

# Sistemas Reconfiguráveis

## Eng. de Computação

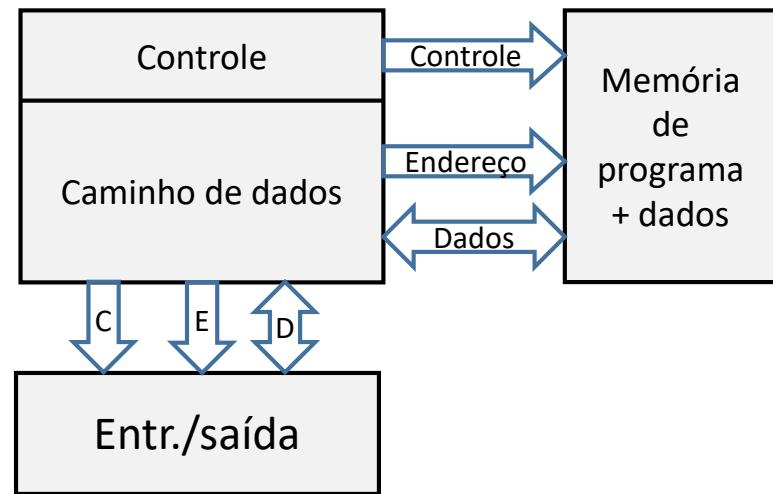
Profs. Francisco Garcia e Antônio Hamilton Magalhães

Aula 13 – Arquiteturas de processadores

# Arquitetura de processadores

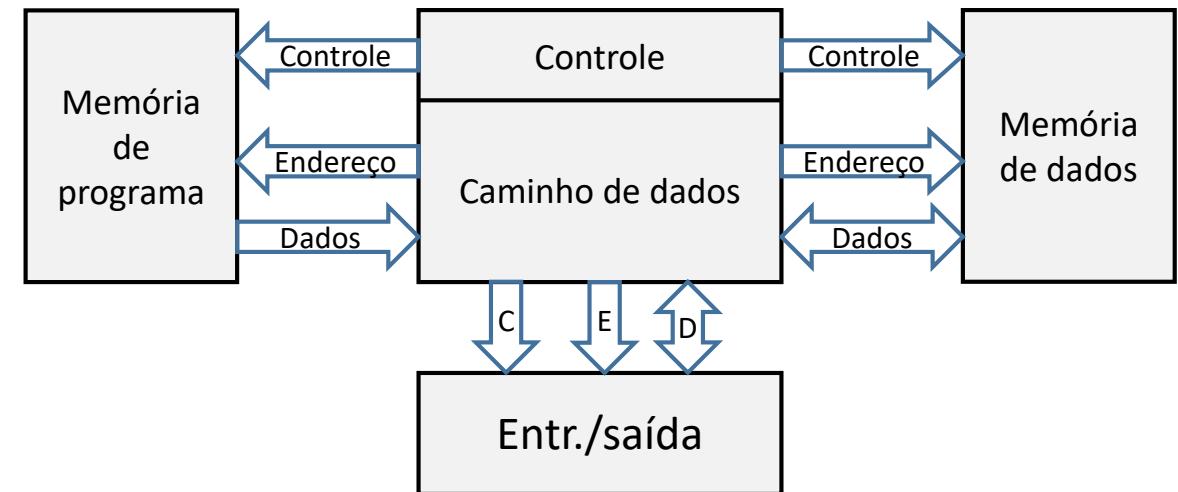
## Arquitetura von Neumann

Memória única para programa e dados



## Arquitetura Harvard

Memória de programa separada da memória de dados

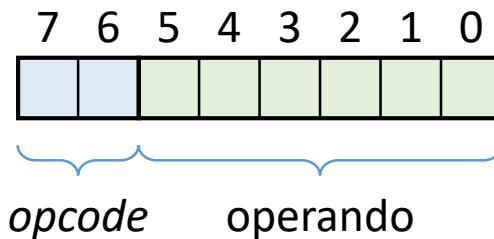


# Exemplo: CPU somadora

CPU somadora - ISA: *instruction set architecture*

Palavra de 8 bits

Quatro instruções

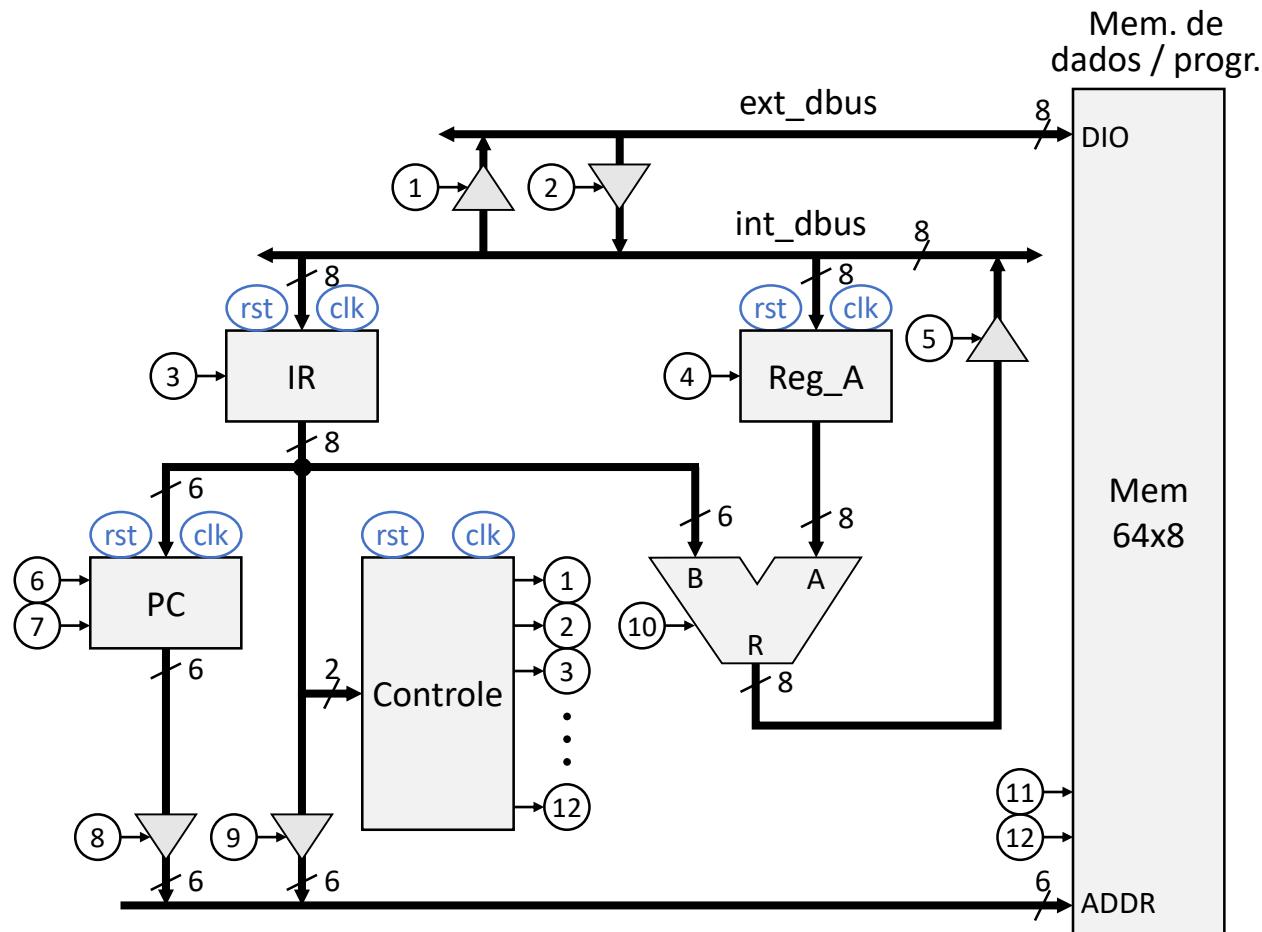


Opcode	Função	Mnem.	Operando	Descrição
00	Carrega	LD	Endereço	Carrega o conteúdo do endereço da memória apontado pelo operando no registrador A
01	Armazena	STO	Endereço	Carrega o conteúdo do registrador A no endereço da memória apontado pelo operando
10	Soma	ADD	Valor imediato	Soma o valor presente no registrador A com o operando, salvando o resultado no próprio registrador A
11	Desvia	JMP	Endereço	Desvia a execução do programa para o endereço especificado pelo operando

# Exemplo: CPU somadora

## Arquitetura von Neumann

CPU somadora - Arquitetura interna (von Neumann)



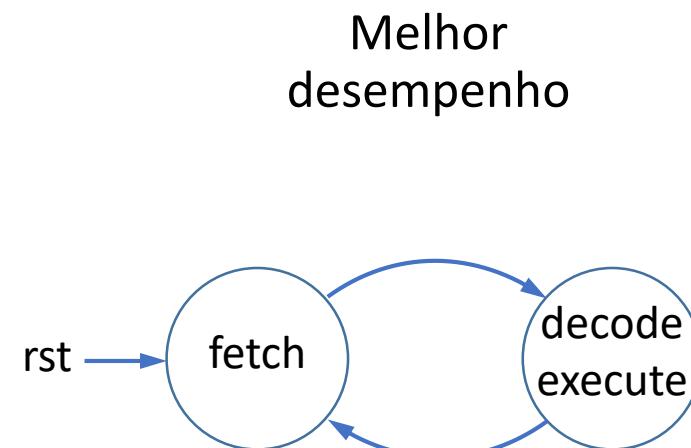
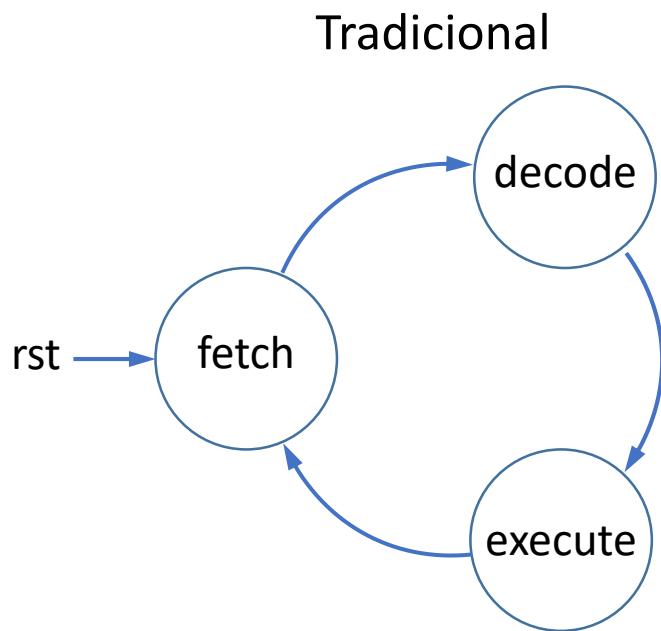
Sinal	Nome
1	dint_on_dext
2	dext_on_dint
3	wr_ir
4	wr_rega
5	au_on_dint
6	inc_pc
7	load_pc
8	pc_on_addr
9	ir_on_addr
10	au_add
11	rd_mem
12	wr_mem

Todos os sinais ativos em nível alto, exceto  
au\_add: '1' => add, '0' => pass\_B

# Exemplo: CPU somadora

## Arquitetura von Neumann

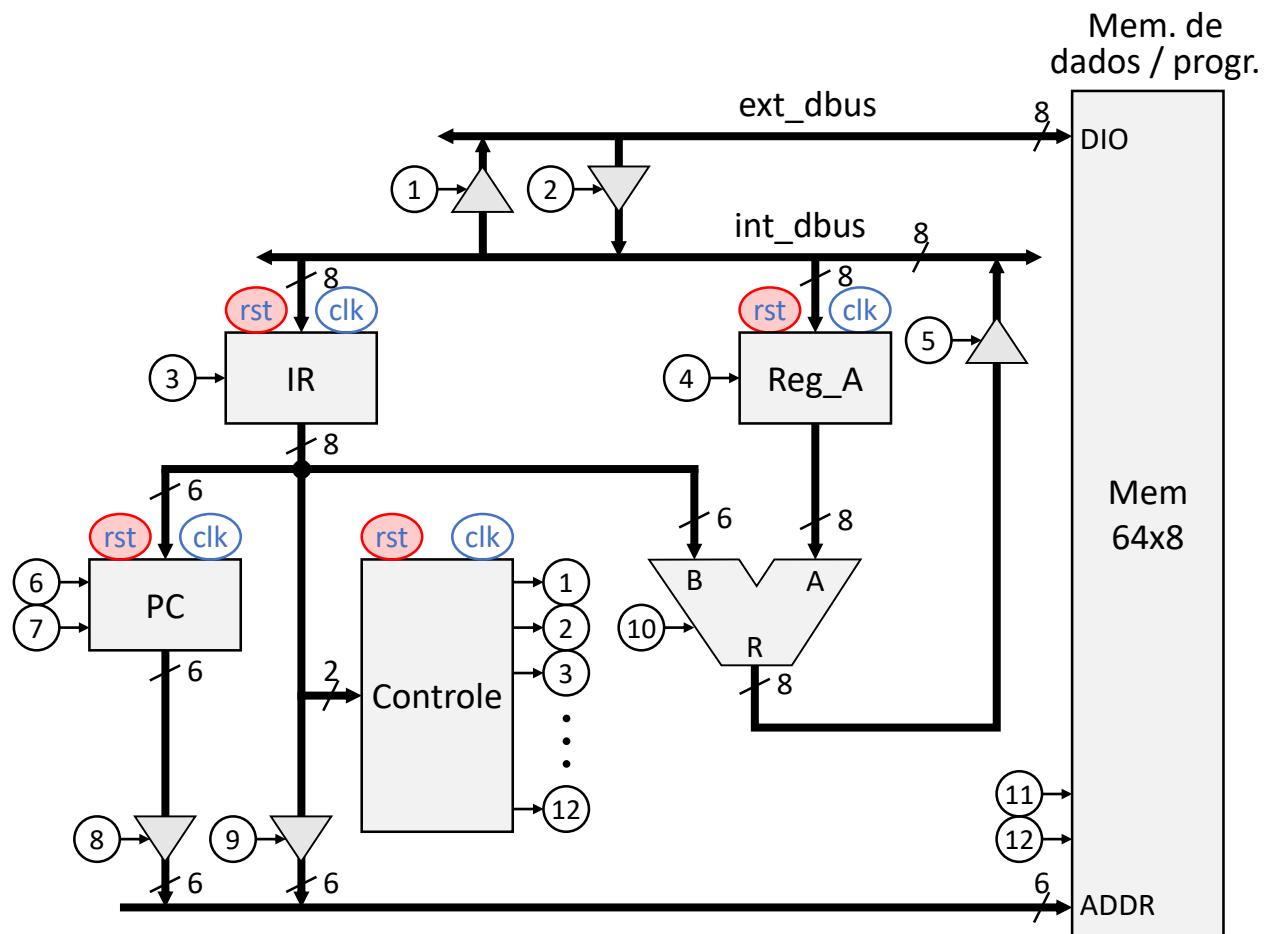
CPU – Estados da máquina de controle



# Exemplo: CPU somadora

## Arquitetura von Neumann

CPU somadora – Sinais de controle



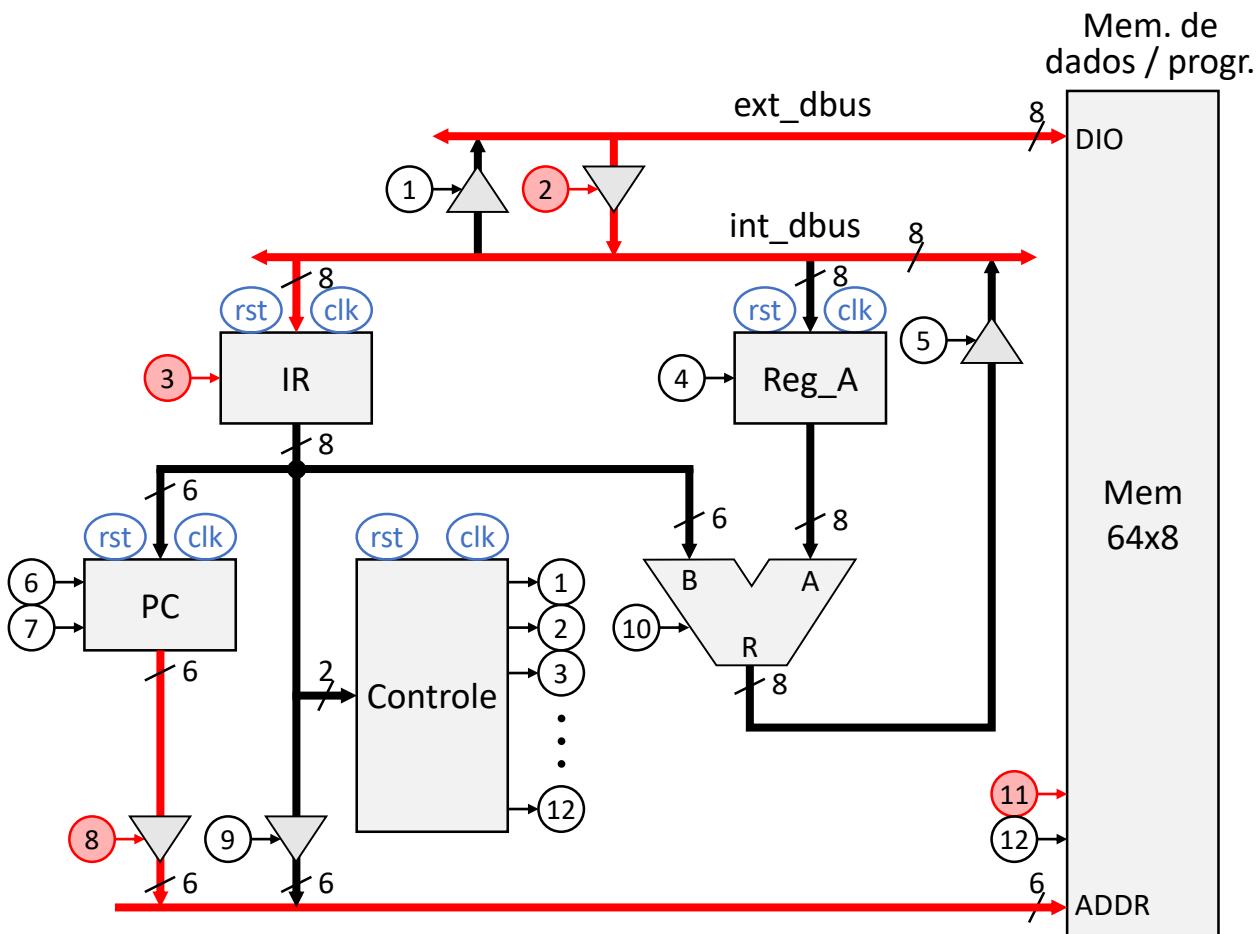
A black arrow points from the "ext\_dbus" connection in the diagram to the "dint\_on\_dext" row in the table below, indicating the mapping of external data bus operations to internal control signals.

Sinal	Nome	reset	fetch	decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	1	1	0	0	0
3	wr_ir	0	1	0	0	0	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	0	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	pc_on_addr	0	1	0	0	0	0
9	ir_on_addr	0	0	1	1	0	0
10	au_add	X	X	X	0	1	X
11	rd_mem	0	1	1	0	0	0
12	wr_mem	0	0	0	1	0	0

# Exemplo: CPU somadora

## Arquitetura von Neumann

CPU somadora – Sinais de controle

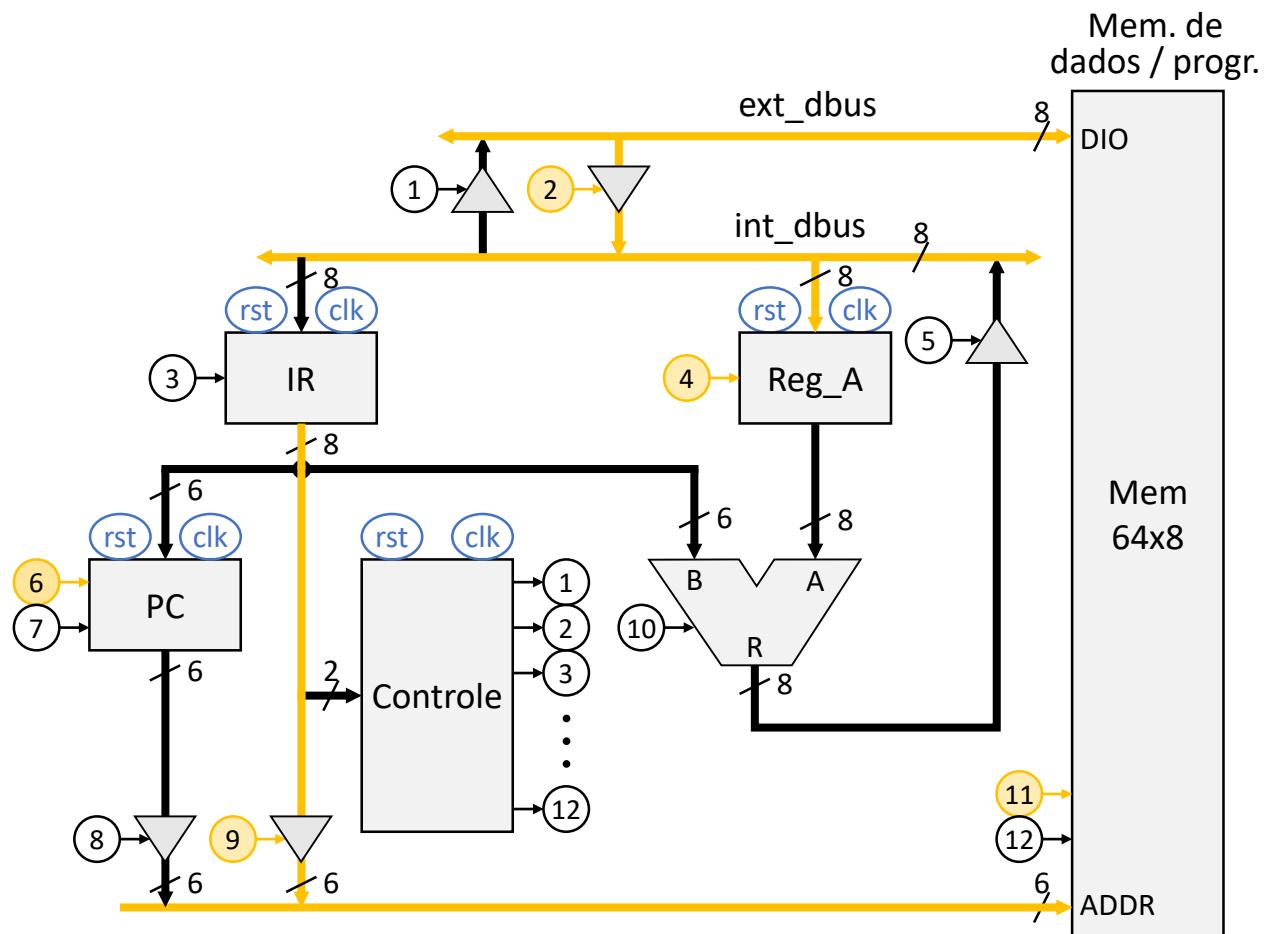


Sinal	Nome	reset	fetch	decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	1	1	0	0	0
3	wr_ir	0	1	0	0	0	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	0	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	pc_on_addr	0	1	0	0	0	0
9	ir_on_addr	0	0	1	1	0	0
10	au_add	X	X	X	0	1	X
11	rd_mem	0	1	1	0	0	0
12	wr_mem	0	0	0	1	0	0

# Exemplo: CPU somadora

## Arquitetura von Neumann

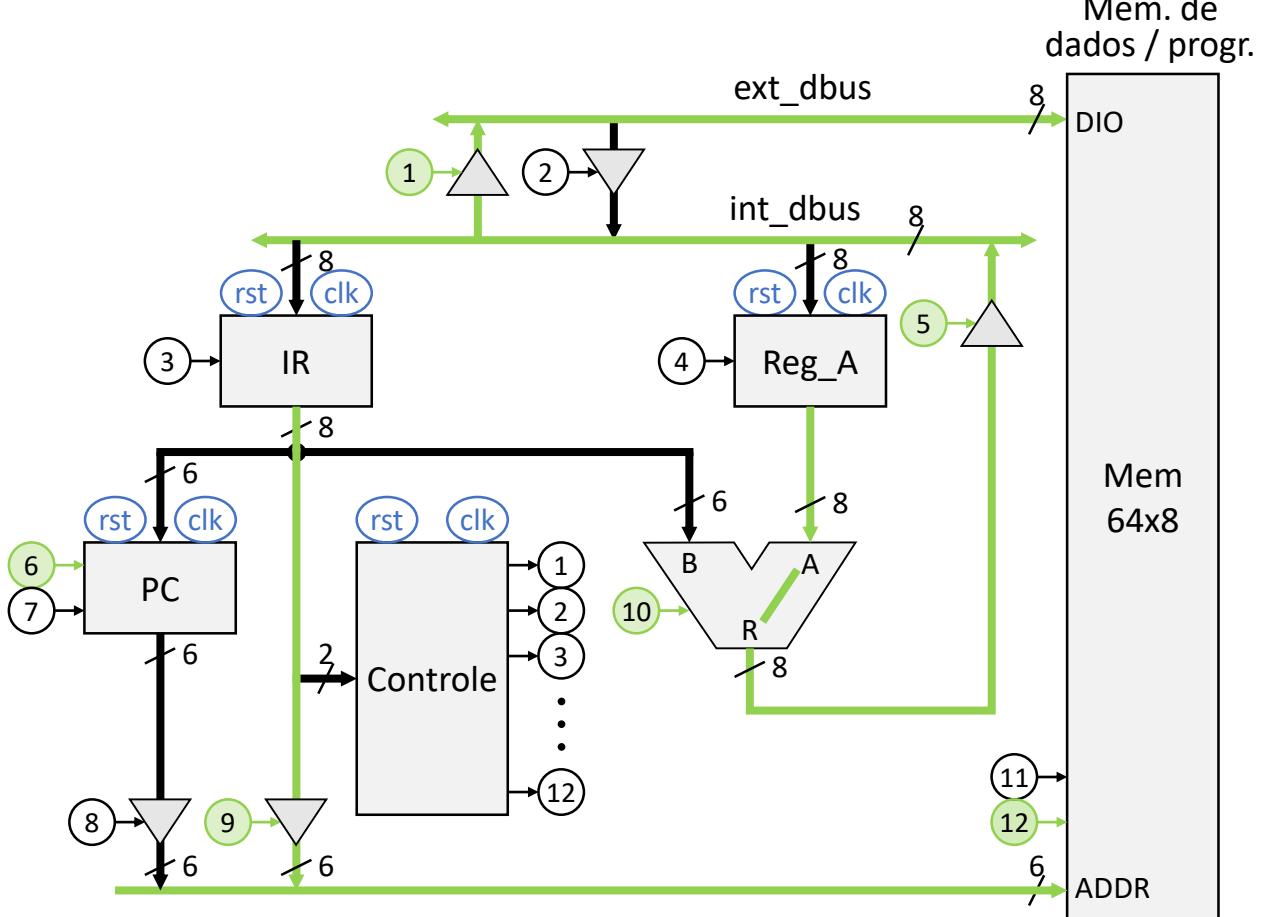
CPU somadora – Sinais de controle



Sinal	Nome	reset	fetch	decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	1	1	0	0	0
3	wr_ir	0	1	0	0	0	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	0	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	pc_on_addr	0	1	0	0	0	0
9	ir_on_addr	0	0	1	1	0	0
10	au_add	X	X	X	0	1	X
11	rd_mem	0	1	1	0	0	0
12	wr_mem	0	0	0	1	0	0

