Toronto Metropolitan University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	Computer Organization and Architecture
Course Number:	COE 608
Semester/Year (e.g. F2017)	W2023

Instructor	Demetres Kostas

Assignment/Lab Number:	5
Assignment/Lab Title:	Control Unit

Submission Date:	March 29, 2023
Due Date:	March 29, 2023

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Hernandez	Spencer	501033228	11	Spencer H
Dumlao	Carlo	501018239	11	C.D

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

1.0 VHDL Code

```
1. library ieee;
2. use ieee.std_logic_1164.ALL;
3.
4. ENTITY Control IS
5.       PORT(
6.                   clk, mclk : IN STD_LOGIC;
7.                   enable : IN STD_LOGIC;
8.                   statusC, statusZ : IN STD_LOGIC;
9.                   INST : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
10.                  A_Mux, B_Mux : OUT STD_LOGIC;
11.                  IM_MUX1, REG_Mux : OUT STD_LOGIC;
12.                  IM_MUX2, DATA_Mux : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
13.                  ALU_op : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
14.                  inc_PC, ld_PC : OUT STD_LOGIC;
15.                  clr_IR : OUT STD_LOGIC;
16.                  ld_IR : OUT STD_LOGIC;
17.                  clr_A, clr_B, clr_C, clr_Z : OUT STD_LOGIC;
18.                  ld_A, ld_B, ld_C, ld_Z : OUT STD_LOGIC;
19.                  T : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
20.                  wen, en : OUT STD_LOGIC
21.            );
22.END Control;
23.
24.ARCHITECTURE description OF Control IS
25.      TYPE STATETYPE IS (state_0, state_1, state_2);
26.      SIGNAL present_state : STATETYPE;
27.      SIGNAL Instruction_sig : STD_LOGIC_VECTOR(3 DOWNTO 0);
28.      SIGNAL Instruction_sig2 : STD_LOGIC_VECTOR(7 DOWNTO 0);
29.BEGIN
30.      Instruction_sig <= INST(31 DOWNTO 28);
31.      Instruction_sig2 <= INST(31 DOWNTO 24);
32.
33.      ----------OPERATION DECODER-------------
34.      PROCESS (present_state, INST, statusC, statusZ, enable,
   instruction_sig, instruction_sig2)
35.      BEGIN
36.            if enable = '1' then
37.                  if present_state = state_0 then
38.                        DATA_Mux <= "00"; --Fetch Address of next
   instruction
39.                        clr_IR <= '0';
40.                        ld_IR <= '1';
41.                        ld_PC <= '0';
42.                        inc_PC <= '0';
43.                        clr_A <= '0';
44.                        ld_A <= '0';
45.                        ld_B <= '0';
46.                        clr_B <= '0';
47.                        clr_C <= '0';
48.                        ld_C <= '0';
```

```
49.                         clr_Z <= '0';
50.                         ld_Z <= '0';
51.                         en <= '0';
52.                         wen <= '0';
53.
54.                   elsif present_state = state_1 then
55.                         clr_IR <= '0'; --Increment PC Counter
56.                         ld_IR <= '0';
57.                         ld_PC <= '1';
58.                         inc_PC <= '1';
59.                         clr_A <= '0';
60.                         ld_A <= '0';
61.                         ld_B <= '0';
62.                         clr_B <= '0';
63.                         clr_C <= '0';
64.                         ld_C <= '0';
65.                         clr_Z <= '0';
66.                         ld_Z <= '0';
67.                         en <= '0';
68.                         wen <= '0';
69.
70.                         if Instruction_sig = "0010" then --STA
71.                               clr_IR <= '0';
72.                               ld_IR <= '0';
73.                               ld_PC <= '1';
74.                               inc_PC <= '1';
75.                               clr_A <= '0';
76.                               ld_A <= '0';
77.                               ld_B <= '0';
78.                               clr_B <= '0';
79.                               clr_C <= '0';
80.                               ld_C <= '0';
81.                               clr_Z <= '0';
82.                               ld_Z <= '0';
83.                               REG_Mux <= '0';
84.                               DATA_Mux <= "00";
85.                               en <= '1';
86.                               wen <= '1';
87.
88.                         elsif Instruction_sig = "0011" then --STB
89.                               clr_IR <= '0';
90.                               ld_IR <= '0';
91.                               ld_PC <= '1';
92.                               inc_PC <= '1';
93.                               clr_A <= '0';
94.                               ld_A <= '0';
95.                               ld_B <= '0';
96.                               clr_B <= '0';
97.                               clr_C <= '0';
98.                               ld_C <= '0';
99.                               clr_Z <= '0';
100.                              ld_Z <= '0';
```

```
101.                                REG_Mux <= '1';
102.                                DATA_Mux <= "00";
103.                                en <= '1';
104.                                wen <= '1';
105.
106.                          elsif Instruction_sig = "1001" then --LDA
107.                                clr_IR <= '0';
108.                                ld_IR <= '0';
109.                                ld_PC <= '1';
110.                                inc_PC <= '1';
111.                                clr_A <= '0';
112.                                ld_A <= '1';
113.                                ld_B <= '0';
114.                                clr_B <= '0';
115.                                clr_C <= '0';
116.                                ld_C <= '0';
117.                                clr_Z <= '0';
118.                                ld_Z <= '0';
119.                                A_Mux <= '0';
120.                                DATA_Mux <= "01";
121.                                en <= '1';
122.                                wen <= '0';
123.
124.                          elsif Instruction_sig = "1010" then --LDB
125.                                clr_IR <= '0';
126.                                ld_IR <= '0';
127.                                ld_PC <= '1';
128.                                inc_PC <= '1';
129.                                clr_A <= '0';
130.                                ld_A <= '0';
131.                                ld_B <= '1';
132.                                clr_B <= '0';
133.                                clr_C <= '0';
134.                                ld_C <= '0';
135.                                clr_Z <= '0';
136.                                ld_Z <= '0';
137.                                B_Mux <= '0';
138.                                DATA_Mux <= "01";
139.                                en <= '1';
140.                                wen <= '0';
141.                          end if;
142.
143.                    elsif present_state = state_2 then
144.                          if Instruction_sig = "0101" then --JUMP
145.                                clr_IR <= '0';
146.                                ld_IR <= '0';
147.                                ld_PC <= '1';
148.                                inc_PC <= '0';
149.                                clr_A <= '0';
150.                                ld_A <= '0';
151.                                ld_B <= '0';
152.                                clr_B <= '0';
```

```
153.                            clr_C <= '0';
154.                            ld_C <= '0';
155.                            clr_Z <= '0';
156.                            ld_Z <= '0';
157.
158.                        elsif Instruction_sig = "0110" then --BEQ
159.                            clr_IR <= '0';
160.                            ld_IR <= '0';
161.                            ld_PC <= '1';
162.                            inc_PC <= '0';
163.                            clr_A <= '0';
164.                            ld_A <= '0';
165.                            ld_B <= '0';
166.                            clr_B <= '0';
167.                            clr_C <= '0';
168.                            ld_C <= '0';
169.                            clr_Z <= '0';
170.                            ld_Z <= '0';
171.
172.                        elsif Instruction_sig = "1000" then --BNQ
173.                            clr_IR <= '0';
174.                            ld_IR <= '0';
175.                            ld_PC <= '1';
176.                            inc_PC <= '0';
177.                            clr_A <= '0';
178.                            ld_A <= '0';
179.                            ld_B <= '0';
180.                            clr_B <= '0';
181.                            clr_C <= '0';
182.                            ld_C <= '0';
183.                            clr_Z <= '0';
184.                            ld_Z <= '0';
185.
186.                        elsif Instruction_sig = "1001" then --LDA
187.                            clr_IR <= '0';
188.                            ld_IR <= '0';
189.                            ld_PC <= '0';
190.                            inc_PC <= '0';
191.                            clr_A <= '0';
192.                            ld_A <= '1';
193.                            ld_B <= '0';
194.                            clr_B <= '0';
195.                            clr_C <= '0';
196.                            ld_C <= '0';
197.                            clr_Z <= '0';
198.                            ld_Z <= '0';
199.                            A_Mux <= '0';
200.                            DATA_Mux <= "01";
201.                            en <= '1';
202.                            wen <= '0';
203.
204.                        elsif Instruction_sig = "1010" then --LDB
```

```
205.                         clr_IR <= '0';
206.                         ld_IR <= '0';
207.                         ld_PC <= '0';
208.                         inc_PC <= '0';
209.                         clr_A <= '0';
210.                         ld_A <= '0';
211.                         ld_B <= '1';
212.                         clr_B <= '0';
213.                         clr_C <= '0';
214.                         ld_C <= '0';
215.                         clr_Z <= '0';
216.                         ld_Z <= '0';
217.                         B_Mux <= '0';
218.                         DATA_Mux <= "01";
219.                         en <= '1';
220.                         wen <= '0';
221.
222.                     elsif Instruction_sig = "0010" then --STA
223.                         clr_IR <= '0';
224.                         ld_IR <= '0';
225.                         ld_PC <= '0';
226.                         inc_PC <= '0';
227.                         clr_A <= '0';
228.                         ld_A <= '0';
229.                         ld_B <= '0';
230.                         clr_B <= '0';
231.                         clr_C <= '0';
232.                         ld_C <= '0';
233.                         clr_Z <= '0';
234.                         ld_Z <= '0';
235.                         REG_Mux <= '0';
236.                         DATA_Mux <= "00";
237.                         en <= '1';
238.                         wen <= '1';
239.
240.                     elsif Instruction_sig = "0011" then --STB
241.                         clr_IR <= '0';
242.                         ld_IR <= '0';
243.                         ld_PC <= '0';
244.                         inc_PC <= '0';
245.                         clr_A <= '0';
246.                         ld_A <= '0';
247.                         ld_B <= '0';
248.                         clr_B <= '0';
249.                         clr_C <= '0';
250.                         ld_C <= '0';
251.                         clr_Z <= '0';
252.                         ld_Z <= '0';
253.                         REG_Mux <= '1';
254.                         DATA_Mux <= "00";
255.                         en <= '1';
256.                         wen <= '1';
```

```
257.
258.                         elsif Instruction_sig = "0000" then --LDAI
259.                               clr_IR <= '0';
260.                               ld_IR <= '0';
261.                               ld_PC <= '0';
262.                               inc_PC <= '0';
263.                               clr_A <= '0';
264.                               ld_A <= '1';
265.                               ld_B <= '0';
266.                               clr_B <= '0';
267.                               clr_C <= '0';
268.                               ld_C <= '0';
269.                               clr_Z <= '0';
270.                               ld_Z <= '0';
271.                               A_Mux <= '1';
272.
273.                         elsif Instruction_sig = "0001" then --LDBI
274.                               clr_IR <= '0';
275.                               ld_IR <= '0';
276.                               ld_PC <= '0';
277.                               inc_PC <= '0';
278.                               clr_A <= '0';
279.                               ld_A <= '0';
280.                               ld_B <= '1';
281.                               clr_B <= '0';
282.                               clr_C <= '0';
283.                               ld_C <= '0';
284.                               clr_Z <= '0';
285.                               ld_Z <= '0';
286.                               B_Mux <= '1';
287.
288.                         elsif Instruction_sig = "0100" then --LUI
289.                               clr_IR <= '0';
290.                               ld_IR <= '0';
291.                               ld_PC <= '0';
292.                               inc_PC <= '0';
293.                               clr_A <= '0';
294.                               ld_A <= '1';
295.                               ld_B <= '0';
296.                               clr_B <= '1';
297.                               clr_C <= '0';
298.                               ld_C <= '0';
299.                               clr_Z <= '0';
300.                               ld_Z <= '0';
301.                               ALU_op <= "001";
302.                               A_Mux <= '0';
303.                               DATA_Mux <= "10";
304.                               IM_MUX1 <= '1';
305.
306.                         elsif Instruction_sig2 = "01111001" then --ANDI
307.                               clr_IR <= '0';
308.                               ld_IR <= '0';
```

```
309.                            ld_PC <= '0';
310.                            inc_PC <= '0';
311.                            clr_A <= '0';
312.                            ld_A <= '1';
313.                            ld_B <= '0';
314.                            clr_B <= '0';
315.                            clr_C <= '0';
316.                            ld_C <= '1';
317.                            clr_Z <= '0';
318.                            ld_Z <= '1';
319.                            ALU_op <= "000";
320.                            A_Mux <= '0';
321.                            DATA_Mux <= "10";
322.                            IM_MUX1 <= '0';
323.                            IM_MUX2 <= "01";
324.
325.                        elsif Instruction_sig2 = "01111110" then --DECA
326.                            clr_IR <= '0';
327.                            ld_IR <= '0';
328.                            ld_PC <= '0';
329.                            inc_PC <= '0';
330.                            clr_A <= '0';
331.                            ld_A <= '1';
332.                            ld_B <= '0';
333.                            clr_B <= '0';
334.                            clr_C <= '0';
335.                            ld_C <= '1';
336.                            clr_Z <= '0';
337.                            ld_Z <= '1';
338.                            ALU_op <= "110";
339.                            A_Mux <= '0';
340.                            DATA_Mux <= "10";
341.                            IM_MUX1 <= '0';
342.                            IM_MUX2 <= "10";
343.
344.                        elsif Instruction_sig2 = "01110000" then --ADD
345.                            clr_IR <= '0';
346.                            ld_IR <= '0';
347.                            ld_PC <= '0';
348.                            inc_PC <= '0';
349.                            clr_A <= '0';
350.                            ld_A <= '1';
351.                            ld_B <= '0';
352.                            clr_B <= '0';
353.                            clr_C <= '0';
354.                            ld_C <= '1';
355.                            clr_Z <= '0';
356.                            ld_Z <= '1';
357.                            ALU_op <= "010";
358.                            A_Mux <= '0';
359.                            DATA_Mux <= "10";
360.                            IM_MUX1 <= '0';
```

```
361.                            IM_MUX2 <= "00";
362.
363.                        elsif Instruction_sig2 = "01110010" then --SUB
364.                            clr_IR <= '0';
365.                            ld_IR <= '0';
366.                            ld_PC <= '0';
367.                            inc_PC <= '0';
368.                            clr_A <= '0';
369.                            ld_A <= '1';
370.                            ld_B <= '0';
371.                            clr_B <= '0';
372.                            clr_C <= '0';
373.                            ld_C <= '1';
374.                            clr_Z <= '0';
375.                            ld_Z <= '1';
376.                            ALU_op <= "110";
377.                            A_Mux <= '0';
378.                            DATA_Mux <= "10";
379.                            IM_MUX1 <= '0';
380.                            IM_MUX2 <= "00";
381.
382.                        elsif Instruction_sig2 = "01110011" then --INCA
383.                            clr_IR <= '0';
384.                            ld_IR <= '0';
385.                            ld_PC <= '0';
386.                            inc_PC <= '0';
387.                            clr_A <= '0';
388.                            ld_A <= '1';
389.                            ld_B <= '0';
390.                            clr_B <= '0';
391.                            clr_C <= '0';
392.                            ld_C <= '1';
393.                            clr_Z <= '0';
394.                            ld_Z <= '1';
395.                            ALU_op <= "010";
396.                            A_Mux <= '0';
397.                            DATA_Mux <= "10";
398.                            IM_MUX1 <= '0';
399.                            IM_MUX2 <= "10";
400.
401.                        elsif Instruction_sig2 = "01111011" then --AND
402.                            clr_IR <= '0';
403.                            ld_IR <= '0';
404.                            ld_PC <= '0';
405.                            inc_PC <= '0';
406.                            clr_A <= '0';
407.                            ld_A <= '1';
408.                            ld_B <= '0';
409.                            clr_B <= '0';
410.                            clr_C <= '0';
411.                            ld_C <= '1';
412.                            clr_Z <= '0';
```

```
413.                            ld_Z <= '1';
414.                            ALU_op <= "000";
415.                            A_Mux <= '0';
416.                            DATA_Mux <= "10";
417.                            IM_MUX1 <= '0';
418.                            IM_MUX2 <= "00";
419.
420.                        elsif Instruction_sig2 = "01110001" then --ADDI
421.                            clr_IR <= '0';
422.                            ld_IR <= '0';
423.                            ld_PC <= '0';
424.                            inc_PC <= '0';
425.                            clr_A <= '0';
426.                            ld_A <= '1';
427.                            ld_B <= '0';
428.                            clr_B <= '0';
429.                            clr_C <= '0';
430.                            ld_C <= '1';
431.                            clr_Z <= '0';
432.                            ld_Z <= '1';
433.                            ALU_op <= "010";
434.                            A_Mux <= '0';
435.                            DATA_Mux <= "10";
436.                            IM_MUX1 <= '0';
437.                            IM_MUX2 <= "01";
438.
439.                        elsif Instruction_sig2 = "01111101" then --ORI
440.                            clr_IR <= '0';
441.                            ld_IR <= '0';
442.                            ld_PC <= '0';
443.                            inc_PC <= '0';
444.                            clr_A <= '0';
445.                            ld_A <= '1';
446.                            ld_B <= '0';
447.                            clr_B <= '0';
448.                            clr_C <= '0';
449.                            ld_C <= '1';
450.                            clr_Z <= '0';
451.                            ld_Z <= '1';
452.                            ALU_op <= "001";
453.                            A_Mux <= '0';
454.                            DATA_Mux <= "10";
455.                            IM_MUX1 <= '0';
456.                            IM_MUX2 <= "01";
457.
458.                        elsif Instruction_sig2 = "01110100" then --ROL
459.                            clr_IR <= '0';
460.                            ld_IR <= '0';
461.                            ld_PC <= '0';
462.                            inc_PC <= '0';
463.                            clr_A <= '0';
464.                            ld_A <= '1';
```

