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

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 passa a passo para a realização de pequenas instruções e por fim, baseado nos 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 **srli** *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 **slli** 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.

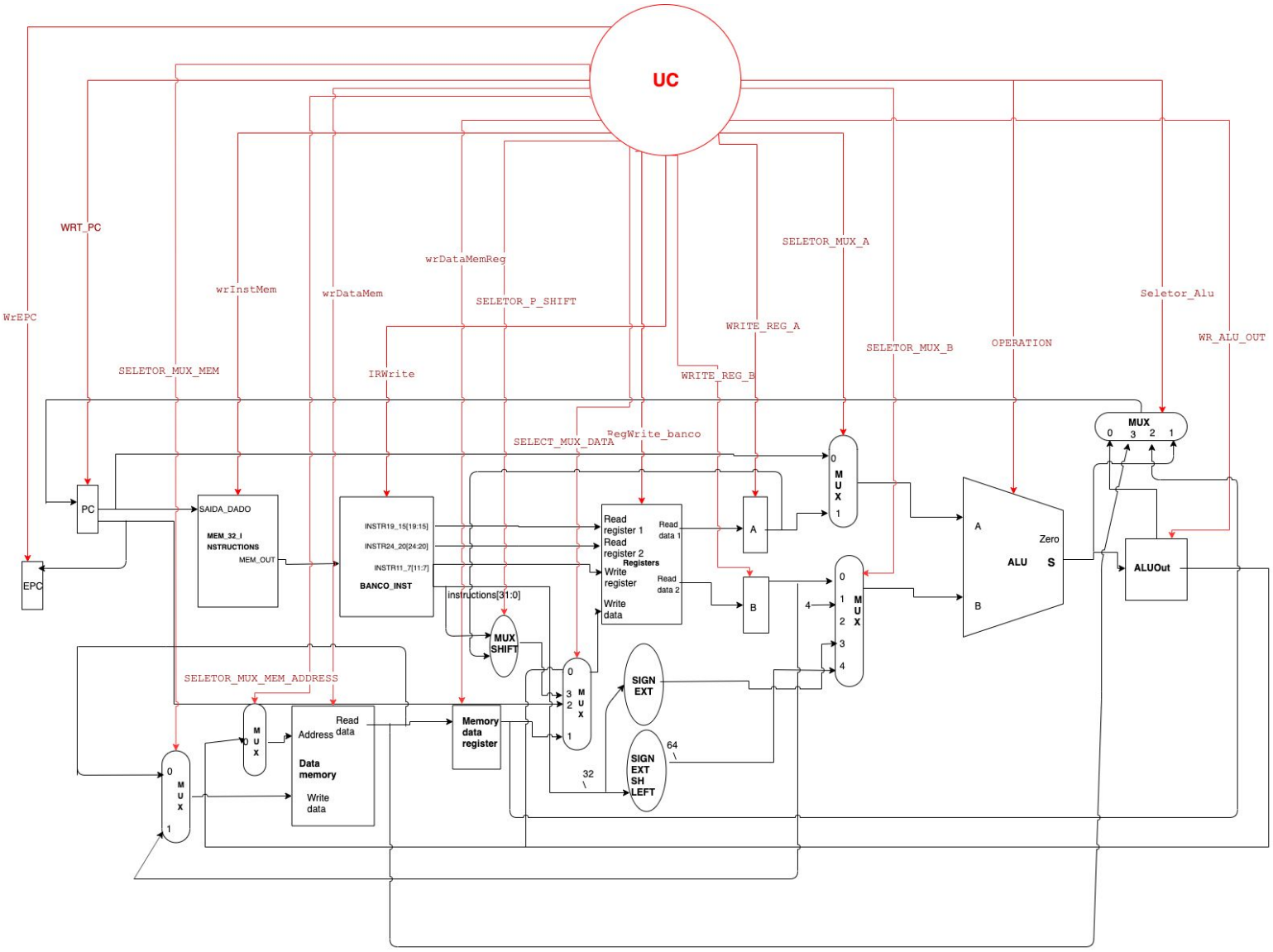
salva_reg

Salva valor de MEM_DATA_REG no registrador de destino.

