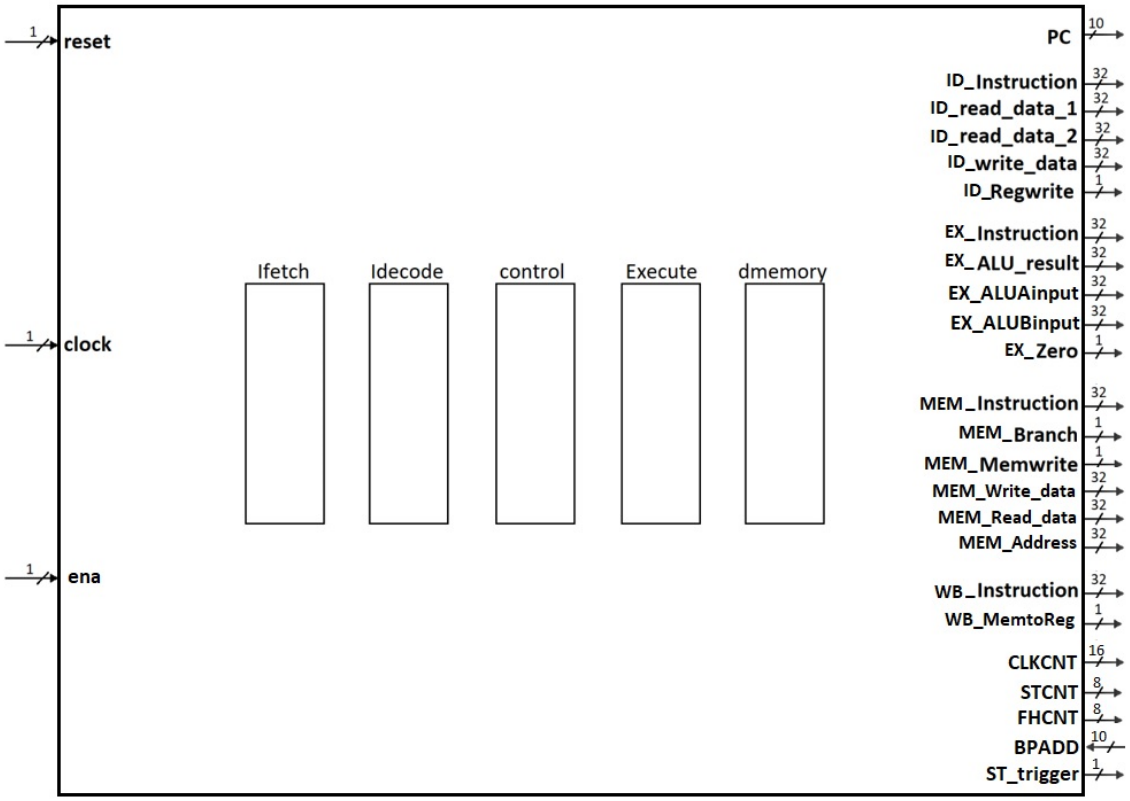


Figure 8: Pipelined MIPS architecture top entity

- The CPU will be based the *standard 32bit MIPS ISA* and the Instructions will be 32 bit wide. The following table shows the MIPS instruction format. For more information, see MIPS technical documents [1].

Type	**-31-**		**format (bits)**			**-0-**
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rt (5)	immediate (16)		
J	opcode (6)	address (26)				

Table 1 : MIPS Instruction format

The Data address space is 4kB. Memory latency will be according to Table 2

Memory	**Write Latency**	**Read Latency**
Program Memory (I-Cache)	1 clk	1 clk
Data Memory (D-Cache)	1 clk	1 clk

Table 2 : Memory sizes and latency

7. Compiler, Simulator and Memory

The MARS compiler and simulator, or any other can be used to compile and simulate the assembly code. MARS compiler can also export the memory contents into the file in format that VHDL can easily read. It can also simulate a cache performance.

