UNIVERSITY OF CRETE
MICROPROCESSOR & HARDWARE LABORATORY
LABORATORY EXERCISES FOR THE COURSE: HRY 312
COMPUTER ORGANIZATION

SPRING SEMESTER 2017-18
Laboratory 4

Implementation of complex instructions in a processor multiple cycles

Purpose of the Workshop

1. The definition of the architecture of a set of complex instructions.
2. The design and completion of a multi-cycle processor that implements complex commands and will build upon a previous deliverable.
3. Deeper understanding of how a multi-cycle processor works.

Complex Command Architecture

You are invited to implement a non-pipelined processor based on an extended subset of the C-CHARIS (Custom-CHAnia Risc Instruction Set, Version 2) instruction set architecture, which**except for the orders he implemented in previous workshops**, will also implement the following set of commands:

Custom commands:

- addi_MMX_byte
- poly2
- rfld
- rfst

The above commands, depending on their function, can belong to one of the two command formats given to you in 2thelaboratory.
The memory (RAM) addressing of your new datapath will be done with byte addresses, while the words in the memory should be aligned in multiples of 4 Bytes.

The coding of the commands as well as their operation will be done according to the following table:

Opcode	FUNC	MANDATE	ACT	Description
110001	-	MMX_add_byte	RF[rd] - RF[rs](31 downto 24) + immed(7 downto 0) & RF[rs](23 downto 16) + immed(7 downto 0) & RF[rs](15 downto 8) + immed(7 downto 0) & RF[rs](7 downto 0) + immed(7 downto 0)	It “sees” the registers as 4 independent bytes. It adds the corresponding bytes of rs to the last byte of immed and joins the results creating a 32-bit word, which is stored in the rd register.
100000	010000	poly2	RF[rd] - RF[rt] * RF[rt] * MEM[RF[rs]] + RF[rt]* MEM[RF[rs]+4] + MEM[RF[rs]+8]	In this operation, the polynomial 2 is calculated$_u$degreepoly2(x) = $_2$+ + , where x is an integer stored in register rt, and a, b, c are also integers stored in consecutive memory locations. The address of a is described by the rs register, while the values of b and c are located in the following memory locations. Multiplication can be implemented in the ALU through the available VHDL operator.
011100	-	rfld	base_addr = RF[rs] + SignExtend(Imm) for(i=1; i<32; i++) RF[i] - MEM[base_addr + 4*i]	Loads what values of all RF registers from 31 consecutive memory locations.
011110	-	rfst	base_addr = RF[rs] + SignExtend(Imm) for(i=1; i<32; i++) MEM[base_addr + 4*i] - RF[i]	Opposite of rfld, it stores the values of all RF registers in 31 contiguous memory locations.

Conduct

A. Study the coding of specialized commands

Study the encoding of the commands and define for each of them the format to which it belongs.

B. Extend the functionality of CHARIS

B1. Datapath implementation of the ECHARIS processor

Study the datapath you implemented in the previous lab. Find and implement the changes required in the datapath, e.g. multiplexes, registers, counters, for the implementation of the above commands and alsoof all the commands from the previous lab .

ATTENTION!!! For all the commands you will use the basic modules that you implemented in the previous workshops.

B2. Study and implementation of the ECHARIS processor control

Revisit all the Controls you built from the previous lab and make the changes-additions needed to control the execution flow of each new command.

ATTENTION!!! After the changes in the datapath of ECHARIS it may be necessary to make changes again in the control of your processor for the correct implementation of the commands of the previous lab.

Finally, join the datapath of ECHARIS with the control you implemented and simulate the correct operation of all the commands.

Implementation

1. **ATTENTION!!!!! Before any changes to its code previous lab keep a backup of Lab 3 code because we will need it in the next lab.**

2. In order to check the correct operation of the specific laboratory, they should work properly**at least the commandsli, lui and sw** from the previous workshop.
3. Change the VHDL code that implements the datapath of the CHARIS processor so that it can implement the new commands.
4. Add whatever additional logic you need to the datapath (registers, counters, multiplexers, etc.).
5. Change the VHDL code that implements the CHARIS processor control module according to the new needs and name your file: Extended_control.vhd

6. Connect the datapath to CONTROL to implement the full functionality of a multi-cycle processor. Name your file: ECHARIS.vhd

7. Confirm the correct operation of the processor by giving actual commands as input and checking the values of the output signals in each state of the FSM.

8. Simulate and verify processor operation by executing commands one by one (by hand).

9. Simulate and verify processor operation**performing the program you were given that uses the CHARIS extended instruction set (test.data)** with twhich you will initialize IMEM.

Deliverable

1. VHDL code files (source) the day before the lab.
2. Schematic diagram of the finite state machine of the control **noting the changes** that you did in relation to the previous one laboratory.

3. Schematic diagram of the datapath**noting the changes** that you implemented in relation to the previous workshop.
4. Simulation waveforms (cover all cases).
5. Brief reference to the design and implementation process (along with comments and possible problems observed for future improvement of the laboratory).

GOOD LUCK