BEE 425
Lab 4
ARM Assembly Language CPU
Testing the CPU

Section AA
Spring 2020

Prepared by
Benjamin Coltrane
Young Beum Cho
Chris Gatata
04.14.2020

Instructor
Joseph Decuir

Abstract

In this lab, we run our test program from Lab 3 on a SystemVerilog implementation of an ARM CPU extended by Joseph Decuir.

Introduction

In this lab, the ARM assembly test program written in Lab 3 is executed on a SystemVerilog implementation of an ARM CPU.

A single cycle processor is a processor which carries out one instruction in a single clock cycle. Thus, it is critical for test engineers to create quality testing modules to examine the capability of a single cycle CPU. Specifically, shifting module cases of MOV, LSR, LSL, ROR, ASR instructions were created for analysis and simulation. The below figures represent the data path of data processing operations(op code = 00) such as, but not limited to the shifting functions of a single cycle CPU.

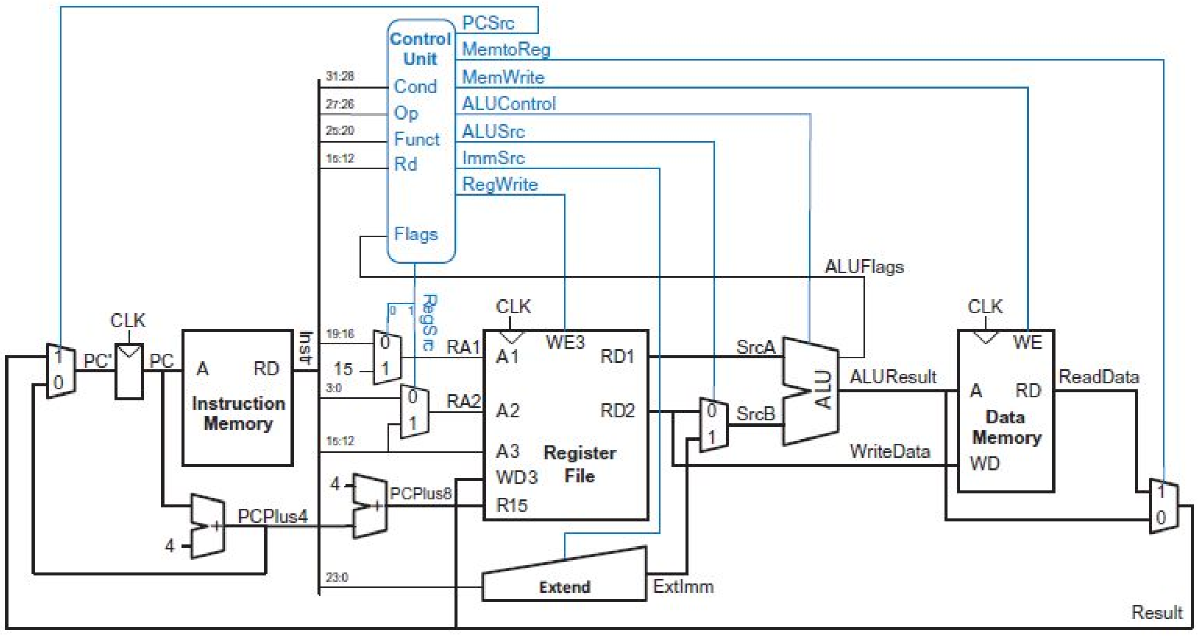
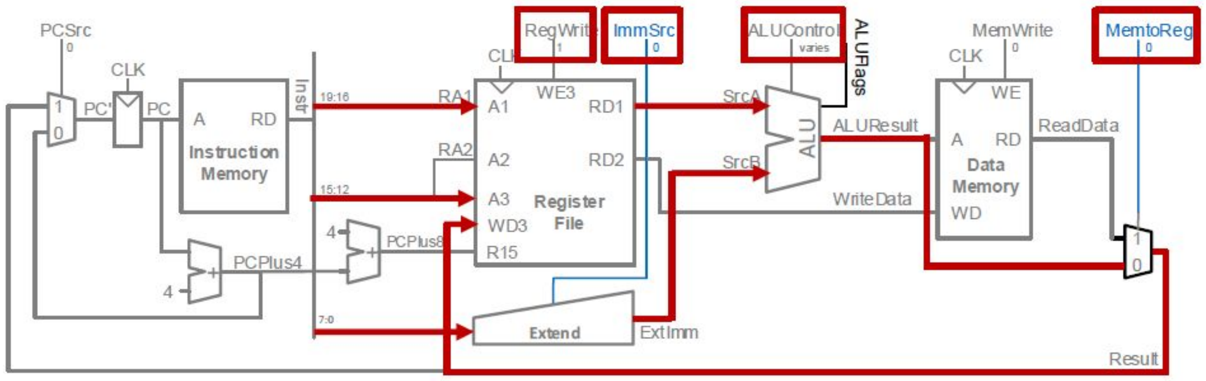


