Seminário

Ponto de Checagem 3

Aluno: Bruno Ogata Franchi

RA: 101893

UC: Laboratório de Arquitetura e Organização de Computadores

Projeto proposto pelo Prof. Dr. Tiago de Oliveira

Projeto

Projeto

Desenvolvimento de um processador em lógica programável, para funcionar em uma FPGA

- Conjunto de Instruções
- Unidade de Processamento
- **Unidade de Controle**
- Testes na FPGA

Processador NorMIPS

- Baseada na Arquitetura MIPS
- Conjunto de Instruções RISC com 16 instruções
- Formato de Instruções

NorMIPS						
Formato	op	rs	rt	rd	shamt	funct
Tipo R	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
Formato	op	rs	rt		endereço	
Tipo I	6 bits	5 bits	5 bits		16 bits	
Formato	op			endereço		
Tipo J	6 bits			26 bits		

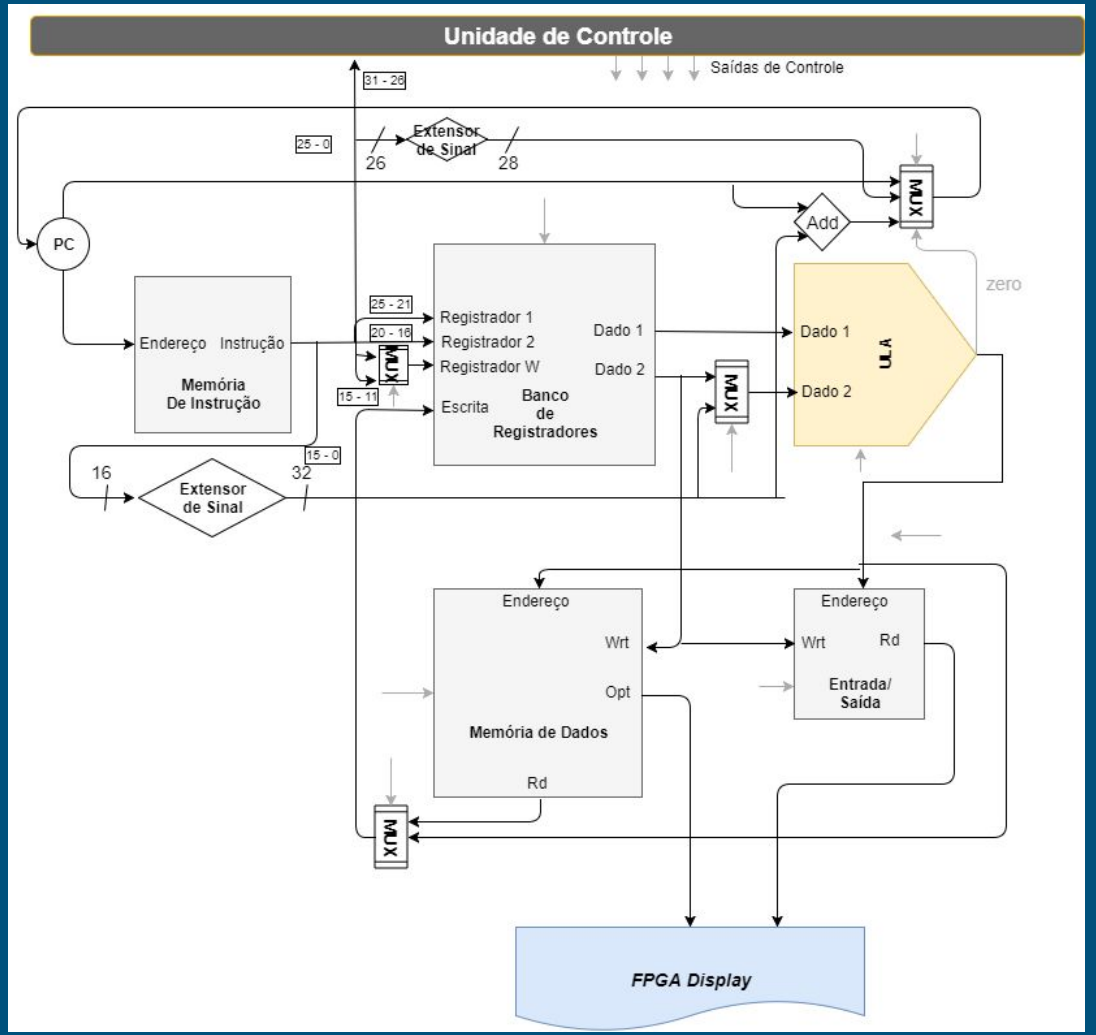
Conjunto de Instruções

NorMIPS		
R-type	I-type	J-type
add	lw	j
sub	sw	
and	slti	Entrada e Saída/Reset
or	beq	in*
slt	bne	out*
	addi	reset*
	subi	

* Instruções ainda não implementadas

Processador NorMIPS

Arquitetura Base:



Unidade de Controle

Fundamentação Teórica

- Unidade de Controle
  - O que é?
  - Importância
  - Tipos de Implementação
    - Unidade de Controle por Microprogramação
