

Pontifícia Universidade Católica de Campinas
Centro de Ciências Exatas, Ambientais e de Tecnologias
Faculdade de Engenharia de Computação

Agatha Talita Acuña Honório
Vitor Rodrigo Vezani

Arquitetura de Computadores
Projeto #1
Professor Edmar Roberto Santana de Rezende

Campinas, Abril de 2012

Introdução

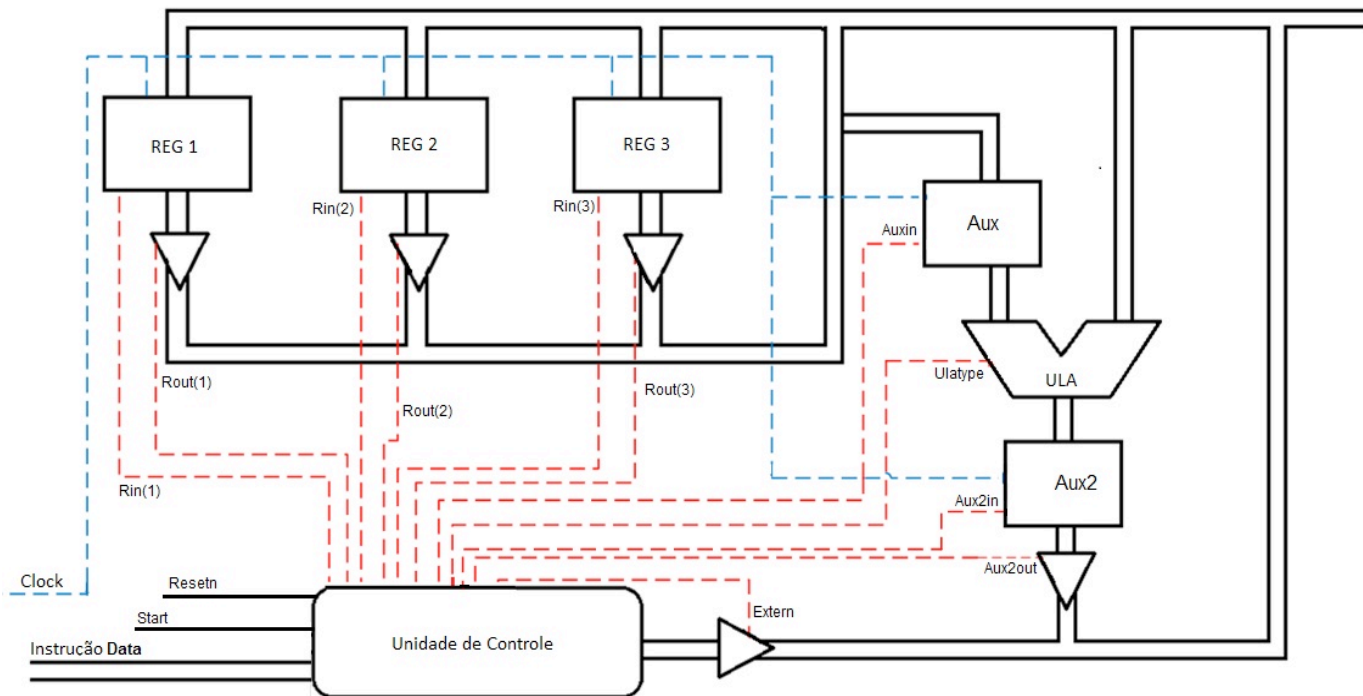
Este primeiro projeto tem objetivo implementar e simular uma CPU básica que realize quatro instruções: MOV , ADD, SUB e XCHG. A instrução MOV deve mover um valor imediato para um registrador, a instrução ADD deve somar o conteúdo de dois registradores, a instrução SUB deve subtrair o conteúdo entre dois registradores e a instrução XCHG deve trocar o conteúdo de dois registradores entre si. Para implementar, serão necessários uma Unidade de controle

Para realizar essas instruções foi preciso de uma Unidade de Controle, feita com uso de uma máquina de estados, e uma Unidade Lógica Aritmética (ULA).

1. Descrição textual do projeto com a topologia da CPU

O funcionamento da CPU será feita da seguinte forma: Com a entrada de DATA na Unidade de Controle, será feita a leitura, e dependendo do seu valor, será determinada a instrução que deverá ocorrer, os registradores e valor imediato envolvido nessa instrução, passando para os próximos estados da máquina de estado. As variáveis de sinais Rin(1), Rin(2), Rin(3), Rout(1), Rout(2), Rout(3), Auxin, Aux2in e Aux2out são sinais que controlam o momento em que o registrador correspondente estará aberto ou fechado para escrita e/ou leitura da BusWires(barramento), quando elas possuírem valor 1, estão abertas, ou seja, se forem Rin(1), Rin(2), Rin(3), Auxin ou Aux2in, elas estarão prontas para ler o conteúdo que estiver na BusWires, e se forem Rout(1), Rout(2), Rout(3), Aux2out, estão prontas para escrever seu conteúdo na BusWires.

O sinal de clock tem período de 40ns e a cada sinal é executado um estado da maquina uma vez dado o Start. Extern é um sinal usado para gravar o valor do imediato da CPU para o BusWires; Ulatype tem a função de informar a ULA qual operação deve ser realizada, se possuir valor 1 é realizada a soma, se possuir valor 0 é realizada a subtração; e Resetn serve para resetar a maquina de estados para começar do início.



