



Universidade Federal de Pernambuco

Centro de Informática

---

## RELATÓRIO DE PROJETO

---

Arthur Henrique Aníbal da Costa, ahac

José Carlos da Silva Filho, jcsf2

Lucas Grisi Oliveira de Queiroz, lgoq

Pedro Henrique Sarmento de Paula, phsp

Recife, 11 de Outubro de 2019

**Professora:** Edna Natividade da Silva Barros

# Sumário

<b>Descrição dos Módulos</b>	<b>2</b>
Módulo: UP.sv	2
Módulo: SING_EXT.sv	2
Módulo: MAQUINA_DE_ESTADO.sv	3
<b>Descrição das operações</b>	<b>4</b>
Tipo R	4
Tipo I	5
Tipo I (Shift)	5
Tipo S	6
Tipo SB	6
Tipo U	6
Tipo UJ	6
<b>Estados</b>	<b>7</b>
reset	7
SomaPC	7
Espera	7
Espera2	7
Espera_store	7
Load_reg	7
check_tipo	7
Overflow	7
Jal_soma	8
BreakOp	8
load_254_pc	8
recebe_pc_wrt_pc	8
wrt_reg_lui	8
write_reg_alu	8
wrt_0_reg	8
wrt_1_reg	8
read_mem_store	8
read_mem	9
espera_reg_mem	9
salva_reg	9
write_mem	9
<b>Arquitetura da CPU</b>	<b>10</b>
<b>Diagrama da Máquina de Estados</b>	<b>11</b>

# Descrição dos Módulos

## Módulo: *UP.sv*

### Entradas:

1. CLK
2. RST

**Saídas:** *Sem saídas.*

### Objetivo:

- Processa as instruções , carrega e salva registradores, envia sinais de controle para os outros módulos.

### Algoritmo:

- Recebe os dados dos módulos e envia para os outros.

## Módulo: *SING\_EXT.sv*

### Entradas:

- instructions[31:0]

### Saídas:

- instructions[63:0]

### Objetivo:

- Tratar a instrução de 32 bit para 64 bit preenchendo os bits restantes com o bit do sinal.

### Algoritmo:

- Vai tratar o *immediate* de acordo com o tipo da instrução e em seguida vai estender o sinal. Se for negativo preenche com “1”, caso contrário, preenche com “0”.

## **Módulo:** MAQUINA\_DE\_ESTADO.sv

### **Entradas:**

- [31:0]INSTRUCAO
- [6:0] op\_code
- ZERO\_ALU
- IGUAL\_ALU
- MAIOR\_ALU
- OVERFLOW

### **Saída:**

- SELETOR\_ALU
- [1:0] SELETOR\_MUX\_MEM\_ADDRESS
- WR\_EPC
- WR\_BANCO\_REG
- [3:0] SELECT\_MUX\_DATA
- wrDataMemReg
- WR\_ALU\_OUT
- wrDataMem
- [1:0] SELECT\_MUX\_MEM
- reset\_wire
- [2:0] operacao
- WRITE\_PC
- LOAD\_IR
- WR\_MEM\_INSTR

- [1:0] SELETOR\_MUX\_A
- [2:0] SELETOR\_MUX\_B
- write\_reg\_A
- write\_reg\_B
- SELETOR\_SHIFT

#### Objetivos:

- O objetivo da máquina de estados é pegar grandes instruções e dividi-las em passos para a realização de pequenas instruções e por fim, baseado no resultado concluir a instrução inicialmente requisitada.

#### Algoritmo:

- A máquina primeiramente checa o tipo da operação e em seguida, baseado no tipo da operação, encaminha para o estado adequado.

## Descrição das operações

### Tipo R

Instrução **add** rd,rs1,rs2

- Após a identificação da operação soma, a operação é realizada. Os valores dos registradores fonte *rs1* e *rs2* são carregados do banco de registradores, o valor é escrito no registrador destino e uma nova instrução é buscada na memória.

Instrução **sub** rd, rs1, rs2

- Após a identificação da operação subtração, a operação é realizada. Os valores dos registradores fonte *rs1* e *rs2* são carregados do banco de registradores, o valor é escrito no registrador destino e uma nova instrução é buscada na memória.

Instrução **and** rd, rs1, rs2

- Após a identificação do tipo da operação e de acordo com dados da alu, é realizada a operação de escrita "0" ou "1" no registrador. E uma nova instrução é buscada na memória.

