Organization of Digital Computer Lab
EECS 112L/CSE 132L

Lab 1 - Single-cycle ARM Datapath and Control

prepared by: Pooria M.Yaghini

University of California, Irvine

January, 23, 2017

Due on February 5^{th}, 11pm. Note: this is a two-week assignment

1 Goal

In this lab, you will implement the basic processor that runs a subset of the ARM ISA.

1.1 Lab Description

In this lab you will build a simplified ARM single-cycle processor using SystemVerilog. Then you will load a test program and confirm that the system works. Next, you will implement two new instructions, and then write a new test program that confirms that new instructions execute properly as well. By the end of this lab, you should thoroughly understand the internal operation of the ARM single-cycle processor.

All of you will be implementing the same datapath, so you will all run into similar problems. Learn from your classmates, and then go apply what you learn to your own design. The only block which you should design by yourself is the ALU. This block might be the critical path of your processor later when you synthesize it in Lab-2.

Before starting this lab, you should be very familiar with the single-cycle implementation of the ARM processor described in Chapter-4 of reference book or in Appendix-B of supplemental reference book. The simplified single-cycle processor schematic, shown in Figure 1.1, from the text is repeated here for your convenience. This version of the ARM single-cycle processor can execute the following instructions: ***add***, ***sub***, ***and***, ***orr***, ***B***, ***or***, ***slt***, ***lw***, ***sw***.

- Data-processing instructions: **ADD, SUB, AND, ORR** (with register and immediate addressing modes but no shifts)
- Memory instructions: **LDR, STR** (with positive immediate offset)
- Branches: **B**

Our model of the single-cycle ARM processor divides the machine into two major units: the *control* and *datapath*. Each unit is constructed from various functional blocks. For example, as shown in Figure 1.1, the datapath contains the 32-bit ALU, the register file, the sign extension logic, and multiplexers to choose appropriate operands.

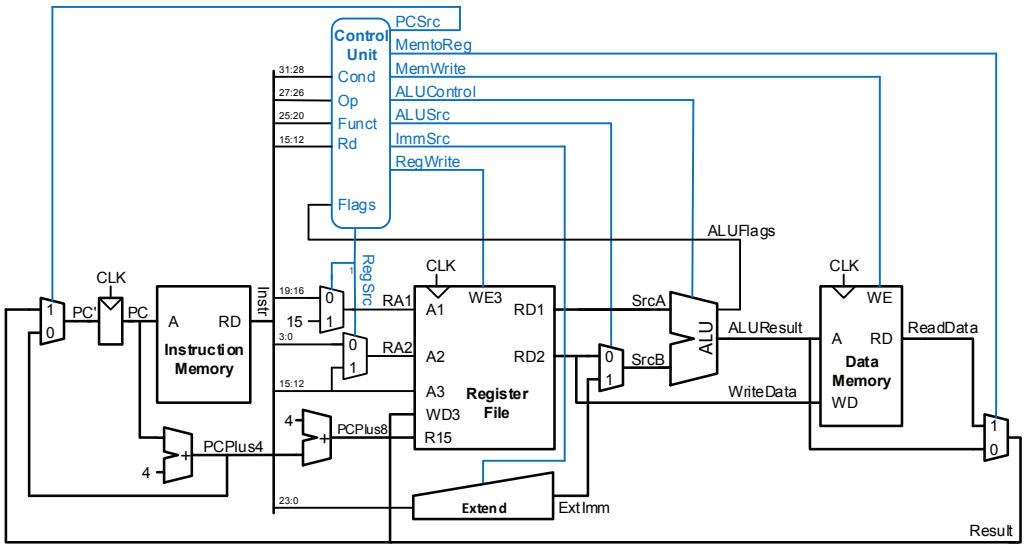

Figure 1: Simplified Single-cycle ARM processor architecture

1.1.1 Processor

It is assumed in this lab that the program instructions are already preloaded in the instruction memory. Later, we will extend the processor to load the memory contents from outside RAM.