Figure 1.0 Complete Single-Cycle Processor



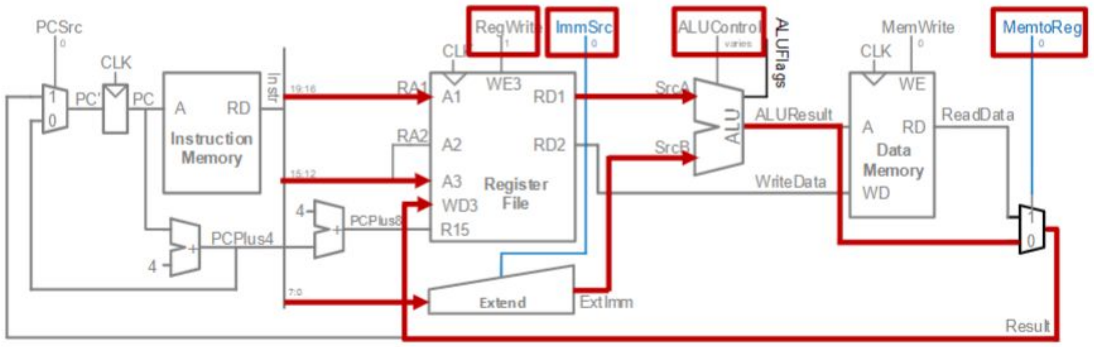
31:28	27:26	25	24:21	20	19:16	15:12	11:8	7:0
cond	op	I	cmd	S	Rn	Rd	rot	imm8

Figure 1.1 Single-Cycle Processor Datapath: Data-Processing

Single-Cycle Datapath: Data-processing

> With immediate Src2:
- Read from `Rn` and `imm8`
- Write *ALUResult* to the register file (Rd)

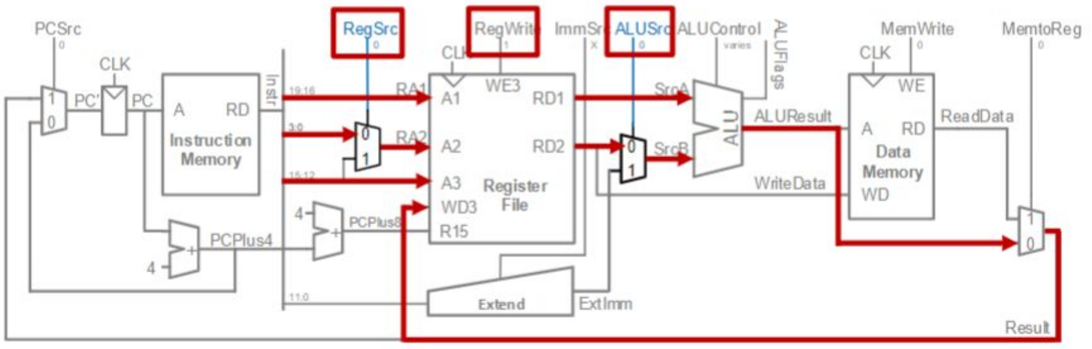
If ImmSrc = 0, ExtImm is zero-extended from $Instr_{7:0}$ for **DP** instr.
If ImmSrc = 1, ExtImm is zero-extended from $Instr_{11:0}$ for **LDR** or **STR**



