Laboratory 05

Single-Cycle MIPS CPU Design – smaller: 16-bits version

One clock cycle per instruction

1. Objectives

Study, design, implement and test

- **Instruction Fetch Unit for the 16-bit Single-Cycle MIPS CPU**

Familiarize the students with

- Single-Cycle CPU design: Defining the instructions / writing the test program (MIPS assembly language, machine code)
- Xilinx® ISE Webpack
- Digilent Development Boards (DDB)
  - **Digilent Basys Board – Reference Manual**

2. Reduced Size MIPS Processor Description

(!) Read Lectures 3 and 4 in order to understand the works needed in this laboratory.

Remember that an instruction execution cycle (lecture 4) has the following phases:

- IF – Instruction Fetch
- ID/OF – Instruction Decode / Operand Fetch
- EX – Execute
- MEM – Memory
- WB – Write Back

Your own Single-Cycle MIPS 16 processor implementation (that you will start in this laboratory and finish in the next ones) will be partitioned in 5 (five) components (new entities). These components will be declared and instantiated in the “test_env” project.

The utility of this implementation will be understood in the future laboratories, when you will implement the pipeline version of your 16-bit MIPS processor!

In this laboratory you will design, VHDL description, implement and test the Instruction Fetch Unit of your own single cycle MIPS processor – MIPS 16.

The data-path of the processor (32-bit version), including the control unit and the necessary control signals, is presented in the next figure. In order to reduce the complexity of the data-path the control signals were not explicitly connected, but rather they can be easily identified by their names.

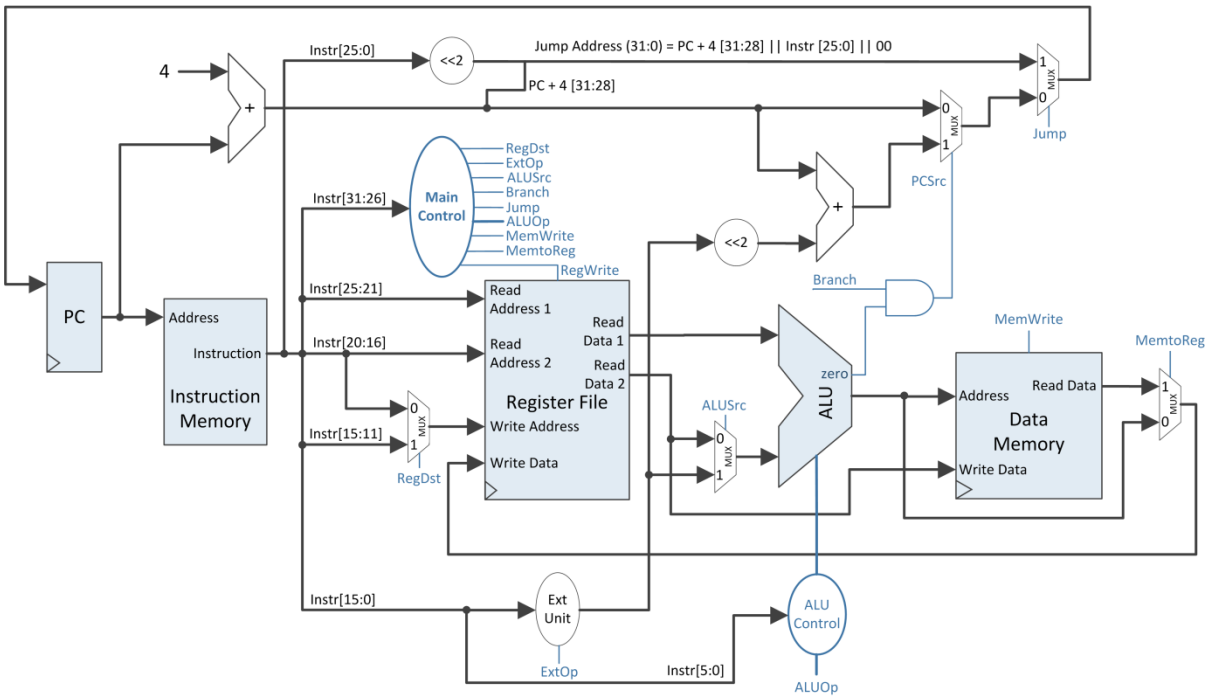

Figure 1: MIPS 32 Single-Cycle Data-Path + Control

As a reminder the instruction formats for your MIPS 16 processor are presented below:

