CprE 381: Computer Organization and Assembly-Level Programming

Project Part 1 Report

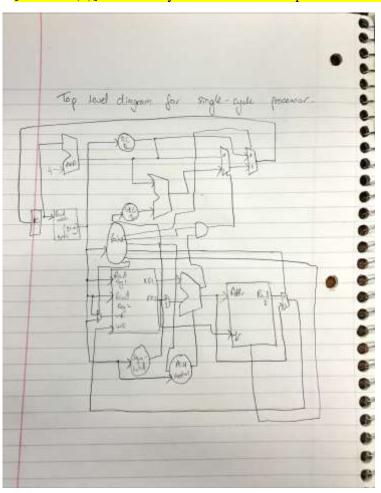
Team Members: Varun Advani

Kalyan Thapaliya

Project Teams Group #: 6

Refer to the highlighted language in the project 1 instruction for the context of the following questions.

[Part 1 (d)] Include your final MIPS processor schematic in your lab report.



[Part 2 (a.i)] Create a spreadsheet detailing the list of *M* instructions to be supported in your project alongside their binary opcodes and funct fields, if applicable. Create a separate column for each binary bit. Inside this spreadsheet, create a new column for the *N* control signals needed by your datapath implementation. The end result should be an *N*M* table where each row corresponds to the output of the control logic module for a given instruction.