# Exemplo: CPU somadora

## Arquitetura von Neumann

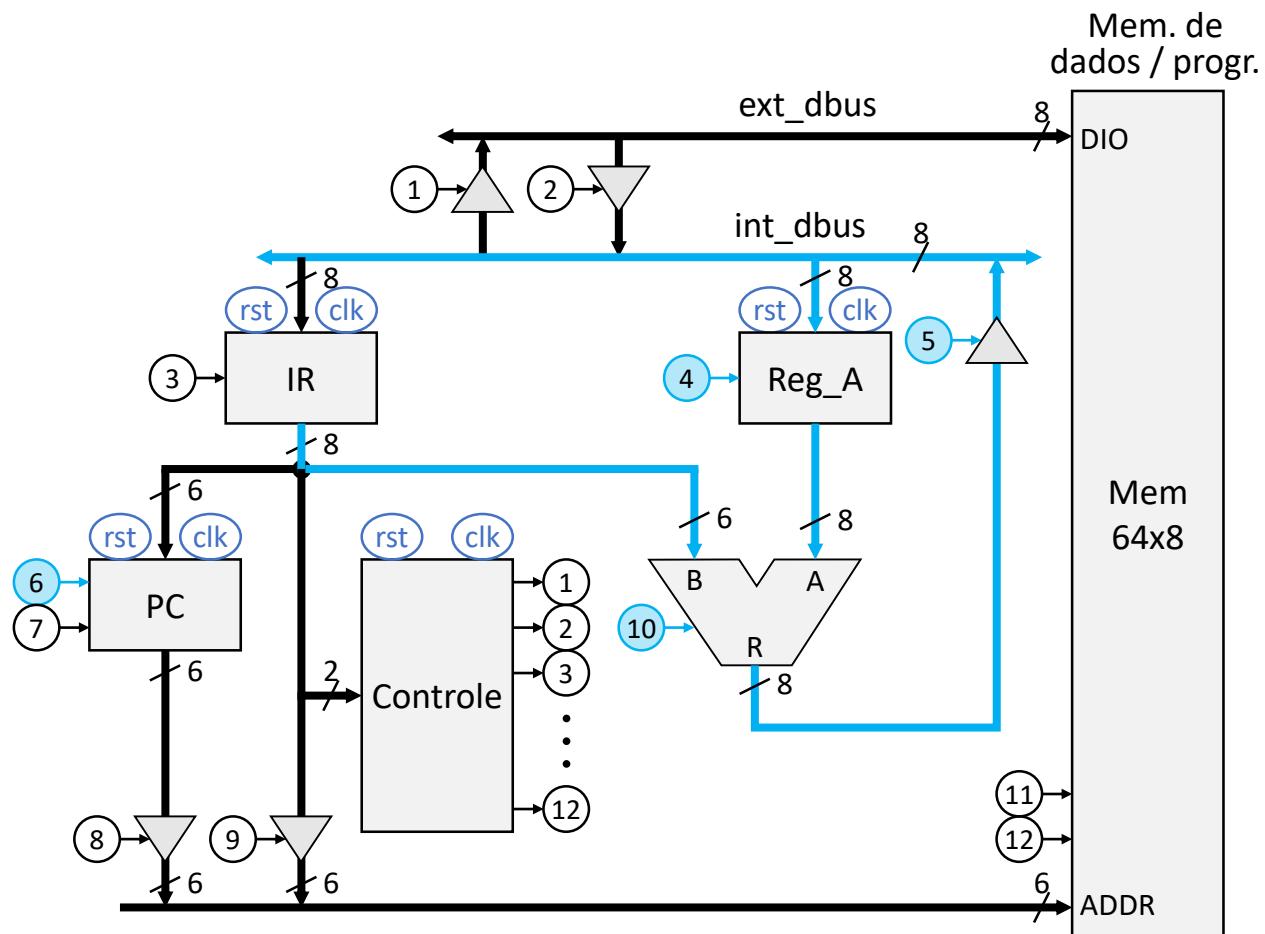


Sinal	Nome	reset	fetch	decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	1	1	0	0	0
3	wr_ir	0	1	0	0	0	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	0	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	pc_on_addr	0	1	0	0	0	0
9	ir_on_addr	0	0	1	1	0	0
10	au_add	X	X	X	0	1	X
11	rd_mem	0	1	1	0	0	0
12	wr_mem	0	0	0	1	0	0

# Exemplo: CPU somadora

## Arquitetura von Neumann

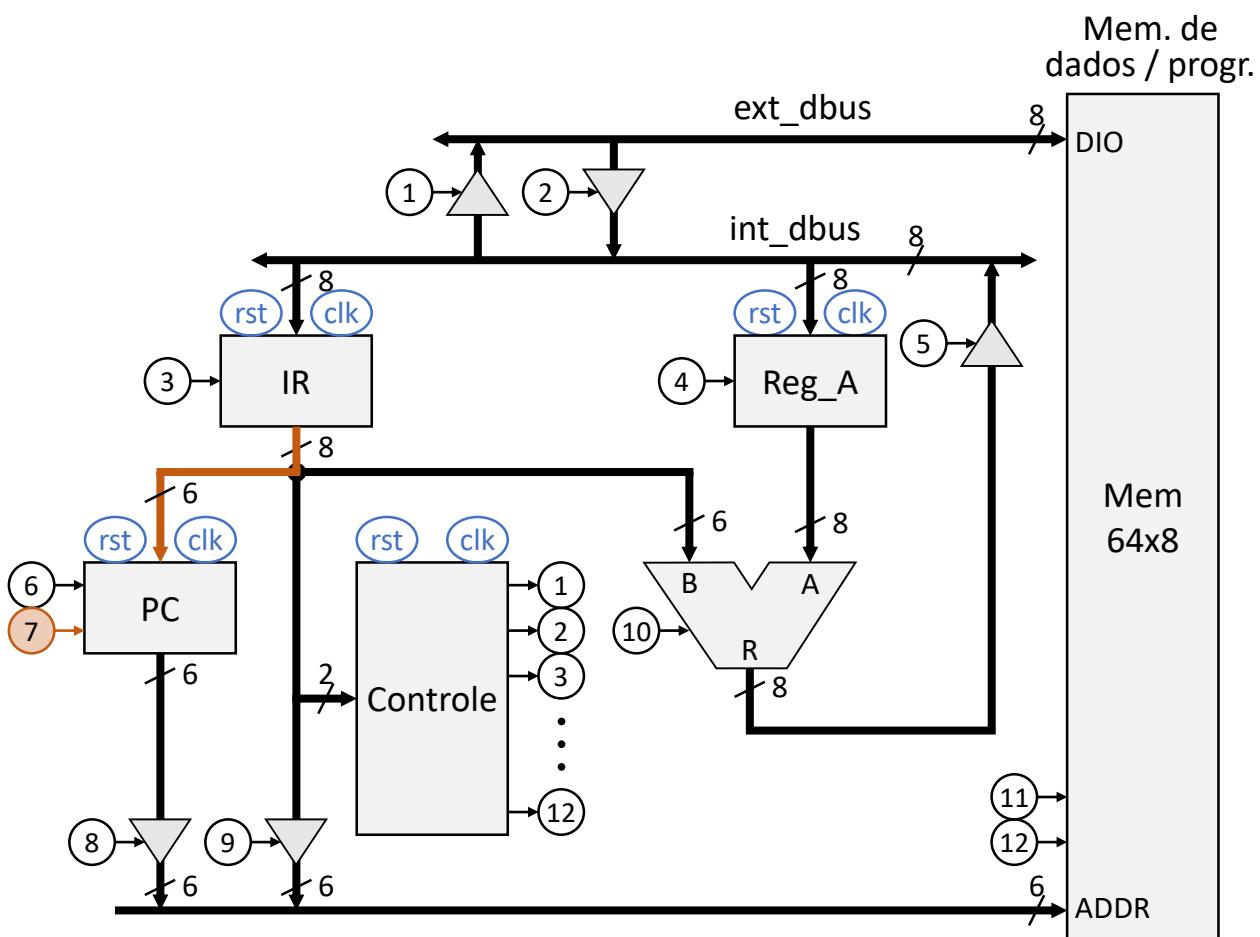
CPU somadora – Sinais de controle



Sinal	Nome	reset	fetch	decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	1	1	0	0	0
3	wr_ir	0	1	0	0	0	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	0	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	pc_on_addr	0	1	0	0	0	0
9	ir_on_addr	0	0	1	1	0	0
10	au_add	X	X	X	0	1	X
11	rd_mem	0	1	1	0	0	0
12	wr_mem	0	0	0	1	0	0

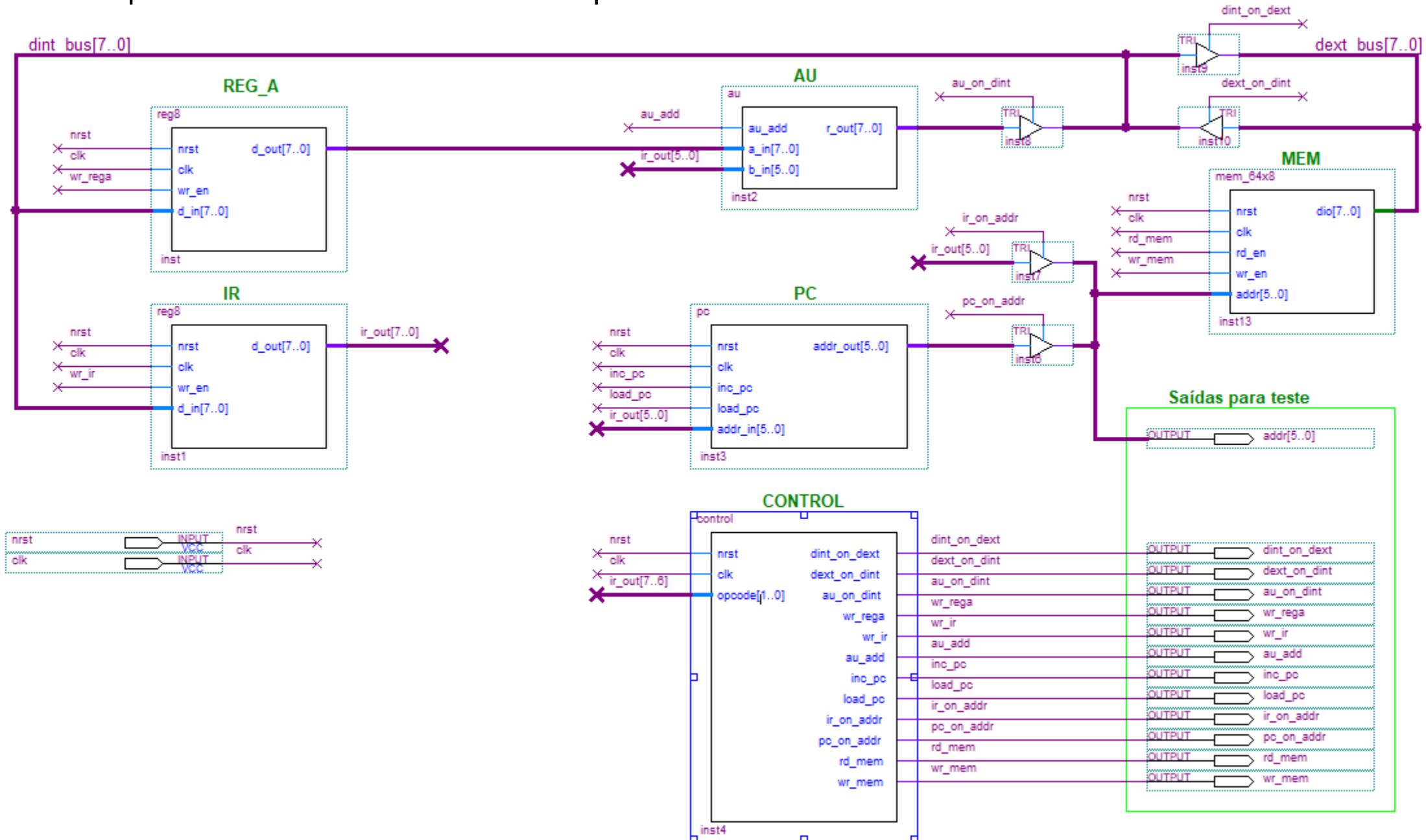
# Exemplo: CPU somadora

## Arquitetura von Neumann



Sinal	Nome	reset	fetch	decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	1	1	0	0	0
3	wr_ir	0	1	0	0	0	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	0	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	pc_on_addr	0	1	0	0	0	0
9	ir_on_addr	0	0	1	1	0	0
10	au_add	X	X	X	0	1	X
11	rd_mem	0	1	1	0	0	0
12	wr_mem	0	0	0	1	0	0

# Exemplo: CPU somadora - Arquitetura von Neumann

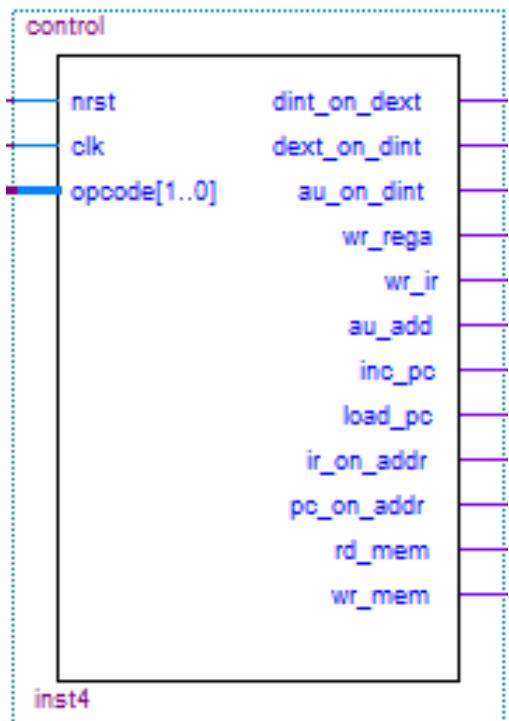


# Exemplo: CPU somadora

## Arquitetura von Neumann

### CPU somadora

#### Módulo de controle



```
library ieee;
use ieee.std_logic_1164.all;

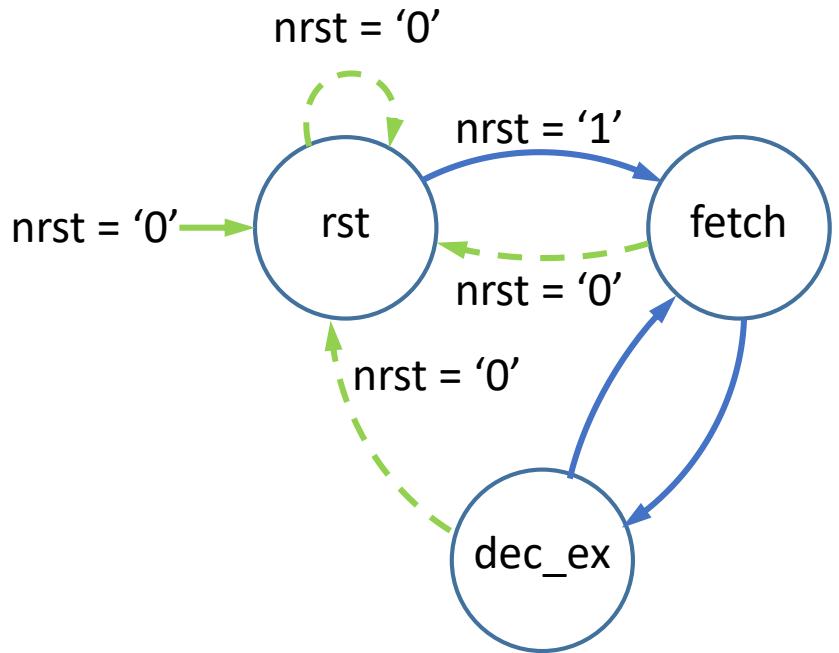
entity control is
  port(
    nrst          : in std_logic;
    clk           : in std_logic;
    opcode        : in std_logic_vector(1 downto 0);
    dint_on_dext : out std_logic;
    dext_on_dint : out std_logic;
    au_on_dint   : out std_logic;
    wr_rega      : out std_logic;
    wr_ir        : out std_logic;
    au_add       : out std_logic;
    inc_pc       : out std_logic;
    load_pc      : out std_logic;
    ir_on_addr   : out std_logic;
    pc_on_addr   : out std_logic;
    rd_mem       : out std_logic;
    wr_mem       : out std_logic
  );
end entity;
```

Continua na próxima página

# Exemplo: CPU somadora

## Arquitetura von Neumann

### CPU somadora Módulo de controle



Obs.: A entrada nrst estabelece a condição inicial da máquina de estado através de uma entrada assíncrona, que é prioritária.

```
architecture arch of control is
    type state_type is (rst, fetch, dec_ex);
    signal pres_state : state_type;
    signal next_state : state_type;

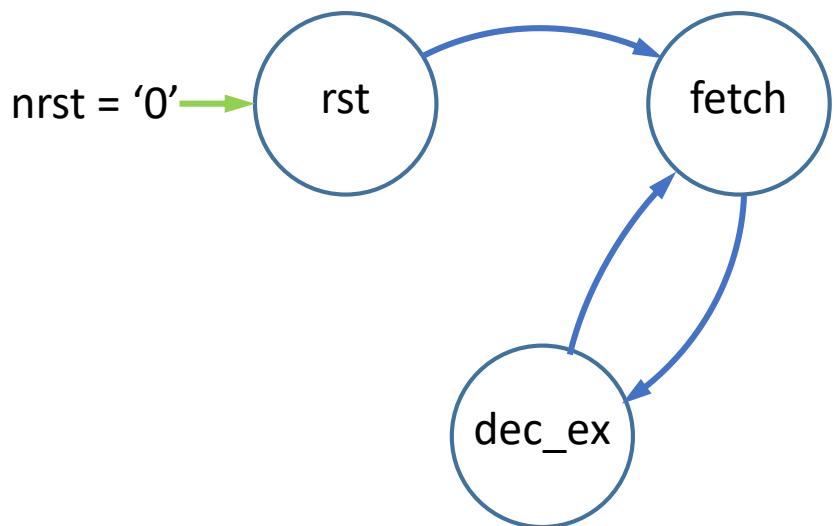
begin
    -----
    -- parte sequencial da FSM
    -----
    process(nrst, clk)
    begin
        if nrst = '0' then
            pres_state <= rst;
        elsif rising_edge(clk) then
            pres_state <= next_state;
        end if;
    end process;
```

Continua na próxima página

# Exemplo: CPU somadora

## Arquitetura von Neumann

### CPU somadora Módulo de controle



```
-- parte combinacional da FSM
-----
process(nrst, pres_state, opcode)
begin
    -- valores default:
    dint_on_dext      <= '0';
    dext_on_dint      <= '0';
    au_on_dint        <= '0';
    wr_rega           <= '0';
    wr_ir             <= '0';
    au_add            <= '-';
    inc_pc            <= '0';
    load_pc           <= '0';
    ir_on_addr        <= '0';
    pc_on_addr        <= '0';
    rd_mem            <= '0';
    wr_mem            <= '0';
```



# Exemplo: CPU somadora

## Arquitetura von Neumann

### CPU somadora Módulo de controle

Sinal	Nome	reset	fetch	decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	1	1	0	0	0
3	wr_ir	0	1	0	0	0	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	0	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	pc_on_addr	0	1	0	0	0	0
9	ir_on_addr	0	0	1	1	0	0
10	au_add	X	X	X	0	1	X
11	rd_mem	0	1	1	0	0	0
12	wr_mem	0	0	0	1	0	0

```
case pres_state is
    when rst =>
        next_state <= fetch;

    when fetch =>
        next_state <= dec_ex;

        dext_on_dint      <= '1';
        wr_ir             <= '1';
        pc_on_addr        <= '1';
        rd_mem            <= '1';
```

# Exemplo: CPU somadora

## Arquitetura von Neumann

### CPU somadora

#### Módulo de controle

Sinal	Nome	reset	fetch	decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	1	1	0	0	0
3	wr_ir	0	1	0	0	0	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	0	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	pc_on_addr	0	1	0	0	0	0
9	ir_on_addr	0	0	1	1	0	0
10	au_add	X	X	X	0	1	X
11	rd_mem	0	1	1	0	0	0
12	wr_mem	0	0	0	1	0	0

```
when dec_ex =>
    next_state <= fetch;

case opcode is
-----
-- load
-----
when "00" =>
    dext_on_dint      <= '1';
    wr_rega          <= '1';
    inc_pc            <= '1';
    ir_on_addr        <= '1';
    rd_mem            <= '1';
```

# Exemplo: CPU somadora

## Arquitetura von Neumann

### CPU somadora Módulo de controle

Sinal	Nome	reset	fetch	decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	1	1	0	0	0
3	wr_ir	0	1	0	0	0	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	0	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	pc_on_addr	0	1	0	0	0	0
9	ir_on_addr	0	0	1	1	0	0
10	au_add	X	X	X	0	1	X
11	rd_mem	0	1	1	0	0	0
12	wr_mem	0	0	0	1	0	0

```
-- store
when "01" =>
    dint_on_dext      <= '1';
    au_on_dint        <= '1';
    au_add            <= '0';
    inc_pc            <= '1';
    ir_on_addr        <= '1';
    wr_mem            <= '1';

-- add
when "10" =>
    au_on_dint        <= '1';
    wr_rega           <= '1';
    au_add            <= '1';
    inc_pc            <= '1';
```

# Exemplo: CPU somadora

## Arquitetura von Neumann

### CPU somadora Módulo de controle

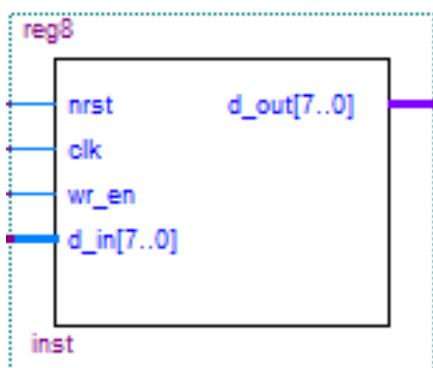
Sinal	Nome	reset	fetch	decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	1	1	0	0	0
3	wr_ir	0	1	0	0	0	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	0	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	pc_on_addr	0	1	0	0	0	0
9	ir_on_addr	0	0	1	1	0	0
10	au_add	X	X	X	0	1	X
11	rd_mem	0	1	1	0	0	0
12	wr_mem	0	0	0	1	0	0

```
-- jump
-----
when "11" =>
    load_pc      <= '1';
end case;
end case;
end process;
end arch;
```

# Exemplo: CPU somadora

## Arquitetura von Neumann

CPU somadora  
IR e Reg\_A



Esse bloco é  
instanciado  
duas vezes

```
library ieee;
use ieee.std_logic_1164.all;

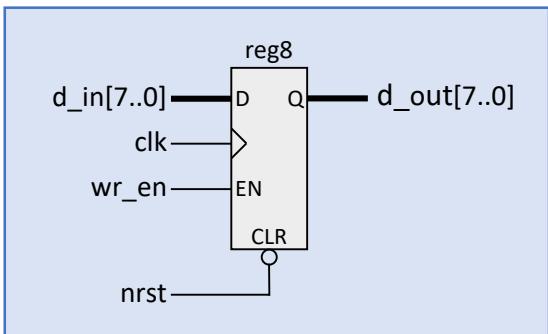
entity reg8 is
  port(
    nrst      : in std_logic;
    clk       : in std_logic;
    wr_en    : in std_logic;
    d_in     : in std_logic_vector(7 downto 0);
    d_out    : out std_logic_vector(7 downto 0)
  );
end entity;
```

Continua na próxima página

# Exemplo: CPU somadora

## Arquitetura von Neumann

CPU somadora  
IR e Reg\_A



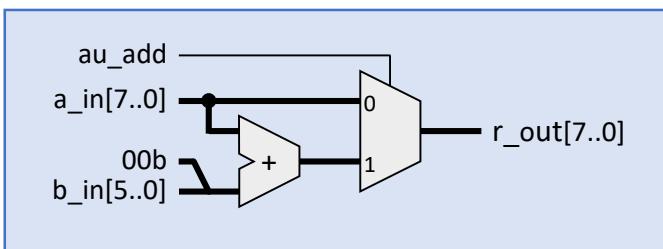
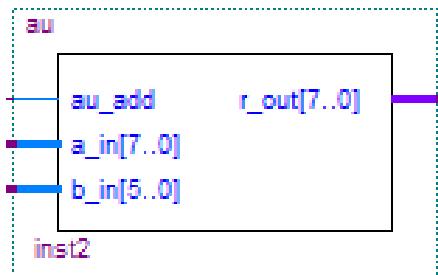
```
architecture arch of reg8 is
    signal reg      : std_logic_vector(7 downto 0);
begin
    process(nrst, clk)
    begin
        if nrst = '0' then
            reg <= (others => '0');
        elsif rising_edge(clk) then
            if wr_en = '1' then
                -- escreve no registrador
                reg <= d_in;
            else
                -- caso contrário permanece como está
            end if;
        end if;
    end process;
    d_out <= reg;
end arch;
```

Continua na próxima página

# Exemplo: CPU somadora

## Arquitetura von Neumann

CPU somadora  
Unidade aritmética



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity au is
    port(
        au_add      : in std_logic;
        a_in        : in std_logic_vector(7 downto 0);
        b_in        : in std_logic_vector(5 downto 0);
        r_out       : out std_logic_vector(7 downto 0)
    );
end entity;

architecture arch of au is
begin

    r_out <= a_in when au_add = '0' else
                a_in + ("00" & b_in);

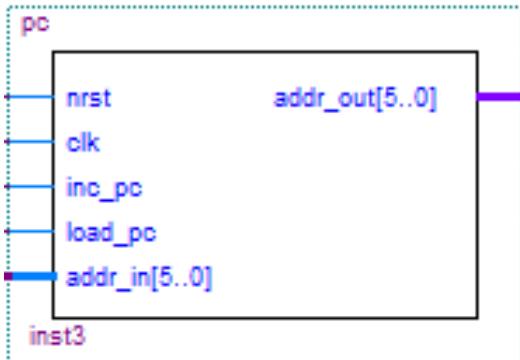
end arch;
```

Os vetores  
têm de ter o  
mesmo  
comprimento

# Exemplo: CPU somadora

## Arquitetura von Neumann

CPU somadora  
Program counter



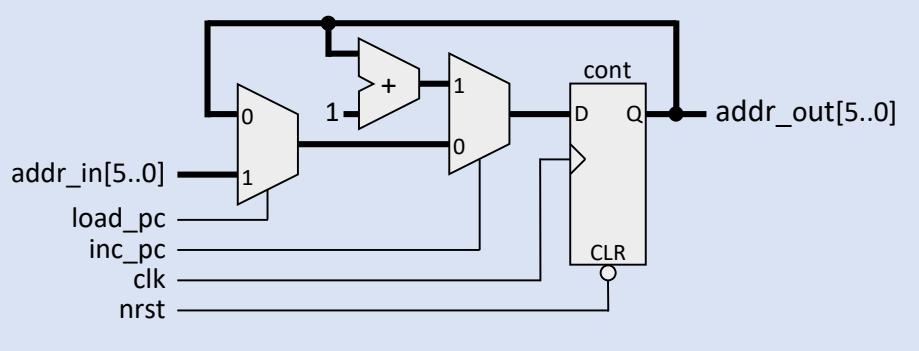
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity pc is
    port(
        nrst      : in std_logic;
        clk       : in std_logic;
        inc_pc    : in std_logic;
        load_pc   : in std_logic;
        addr_in   : in std_logic_vector(5 downto 0);
        addr_out  : out std_logic_vector(5 downto 0)
    );
end entity;
```