2.Especificação

2.1. Registradores

Foram utilizados cinco registradores (reg1, reg2 e os registradores auxiliares reg3, Aux e Aux2) de 4 bits cada um. O valor de DATA recebido contém o endereço de cada registrador que será utilizado na posição (5 DOWNT0 4). Todos os registradores utilizados têm tamanho de 4 bits.

DATA(5 DOWNT0 4)	Registrador
00	reg1
01	reg2

(registradores endereçáveis.)

2.2. Formato das instruções (OPCODE)

As instruções (OPCODE) foram determinadas da seguinte forma: O valor passado por DATA possui 8 bits, sendo, da esquerda para a direita (7 DOWNT0 6) 2 bits representando a identificação do OPCODE :

OPCODE(7 DOWNT0 6)	Função
00	Mov
01	Add
10	Sub
11	Xchg

Imediato:

O valor imediato foi projetado com tamanho 4 bits e é contido na posição (3 DOWNT0 0) do vetor DATA .
i.exemplo geral : DATA[00010111] = Mover imediato 0111 (7) no registrador 1.

2.3. Unidade de Controle Máquina de Estados

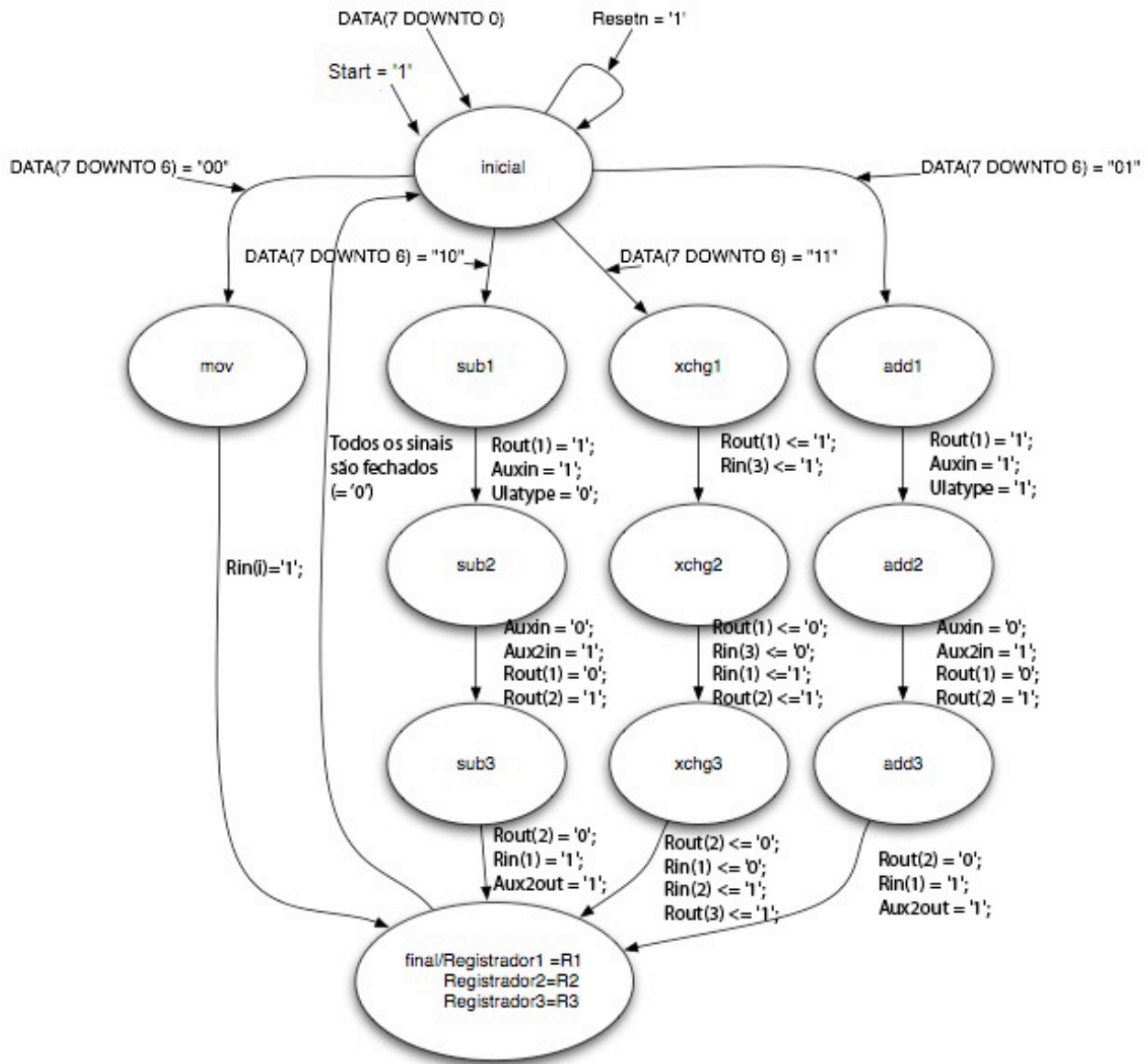


Tabela de Estados

Para poder apresentar detalhadamente, a tabela de estados foi dividida em partes, que serão mostradas a seguir:

- Estado seguinte, dependendo do valor do OPCODE

Estado Atual	Próximo Estado (start = 1)			
	OPCODE[00]	OPCODE[01]	OPCODE[10]	OPCODE[11]
INICIAL	MOV	ADD1	SUB1	XCHG1
MOV	FINAL			
ADD1		ADD2		
ADD2		ADD3		
ADD3		FINAL		
SUB1			SUB2	
SUB2			SUB3	
SUB3			FINAL	
XCHG1				XCHG2
XCHG2				XCHG3
XCHG3				FINAL
FINAL	INICIAL	INICIAL	INICIAL	INICIAL

- Valores das saídas quando o a instrução for MOV, este pode ser movido para 3 registradores.