whater the	Marie Marie										
1.7000	Charles (Bridge)	Table I (British)	Firmal	of brook	Monteling	Michigan Law Comp.	Analysis (m. mil	Halid	frmt/5bl	Special Section from the reduced on 30 to 50 an extension	Libre
444	"Milder"	100	- 0	- 4	A facility description of the community	A paral steas ACT with the territory.) paid within hard in a region.	Contract to determine agency when the old	Charlest Avenue	Lipschill algo substituti	A fire pers
page 1	"menant"	-		- 8 -	Dipolic and Act and have record	Spanish and APP arms to receipt	James wellen hand to a legisler)	Claditical in Co. decimely expete softening in	Office CD of the Asserts)	Linchtings-retiremed	S (na just
-	1011001	440	- 1		A Jahor WIT seed (free recently)	A literary NOT parks to resemble	Edward hard to a registroit	Town in protestion regions when then off	530e (Callactaver)	Almo principal	Time
med	THEOLOGY	Tour.	-		G Salama NEW saled (from manager)	District MIX sorth on markety.	1 Switze Switze a copposed	E System of an abrellowing register methor them only	District PENALTER Angelos	A (non-extended)	4 Desput
40	"991399"	100	4		Calver NOV least them memory.	District NOV write to mymetral	Service their to a region of	E (lates it as postinates regular rather shan of)	6.bar FEH he branch	Time extended	- The part
200	"001002"	Sant S	- 1		Care NO section remain	2 Steen ROT and a to medium in	Livertranse to consiste.	Elizabet if an alcomoston legislar latter time oil.	State (City to bornic)	Lines extensely	TO COM A PARTY
M	581000				Alter FOT version record	Open SP witch revenue	Definited transmit	Time to be desired that other than its	Alm Witnessen	I like of parts for those on become our sin returned	Almin
in.	'uneque	0	- 1		d place NOV sent (flore intercent)	O Diles NOT as to be removed.	d plant 607 with the a regardly	EDBH OF BUILD	1 Jun Interest of benefits!	I (they not petter has alread one because once upon extremited)	Street, and
him	"MAKE"	~	- 6		Charles and her being	Uplied MV and in removal	A fallow MAT surface to a regarder/	COOK OF PARKS	A Since beautiful photology	A filter set natte and change on business race up nationing	A Deputy
No.	.3000H.	71-71		- 4	12mm from enemy)	Silling NOT yet to contact	3. Section than to program (Elizabeth and destruction register before there is:	Dissification (Venil)	1/ter monthl.	- B (No pure
No.	,3004811.	120			C binne and profession	E/lentes to reamons	3 place 807 webs to a maken	Estimativas material	Stray PC of my drawning) from malescaled	11 ht jum
K.7916											
solvii.	,0000000.	"Yellows"			O. Dolld vilsen, NOT vent/ (Fore-renners)	SQuebl disec NOT works by working!	I Jackine Northwest (to a regulated)	5 (Indicates of as destinating)	Sidner PCH Not Avenual.	3.50ms and stables has allowed day have one own ages what stored	A Dra jum
1000	,000000,	289391			9 Self-may Self man from memory)	E/Jorthusines ACT winte he memory!	Limbbs within high to in ingreners	A Shahili woon tel per aleutomentanii.	DOM FOR RESIDENCE	1. Story and matter but of once one become more age wetersome;	11/19 (40)
irel	,000000.	200300			of place took restrictions resemble	If place her? we for the revenues?	A Switzs built to a regional	A [lase of an electronical]	A facts (Contractions II)	A Steel not marker during some feet date where age verteranes;	10 Test puts
400	1000	3800			(Epiker BOT seal (furti-ferming)	Statement with a replacement	A perhat peri for a repoted	A / (man-of type-despitated)	Elizab PENkins Aspecia	I Date hel habbe had of one one freeze other age arbeinger;	B (he pot
-	,200300,		-0		Statute Will and Dark recently	Statem WT solts to certain	Charling mant to a rygonical	A description of our description of	Staw fit of our disposit.	I liver not martir but disma ere bissons over upo entereland	I be just
-	,000080.	.166701,	4	4.	O Johan ROV west (from nemons)	O Jahram MSV yor'de Co. Frankrykyl	1 (New York State No. of Anguares)	1 Description destinated	School PENERS deserving	3 block not souther but change one because more age extensional	STHAT
100	1000807	781007	4.		If plan 807 and from remoral	Spline MIT enter in evening	T. berits itself to a registre.	T/accord on detectabled	Riber (Thirtmork)	Clime not starter but shows use flavour more age extension;	The feet
-4	THE REAL PROPERTY.	1	-	- 4	O Marc 800" sent (from immung)	SLJainer MCE per by his expression?	1. Declaration to a register.)	1/min of problemation)	(Class Windowski Annes)	I little out marker tool closure one because other age extremiting	7.8 free just
-	7000000	1000001			A Mark HOT HITTER FRANCE	A Johnson New York to American	A James of the symposis.	A faces of problem about	Sharpfining breeze	A plan i stori si prompre l'anni artenitati	A (19 put
-01	000000	.266021,			9 Jahres HOT neet/freet recrook/	2 Marc NOT with 1st draftered	A liwites tient to a regional	. I (more not not described)	Since PS-Arrestment	1. They not metter but chanse and because there also extensions	A Dry Am
0.0	,000000	160007			Callet NO rest from removal	S plays MSV with to remain!	1 jurgest heat to a vegetate?	A lines of an abstraction	0.349 PEH ne brench	. Littory had matter out of your and because they you extensional	this put
white	,500,000,	78880	4		Chicago NOT seat (hare record)	Distance MOT and the community	Chefforday to require	*A Green half the dead health and the	Class Control blooks	Libber sel matter hat drawn you become even upp orbitology	X Daylor
	200	36100	-	-1	Alleman server	A place MCF and a second	Cities est martir C	A place and protect	Option not contract	I like not particular from our businesses care size extensional	Libins
ATTYS:	Temper		-	- 20	7000000	Albert out maked	Total and market	Edition of Artist	A principal and sealers?		Library