Instrução **slt** rd, rs1, rs2

- Após a identificação da operação "*slt*", a operação é realizada. Os valores dos registradores fonte *rs1* e *rs2* são carregados do banco de registradores, o valor booleano é escrito no registrador destino e uma nova instrução é buscada na memória.

## Tipo I

Instrução **jalr** *rd, rs1, imm*

- Salva o valor de PC no registrador destino e soma o imediato no novo valor do PC.

Instruções **ld, lb, lw, lh, lbu, lwu, lhu** *rd, imm(rs1)*

- Carrega o valor na posição de memória *rs1*, os diferentes tipos de LOAD são para diferentes tamanhos de dados, indo de 8 bits a 64 bits.

Instrução **NOP**

- Não faz nada. Apenas serve pra gastar um ciclo de clock.

Instrução **addi** *rd, rs1, imm*

- Após a identificação da operação soma, a operação é realizada. Os valores dos registradores fonte *rs1* é carregado do banco de registradores e do *immediate* é fornecido pela entrada da instrução, o valor é escrito no registrador destino e uma nova instrução é buscada na memória.

## Tipo I (Shift)

Instrução **srlr** *rd, rs1, shamt*

- Shift lógico para direita, o tamanho do shift é definido pelo *shamt*.

Instrução **srai** *rd, rs1, shamt*

- Shift aritmético para direita, o tamanho do shift é definido pelo *shamt*.

Instrução **sllr** *rd, rs1, shamt*

- Shift aritmético para esquerda, o tamanho do shift é definido pelo *shamt*.

## Tipo S

Instruções **sd, sw, sh, sb** *rs2, imm(rs1)*

- Funções de **STORE**, responsáveis por salvar na memória os valores. A diferença entre as funções são o tamanho do valor carregado, indo de 8 a 64bits.

## Tipo SB

Instrução **beq** *rs1, rs2, imm*

- Operação de pulo condicional, no caso do beq, e os valores dos registradores rs1 e rs2 são carregados do banco de registradores. Caso o valor do rs1 seja igual ao valor do rs2, o PC pula *immediate* instruções. e uma nova instrução é buscada na memória.

Instrução **bne** *rs1, rs2, imm*

- Operação de pulo condicional, no caso do bne, e os valores dos registradores rs1 e rs2 são carregados do banco de registradores. Caso o valor do rs1 seja diferente do valor do rs2, o PC pula *immediate* instruções e uma nova instrução é buscada na memória.

Instrução **bge** *rs1, rs2, imm*

- Operação de pulo condicional, no caso do bge, e os valores dos registradores rs1 e rs2 são carregados do banco de registradores. Caso o valor do rs1 seja maior ou igual ao valor do rs2, o PC pula *immediate* instruções.

## Tipo U

Instrução **lui** *rd, imm*

## Tipo UJ

Instrução **jal** *rd, imm*

- O valor do *immediate* é fornecido pela instrução. E a operação de pulo incondicional é realizada dando um “pulo” com a “distância” do *imm*.

# Estados

## reset

O estado reset reinicia todos os valores das entradas e saídas da máquina de estado. Pulando para o estado de espera.

## SomaPC

É o estado em que o PC recebe a atualização  $PC = PC + 4$ . Os valores vêm do Mux A na posição 1, que é a saída do valor do PC e do Mux B na posição 1, que é o valor fixo 4.

## Espera

É um estado que não faz nada, é usado para esperar os registradores serem carregados. A diferença para os demais estados de espera é que esse pula para o estado SomaPC. Aqui é checado se a flag **OVERFLOW** da **ULA** foi ativada, caso seja, será tratada a exceção.

## Espera2

É um estado que não faz nada, é usado para esperar os registradores serem carregados. A diferença para os demais estados de espera, esse pula para o estado read\_mem.

## Espera\_store

É um estado que não faz nada, usado apenas na função *store*, funciona esperando um ciclo de clock para atualizar, escrevendo, o valor do registrador destino.

## Load\_reg

É o estado geral para carregar os registradores A e B.

## check\_tipo

Sua função é detectar o tipo de instrução que será realizada no próximo estado.

## Overflow

Estado de tratamento de exceção para quando o valor do registrador destino extrapola 64 bits.



## Jal\_soma

Soma *immediate* com shift à esquerda com PC e escreve o resultado em PC.

## BreakOp

Estado que mata a execução da instrução.

## load\_254\_pc

Carrega valor do endereço 254 da memória em PC, tem finalidade o tratamento da exceção do op code inexistente.

## recebe\_pc\_wrt\_pc

Escreve PC no registrador destino, e escreve o novo valor de PC.

