Carleton University

Department of Systems and Computer Engineering

SYSC 3006 (Computer Organization) summer 2020

Lab / Assignment 3– Answers file

Student Name: Solution file ID#:

Part 1 – [2.4-mark/5]

1-1

a) [0.50-mark] Fill in the Part 1 Instruction Encoding Table for the following instructions.

OPR	Instruction	Encoding (hex)
NOT	R5 ← NOT R4	**0x07504000**
SUB	R7 ← R6 – R5	**0x02765000**
NOP	NOP	**0x00000000**
MOV	R0 ← R7	**0x03007000**

b) [0.50-mark] Complete the FSM Output ROM for part 1.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
State Hex encoding	Unused (0)	IRCE	PCOE	C1OE	AADD	MARCE	MAROE	MDRCE	MDROE	MDRget	MDRput	IBRead	IBWrite	AOP	ANOP	DR	SXR	SYR	RegSEL	RegLD	T1CE	T1OE	T2CE	T2OE	Q7+	Q6+	Q5+	Q4+	Q3+	Q2+	Q1+	Q0+	Hex Encoding
F0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0002 0001
F1 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002 0002
F2 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0002 0003
Decode 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0002 0007
E0 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	1	0	1	0002 B805
E1 5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	1	1	0	0004 7606
E2 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0003 2100
Dead 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0000 0007

c) [0.50-mark] Complete the FSM Decode ROM Tables (this table is same for Part 1 and 2 of this lab).

Instruction	Address (hex)	Contents (hex)
NOP	00	**00**
ADD	01	**04**
SUB	02	**04**
MOV	03	**05**
AND	04	**04**
OR	05	**04**
XOR	06	**04**
NOT	07	**05**

NOTE: In the supplied circuit, the Control FSM outputs the RegSEL and RegLD signals with the behaviour expected by the Registers RAM. The FSM Output ROM does not use RegR and RegW signals as done in Lab-II and in class; the RegSEL and RegLD signals (with Logisim RAM signaling behaviour) are used instead.

Do not change any of the values that have been pre-entered into the table, except for the empty cases in the hexadecimal encodings (in Part 1 FSM Output ROM Table). The first 3 states are there only as placeholders in Part 1 and have been designed to have no undesirable effects on the circuit. This allows you to always start the FSM in state 0.

The resulting FSM Output ROM and FSM Decode ROM values should work for any of the instructions discussed in class that use the ALU (i.e. NOP, ADD, SUB, MOV, AND, OR, XOR and NOT).

d) [0.50-mark] Save your circuit as Lab-3_Part1.circ. and submit it with your assignment for verification and to get the marks for this section. Your circuit will get checked and tested.

e) [0.40-mark] Same as you did in lab 2, execute the instructions in the table above in the given sequence with all registers initially containing 0x0. Log the execution of the sequence on your implementation to validate the execution of the required instructions and show the results here. (The simulation log should include: Current State, IR, PC, registers, and Next State. Set the log radix to hex).

Current State	next state	Instruction Register	RAM(1060,270)[0]	RAM(1060,270)[5]	RAM(1060,270)[6]	RAM(1060,270)[7]	RAM(1060,270)[15]
00	01	07504000	00000000	00000000	00000000	00000000	00000000
01	02	07504000	00000000	00000000	00000000	00000000	00000000
02	03	07504000	00000000	00000000	00000000	00000000	00000000
03	07	07504000	00000000	00000000	00000000	00000000	00000000
05	06	07504000	00000000	00000000	00000000	00000000	00000000
06	00	07504000	00000000	00000000	00000000	00000000	00000000
06	00	07504000	00000000	ffffffff	00000000	00000000	00000000
00	01	02765000	00000000	ffffffff	00000000	00000000	00000000
01	02	02765000	00000000	ffffffff	00000000	00000000	00000000
02	03	02765000	00000000	ffffffff	00000000	00000000	00000000
03	07	02765000	00000000	ffffffff	00000000	00000000	00000000
04	05	02765000	00000000	ffffffff	00000000	00000000	00000000
05	06	02765000	00000000	ffffffff	00000000	00000000	00000000
06	00	02765000	00000000	ffffffff	00000000	00000000	00000000
06	00	02765000	00000000	ffffffff	00000000	00000001	00000000
00	01	00000000	00000000	ffffffff	00000000	00000001	00000000
01	02	00000000	00000000	ffffffff	00000000	00000001	00000000
02	03	00000000	00000000	ffffffff	00000000	00000001	00000000
03	07	00000000	00000000	ffffffff	00000000	00000001	00000000
00	01	03007000	00000000	ffffffff	00000000	00000001	00000000
01	02	03007000	00000000	ffffffff	00000000	00000001	00000000
02	03	03007000	00000000	ffffffff	00000000	00000001	00000000
03	07	03007000	00000000	ffffffff	00000000	00000001	00000000
05	06	03007000	00000000	ffffffff	00000000	00000001	00000000
06	00	03007000	00000000	ffffffff	00000000	00000001	00000000
06	00	03007000	00000001	ffffffff	00000000	00000001	00000000

