

## Innehållsförteckning

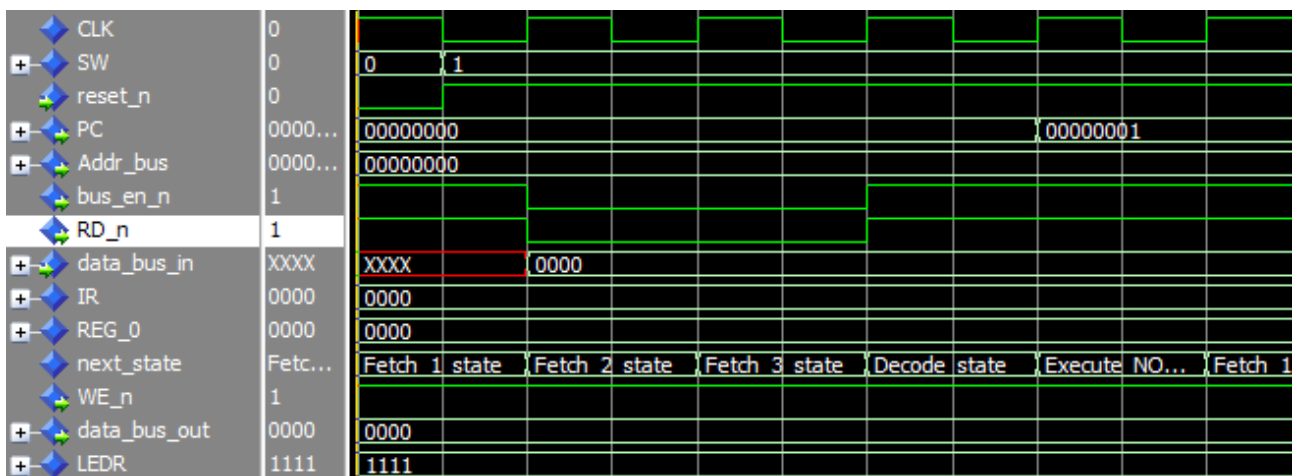
1 Sammanfattning .....	1
2 Tidsanalys .....	
2.1 Cykler 1-14.....	1
2.2 Cykler 15-19.....	4
2.3 Cykler 20 – 24 .....	5
2.4 Cykler 25-29.....	7
2.5 Cykler 30 – 34.....	8
2.6 Cykler 35-39.....	9
2.7 Cykler 40 – 44.....	10
2.8 Cykler 45-49.....	11

## 1 Sammanfattning

Verifiering med Modelsim har gjorts av enkel CPU. 50 instruktionscykel har stegats igenom, och värdet i CPU:ns register, adress och data bussar, samt lysdioderna status har analyserats.

## 2 Tidsanalys

### 2.1 Cykler 1-14



CLK_PERIOD	1 to 10	11	12	13	14
SW	0	1	1	1	1
reset_n	0	1	1	1	1
PC	0b00000000	0b00000000	0b00000000	0b00000000	0b00000001
Addr_bus	0b00000000	0b00000000	0b00000000	0b00000000	0b00000000
bus_en_n	1	0	0	1	1
RD_n	1	0	0	1	1
data_bus_in	0xUUUU	0x0000	0x0000	0x0000	0x0000
IR	0x0000	0x0000	0x0000	0x0000	0x0000
REG_0	0x0000	0x0000	0x0000	0x0000	0x0000
next_state	Fetch_1_state	Fetch_2_state	Fetch_3_state	Decode	Execute_NOP
WE_n	1	1	1	1	1
data_bus_out	0x0000	0x0000	0x0000	0x0000	0x0000
LEDR	0b1111	0b01111	0b1111	0b1111	0b1111

Fetch\_1\_state ligger som next\_state fram till att reset\_n går från hög till låg, samtliga register är nollställda under reset-fasen, vi har nämligen kod (klistrar in en del av den)

```
process(reset_n, clk_50)
begin

if reset_n = '0' then
    PC_reg <= X"00"; -- after reset, PC is zero (i.e address 0)
    IR <= X"0000"; -- NOP, no operation
    Addr_bus <= X"00" ; -- initialize registers
    CPU_REG_0 <= (others => '0'); -- initialize registers
    next_state <= Fetch_1_state;
    bus_en_n <= '1';
    data_bus_out <= "0000000000000000"; -- initialize registers
    WE_n <= '1';
    RD_n <= '1';
    CPU_state <= "00";
```

```
elsif(rising_edge(clk_50)) then
```