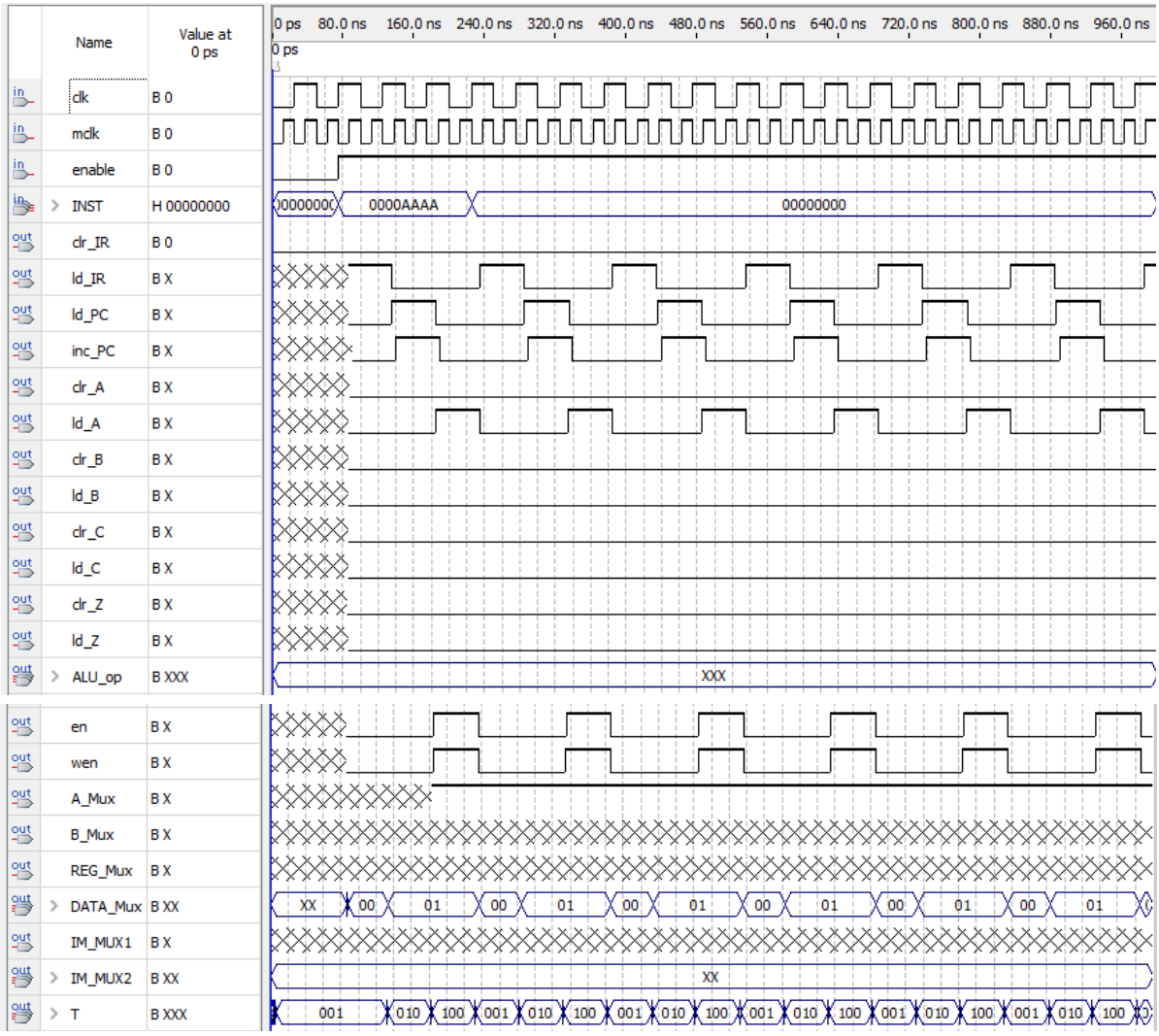
```
465.                            ld_B <= '0';
466.                            clr_B <= '0';
467.                            clr_C <= '0';
468.                            ld_C <= '1';
469.                            clr_Z <= '0';
470.                            ld_Z <= '1';
471.                            ALU_op <= "100";
472.                            A_Mux <= '0';
473.                            DATA_Mux <= "10";
474.                            IM_MUX1 <= '0';
475.
476.                        elsif Instruction_sig2 = "01111111" then --ROR
477.                            clr_IR <= '0';
478.                            ld_IR <= '0';
479.                            ld_PC <= '0';
480.                            inc_PC <= '0';
481.                            clr_A <= '0';
482.                            ld_A <= '1';
483.                            ld_B <= '0';
484.                            clr_B <= '0';
485.                            clr_C <= '0';
486.                            ld_C <= '1';
487.                            clr_Z <= '0';
488.                            ld_Z <= '1';
489.                            ALU_op <= "101";
490.                            A_Mux <= '0';
491.                            DATA_Mux <= "10";
492.                            IM_MUX1 <= '0';
493.
494.                        elsif Instruction_sig2 = "01110101" then --CLR_A
495.                            clr_IR <= '0';
496.                            ld_IR <= '0';
497.                            ld_PC <= '0';
498.                            inc_PC <= '0';
499.                            clr_A <= '1';
500.                            ld_A <= '0';
501.                            ld_B <= '0';
502.                            clr_B <= '0';
503.                            clr_C <= '0';
504.                            ld_C <= '0';
505.                            clr_Z <= '0';
506.                            ld_Z <= '0';
507.
508.                        elsif Instruction_sig2 = "01110110" then --CLR_B
509.                            clr_IR <= '0';
510.                            ld_IR <= '0';
511.                            ld_PC <= '0';
512.                            inc_PC <= '0';
513.                            clr_A <= '0';
514.                            ld_A <= '0';
515.                            ld_B <= '0';
516.                            clr_B <= '1';
```

```
517.                        clr_C <= '0';
518.                        ld_C <= '0';
519.                        clr_Z <= '0';
520.                        ld_Z <= '0';
521.
522.                  elsif Instruction_sig2 = "01110111" then --CLR_C
523.                        clr_IR <= '0';
524.                        ld_IR <= '0';
525.                        ld_PC <= '0';
526.                        inc_PC <= '0';
527.                        clr_A <= '0';
528.                        ld_A <= '0';
529.                        ld_B <= '0';
530.                        clr_B <= '0';
531.                        clr_C <= '1';
532.                        ld_C <= '0';
533.                        clr_Z <= '0';
534.                        ld_Z <= '0';
535.
536.                  elsif Instruction_sig2 = "01111000" then --CLR_Z
537.                        clr_IR <= '0';
538.                        ld_IR <= '0';
539.                        ld_PC <= '0';
540.                        inc_PC <= '0';
541.                        clr_A <= '0';
542.                        ld_A <= '0';
543.                        ld_B <= '0';
544.                        clr_B <= '0';
545.                        clr_C <= '0';
546.                        ld_C <= '0';
547.                        clr_Z <= '1';
548.                        ld_Z <= '0';
549.
550.                  elsif Instruction_sig2 = "01111010" then --TSTZ
551.                        if(statusZ = '1') then
552.                              clr_IR <= '0'; --Increment PC Counter
553.                              ld_IR <= '0';
554.                              ld_PC <= '1';
555.                              inc_PC <= '1';
556.                              clr_A <= '0';
557.                              ld_A <= '0';
558.                              ld_B <= '0';
559.                              clr_B <= '0';
560.                              clr_C <= '0';
561.                              ld_C <= '0';
562.                              clr_Z <= '0';
563.                              ld_Z <= '0';
564.                        end if;
565.
566.                  elsif Instruction_sig2 = "01111100" then --TSTC
567.                        if(statusC = '1') then
568.                              clr_IR <= '0'; --Increment PC Counter
```