## wrt\_reg\_lui

Após a operação de *lui* é necessário esperar o ciclo de clock para escrita do resultado no registrador destino. Então após a execução do *lui*, um ciclo de clock é passado e a escrita é realizada.

## write\_reg\_alu

Após a execução de algumas instruções seus resultados precisam ser escritos no banco de registradores.

## wrt\_0\_reg

Escreve 0 no registrador como resposta *booleana* da instrução *sli*.

## wrt\_1\_reg

Escreve 1 no registrador como resposta *booleana* da instrução *sli*.

## read\_mem\_store

Escreve a saída da memória em MEM\_DATA\_REG.

## read\_mem

Espera leitura da memória para as operações de load.

## espera\_reg\_mem

Escreve a saída da memória em MEM\_DATA\_REG.

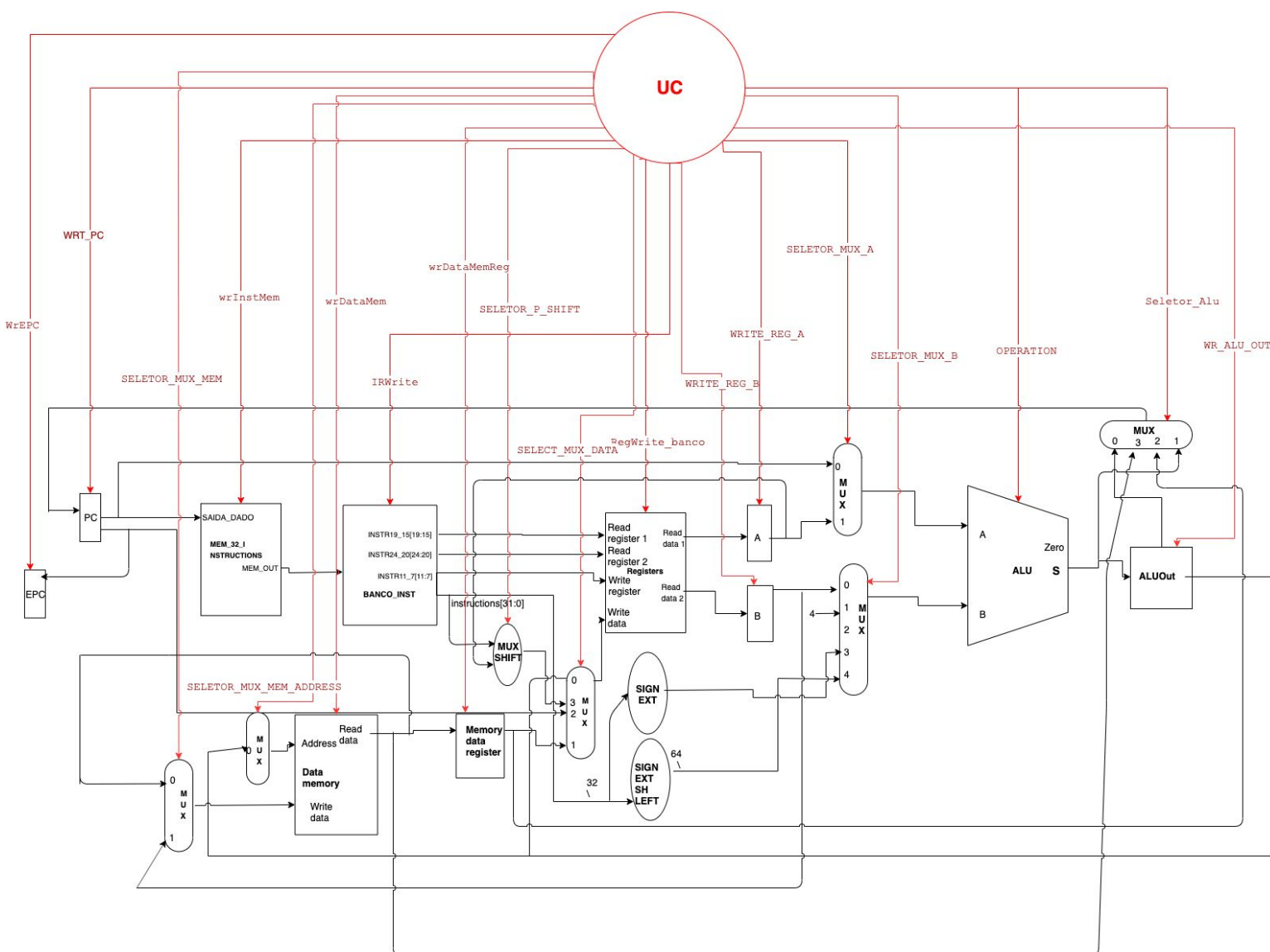
## salva\_reg

Salva valor de MEM\_DATA\_REG no registrador de destino.