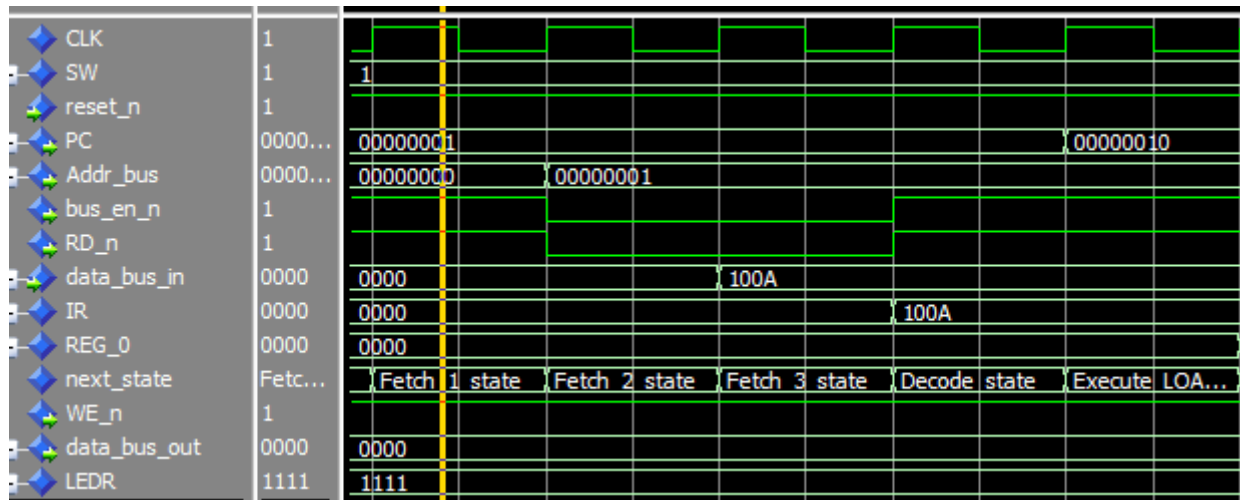
Lysdioderna lyser p.g.a. koden i OUT\_LED

```
process(reset_n, clk_50)
begin
    if reset_n = '0' then
        LEDG <= "1111";    -- LED-signals go high during reset

    elsif(rising_edge(clk_50)) then
```

Instruktionsläsningen kräver 2 cykler, där IR har laddats med instruktionskoden för NOP 0x0000 och är giltig under cykel 14.