    - Unidade de Controle *Hardwire*
      - Máquina de Estados

Fundamentação Teórica

O que é?

Unidade de Controle é o componente presente nos processadores, responsáveis por conduzir os sinais que controlam as operações nos outros módulos da CPU.

Importância:

- **Controle** - Controla a ordem que as operações nos módulos são executadas.
- **Generalização** - Sem ela, o processador executaria apenas uma única função!

Fundamentação Teórica

Tipos de Implementação:

- **Unidade de Controle por Microprogramação:** Possui um circuito decodificador que decodifica as instruções e as usam em um microprograma interno da UC.

- **Unidade de Controle *Hardwire*:** A Unidade de Controle é dada como uma Máquina de Estados Finitos. Os estados são determinados pela instrução em execução.

UC de NorMIPS

Instruções e opcodes:

add	100000	RD = RS + RT	beq	010110	condicional ==
addi	010000	RD = RS + X	bnq	010111	condicional !=
sub	100001	RD = RS - RT	lw	011111	RT = Mem[adr]
subi	010001	RD = RS - X	sw	011110	Mem[adr] = RT
and	100010	RD = RS & RT	j	111111	Jump
or	100011	RD = RS | RT	reset	000000	Reset CPU
slt	100100	RD = (RS < RT)	in	000111	Input
slti	010100	RD = (RS < X)	out	111000	Output