[Part 2 (a.ii)] Implement the control logic module using whatever method and coding style you prefer. Create a testbench to test this module individually, and show that your output matches the expected control signals from problem 1(a).

```
library IEEE;
use IEEE.std logic 1164.all;
entity control unit is

port(i_opcode

i_funct
                                                       : in std_logic_vector(5 downto 0);
: in std_logic_vector(5 downto 0);
: out std_logic_vector(14 downto 0));
                        1_funct
o_Ctrl_Unt
end control unit;
architecture dataflow of control unit is
signal s_RTYPE : std_logic_vector(14 downto 0);
begin
      h i_funct select s RTYPE =
"050111000110100" when "100000",
"0500000000110100" when "100000",
"050001000110100" when "100111",
"05001100110100" when "100111",
"050001100110100" when "101011",
"050011100110100" when "101011",
"050010100110100" when "0500000",
"050010000110000" when "050010",
"050010000110100" when "050010",
"0500101000110100" when "050010",
"050000000000000000" when "050100",
"050000000000000000" when "050100",
"0500000000000000000" when others;
with i funct select s RTYPE <=
with i_opcode select o_Ctrl_Unt <=
       s RTYPE
        s RTYPE when "000000",
"001111000100100" when "001000",
        "9898888888888888888888" when "8181888",
        "001000000100100" when "901001",
        "001001000100000" when "001100",
       "80101000100000" when "80110",
"801001000100000" when "801101",
       "018080080100110" when "000011",
       "0000000000000000 when others;
```

end dataflow;

≨	Msgs										
/tb_control_unit/cCLK_PER	100 ns	100 ns									
 → /tb_control_unit/expected	15'	0E34	0034	0234	0534	0434	0334	0734	0934	0830	0A34
/tb_control_unit/gCLK_HP	50 ns	50 ns									
/tb_control_unit/s_CLK	0										
<u>★</u> ·◆ /tb_control_unit/s_Ctrl_Unt	15'	0E34	0034	0234	0534	0434	0334	0734	0934	0830	0A34
	6'h20	20	21	24	27	26	25	2A	00	02	03
<u>★</u>	6'h00	00									
control_unit											
— input ———											
 → opcode	6'b	000000									
⊕	6'b	100000	100001	100100	100111	100110	100101	101010	000000	000010	0000
— output ———											
O_Ctrl_Unt O_Ctrl_Unt	15'	0E34	0034	0234	0534	0434	0334	0734	0934	0830	0A34
_ → ◆ expected_out	15'	0E34	0034	0234	0534	0434	0334	0734	0934	0830	0A34
△ 🛒 🏵 Now	250 ns	ns		liiiiiiiiii Ons	20	liiiiiiiii Ons		1:::::::: 0 ns		liiiiiiiii Ons	1111111111
	0 ns	0 ns									
√	Q D										-

[Part 2 (b.i)] What are the control flow possibilities that your instruction fetch logic must support? Describe these possibilities as a function of the different control flow-related instructions you are required to implement.

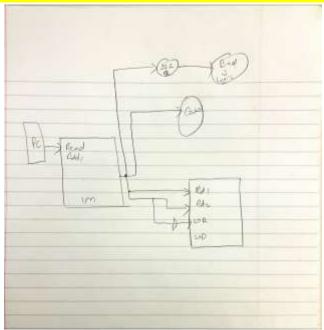
The given instructions that need to be simulate would all have unique control unit values depending on the path. In the I-type instructions, the ALUSrc receives an immediate value instead of a value read from the register.

There are two control signals that control the addressing of the Jump and branch instructions. The branch instruction needs a ALU output The jump register can receive an address from a register or a 26 bit immediate value. The control signal for the Jump and Link writes the address +4 that is computed the ALU.

There are two control signal for the write enable, and two control signals that control the write back data of the rs and rd registers from the data memory or the ALU.

The sign extend control signal determines whether the immediate value would be zero or sign extended.

[Part 2 (b.ii)] Draw a schematic for the instruction fetch logic and any other datapath modifications needed for control flow instructions. What additional control signals are



needed?

Control Signal is roughly 15 bits for the fetch instruction which includes, jr, jal, ALUSrc, MemtoReg, MemWrite, RegWrite, RegDst, Branch, SignExt, j, and the 4 bit ALU opcode.