## 2. 2 Cykler 15-19



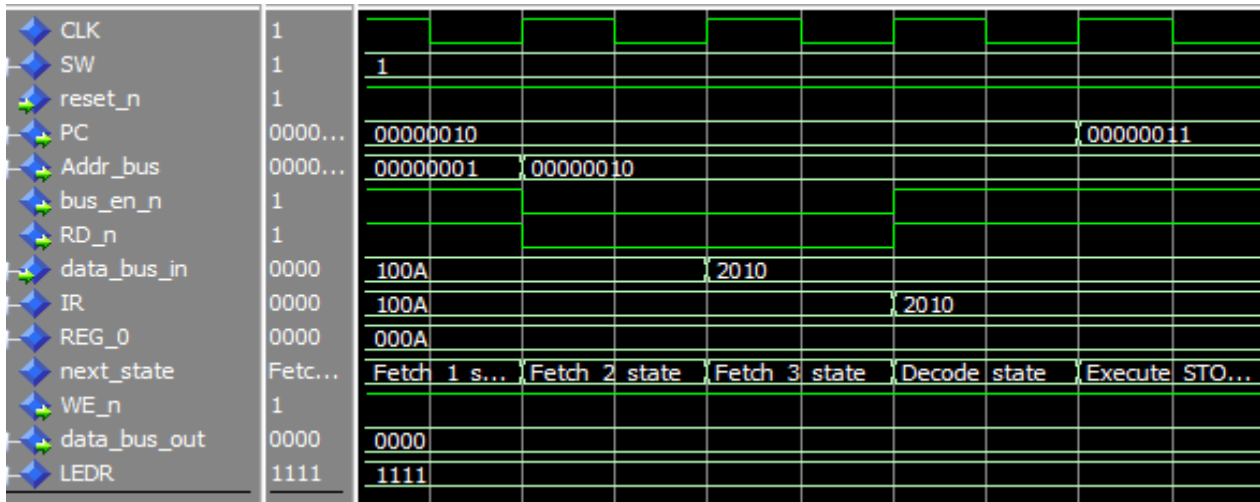
CLK_PERIOD	15	16	17	18	19
SW	1	1	1	1	1
reset_n	1	1	1	1	1
PC	0b00000001	0b00000001	0b00000001	0b00000001	0b00000010
Addr_bus	0b00000000	0b00000001	0b00000001	0b00000001	0b00000001
bus_en_n	1	0	0	1	1
RD_n	1	0	0	1	1
data_bus_in	0x0000	0x0000	0x100A	0x100A	0x100A
IR	0x0000	0x0000	0x0000	0x100A	0x100A
REG_0	0x0000	0x0000	0x0000	0x0000	0x0000
next_state	Fetch_1	Fetch_2	Fetch_3	Decode	Execute_LOAD
WE_n	1	1	1	1	1
data_bus_out	0x0000	0x0000	0x0000	0x0000	0x0000
LEDR	0b1111	0b1111	0b1111	0b1111	0b1111

Här hämtas och exekveras instruktionen `LOAD_R0 #A` med op-kod `0x100A` på adressposition 1 i kod-minnet. PC inkrementerades redan under exekveringsfasen av NOP, under cykel 14. `Addr_bus` kan inte adressera kod-minnet under decode-fasen av NOP, eftersom samma adressbuss används för både kod och data.

En hel cykel används (16) till att låta `RD_n` och `bus_en_n` bli stabil, och koppla PC till adressbussen.

Nästa cykel (17) ligger kod-data på kod-bussen, för att läsas in till IR cykeln därpå. Man borde kunna slå ihop cykel 16 och 17, dvs. att `RD_n` och `bus_en_n` bara får vara låga under en cykel och IR läses samtidigt som kod ligger stabil på kodbussen utan något mellansteg.