Part 2 – [total of 2.6-mark/5]

2.1 - Tables

a) [0.50-mark] Complete the Part 2 FSM Output ROM Table.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
State Hex encoding	**Unused (0)**	**IRCE**	**PCOE**	**C1OE**	**AADD**	**MARCE**	**MAROE**	**MDRCE**	**MDROE**	**MDRget**	**MDRput**	**IBRead**	**IBWrite**	**AOP**	**ANOP**	**DR**	**SXR**	**SYR**	**RegSEL**	**RegLD**	**T1CE**	**T1OE**	**T2CE**	**T2OE**	**Q7+**	**Q6+**	**Q5+**	**Q4+**	**Q3+**	**Q2+**	**Q1+**	**Q0+**	**Hex Encoding**
F0 0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	24023801
F1 1	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	1B100602
F2 2	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	40C20003
Decode 3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	1	20022107
E0 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	1	0	1	0002B805
E1 5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	1	1	0	00047606
E2 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	00032100
Dead 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0000 0007

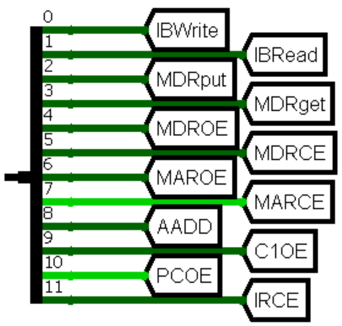
b) [0.40-mark]Complete the Part 2 Main Memory Instructions Table for the following instructions.

Instruction	Address (hex)	Contents (hex)
R10 ← R1 OR R2	00	0x05A12000
R11 ← R2 – R10	01	0x02B2A000
R12 ← NOT (R11)	02	0x07C0B000
R13 ← R0 + R12	03	0x01D0C000

2.2 - Circuit wiring

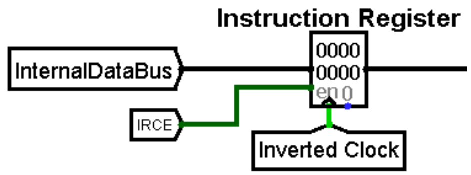
Save a copy of your part 1 (Save as) and name it "Lab3-Part-2.circ". Then extend your Processor solution to Part 1 to read and execute instructions from Main Memory. You will not need to modify the circuit beyond the description of modifications given above.

a) [0.30-mark] Show below a screenshot of the new control FMS outputs that you added in the Logisim circuits, and a short description about each output.



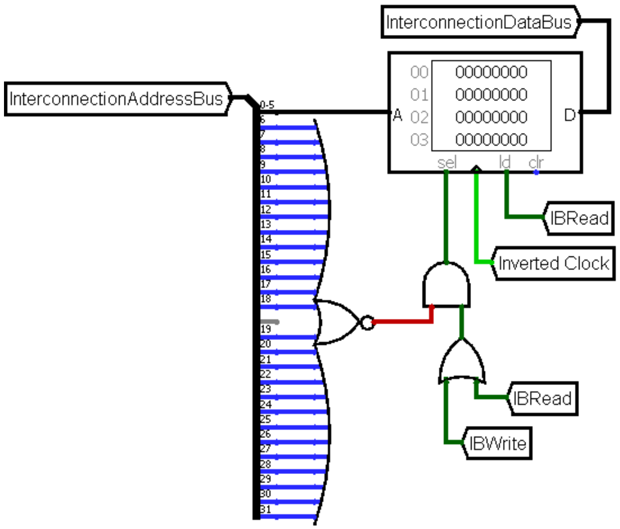
Any short comprehensive description of the outputs is acceptable.

b) [0.40-mark] Show here a screenshot of the new hardware components that you added in the Logisim circuits. Include a short description about each component function.

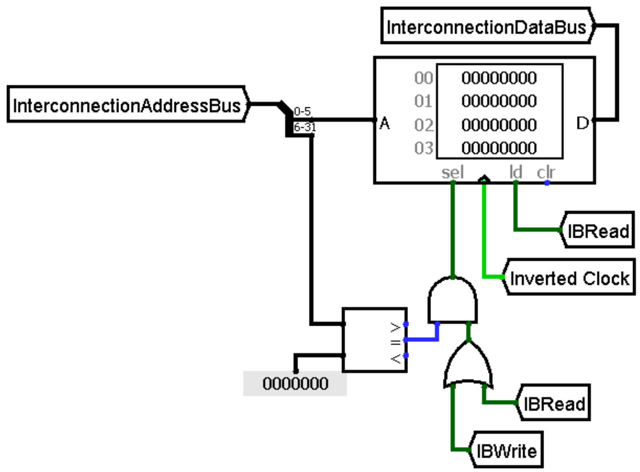


Connecting IR to the internal data bus and adding IRCE control and Inverted Clock to allow fetching the Instruction from the internal data into IR at the falling edge of the clock.