[Part 2 (b.iii)] Implement your new instruction fetch logic using VHDL. Use Modelsim to test your design thoroughly to make sure it is working as expected. Describe how the execution of the control flow possibilities corresponds to the Modelsim waveforms in your writeup.

I've tested out each instruction to see if it fetches correctly, did not have time to make a testbench for it.

[Part 2 (c.i.1)] Describe the difference between logical (srl) and arithmetic (sra) shifts. Why does MIPS not have a sla instruction?

Srl is a logical shift right which means that it shifts the bits from left to rights, and the bits added would be zero, irrespective of the most significant bit. An arithmetic shift would account for the Most Significant bit and hence is useful for shifting negative numbers whilst holding its value.

SLA shifts from right to left however, the Least Significant bit won't impact the sign and hence an arithmetic instruction for shift left is not required.

[Part 2 (c.i.2)] In your writeup, briefly describe how your VHDL code implements both the arithmetic and logical shifting operations.

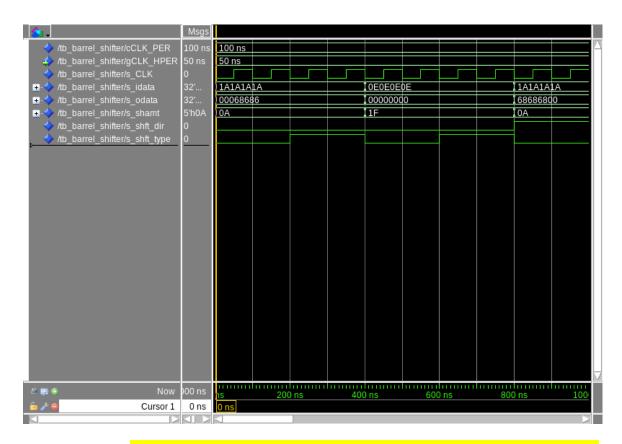
We added a 0 or the binary value of the MSB, with the help of a 2 to 1 MUX that is controlled by a control signal of the control unit.

[Part 2 (c.i.3)] In your writeup, explain how the right barrel shifter above can be enhanced to also support left shifting operations.

The left shifting would make the MSB the LSB, bit 30 bit 1 up until the shift amount. Therefore a 2 to 1 MUX canbe used to either re-shift the left shifted output, or use the right shift output. A second MUX may also be required, select between the right shift input and the left shift input. Control bit would be 0.

[Part 2 (c.i.4)] Describe how the execution of the different shifting operations corresponds to the Modelsim waveforms in your writeup.

Type gives you the type of shift, 0 is logical and 1 is arithmetic. Dir gives direction, 0 is right and 1 is left, and shamt gives shift amount and data is the value being shifted.



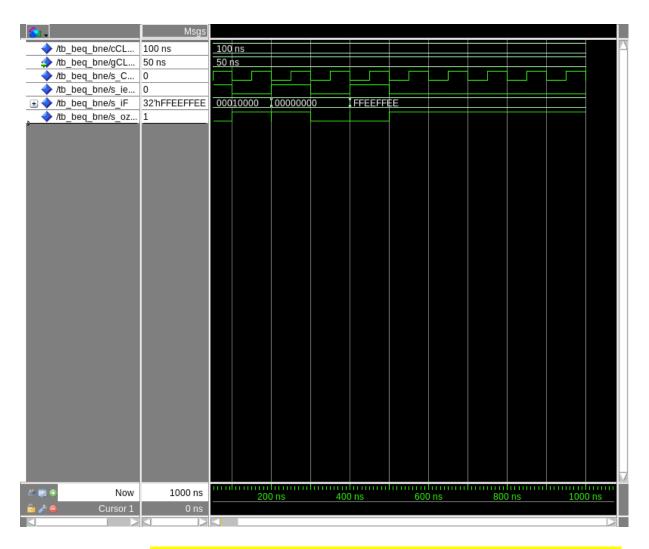
[Part 2 (c.ii.1)] In your writeup, briefly describe your design approach, including any resources you used to choose or implement the design. Include at least one design decision you had to make.