Unidade de Controle de NorMIPS

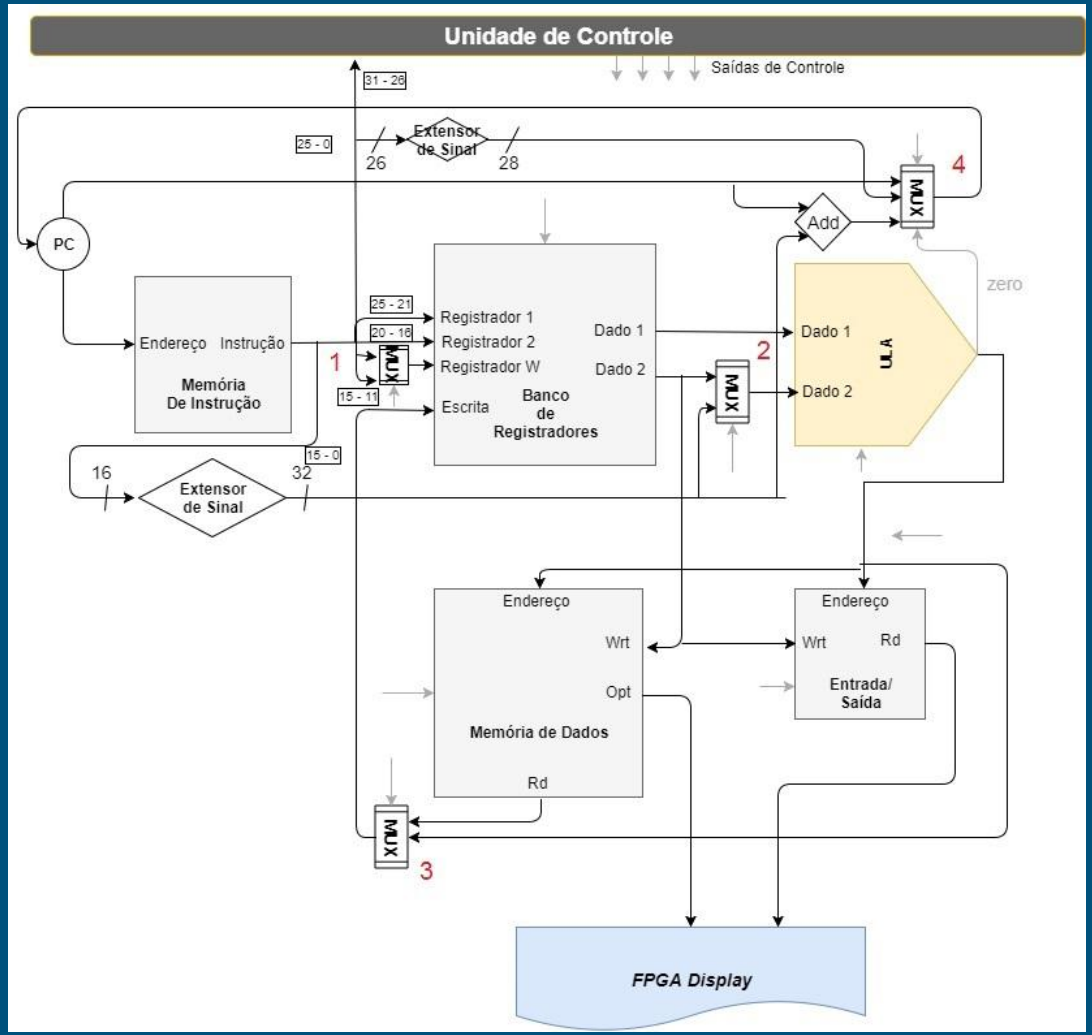
A partir do *OpCode* (6 bits) é gerado os sinais de controle:

controle_BANCOREG	1 bit	controle_MUX2	1 bit
controle_ALU	3 bits	controle_MUX3	1 bit
controle_MEMDADOS	1 bit	controle_MUX4	2 bits
controle_MUX1	1 bit		

