UNIVERSIDADE FEDERAL DE MINAS GERAIS

IMPLEMENTAÇÃO DE PROCESSADORES SUPERSCALARES
I2OI & IO2I

Trabalho final da disciplina de ORGANIZAÇÃO DE COMPUTADORES II

Alunos:

Lecio Charlles
Breno Pimenta
Tiago Coelho Magalhães

Prof. Omar Paranaiba Vilela Neto

BELO HORIZONTE
2020

1 **INTRODUÇÃO**

Este trabalho prático consiste em descrever as partes fundamentais dos processadores superescalares utilizando a linguagem de descrição de hardware (*Hardware Description Language*, HDL) Verilog, que permite o modelamento de sistemas eletrônicos ao nível de circuito.

É sabido que processadores superescalares tem como objetivo maximizar a capacidade de execução de instruções por ciclo de clock (**CPI < 1**), o que se dá por meio de um paralelismo a nível de instruções. Os processadores superescalares podem ser caracterizados por realizar determinadas etapas do ciclo de processamento em ordem (*in-order*) ou fora de ordem (*out-of-order*). A *Tabela 1* ilustra alguma destas possibilidades e evidencia a existência de blocos específicos desta tecnologia. É possível observar que a busca da instrução (*fetch*) ocorre sempre em ordem, pois, do contrário, teríamos uma aleatoriedade na execução do programa, o qual possui uma lógica programada sequencialmente.

Name	Frontend	Issue	Writeback	Commit	
I4	IO	IO	IO	IO	Fixed Length Pipelines Scoreboard
I2O2	IO	IO	OOO	OOO	Scoreboard
I2OI	IO	IO	OOO	IO	Scoreboard, Reorder Buffer, and Store Buffer
I03	IO	OOO	OOO	OOO	Scoreboard and Issue Queue
IO2I	IO	OOO	OOO	IO	Scoreboard, Issue Queue, Reorder Buffer, and Store Buffer

Tabela 1 - Tipos de processadores superescalares

É válido ressaltar que o desempenho dos processadores também depende do compilador utilizado, o qual é responsável por agendar as instruções de forma a minimizar as dependências e, consequentemente, evitar *hazards*.

O presente trabalho tem como proposta abordar a implementação dos processadores I2OI e IO2I. Um subconjunto de instruções do MIPS fora proposto, com alteração de *opcodes*, para a realização das operações lógicas, aritméticas, desvios e de memória (*load* e *store*), conforme *Tabela 2*.

```
Nao faz nada:
NOP
00000000000000000000000000000000

Tipo R
31:26   25:21   20:16   15:11   10:0
DESCRICAO: rC = rA OPER rB

opcode  rA      rB      rC      funct
ADD
000001  00001   00001   00001   00000000001
SUB
000001  00001   00001   00001   00000000010
AND
000001  00001   00001   00001   00000000100
MUL
000001  00001   00001   00001   00000001000
SLT
000001  00001   00001   00001   00000010000


Tipo Imm
DESCRICAO: rB = rA OPER imm
31:26   25:21   20:16   15:0
opcode  rA      rB      imm
ADDI
000101  00001   00001   0000000000000000

Desvio

opcode  rA      rB      offset
BEQ
if(rA == rB) PC += offset*4
010000  00000   00000   0000000000000000
if(rA == rB) PC -= offset*4
010000  00000   00000   1000000000000000
BLT
if(rA < rB) PC += offset*4
110000  00010   00000   0000000000000000
if(rA < rB) PC -= offset*4
110000  00010   00000   1000000000000000


Memoria
DESCRICAO: rB = MEM[offset + rA]
opcode  rA      rB          offset
LW
001001  00010   00100   0000000000000000
MEM[offset + rA] = rB
SW
001000  00010   00010   0000000000000000
```

Tabela 2 - Definição de Instruções

2 **PROCESSADORES SUPERESCALARES**

Na sequência, serão apresentadas as arquiteturas projetadas para os processadores I2OI e IO2I e as decisões de projeto adotadas durante o planejamento.

Nessas duas arquiteturas, bem como em outras arquiteturas superescalares, várias instruções podem ser executadas simultaneamente e de forma independente umas das outras (desde que estejam em estágios diferentes do pipeline), dada a existência de um caminho para as instruções associadas a diferentes operações, como aritméticas, lógicas e desvios (seguindo pelo caminho de X0), de *load* (pelo caminho de L0 e L1), *store* (pelo caminho de S0), e de multiplicação (pelo caminho de Y0, Y1, Y2 e Y3). Como já comentado anteriormente, o disparo da execução ou a escrita do resultado na memória podem acontecer em ordem ou fora de ordem. Basicamente, são estas as características que diferenciarão os processadores que serão desenvolvidos neste trabalho.