The way we thought of doing the ALU was to figure out how we got implement similar types of instructions into smaller blocks within the ALU. This would make it easier for testing and improve reusability to try and make the ALU as efficient as possible. Some of the instructions like the barrel shifter for the shifts were a little complicated.

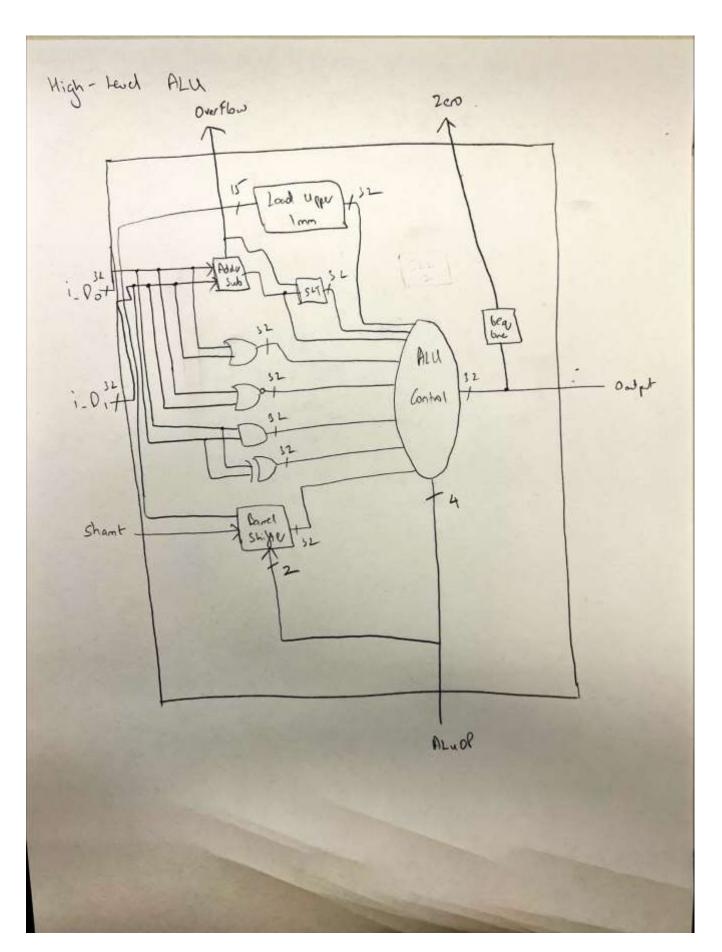
[Part 2 (c.ii.2)] Describe how the execution of the different operations corresponds to the Modelsim waveforms

This question is repeated I think but I put in one of the components anyway

The execution of the different operations will be shown and annotated in the testbench simulations below. The other important components of the ALU that have been tested is the component for the **BEQ** and **BNE** logic the waveform for which as attached below. is



[Part 2 (c.iii)] Draw a simplified, high-level schematic for the 32-bit ALU. Consider the following questions: how is Overflow calculated? How is Zero calculated? How is slt implemented?



Slt logic was the computed by subtracting the D1-D0 with the sign of the output being

considered. The logic for the overflow was a simple AND gate. Overflow was computed inside the adder/subtracter. The zero was computed through the BEQ and BNE functional block which takes the ALU control output as its input, to compute the zero.

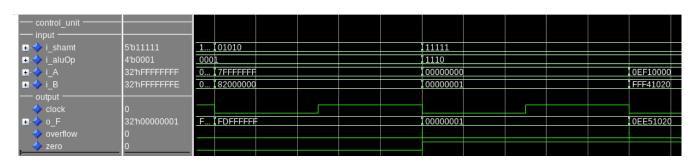
[Part 2 (c.v)] Describe how the execution of the different operations corresponds to the Modelsim waveforms in your writeup.

