UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

RELATÓRIO DE PROJETO

Alexandro Jose da Silva, ajs6

José Carlos da Silva Cruz, jcsc

Nathalia Paiva Lima, npl

Zenio Angelo Oliveira Neves, zaon

RECIFE, 30 de Abril de 2019

Professora: Edna Natividade da Silva Barros

Sumário

1. Descrição de Módulos

Nessa seção cada um dos módulos implementados será apresentado, abordando o objetivo almejado a partir da sua construção e a sua funcionalidade dentro do projeto. Além disso, especificamos os sinais de entrada e saída presentes em cada um deles e como funciona o seu algoritmo interno.

1.1. Controle

Módulo: UC

- Entradas:
  - clock (1bit) - representa o clock do sistema.
  - reset (1bit) - sinal que, quando ativado, zera o conteúdo da unidade de controle.
  - Register_Intruction_Instr31_0 (32bits) - envia a instrução completa.
  - z (1bit) - flag que indica se o resultado da operação foi 0.
  - igual (1bit) - flag que indica se A=B.
  - maior (1bit) - flag que indica se A>B.
  - menor (1bit) - flag que indica se A<B.
  - overFlow (1bit) - flag que indica se houve overflow.
  - multiplicador(128bits) - multiplicando.

- Saídas:
  - PC_Write (1bit) - sinal que ativa inscrita em PC.
  - Seletor_Ula (3bits) - seleciona a operação aritmética na ula.
  - mux_A_seletor (3bits) - seleciona o valor que irar para a saida do mux_A.
  - mux_B_seletor (3bits) - seleciona o valor que irar para a saida do mux_B.
  - Data_Memory_wr (1bit) - sinal que ativa escrita em Data_Memory.
  - bancoRegisters_wr (1bit) - sinal que ativa escrita no banco de registradores.
  - reset_A - bit de sinal que zera (reseta) o registrador A
  - Mux_Banco_Reg_Seletor (3bit) - seleciona o valor que irar para a entrada do banco de registradores.
  - Shift_Control (2bits) - controle de tipo de shift que o deslocador de n bits irar realizar.
  - Reg_A_Write (1bit) - controle de escrita no registrador A.
  - Reg_B_Write (1bit) - controle de escrita no registrador A.
  - Reg_Memory_Data_wr (1bit) - controle de escrita no registrador Reg_Memory_Data.
  - Load_ir (1bit) - controle de leitura/escrita no registrador de instrução.
  - EPC_wr (1bit) - controle de escrita no registrador EPC.

    - Reg_Causa_wr (1bit) - controle de escrita no registrador Reg_Causa.
    - Reg_Causa_Dados_In (64bits) - entrada do Reg_Causa.
    - Mux64_PC_Extend_Seletor (3bits) - seleção do valor de entrada de PC.
    - flag_overFlow (1bit) - flag para auxílio na exceção de overflow.
    - Mux_Seletor_C_In_128 - Seletor do Mux de entrada do registrador C.
    - C_wr(1bit) - controle de escrita no registrador C.
    - Soma_wr(1bit) - controle de escrita no registrador Soma.
    - Mux_Seletor_Mux_Hi_Low(3bits) - Seleção para o envio dos bit mais ou menos significativos da instrução multiplicação.
    - Seletor_Ula_128(3bits) - Seletor da Ula de multiplicação.

- Objetivo: controlar os demais módulos do processador a para que eles atuem de maneira coerente na execução das instruções.
- Algoritmo: é implementada uma máquina de estados finita, onde em cada estado há a atribuição de valores para os diversos sinais de saída, que atuaram nos demais módulos realizando, auxiliando e controlando, a execução das instruções solicitadas.

1.2. Battousai_Store

- Entradas:
    - Reg_B_Out (64bits) - valor do registrador B.
    - Register_Intruction_Instr31_0 (32bits) - valor da instrução lida.
- Saídas:
    - DadoIn_64 (64bits) - modificação de 1,2 ou 4 bytes para entrada da memoria 64.
- Objetivo: tratamento de instruções de store.
- Algoritmo: pega a e concatena valores.

