Lab Assignment #1

EE-126/COMP-46: Computer Engineering w/lab
Professor: Mark Hempstead **TA:** Parnian Mokri, Jessica Nordlund
Tufts University, Fall 2022

Due as per the class calendar, via 'provide'

The ultimate goal of this project is to design a pipelined version of a ARM processor that can detect control and data hazards. The functionality of the processor and its components should match the descriptions in the textbook unless otherwise noted. All work must be your own; copying of code will result in a zero for the project and a report to the administration.

Overview

List of assignments

- Lab 0: Set up modelsim, simulate an AND gate with 2 inputs and 1 output
- **Lab 1: Basic Processor Components and Testbenches**
- Lab 2: Remaining Processor Components including ALU, Memories, and control logic
- Lab 3: Implementation LEGv8: a single cycle processor that executes a subset of ARM v8-64bit ISA
- Lab 4: Pipelined processor with no hardware hazard detection
- Lab 5: Overcoming data-hazards using forwaring and stalling
- Lab 6: Overcoming control hazards by resolving conditional/unconditional branches in ID and using flushing
- Lab 7: Advanced Topics: open-ended team project (groups up to 2 people)

Lab Submission

Please submit your VHDL files *and* a PDF report via 'provide' command on the EE/CS machines. Please follow the announcement on Canvas about Provide to submit your labs and pay attention to messages you get when you try to provide.
VHDL Files: Submit the VHDL source files **(*.vhd)**[1] and any dependencies thereof. Use the entity descriptions provided at the end of this document.[2] These descriptions can also be found in assignment1.zip.
Scripts/Makefile/README: The course staff needs to know how to compile your code and run your testbenches. Please include a README and/or runscripts/Makefile. In addition, do no change the entitity definitions or it will not run with our tests
Report: Submit your report as a PDF**(*.pdf)**. Demonstrate the functionality of your code by providing waveforms as detailed in the Deliverables Section. Label/annotate important signals and events in your waveforms and then provide a brief description of what is happening.

Lab 1 Objectives

- Implement AND2, MUX5, MUX64, SignExtend, Shift Left 2, and PC in VHDL
- Write a simple testbench for each of the above components
- Verify functionality of each component via simulation using Questasim
- Submit a report with the requested waveforms and analysis

[1]Do NOT submit your entire Modelsim project (including but not limited to ***.mpf** and files in **work/**)
[2]Submissions that fail to to follow any of these directions may be penalized at the discretion of the grader. If you have questions, contact the TA (Parnian Mokri: parnian.mokri@tufts.edu).