# Exemplo: CPU somadora

## Arquitetura von Neumann

CPU somadora  
Program counter



```
architecture arch of pc is
    signal cont : std_logic_vector(5 downto 0);
begin
    process(nrst, clk)
    begin
        if nrst = '0' then
            cont <= (others => '0');
        elsif rising_edge(clk) then
            if inc_pc = '1' then
                -- incrementa contador
                cont <= cont + 1;
            elsif load_pc = '1' then
                -- carrega contador com novo endereço
                cont <= addr_in;
            end if;
        end if;
    end process;
    addr_out <= cont;
end arch;
```

# Exemplo: CPU somadora

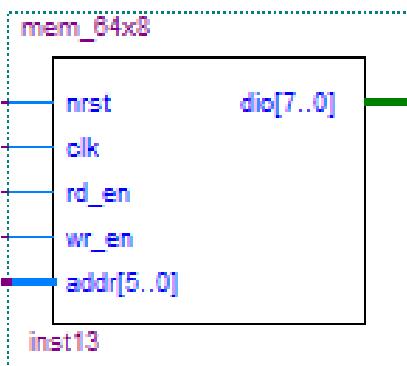
## Arquitetura von Neumann

CPU somadora

Memória

0..15 ROM

16..63 RAM



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mem_64x8 is
    port(
        nrst      : in std_logic;
        clk       : in std_logic;
        rd_en     : in std_logic;
        wr_en     : in std_logic;
        addr      : in std_logic_vector(5 downto 0);
        dio       : inout std_logic_vector(7 downto 0)
    );
end entity;
```

Para conversão de tipos

# Exemplo: CPU somadora

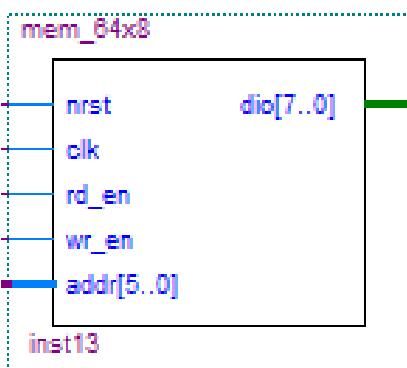
## Arquitetura von Neumann

CPU somadora

Memória

0..15 ROM

16..63 RAM



```
architecture arch of mem_64x8 is
  type ram_type is array (16 to 63) of
    std_logic_vector(7 downto 0);
  signal ram : ram_type;
  signal addr_int : integer range 0 to 63;
begin
  addr_int <= to_integer(unsigned(addr));
```

# Exemplo: CPU somadora

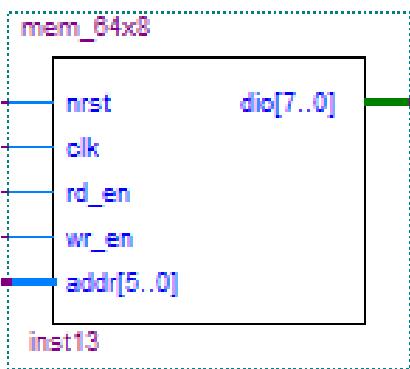
## Arquitetura von Neumann

CPU somadora

Memória

0..15 ROM

16..63 RAM



```
-- escreve na ram  
-----  
process(nrst, clk)  
begin  
    if nrst = '0' then  
        for i in 16 to 63 loop  
            ram(i) <= (others => '0');  
        end loop;  
    elsif rising_edge(clk) then  
        if wr_en = '1' and addr_int >= 16 then  
            ram(addr_int) <= dio;  
        end if;  
    end if;  
end process;
```

-----

# Exemplo: CPU somadora

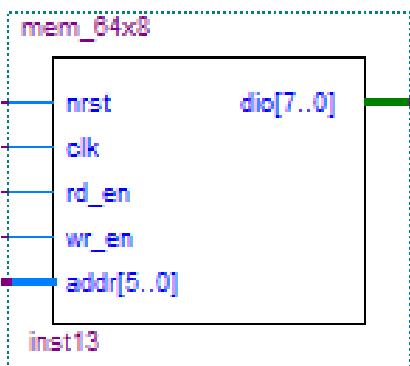
## Arquitetura von Neumann

CPU somadora

Memória

0..15 ROM

16..63 RAM



Programa a  
ser  
executado