1.3. Battousai_Load

- Entradas:
    - Dataout (64bits) - valor do registrador de memoria de dados Reg_Memory_Data.
    - Register_Intruction_Instr31_0 (32bits) - valor da instrução lida.

- Saídas:
    - Saida_Memory_Data (64bits) - modificação de 1,2 ou 4 bytes para entrada no banco de registradores.
- Objetivo: tratamento de instruções de load.
- Algoritmo: pega a e concatena valores.

1.4. Sinal_Extend

- Entradas:

    - Sinal_In (32bits) - entrada que terar bits extendidos.
- Saídas:
    - Sinal_Out (64bits) - valor de 32 bits extendido para 64 bits.
- Objetivo: extender a entrada de 32 bits para 64 bits.
- Algoritmo: desembaralha valores de imediatos e concatenar valores de acordo com a instrução.

1.5. Sinal_Extend_8_32

- Entradas:
    - Entrada_8_bits (32bits) - entrada que terar bits extendidos.
- Saídas:
    - Saida_32 (64bits) - valor de 32 bits extendido para 64 bits.
- Objetivo: estender a entrada de 32 bits para 64 bits.
- Algoritmo: desembaralha valores de imediatos e concatenar valores de acordo com a instrução.

1.6. Sign_Extend_64_128

- Entradas:
    - Entrada_64 (64bits)
- Saídas:
    - Saída_128 (128bits)
- Objetivo: Estender o tamanho máximo do registrador
- Algorítmo: Estende sinal

1.7. Sign_Reduct_128_64

- Entradas:
    - Entrada_128 (128bits)
- Saídas:
    - Saida_Hi_128 (64bits)
    - Saida_Low_128 (64bits)
- Objetivo: Dividir a entrada de 128 bits em duas saídas de 64, contendo os 64 bits mais significativos e os 64 menos significativos.
- Algorítmo: Divide sinal

1.8. Ula_para_mult_128_bits

- Entradas:
    - A (128bits)
    - B (128bits)
- Saídas:
    - S (128bits)
- Objetivo: Realizar contas aritméticas entre os registradores de 128 bits

2. Descrição das operações

2.1. Tipo R

2.1.1. add:

- Instrução: add rd, rs1, rs2;
- Objetivo: Os valores dos registradores rs e rt são carregados e a operação soma da ALU é selecionada, somando os conteúdos dos registradores.
- Algoritmo: Após o estado SELECAO identificar que a instrução é do tipo R, o opcode (os 7 bits menos significativos de Register_Intruction_Instr31_0) for 0110011 (7d'51). É enviado um sinal de controle para a ULA, onde, através da identificação do Funct3 para a instrução add, será realizada a soma dos valores carregados na entrada. Busca os valores rs1 e rs2 no banco de registradores, onde tais valores seẽao passados para a ULA. O resultado que sair da ALU irá para o banco de registradores, onde vai salvar no registrador rd o resultado. O rd é escolhido como registrador de destino, no banco de registradores, recebendo o valor originado na operação na ULA.

2.1.2. sub:

- Instrução: sub rd, rs1, rs2
- Objetivo: Os registradores rs1 e rs2 são carregados e a operação subtração da ALU é selecionada, subtraindo os conteúdos dos registradores.
- Algoritmo: Após o estado SELECAO identificar que a instrução é do tipo R, é enviado um sinal de controle para a ULA, onde, através da identificação do Funct3 para a instrução sub, Register_Intruction_Instr31_0[14:12] for igual a 3'd0, será realizada a subtração dos valores carregados na entrada. Busca os valores rs1 e rs2 no banco de registradores, onde tais valores seẽao passados para a ULA e o valor originado na operação vai para rd, no banco de registradores.