Estado Atual	Saídas											
	OPCODE [00] e DATA (5 downto 4) = 00											
	Rin(1)	Rout(1)	Rin(2)	Rout(2)	Rin(3)	Rout(3)	Auxin	Aux2in	Aux2out	Ulatype	Extern	
INICIAL											1	
MOV	1											
ADD1												
ADD2												
ADD3												
SUB1												
SUB2												
SUB3												
XCHG1												
XCHG2												
XCHG3												
FINAL	0										0	
Estado Atual	Saídas											
	OPCODE [00] e DATA (5 downto 4) = 01											
	Rin(1)	Rout(1)	Rin(2)	Rout(2)	Rin(3)	Rout(3)	Auxin	Aux2in	Aux2out	Ulatype	Extern	
INICIAL											1	
MOV			1									
ADD1												
ADD2												
ADD3												
SUB1												
SUB2												
SUB3												
XCHG1												
XCHG2												
XCHG3												
FINAL			0								0	
Estado Atual	Saídas											
	OPCODE [00] e DATA (5 downto 4) = 10											
	Rin(1)	Rout(1)	Rin(2)	Rout(2)	Rin(3)	Rout(3)	Auxin	Aux2in	Aux2out	Ulatype	Extern	
INICIAL											1	
MOV					1							
ADD1												
ADD2												
ADD3												
SUB1												
SUB2												
SUB3												
XCHG1												
XCHG2												
XCHG3												
FINAL					0						0	

- Valores das saídas quando o a instrução for ADD

Estado Atual	Saídas										
	OPCODE [01]										
	Rin(1)	Rout(1)	Rin(2)	Rout(2)	Rin(3)	Rout(3)	Auxin	Aux2in	Aux2out	Ulatype	Extern
INICIAL											
MOV											
ADD1		1					1			1	
ADD2		0		1			0	1			
ADD3			1	0					1		
SUB1											
SUB2											
SUB3											
XCHG1											
XCHG2											
XCHG3											
FINAL	0	0	0	0	0	0	0	0	0		

- Valores das saídas quando o a instrução for SUB

Estado Atual	Saídas										
	OPCODE [10]										
	Rin(1)	Rout(1)	Rin(2)	Rout(2)	Rin(3)	Rout(3)	Auxin	Aux2in	Aux2out	Ulatype	Extern
INICIAL											
MOV											
ADD1											
ADD2											
ADD3											
SUB1		1					1			0	
SUB2		0		1			0	1			
SUB3	1			0					1		
XCHG1											
XCHG2											
XCHG3											
FINAL	0	0	0	0	0	0	0	0	0		

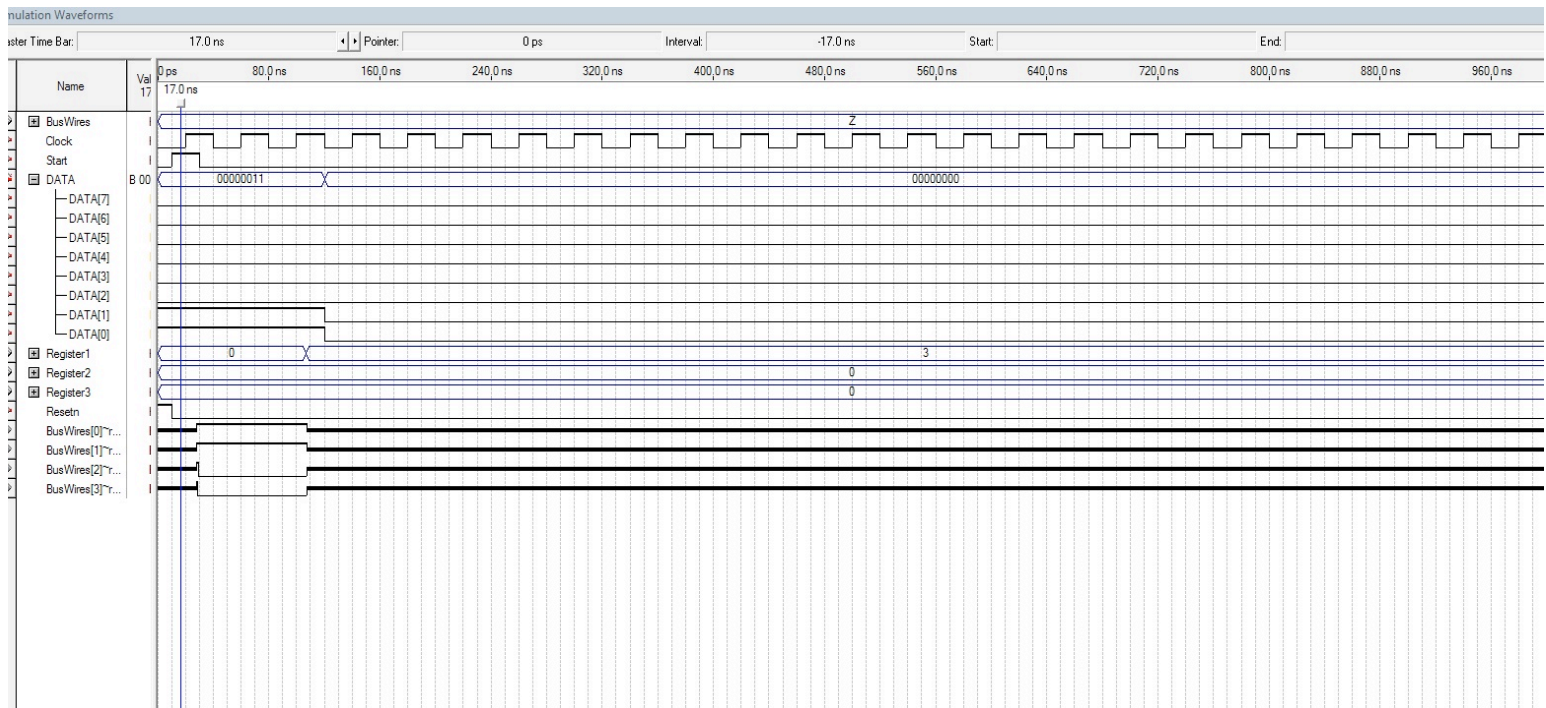
- Valores das saídas quando o a intrução for XCHG