2.1 Processador I2OI

A *Figura 1* apresenta a arquitetura básica do processador superescalar I2OI.

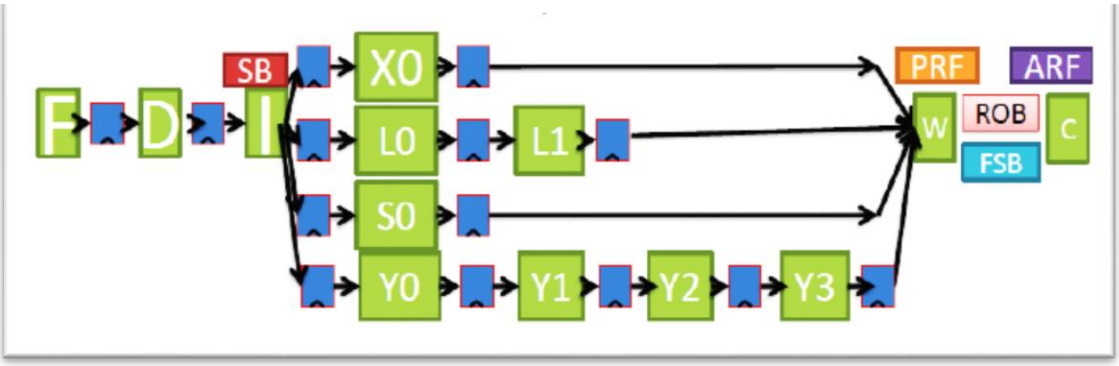

Figura 1 - Processador I2OI (teórico)

Além dos caminhos associados a execução das instruções propriamente ditas, a *Figura 1* evidencia um conjunto de estágios iniciais, que inclui FETCH, responsável pela busca de instruções; um estágio de DECODE, que interpreta os diversos operandos associados às instruções programadas; bem como o estágio de ISSUE, o qual verifica os conflitos de recursos para determinar quais instruções podem ser disparadas em cada etapa. Para esse processador, as instruções serão disparadas em ordem, entretanto, um estágio posterior à execução das instruções, o WRITEBACK ocorre fora de ordem. Um último estágio de COMMIT, que atualiza a memória, acontece em ordem.

Um ponto a ser destacado é a existência de um bloco SB, denominado *scoreboard*, associado ao módulo de ISSUE. Para o processador I2OI, e levando em consideração o compilador que organiza as instruções de forma a evitar dependências, como descrito anteriormente, o scoreboard construído para o projeto objetiva tratar *hazards* estruturais, construindo uma tabela de controle do uso e disponibilidade dos dados, ao longo do pipeline de execução.

A implementação do processador buscou evidenciar, de maneira visível, os mesmos blocos da *Figura 1*, e resultou na implementação vista na *Figura 17* (Ap. A).

2.1.1 Detalhamento dos Blocos

O projeto buscou apresentar os módulos principais da arquitetura da *Figura 1*, os quais serão detalhados a seguir, incluindo todos os submódulos existentes.

2.1.1.1 FECTH

O módulo é apresentado na *Figura 2*. Sua funcionalidade é a busca de instruções e, por isso, associado a ele está o submódulo contador de programa (*Program Counter*, PC), que contém o endereço da instrução que está sendo executada no momento. Conforme cada instrução é buscada, o contador do programa aumenta seu valor armazenado em 4. Existe, entretanto, uma lógica associada que decide se o endereço do PC será atualizado com o valor do “fluxo normal” ou do endereço de um eventual desvio.

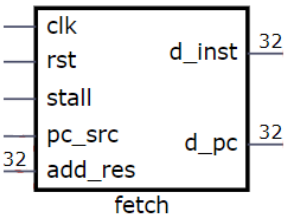

Figura 2 - Módulo FETCH

Nome	Input/Output	Tamanho	Função
clk	Input	1	Sinal de clock global
rst	Input	1	Sinal de reset global
stall	Input	1	Sinal de paralisação de pipeline (atraso na execução de uma instrução para resolver um *hazard*)
pc_src	Input	1	Informar se o PC será atualizado com o valor de desvio ou com +4
add_res	input	32	Endereço em caso de desvio
d_inst	output	32	Instrução buscada
d_pc	output	32	Valor de endereço da instrução atual

Tabela 3 - Descrição de Sinais do Módulo FETCH

2.1.1.2 DECODE

O módulo decodificador de instruções é apresentado na *Figura 2*.