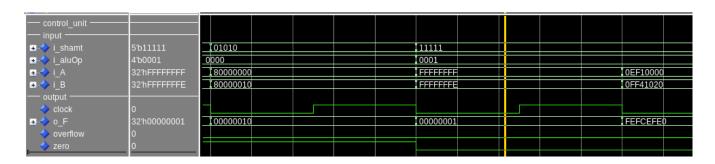
We ran a testbench with all the possible arithmetic instructions supported by the ALU, as well as edge cases to make sure the ALU works as expected. The testbench and the do file is in the test folder and attached below is a screenshot of the waveforms:

Tests conducted were:

- 1. Test for beq
- 2. Test for bne
- 3. Test for addu
- 4. Test for subu
- 5. Test for add
- 6. Test for sub
- 7. Test for and
- 8. Test for or
- 9. Test for xor
- 10. Test for nor
- 11. Test for lui
- 12. Test for slt
- 13. Test for sll
- 14. Test for srl
- 15. Test for sra

Could not attached screenshots for all but here is some of them.





control_unit								
	5'b11111	01010		11111				
	4'b1010	1100		0000				
⊕ ♦ i_A	32'hFFFFFFF	00000000					0EF10000	
⊕ ♦ i_B	32'h80000000	00000000		00000001			FFF41020	
— output —	_							
→ clock	0							
⊕ ♦ o_F	32'hFFFFFFF	00000000		00000001			0EE51020	
overflow	0							
→ zero	1							
	1							

≙	Msgs						
control_unit							
— input ———							
	5'b11111	11111					01010
⊕ 🔷 i_aluOp	4'b1010	1011					
	32'hFFFFFFF	(EEFFFFF		00001000			00000000
	32'h80000000	(EEFFFFFF		00000100			00000000
— output ———							
clock	0						
⊕ ♦ o_F	32'hFFFFFFF	00000000		00001100			00000000
overflow	0						
zero	1						

control_unit							
— input —							
<u>+</u> ♦ i_shamt	5'b11111	11111					01010
 → i_aluOp	4'b1110	1111					
⊕ ♦ i_A	32'h0EF10000	FFFFFFF		0EF10000			7FFFFFFF
 → i_B	32'hFFF41020	FFFFFFE		0FF41020			82000000
— output —							
clock	1						
 ♦ o_F	32'h0EE51020	00000001		FEFCEFE()		FDFFFFF
overflow	0						
zero	1						

[Part 3] In your writeup, show the Modelsim output for each of the following tests, and provide a discussion of result correctness. It may be helpful to also annotate the waveforms directly.

No time to annotate waveforms but all tests ran on the testing framework with no issues.

[Part 3 (a)] Create a test application that makes use of every required arithmetic/logical instruction at least once. The application need not perform any particularly useful task, but it should demonstrate the full functionality of the processor (e.g., sequences of many instructions executed sequentially, 1 per cycle while data written into registers can be effectively retrieved and used by later instructions). Name this file Proj1_base_test.s.

[Part 3 (b)] Create and test an application which uses each of the required control-flow instructions and has a call depth of at least 5 (i.e., the number of activation records on the stack is at least 4). Name this file Proj1_cf_test.s.

[Part 3 (c)] Create and test an application that sorts an array with *N* elements using the BubbleSort algorithm (link). Name this file Proj1_bubblesort.s.

[Part 4] report the maximum frequency your processor can run at and determine what your critical path is. Draw this critical path on top of your top-level schematics. What components would you focus on to improve the frequency?

Fmax is 26.3 mhz with a slack of -17.67 ns

Critical Path seems to be the branch instruction from the ALU, without going through the Data Memory, which comes to approximately 43 ns

I think the ALU itself could definitely be optimized because the ALU must make a redundant comparison to 0 based on the output of ALU output signal, which makes the branch instruction's path the longest one in the processor. I feel like we could've come up with a better logic flow by assigning a sign bit in the control unit and better usage of logic gates to improve our processor, but we just ran out of time!