add

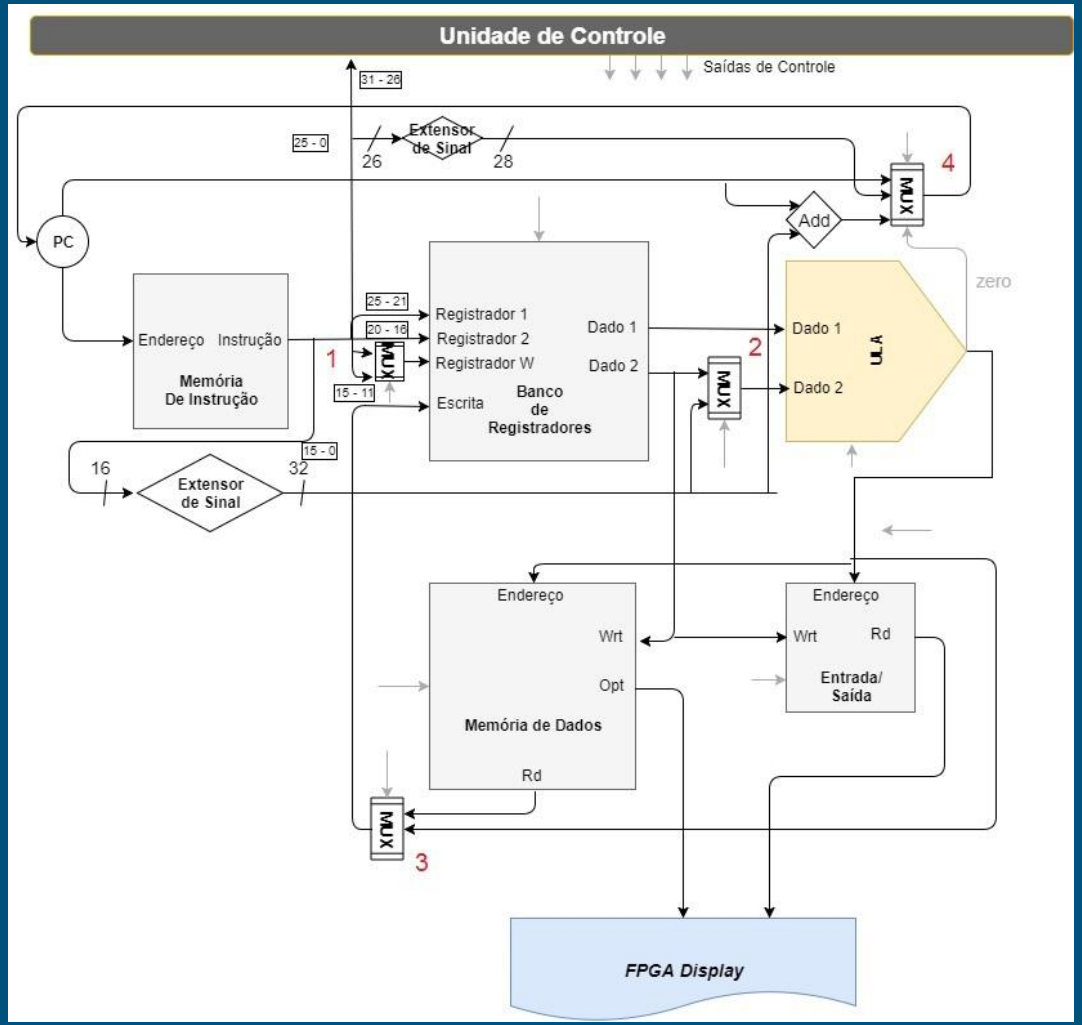
```
6'b100000: begin // add
    controle_BANCOREG = 1'b1;
    controle_ALU = 3'b000;
    controle_MEMDADOS = 1'bx;
    controle_MUX1 = 1'b1;
    controle_MUX2 = 1'b0;
    controle_MUX3 = 1'b0;
    controle_MUX4 = 2'b00;
end
```

Para instruções do tipo-R o processo é semelhante.