Estado Atual	Saídas										
	OPCODE [11]										
	Rin(1)	Rout(1)	Rin(2)	Rout(2)	Rin(3)	Rout(3)	Auxin	Aux2in	Aux2out	Ulatype	Extern
INICIAL											
MOV											
ADD1											
ADD2											
ADD3											
SUB1											
SUB2											
SUB3											
XCHG1		1			1						
XCHG2	1	0		1	0						
XCHG3	0		1	0		1					
FINAL	0	0	0	0	0	0					

3. Resultados

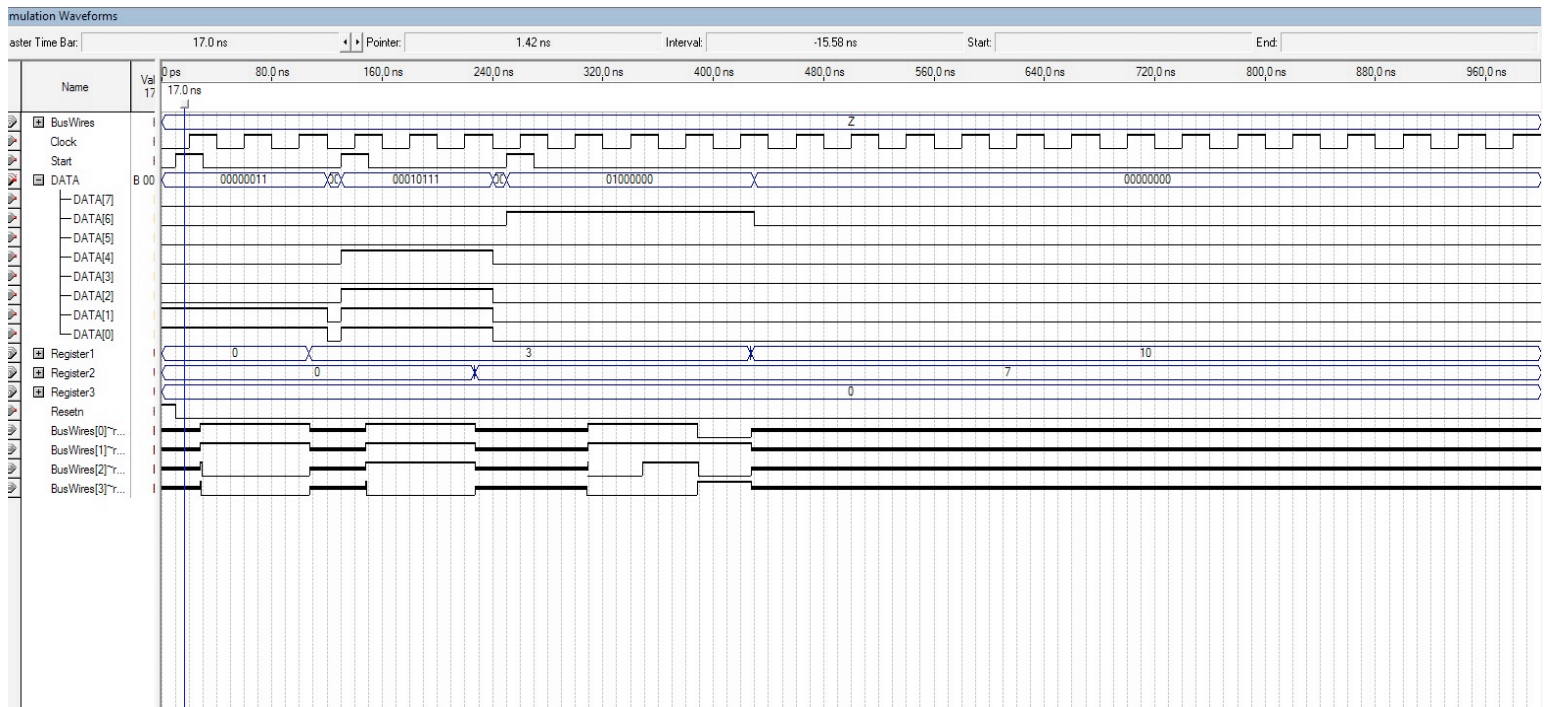
3.2 Resultados e discussão

Para a análise foram coletados 5 imagens que demonstram a simulação do projeto de uma CPU básica e suas quatro instruções.