2.1.3. and:

- Instrução: and rd, rs1, rs2
- Objetivo: Os registradores rs1 e rs2 são carregados e a operação and da ALU é selecionada, fazendo and bit a bit com o conteúdo dos registradores.
- Algoritmo: Após o estado SELECAO identificar que a instrução é do tipo R. É enviado um sinal de controle para a ULA, onde, através da identificação do Funct3 para a instrução and, será realizado um and entre os valores carregados na entrada. Busca os valores rs1 e rs2 no banco de registradores, onde tais valores seẽao passados para a ULA. O resultado que sair da ALU irá para o banco de registradores, onde vai salvar no registrador rd o resultado. O rd é escolhido como registrador de destino, no banco de registradores, recebendo o valor originado na operação na ULA.

2.1.4. slt;

- Instrução: slt rd, rs1, rs2
- Objetivo: Os valores dos registradores rs e rt são carregados e a operação comparação da ALU é selecionada, comparando os conteúdos dos registradores. Se o conteúdo de rs for menor que rt, é colocado o valor 1 no registrador rd, caso contrário é colocado 0.
- Algoritmo: após a identificação da operação slt no estágio de decodificação, os valores dos rs1 e rs2 são carregados para as entradas A e B da ULA, respectivamente. É enviado uma flag para a ULA e então será realizada uma comparação O reg1 é escolhido como registrador de destino, no banco de registradores, recebendo o menor valor.

2.1.5. Mul

- Instrução: mul rd, rs1, rs2
- Objetivo: Os valores dos registradores rs1 e rs2 são carregados e a operação comparação da ALU é selecionada, fazendo a multiplicação entre o conteúdo dos dois registradores.
- Algorítmo: Haverá uma comparação bit a bit, multiplicando o que foi enviado para a UC, onde para cada bit 1 do multiplicando será somado o registrador

C com o valor do registrador Soma, e no final os bits do registrador C serão shifitados para esquerda. Quando o bit do multiplicando for igual a 0, o registrador C apen.as será shifitado. E os 64 bits menos significativos serão salvos em rd.

2.1.6. Muh

- Instrução: mul rd, rs1, rs2
- Objetivo: Os valores dos registradores rs1 e rs2 são carregados e a operação comparação da ALU é selecionada, fazendo a multiplicação entre o conteúdo dos dois registradores.
- Algorítmo: Haverá uma comparação bit a bit, multiplicando o que foi enviado para a UC, onde para cada bit 1 do multiplicando será somado o registrador C com o valor do registrador Soma, e no final os bits do registrador C serão shifitados para esquerda. Quando o bit do multiplicando for igual a 0, o registrador C apenas será shifitado. E os 64 bits mais significativos serão salvos em rd.

2.2. Tipo I

2.2.1. nop:

- Objetivo: Abreviação para *No Operation*, Sem Operação. É uma instrução que efetivamente faz nada.
- Algoritmo: Após a seleção da opção de default no estágio de SELECAO, o valor de PC é carregado na entrada A da ULA e o valor 4 é carregado na entrada B da ULA, sendo realizada uma soma desses valores a partir do recebimento de um sinal de controle pela ULA. É carregado o valor referente ao tratamento da exceção no registrador PC, o registrador EPC recebe o valor resultante da operação realizada na ULA e uma nova instrução é buscada na memória.

2.2.2. addi:

- Instrução: addi rd, rs1, imm
- Objetivo: O imediato passa pelo Sign Extend e é carregado o valor do registrador rs, os quais são selecionados como entrada na ALU para a soma.

- Algoritmo: No estado de seleção, o controle do shift no registrador A é setado para que a saída permaneça igual a entrada. Na ULA é selecionado a operação de soma, o mux A deixa passar o valor contido no registrador A e no mux B o valor passado é dado pelo valor contido na saida do "signal extend". Apos a soma ocorrer na ULA, o estado muda para MEM_INST. Nesse estado, a saída da ALU_OUT já está disponível na entrada do banco de registradores, então é necessário ativar a escrita setando a o "bancoRegisters_wr" para 1. O load_ir da instrução é ativado e o estado muda para "BUSCA" para buscar a próxima instrução.