MMemory circuit:

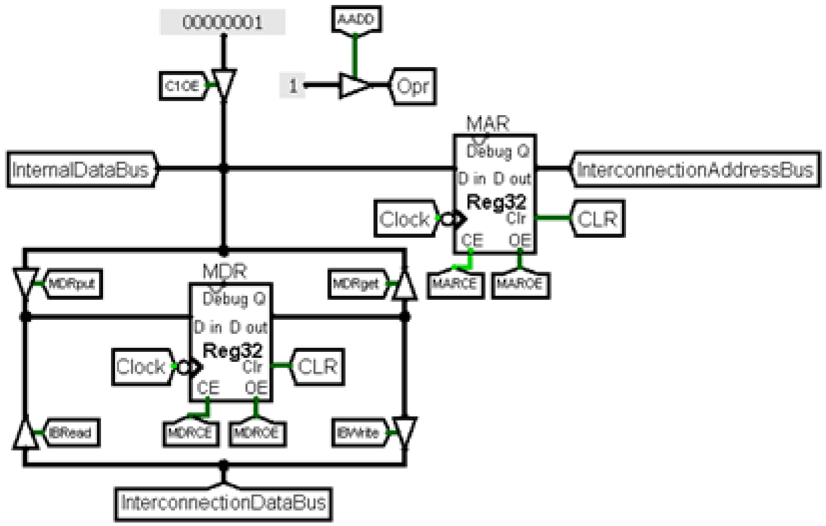


or

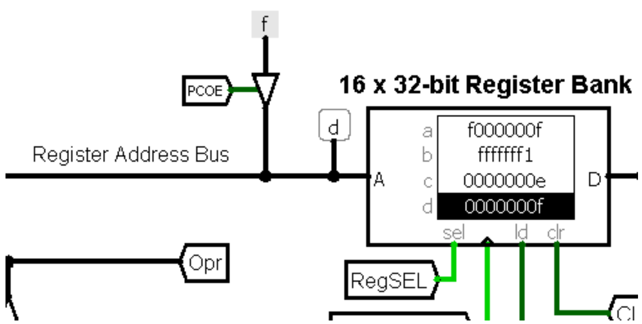


(Also any equivalent address decoding hardware that works is acceptable).

Any short comprehensive description of the circuit is acceptable most talk about address decoding (0x00000000 to 0x0000003F address range, 32x32-bit words) and decoding hardware (in the images above the 26-input NOR (or the comparator in the second image) that generate a 1 output only when all 26 inputs are set to 0).



Short description like : AADD and C1OE to increment PC+1 those hardware and controls are required as they are not decoded into the instruction,……Talk about MDR and MAR….



PCOE and constant F are added to the Register Address Bus to allow addressing PC register.

c) [0.30-mark] Include a copy of your Lab3-Part-2.circ circuit with your submission. We must verify your circuit functionality in order to assign you marks for this part (c) Your circuit will get cheeked and tested

2.3 Execution test

a) [0.40-mark] The Part 2 Main Memory Instructions Table in 2.1 –b) contain same instruction sequence from your lab 2-part 2, but here they are encoded over 32-bit word width. Clear the 16 internal register bloc and the Main memory to 0x0. Then as you did in lab 2, initiate R0 to 0x00000001, R1 to 0x1000000F and R2 to 0xF0000000. Then insert the instructions from the Table in 2.1 –b) above into the Main Memory starting at address 0 and up (one instruction per address word as indicated in the table) in the given sequence. Now, execute all the instructions (repeat fetch-decode-execute cycle till all instruction are fully executed). To execute all instruction just poke the Toggle Switch to advance the FSM through the operation till all instructions are fully executed (while observing their execution). Make sure you PC (R15), holds the address of the first instruction to be fetched, then observe it increments… to fetch the next... Same as you did in part 1, log the execution of the sequence on your implementation to validate the execution of the required instructions and show the results here.