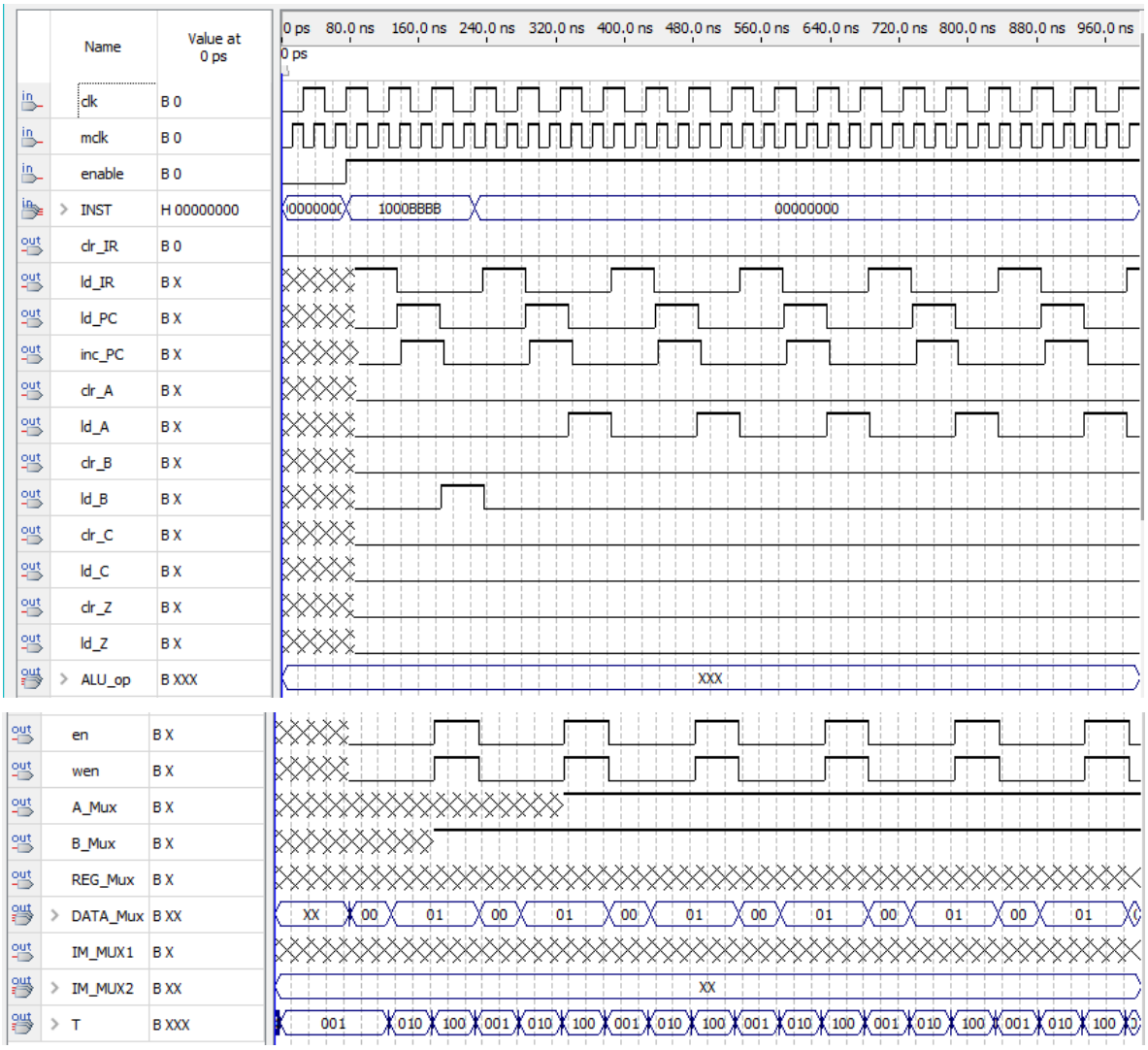
```
                                        ld_IR <= '0';
                                        ld_PC <= '1';
                                        inc_PC <= '1';
                                        clr_A <= '0';
                                        ld_A <= '0';
                                        ld_B <= '0';
                                        clr_B <= '0';
                                        clr_C <= '0';
                                        ld_C <= '0';
                                        clr_Z <= '0';
                                        ld_Z <= '0';
                                  end if;
                            end if;
                      end if;
                end if;
        END process;

    ---------STATE MACHINE------------
        PROCESS(clk, enable)
        begin
                if enable = '1' then
                      if rising_edge(clk) then
                            if present_state = state_0 then present_state <=
    state_1;
                            elsif present_state = state_1 then present_state
    <= state_2;
                            else present_state <= state_0;
                            end if;
                      end if;
                else present_state <= state_0;
                end if;
        END process;

        WITH present_state select
                T <=   "001" when state_0,
                            "010" when state_1,
                            "100" when state_2,
                            "001" when others;
    END description;
```