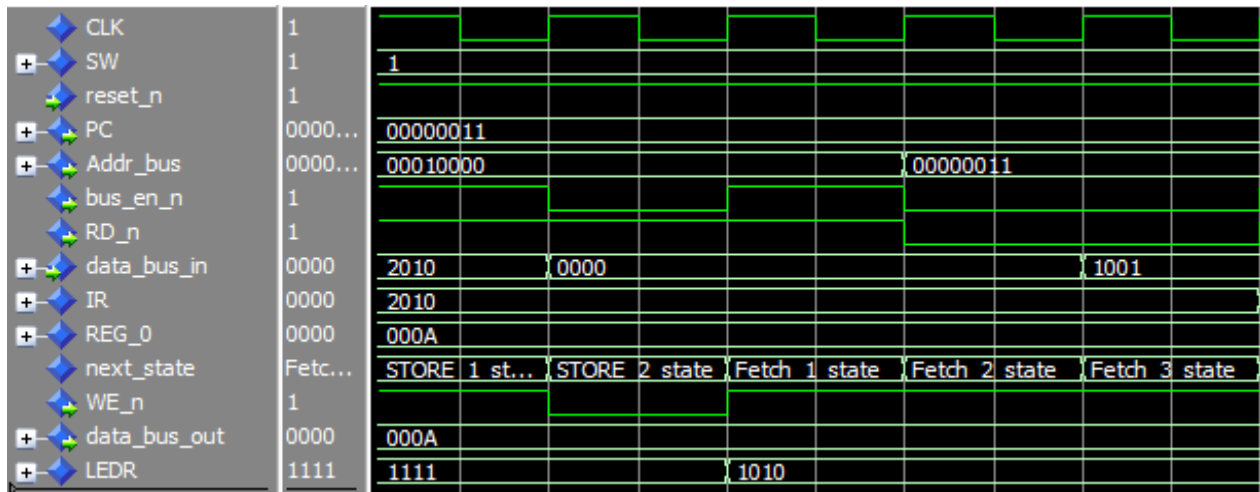
## 2.3 Cykler 20 – 24



CLK_PERIOD	20	21	22	23	24
SW	1	1	1	1	1
reset_n	1	1	1	1	1
PC	0b00000010	0b00000010	0b00000010	0b00000010	0b00000011
Addr_bus	0b00000001	0b00000010	0b00000010	0b00000010	0b00000010
bus_en_n	1	0	0	1	1
RD_n	1	0	0	1	1
data_bus_in	0x100A	0x100A	0x2010	0x2010	0x2010
IR	0x100A	0x100A	0x100A	0x2010	0x2010
REG_0	0x000A	0x000A	0x000A	0x000A	0x000A
next_state	Fetch_1	Fetch_2	Fetch_3	Decode:execute_STORE	
WE_n	1	1	1	1	1
data_bus_out	0x0000	0x0000	0x0000	0x0000	0x0000
LEDR	0b1111	0b1111	0b1111	0b1111	0b1111

Under cykel 20 sker exekvering av LOAD instruktionen där R0 laddas med argumentet 0xA. PC inkrementerades under decode-fasen av LOAD\_R0 #A. Nästa cykel (21) är Fetch\_1, och då är PC kopplad till adressbusesen, och då bus\_en\_n och RD\_n går från låg till hög laddas IR med op-koden 0x2010, vars mnemonic är STORE\_R0 #10. Under decode (cykel 24) inkrementeras PC återigen.

## 2.4 Cykler 25-29

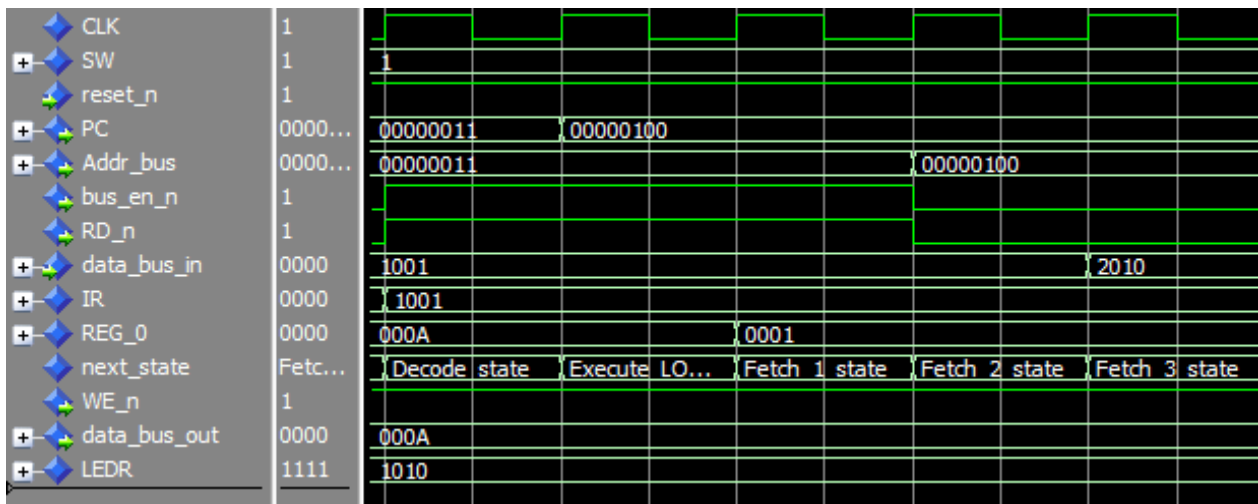


CLK_PERIOD	25	26	27	28	29
SW	1	1	1	1	1
reset_n	1	1	1	1	1
PC	0b00000011	0b00000011	0b00000011	0b00000011	0b00000011
Addr_bus	0b00010000	0b00010000	0b00010000	0b00000011	0b00000011
bus_en_n	1	0	1	0	0
RD_n	1	1	1	0	0
data_bus_in	0x2010	0x0000	0x0000	0x0000	0x1001
IR	0x2010	0x2010	0x2010	0x2010	0x2010
REG_0	0x000A	0x000A	0x000A	0x000A	0x000A
next_state	STORE_1	STORE_2	Fetch_1	Fetch_2	Fetch_3
WE_n	1	0	1	1	1
data_bus_out	0x000A	0x000A	0x000A	0x000A	0x000A
LEDR	0b1111	0b1111	0b1010	0b1010	0b1010