2.2.3. slti:

- Instrução: slti rd, rs1, imm
- Objetivo: O valor do registrador rs é carregado e o imediato passa pelo Sign Extend, a operação de comparação da ALU é selecionada, comparando seus conteúdos. Se o conteúdo de rs for menor que o imediato, é colocado o valor 1 no registrador rt, caso contrário é colocado 0.
- Algoritmo: Nessa instrução, o controle do shift próximo ao registrador A é setado para 3. De forma que o valor da saída seja idêntico ao valor da entrada. A operação de comparação da ULA é selecionada, os valores a serem comparados são o valor de rs1 que sai do mux A (mux A selecionando apenas o registrador A) e o mux B seleciona o valor que sai do immediate após ter o sinal estendido. O estado é mudado para "FLAG". No estado "FLAG" é analisado o valor das flags da própria Ula. Nesta operação é analisado o valor da flag "LT". Caso o valor desta flag seja 1, então o registrador A é resetado, a operação de incremento na ULA é selecionada e o valor "mux A" seleciona o registrador A, dessa forma a saída da Ula é 1. Caso contrário, o registrador A é resetado e a operação selecionada da ULA é apenas "carregar A", logo, a saída da ULA terá valor 0. Ambos os casos, o estado é forçado a mudar para "MEM_INST". No estado MEM_INST, a saída da ULA vai diretamente para a entrada do banco de registradores, e o sinal de escrita é ativado como 1. Possibilitando a escrita no registrador de destino (Rd). Após a escrita, o "load_ir" é ativado e o estado muda para "BUSCA", fazendo o programa ir para a próxima instrução.

2.2.4. srli:

- Instrução: srli rd, rs1, shamt
- Objetivo: Desloca um valor de registro diretamente pelo valor de deslocamento (shamt) e coloca o valor no registro de destino. Zeros são deslocados.
- Algoritmo: O shift do registrador A recebe 1, de forma que ele selecione a operação de shift a direita lógico, a Ula recebe 0 de forma que a operação de carregamento do valor do registrador A seja selecionada. Assim, o valor

contido no registrador A é o valor que estava contido no registrador 1 deslocado a direita de acordo com o "shamt". A Ula só é encarregada de carregar o valor do registrador 1 e mandar diretamente na entrada do banco de registradores. O estado muda para "MEM_INST". Neste novo estado, o mux do banco de registradores é setado para 0, permitindo que o valor da saída da ULA seja selecionado. O valor para escrita no banco de registradores é setado para 1, possibilitando a escrita no banco de registradores. "Load_ir" é setado para 1 possibilitando a lida de uma nova instrução e o estado muda para "BUSCA".

2.2.5. srai:

- Instrução: srai rd, rs1, shamt
- Objetivo: Desloca um valor de registro diretamente pelo valor de deslocamento (shamt) e coloca o valor no registro de destino. O bit de sinal é deslocado para dentro.
- Algoritmo: A operação é parecida com o srli, a diferença é que a extensão do sinal importa. O controle do shift do registrador “A“ seleciona a operação de shift aritmético à direita, no próprio módulo do shift é tratado a extensão do sinal. O bit mais significativo é avaliado antes do deslocamento; caso seja um, o valor final é concatenado com 1s; caso seja zero, o valor final é concatenado com 0s. A saída é passada para o registrador A. O mux da entrada A da ULA é selecionado escolhendo o registrador A, a operação "carrega A" da Ula é setada. O estado então muda para "MEM_INST". De maneira análoga, vista anteriormente, o estado "MEM_INST" apenas seleciona no mux da entrada do banco de registradores o valor da saída da ULA, a flag de escrita é setada para 1 e o valor é escrito no registrador de destino. Logo após tais operações, o a flag de leitura de instrução é setada para 1 e o estado muda para "BUSCA".