```
-- leitura na memória:  
--  
process(rd_en, addr_int, ram)  
begin  
    if rd_en = '1' then  
        case addr_int is  
            -- ROM (16 bytes):  
                when 0 => dio <= "00100111"; -- ld (27h)  
                when 1 => dio <= "10000011"; -- add 03h  
                when 2 => dio <= "01100111"; -- sto (27h)  
                when 3 => dio <= "11000000"; -- jmp 00h  
                when 4 to 15 => dio <= (others => '0');  
            --  
            -- RAM (48 bytes):  
                when others => dio <= ram(addr_int);  
        end case;  
        else  
            dio <= (others => 'Z');  
        end if;  
    end process;  
end arch;
```

# Exemplo: CPU somadora

## Arquitetura von Neumann

CPU somadora  
Programa exemplo

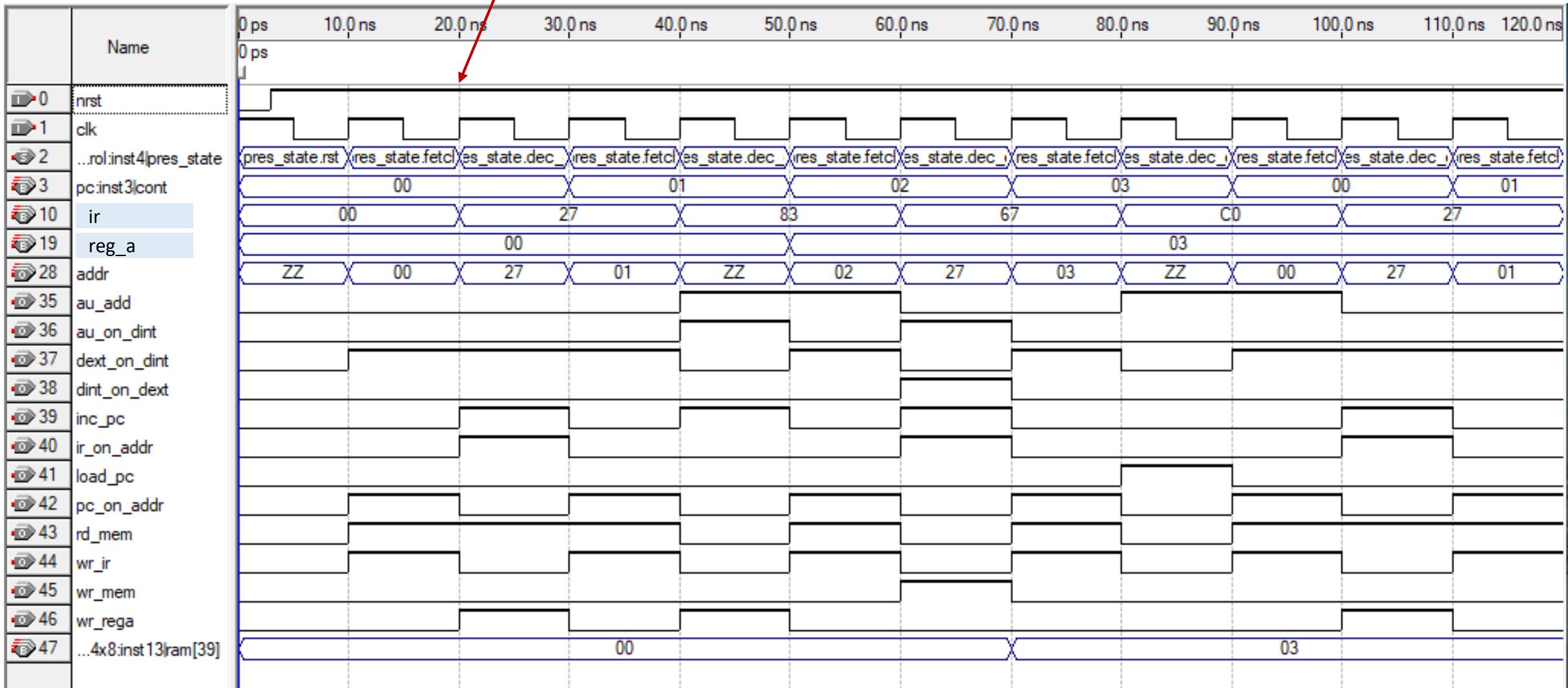
End.	Inst.	Instr. (bin)	Instr. (hexa)	Descrição
0	LD 27h	00100111	27	Carrega em reg_A o que está no endereço 39 (27h) da memória
1	ADD 03h	10000011	83	Soma 3 ao que está em reg_A
2	STO 27h	01100111	67	Escreve no endereço 39 (27h) da memória o que está em reg_A
3	JMP 00h	11000000	C0	Desvia o programa para o endereço 0

# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional



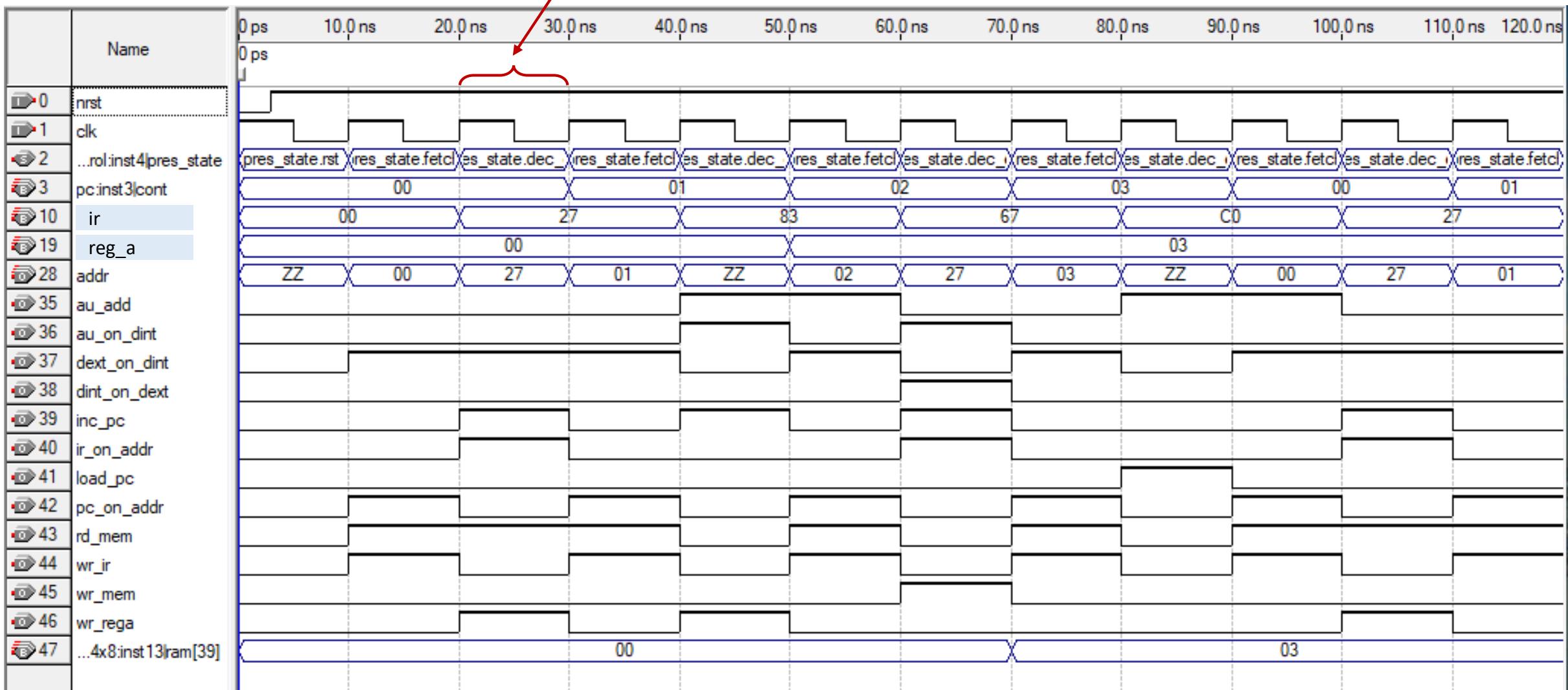
# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

A instrução do endereço 0 (27h) foi para o IR



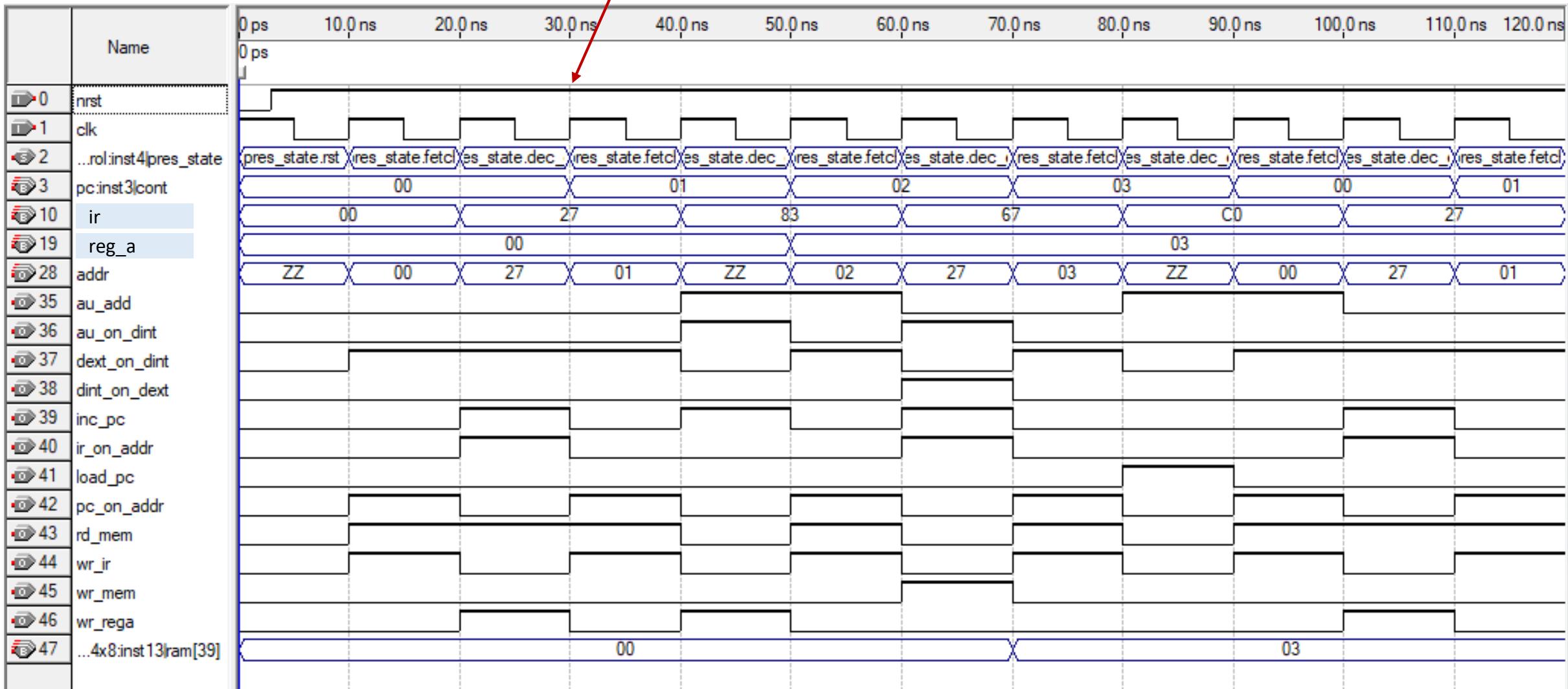
# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

Decodificando e executando a instrução LD 27h

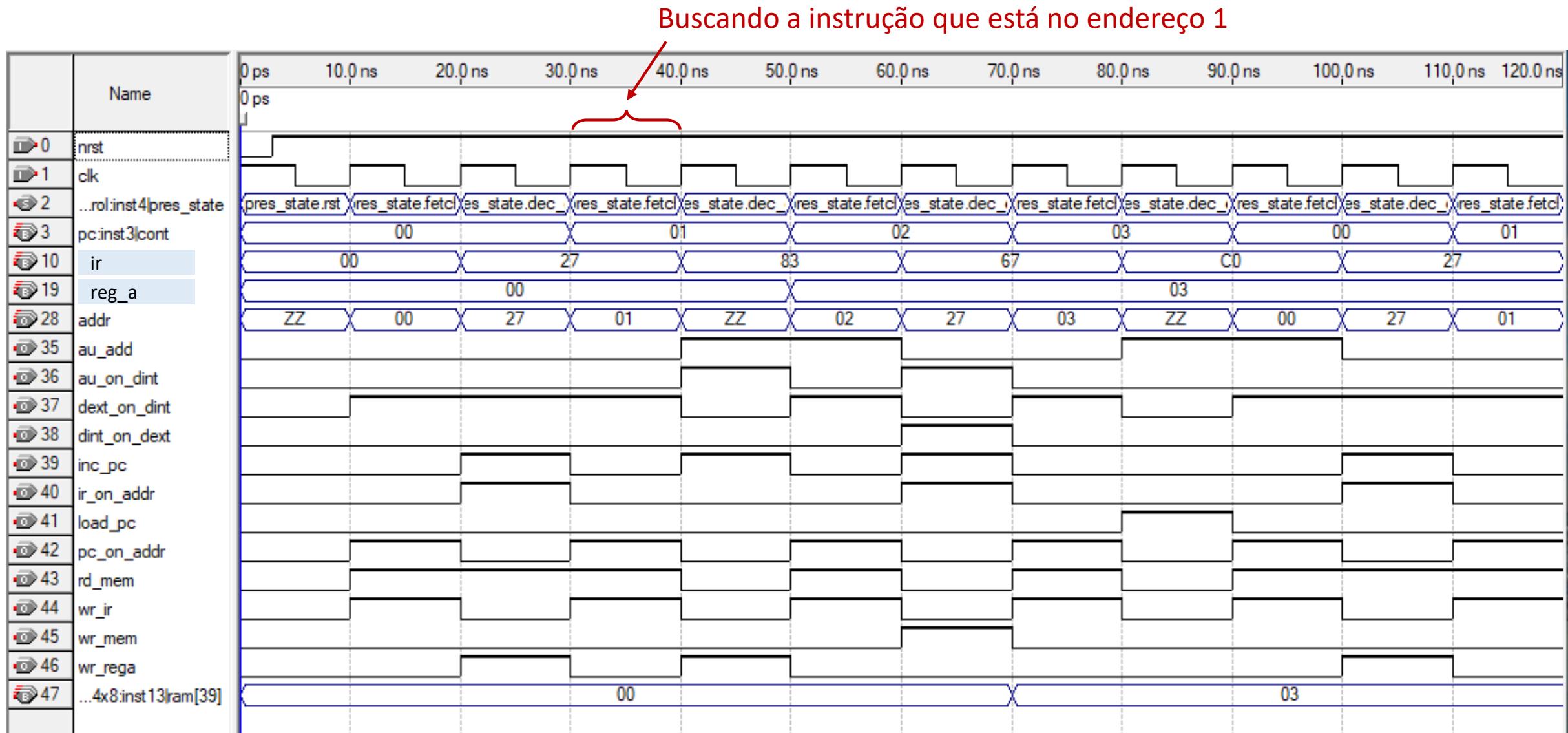


## Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

O que estava no endereço 27h da memória foi para o **reg\_a**

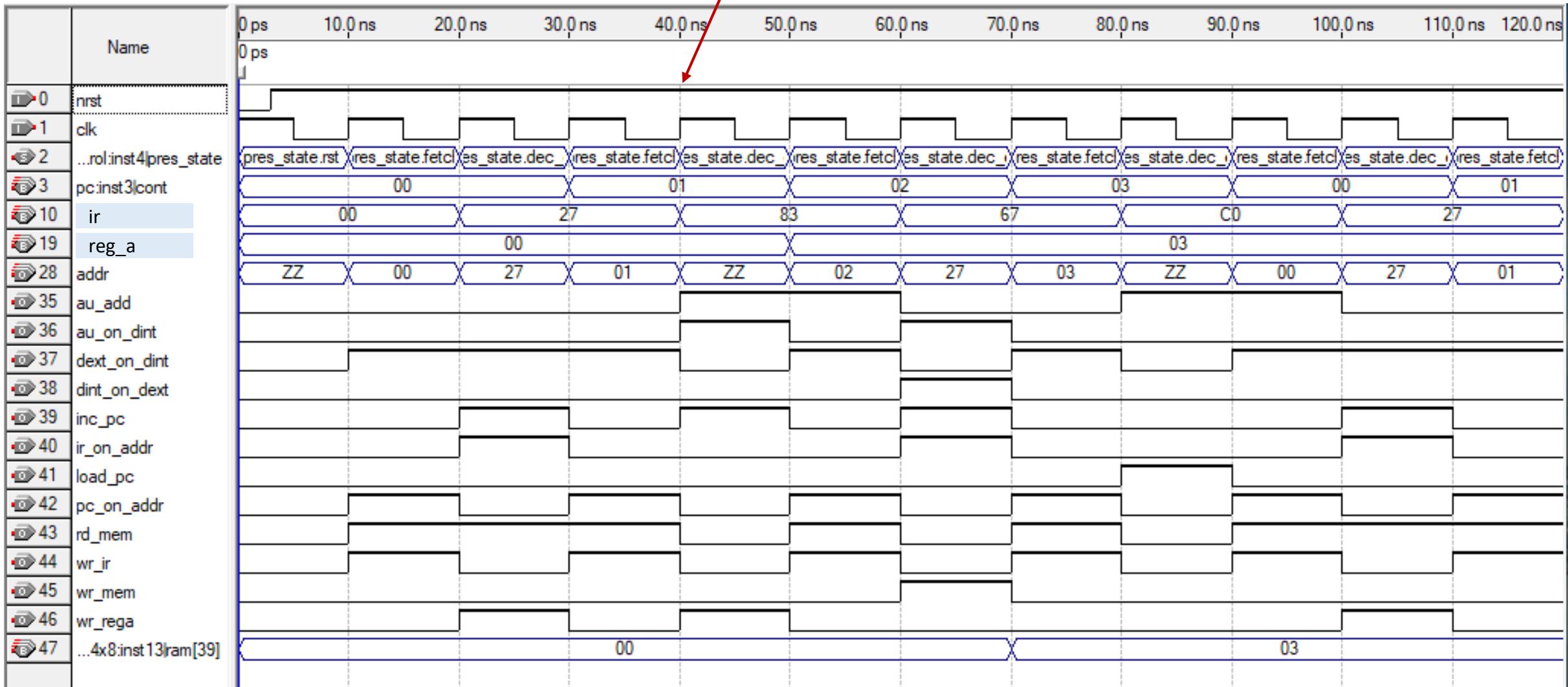


# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional



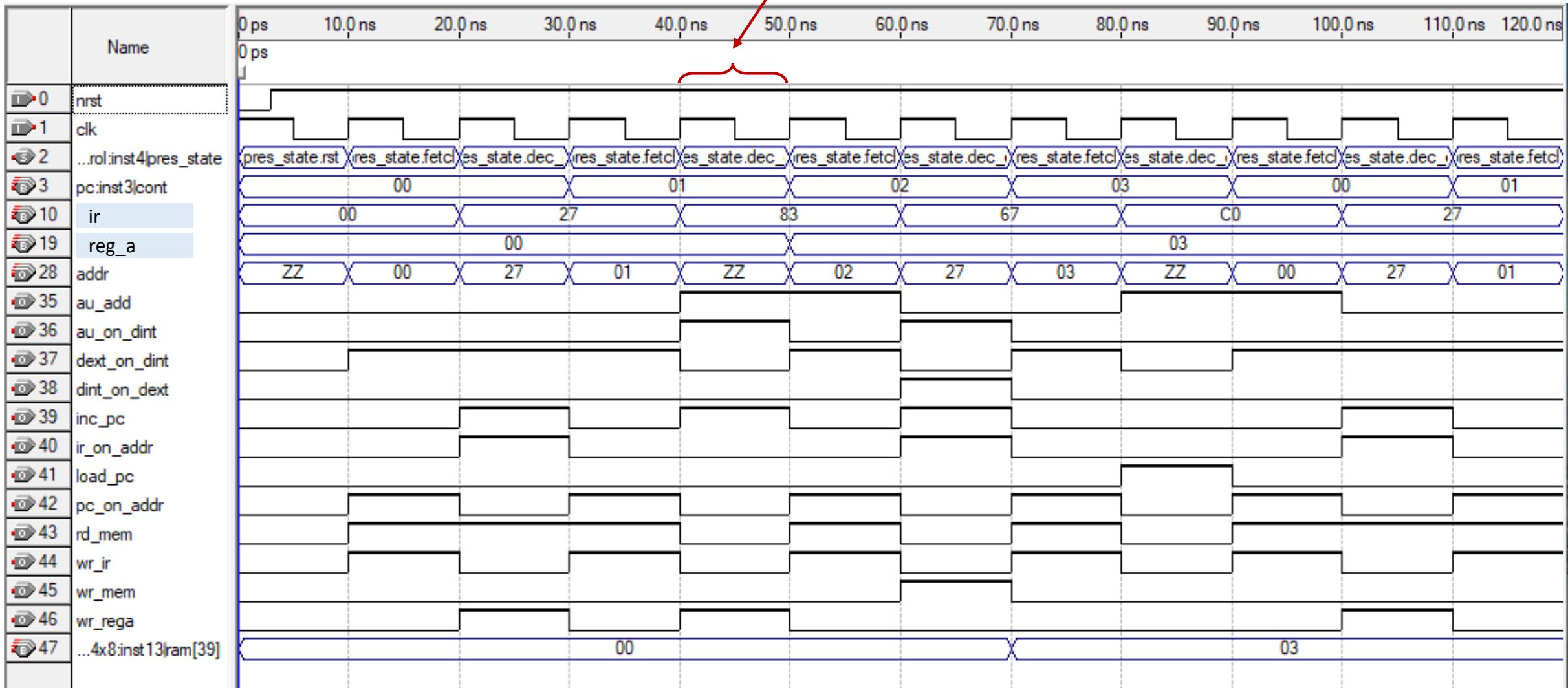
# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

A instrução do endereço 1 (83h) foi para o IR



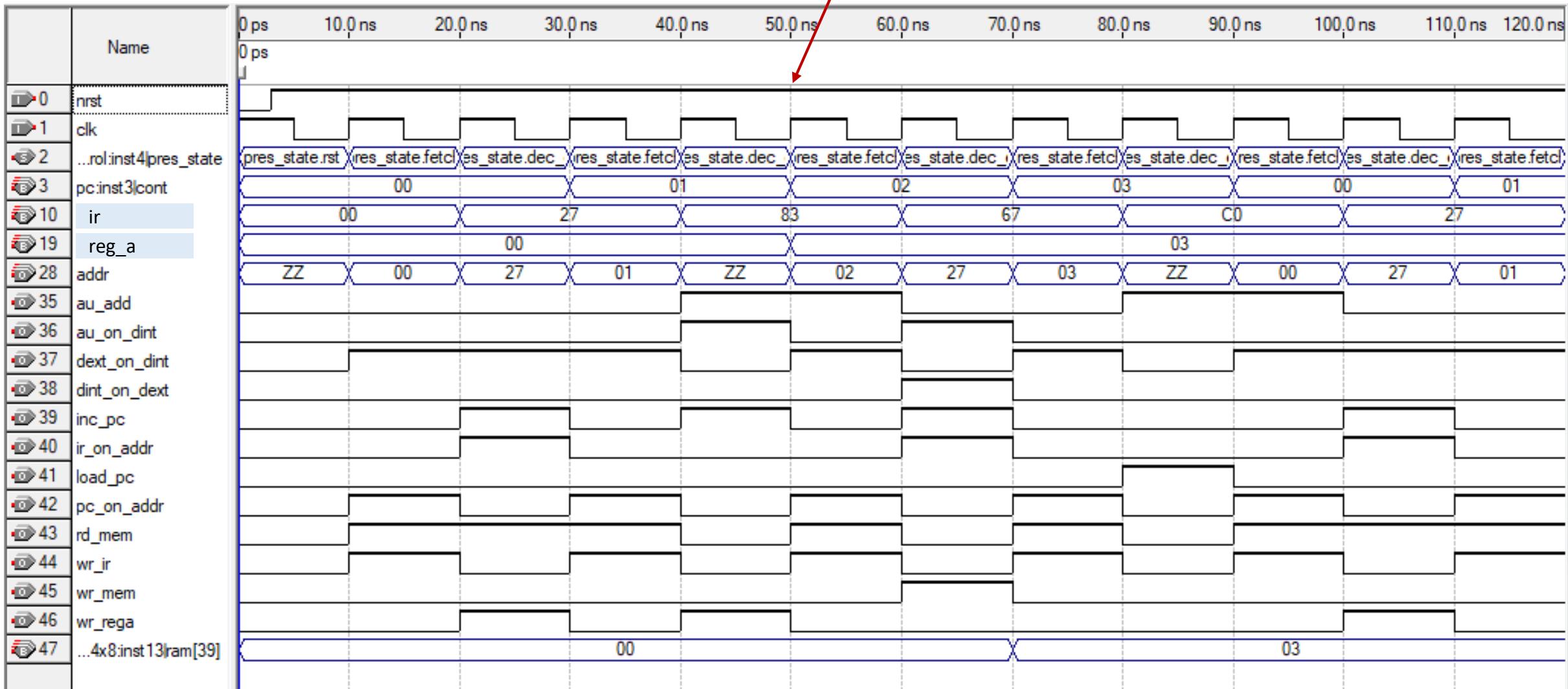
# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

Decodificando e executando a instrução ADD 03h

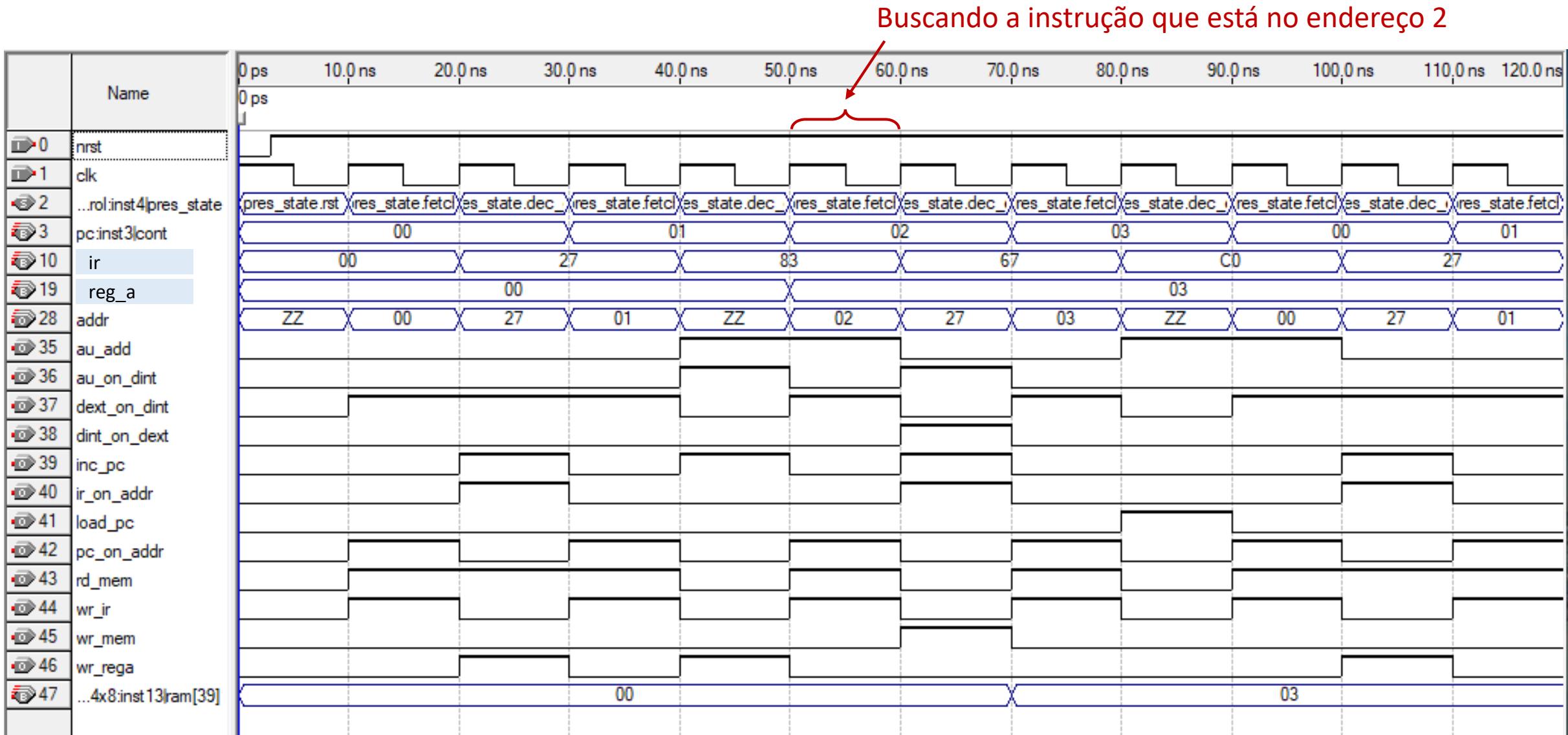


## Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

O resultado da soma foi para o **reg\_a**

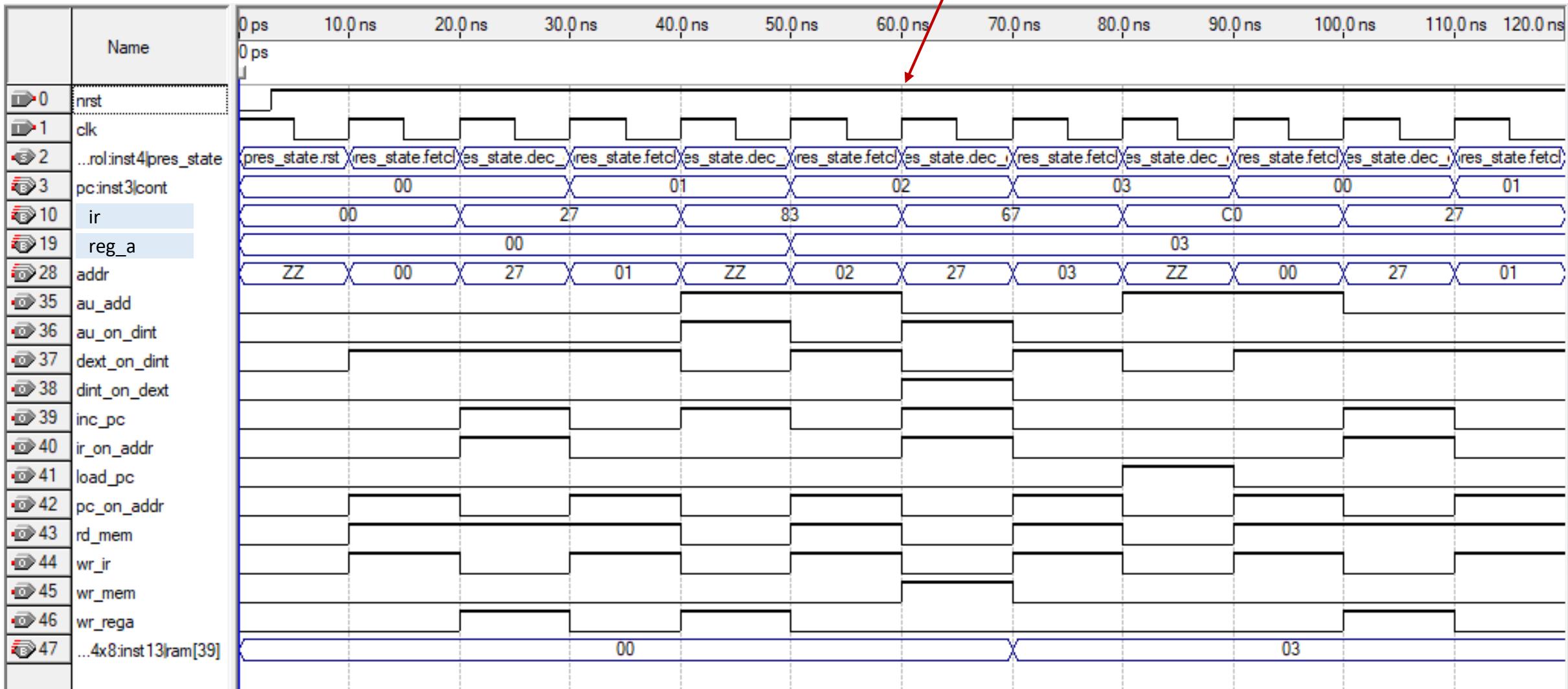


# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional



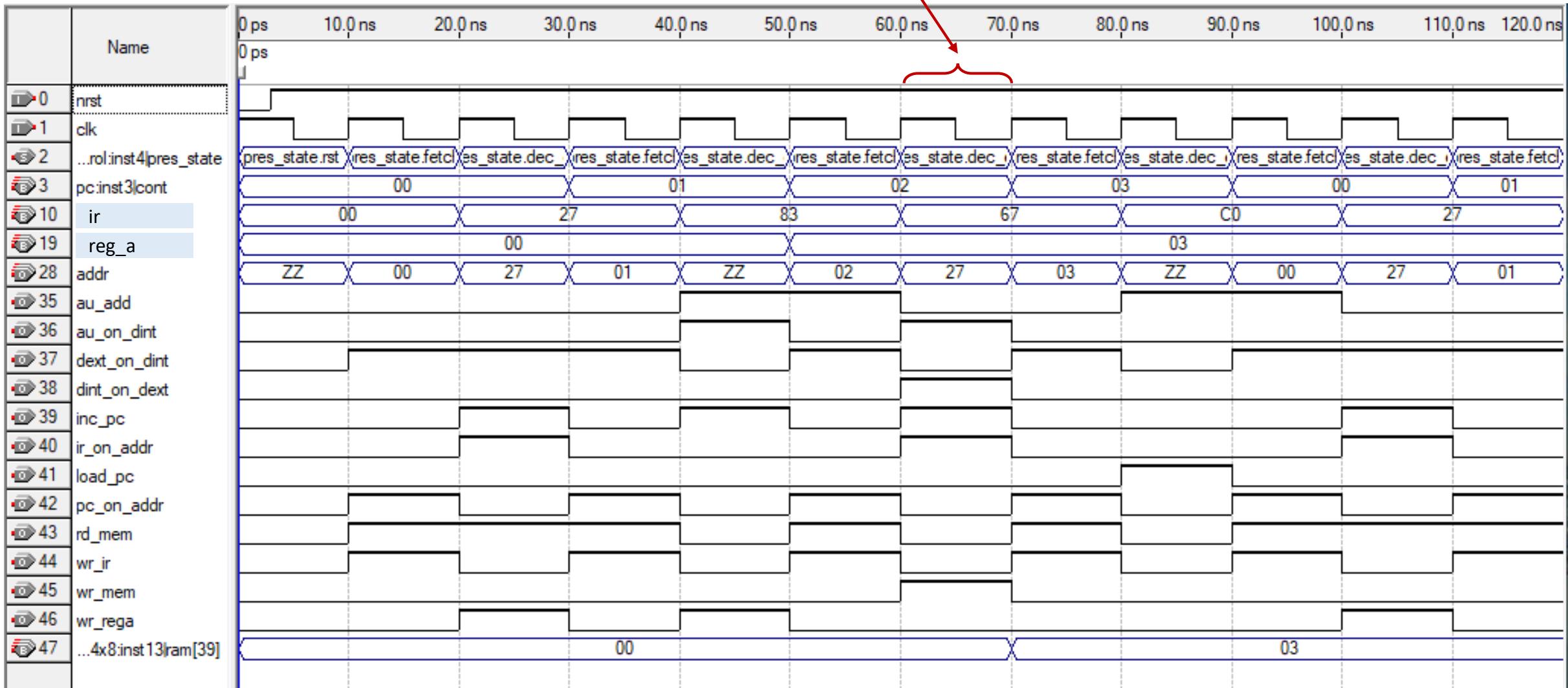
## Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

A instrução do endereço 2 (67h) foi para o IR



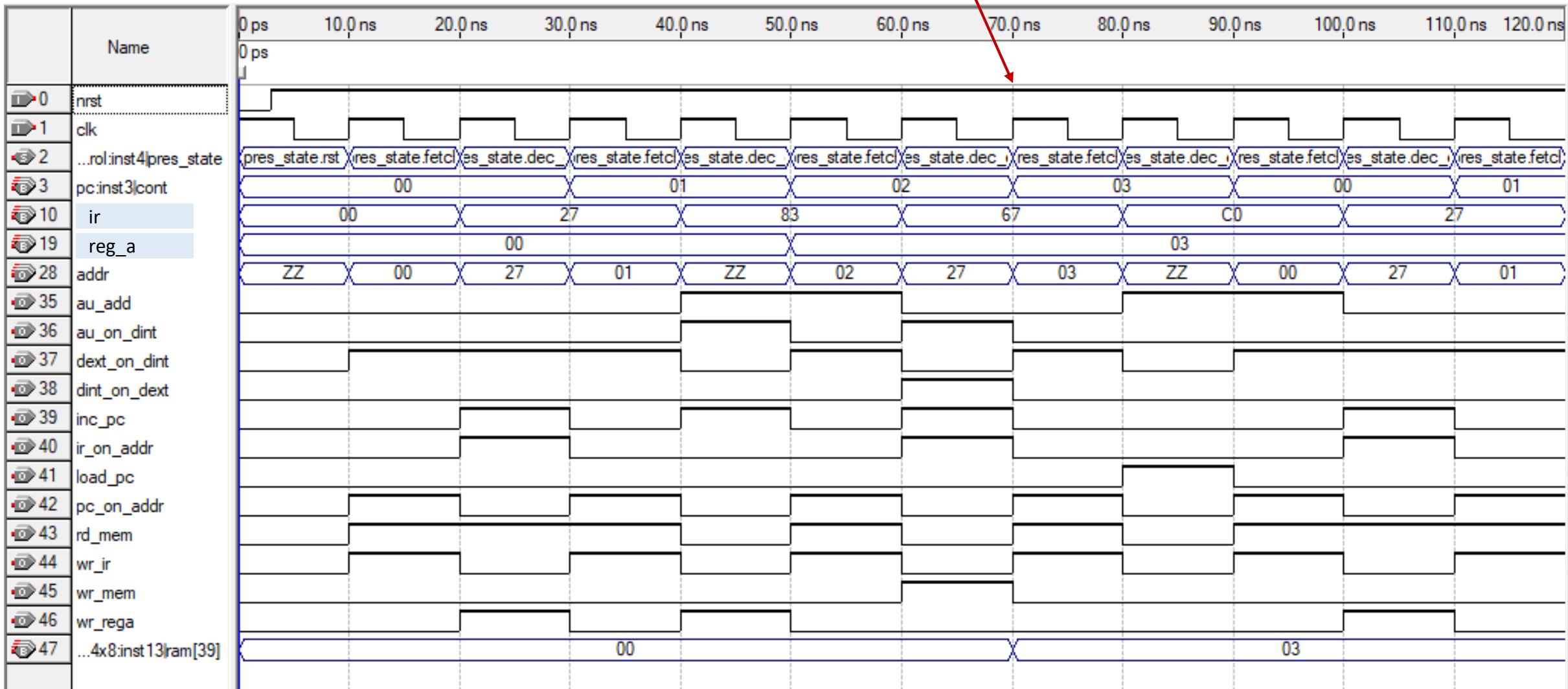
# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

Decodificando e executando a instrução STO 27h



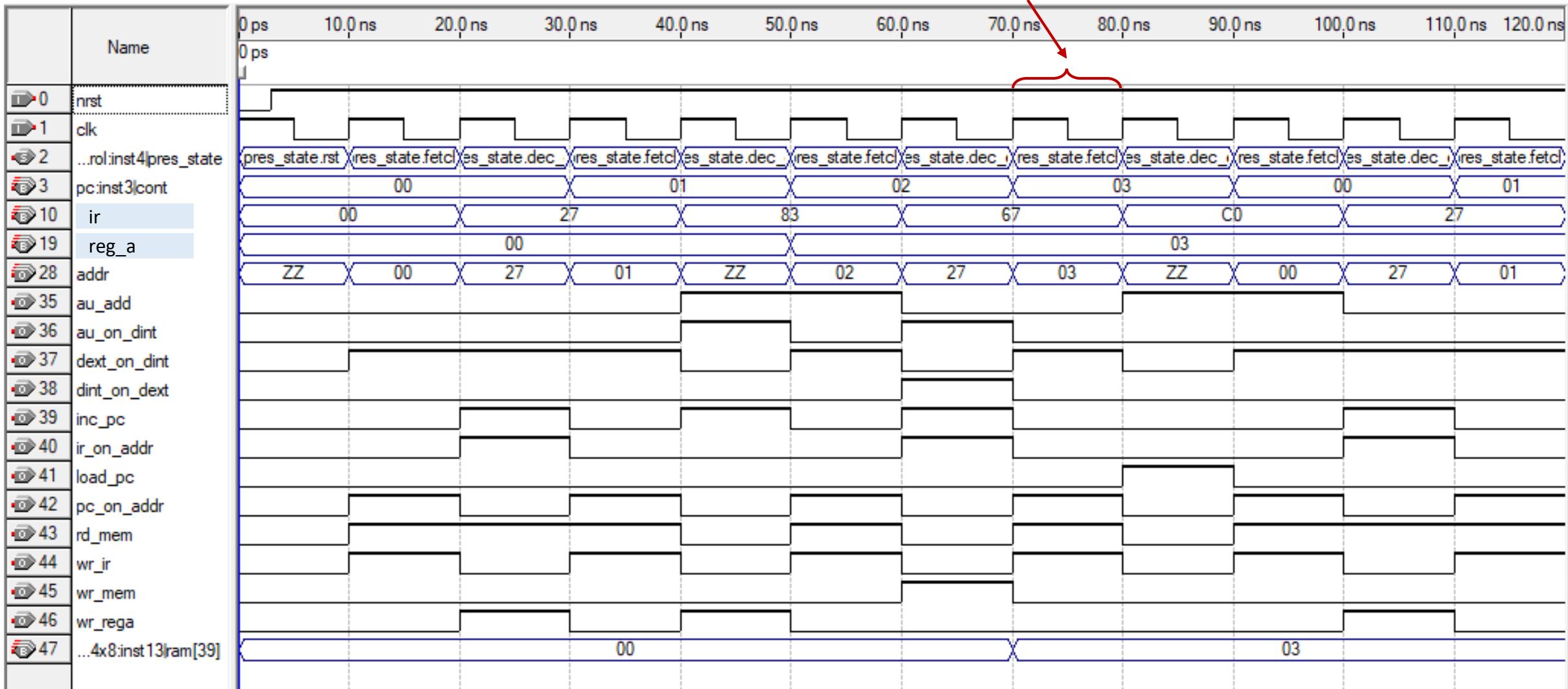
# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

O que está em **reg\_a** foi para o endereço 27h da memória



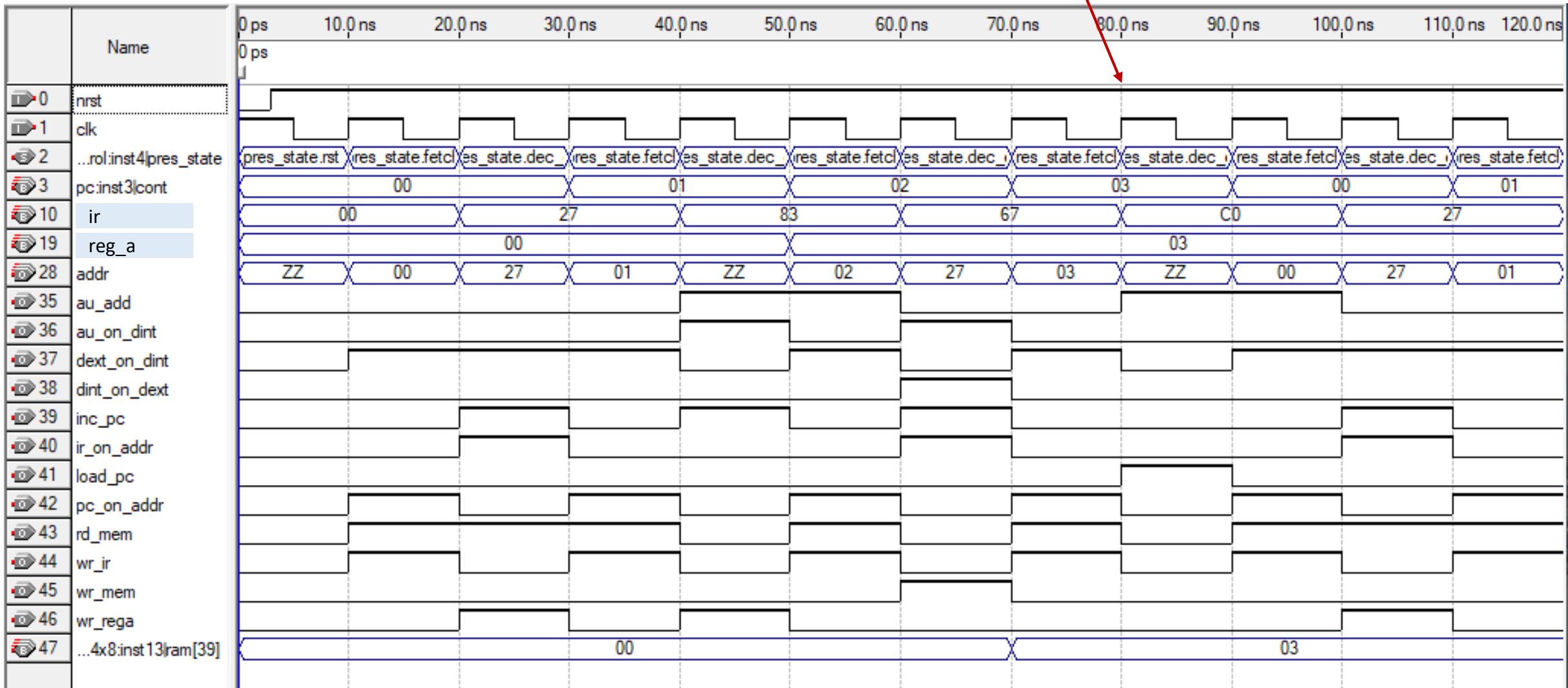
# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

Buscando a instrução que está no endereço 3



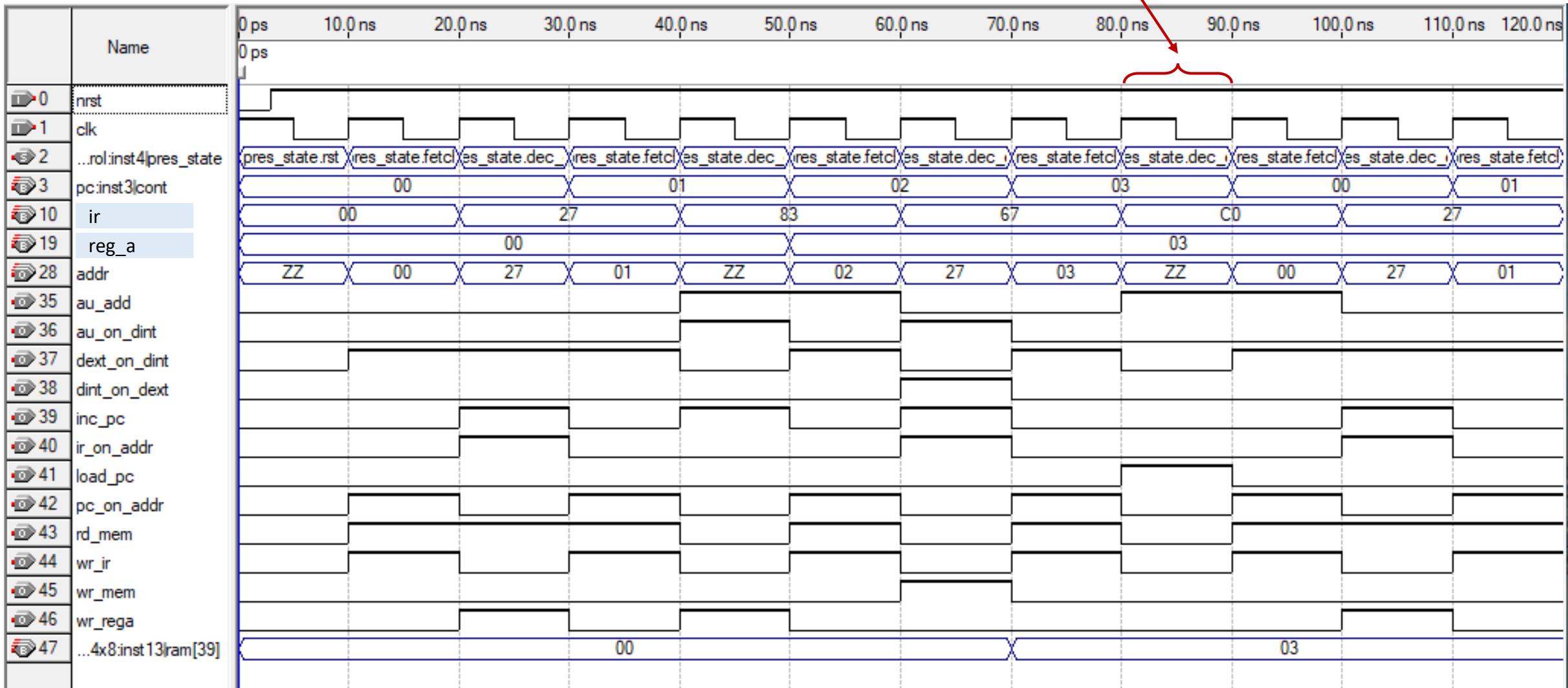
# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

A instrução do endereço 3 (C0h) foi para o IR

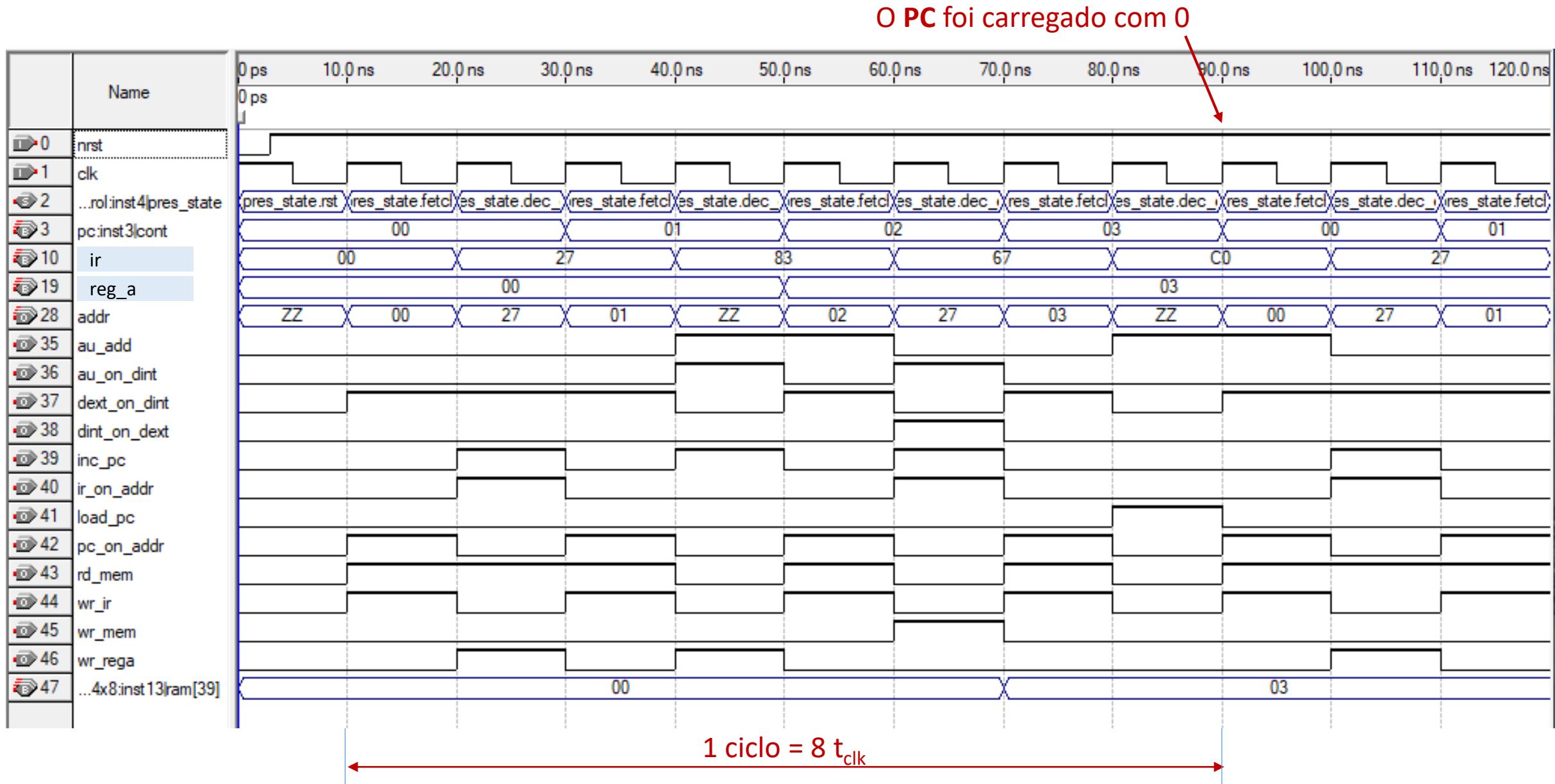


# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

Decodificando e executando a instrução JMP 00h

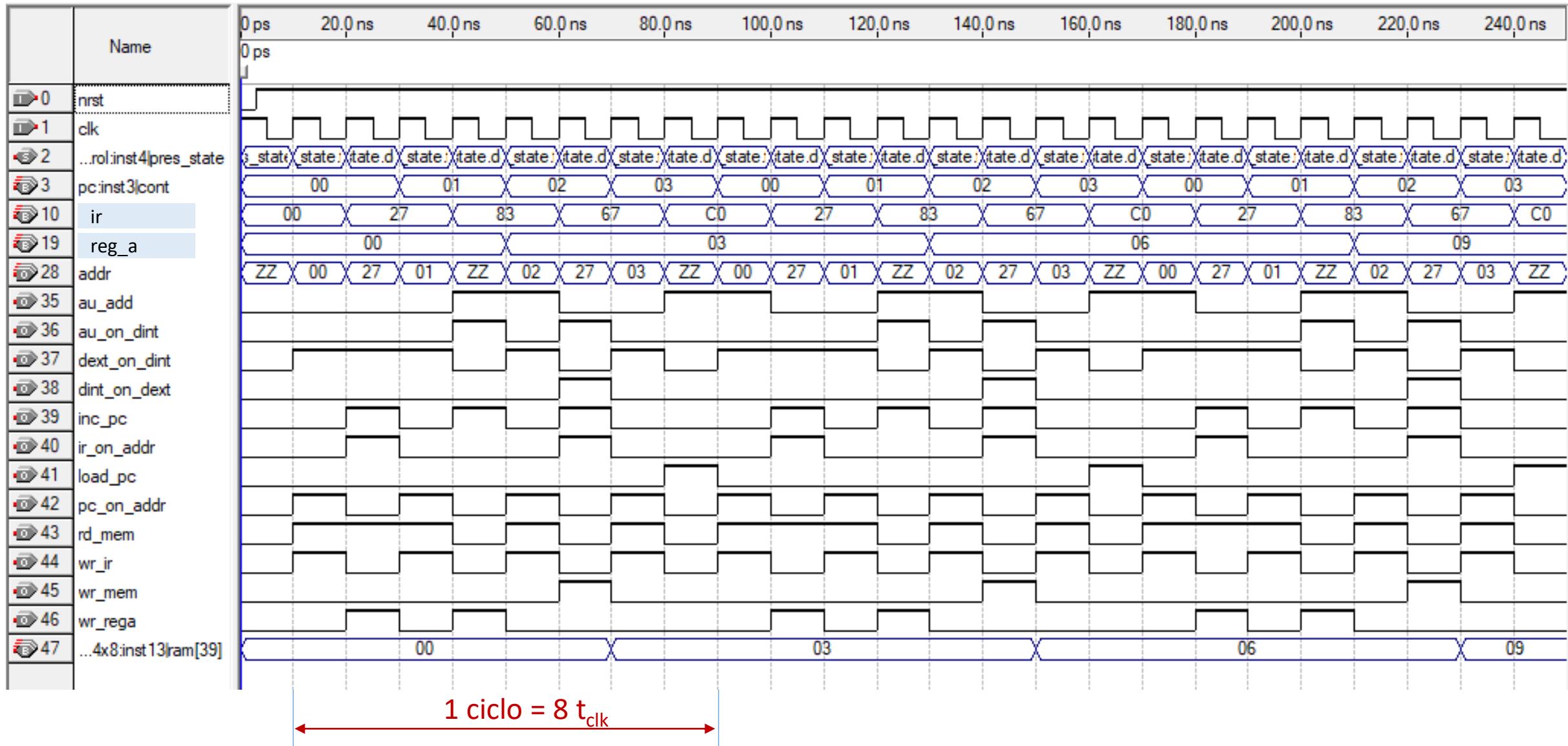


# Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional



## Exemplo: CPU somadora - Arquitetura von Neumann – simulação funcional

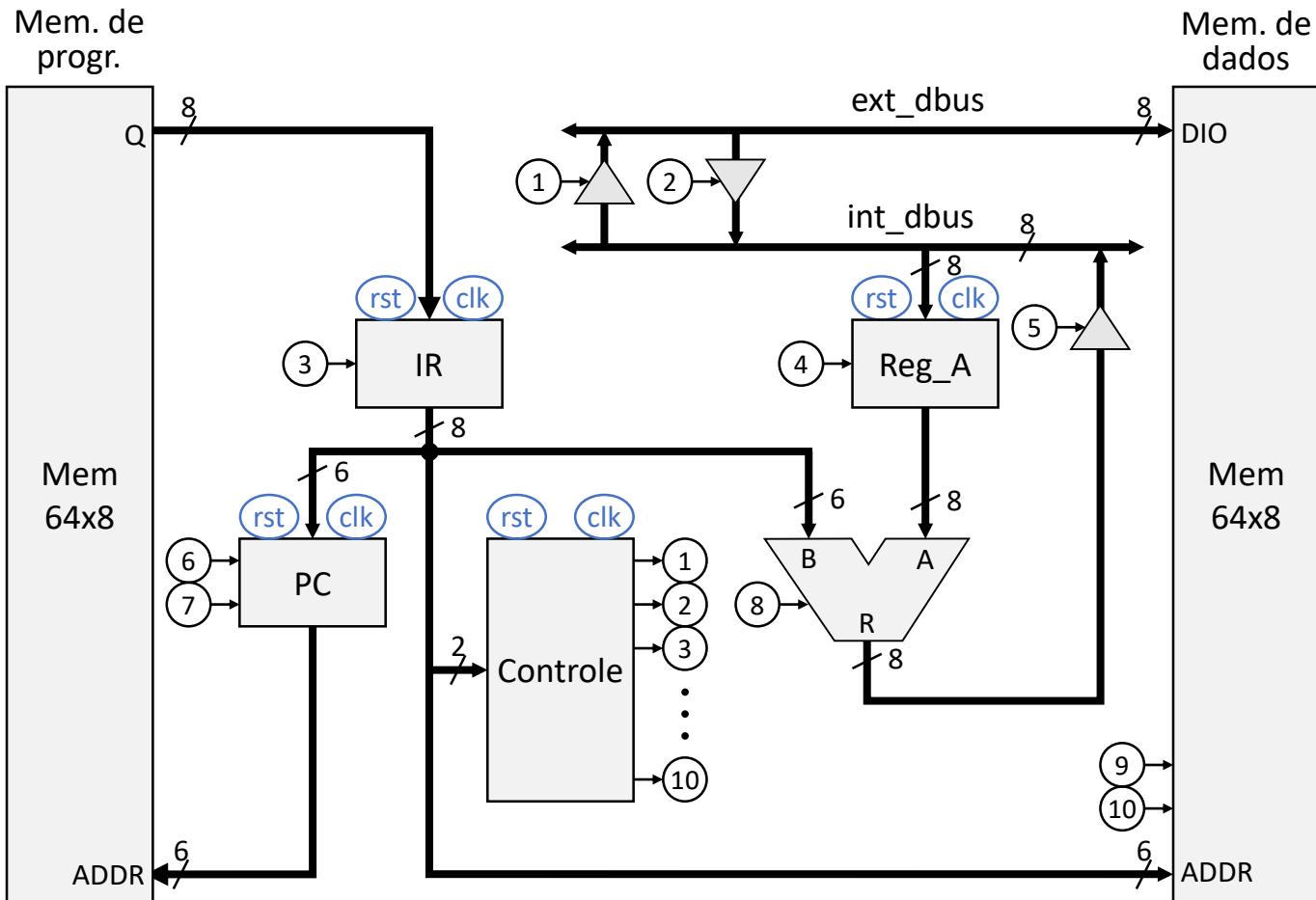
