

# [CprE 381] Computer Organization and Assembly-Level Programming, Fall 2018

## Project A Report

Name(s) Ben Kenkel & Bryan Kalkhoff

Section/Lab Time F

*Refer to the highlighted language in the Project A instruction for the context of the following questions.*

- a. [Part 1.a] Provide a description of how each of the control values in the architecture (i.e., the outputs of the main control unit) is used.
- For each output control signal, explain how it is used (i.e., what does it do?)
    - o\_reg\_dest
      - select signal for mux to switch the input of the write select between bits 20-16 and 15-11 of the instruction.
    - o\_jump
      - select signal for mux to switch the input of the pc between the calculated instruction address and the output of a switch that is either pc+4 or pc+4+(imm\*4)
    - o\_branch
      - used in conjunction with the o\_zero from the alu to set a mux for the pc input to switch between pc+4 and pc+4+(imm\*4)
    - o\_mem\_to\_reg
      - select signal for a mux for the write to register, switches between the ALU output and memory read
    - o\_ALU\_op
      - the operation for the alu
    - o\_mem\_write
      - enables writing into memory
    - o\_ALU\_src
      - select signal for a mux for the input for the second input of the alu, switches between read register 2 and sign extended immediate value
    - o\_reg\_write
      - enables writing to registers
  - For the 6 MIPS instructions to support, ADD, ADDI, LW, SW, BEQ and J, complete the following table.

	ADD	ADDI	LW	SW	BEQ	J
o_reg_dest	0	0	0	d	d	D
o_jump	0	0	0	0	0	1
o_branch	0	0	0	0	1	0 (d)
o_mem_to_reg	0	0	1	d	d	d
o_ALU_op	b0000	B0000	B0000	B0000	B0001	D
o_mem_write	0	0	0	1	0	0
o_ALU_src	0	1	1	1	0	D
o_reg_write	1	1	1	0	0	0

D means we don't care about the value

- (continuing from the previous question) For each of the 6 MIPS instructions, explain why you set the values as you described in the table above.
  - ADD
    - We don't want to change the order of instructions
    - Need to add
    - Only write to register
  - ADDI
    - We don't want to change the order of instructions
    - Need to add
    - Only write to register
    - Take in an immediate value by changing ALU\_src
  - LW
    - Add immediate value and write to memory
  - SW
    - Add immediate value and read from memory, writing to the register
  - BEQ
    - Set branch to 1 to allow for branching
    - If  $A = B$ , then  $A - B = 0$ , which sets the alu o\_ZERO to 1
  - J
    - The only thing we care about is  $j = 1$ , everything else is handled regardless of inputs
- b. [Part 1.a] How is the zero flag from the ALU used?  
 If the output of the ALU is zero, this flag indicates as such, which is anded with the branch output to switch the pc increment to the calculated jump address.
- c. [Part 2] Simulate your processor in ModelSim and run the provided program in imem.mif with the data in dmem.mif.
  - The [projectA > ASM Files > test\_with\_data\_seg.asm ] file has the assembly code with data specified which is equivalent to the given imem/dmem.mif. Before simulating the final design, just by studying the code, please explain
    - What the code does
    - Which values will be written on DMEM at the locations of the first 11 words starting from the address 0.
    - Which values will be written on the following registers: \$8 (=\$t0), \$9 (=\$t1), \$10 (=\$t2), and \$16 (=\$s0).
  - Provide a screenshot for each of the 6 MIPS instructions to support showing the correct functionality and explain how the instruction is working in accordance with the screenshot
    - ADDI