2.0 Timing Waveforms

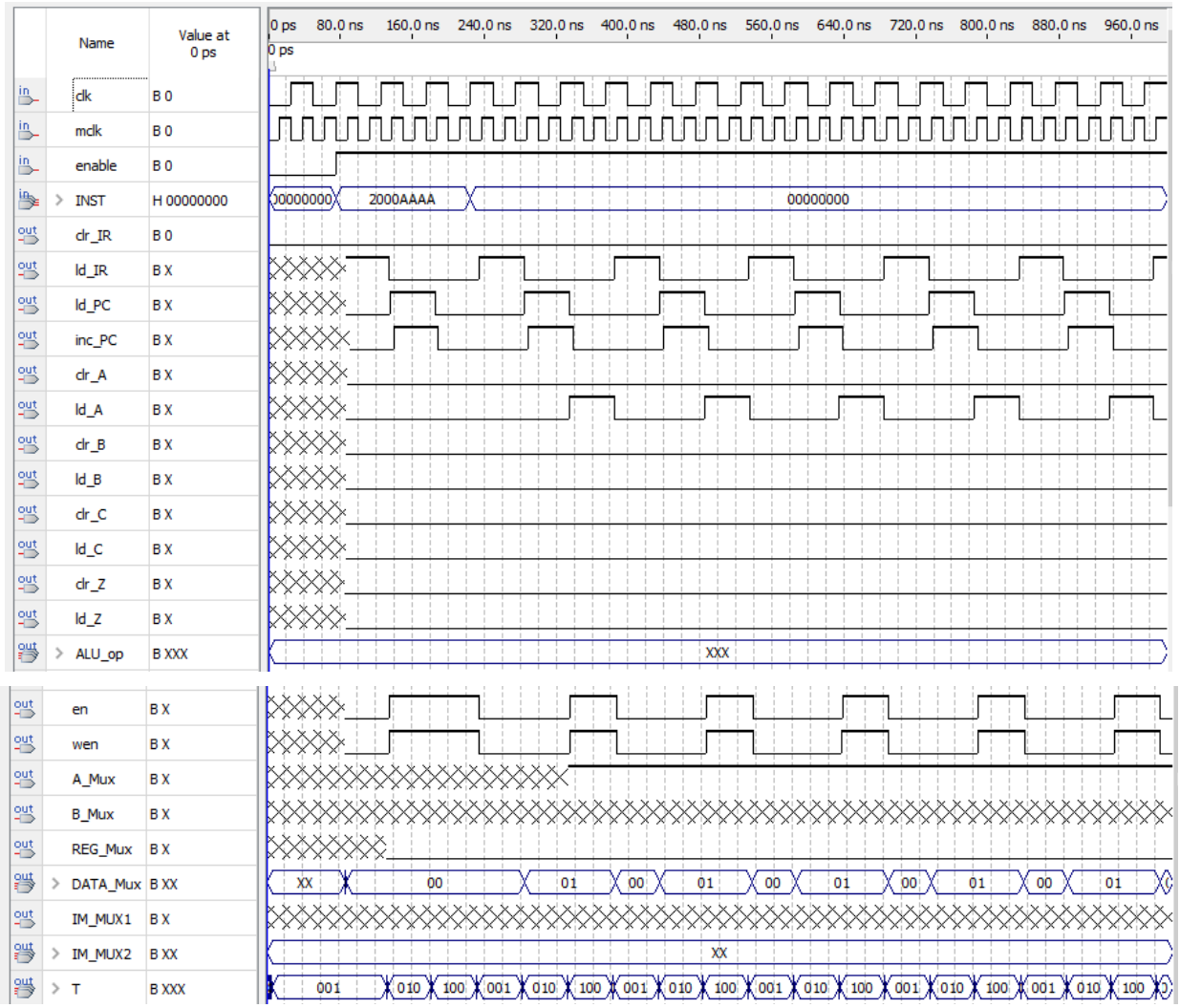
2.1 LDAI



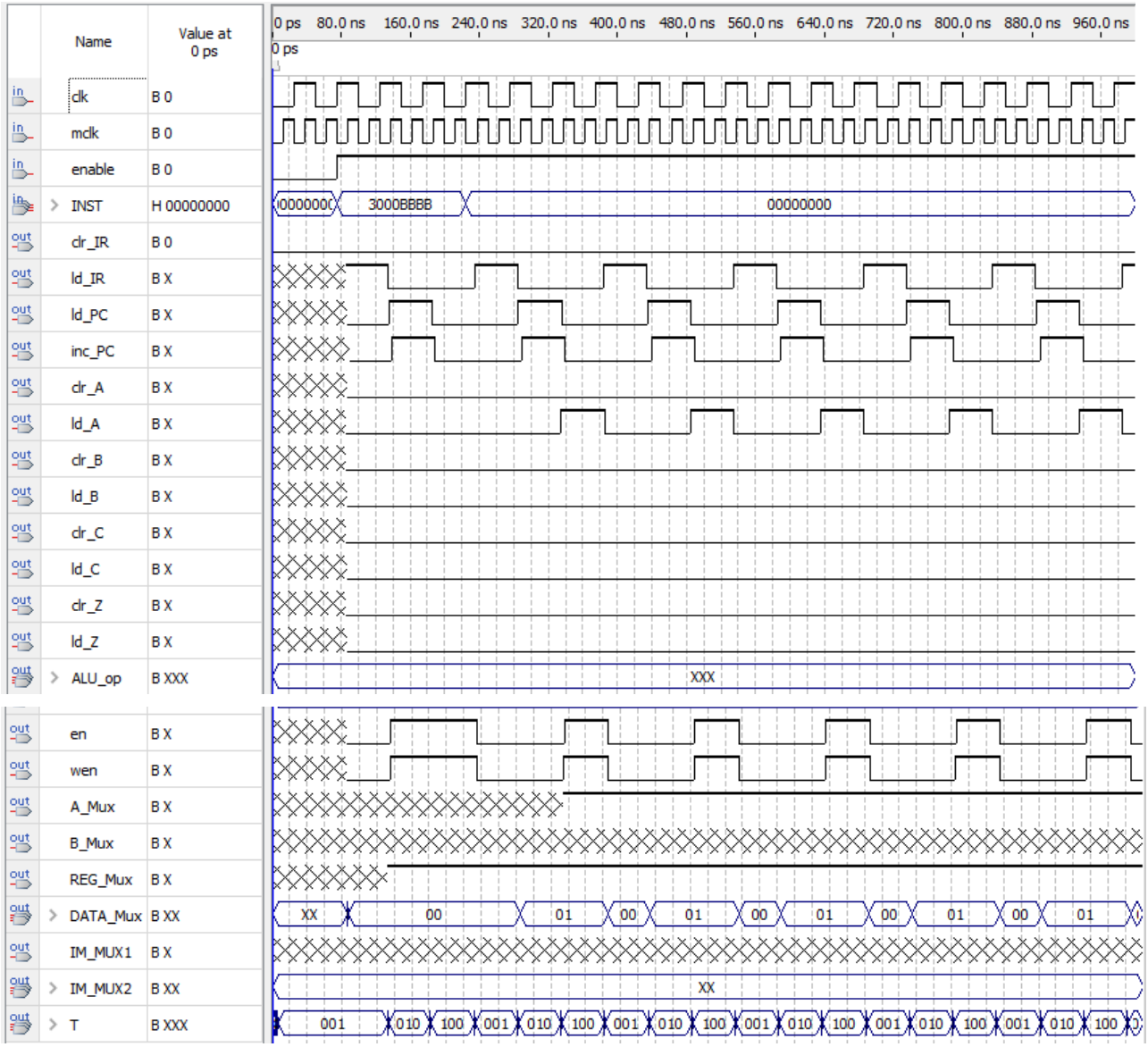
2.2 LDBI


Name
Value at 0 ps
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
880.0 ns
960.0 ns
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
clr_Z
ld_Z
ALU_op
B 0
H 00000000
B X
B XXX
0000000
1000BBBB
00000000
XXX
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
B XX
B XXX
XX
00
01
001
010
100

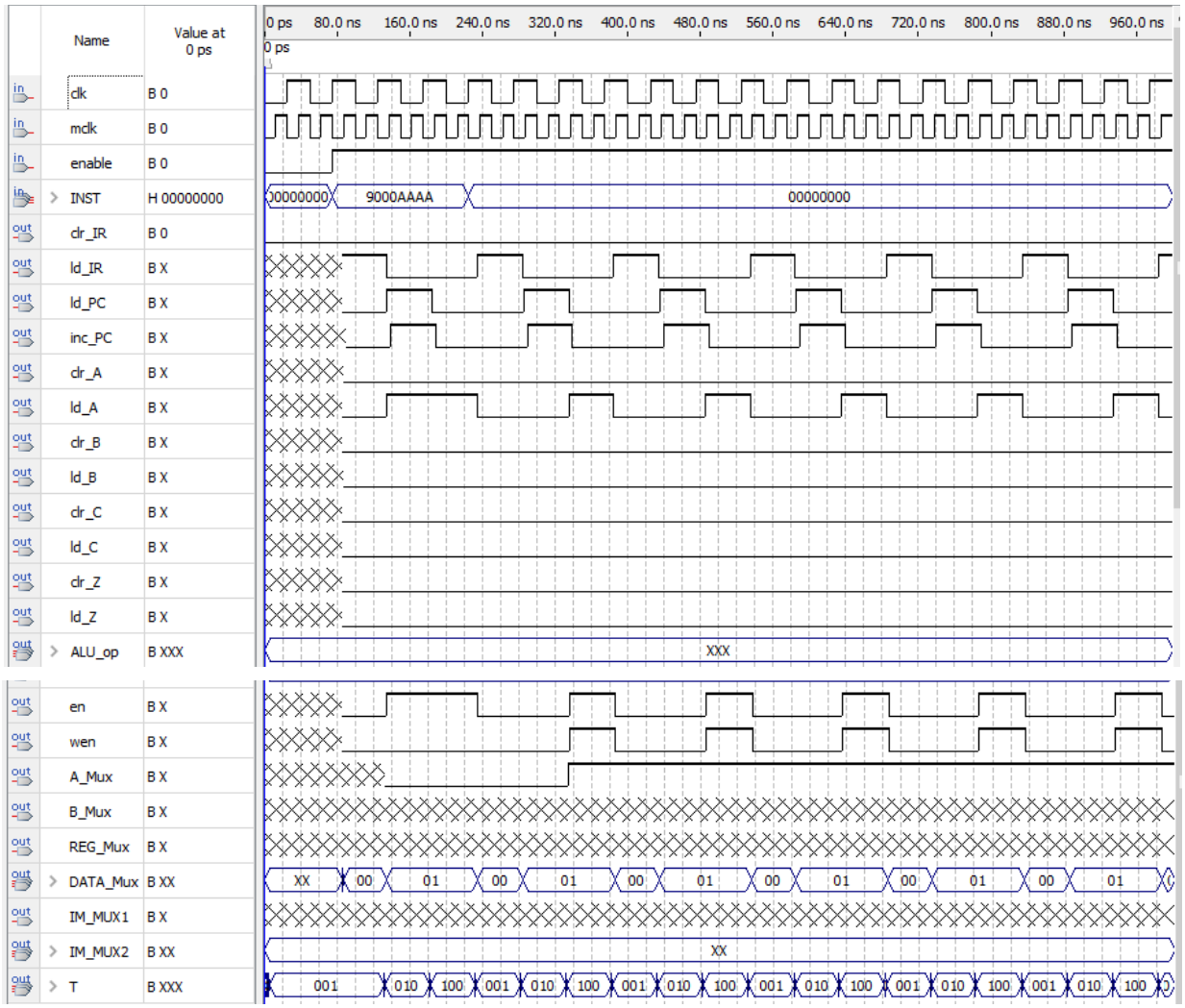
2.3 STA


Name
Value at 0 ps
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
880.0 ns
960.0 ns
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
clr_Z
ld_Z
ALU_op
00000000
2000AAAA
00000000
XXX
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
XX
00
01
XX
001
010
100

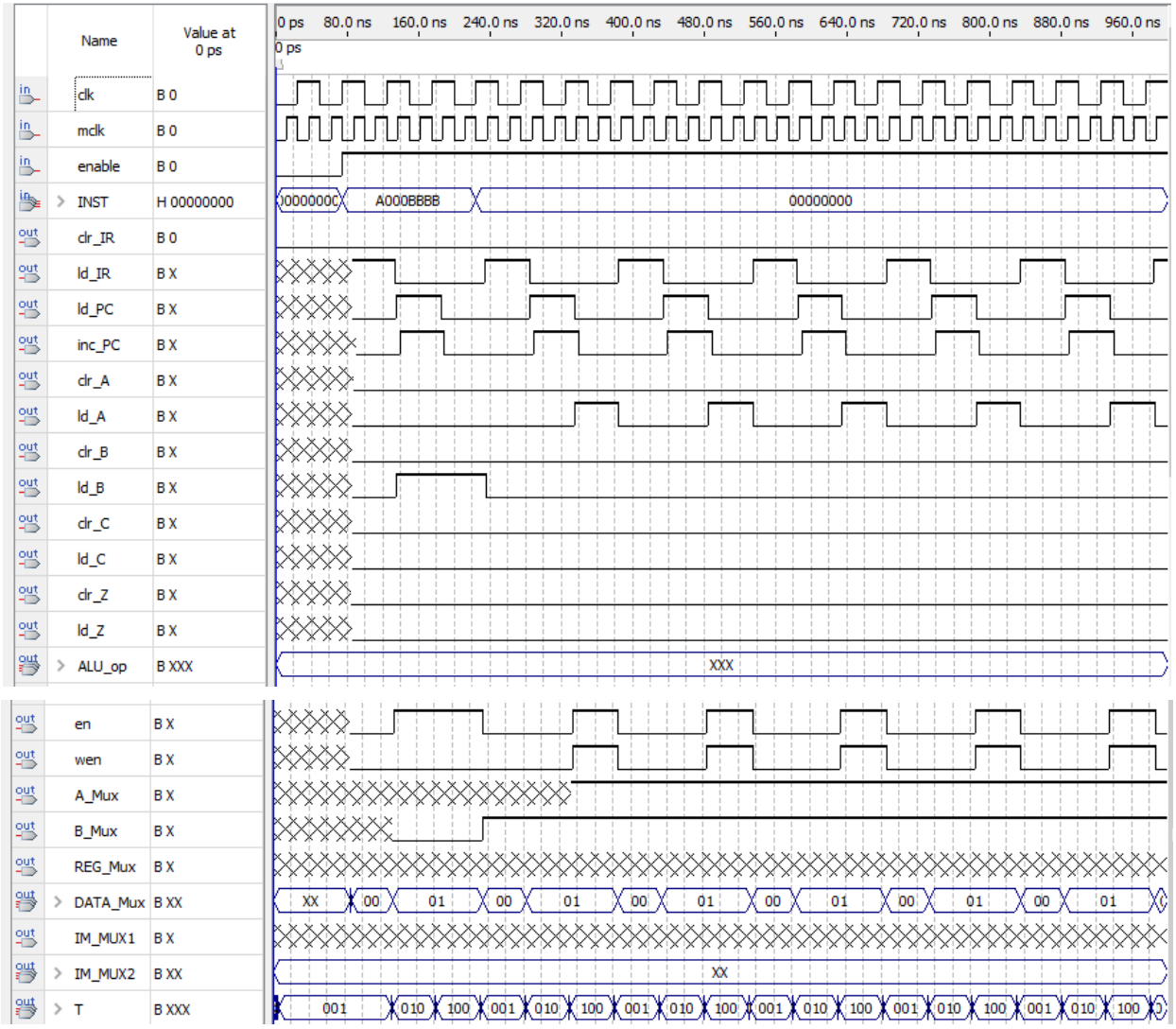
2.4 STB


Name
Value at 0 ps
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
880.0 ns
960.0 ns
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
clr_Z
ld_Z
ALU_op
B 0
H 00000000
B X
B XXX
3000BBBB
00000000
XXX
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
B XX
B XXX
XX
00
01
001
010
100