write_mem

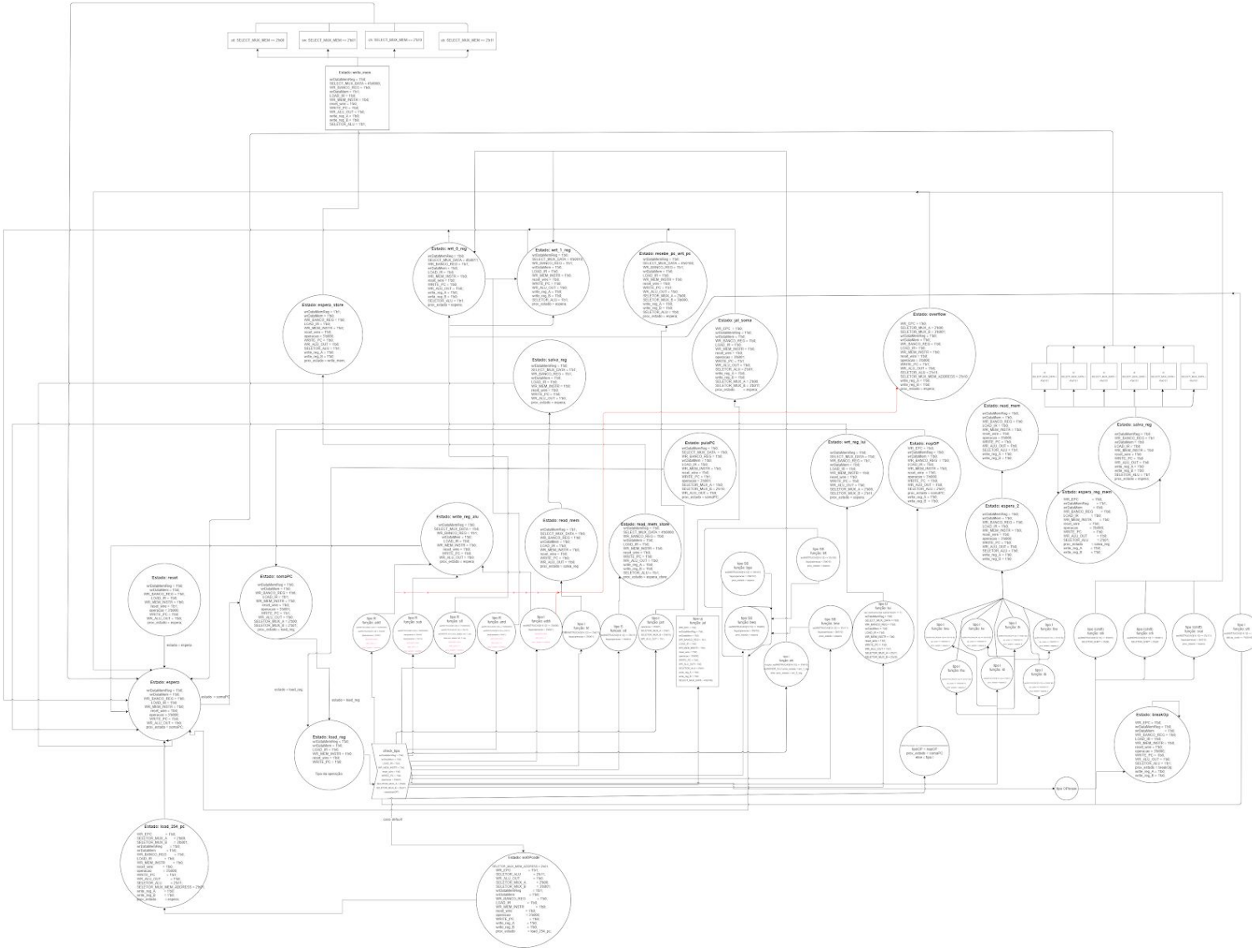
Escreve o valor da saída de MUX_WRT_DATA no endereço de memória fornecido pela instrução de store.

Arquitetura da CPU



***Ou*, [LINK]**

Diagrama da Máquina de Estados



Ou, [LINK]