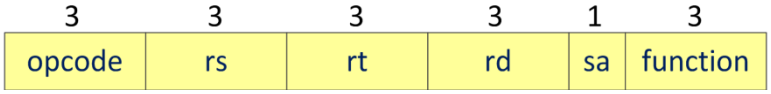

Figure 2: R-type Instruction format

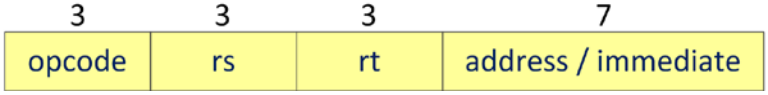

Figure 3: I-type Instruction format

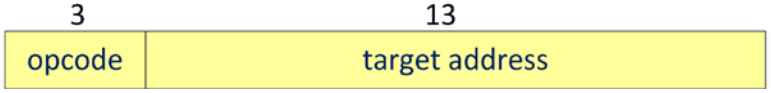

Figure 4: J-type Instruction format

The IF (Instruction Fetch) unit consists in the following components (you will not declare new entities):

- Program Counter
- Instruction Memory (ROM)
- Adder

In addition there exist two multiplexers for selecting the next instruction address. Please refer to the previous laboratory for the characteristics of these components for your Single-Cycle MIPS 16 processor. The data-path of the Instruction Fetch Unit is presented in Figure 5.

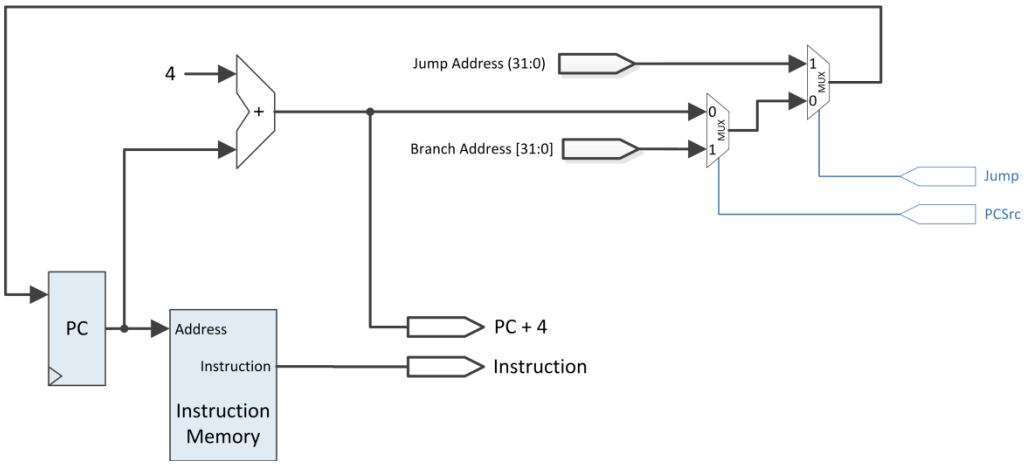

Figure 5: Instruction Fetch Data-Path for MIPS 32

Usually the IF unit provides, as output, the instruction to be executed as well as the next sequential instruction address. In the case of jump or branch instructions, the IF unit must also receive, as inputs, the branch target address and the jump address, together with the control signals that will select the next instruction address.