/proj/ALU_src	0						
/proj/branch	0						
+ /proj/combined_next_pc	32'd16	(32'd2097152			32'd2359312		32'd13120
+ /proj/i_next_pc	32'd16	32'd4			32'd8		32'd12
+ /proj/instruction	32'h08000004	32'h8C080000			32'h20090004		32'h00008
/proj/jump	1						
/proj/mem_to_reg	0						
/proj/mem_we	0						
+ /proj/o_PC	32'd40	32'd0			32'd4		32'd8
+ /proj/o_pc_plus_4	32'd44	32'd4			32'd8		32'd12
+ /proj/pc_sll2	32'd536870928	32'd807403520			-32'd2145124...		32'd13120
/proj/register_we	0						
+ /proj/rs_data	32'd0	32'd0					
+ /proj/rt_data	32'd0	32'd0			32'd0		32'd0
- /proj/b2v_inst4/registers	{32'd0} {32'd0} {32'd0} {32'd0} {...	{32'd0} {32'd0} {32'd0} {32'd0} ...			{32'd0} {32'd0} {32'd0} {32'd0} ...		{32'd0} {32'd0} {32'd0} {32'd0} ...
+ (31)	32'd0	32'd0					
+ (30)	32'd0	32'd0					
+ (29)	32'd0	32'd0					
+ (28)	32'd0	32'd0					
+ (27)	32'd0	32'd0					
+ (26)	32'd0	32'd0					
+ (25)	32'd0	32'd0					
+ (24)	32'd0	32'd0					
+ (23)	32'd0	32'd0					
+ (22)	32'd0	32'd0					
+ (21)	32'd0	32'd0					
+ (20)	32'd0	32'd0					
+ (19)	32'd0	32'd0					
+ (18)	32'd0	32'd0					
+ (17)	32'd0	32'd0					
+ (16)	32'd0	32'd0					
+ (15)	32'd0	32'd0					
+ (14)	32'd0	32'd0					
+ (13)	32'd0	32'd0					
+ (12)	32'd0	32'd0					
+ (11)	32'd0	32'd0					
+ (10)	32'd0	32'd0					
+ (9)	32'd4	32'd0					32'd4
+ (8)	32'd10	32'd0			32'd10		
+ (7)	32'd0	32'd0					
+ (6)	32'd0	32'd0					
+ (5)	32'd0	32'd0					
+ (4)	32'd0	32'd0					
+ (3)	32'd0	32'd0					
+ (2)	32'd0	32'd0					
+ (1)	32'd0	32'd0					
+ (0)	32'd0	32'd0					

- The instruction is 0x00008020 which is addi. The addi is adding the immediate 4 to register 9 which is seen in the picture.

+ (0)	32'd10	32'd10					
+ /proj/b2v_inst13/i...	-16'd32736	16'd0			16'd4		

- ADD
- + /proj/instruction 32'h8C080000 (32'h8C080000 | 32'h20090004 32'h00008020)

Instruction for add

- LW
- + (0) 32'd10 (32'd10

◆ Taking from address 0 which is 10

(11)	32'd0	32'd0
(10)	32'd0	32'd0
(9)	32'd4	32'd0
(8)	32'd10	32'd0
(7)	32'd0	32'd10

Storing it into register 8

/proja/instruction	32'h20090004	32'h8C080000
--------------------	--------------	--------------

Actual instruction



- SW

32'h...	32'h20090004	32'h00008020	32'h08000009	32'h11000001	32'h08000004	32'h8D2A0000	32'h01508020	32'hAD300000	32'h21290004
---------	--------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------

Instruction for SW R16,0(R9)

(3)	32'd3	32'd3
(2)	32'd3	32'd3
(1)	32'd1	32'd1

Stores into DMEM the result

- Attach a screenshot & explain

- BEQ

/proja/instruction	32'h08000009	32'h8C080000	32'h20090004	32'h00008020	32'h08000009	32'h11000001	32'h08000004
--------------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------

Instruction saying to branch if equal.

/proja/ALU_is_zero	0
--------------------	---

Added signal indicating that the ALU isn't outputting 0 (that they are equal)

- J

/proja/instruction	32'h11000001	32'h8C080000	32'h20090004	32'h00008020	32'h08000009
/proja/jump	1				

The jump signal going high telling the processor to jump

/proja/next_PC	32'd36	32'd4	32'd8	32'd12	32'd36
/proja/instruction	32'h08000009	32'h8C080000	32'h20090004	32'h00008020	32'h08000009

The PC goes from 12 to 36 showing that there was a jump that happened.

(11)	32'd0
(10)	32'd55
(9)	32'd45
(8)	32'd36
(7)	32'd28
(6)	32'd21
(5)	32'd15
(4)	32'd10
(3)	32'd6
(2)	32'd3
(1)	32'd1
(0)	32'd10

Value stored in DMEM

- REGISTERS

+	(20)	32'd0	32'd0		
+	(19)	32'd0	32'd0		
+	(18)	32'd0	32'd0		
+	(17)	32'd0	32'd0		
+	(16)	32'd55	32'd55		
+	(15)	32'd0	32'd0		
+	(14)	32'd0	32'd0		
+	(13)	32'd0	32'd0		
+	(12)	32'd0	32'd0		
+	(11)	32'd0	32'd0		
+	(10)	32'd10	32'd10		
+	(9)	32'd44	32'd44		
+	(8)	32'd0	32'd0		
+	(7)	32'd0	32'd0		
+	(6)	32'd0	32'd0		
+	(5)	32'd0	32'd0		
+	(4)	32'd0	32'd0		
+	(3)	32'd0	32'd0		
+	(2)	32'd0	32'd0		
+	(1)	32'd0	32'd0		
+	(0)	32'd0	32'd0		

o