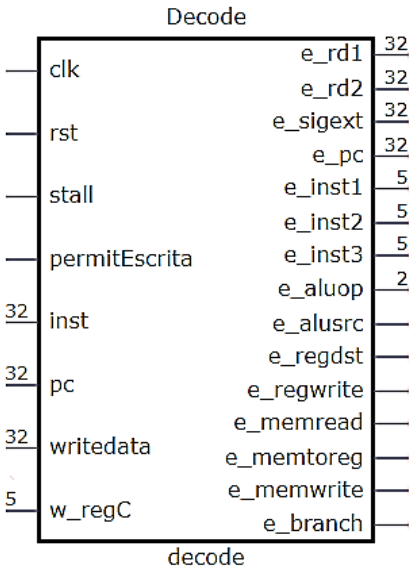

Figura 3 - Módulo DECODE

Nome	Input/Output	Tamanho	Função
clk	Input	1	Sinal de clock global
rst	Input	1	Sinal de reset global
stall	Input	1	Sinal de paralisação de pipeline (atraso na execução de uma instrução para resolver um *hazard*)
permitEscrita	Input	1	Sinal de controle para escrita no banco de registradores
inst	input	32	Instrução a ser decodificada
pc	Input	32	Endereço no contador de programa
writedata	Input	32	Valor a ser gravado no banco de registradores
w_regC	Input	5	endereço para escrita nos registradores, regC de inst tipo R e destino do load
e_rd1	output	32	Dado associado ao primeiro operando
e_rd2	output	32	Dado associado ao segundo operando
e_sigext	output	32	Valores imediatos/externos
e_pc	output	32	Endereço no contador de programa
e_inst1	output	5	Endereço primeiro operando
e_inst2	output	5	Endereço segundo operando
e_inst3	output	5	Endereço de destino, caso seja tipo R
e_aluop	output	2	Operador da ALU
e_alusrc	output	1	Sinal de controle
e_regdst	output	1	Sinal de controle
e_regwrite	output	1	Sinal de controle
e_memread	output	1	Sinal de controle
e_memtoreg	output	1	Sinal de controle
e_memwrite	output	1	Sinal de controle
e_branch	output	1	Sinal de controle

Tabela 4 - Descrição de Sinais do Módulo DECODE

Os submodulos do DECODE são mostrados na *Figura* 4, sendo os principais, *RegisterBank*, que corresponde a um banco de registradores; *Control*, responsável por disparar os sinais de controle para o processador; e *IDI*, um bloco intermediário

de registradores que armazena os valores de saída, que serão enviados ao próximo bloco, realiza o controle dos dados no momento do stall, reseta os valores em caso do RST e apenas replica os sinais de entrada para a saída durante a execução do processador.: ISSUE.