31:28	27:26	25	24:21	20	19:16	15:12	11:8	7:0
cond	op	I	cmd	S	Rn	Rd	rot	imm8

Single-Cycle Datapath: Data-processing

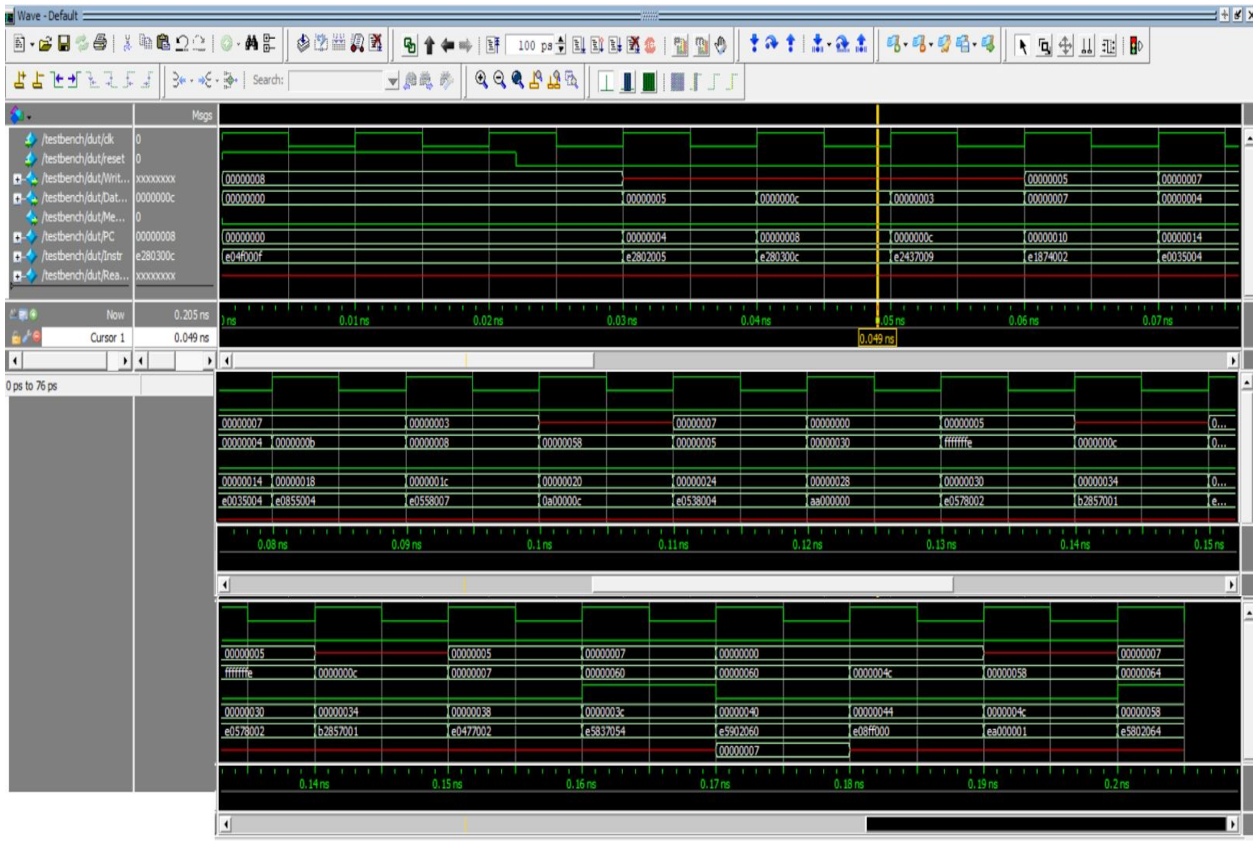
> With register Src2:
- Read from `Rn` and `Rm` (instead of `Imm8`)
- Write *ALUResult* to the register file (Rd)