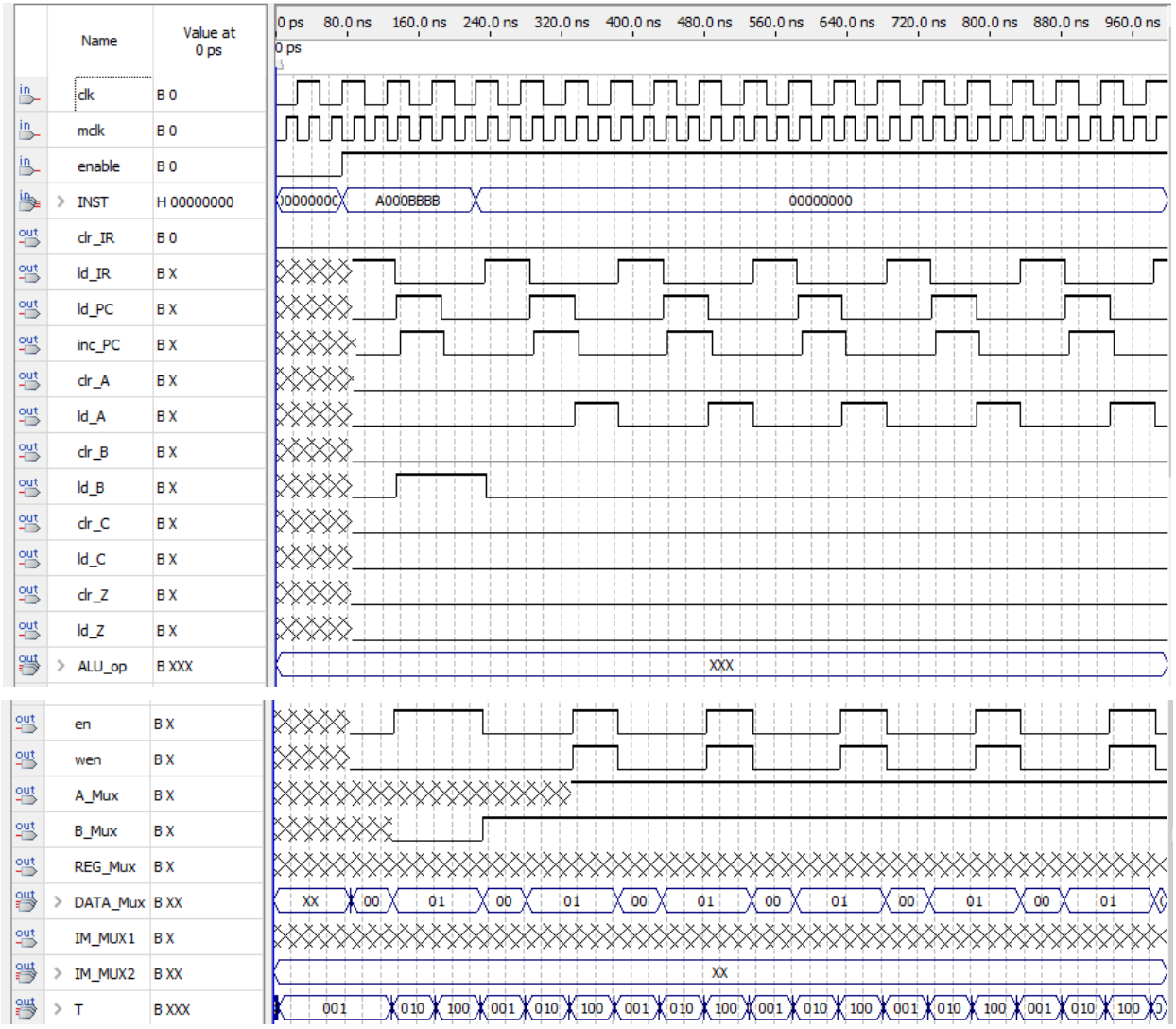
2.5 LDA


Name
Value at 0 ps
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
880.0 ns
960.0 ns
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
clr_Z
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
9000AAAA
00000000
XXX
XX
00
01
001
010
100

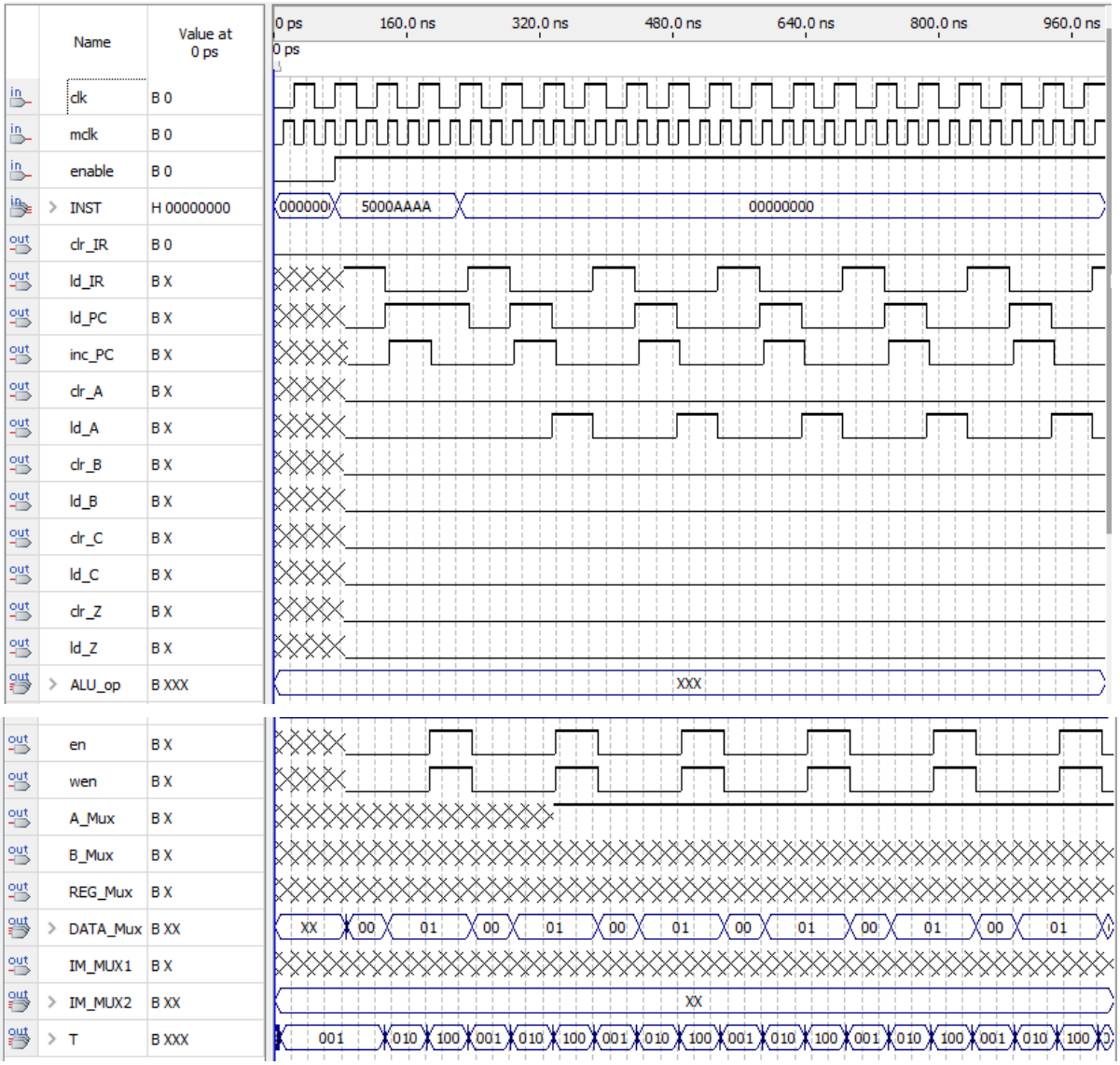
2.6 LDB


Name
Value at 0 ps
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
880.0 ns
960.0 ns
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
clr_Z
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
A000BBBB
00000000
XXX
XX
001
010
100

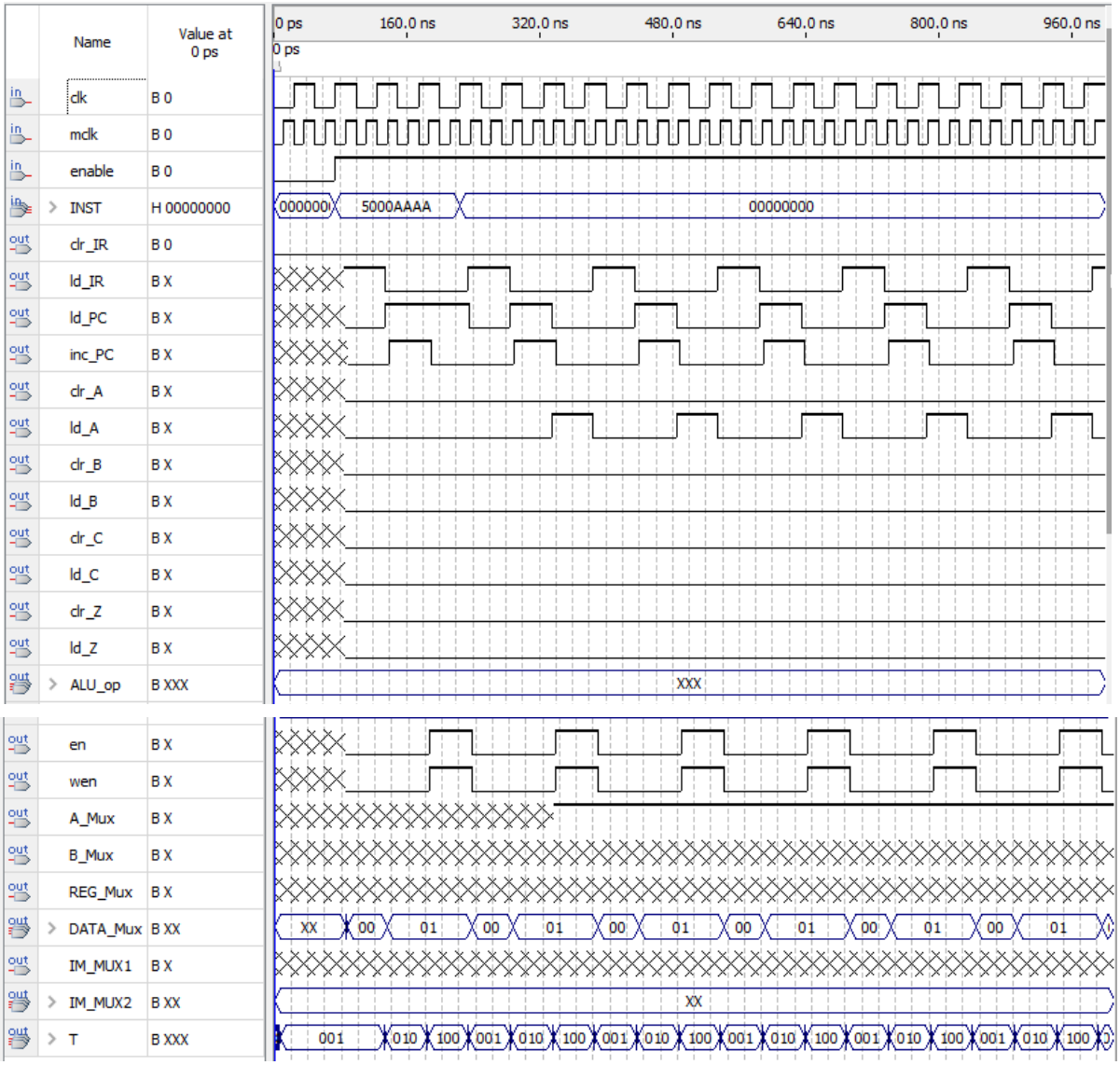
2.7 LUI


Name
Value at 0 ps
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
880.0 ns
960.0 ns
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
clr_Z
ld_Z
ALU_op
B 0
H 00000000
B X
B XXX
A000BBBB
00000000
XXX
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
B XX
B XXX
XX
00
01
001
010
100