[img1]

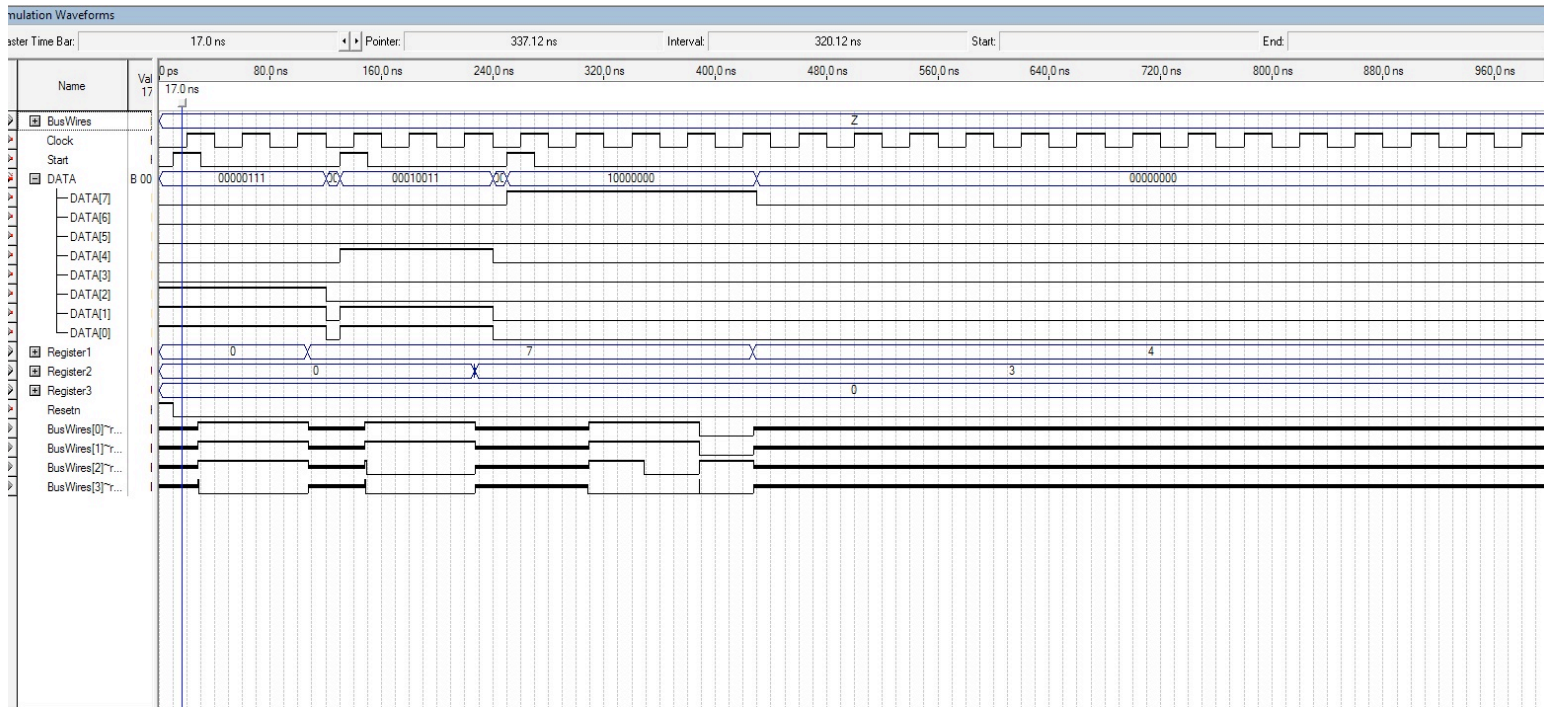
Na imagem 1



[img2]

Na imagem 2

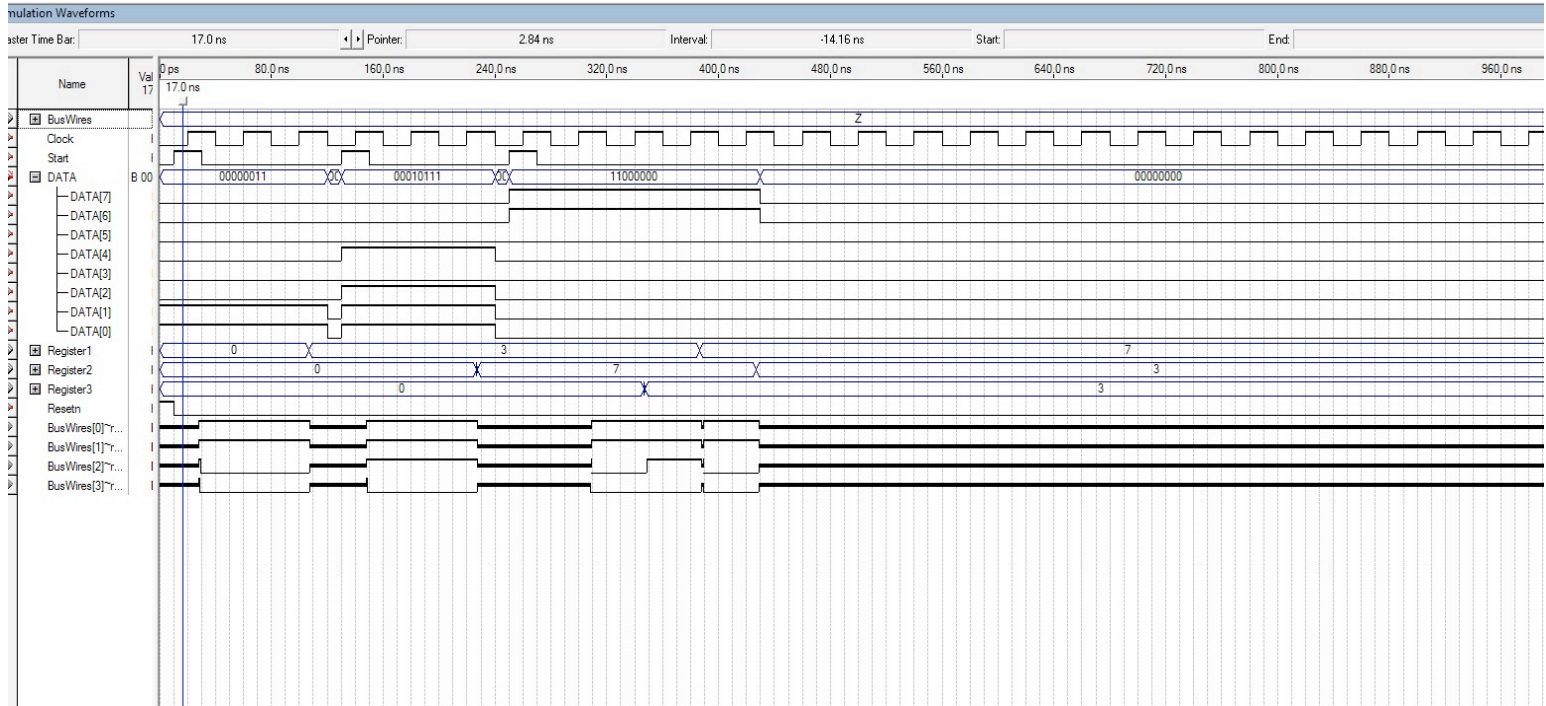
Para a demonstração principal da instrução ADD a entrada DATA é [01000000] onde o OPCODE [01] simboliza tal operação, são necessários 5 ciclos de clock que executam os 5 estados do ADD (inicial, add1, add2, add3, final) que tem como função somar o valor do registrador 1 com o valor do registrador 2 e guardar o resultado no registrador 1.