Em outra escala de tempo:



# Exemplo: CPU somadora

## Arquitetura Harvard

CPU somadora - Arquitetura interna (Harvard)

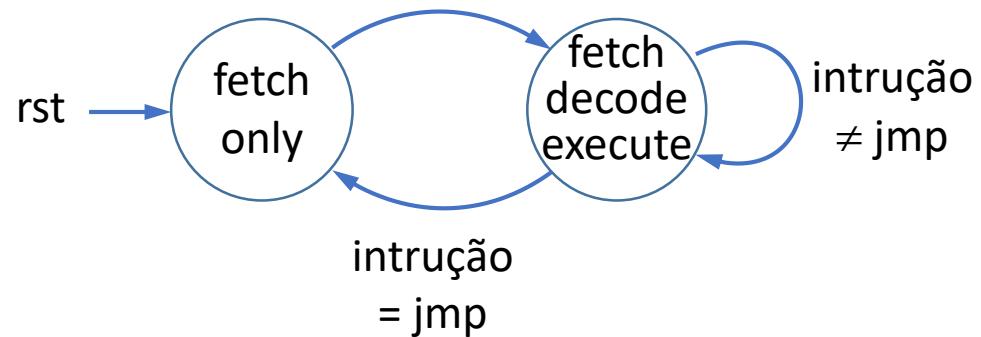


Sinal	Nome
1	dint_on_dext
2	dext_on_dint
3	wr_ir
4	wr_rega
5	au_on_dint
6	inc_pc
7	load_pc
8	au_add
9	rd_mem
10	wr_mem

Todos os sinais ativos em nível alto, exceto  
au\_add: '1' => add, '0' => pass\_B

# Exemplo: CPU somadora Arquitetura Harvard

## CPU somadora – Estados da máquina de controle



# Exemplo: CPU somadora

## Arquitetura Harvard

### CPU somadora – Sinais de controle

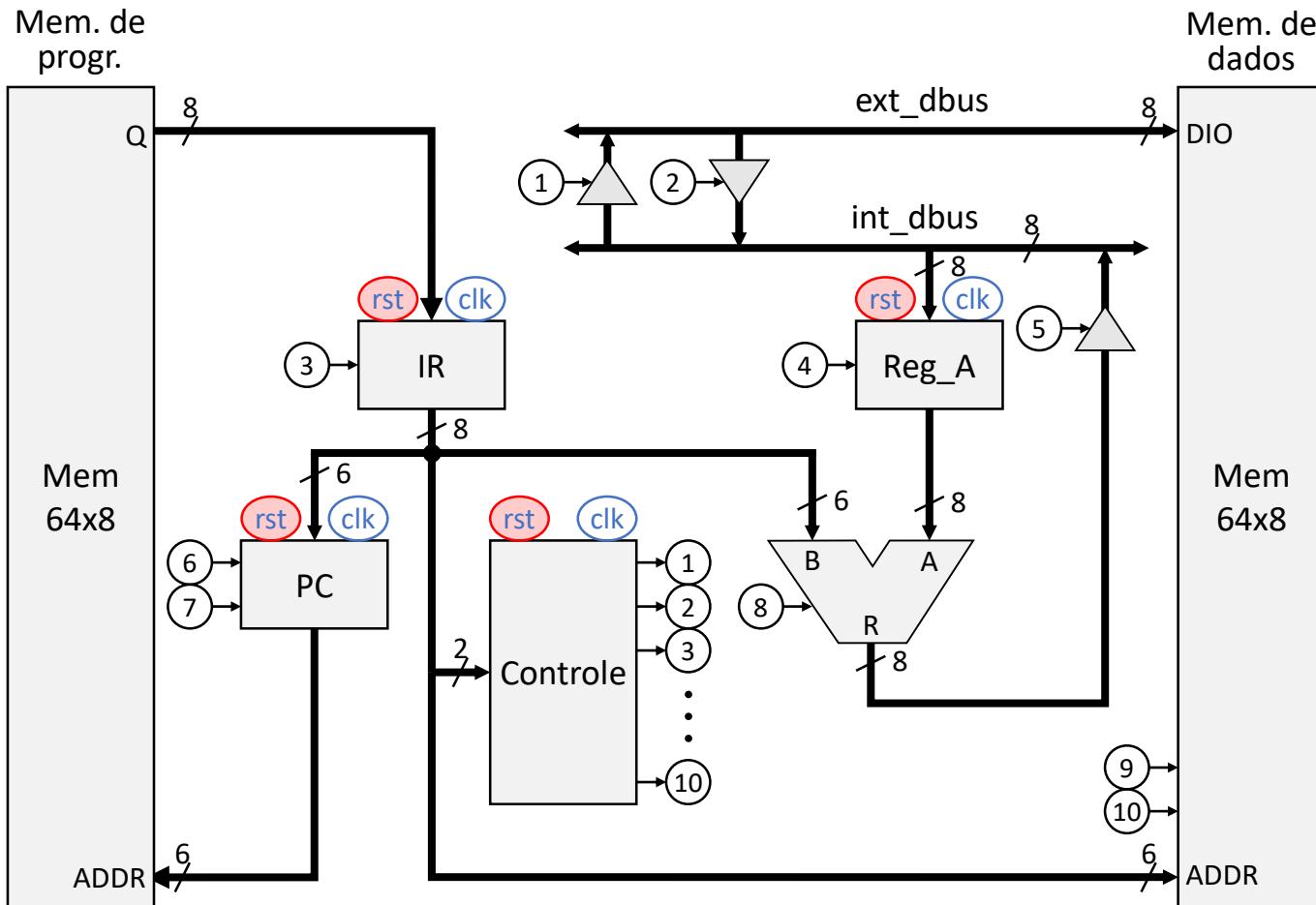


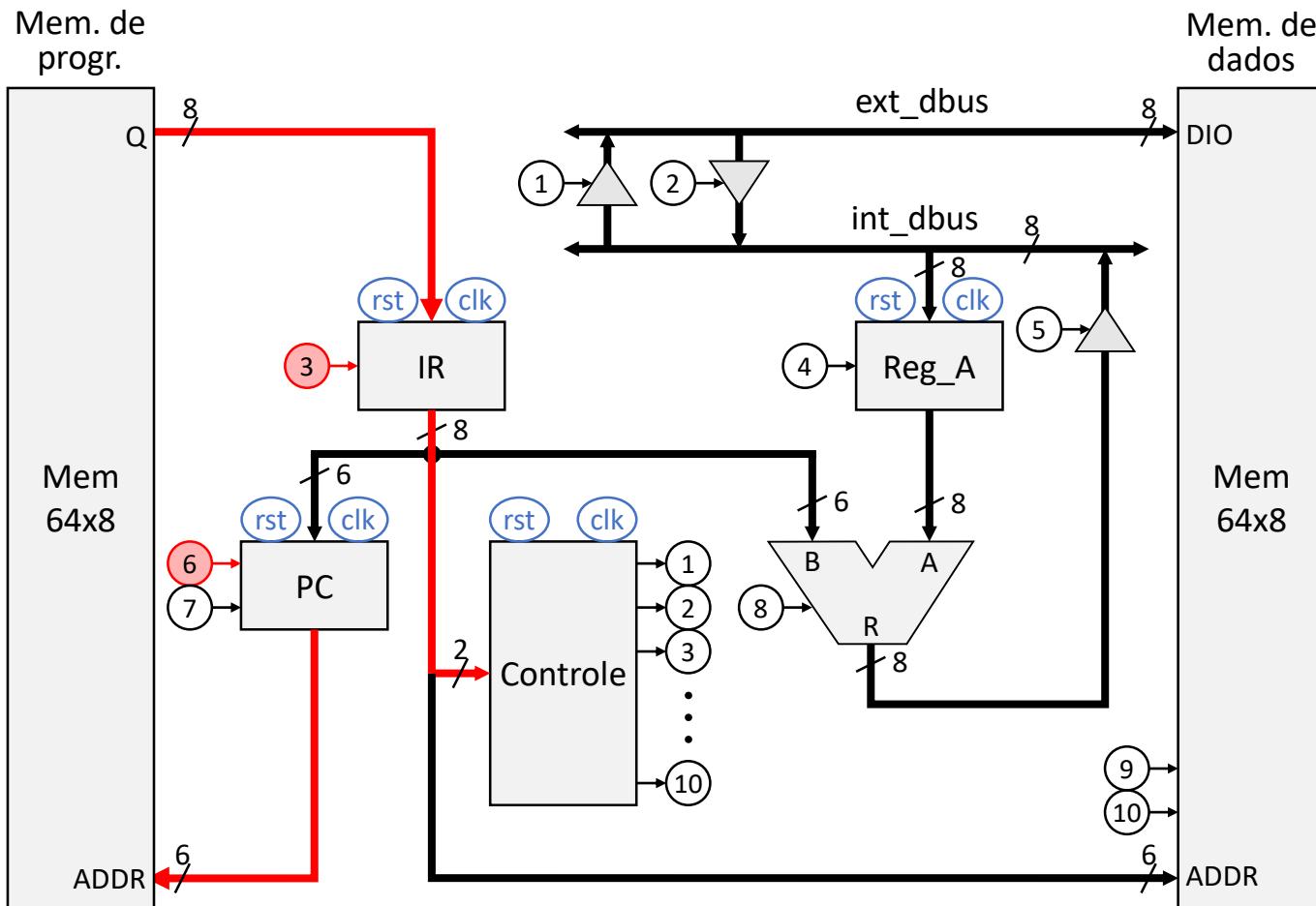
Diagram of the CPU control signals:

Sinal	Nome	reset	fetch only	fetch_decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	0	1	0	0	0
3	wr_ir	0	1	1	1	1	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	1	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	au_add	X	X	X	0	1	X
9	rd_mem	0	0	1	0	0	0
10	wr_mem	0	0	0	1	0	0

# Exemplo: CPU somadora

## Arquitetura Harvard

### CPU somadora – Sinais de controle

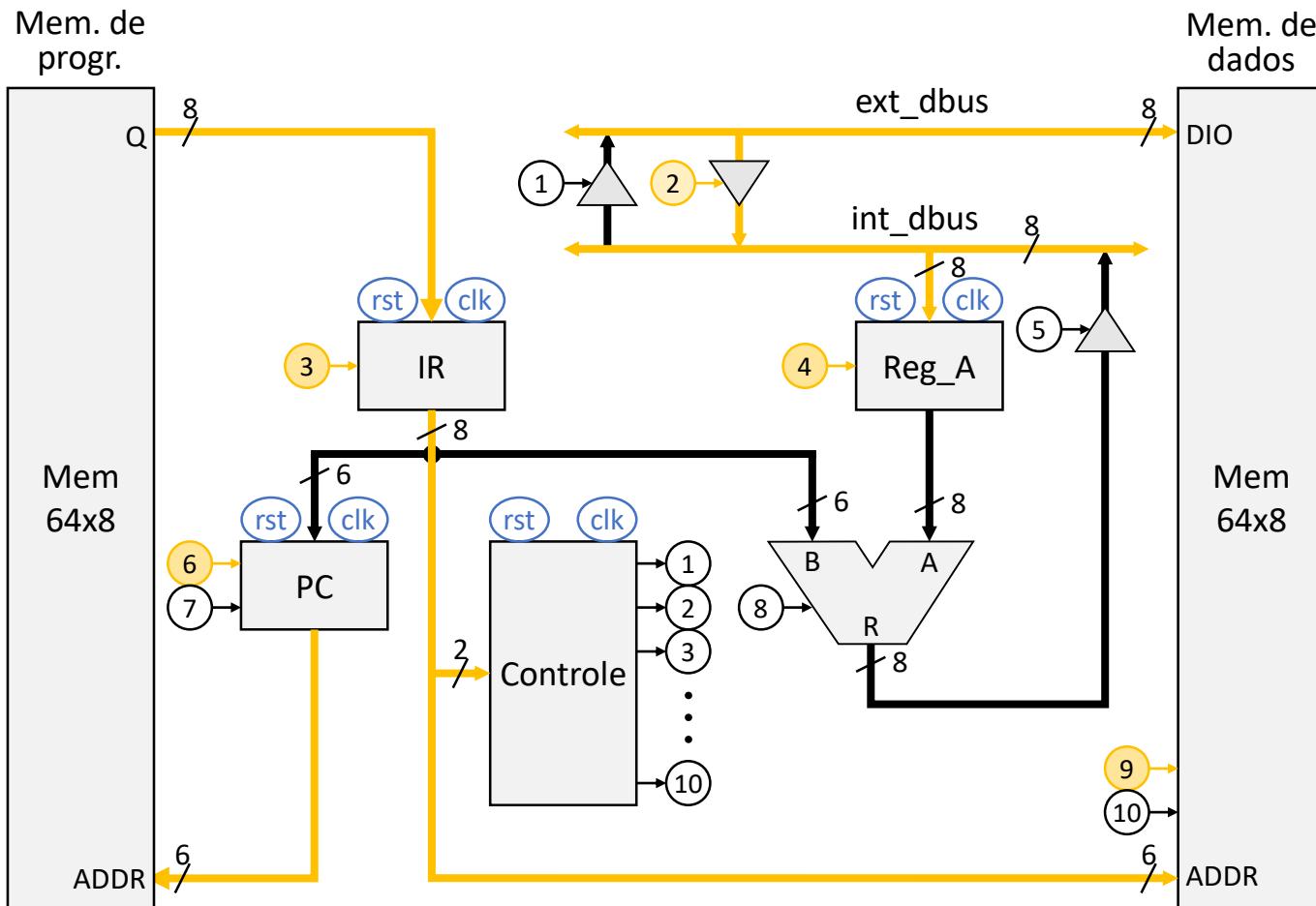


Sinal	Nome	reset	fetch only	fetch_decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	0	1	0	0	0
3	wr_ir	0	1	1	1	1	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	1	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	au_add	X	X	X	0	1	X
9	rd_mem	0	0	1	0	0	0
10	wr_mem	0	0	0	1	0	0

# Exemplo: CPU somadora

## Arquitetura Harvard

### CPU somadora – Sinais de controle

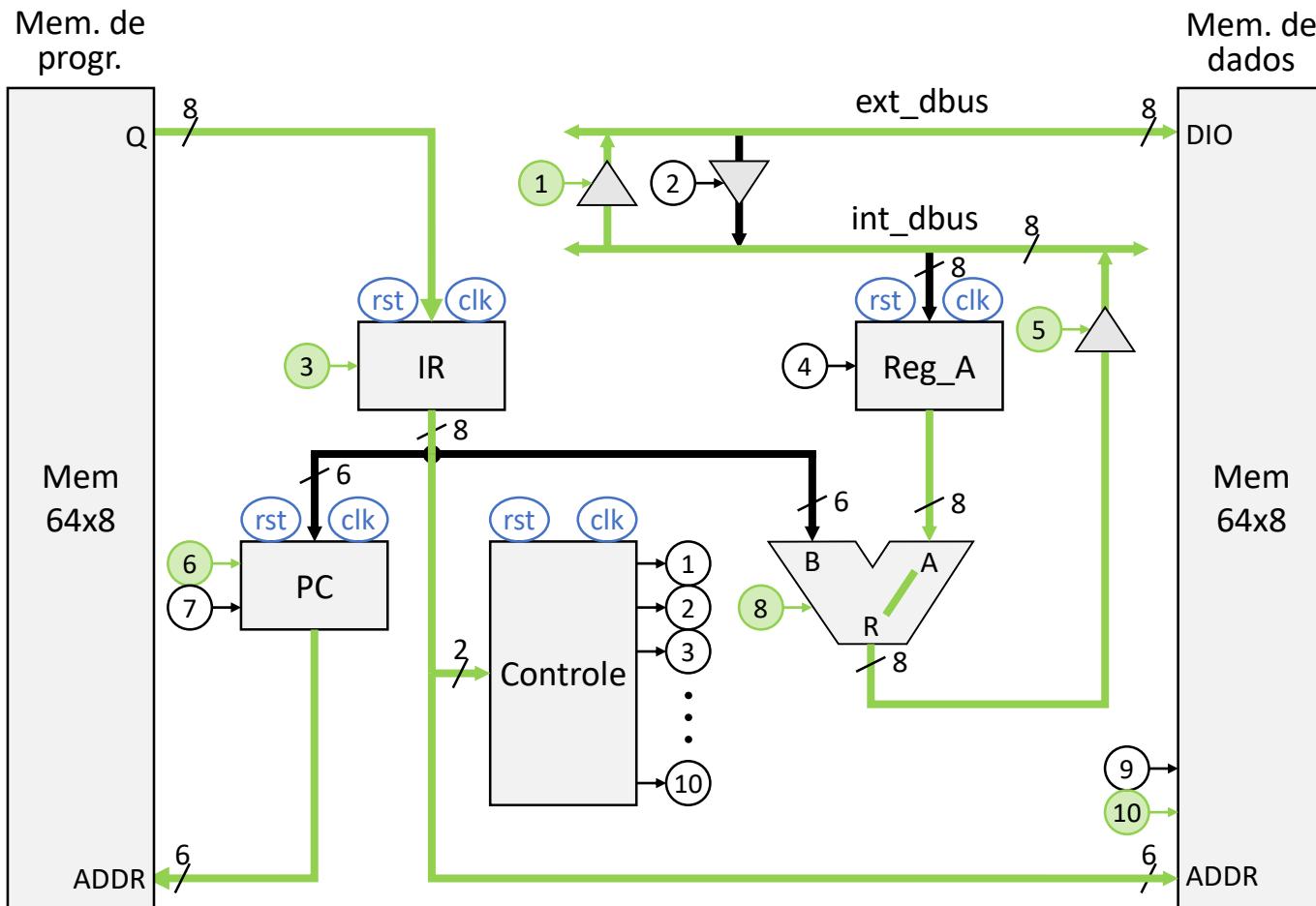


Sinal	Nome	reset	fetch only	fetch_decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	0	1	0	0	0
3	wr_ir	0	1	1	1	1	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	1	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	au_add	X	X	X	0	1	X
9	rd_mem	0	0	1	0	0	0
10	wr_mem	0	0	0	1	0	0

# Exemplo: CPU somadora

## Arquitetura Harvard

### CPU somadora – Sinais de controle

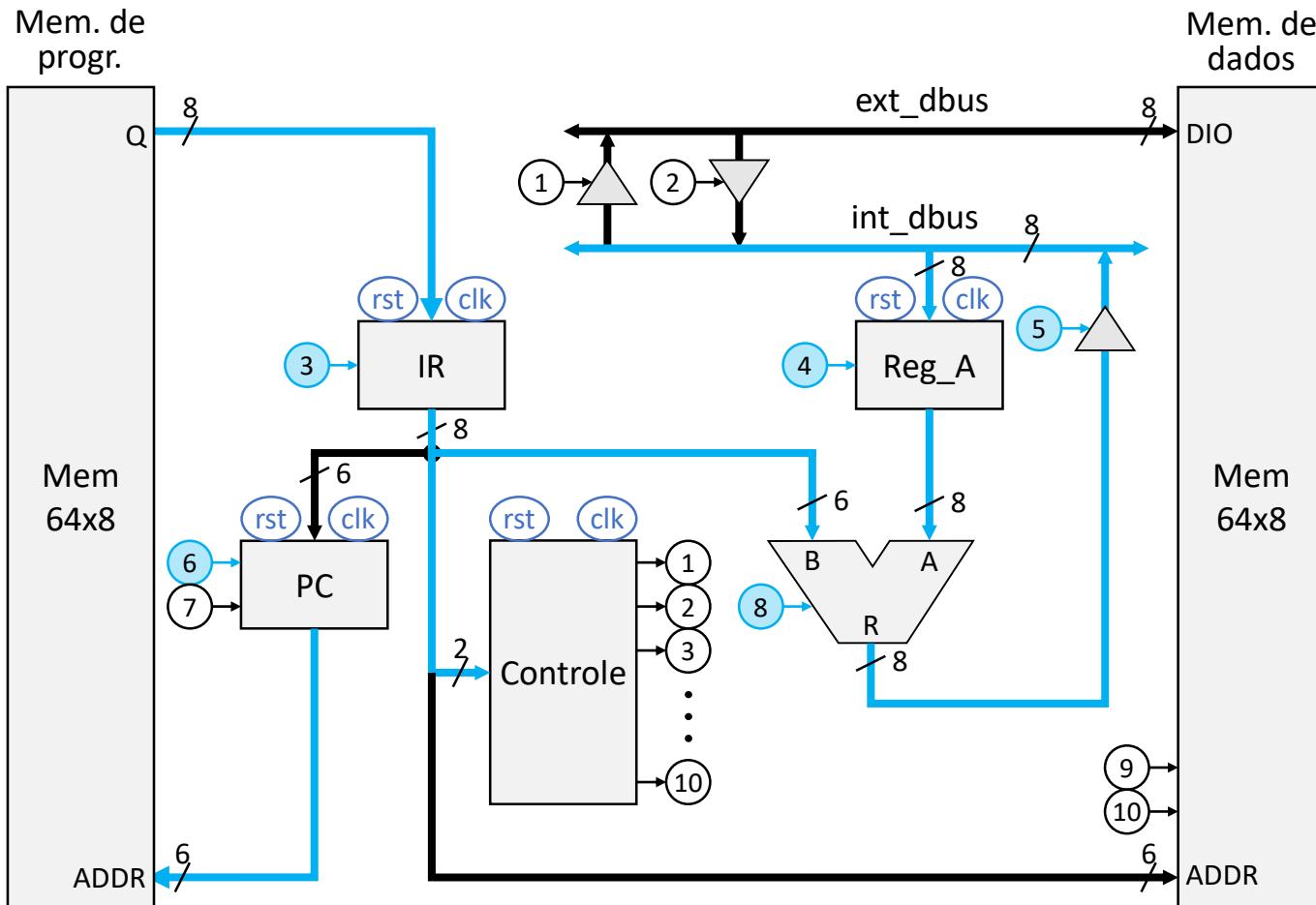


Sinal	Nome	reset	fetch only	fetch_decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	0	1	0	0	0
3	wr_ir	0	1	1	1	1	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	1	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	au_add	X	X	X	0	1	X
9	rd_mem	0	0	1	0	0	0
10	wr_mem	0	0	0	1	0	0

# Exemplo: CPU somadora

## Arquitetura Harvard

### CPU somadora – Sinais de controle

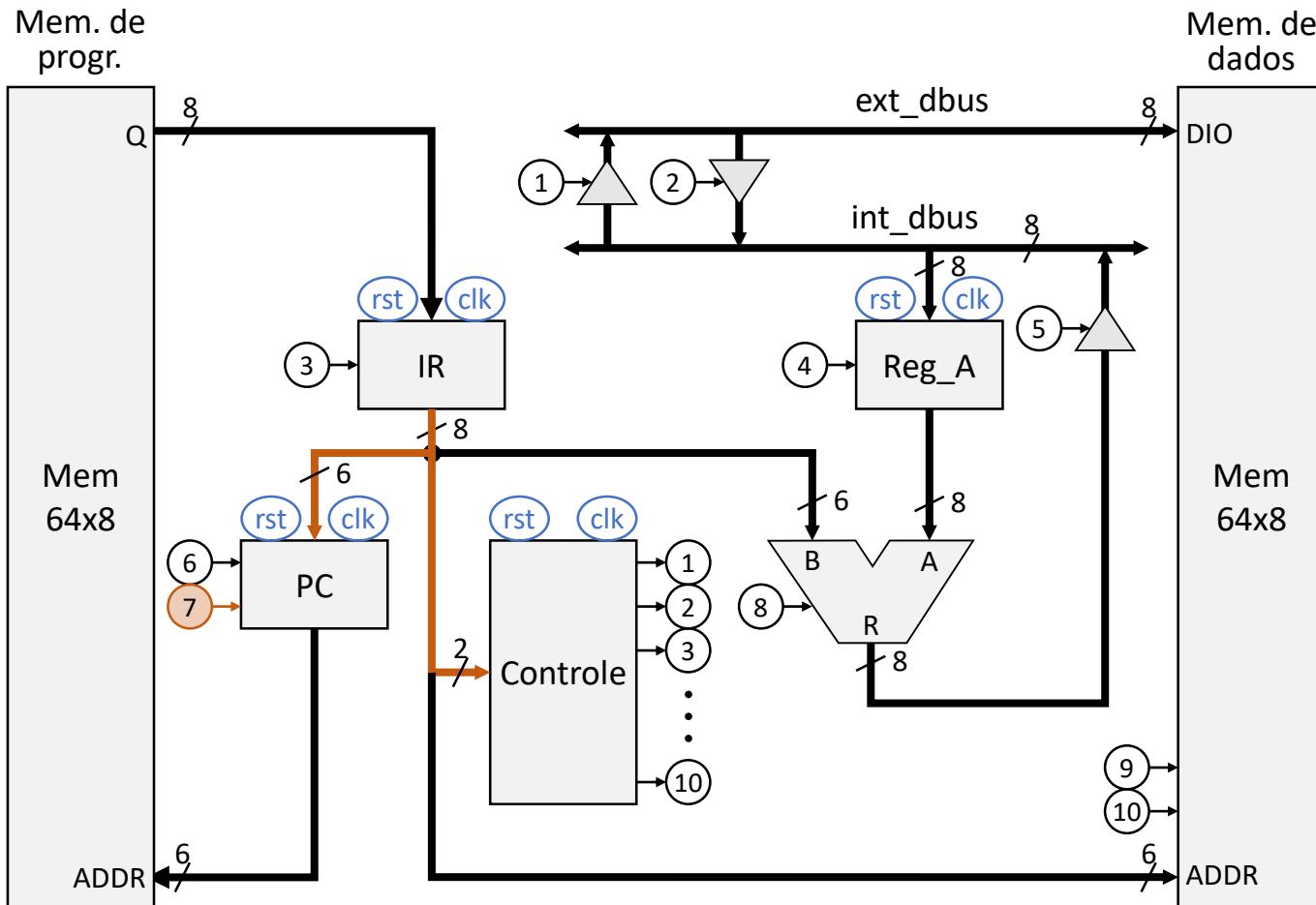


Sinal	Nome	reset	fetch only	fetch_decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	0	1	0	0	0
3	wr_ir	0	1	1	1	1	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	1	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	au_add	X	X	X	0	1	X
9	rd_mem	0	0	1	0	0	0
10	wr_mem	0	0	0	1	0	0

# Exemplo: CPU somadora

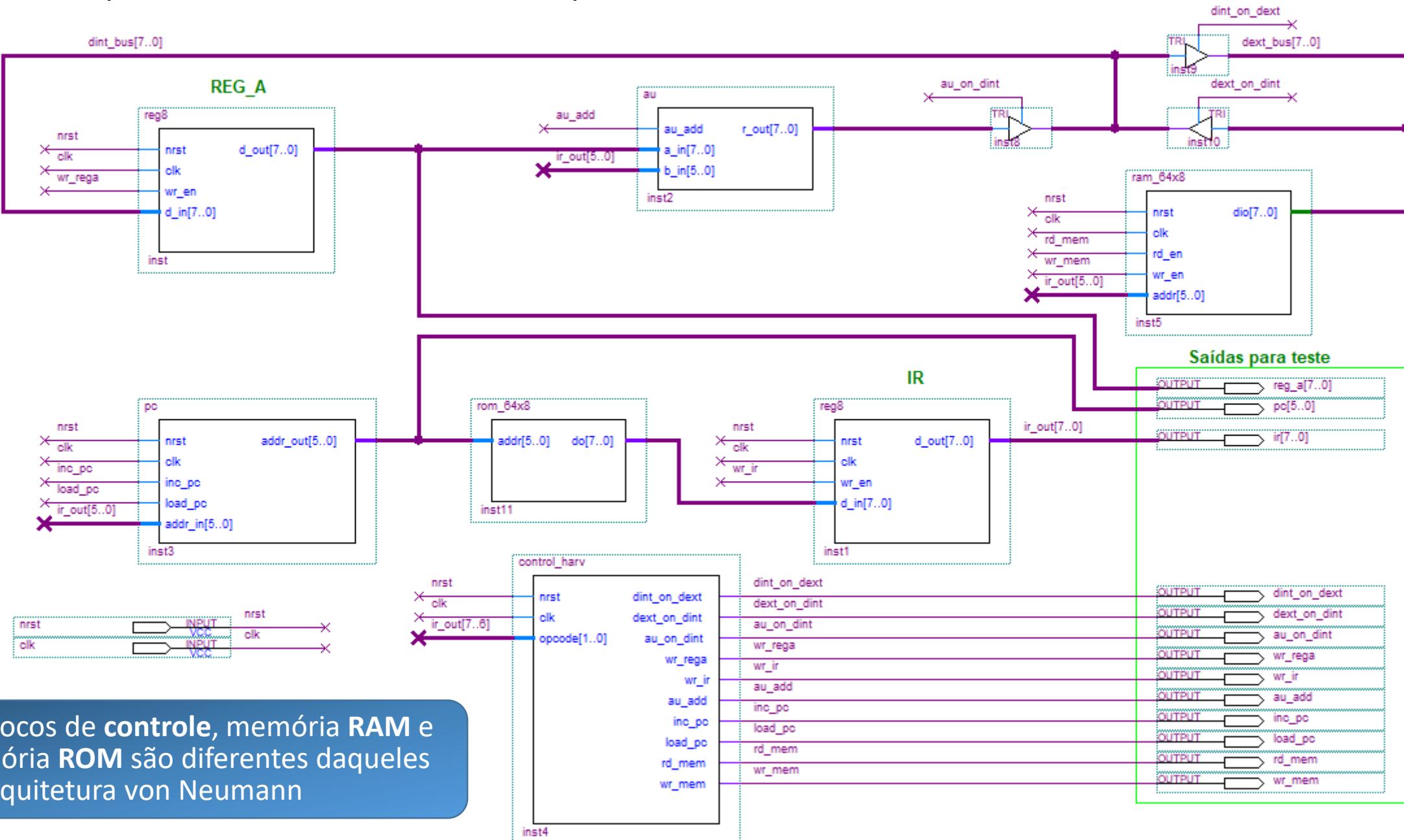
## Arquitetura Harvard

### CPU somadora – Sinais de controle



Sinal	Nome	reset	fetch only	fetch_decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	0	1	0	0	0
3	wr_ir	0	1	1	1	1	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	1	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	au_add	X	X	X	0	1	X
9	rd_mem	0	0	1	0	0	0
10	wr_mem	0	0	0	1	0	0

# Exemplo: CPU somadora - Arquitetura Harvard



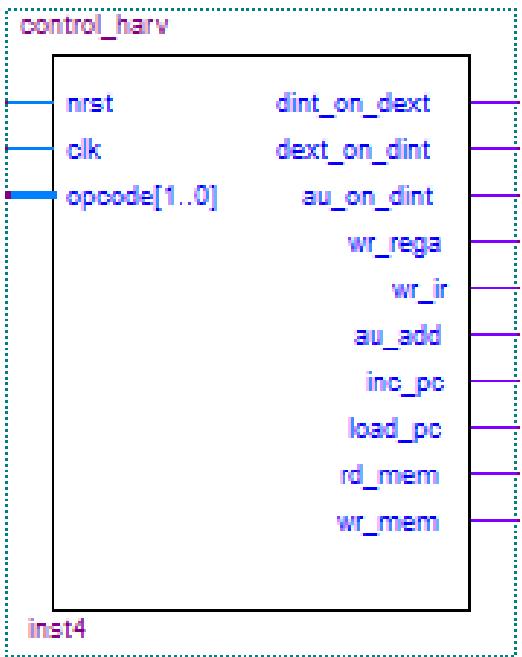
Os blocos de **controle**, memória **RAM** e memória **ROM** são diferentes daqueles da arquitetura von Neumann

# Exemplo: CPU somadora

## Arquitetura Harvard

### CPU somadora

#### Módulo de controle