Code 1: ARM Processor top module in SystemVerilog

```
module top(
    input  logic clk, reset,
    output logic [31:0] DataAdr,
    output logic [31:0] WriteData,
    output logic MemWrite
    );

    logic [31:0] PC, Instr, ReadData;


    // instantiate processor and memories
    arm  arm  (clk, reset, PC, Instr, MemWrite, DataAdr, WriteData, ReadData);
    imem imem (PC, Instr);
    dmem dmem (clk, MemWrite, DataAdr, WriteData, ReadData);
endmodule


module arm(
    input  logic clk, reset,
    output logic [31:0] PC,
    input  logic [31:0] Instr,
    output logic        MemWrite,
    output logic [31:0] ALUResult,
    output logic [31:0] WriteData,
    input  logic [31:0] ReadData
  );

    // Write your code here ...

endmodule
```

1.1.2 Instruction

The instruction memory for now is a implemented as a Read-Only Memory (ROM) which your processor will read its instructions from. This memory is located outside ARM processor. The test program is given in Figure 7.60 of the textbook. Study the program until you understand what it does.

Code 2: Sample Instruction Memory for testbench

```
module imem(
    input  logic [31:0] a,
    output logic [31:0] rd
  );


    logic [31:0] RAM[63:0];

    // Hardcoded version of the instruction memory in HDL Example 7.15
    // The instructions are from Figure 7.60

    assign RAM[0]  = 32'hE04F000F;
    assign RAM[1]  = 32'hE2802005;
    assign RAM[2]  = 32'hE280300C;
    assign RAM[3]  = 32'hE2437009;
    assign RAM[4]  = 32'hE1874002;
    assign RAM[5]  = 32'hE0035004;
    assign RAM[6]  = 32'hE0855004;
    assign RAM[7]  = 32'hE0558007;
    assign RAM[8]  = 32'h0A00000C;
    assign RAM[9]  = 32'hE0538004;
    assign RAM[10] = 32'hAA000000;
    assign RAM[11] = 32'hE2805000;
    assign RAM[12] = 32'hE0578002;
    assign RAM[13] = 32'hB2857001;
    assign RAM[14] = 32'hE0477002;
    assign RAM[15] = 32'hE5837054;
    assign RAM[16] = 32'hE5902060;
    assign RAM[17] = 32'hE08FF000;
    assign RAM[18] = 32'hE280200E;
    assign RAM[19] = 32'hEA000001;
    assign RAM[20] = 32'hE280200D;
    assign RAM[21] = 32'hE280200A;
    assign RAM[22] = 32'hE5802064;

    assign rd = RAM[a[31:2]]; // word aligned

endmodule
```

1.1.3 Data Memory

This memory is also located outside ARM processor. One way to enhance your processor speed would be to implement SRAM Instruction and Data Caches inside the ARM processor.

Code 3: Sample Data Memory for testbench

```
module dmem(
    input  logic clk, we,
    input  logic [31:0] a,
    input  logic [31:0] wd,
    output logic [31:0] rd
  );

    logic [31:0] RAM[63:0];

    assign rd = RAM[a[31:2]]; // word (4-byte) aligned

    always_ff @(posedge clk)
        if (we)
          RAM[a[31:2]] <= wd;

endmodule
```

1.2 Assignment Deliverables

Your submission should include the following:

- Block diagram of your design in SystemVerilog.
- Design and testbench files.

- A comprehensive report of the Design and testbench architecture. In the report, include the information on how you have make sure the processor works fine, and if you have found any bugs in your design. Include group member names and their tasks and contributions.

IMPORTANT: only **ONE** submission per group is required and the group leader should be the submitter.

Note1: Compress all your files in "zip" or "tar" format and then submit the compressed file.

Note2: The compressed file should include **design, sim, verif, doc** directories and the corresponding files inside each directory. Your report's tex, if in Latex and pdf files are expected to be inside **doc** directory.

Note3: Remember to include the group ID and the name and student ID of each group member in the report. When in doubt about your group ID, please ask TAs.

Note4: Assuming that the parent directory that you are working under that is named `Lab-1`, the following command will compress it for you:

```
$  tar cvf lab-1.tar lab-1
```