ARCH CS3501 – 2023I
Lab 05 Part 2: Single-cycle Processor Implementation
Prof.: jgonzalez@utec.edu.pe
Src: Harris S., Harris D., Digital Design and Computer Architecture

Introduction

- **To solve this lab: use IcarusVerilog and GTKWave**
- **Use the SingleCycle. zip source file from Canvas. This only works if you solved Lab 05 Part 1.**
- **Separate the Verilog source in files. E.g.: alu.v, datamem.v, etc.**

In this lab you will build a simplified ARM single-cycle processor using Verilog. You will combine your ALU from previous Lab with the code for the rest of the processor taken from the textbook. 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 the new instructions work as well. By the end of this lab, you should thoroughly understand the internal operation of the ARM single-cycle processor.

Please read and follow the instructions in this lab carefully. In the past, many students have lost points for silly errors like not printing all the signals requested.

Before starting this lab, you should be very familiar with the single-cycle implementation of the ARM processor described in Section 7.3 of the Chapter. **Remember,** this version of the ARM single-cycle processor can execute the following instructions: ADD, SUB, AND, ORR, LDR, STR, and B.

Our model of the single-cycle ARM processor divides the machine into two major units: the control and the datapath. Each unit is constructed from various functional blocks. For example, as shown in the figure on the last page of this lab, the datapath contains the 32-bit ALU that you designed in Lab, the register file, the sign extension logic, and five multiplexers to choose appropriate operands.

Exercise 2: Modifying the ARM single-cycle processor

You now need to modify the ARM single-cycle processor by adding the EOR and LDRB instructions. First, modify the ARM processor schematic/ALU at the end of this lab to show what changes are necessary. You can draw your changes directly onto the schematics. Then modify the main decoder and ALU decoder as required. Show your changes in the tables at the end of the lab. Finally, modify the Verilog code as needed to include your modifications. Explain all your changes.

A. Testing your modified ARM single-cycle processor

Next, you'll need a test program to verify that your modified processor works. The program should check that your new instructions work properly and that the old ones didn't break. Use memfile2.asm below.

```
; memfile2.asm
; david_harris@hmc.edu and sarah_harris@hmc.edu 3 April 2014
MAIN    SUB  R0, R15, R15
        ADD  R1, R0, #255
        ADD  R2, R1, R1
        STR  R2, [R0, #196]
        EOR  R3, R1, #77
        AND  R4, R3, #0x1F
        ADD  R5, R3, R4
        LDRB R6, [R5]
        LDRB R7, [R5, #1]
        SUBS R0, R6, R7
        BLT  MAIN
        BGT  HERE
        STR  R1, [R4, #110]
        B    MAIN
HERE    STR  R6, [R4, #110]
```

Figure 1. ARM assembly program: memfile2.asm

Convert the program to machine language and put it in a file named memfile2.dat. Modify imem to load this file. Modify the testbench to check for the appropriate address and data value indicating that the simulation succeeded. Run the program and check your results. Debug if necessary. When you are done, print out the waveforms as before and indicate the address and data value written by the final STR instruction.

Guideline

Create your folders with the names: Exercise 2. Submit your solution as a .tar file via Canvas with: 1) folder with required files (Verilog source, testbench, and vcd waveforms), 2) report in .pdf (outside of the folders).

Deadline: Check in Canvas "Tareas" Section> Lab05.

It is not allowed to use partially or total solutions from online forums or another type of source. **Propose your own solutions.** If not, grade is zero (0) according to UTEC rules.

Please turn in each of the following items in the report, clearly labeled and in the following order:

1. Marked up versions of the datapath schematic and decoder tables that adds the EOR and LDRB instructions. Draw the modified schematic, include all signals.
2. Your Verilog code for your modified ARM computer (including EOR and LDRB functionality) with the changes highlighted and commented in the code.
3. The contents of your memfile2.dat containing your machine language code. Include comments.
4. An image of the simulation waveforms showing correct operation of your modified processor on the new program. What address and data values are written by the final STR instruction? Explain the waveforms.