```
library ieee;
use ieee.std_logic_1164.all;

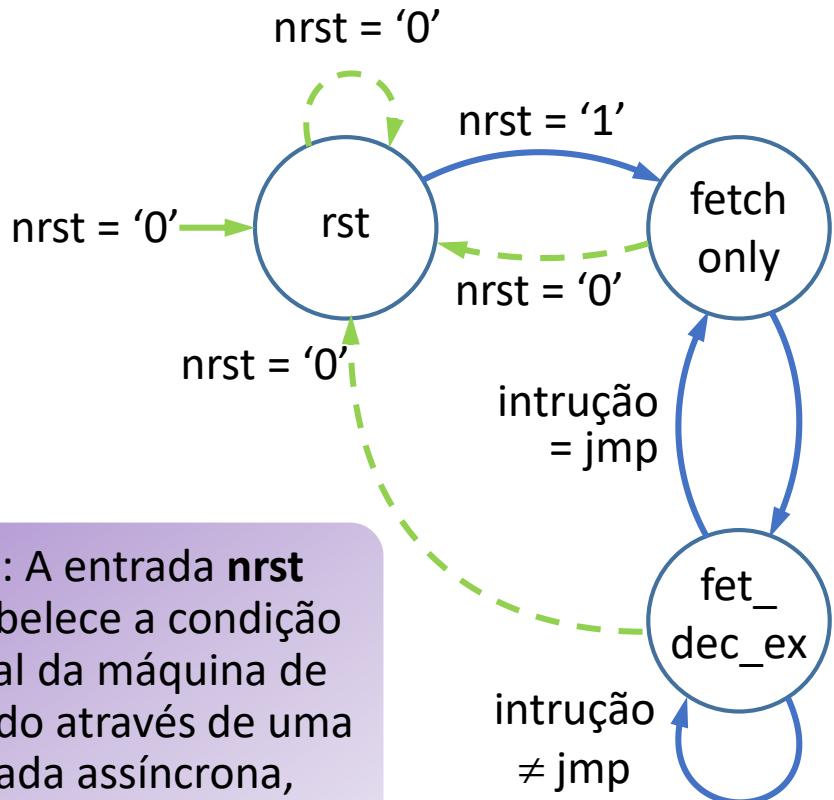
entity control_harv is
    port(
        nrst          : in std_logic;
        clk           : in std_logic;
        opcode        : in std_logic_vector(1 downto 0);
        dint_on_dext  : out std_logic;
        dext_on_dint  : out std_logic;
        au_on_dint    : out std_logic;
        wr_rega       : out std_logic;
        wr_ir         : out std_logic;
        au_add        : out std_logic;
        inc_pc        : out std_logic;
        load_pc       : out std_logic;
        rd_mem        : out std_logic;
        wr_mem        : out std_logic
    );
end entity;
```

# Exemplo: CPU somadora

## Arquitetura Harvard

### CPU somadora

#### Módulo de controle



```
architecture arch of control_harv is
    type state_type is (rst, fetch_only, fet_dec_ex);
    signal pres_state : state_type;
    signal next_state : state_type;

    constant op_ld : std_logic_vector(1 downto 0) := "00";
    constant op_sto : std_logic_vector(1 downto 0) := "01";
    constant op_add : std_logic_vector(1 downto 0) := "10";
    constant op_jmp : std_logic_vector(1 downto 0) := "11";

begin
    -- parte sequencial da FSM

process(nrst, clk)
begin
    if nrst = '0' then
        pres_state <= rst;
    elsif rising_edge(clk) then
        pres_state <= next_state;
    end if;
end process;
```

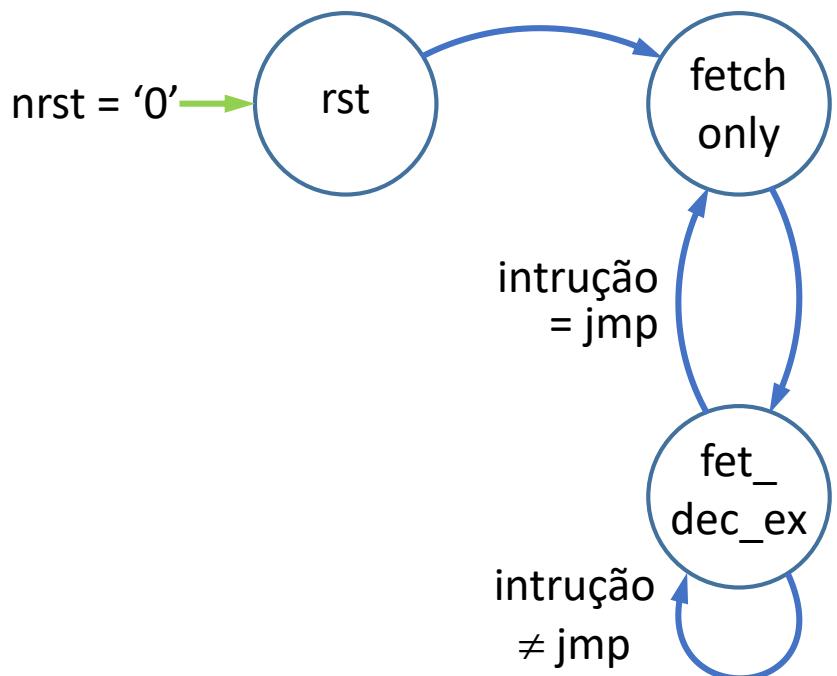
Continua na próxima página

# Exemplo: CPU somadora

## Arquitetura Harvard

### CPU somadora

#### Módulo de controle



```
-- parte combinacional da FSM
-----
process(nrst, pres_state, opcode)
begin
    -- valores default:
    dint_on_dext      <= '0';
    dext_on_dint      <= '0';
    au_on_dint        <= '0';
    wr_rega           <= '0';
    wr_ir             <= '0';
    au_add             <= '-';
    inc_pc             <= '0';
    load_pc            <= '0';
    rd_mem             <= '0';
    wr_mem             <= '0';
```

Don't care

Continua na próxima página

# Exemplo: CPU somadora

## Arquitetura Harvard

### CPU somadora

#### Módulo de controle

Sinal	Nome	reset	fetch only	fetch_decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	0	1	0	0	0
3	wr_ir	0	1	1	1	1	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	1	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	au_add	X	X	X	0	1	X
9	rd_mem	0	0	1	0	0	0
10	wr_mem	0	0	0	1	0	0

```
case pres_state is
when rst =>
    next_state <= fetch_only;

when fetch_only =>
    next_state <= fet_dec_ex;
    wr_ir           <= '1';
    inc_pc          <= '1';

when fet_dec_ex =>
    case opcode is
        -----
        -- load
        -----
        when op_ld =>
            next_state      <= fet_dec_ex;
            dext_on_dint   <= '1';
            wr_ir          <= '1';
            wr_rega         <= '1';
            inc_pc          <= '1';
            rd_mem          <= '1';
```

# Exemplo: CPU somadora

## Arquitetura Harvard

### CPU somadora

#### Módulo de controle

Sinal	Nome	reset	fetch only	fetch_decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	0	1	0	0	0
3	wr_ir	0	1	1	1	1	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	1	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	au_add	X	X	X	0	1	X
9	rd_mem	0	0	1	0	0	0
10	wr_mem	0	0	0	1	0	0

-- store

```
when op_sto =>
    next_state      <= fet_dec_ex;
    dint_on_dext   <= '1';
    wr_ir          <= '1';
    au_on_dint     <= '1';
    inc_pc          <= '1';
    au_add          <= '0';
    wr_mem          <= '1';
```

-- add

```
when op_add =>
    next_state      <= fet_dec_ex;
    wr_ir          <= '1';
    wr_rega         <= '1';
    au_on_dint     <= '1';
    inc_pc          <= '1';
    au_add          <= '1';
```

# Exemplo: CPU somadora

## Arquitetura Harvard

CPU somadora  
Módulo de controle

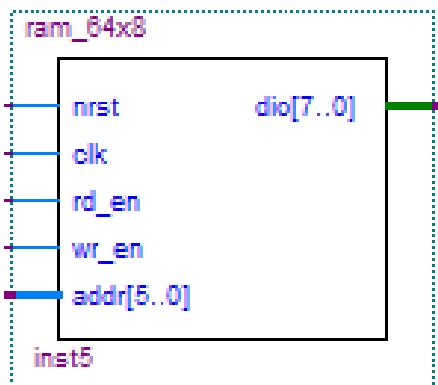
Sinal	Nome	reset	fetch only	fetch_decod_exec			
				LD	STO	ADD	JMP
1	dint_on_dext	0	0	0	1	0	0
2	dext_on_dint	0	0	1	0	0	0
3	wr_ir	0	1	1	1	1	0
4	wr_rega	0	0	1	0	1	0
5	au_on_dint	0	0	0	1	1	0
6	inc_pc	0	1	1	1	1	0
7	load_pc	0	0	0	0	0	1
8	au_add	X	X	X	0	1	X
9	rd_mem	0	0	1	0	0	0
10	wr_mem	0	0	0	1	0	0

```
-- jump|  
-----  
when op_jmp =>  
    next_state  
    load_pc  
    <= fetch_only;  
    <= '1';  
end case;  
end case;  
end process;  
end arch;
```

# Exemplo: CPU somadora

## Arquitetura Harvard

CPU somadora  
Memória RAM  
64 x 8



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

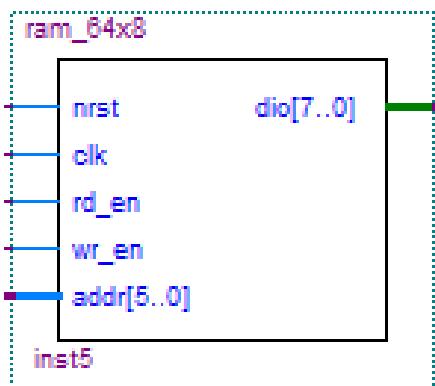
entity ram_64x8 is
    port(
        nrst      : in std_logic;
        clk       : in std_logic;
        rd_en     : in std_logic;
        wr_en     : in std_logic;
        addr      : in std_logic_vector(5 downto 0);
        dio       : inout std_logic_vector(7 downto 0)
    );
end entity;
```

Para conversão de tipos

# Exemplo: CPU somadora

## Arquitetura Harvard

CPU somadora  
Memória RAM  
64 x 8



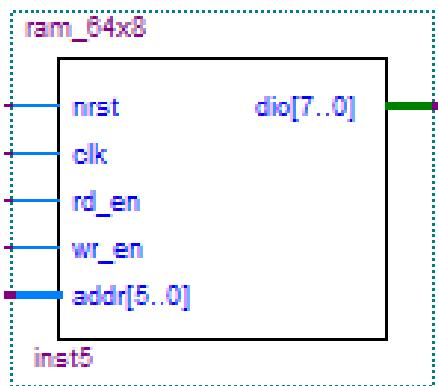
```
architecture arch of ram_64x8 is
    type ram_type is array (0 to 63) of
        std_logic_vector(7 downto 0);
    signal ram : ram_type;
    signal addr_int : integer range 0 to 63;

begin
    addr_int <= to_integer(unsigned(addr));
```

# Exemplo: CPU somadora

## Arquitetura Harvard

CPU somadora  
Memória RAM  
64 x 8



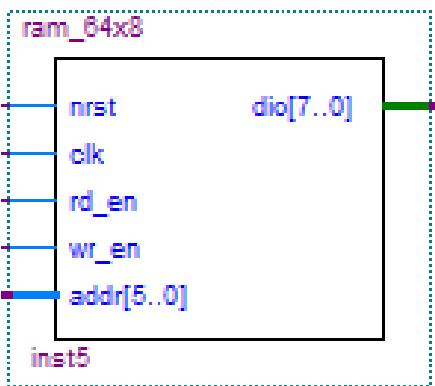
```
-- escreve na ram

process(nrst, clk)
begin
    if nrst = '0' then
        for i in 0 to 63 loop
            ram(i) <= (others => '0');
        end loop;
    elsif rising_edge(clk) then
        if wr_en = '1' then
            ram(addr_int) <= dio;
        end if;
    end if;
end process;
```

# Exemplo: CPU somadora

## Arquitetura Harvard

CPU somadora  
Memória RAM  
64 x 8



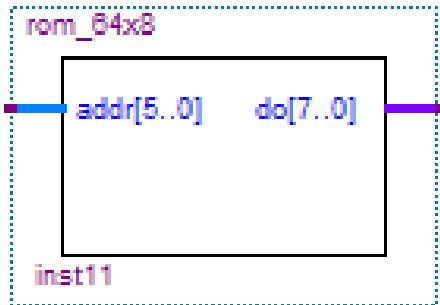
```
-- leitura na ram:
```

```
process(rd_en, addr_int, ram)
begin
    if rd_en = '1' then
        dio <= ram(addr_int);
    else
        dio <= (others => 'Z');
    end if;
end process;
end arch;
```