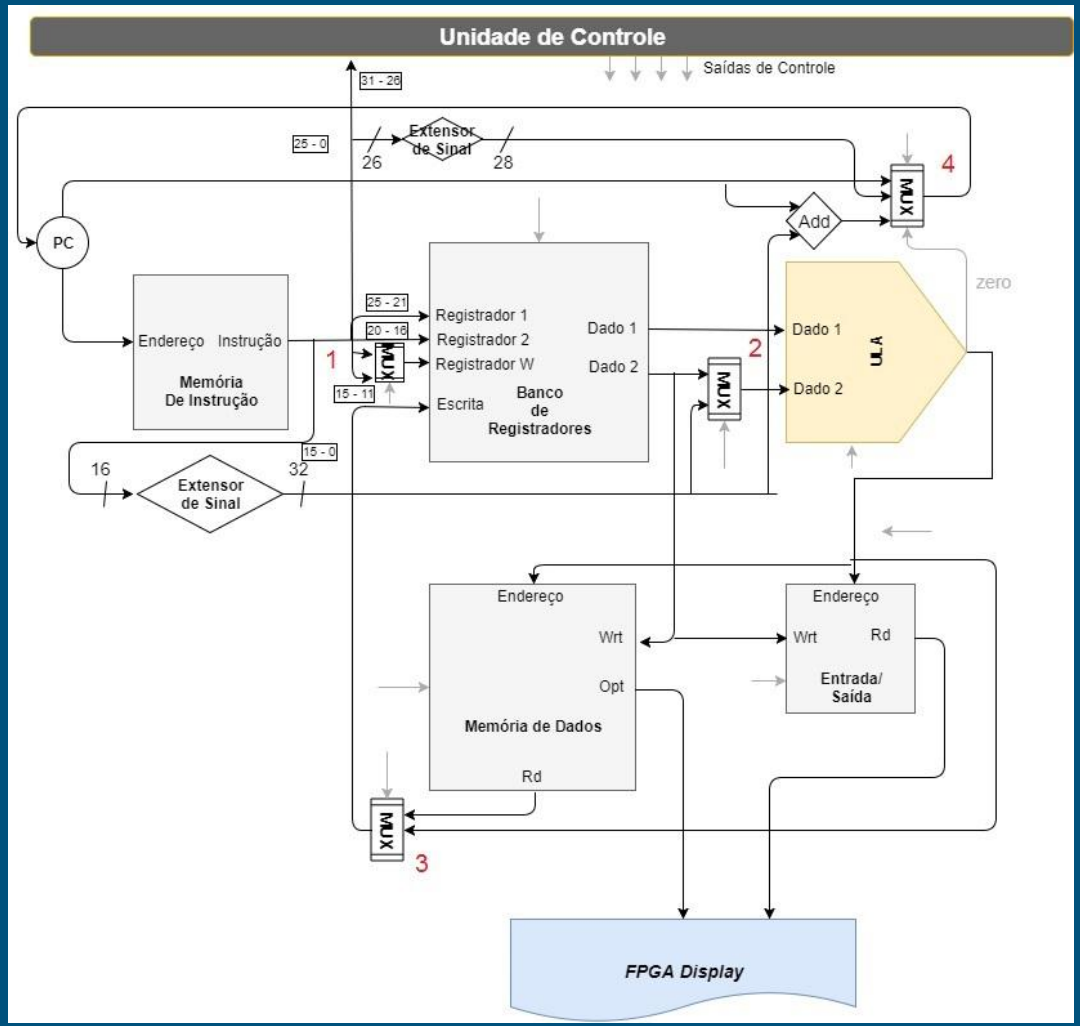
lw

```
6'b011111: begin // lw
    controle_BANCOREG = 1'b1;
    controle_ULA = 3'b000;
    controle_MEMDADOS = 1'b0;
    controle_MUX1 = 1'b0;
    controle_MUX2 = 1'b1;
    controle_MUX3 = 1'b1;
    controle_MUX4 = 2'b00;
end
```