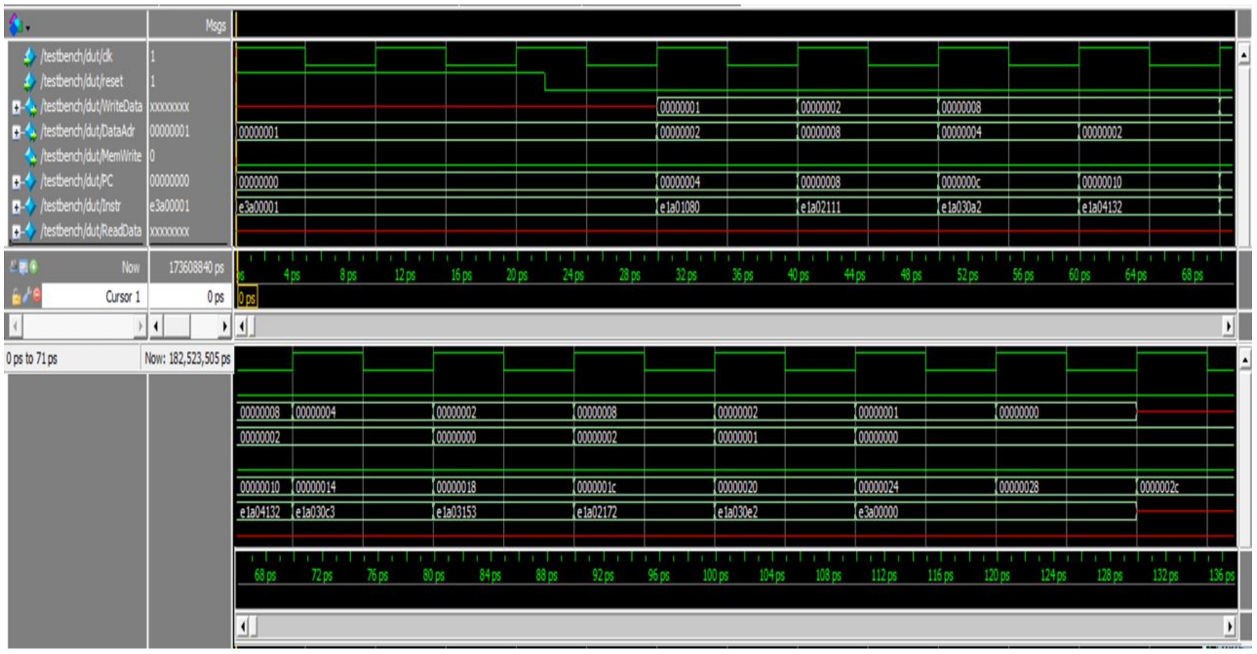
31:28	27:26	25	24:21	20	19:16	15:12	11:7	6:5	4	3:0
cond	op	I	cmd	S	Rn	Rd	shamt5	sh	0	Rm