[img3]

Na imagem 3[img3] a entrada DATA é [00000111] nos 3 primeiros clocks que realiza um MOV do valor 3 para o registrador 1, depois a entrada DATA é [00000111] pelos próximos 3 clocks e o valor 7 é movido no registrador 2.

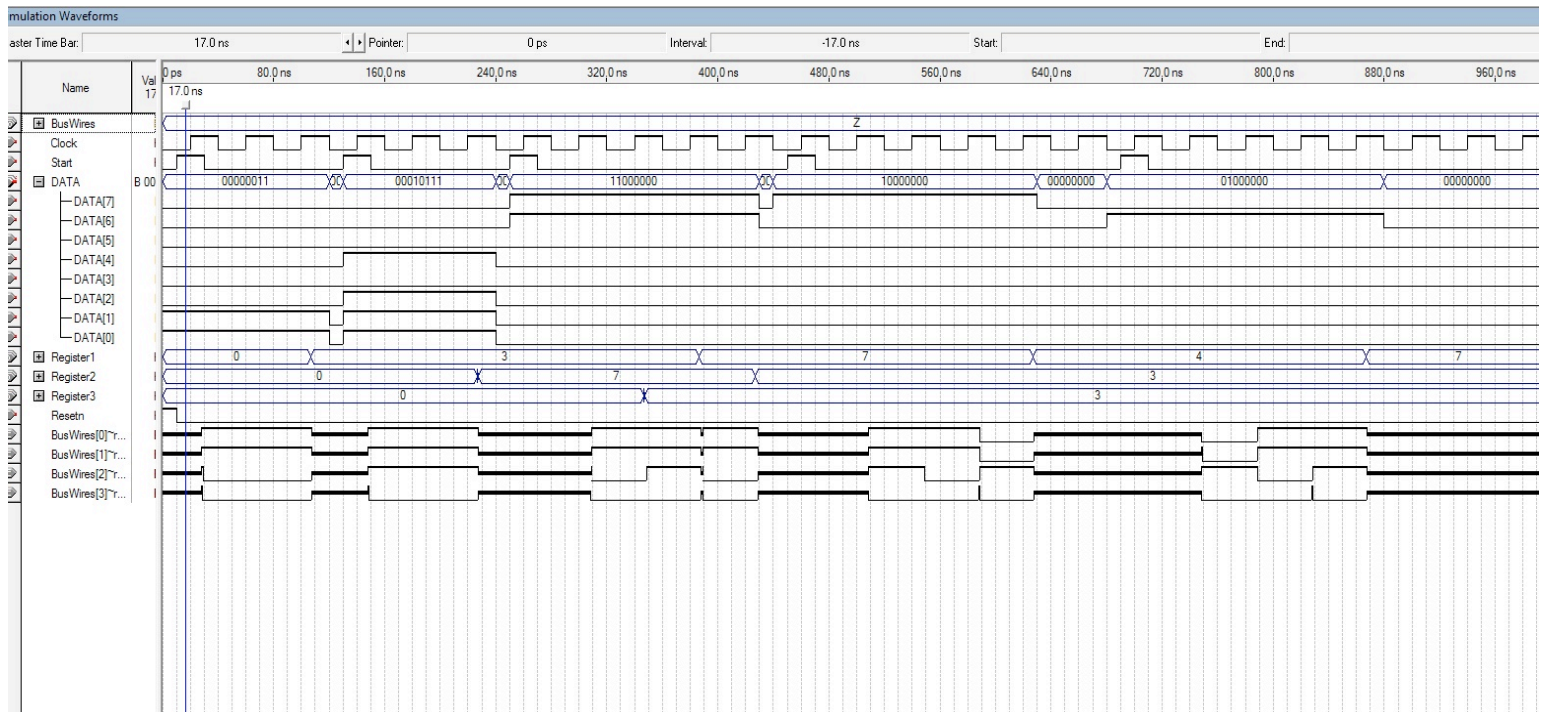
Para a demonstração principal da instrução SUB a entrada DATA é [10000000] onde o OPCODE [10] simboliza tal operação, são necessários 5 ciclos de clock que executam os 5 estados do ADD (inicial, sub1, sub2, sub3, final) que tem como função subtrair do valor do registrador 1 o valor do registrador 2 e guardar o resultado da operação no registrador 1.