Under Execute\_STORE (cykel 25) kopplas IR:s argument 0x10 till Adressbussen och Reg\_0 läggs ut på data\_bussen. Under cykel 26 görs data giltig genom att WE\_n går låg. Senast cykel 27 lyser dioderna enligt bitmönstret 0b1010 (0xA).

PC har varit inkrementerad sedan flera klockcykler tidigare. Under Fetch\_1 (cykel 28) koppas PC ut till Adressbussen och bus\_en\_n och RD\_n går då återigen samtidigt låga. IR kommer laddas under Fetch\_3 (cykel 30, se nästa avsnitt).

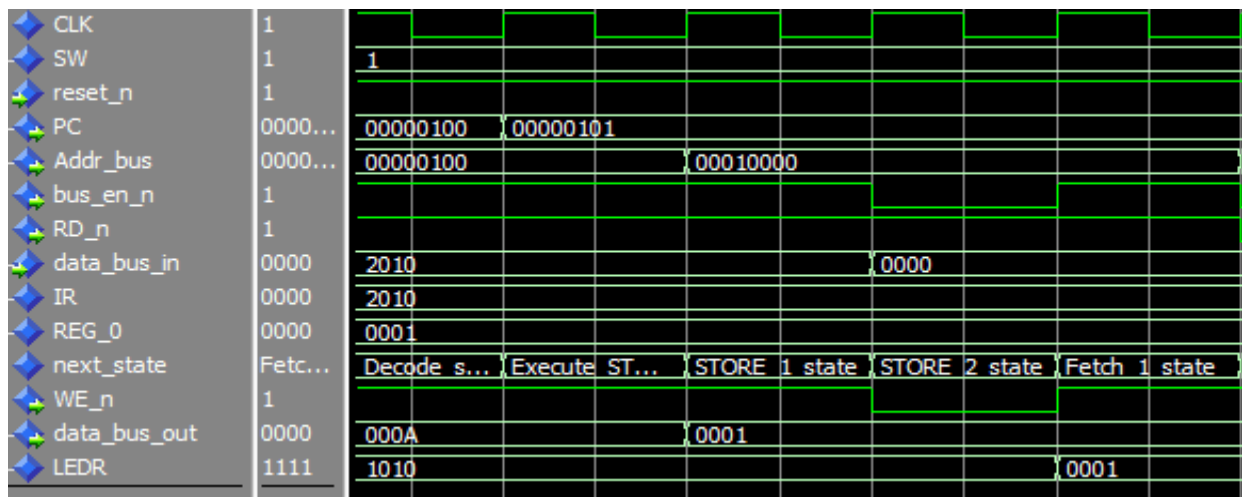
## 2.5 Cykler 30 – 34



CLK_PERIOD	30	31	32	33	34
SW	1	1	1	1	1
reset_n	1	1	1	1	1
PC	0b00000011	0b00000100	0b00000100	0b00000100	0b000001000
Addr_bus	0b00000011	0b00000011	0b00000011	0b00000100	0b000001000
bus_en_n	1	1	1	0	0
RD_n	1	1	1	0	0
data_bus_in	0x1001	0x1001	0x1001	0x1001	0x2010
IR	0x1001	0x1001	0x1001	0x1001	0x1001
REG_0	0x000A	0x000A	0x0001	0x0001	0x0001
next_state	Decode	Execute_LOAD	Fetch_1	Fetch_2	Fetch_3
WE_n	1	1	1	1	1
data_bus_out	0x000A	0x000A	0x000A	0x000A	0x000A
LEDR	0b1010	0b1010	0b1010	0b1010	0b1010

Op-koden som IR håller är 0x1001 vars mnemonic är `LOAD_R0 #1`. Under decode (cykel 31) inkrementeras PC. Under Execute\_LOAD (cykel 32) laddas Reg\_0 med OP-kodens argument. Under Fetch\_1 (cykel 33) kopplas PC till adressbussen bus\_en\_n och RD\_n går då låga. Nästa instruktion som är resultatet av adresseringen av kod-minnet läggs ut på kod-bussen under cykel 34, och läses in till IR under cykel 36.