SW

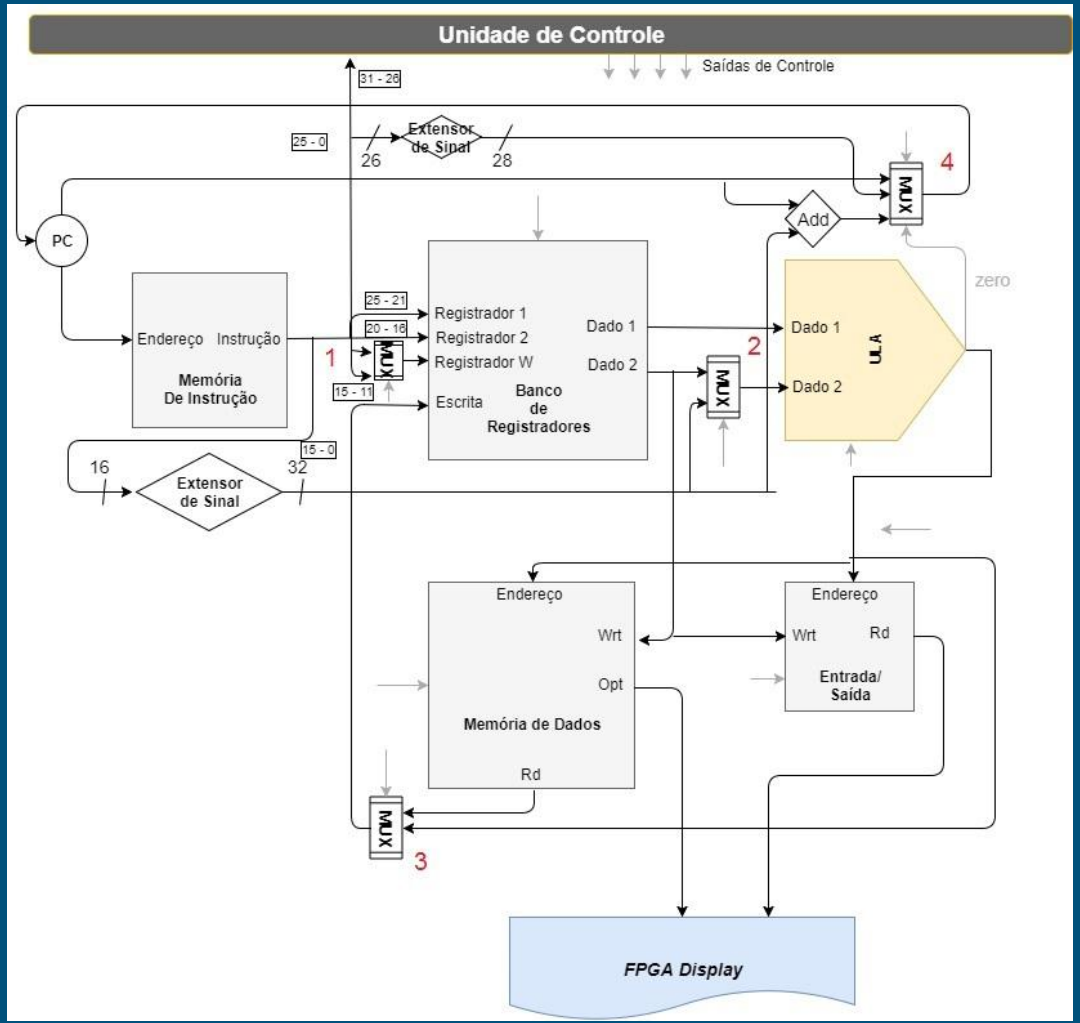
```
6'b011110: begin // sw
    controle_BANCOREG = 1'b0;
    controle_ALU = 3'b000;
    controle_MEMDADOS = 1'b1;
    controle_MUX1 = 1'bx;
    controle_MUX2 = 1'b1;
    controle_MUX3 = 1'bx;
    controle_MUX4 = 2'b00;
end
```