[img4]

Na imagem 4[img4] a entrada DATA é [00000011] nos 3 primeiros clocks que realiza um mov do valor 3 para o registrador 1, nos próximos 3 clocks a entrada DATA é [00001011], o valor 7 é movido no registrador 2.

Para a demonstração principal da instrução xchg a entrada DATA é [11000000] onde o OPCODE [11] simboliza tal operação, são necessários 5 ciclos de clock que executam os 5 estados do add (inicial, xchg1, xchg2, xchg3, final) que tem como função trocar o valor do registrador 1 com o valor do registrador 2 usando o registrador 3 como auxiliar, podemos observar que o registrador 3 permanece com o valor do registrador 1.



[img5]

Na imagem 5 a entrada DATA é [00000011] nos 3 primeiros clocks que realiza um MOV do valor 3 para o registrador 1, depois a entrada DATA é [00000111] pelos próximos 3 clocks e o valor 7 é movido no registrador 2.

Nos próximos 5 clocks a entrada DATA é [11000000] onde o OPCODE [11] a operação de XCHG e os valores do registrador 1 e registrador 2 são trocados entre si e o valor do registrador 3 permanece igual (este valor é o antigo valor do registrador 1, que foi armazenado no registrador 3 quando foi executado o XCHG).

Ao terminar a instrução de XCHG, DATA tem como entrada [10000000], com OPCODE[10] de SUB, então os 5 clocks subsequentes são utilizados para realizar a instrução subtração dos registradores 1 e 2.

Por fim, nos clocks restantes DATA é setado como [01000000], com OPCODE [01] de ADD, então é realizado a soma dos valores do registrador 1 e 2 e guardado no registrador 1.

Para todas as instruções são necessários 21 clocks.

Conclusão

Com o primeiro projeto foi possível observar o comportamento de uma CPU que realiza as funções básicas de mover elementos de um registrador para outro, somar conteúdos entre dois registradores, subtrair conteúdo entre dois registradores e trocar de valores de dois registradores com ajuda de um registrador auxiliar. Esses conceitos utilizados, que foram obtidos a partir do conhecimento em máquinas de estado, e o conhecimento da linguagem VHDL, já com a implementação dos packages de registradores e tri-states (nbits), components, e ULA, contribuiu muito na implementação do projeto e no entendimento dos objetivos deste, tornando-se muito proveitoso para realizações futuras.

Bibliografia

-Brown, S.; Vranesic, Z. Fundamentals of Digital Logic with VHDL Design. Third Edition, Higher Education.

Anexo

REGN (Registrador):

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY regn IS
    GENERIC ( N : INTEGER := 4 ) ;
    PORT (R : IN STD_LOGIC_VECTOR(N-1 DOWNT0 0) ;
          Rin, Clock : IN STD_LOGIC ;
          Q : OUT STD_LOGIC_VECTOR(N-1 DOWNT0 0) ) ;
END regn ;

ARCHITECTURE Behavior OF regn IS
BEGIN
    PROCESS
    BEGIN
```

```

        WAIT UNTIL Clock' EVENT AND Clock = '1' ;
        IF Rin = '1' THEN
            Q <= R ;
        END IF ;
    END PROCESS ;
END Behavior;

```

TRIN (tri-state):

```

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY trin IS
    GENERIC ( N : INTEGER := 4 ) ;
    PORT ( X : IN STD_LOGIC_VECTOR(N-1 DOWNT0 0) ;
          E : IN STD_LOGIC ;
          F : OUT STD_LOGIC_VECTOR(N-1 DOWNT0 0) ) ;
    END trin ;
ARCHITECTURE Behavior OF trin IS
BEGIN
    F <= (OTHERS => 'Z') WHEN E = '0' ELSE X ;
END Behavior ;

```

Components:

```

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

PACKAGE components IS
    COMPONENT regn -- register
        GENERIC ( N : INTEGER := 4 ) ;
        PORT ( R : IN STD_LOGIC_VECTOR(N-1 DOWNT0 0);
              Rin, Clock : IN STD_LOGIC;
              Q : OUT STD_LOGIC_VECTOR(N-1 DOWNT0 0) );
    END COMPONENT ;

    COMPONENT trin --tri-state buffers
        GENERIC ( N : INTEGER := 4 ) ;
        PORT (X : IN STD_LOGIC_VECTOR(N-1 DOWNT0 0);
              E : IN STD_LOGIC ;
              F : OUT STD_LOGIC_VECTOR(N-1 DOWNT0 0));
    END COMPONENT ;

```

```

COMPONENT ula
PORT (
    A, BusWires: IN STD_LOGIC_VECTOR(3 DOWNT0 0);
    Ulatype : IN STD_LOGIC;
    Result: OUT STD_LOGIC_VECTOR(3 DOWNT0 0));
END COMPONENT;

END components;

```

ULA (Unidade Lógica Aritmética)

```

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all;