Table 1. First nineteen cycles of executing armtest.asm (all in hexadecimal, except $Flags_{3:0}$ in binary)ç

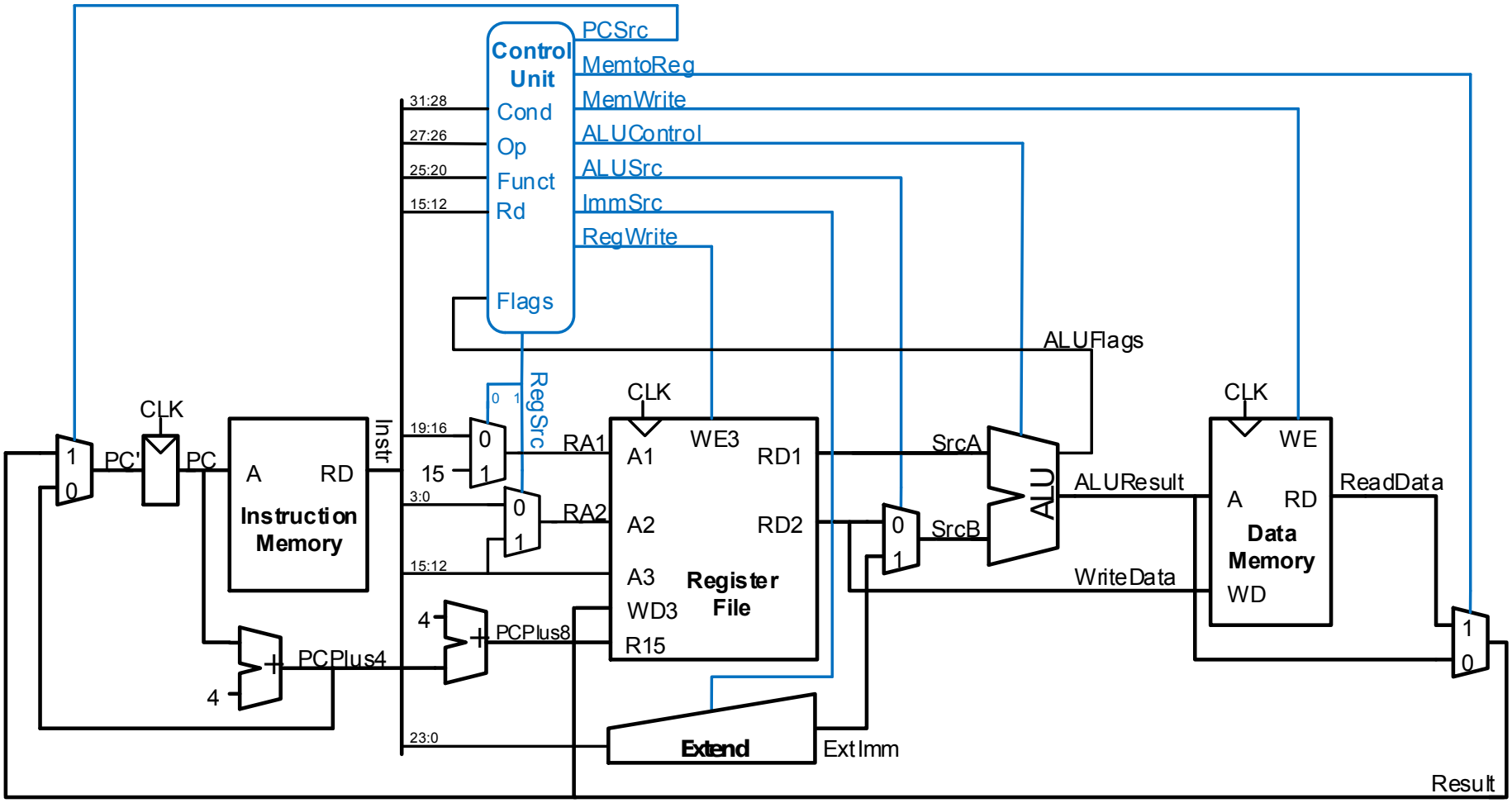

Figure 1. Single-cycle ARM processor

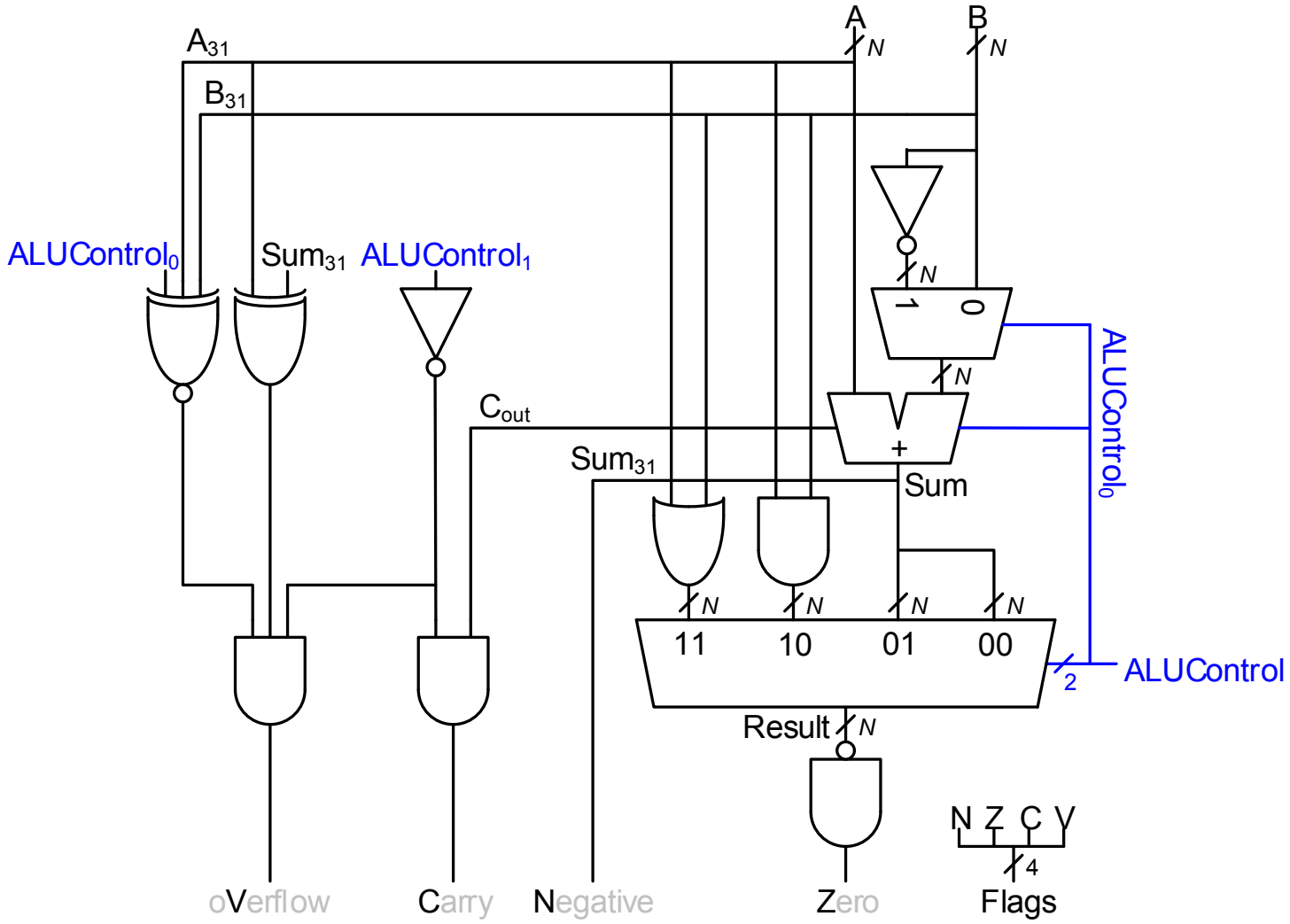

Figure 2. ARM ALU

Table 2. **Extended functionality: Main Decoder**

Op	$Funct_5$	$Funct_0$	Type	Branch	MemtoReg	MemW	ALUSrc	ImmSrc	RegW	RegSrc	ALUOp
00	0	X	DP Reg	0	0	0	0	XX	1	00	1
00	1	X	DP Imm	0	0	0	1	00	1	X0	1
01	X	0	STR	0	X	1	1	01	0	10	0
01	X	1	LDR	0	1	0	1	01	1	X0	0
10	X	X	B	1	0	0	1	10	0	X1	0

Table 3. **Extended functionality: ALU Decoder**

ALUOp	$Funct_{4:1}$ (*cmd*)	$Funct_0$ (*S*)	Notes	$ALUControl_{1:0}$	$FlagW_{1:0}$
0	X	X	Not DP	00	00
1	0100	0	ADD	00	00
		1			11
	0010	0	SUB	01	00
		1			11
	0000	0	AND	10	00
		1			10
	1100	0	ORR	11	00
		1			10