2.2.6. slli:

- Instrução: slli rd, rs1, shamt
- Objetivo: Desloca um valor de registro deixado pelo valor do turno listado na instrução e coloca o resultado em um terceiro registro. Zeros são deslocados.
- Algoritmo: A instrução slli funciona de maneira bastante semelhante as duas vistas anteriormente. O controle do shift seleciona a operação do "shift a esquerda lógico", em seguida, o valor shiftado é encaminhado para o registrador A. A ULA tem a operação de "carregar A" selecionada e o estado muda para "MEM_INST". Mais uma vez, o estado "MEM_INST" apenas seleciona o valor que sai da ULA por meio do mux presente antes do "datain" do banco de registradores. O valor para escrever no banco dos registradores é ativado e o valor da saída da ULA é adicionado no registrador de destino. Após essa operação ser concluída, o valor de permitir a leitura de instrução é ativada e o estado muda para "BUSCA".

2.2.7. break:

- Objetivo: Parar a execução
- Algoritmo: Zera todos os sinais do módulo Controle e entra em um loop infinito.

2.2.8. ld:

- Instrução: ld rd, imm(rs1)
- Objetivo: Uma palavra dupla é carregada em um registro do endereço especificado
- Algoritmo: Nessa instrução não será necessário o uso do shift, portanto ele recebe o valor 3 e deixa a saída da mesma forma da entrada. A ULA é selecionada de forma que opere a soma, as entradas da ULA são determinadas pelos muxes A e B. O mux A seleciona o registrador A, enquanto o mux B seleciona o valor que sai do mux do sinal extendido, pois o valor é dado pelo immediate da função. O valor que determina a escrita na memória é setado para 1 e o estado muda para "MEM_DATA". NO "MEM_DATA", o valor que determina a escrita na memória é setado com 0, impedindo nova escrita já que se passou um ciclo de clock e o estado muda para "MEM_INST_2". Neste novo estado, o seletor do mux do banco de registradores é setado com 1, pois o novo valor que vai para o registrador de destino vem da memória e não da ULA como anteriormente. O banco de registradores tem a escrita permitida sentado a flag de escrita com 1, o valor que permite a leitura de uma nova instrução é setado e o estado muda para "BUSCA".

2.2.9. lb:

- Instrução: lb, rd imm(rs1)
- Objetivo: Um bit é carregado no registrador do endereço especificado
- Algoritmo: De maneira quase semelhante ao ld,o lb apenas carrega um byte no valor destinado da memória. Para isto, o controle do shift é setado para que nenhuma operação de shift ocorra no valor que entrará no registrador "A", em seguida a ULA é setada com a operação de soma e o mux B seleciona o valor immediate vindo do sinal estendido. A flag que possiblida leitura na memória é setada permitindo que o valor da ULA acesse uma posicção da memória. O estado agora muda para "MEM_DATA". As operações no "MEM_DATA" ocorrem de maneira semelhante ao ld descrito anteriormente.

2.2.10. lh:

- Instrução: lh rd, imm(rs1)
- Objetivo: Uma half-word é carregado no registrador do endereço especificado

- Algoritmo: Apenas com a mudança de leitura de bytes, as operações são semelhantes ao lb e ld.

2.2.11. lw:

- Instrução: lw rd, imm(rs1)
- Objetivo: Uma palavra é carregada em um registro do endereço especificado
- Algoritmo: vide ld.

2.2.12. lbu:

- Instrução: lbu rd, imm(rs1)
- Objetivo: um byte é carregado na memória e estendido com zeros.
- Algoritmo: Maneira semelhante ao lb, a diferença está na concatenação de zeros realizada indiretamente pela ULA. Pois, o valor immediate já é extendido com zeros.

2.2.13. lhu:

- Instrução: lhu rd, imm(rs1)
- Objetivo: uma half-word é carregado na memória e estendido com zeros
- Algoritmo: maneira análoga ao lh, alterando apenas a concatenação de zeros.

2.2.14. lwu:

- Instrução: lwu rd, imm(rs1)
- Objetivo: uma palavra é carregada na memória e estendido com zeros.
- Algoritmo: Maneira análoga ao lw, a diferença está na concatenação de zeros que ocorre indiretamente pela ULA, pois o immediate tem o valor concatenado com zeros.