ENTITY ula IS
PORT (
    A, BusWires: IN STD_LOGIC_VECTOR(3 DOWNT0 0);
    Ulatype : IN STD_LOGIC;
    Result: OUT STD_LOGIC_VECTOR(3 DOWNT0 0)
);
END ula;

ARCHITECTURE Behavioral OF ula IS
BEGIN
    PROCESS (Ulatype, A, BusWires)
    BEGIN
        IF (Ulatype='1') THEN
            Result <= A+BusWires;
        ELSE
            Result <= A-BusWires;
        END IF;
    END PROCESS;
END Behavioral;

```

Controle:

```

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE work.components.all ;

```

ENTITY controle IS

```
PORT ( DATA    : IN  STD_LOGIC_VECTOR(7 DOWNT0 0) := (OTHERS => '0');
      Resetn    : IN  STD_LOGIC;
      Clock     : IN  STD_LOGIC;
      Start     : IN  STD_LOGIC;
      BusWires  : INOUT STD_LOGIC_VECTOR(3 DOWNT0 0) := (OTHERS => '0');
      Register1,Register2,Register3: OUT STD_LOGIC_VECTOR (3 DOWNT0 0):=(OTHERS => '0'));
END controle ;
```

ARCHITECTURE Behavior OF controle IS

```
    TYPE state_type IS (inicial, mov, add1, add2, add3, sub1, sub2, sub3, xchg1, xchg2, xchg3,
final);
    SIGNAL w      : state_type ;
    SIGNAL Rin, Rout : STD_LOGIC_VECTOR(1 TO 3) ;
    SIGNAL Ulatype : STD_LOGIC;
    SIGNAL R1, R2, R3, Aux, Aux2, Result : STD_LOGIC_VECTOR(3 DOWNT0 0) := (OTHERS =>
'0');
    SIGNAL Auxin,Extern, Aux2in, Aux2out  : STD_LOGIC := '0';
```

BEGIN

```
    reg1: regn PORT MAP  (BusWires, Rin(1), Clock, R1 );
    tri1: trin PORT MAP  (R1, Rout(1), BusWires );

    reg2: regn PORT MAP  (BusWires, Rin(2), Clock, R2 );
    tri2: trin PORT MAP  (R2, Rout(2), BusWires );

    reg3: regn PORT MAP  (BusWires, Rin(3), Clock, R3 );
    tri3: trin PORT MAP  (R3, Rout(3), BusWires );

    regT1: regn PORT MAP  (BusWires, AUXin, Clock, Aux);
    triT1: trin PORT MAP  (DATA(3 DOWNT0 0), Extern, BusWires );

    regT2: regn PORT MAP  (Result, Aux2in, Clock, Aux2);
    triT2: trin PORT MAP  (Aux2, Aux2out, BusWires );

    opula: ula PORT MAP  (Aux, BusWires, Ulatype, Result);

    PROCESS(Resetn, Clock)
    BEGIN
        IF Resetn='1'THEN
            w<=Inicial;
```

```

ELSIF(clock'EVENT AND Clock='1')THEN

CASE w IS
  WHEN inicial =>

    IF Start = '1' THEN

      CASE DATA(7 DOWNT0 6)IS
        WHEN "00" =>

          Extern <= '1';

          w <= mov;

        WHEN "01" =>

          w <= add1;

        WHEN "10" =>

          w <= sub1;

        WHEN "11" =>

          w <= xchg1;
      END CASE;
    END IF;

  WHEN mov =>

    IF DATA(5 DOWNT0 4)="00" THEN
      Rin(1)<='1';

    ELSIF DATA(5 DOWNT0 4)="01" THEN
      Rin(2)<='1';

    ELSIF DATA(5 DOWNT0 4)="10" THEN
      Rin(3)<='1';
    END IF;

    w <= final;

  WHEN add1 =>

```



```

        Rout(1) <= '1';
        Auxin <= '1';
        Ulatype <= '1';

w <= add2;

WHEN add2 =>

    Auxin <= '0';
    Aux2in <= '1';
    Rout(1) <= '0';
    Rout(2) <= '1';

w <= add3;

WHEN add3 =>

    Rout(2) <= '0';
    Rin(1) <= '1';
    Aux2out <= '1';

w <= final;

WHEN sub1 =>

    Rout(1) <= '1';
    Auxin <= '1';
    Ulatype <= '0';

w <= sub2;

WHEN sub2 =>

    Auxin <= '0';
    Aux2in <= '1';
    Rout(1) <= '0';
    Rout(2) <= '1';

w <= sub3;

WHEN sub3 =>

```

```

        Rout(2) <= '0';
        Rin(1) <= '1';
        Aux2out <= '1';

        w <= final;

    WHEN xchg1 =>
        Rout(1) <= '1';
        Rin(3) <= '1';
        w <= xchg2;

    WHEN xchg2 =>
        Rout(1) <= '0';
        Rin(3) <= '0';
        Rin(1) <= '1';
        Rout(2) <= '1';
        w <= xchg3;

    WHEN xchg3 =>
        Rout(2) <= '0';
        Rin(1) <= '0';
        Rin(2) <= '1';
        Rout(3) <= '1';
        w <= final;

    WHEN final =>
        Rin(1) <= '0';
        Rin(2) <= '0';
        Rin(3) <= '0';

        Auxin <= '0';
        Aux2in <= '0';
        Extern <= '0';
        Aux2out <= '0';

        Rout(1) <= '0';
        Rout(2) <= '0';
        Rout(3) <= '0';

        w <= inicial;

    END CASE;
END IF;
END PROCESS;

```

```
Register1 <= R1;  
Register2 <= R2;  
Register3 <= R3;
```

```
END Behavior;
```