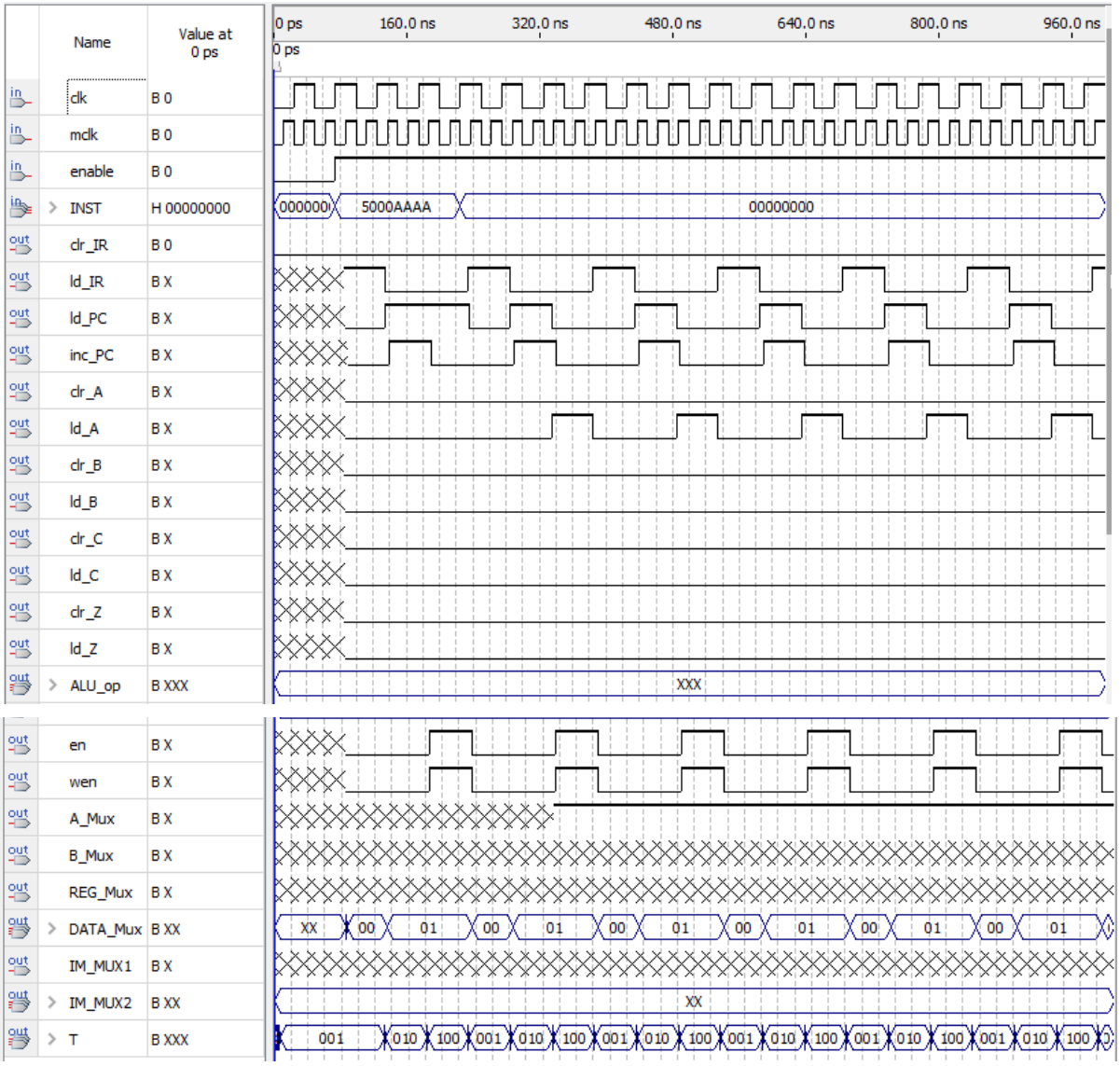
2.8 JMP


Name
Value at 0 ps
0 ps
160.0 ns
320.0 ns
480.0 ns
640.0 ns
800.0 ns
960.0 ns
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
clr_Z
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
000000
5000AAAA
00000000
XXX
XX
00
01
001
010
100

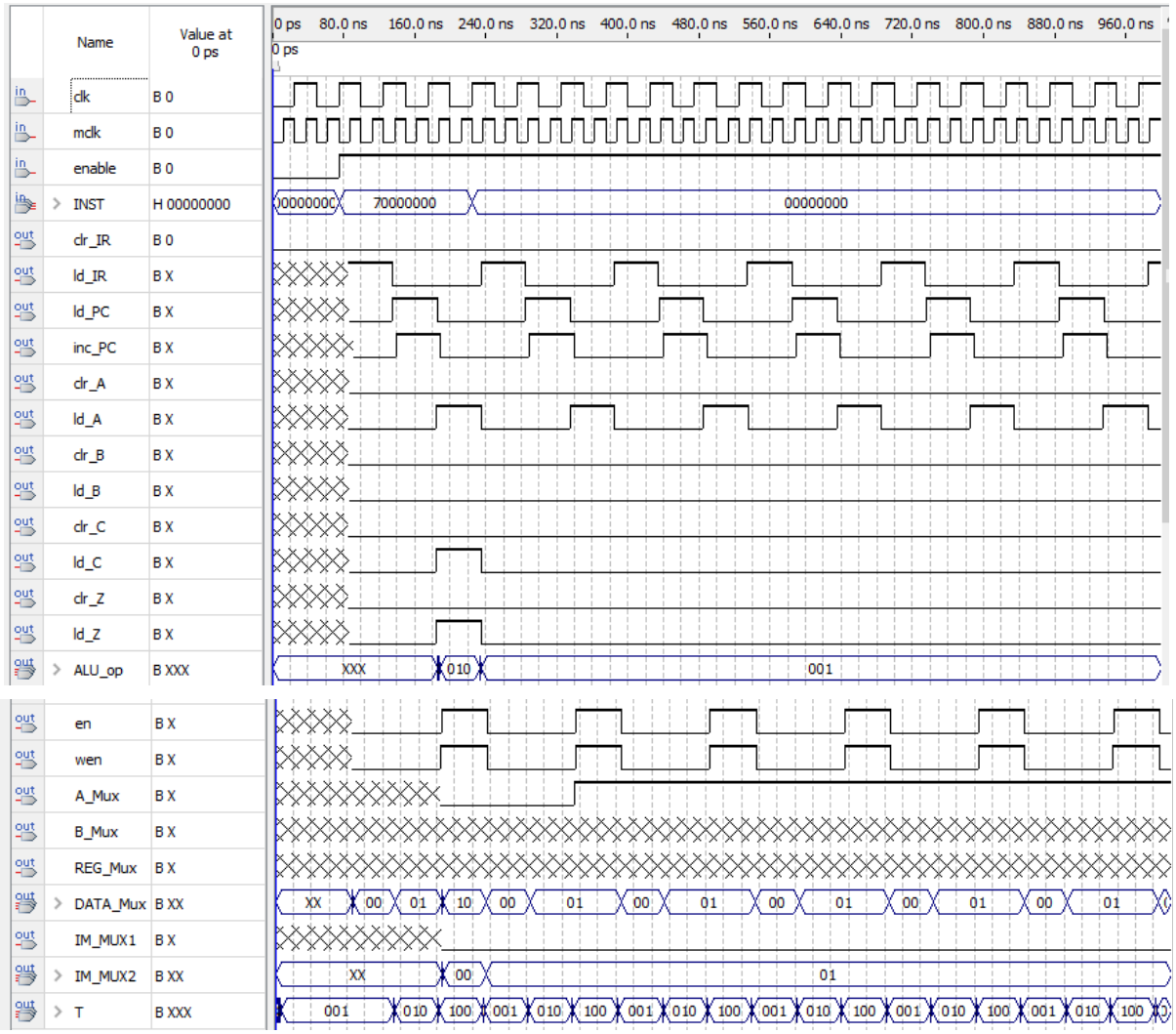
2.9 BEQ


Name
Value at 0 ps
0 ps
160.0 ns
320.0 ns
480.0 ns
640.0 ns
800.0 ns
960.0 ns
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
clr_Z
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
000000
5000AAAA
00000000
XXX
XX
00
01
001
010
100

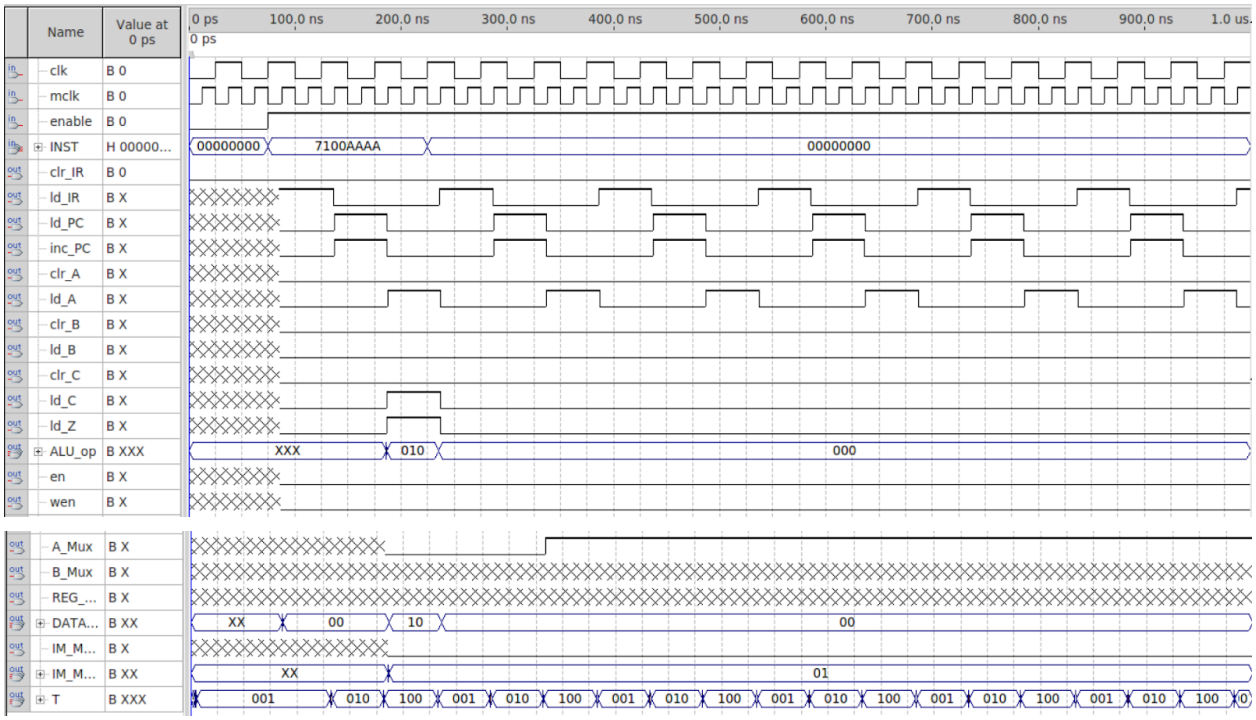
2.10 BNE


Name
Value at 0 ps
0 ps
160.0 ns
320.0 ns
480.0 ns
640.0 ns
800.0 ns
960.0 ns
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
clr_Z
ld_Z
ALU_op
000000
5000AAAA
00000000
XXX
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
XX
00
01
XX
001
010
100

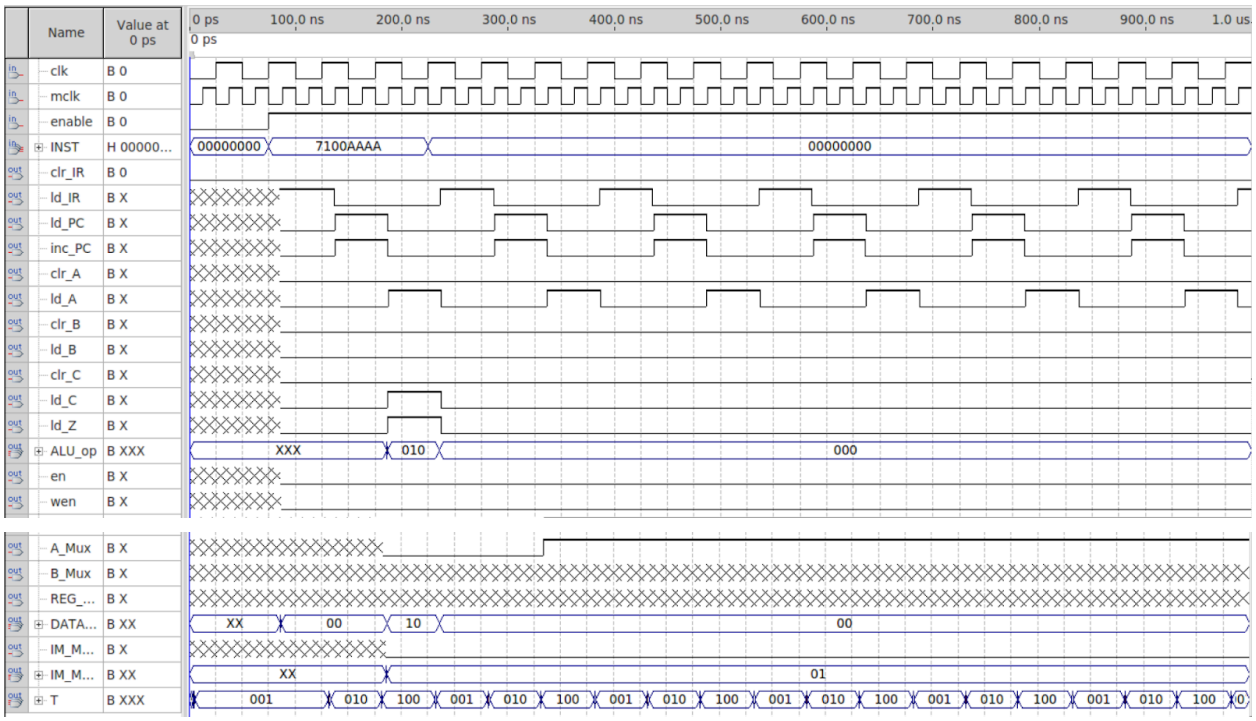
2.11 ADD


Name
Value at 0 ps
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
880.0 ns
960.0 ns
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
clr_Z
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T