## 2.6 Cykler 35-39

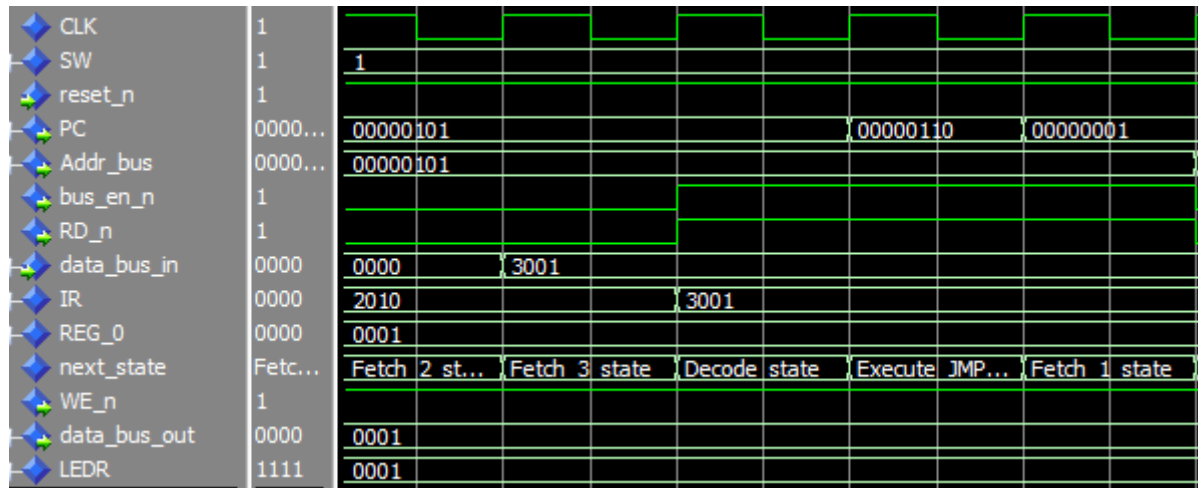


CLK_PERIOD	35	36	37	38	39
SW	1	1	1	1	1
reset_n	1	1	1	1	1
PC	0b00000100	0b00000101	0b00000101	0b00000101	0b00000101
Addr_bus	0b00000100	0b00000100	0b00010000	0b00010000	0b00010000
bus_en_n	1	1	1	0	1
RD_n	1	1	1	1	1
data_bus_in	0x2010	0x2010	0x2010	0x0000	0x0000
IR	0x2010	0x2010	0x2010	0x2010	0x2010
REG_0	0x0001	0x0001	0x0001	0x0001	0x0001
next_state	Decode	Execute_STORE	STORE_1	STORE_2	Fetch_1
WE_n	1	1	1	0	1
data_bus_out	0x000A	0x000A	0x0001	0x0001	0x0001
LEDR	0b1010	0b1010	0b1010	0b1010	0b0001

Inläsningen till IR innebär att op-koden 0x2010, vilket är menomic för STORE\_R0 #10.

Under deode inkrementeras PC (cykel 36), och Execute\_STORE görs under cykel 37. Då kopplas IR:s argument till adressbussen och REG\_0 läggs ut på databussen. WE\_n går låg under följande cykel (38) och senast cykel 39 lyser lysdioderna i enlighet med vad som lades ut på databussen dvs. 0b0001.