2.3. Tipo S

2.3.1. sd:

- Instrução: sd rs2, imm(rs1)
- Objetivo: A palavra dupla de rs2 é armazenado no endereço rs1+immediate
- Algoritmo: após a identificação da operação sd no estágio de SELECAO, o valor do rs2 é carregado a partir do banco de registradores para a entrada. É enviado um sinal de controle para a ULA, onde, através da identificaçao do OpCode para a instrução sd, será realizada uma soma dos valores carregados na entrada, cujo valor fica guardado em AluOut. No endereço referenciado pelo valor contido em AluOut, é escrito o valor contido no rs1 e uma nova instrução é buscada.

2.3.2. sw:

- Instrução: sw rs2, imm(rs1)

- Objetivo: A palavra de rs2 é armazenado no endereço rs1+immediate
- Algoritmo: vide o sd

2.3.3. sh:

- Instrução: sh rs2, imm(rs1)
- Objetivo: A half-word de rs2 é armazenado endereço rs1+immediate
- Algoritmo: vide o sd

2.3.4. sb:

- Instrução: sb rs2, imm(rs1)
- Objetivo: O bit de rs2 é armazenado no endereço rs1+immediate
- Algoritmo: vide o sd

2.4. Tipo SB

2.4.1. beq:

- Instrução: beq rs1, rs2, imm
- Objetivo: Desvia, se rs1 for igual a rs2.
- Algoritmo: O deslocador funcional não faz nada, a operação de comparação da ULA é selecionada e o mux "A" seleciona o valor do registrador "A", enquanto o mux "B" seleciona o valor do registrador "B". O estado agora vai para o "SALTO". Nesse estado, se a flag da ULA "ET" estiver com o valor 1, então a ULA é selecionada para realizar uma soma e o endereço de PC é selecionado pelo MUX "A", enquanto o valor do MUX "B" é selecionado pelo immediante que vem do sinal estendido. Logo após, PC_WRITE é setado como 1, banco de registradores tem a escrita bloqueada e o "load_ir" é ativado. O estado troca para a BUSCA.

2.4.1. bne:

- Instrução: bne rs1, rs2, imm
- Objetivo: Desvia, se rs1 for diferente a rs2.
- Algoritmo: Ocorre de maneira análoga ao beq, a diferença está em verificar se a flag “ET” da ULA está setada com zero. Se sim, realiza o desvio descrito no beq.

2.4.1. bge:

- Instrução: bge rs1, rs2, imm
- Objetivo: Desvia, se rs1 é maior do que/ ou igual a rs2.
- Algoritmo: De maneira análoga ao beq, realiza o desvio apenas se a flag “ET” ou “GT” da ula estiverem setadas com 1. Se sim, realiza o desvio como descrito anteriormente no beq.

2.4.1. blt:

- Instrução: blt rs1, rs2, imm
- Objetivo: Desvia, se rs1 for menor do que rs2
- Algoritmo: De maneira análoga ao beq, a diferença está na verificação da flag “LT”, se ela estiver setada com 1, então realiza o desvio, caso contrário não realiza.

2.5. Tipo U

2.5.1. lui:

- Instrução:lui rd, imm
- Objetivo: O valor imediato é deslocado para a esquerda 16 bits e armazenado no registrador. Os 16 bits mais baixos são zeros
- Algoritmo: Trava o registrador de instrução setando seu "load_ir" com zero. Após isso, reseta o registrador A e seleciona o "shift_control" para que ele não opere e deixe a saída da mesma forma que a entrada. Seleciona a operação de soma da ULA e os valores da entrada são selecionados pelos muxes, o mux A seleciona o registrador A (zerado), enquanro o mux_B seleciona o valor contido no immediate que é dado pela extenção de sinal. O estado muda para "MEM_INST" já descrito anteriormente.