The mars compiler, installation instructions and documentation are available at: http://courses.missouristate.edu/KenVollmar/MARS/

8. CPU and MCU Test

a) As a test bench, you need to write assembly code, which compiles the next C code. At the end, calculate **IPC** using ***CLKCNT***, ***STCNT***, **FHCNT** registers and compare the results to the predicted manual **IPC** calculation.

Note: see on Moodle file contains the Required Support of Assembly Instruction set.

```
#define M 16

void Transpose(int Mat1[M][M], int Mat2[M][M]){define it yourself …}

void main(){
     int Mat1={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16},Mat2[M]; //int=32bit
     Transpose(Mat1, Mat2); // Mat2 = Transpose(Mat1)
}
```

b) Test the performance of your pipelined MIPS CPU design with a suggested code contains many data and control hazards. Calculate the IPC and make a comparison with the theory where the hazards are marked on written source code.

9. Requirements

You have to do the following tasks:

- ModelSim Simulation with maximal coverage.
- Analyze the critical path, explain where it is in your VHDL design and find the maximal operating clock.
- Load the design onto the FPGA and verify the simulation results.
- **Run the required assembly source codes and explore them.**

The following must be presented in **Lab5.pdf** report file.

1. Top level block review diagram of your design.
2. For each block in the top level design:
   - RTL Viewer results
   - Logic usage for each block (Combinational and Flip-Flops).
   - Graphical description (a square with ports going in and out).
   - Port Table (direction, size, functionality).
   - Short description.
3. Maximum (Critical) path of your design – explain where it is in the code and how it is possible to optimize if you would have more time. What is the maximum clock frequency?
4. Minimum path analysis.
5. Documentation Style - Content with page numbers, Images and tables will be numbered. The caption of an images and tables below the images or tables.
6. Elaborated analysis and wave forms:
   - Maximal Frequency and critical paths from Timing Analyzer.
   - Proof of work using Signal Tap shot screens.
     Recall that, proof of work using Signal Tap is mandatory.
   - **One** basic waveform to explain the system timing.

Design requirements:

1. The design must be well commented.
2. The system must work from only one clock.
3. System RESET (KEY0) must be synchronous.
4. Conclusions
5. A ZIP file in the form of **id1_id2.zip** (where id1 and id2 are the identification number of the submitters, and id1 < id2) *must be upload to Moodle only by student with id1* (any of these rules violation disqualifies the task submission).
6. The **ZIP** file will contain the next six subdirectories (***only the exact next sub folders***):

Directory	**Contains**	**Comments**
VHDL	Project VHDL files	Only VHDL files, excluding test bench **Note: your project files must be well compiled (in ModelSim and Quartus separately) without errors as a basic condition before submission**
TB	VHDL files that are used for test bench	Only one **tb.vhd** for the overall DUT
SIM	ModelSim DO files	Only for **tb.vhd** of the overall DUT

DOC	Project documentation	**Readme.txt** and **Lab5.pdf** full report file
Quartus	• Signal Tap files used in project verification • Project SOF file • Project SDC file	Do not place files that are not relevant for compilation or is a result of compilation!
CODE	The assembly source code of clause 5a	

Table 3 : Directory Structure

10. Grading policy

Weight	Task	Description
10%	Documentation	The "clear" way in which you presented the requirements and the analysis and conclusions on the work you've done
90%	Analysis and Test	The correct analysis of the system (under the requirements)

Table 4: Grading

Late submissions will be not gotten.

11. References

[1]. MIPS32® Architecture for Programmers Volume I to III (from Moodle under Final Project)
[2]. ALTPLL User Guide: http://www.altera.com/literature/ug/ug_altpll.pdf
[3]. Altera RAM user guide: http://www.altera.com/literature/ug/ug_ram_rom.pdf
[4]. Altera MegaFunction User Guide: www.altera.com/literature/ug/ug_intro_to_megafunctions.pdf
[5]. Bin2Hex utility
32bit - http://www.keil.com/download/docs/113.asp
64bit - http://www.ht-lab.com/freeutils/bin2hex/bin2hex.html