# Exemplo: CPU somadora

## Arquitetura Harvard

CPU somadora  
Memória ROM  
64 x 8



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

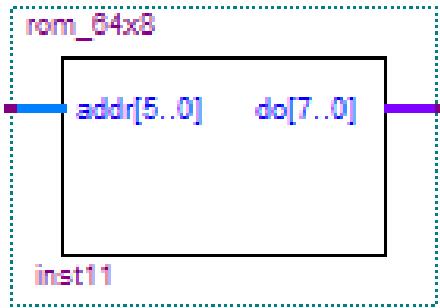
entity rom_64x8 is
    port(
        addr      : in std_logic_vector(5 downto 0);
        do       : out std_logic_vector(7 downto 0)
    );
end entity;
```

Para conversão de tipos

# Exemplo: CPU somadora

## Arquitetura Harvard

CPU somadora  
Memória ROM  
64 x 8



Programa a ser executado

```
architecture arch of rom_64x8 is

    signal addr_int : integer range 0 to 63;
    type rom_type is array (0 to 63) of
        std_logic_vector(7 downto 0);
    signal rom : rom_type := (
        "00100111", -- ld 27h
        "10000011", -- add 03h
        "01100111", -- sto 27h
        "11000000", -- jmp 00h;
        others => (others => '0')
    );
begin

    addr_int <= to_integer(unsigned(addr));

    -- leitura na memória:

    do <= rom(addr_int);