2.6. Tipo UJ

2.6.1. jal:

- Instrução: jal rd, imm
- Objetivo: Salta para o endereço calculado e armazena o endereço de retorno na pilha.
- Algoritmo: Nesta instrução, a ula é selecionada para carregar apenas o valor de PC. Pois, o mux A seleciona o valor de PC e na operação da ULA é selecionado o valor de carregar PC. Em seguida o estado muda para "MEM_INST". Como a instrução é do tipo jal, o estado muda para o estado de "SALTO". Neste novo estado, a ULA é selecionada para realizar uma soma, o MUX A seleciona o valor de PC, enquanto o Mux B seleciona o immediate. PC_write é ativado e o estado muda para BUSCA.

3. Descrição dos Estados de Controle

3.1. BUSCA

- É realizada a leitura da instrução, e paralelamente PC é somado + 4 .

Próximo: SELECAO.

3.2. SELECAO

- Ocorre a decodificação da instrução, de acordo com o opcode e os valores em funct3 e funt7 (se existirem), é selecionado o tratamento das instruções e os comandos necessários, como setar valores para escrita no banco de registrador, setar quando deve-se mudar de estado, entre outros.

3.3. SALTO

- Tratamento de instruções de desvios. Dependendo da instrução, beq, bne, jal, etc, seta valores para o desvio acontecer.

3.4. MEM_INST

- Escrita no rd o que vem da entrada 0 (ULA) do mux. Esse estado seta o bit de escrita do banco de registradores e seleciona a entrada 0 do MUX.

3.5. MEM_INST_2

- Escreve no rd o que vem da entrada 1(Memória de Dados) do mux. Esse estado seta o bit de escrita do banco de registradores e seleciona a entrada 0 do MUX.

3.6. MEM_DATA

- Permite que o valor no endereço rs1+immediate(ALU_OUT) seja lido.

3.7. MEM_DATA_2

- Permite a memória de dados guardar valor de rs2 no endereço rs1+immediate.

3.8. FLAG

- Analise de flags.

3.9. NOP

- Estado auxiliar, para a instrução nop.

3.10. EXECECAO

- Estado que trata os dois tipos de exceção.

3.11. EXECECAO_OVEFLOW

- Estado que trata a exceção de overflow.

3.12. EXECECAO_INEXISTENTE

- estado que trata a exceção de opcode ou function inexistente.

3.13. WAIT_MEM

- Estado para espera de carregamento da instrução na memória, no tratamento de exceção.

3.14. WAIT_EXTEND

- Estado para espera de valor carregado em WAIT_MEM no registrador PC no tratamento de exceção.

3.15. WAIT_EPC_SOMA

- Estado para espera de valor carregamento da instrução que causou a exceção no registrador EPC no tratamento de exceção.

3.16. WAIT_PC

- Estado para espera de carregamento do valor da memória de instrução em PC no tratamento de exceção.

3.17. MULT_COMPARAR

- Estado onde há a comparação entre os valores nos registradores C e Soma.

3.18. MULT_SOMA

- Estado onde há a soma entre os conteúdos dos registradores C e Soma.

