Laboratory 4

4. Single-Cycle MIPS CPU Design: 16-bits version – One clock cycle per instruction

4.1 Objectives

Study, design, implement and test

- **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
  - Digilent Basys 2 Board – Reference Manual
  - Digilent Basys 3 Board – Reference Manual

4.2 Reduced Size MIPS Processor Description

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

In this laboratory, you will design and start the implementation of your own single cycle MIPS processor – MIPS 16.

The microprocessor will be a simpler version of the MIPS 32 microarchitecture described during the lectures. What does simpler mean? The instruction set will be smaller (fewer instructions to implement); the width of the instructions and data fields will be of 16-bits. Implicitly, the number of registers used in the register file will be smaller; the instruction and data memories will be smaller. The rest of the principles described during the lectures are the same (data-path and control).

The main reason for implementing MIPS 16 is the reduced methodologies for data display (8 or 16 LEDs and 4-digit Seven Segment Display). In this manner, one avoids using other multiplexing mechanisms for signal display purposes (32 bits); and the on-chip debugging process is simplified (testing your program on the FPGA board).

The dimension/width of both instructions and data will be of 16-bits. The 3 instruction formats are given below. Compare this format with the 32-bit instruction format from the lectures. Observe the differences/limitations.

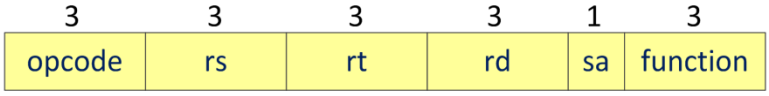

Figure 4-1: R-type Instruction format

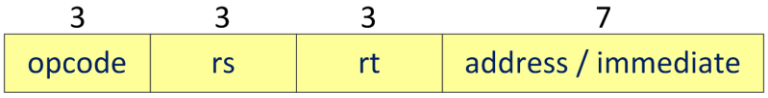

Figure 4-2: I-type Instruction format

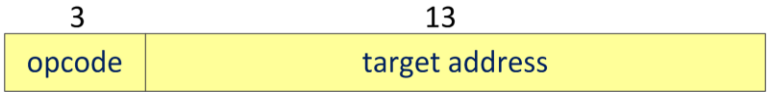

Figure 4-3: J-type Instruction format

These instruction formats obey the formats presented in the MIPS32 ISA, except the width of each field.

The opcode is encoded on 3-bits. For I-type and J-type instructions, the opcode uniquely encodes the instruction to be executed. In the case of R-type instructions, in accordance to the MIPS standard, the opcode is 0 and the function field identifies the ALU operation for each instruction. The function field is encoded on 3-bits. This means that your processor can implement at most 15 instructions:

- 8 R-type Instructions
- 7 I-type Instructions and J-type instructions.

The table below presents the minimum number of instructions, of each type, that will be implemented on the MIPS 16 processor. On the doted positions, you will choose or define new instructions for your MIPS processor (depending on the program that you will implement).

R-type Instructions	Addition	add
	Subtraction	sub
	Shift Left Logical (with shift amount – sa)	sll
	Shift Right Logical (with shift amount – sa)	srl
	Logical AND	and
	Logical OR	or
	….	…
	….	…
I-type Instructions	Add Immediate	addi
	Load Word	lw
	Store Word	sw
	Branch on Equal	beq
	….	…

	….	…
J-type Instruction	Jump	j

Table 4.1: Instructions for MIPS16

The description of each MIPS 16 data-path component characteristics is given below. (!) These characteristics are not only valid for this laboratory, but also for the future laboratory works.