end arch;
```

# Exemplo: CPU somadora

## Arquitetura Harvard

CPU somadora

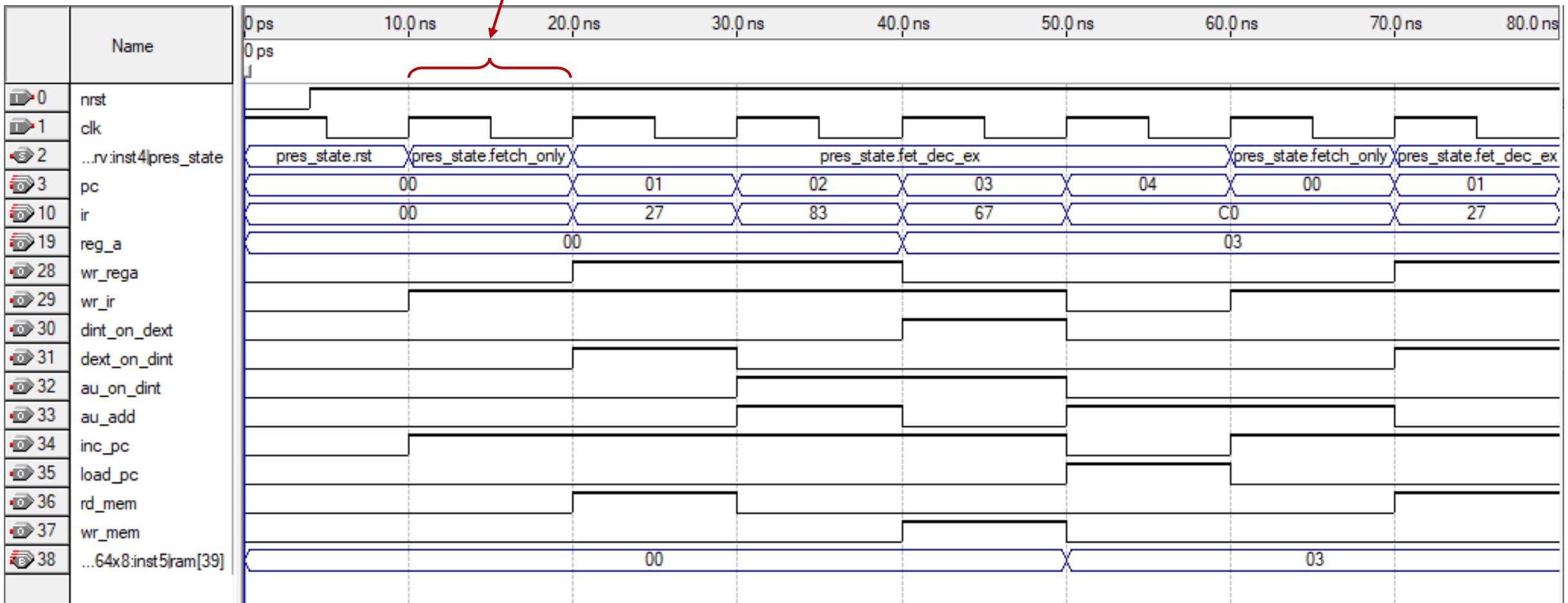
Programa exemplo

End.	Inst.	Instr. (bin)	Instr. (hexa)	Descrição
0	LD 27h	00100111	27	Carrega em reg_A o que está no endereço 39 (27h) da memória
1	ADD 03h	10000011	83	Soma 3 ao que está em reg_A
2	STO 27h	01100111	67	Escreve no endereço 39 (27h) da memória o que está em reg_A
3	JMP 00h	11000000	C0	Desvia o programa para o endereço 0

O programa é o mesmo usado para testar a arquitetura von Neumann, repetido aqui por conveniência

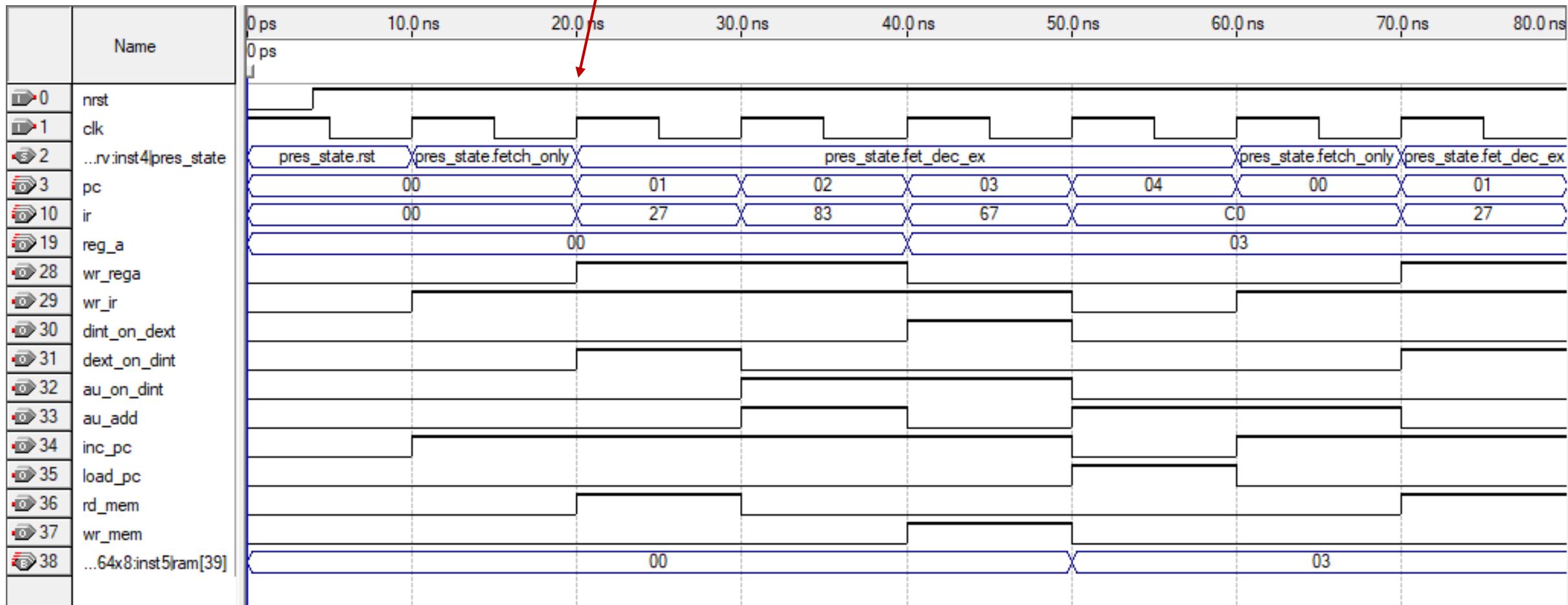
# Exemplo: CPU somadora - Arquitetura Harvard – simulação funcional

Buscando a instrução que está no endereço 0



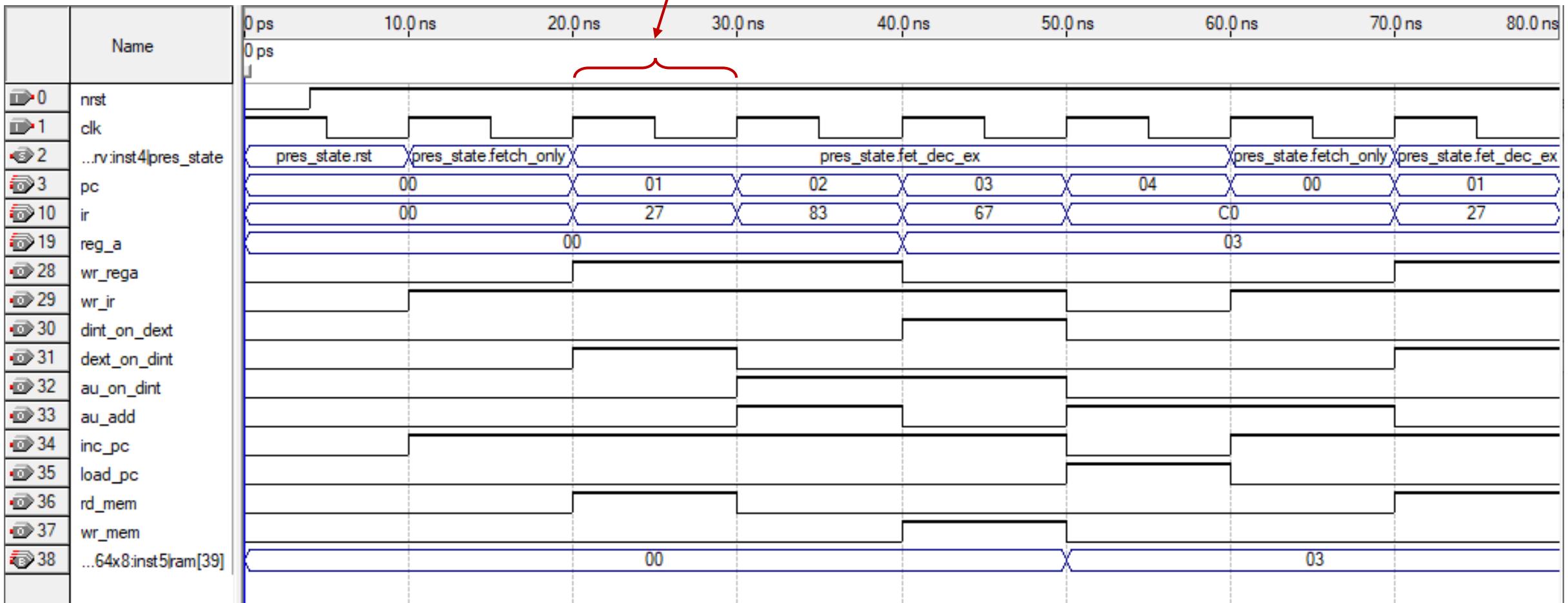
# Exemplo: CPU somadora - Arquitetura Harvard – simulação funcional

A instrução do endereço 0 (27h) foi para o IR



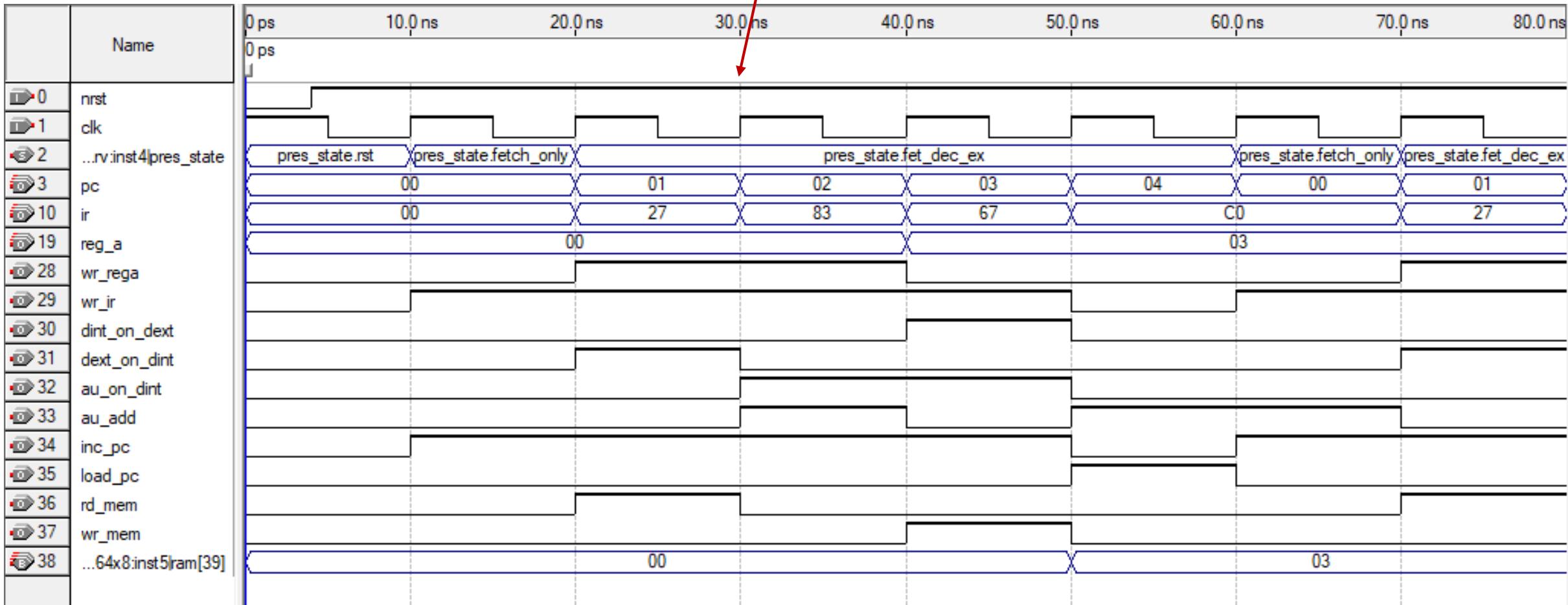
# Exemplo: CPU somadora - Arquitetura Harvard – simulação funcional

Decodificando e executando a instrução LD 27h  
Buscando a instrução que está no endereço 1



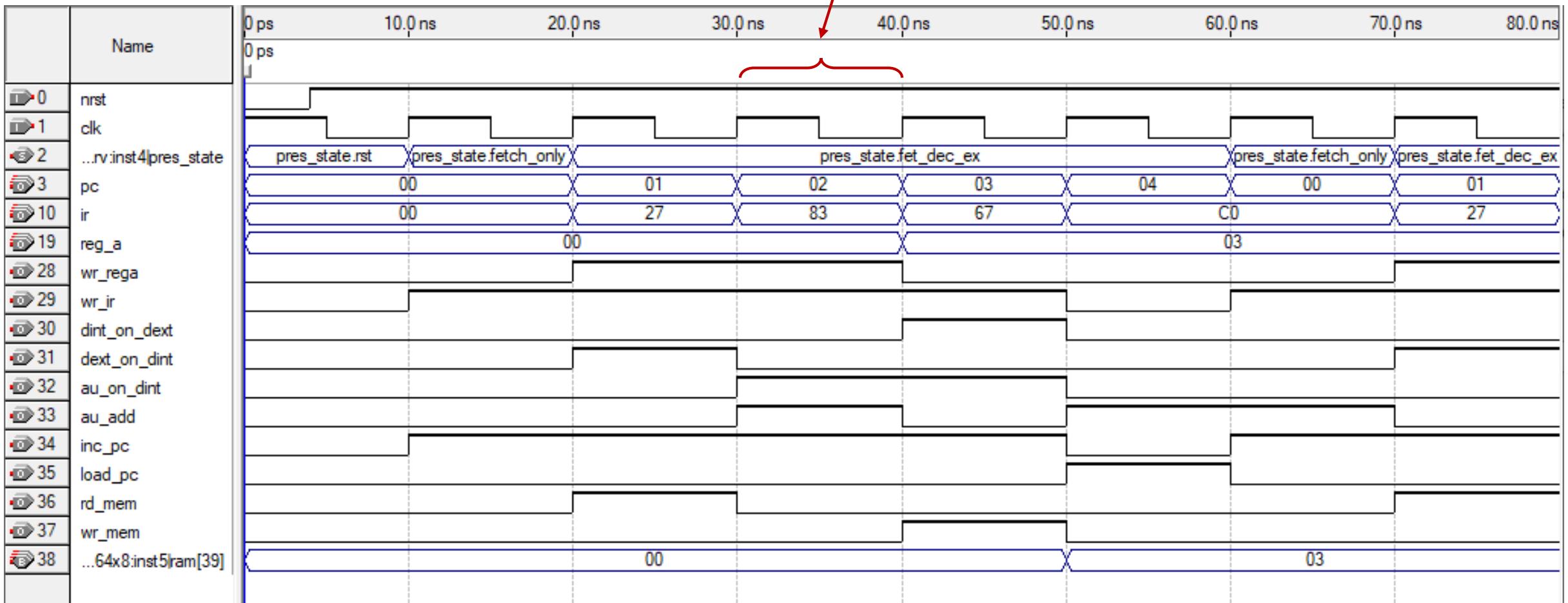
# Exemplo: CPU somadora - Arquitetura Harvard – simulação funcional

O que está no endereço 27h da memória foi para **reg\_a**  
A instrução do endereço 1 (83h) foi para o **IR**



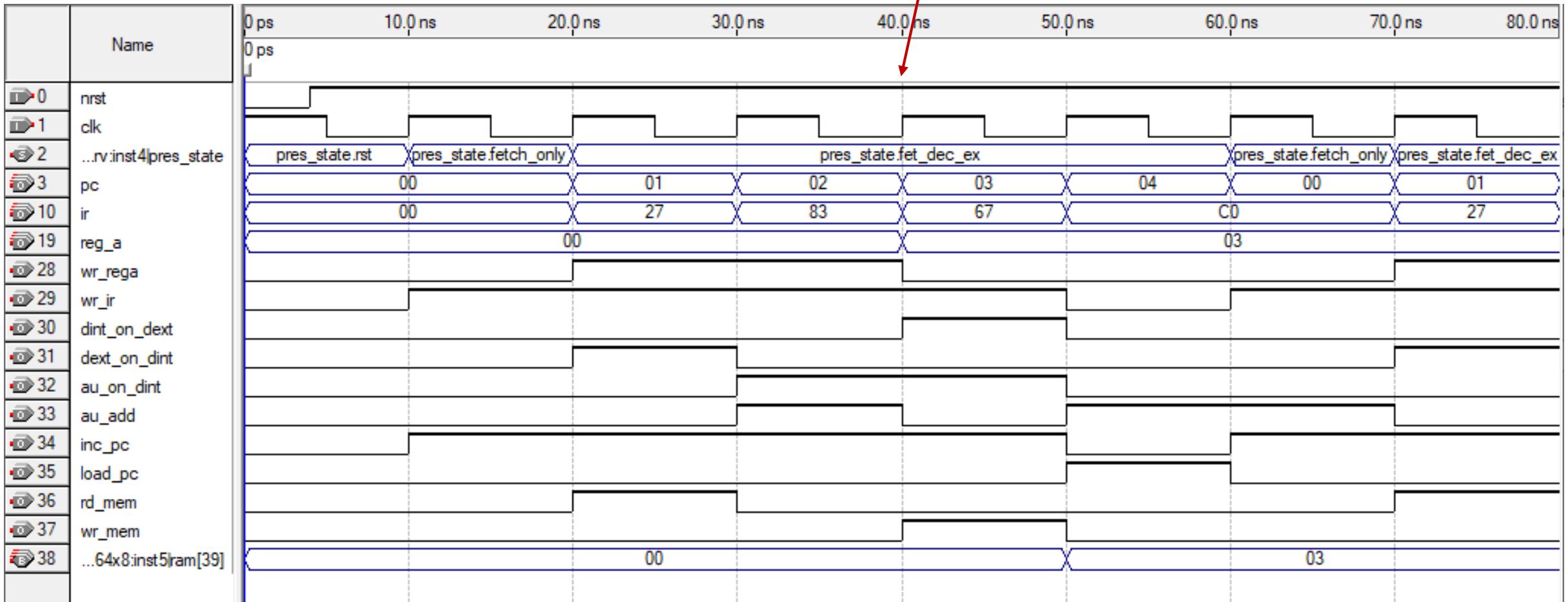
# Exemplo: CPU somadora - Arquitetura Harvard – simulação funcional

Decodificando e executando a instrução ADD 03h  
Buscando a instrução que está no endereço 2



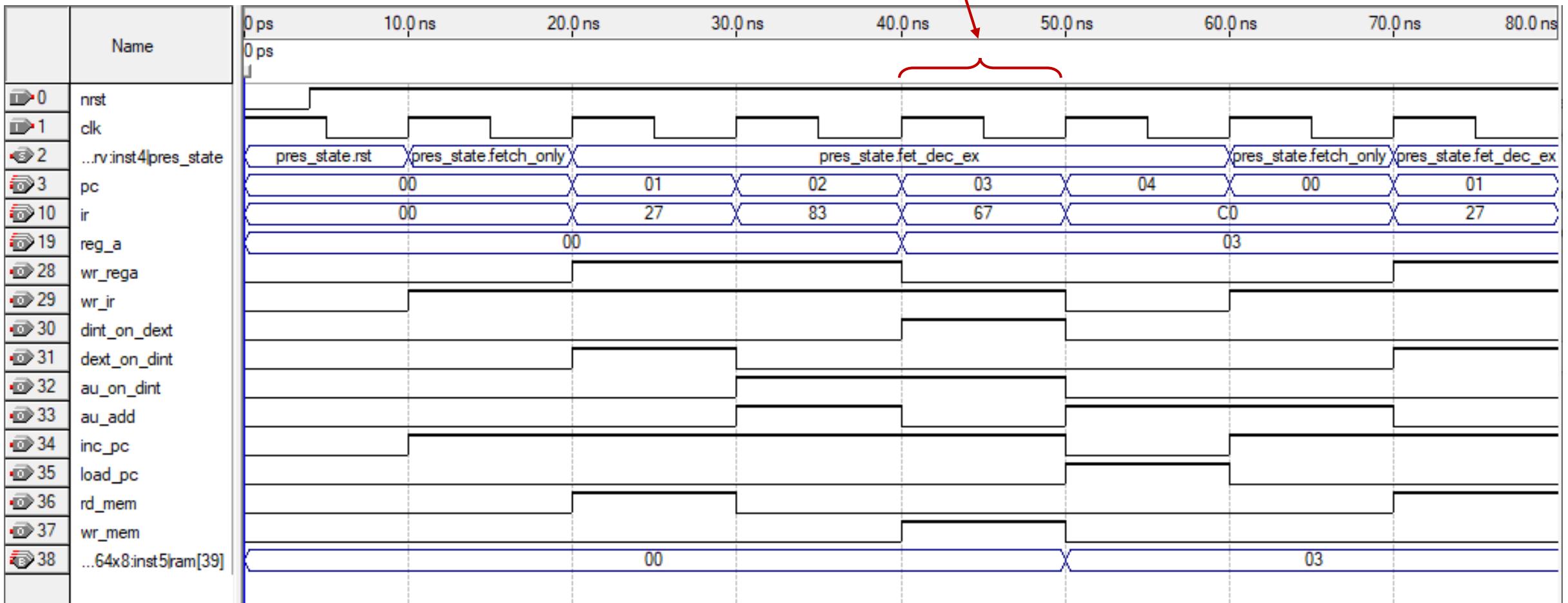
# Exemplo: CPU somadora - Arquitetura Harvard – simulação funcional

O resultado da soma foi para **reg\_a**  
A instrução do endereço 2 (67h) foi para o IR



# Exemplo: CPU somadora - Arquitetura Harvard – simulação funcional

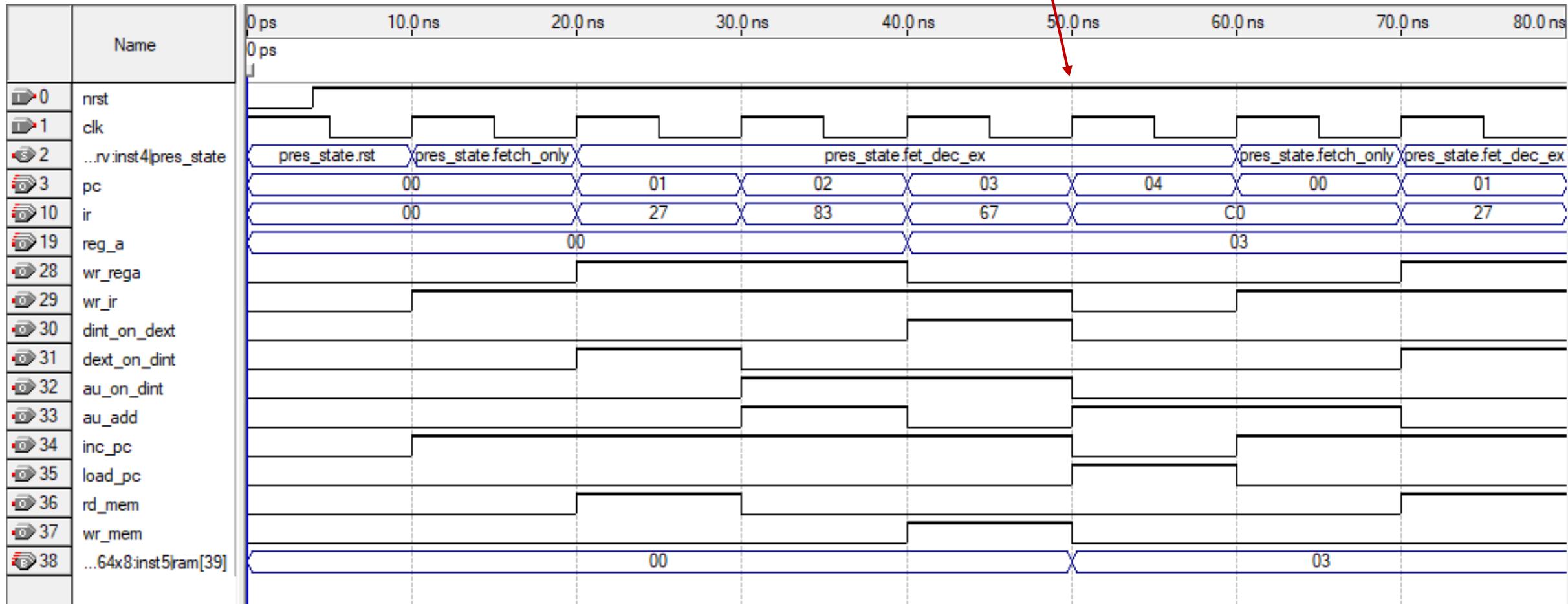
Decodificando e executando a instrução STO 27h  
Buscando a instrução que está no endereço 3



# Exemplo: CPU somadora - Arquitetura Harvard – simulação funcional

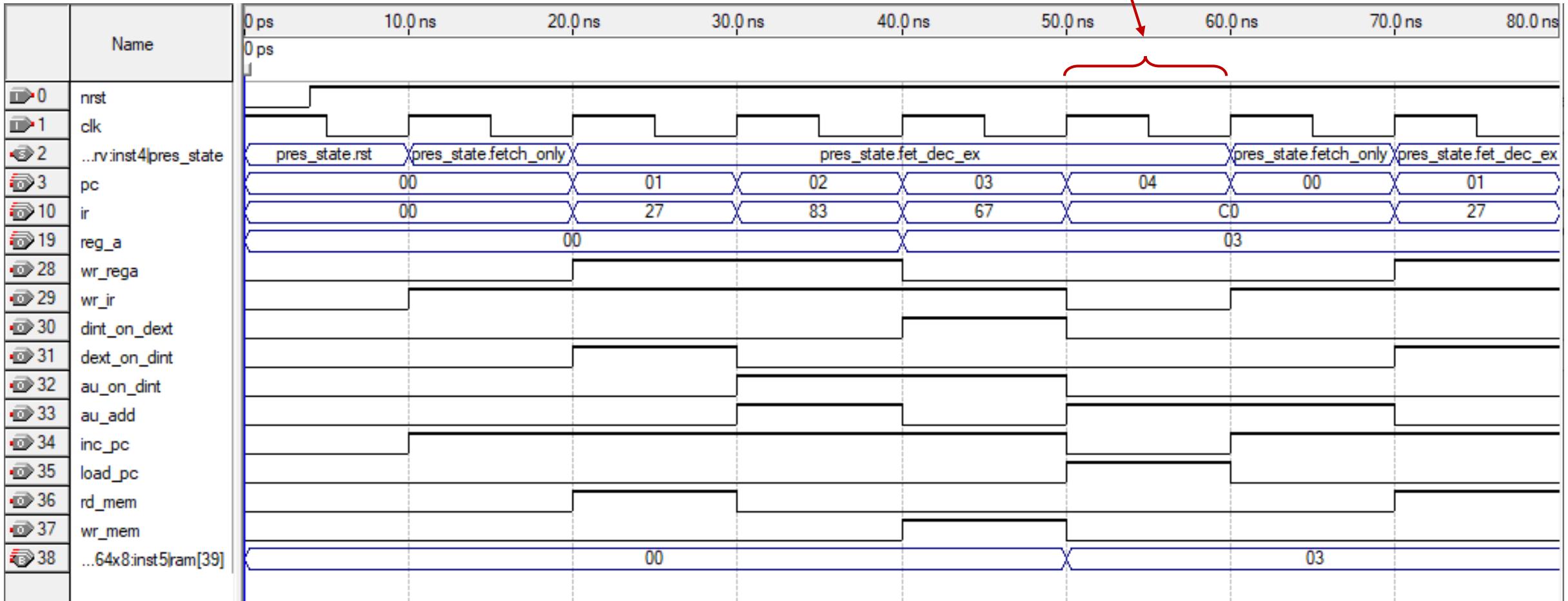
O que está em **reg\_a** foi para o endereço 27h da memória

A instrução do endereço 3 (C0h) foi para o **IR**

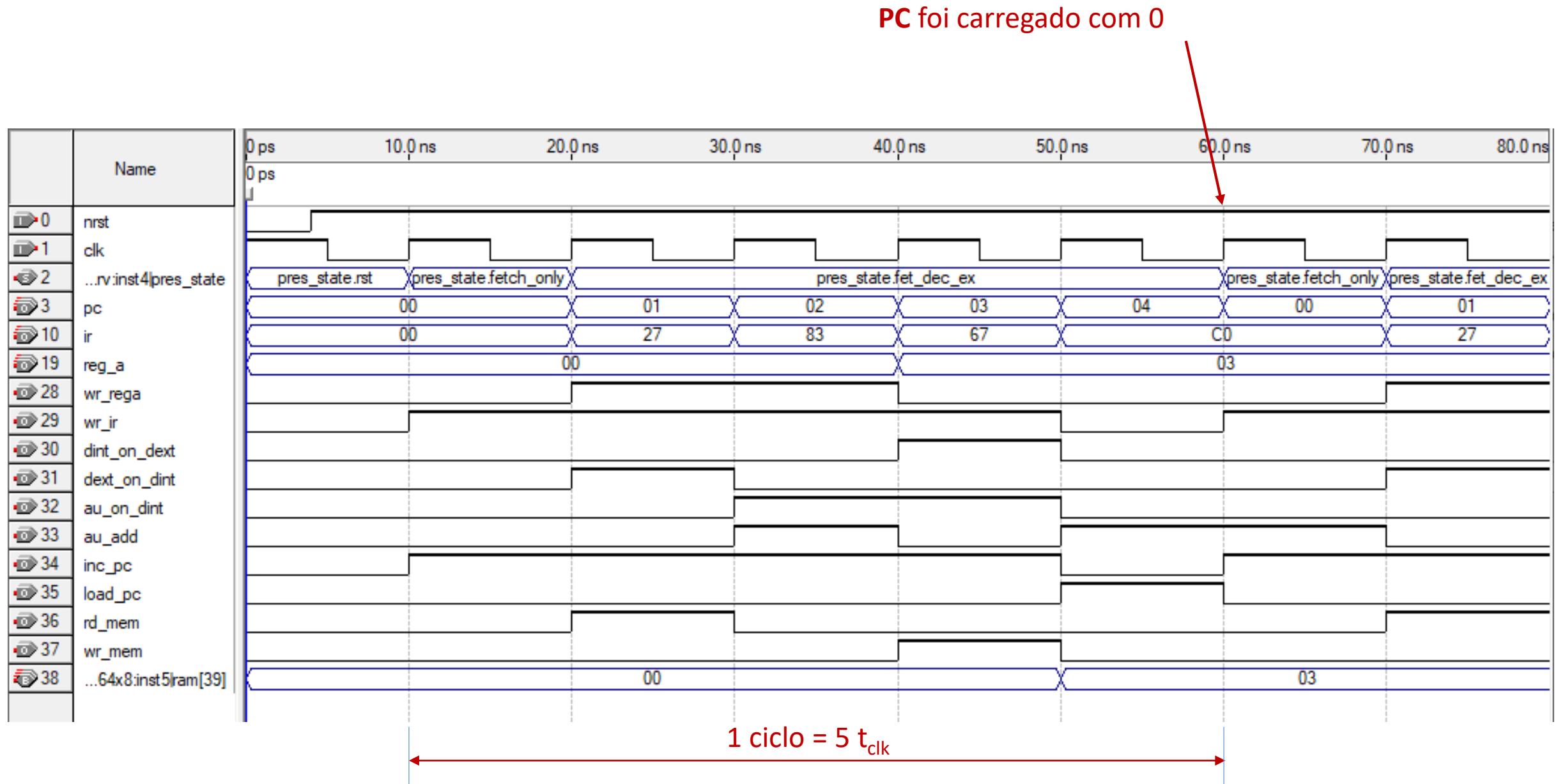


# Exemplo: CPU somadora - Arquitetura Harvard – simulação funcional

Decodificando e executando a instrução JMP 00h

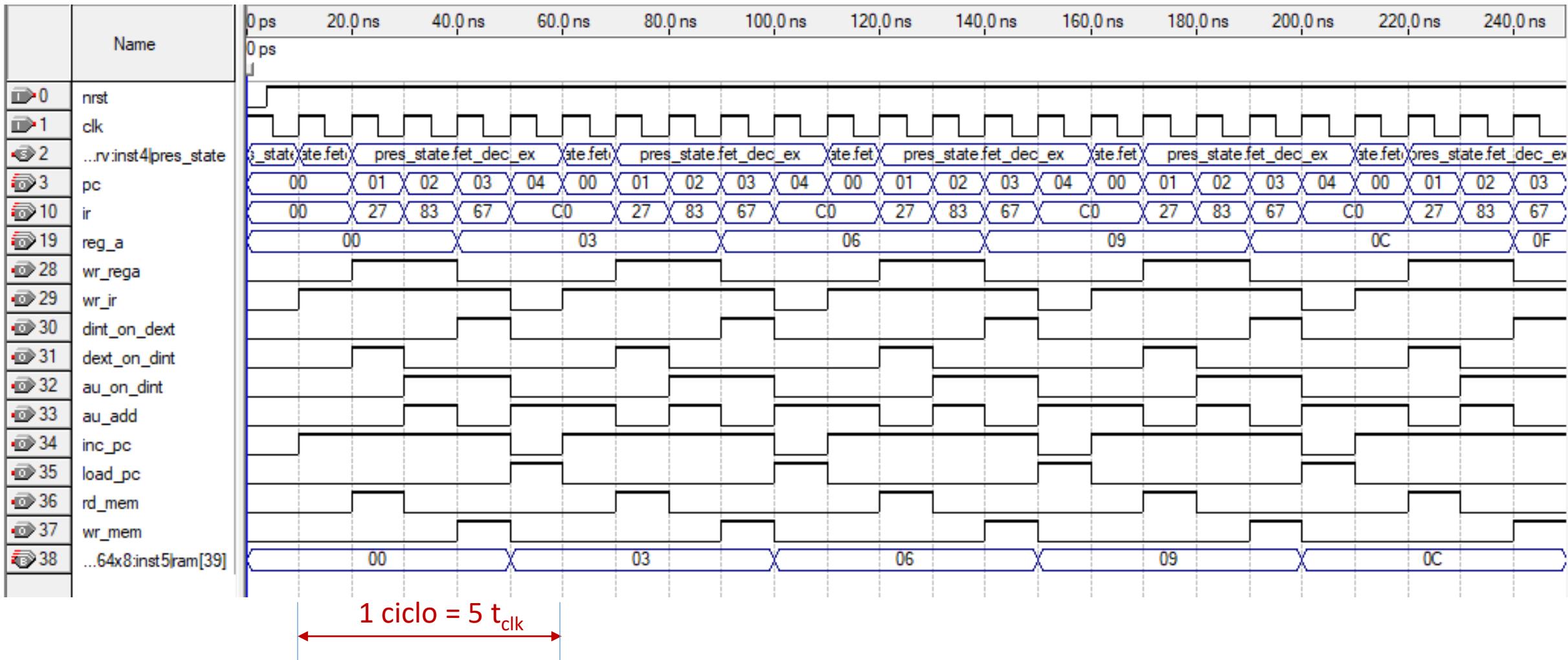


# Exemplo: CPU somadora - Arquitetura Harvard – simulação funcional



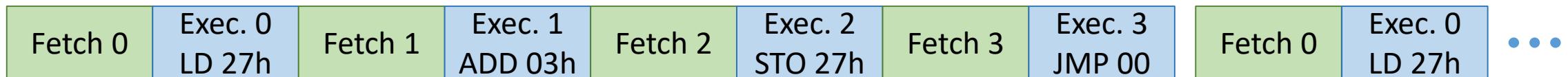
# Exemplo: CPU somadora - Arquitetura Harvard – simulação funcional

Em outra escala de tempo:

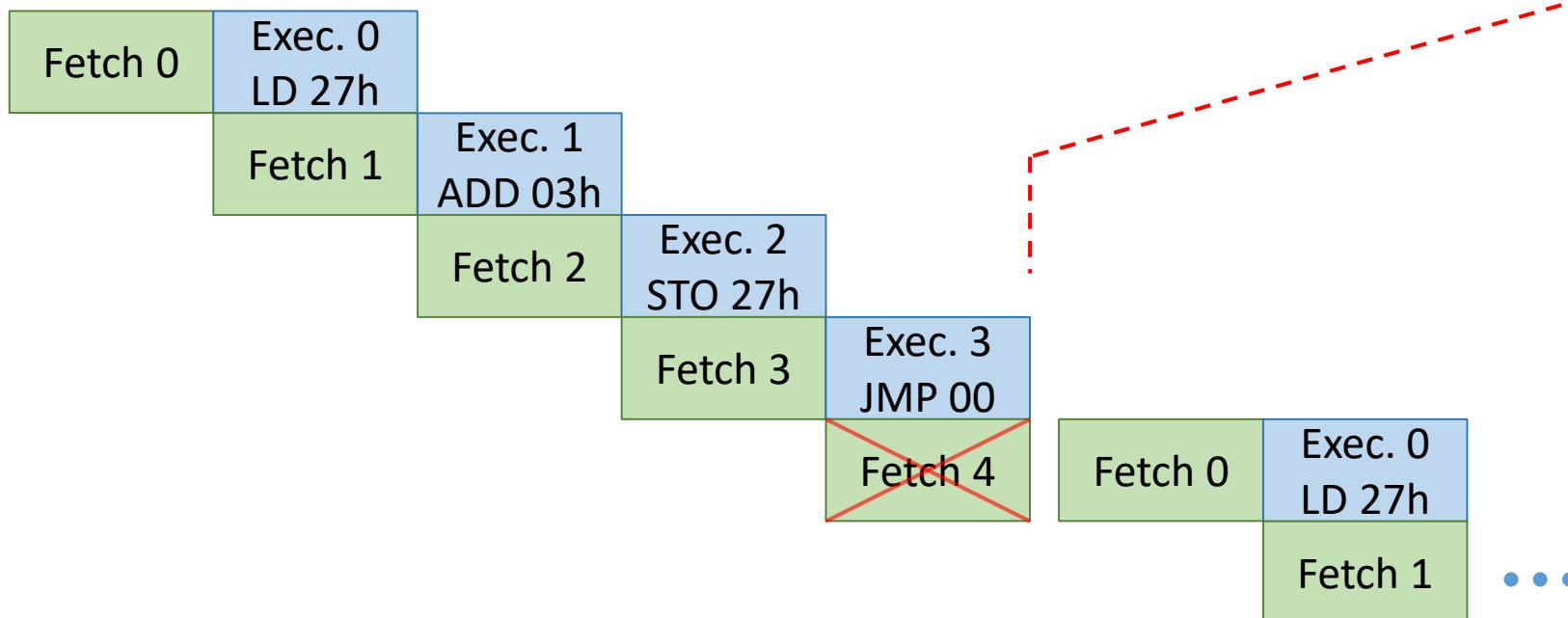


# Exemplo CPU somadora

Arquitetura von Neumann:



Arquitetura Harvard:



# Exercício

Um micro controlador tem as seguintes características: arquitetura Harvard, um ciclo por instrução (exceto as instruções que carregam um novo valor no PC), 256 palavras de ROM com 11 bits de largura, 256 bytes de RAM e um registrador de 8 bits, chamado de **reg\_a**. O seu conjunto de instruções possui as seguintes instruções:

Código	Mnem.	Descrição
000 iiiiiji	ADD	Soma o valor presente no registrador reg_a com o valor imediato iiiiiji. O resultado é armazenado no registrador reg_a.
001 iiiiiji	AND	Faz a operação AND entre o valor presente no registrador reg_a com o valor imediato iiiiiji. O resultado é armazenado no registrador reg_a.
010 iiiiiji	OR	Faz a operação OR entre o valor presente no registrador reg_a com o valor imediato iiiiiji. O resultado é armazenado no registrador reg_a.
011 iiiiiji	XOR	Faz a operação XOR entre o valor presente no registrador reg_a com o valor imediato iiiiiji. O resultado é armazenado no registrador reg_a.
100 iiiiiji	MVI	Carrega no registrador reg_a o valor iiiiiji.
101 aaaaaaaaa	LDM	Carrega no registrador reg_a o conteúdo do endereço aaaaaaaaa da RAM.
110 aaaaaaaaa	STO	Armazena no endereço aaaaaaaaa da RAM o conteúdo do registrador reg_a.
111 aaaaaaaaa	JMP	Desvia a execução do programa para o endereço aaaaaaaaa da memória de programa (ROM)

# Exercício

---

Para esse processador:

1. Modifique o caminho de dados do exemplo da CPU somadora apresentado anteriormente, adequando a largura de cada barramento usado nesse caminho de dados ao exercício proposto;
2. Faça uma tabela das funções necessárias da ULA atribuindo valores binários a cada uma das funções.
3. Faça uma tabela com os valores de cada sinal de controle, para todos os estados da FSM e instruções.
4. Verifique quais os blocos do exemplo da CPU somadora podem ser aproveitados sem modificações e quais devem ser modificados.
5. Usando linguagem VHDL, crie os blocos necessários.
6. Faça a integração de todos os blocos e teste o projeto usando simulação funcional.

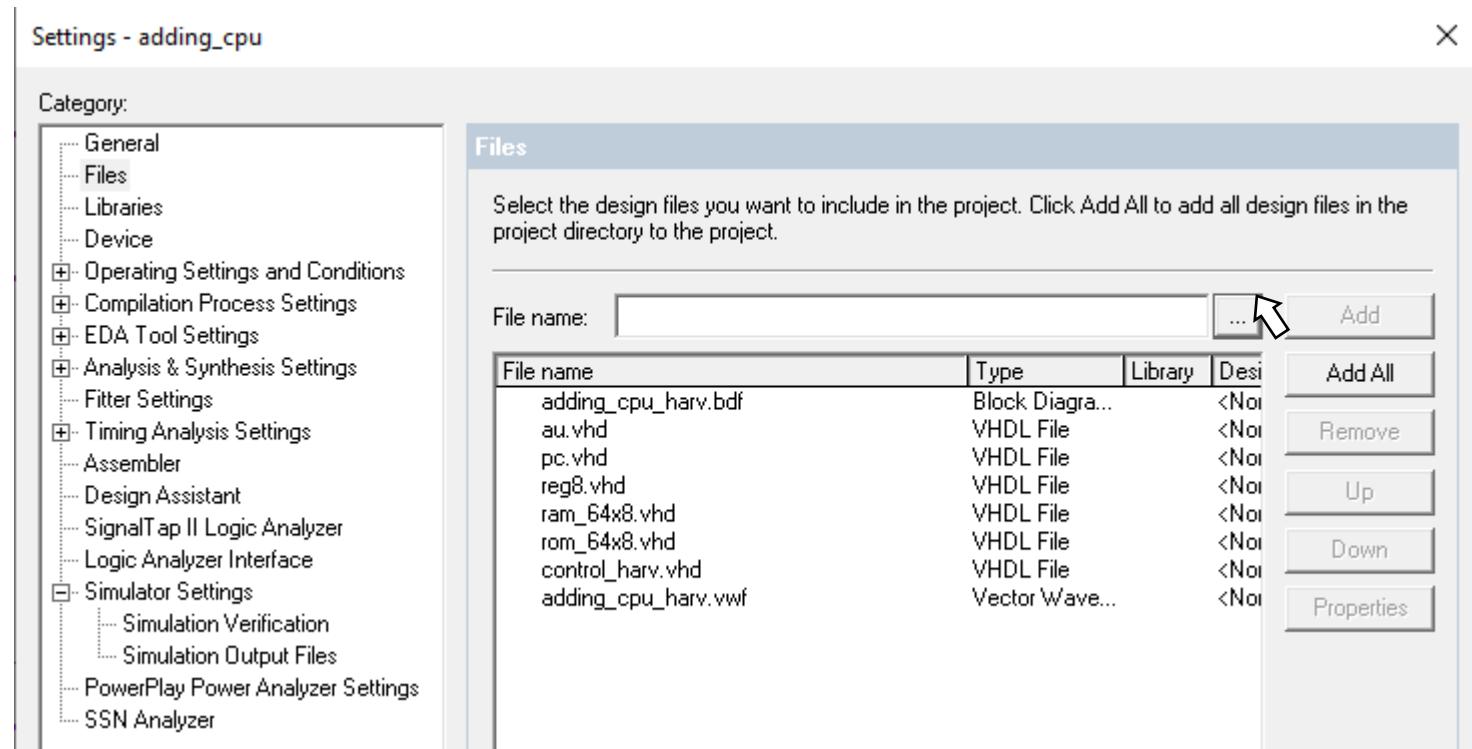
# Obs. Instaciando blocos

Ao instanciar blocos em uma arquitetura hierárquica, se o arquivo .vhd do bloco estiver em uma pasta diferente da pasta do projeto, é necessário adicionar o arquivo no projeto.

Para isso:

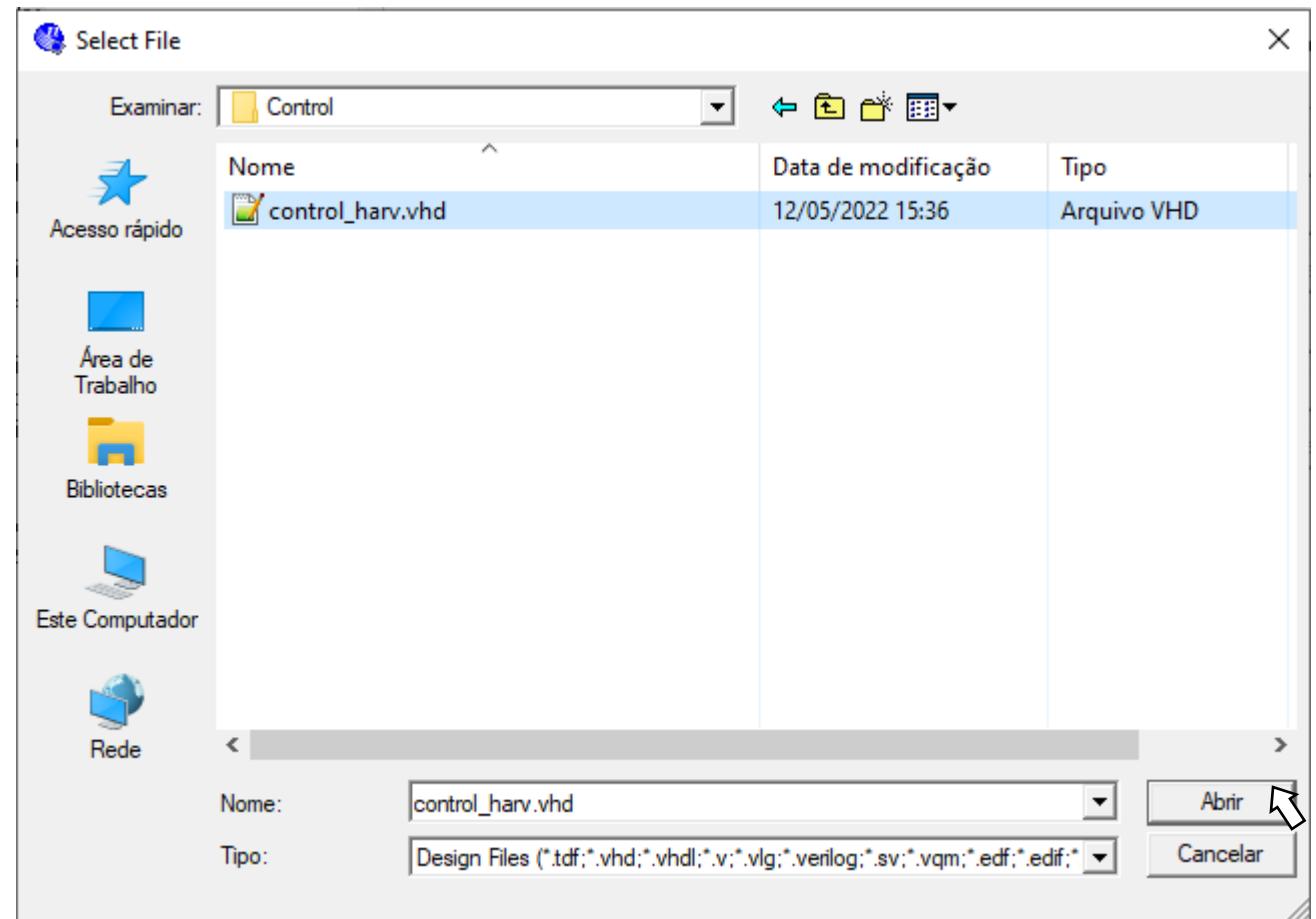
No menu “[Project](#)”, escolher a opção “[Add/Remove Files in Project...](#)”;

Clicar no botão “...” e procurar a pasta que contém o arquivo do bloco



# Obs. Instaciando blocos

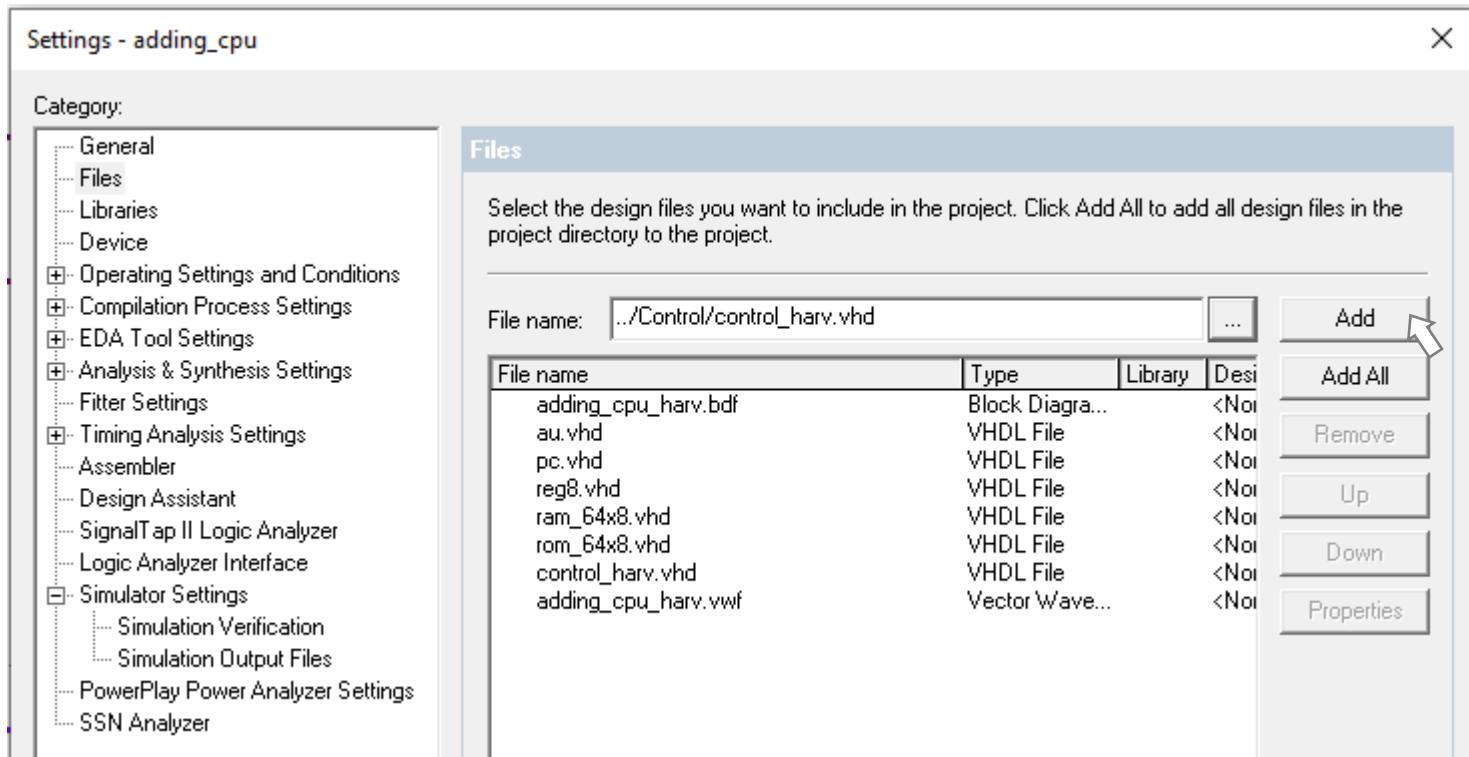
Selecionar o arquivo **.vhd** e, depois,  
clicar no botão “**abrir**”



# Obs. Instaciando blocos

Clicar no botão “Add”

Repetir para outros arquivos, se necessário.





*Fim*