Procedures

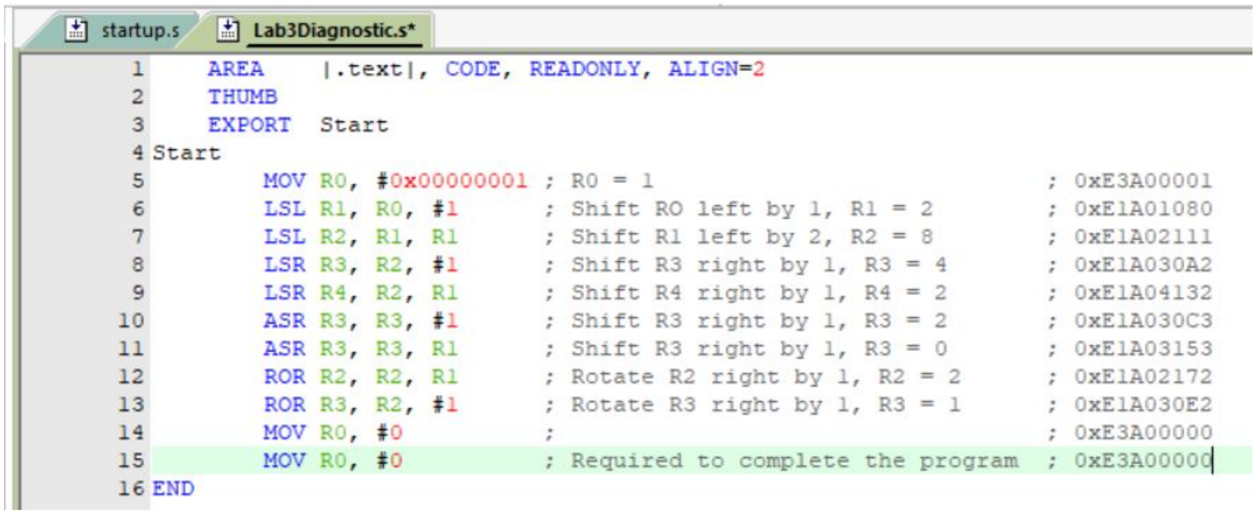
At the prior lab (i.e. lab 3), shifter machine codes generated from the coding of a single-cycle CPU testing method. Then, the diog.dat files provided were used to simulate arm_L4DP-3.sv and analyze the result derived from RTL simulation and waveform analysis. The waveforms posted below demonstrate two cases of simulation which used different machine codes (i.e. memfile.dat and diog.dat).


```
Wave - Default
                         Msgs
/testbench/dut/clk       0
/testbench/dut/reset     0
/testbench/dut/Writ...   xxxxxxxx
/testbench/dut/Dat...    0000000c
/testbench/dut/Me...     0
/testbench/dut/PC        00000008
/testbench/dut/Instr     e280300c
/testbench/dut/Rea...    xxxxxxxx
Now                      0.205 ns
Cursor 1                 0.049 ns
0 ps to 76 ps
```

Original Code Waveform Test Result



Shifter-enhanced Code Waveform Test Result



```
    AREA    |.text|, CODE, READONLY, ALIGN=2
    THUMB
    EXPORT  Start
Start
        MOV R0, #0x00000001 ; R0 = 1                           ; 0xE3A00001
        LSL R1, R0, #1      ; Shift RO left by 1, R1 = 2       ; 0xE1A01080
        LSL R2, R1, R1      ; Shift R1 left by 2, R2 = 8       ; 0xE1A02111
        LSR R3, R2, #1      ; Shift R3 right by 1, R3 = 4      ; 0xE1A030A2
        LSR R4, R2, R1      ; Shift R4 right by 1, R4 = 2      ; 0xE1A04132
        ASR R3, R3, #1      ; Shift R3 right by 1, R3 = 2      ; 0xE1A030C3
        ASR R3, R3, R1      ; Shift R3 right by 1, R3 = 0      ; 0xE1A03153
        ROR R2, R2, R1      ; Rotate R2 right by 1, R2 = 2     ; 0xE1A02172
        ROR R3, R2, #1      ; Rotate R3 right by 1, R3 = 1     ; 0xE1A030E2
        MOV R0, #0          ;                                  ; 0xE3A00000
        MOV R0, #0          ; Required to complete the program ; 0xE3A00000
END
```

Diagnostic Code

Conclusion

The period of data processing completion task for the memfile.dat test case took nearly 0.2 nano seconds with about 23 inputs of machine codes. On the other hand, the waveform simulation conducted with diog.dat with 11 machine code inputs to arm_L4DP-3.sv resulted in 136pico seconds of data processing completion time. These two comparative results show that the processing time taken for a single-cycle does not only depend on the amount of code-line inputs. In other words, result from memfile.dat took average processing time per instruction of 8.69ps while result from diog.dat took nearly 1.236ps per instruction.

Most importantly, all the results represented in “WriteData” objects in two waveforms demonstrate success in regression testing of both an original code and the enhanced shifter testing module.

Bibliography

Data Processing data path:
https://canvas.uw.edu/courses/1386311/files/folder/Lectures?preview=65402923

Arm_single.sv and arm_L4DP.sv code extended by Joseph Decuir