Current State	Next State	Instruction Register	RAM(1060,270)[0]	RAM(1060,270)[1]	RAM(1060,270)[2]	RAM(1060,270)[10]	RAM(1060,270)[11]	RAM(1060,270)[12]	RAM(1060,270)[13]	RAM(1060,270)[15]
00	01	00000000	00000001	1000000f	f0000000	00000000	00000000	00000000	00000000	00000000
01	02	00000000	00000001	1000000f	f0000000	00000000	00000000	00000000	00000000	00000000
02	03	00000000	00000001	1000000f	f0000000	00000000	00000000	00000000	00000000	00000000
02	03	05a12000	00000001	1000000f	f0000000	00000000	00000000	00000000	00000000	00000000
03	07	05a12000	00000001	1000000f	f0000000	00000000	00000000	00000000	00000000	00000000
03	07	05a12000	00000001	1000000f	f0000000	00000000	00000000	00000000	00000000	00000001
04	05	05a12000	00000001	1000000f	f0000000	00000000	00000000	00000000	00000000	00000001
05	06	05a12000	00000001	1000000f	f0000000	00000000	00000000	00000000	00000000	00000001
06	00	05a12000	00000001	1000000f	f0000000	00000000	00000000	00000000	00000000	00000001
06	00	05a12000	00000001	1000000f	f0000000	f000000f	00000000	00000000	00000000	00000001
00	01	05a12000	00000001	1000000f	f0000000	f000000f	00000000	00000000	00000000	00000001
01	02	05a12000	00000001	1000000f	f0000000	f000000f	00000000	00000000	00000000	00000001
02	03	05a12000	00000001	1000000f	f0000000	f000000f	00000000	00000000	00000000	00000001
02	03	02b2a000	00000001	1000000f	f0000000	f000000f	00000000	00000000	00000000	00000001
03	07	02b2a000	00000001	1000000f	f0000000	f000000f	00000000	00000000	00000000	00000001
03	07	02b2a000	00000001	1000000f	f0000000	f000000f	00000000	00000000	00000000	00000002
04	05	02b2a000	00000001	1000000f	f0000000	f000000f	00000000	00000000	00000000	00000002
05	06	02b2a000	00000001	1000000f	f0000000	f000000f	00000000	00000000	00000000	00000002
06	00	02b2a000	00000001	1000000f	f0000000	f000000f	00000000	00000000	00000000	00000002
06	00	02b2a000	00000001	1000000f	f0000000	f000000f	fffffff1	00000000	00000000	00000002
00	01	02b2a000	00000001	1000000f	f0000000	f000000f	fffffff1	00000000	00000000	00000002
01	02	02b2a000	00000001	1000000f	f0000000	f000000f	fffffff1	00000000	00000000	00000002
02	03	02b2a000	00000001	1000000f	f0000000	f000000f	fffffff1	00000000	00000000	00000002
02	03	07c0b000	00000001	1000000f	f0000000	f000000f	fffffff1	00000000	00000000	00000002
03	07	07c0b000	00000001	1000000f	f0000000	f000000f	fffffff1	00000000	00000000	00000002
03	07	07c0b000	00000001	1000000f	f0000000	f000000f	fffffff1	00000000	00000000	00000003
05	06	07c0b000	00000001	1000000f	f0000000	f000000f	fffffff1	00000000	00000000	00000003
06	00	07c0b000	00000001	1000000f	f0000000	f000000f	fffffff1	00000000	00000000	00000003
06	00	07c0b000	00000001	1000000f	f0000000	f000000f	fffffff1	0000000e	00000000	00000003
00	01	07c0b000	00000001	1000000f	f0000000	f000000f	fffffff1	0000000e	00000000	00000003
01	02	07c0b000	00000001	1000000f	f0000000	f000000f	fffffff1	0000000e	00000000	00000003
02	03	07c0b000	00000001	1000000f	f0000000	f000000f	fffffff1	0000000e	00000000	00000003
02	03	01d0c000	00000001	1000000f	f0000000	f000000f	fffffff1	0000000e	00000000	00000003
03	07	01d0c000	00000001	1000000f	f0000000	f000000f	fffffff1	0000000e	00000000	00000003
03	07	01d0c000	00000001	1000000f	f0000000	f000000f	fffffff1	0000000e	00000000	00000004
04	05	01d0c000	00000001	1000000f	f0000000	f000000f	fffffff1	0000000e	00000000	00000004
05	06	01d0c000	00000001	1000000f	f0000000	f000000f	fffffff1	0000000e	00000000	00000004
06	00	01d0c000	00000001	1000000f	f0000000	f000000f	fffffff1	0000000e	00000000	00000004
06	00	01d0c000	00000001	1000000f	f0000000	f000000f	fffffff1	0000000e	0000000f	00000004

b) [0.30-mark] Compare the concept used in this part to your lab 2-part 2, briefly describe what is the advantage of the concept here over lab2-part 2?
In this lab 2-part 2 we needed to load every single instruction manually into IR, in this part all instructions are fetched into IR automatically and executed one after another. Both give same execution results but the concept here is automated and practical to execute programs (set of instructions).

Submission deadline

Must be submitter on cuLearn, locate (Assignment 3 submission) and follow instructions. Submission exact deadline (date and time) is displayed clearly within the Assignment 3 submission on cuLearn.

Note: If you have any question please contact your respective group TA (see TA / group information posted on cuLearn) or use Discord class server.

Good Luck