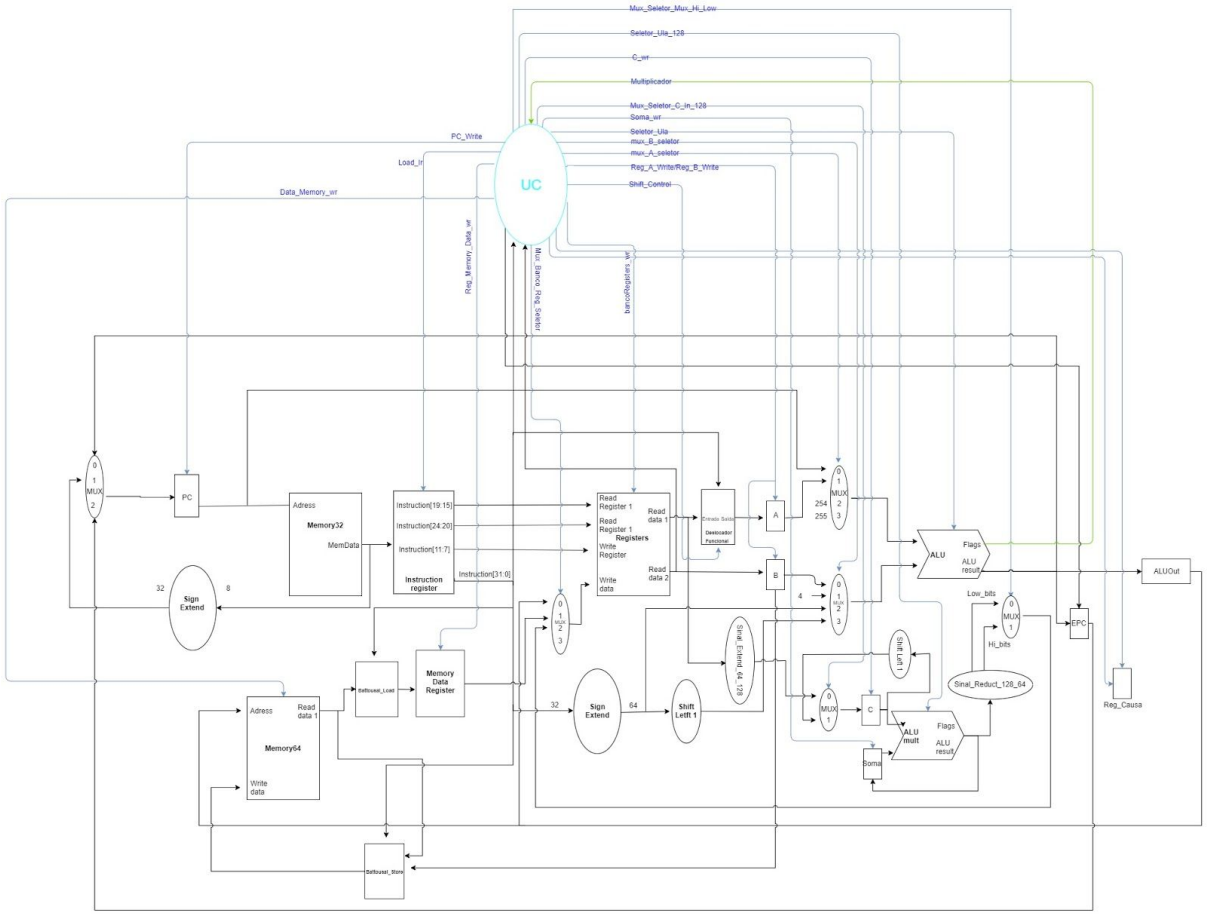
3.19.MULT_SHIFT

- Estado onde há o shift_right no conteúdo do valor do registrador C.

3.20.MULT_WRITE_REG

- Estado onde haverá a escrita no registrador rd na operação mul, muh.

4. CPU


Mux_Seletor_Mux_Hi_Low
Seletor_Ula_128
C_wr
Multiplicador
Mux_Seletor_C_in_128
Soma_wr
Seletor_Ula
mux_B_seletor
mux_A_seletor
Reg_A_Write/Reg_B_Write
Shift_Control
PC_Write
Load_Ir
Data_Memory_wr
UC
Reg_Memory_Data_wr
Mux_Banco_Reg_Seletor
banco/Registers_wr
0
1
MUX
2
PC
Adress
Memory32
MemData
Instruction[19:15]
Instruction[24:20]
Instruction[11:7]
Instruction register
Instruction[31:0]
Read Register 1
Read Register 1
Registers
Write Register
Write data
Read data 1
Read data 2
Conexão Saída
Deslocador Funcional
A
B
4
254
255
3
ALU
Flags
ALU result
ALUOut
Low_bits
Hi_bits
EPC
Reg_Causa
32
Sign Extend
8
64
Shift Left 1
Sinal_Extend_64_128
Sinal_Reduct_128_64
C
ALU mult
Soma
Memory Data Register
Memory64
Write data
Read data 1
Dataout_Load
Dataout_Store