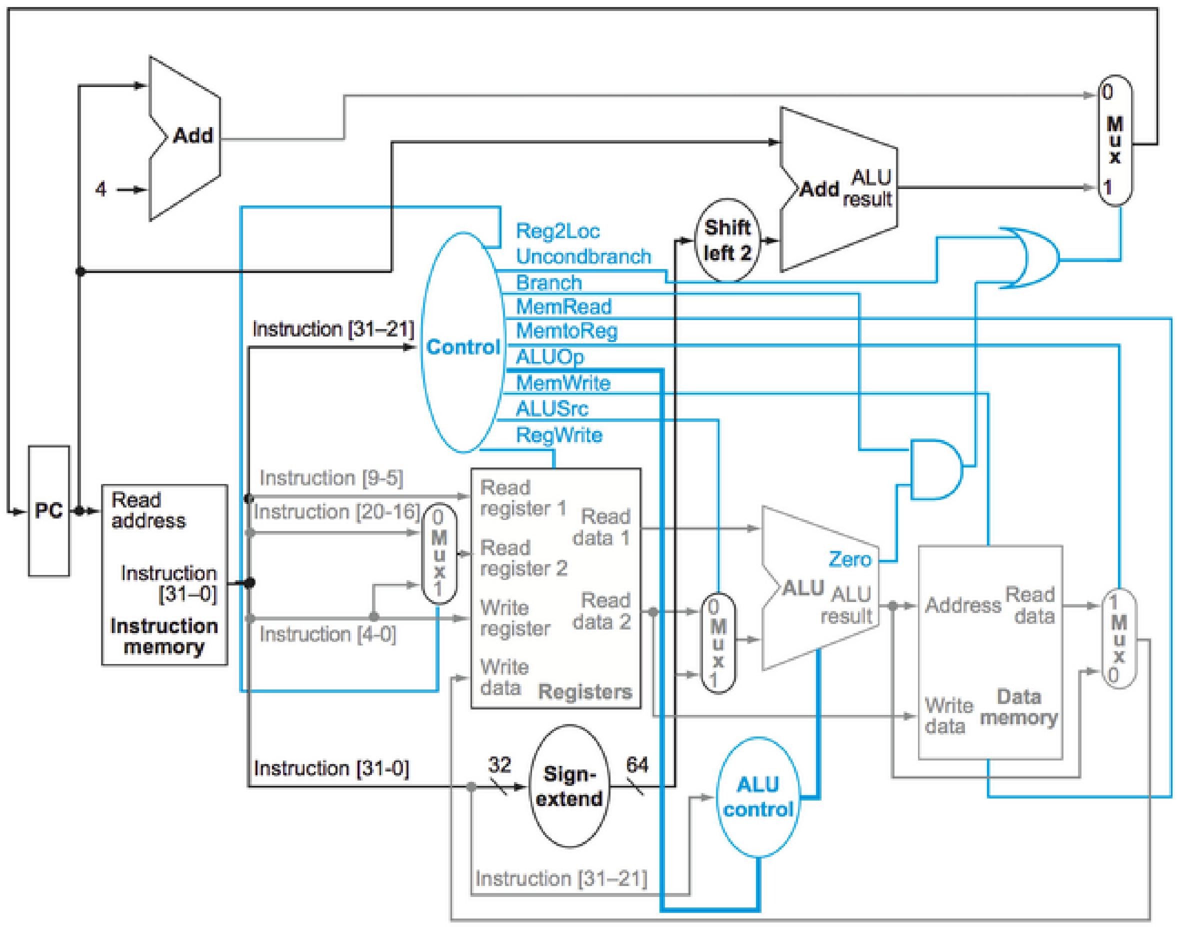

Figure 1: **Components of ARM processor for Lab 1.** Components that will be implemented in subsequent labs are crossed out. *Note*: some component are used in multiple places (Shift left 2, MUX64)– you need only implement these components once (see 'Deliverables' section). This is Figure 4.24 in the textbook.

Deliverables

VHDL Files:
AND2, MUX5, MUX64, SignExtend, Shift Left 2, and PC (with a testbench for each). These components are shown in Figure 1.

Report:

- A brief description (a sentence or two) of what each component does
- A brief explanation of your choices for modeling type (dataflow, behavioral, structural)
- Waveforms obtained by simulating your testbenches with brief descriptions for all the entities. For And2, Mux5, and PC follow these specific steps:
  - AND2/MUX5: (no labeling is necessary)
  - PC: Annotate/highlight functionality of the asynchronous reset and write-enable inputs

Entity Descriptions **(provided in assignment1.zip)**

AND2

```
entity AND2 is
port (
      in0    : in  STD_LOGIC;
      in1    : in  STD_LOGIC;
      output : out STD_LOGIC -- in0 and in1
);
end AND2;
```

MUX5

```
entity MUX5 is -- Two by one mux with 5 bit inputs/outputs
port(
    in0    : in STD_LOGIC_VECTOR(4 downto 0); -- sel == 0
    in1    : in STD_LOGIC_VECTOR(4 downto 0); -- sel == 1
    sel    : in STD_LOGIC; -- selects in0 or in1
    output : out STD_LOGIC_VECTOR(4 downto 0)
);
end MUX5;
```

MUX64

```
entity MUX64 is -- Two by one mux with 64 bit inputs/outputs
port(
    in0    : in STD_LOGIC_VECTOR(63 downto 0); -- sel == 0
    in1    : in STD_LOGIC_VECTOR(63 downto 0); -- sel == 1
    sel    : in STD_LOGIC; -- selects in0 or in1
    output : out STD_LOGIC_VECTOR(63 downto 0)
);
end MUX64;
```

SignExtend

```
entity SignExtend is
port(
     x : in  STD_LOGIC_VECTOR(31 downto 0);
     y : out STD_LOGIC_VECTOR(63 downto 0) -- sign-extend(x)
);
end SignExtend;
```

Shift Left 2

```
entity ShiftLeft2 is -- Shifts the input by 2 bits
port(
     x : in  STD_LOGIC_VECTOR(63 downto 0);
     y : out STD_LOGIC_VECTOR(63 downto 0) -- x << 2
);
end ShiftLeft2;
```

PC (Program Counter)

```
entity PC is -- 64-bit rising-edge triggered register with write-enable and Asynchronous reset
-- For more information on what the PC does, see page 251 in the textbook
port(
     clk          : in  STD_LOGIC; -- Propogate AddressIn to AddressOut on rising edge of clock
     write_enable : in  STD_LOGIC; -- Only write if '1'
     rst          : in  STD_LOGIC; -- Asynchronous reset! Sets AddressOut to 0x0
     AddressIn    : in  STD_LOGIC_VECTOR(63 downto 0); -- Next PC address
     AddressOut   : out STD_LOGIC_VECTOR(63 downto 0) -- Current PC address
);
end PC;
```

Provide

To provide,

- Log in to homework.cs.tufts.edu : ssh -X eecsUsername@homework.cs.tufts.edu
- Type: **provide ee126 lab1 "files"**
- Type **yes** for the following question: Are you ready to provide these for testing (yes or no)?
- Make sure you see the following message:
  your submission has been accepted!
  your submission is complete!