Program Counter characteristics:

- 16-bit edge triggered D flip-flop

Instruction Memory (ROM) characteristics:

- One input bus: Instruction Address
- One output bus: Instruction Data
- Memory word is 16-bit (selected by instruction address)
- No control signals

Register File characteristics:

- Two read addresses and one write address
- Eight 16-bit registers (rs, rt, rd encoded on 3-bits)
- Two 16-bit data outputs: Read data 1 and Read data 2
- One 16-bit data input: Write Data
- Multiple accesses: 2 asynchronous reads and 1 synchronous (edge triggered) write. During read operation, the register file behaves as a combinational logic block.
- One control signal RegWrite. When RegWrite is asserted the value on the Write Data line is written in the register indicated by the write address line

Data Memory (RAM) characteristics:

- One 16-bit input address bus: Address
- One 16-bit input data bus: Write Data
- One 16-bit output data bus: Read Data
- One control Signal: MemWrite

Extension Unit characteristics:

- ExtOp = 1 → Sign Extension
- ExtOp = 0 → Zero Extension

ALU characteristics:

- ALU performs arithmetical / logical operations
- (!) You need to identify all the operations that the ALU needs to perform after completing the definition of the instructions from Table 4.1. You are encouraged to choose two more R-type instructions and two more I-type Instructions.
- You need to identify how many control bits are necessary to encode the ALU operations (ALUCtrl).

4.3 Laboratory Assignments

Read carefully and completely each activity before you begin!

4.3.1. Define the instructions for MIPS 16 – Paper and Pencil

Starting from the instruction formats presented in the previous section write (Paper and Pencil) the instruction format (on bits, including the opcode and function fields) for each of the instructions presented in Table 4.1.

Add two more R-type instructions and 2 more I-type instructions in order to complete the whole number of instructions that your processor is capable of performing.

Besides the lecture materials, you can also use Appendix 6 – MIPS Instruction Reference for a reference on MIPS instructions.

You need to specify the encoding (on bits) in all of the fields from the instruction.

Write the RTL abstract for all the 15 instructions from your MIPS 16 instruction set. Draw the processing diagram for all the instructions (add, and, sll, lw, beq, j – in the laboratory, the rest as homework).

For your MIPS 16 processor you will ignore the overflow exceptions that can appear during ALU operations (example: add instruction)

Give an example for each instruction including the bit encoding off all fields (including the instruction operands). Example: add $2, $4, $3 → "… the 16 bits…".

Attention: in order to increase the encoding readability of each instruction use the "_" symbol between the instruction fields (opcode, rs, etc.). This is also supported by VHDL and has no effect on the bit string. For VHDL it is mandatory to specify the binary encoding B before the string of bits (or X, O for hexadecimal and octal encoding respectively).

`B"001_010_011_100_1_111"` is equivalent to `"0010100111001111"`

4.3.2. MIPS 16 test program

Write a short program with the instructions that you have defined for your MISP 16 processor (paper and pencil). Describe the program in assembly language, then each instruction in machine code (16-bit encoding for each instruction, use "_" between the fields of the instructions).

Using assignment 3.1 from laboratory 3 (ROM memory whose addresses are generated by a Mono Pulse Generator controlled counter), introduce your program in

the ROM memory and trace it on the development board. When writing your program in the ROM memory you have to write a comment for each instruction, i.e. the assembly language description for each instruction. Your program should be visible in parallel to the machine code.

Optionally, you are invited to write a more complex program for your MIPS processor.

4.3.3. Data-Path for MIPS 16

Draw the Data-Path of your single-cycle MIPS 16 processor. Be sure to include all the necessary components on the data-path, such that all the 15 instructions execute correctly.

Starting from the RTL abstract description, identify the values of the control signals for each instruction. Draw a table with the control signals and their values (see lecture 4 for details).

4.4 References

[1] Computer Architecture Lectures 3 & 4 slides.
[2] D. A. Patterson, J. L. Hennessy, “Computer Organization and Design: The Hardware/Software Interface”, 5th edition, ed. Morgan–Kaufmann, 2013.
[3] D. A. Patterson and J. L. Hennessy, “Computer Organization and Design: A Quantitative Approach”, 5th edition, ed. Morgan-Kaufmann, 2011.
[4] MIPS® Architecture for Programmers, Volume I-A: Introduction to the MIPS32® Architecture, Document Number: MD00082, Revision 5.01, December 15, 2012
[5] MIPS® Architecture for Programmers Volume II-A: The MIPS32® Instruction Set Manual, Revision 6.02
[6] 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.