## write\_mem

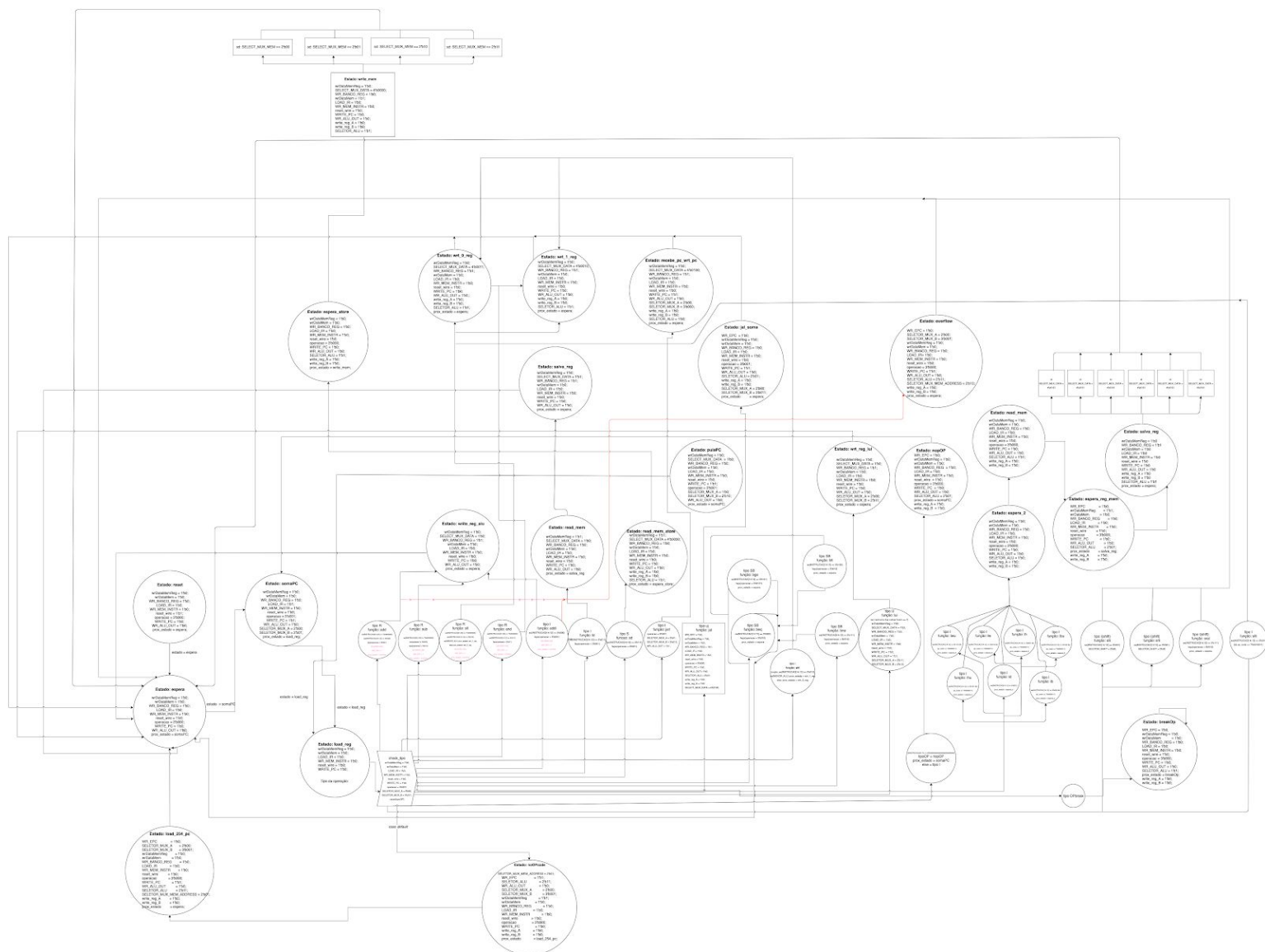
Escreve o valor da saída de MUX\_WRT\_DATA no endereço de memória fornecido pela instrução de store.

# Arquitetura da CPU



Ou, [LINK](#)

## Diagrama da Máquina de Estados



Ou, [LINK](#)