## 2.7 Cykler 40 – 44

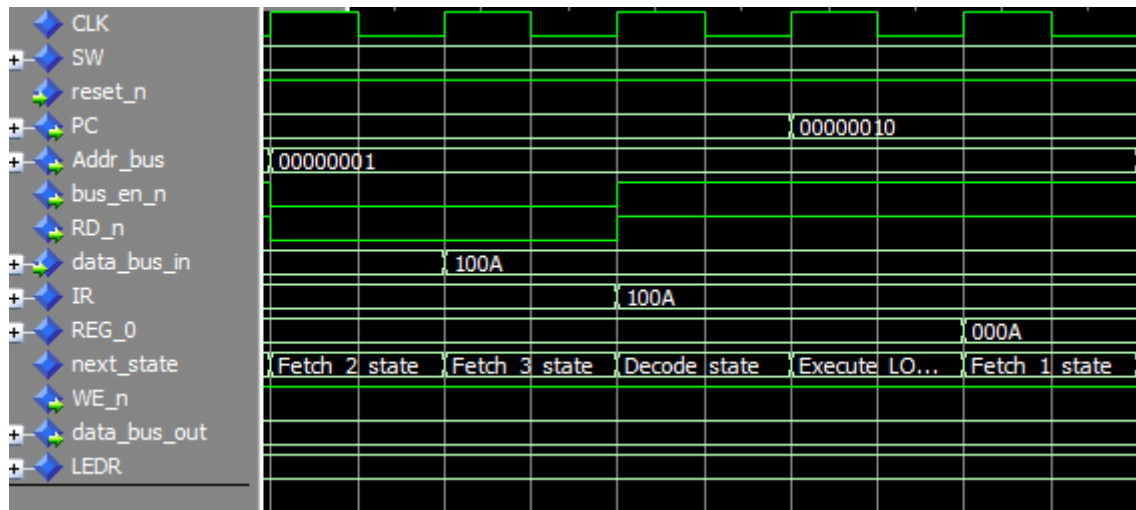


CLK_PERIOD	40	41	42	43	44
SW	1	1	1	1	1
reset_n	1	1	1	1	1
PC	0b00000101	0b00000101	0b00000101	0b00000110	0b00000001
Addr_bus	0b00000101	0b00000101	0b00000101	0b00000101	0b00000101
bus_en_n	0	0	1	1	1
RD_n	0	0	1	1	1
data_bus_in	0x0000	0x3001	0x3001	0x3001	0x3001
IR	0x2010	0x2010	0x3001	0x3001	0x3001
REG_0	0x0001	0x0001	0x0001	0x0001	0x0001
next_state	Fetch_2	Fetch_3	Decode	Execute_JMP	Fetch_1
WE_n	1	1	1	1	1
data_bus_out	0x0001	0x0001	0x0001	0x0001	0x0001
LEDR	0b0001	0b0001	0b0001	0b0001	0b0001

Under Fetch\_1 (cykel 40) kopplas PC till adressbussen och bus\_en\_n och RD\_n går båda låga. Nästa instruktion, dvs. den på adress adresserad av PC finns på kodbussen under cykel 41, och inläsning till IR sker under nästa cykel (42). Op koden är 0x3001 vars mnemonic är JMP #1. Under decode inkrementeras PC (cykel 43), men under Execute\_JMP (cykel 44) laddas PC med instruktionens argument dvs. IR:s argument-del kopplas till PC.



## 2.8 Cykler 45-49



CLK_PERIOD	45	46	47	48	49
SW	1	1	1	1	1
reset_n	1	1	1	1	1
PC	0b00000001	0b00000001	0b00000001	0b00000010	0b00000010
Addr_bus	0b00000001	0b00000001	0b00000001	0b00000001	0b00000001
bus_en_n	0	0	1	1	1
RD_n	0	0	1	1	1
data_bus_in	0x3001	0x100A	0x100A	0x100A	0x100A
IR	0x3001	0x3001	0x100A	0x100A	0x0000
REG_0	0x0001	0x0001	0x0001	0x0001	0x000A
next_state	Fetch_2	Fetch_3	Decode	Execute_LOAD	Fetch_1
WE_n	1	1	1	1	1
data_bus_out	0x0001	0x0001	0x0001	0x0001	0x0001
LEDR	0b0001	0b0001	0b0001	0b0001	0b0001

Under Fetch\_1 (cykel 45) kopplas PC till Adressbussen och bus\_en\_n och RD\_n går båda låga samtidigt. Instruktions som läses in till IR (cykel 47) är från adress 1 i kodminnet LOAD\_R0 #A med OP-kod 0x100A, som exekverats förut.

Under decode (cykel 48) inkrementeras PC, och under Execute\_LOAD (cykel 49) har Reg\_0 laddats med argument delen av IR.