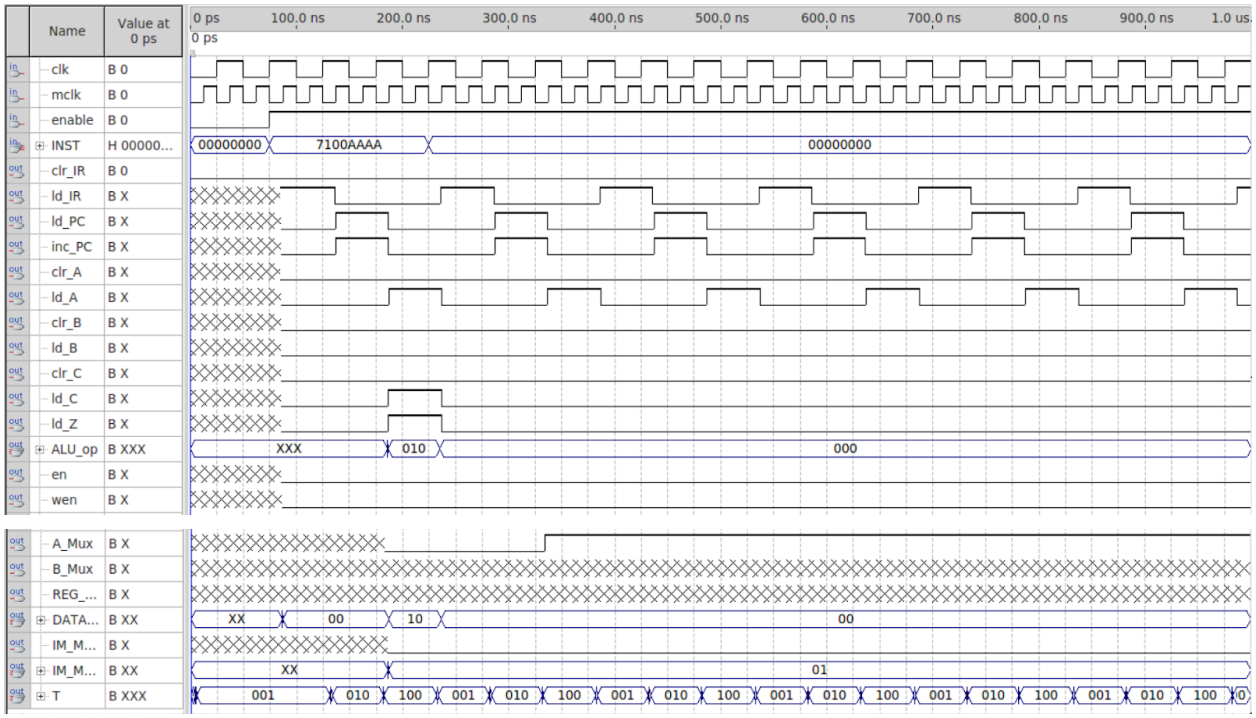
2.12 ADDI



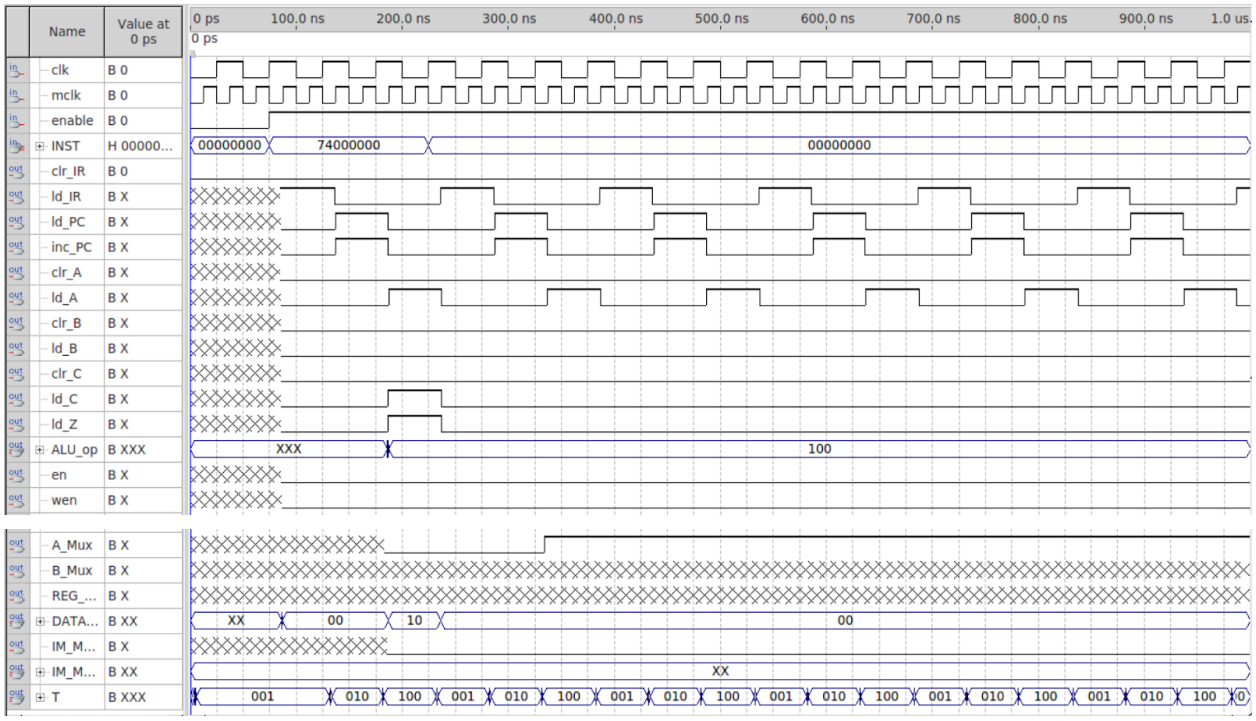
2.13 SUB



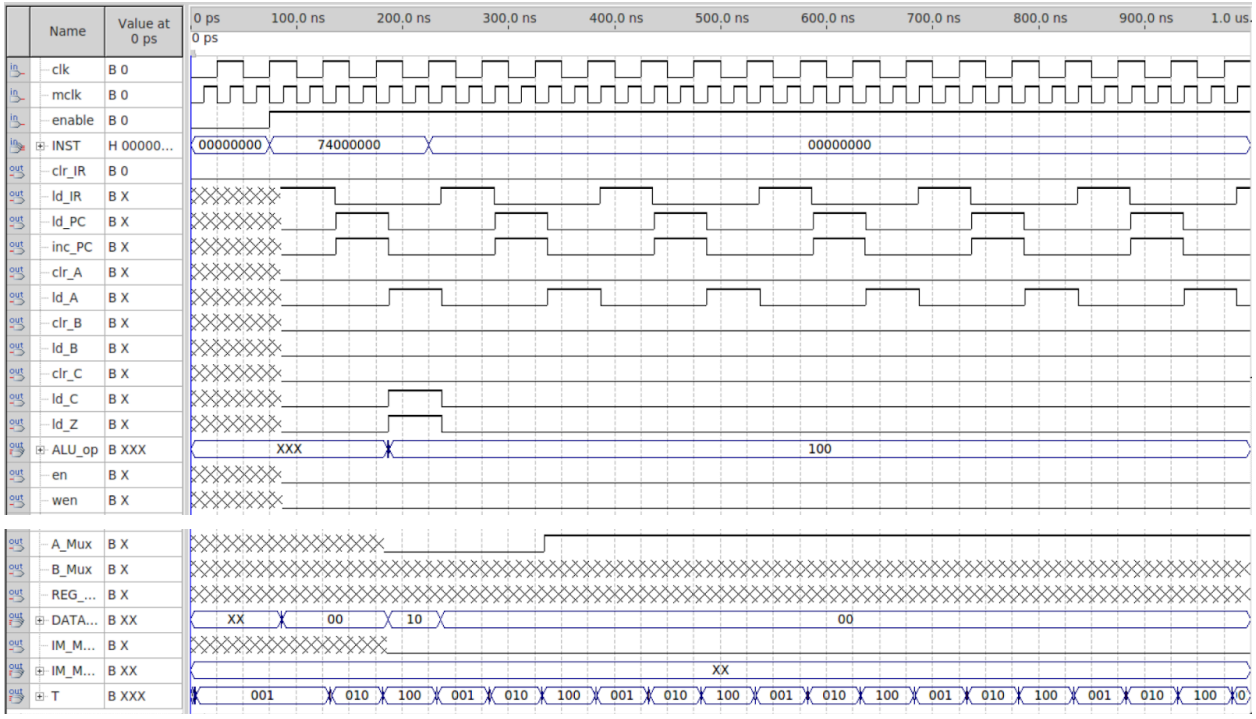
2.14 INCA



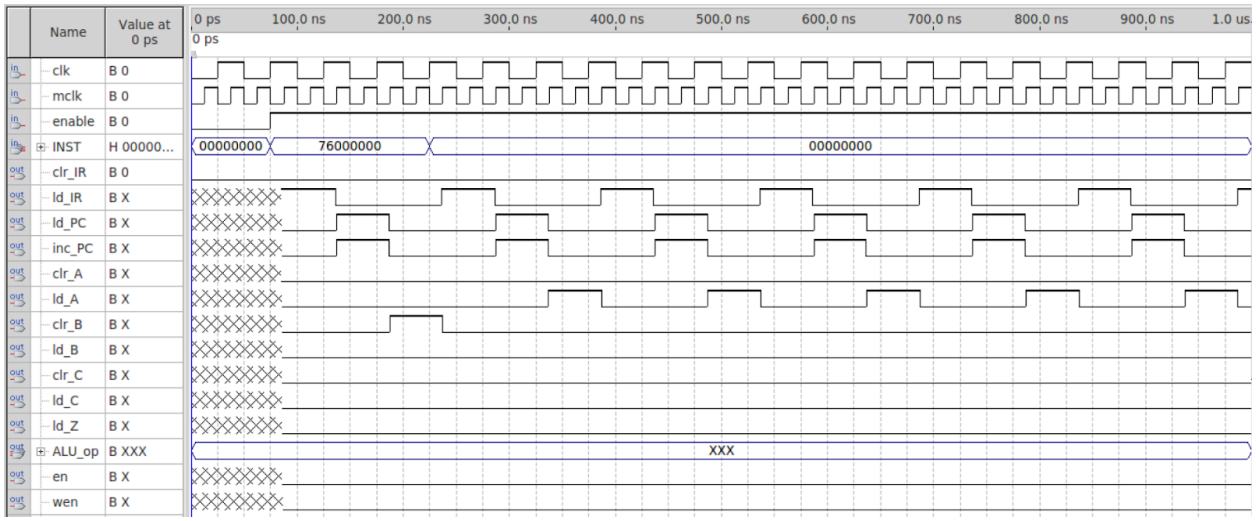
2.15 ROL

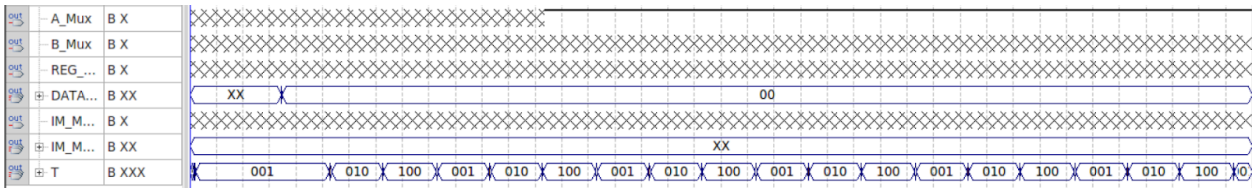


2.16 CLRA

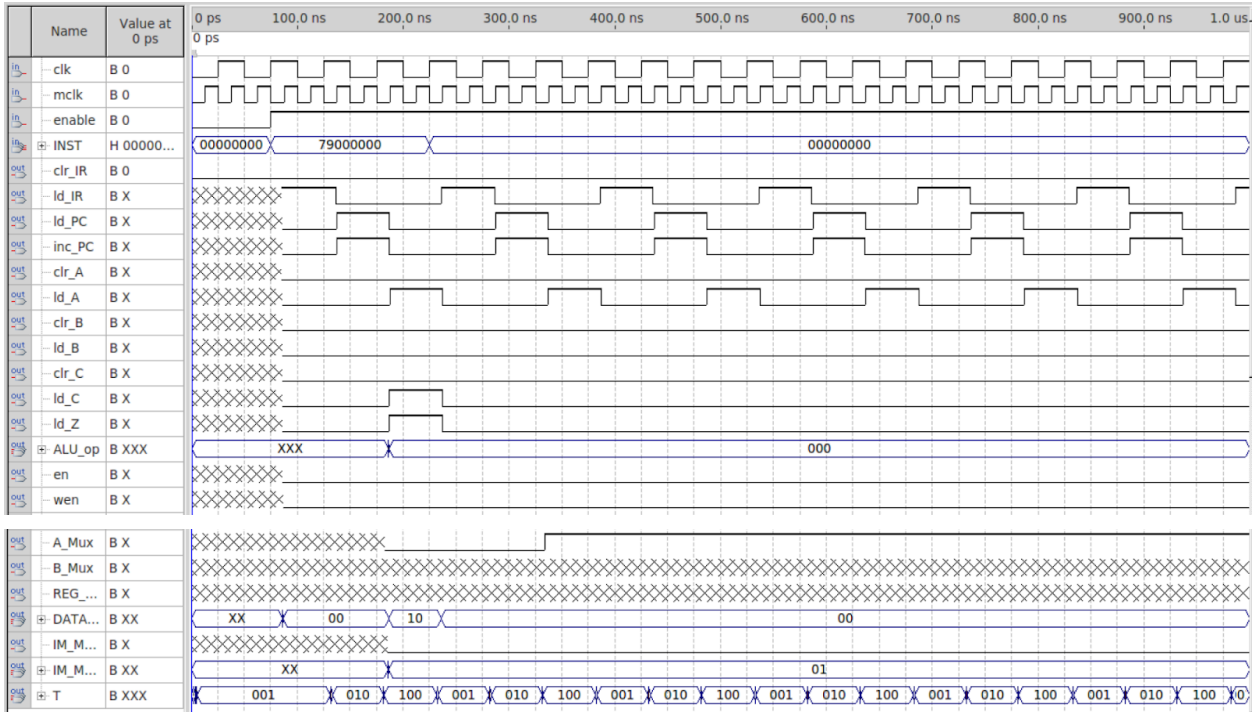


2.17 CLRB

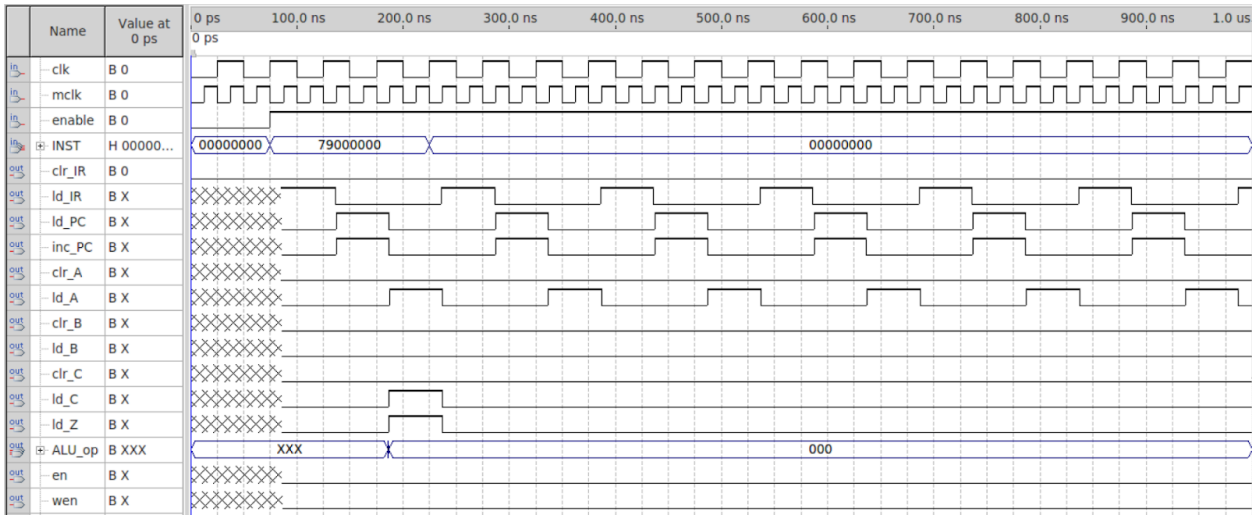


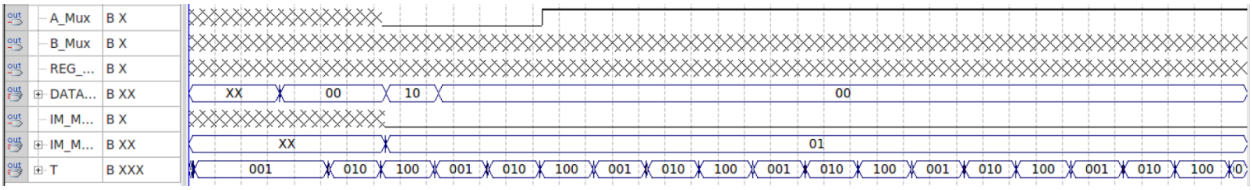


2.18 ANDI

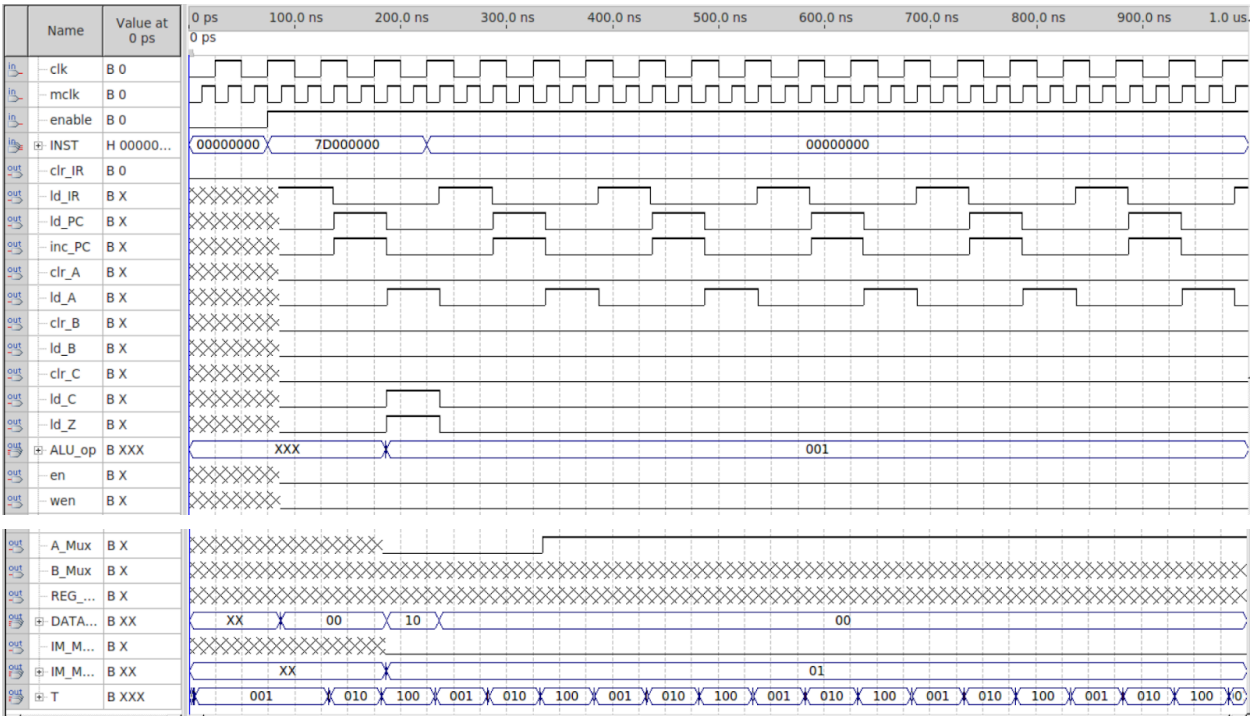


2.19 AND

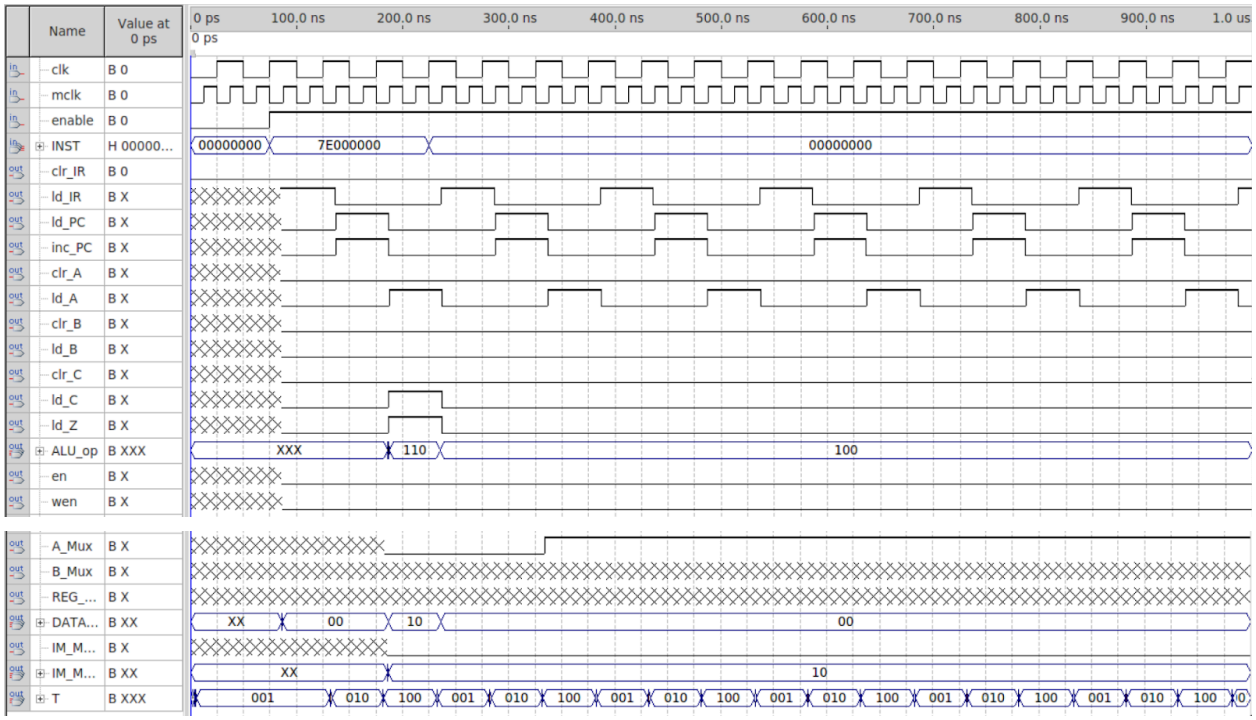



A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
IM_M... B X
IM_M... B XX
T B XXX
XX
00
10
00
XX
01
001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 001 010 100

2.20 ORI


Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
ALU_op B XXX
en B X
wen B X
00000000
7D000000
00000000
XXX
001
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
IM_M... B X
IM_M... B XX
T B XXX
XX
00
10
00
XX
01
001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 001 010 100

2.21 DECA



2.22 ROR