beq

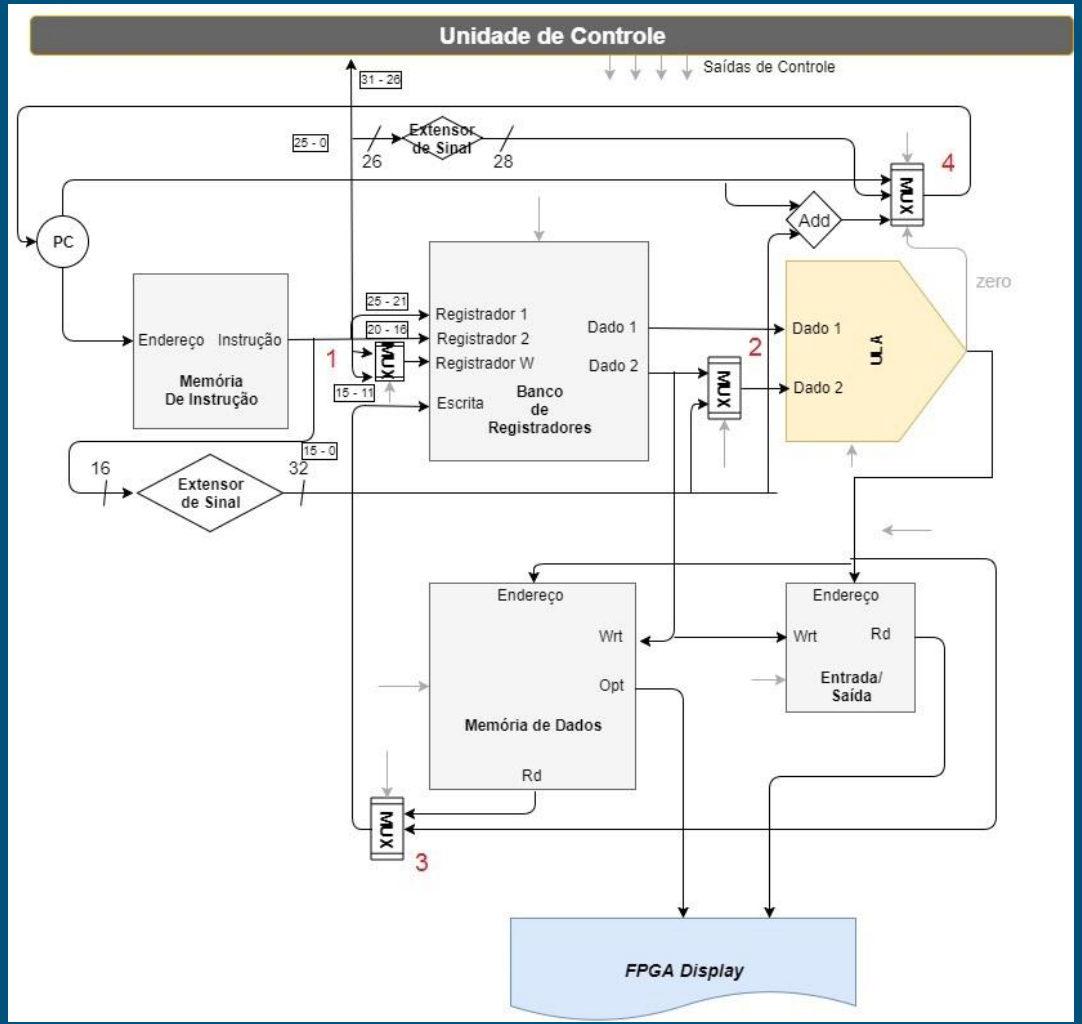
```
6'b010110: begin // beq
    controle_BANCOREG = 1'b0;
    controle_ALU = 3'b001;
    controle_MEMDADOS = 1'bx;
    controle_MUX1 = 1'bx;
    controle_MUX2 = 1'b0;
    controle_MUX3 = 1'bx;
    controle_MUX4 = 2'b01;
end
```

Para instrução do tipo BNQ o processo é semelhante.



addi

```
6'b010000: begin // addi
    controle_BANCOREG = 1'b1;
    controle_ALU = 3'b000;
    controle_MEMDADOS = 1'bx;
    controle_MUX1 = 1'b0;
    controle_MUX2 = 1'b1;
    controle_MUX3 = 1'b0;
    controle_MUX4 = 2'b00;
end
```



jump

```
6'b111111: begin // jump
    controle_BANCOREG = 1'bx;
    controle_ALU = 3'bxxx;
    controle_MEMDADOS = 1'bx;
    controle_MUX1 = 1'bx;
    controle_MUX2 = 1'bx;
    controle_MUX3 = 1'bx;
    controle_MUX4 = 2'b11;
end
```