The inputs of the IF unit are:

- The clock signal (for the PC)
- The branch target address
- The jump address
- Jump Control signal
- PCSrc Control signal (for branch)

The outputs of the IF unit are:

- The instruction to be executed by the MIPS processor
- The next sequential instruction address (PC + 4)

The meaning of the control signals:

- Jump = 1 → PC ← jump address
- Jump = 0
  - If PcSrc = 0 → PC ← PC + 4
  - If PcSrc = 1 → PC ← branch target address

3. Laboratory Assignments

Read carefully and completely each activity before you begin!

Prerequisites:

- You need to have all the assignments from the previous lab completed
- 15 instructions for your own Single-Cycle MIPS 16 defined (Lab04 Assignment 3.1)
- RTL abstract / instruction formats written on paper for all the 15 instructions
- Data-Path for MIPS 16 – paper and pencil (Lab 04 Assignment 3.3; use Figure 1 from this laboratory as reference, or lecture 04 for more details)
- Xilinx project with test_env including at least the ROM with the test program written in machine code (Instruction Memory) (Lab 04 Assignment 3.2)

Attention: If the homework from the previous laboratory is not completed you will receive a 1 for this and all future laboratories until the homework is done without the possibility of any corrections to the mark!

3.1. Instruction Fetch design

Taking into account the instruction fetch data-path from Figure 5 design a new component (new entity) in the "test_env" project for your own single-cycle MIPS 16. All the data fields are 16-bits wide.

The IF entity will contain the hardware components described in Figure 5, that will not be implemented with other components (use behavioral VHDL description).

The instruction memory will be the ROM memory from the previous laboratory with the program written in machine code. Do not increase the memory size to 2^{16} locations, but rather use a subset of the program counter to address the ROM memory (least significant 8 bits).

The adder will be implemented with + 1 in the VHDL description (not + 4 – MIPS32 case).

The program counter register will be a rising edge triggered D Flip-Flop with all the necessary input addresses (sequential operation (PC + 1), branch target address and jump address).

Attention! The new PC value will only be written in the PC register when a button from the Basys board is pressed (use one enable signal from the MPG as input to the IF Unit in order to activate the write of the PC register). Additionally use another button (MPG enable signal) to reset the PC register (another input for the IF Unit).

3.2. Testing of the Instruction Fetch Unit

In the “test_env” entity declare and instantiate your Instruction Fetch Unit. Connect the IF Unit together with the MPG and SSD components previously designed. Use two enable signals from the MPG to reset and to validate the writing in the PC register. For connecting to the SSD use both outputs of the instruction fetch unit (instruction and PC+1).

Use sw(7) to control the display on the SSD (use a multiplexor):

- sw(7) = 0 → display the instruction on the SSD
- sw(7) = 1 → display the next sequential PC (PC + 1 output) on the SSD

For implementing and testing conditional and unconditional jumps you will map the two control signals to two switches:

- Use sw(0) for Jump control signal.
- Use sw(1) for the PCSrc control signal.

Use “hard-coded” values for the branch target address and jump address inputs of the instruction fetch unit:

- you can use x”0000” for jump as an alternative reset mechanism for the PC register (jump to the first instruction in the ROM)
- use an intermediate address for the branch target address. This address must be an address of an instruction in your program (within the range of your previously implemented program in the ROM – machine code)

4. References

- Computer Architecture Lectures 3 & 4 slides.
- MIPS® Architecture For Programmers, Volume I-A: Introduction to the MIPS32® Architecture, Document Number: MD00082, Revision 5.01, December 15, 2012
- MIPS® Architecture For Programmers Volume II-A: The MIPS32® Instruction Set Manual, Revision 6.02
- MIPS32® Architecture for Programmers Volume IV-a: The MIPS16e™ Application-Specific Extension to the MIPS32™ Architecture, Revision 2.62.
    - Chapter 3: The MIPS16e™ Application-Specific Extension to the MIPS32® Architecture.