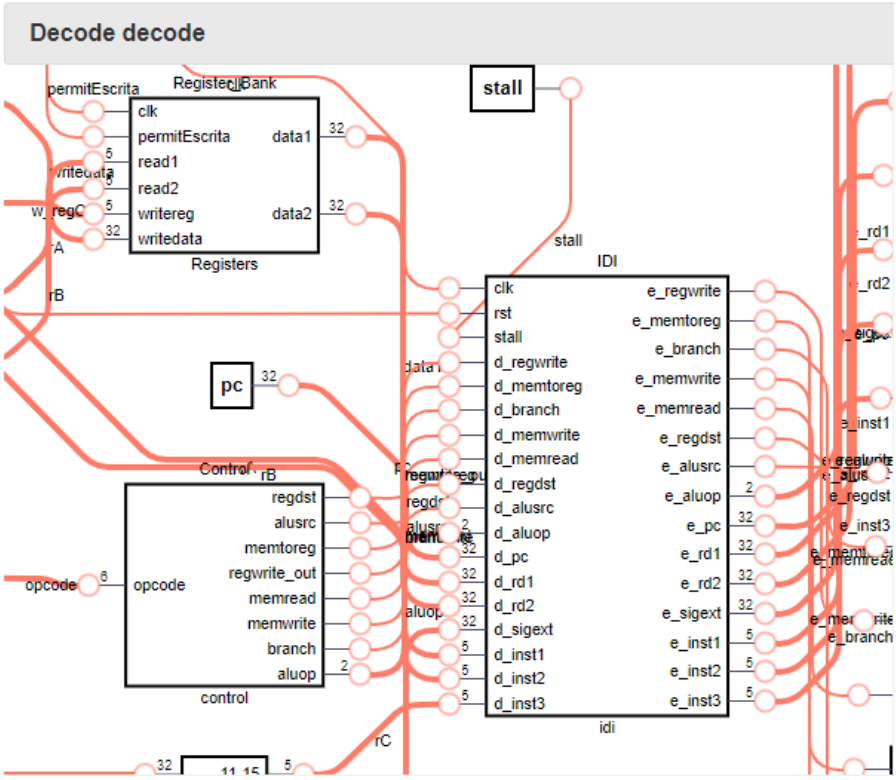

Figura 4 - Submodulos do Módulo DECODE

Os sinais de controle do módulo CONTROL são disparados de acordo com o *opcode* definido à princípio. Parte do código que atualiza o estado dos sinais de controle para a instruções tipo R é mostrado na *Figura 5*

```
always @(opcode)
begin
    case(opcode)
        6'b000001:
        begin // Tipo R
            regdst <= 1 ;//reg de destino, apenas tipo r usa regC
            alusrc <= 0 ; //se tem imediato, alusrc = 1 , uso no X0
            memtoreg <= 0 ; //caso load
            regwrite_out <= 1 ; //indica inst que escreve no reg; todas exceto
            memread <= 0 ; //le da memoria, apenas load
            memwrite <= 0 ;//escreve mem, apenas store
            branch <= 0 ; //indica se eh desvio, apenas branch
            aluop <= 2'b10 ; //indica tipo R
        end
```

Figura 5 - Parte do código do módulo de controle

2.1.1.3 ISSUE & SCOREBOARD

O módulo ISSUE é apresentado na *Figura 6*. Sua funcionalidade é disparar as instruções para que possam ser executadas. Com relação ao processador I2OI, o disparo da execução das instruções acontece em ordem (*in-order*). Para tanto, o módulo conta com o módulo SCOREBOARD para agendar dinamicamente a execução das instruções quando não houver *hazards* e o hardware estiver disponível.

Uma decisão de projeto adotada para a implementação do disparo da instrução a ser executada é a criação de um sinal "**e_select_execute**" responsável por informar o caminho responsável pelo processamento da instrução, ou seja, através de X0 ou L0/L1 ou M0/M1/M2/M3 ou S0.

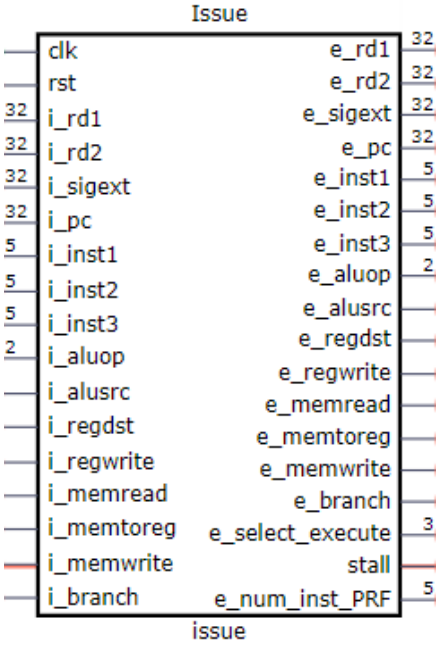

Figura 6 - Módulo ISSUE

Nome	Input/Output	Tamanho	Função
clk	Input	1	Sinal de clock global
rst	Input	1	Sinal de reset global
i_rd1	Input	32	Dado associado ao primeiro operando
i_rd2	Input	32	Dado associado ao segundo operando
i_sigext	Input	32	Valores imediatos/externos
i_pc	Input	32	Endereço no contador de programa
i_inst1	Input	5	Endereço primeiro operando
i_inst2	Input	5	Endereço segundo operando
i_inst3	Input	5	Endereço de destino, caso seja tipo R
i_aluop	Input	2	Operador da ALU
i_alusrc	Input	1	Sinal de controle
i_regdst	Input	1	Sinal de controle
i_regwrite	Input	1	Sinal de controle
i_memread	Input	1	Sinal de controle
i_memtoreg	Input	1	Sinal de controle
i_memwrite	Input	1	Sinal de controle
i_branch	Input	1	Sinal de controle

e_rd1	output	32	Dado associado ao primeiro operando
e_rd2	output	32	Dado associado ao segundo operando
e_sigext	output	32	Valores imediatos/externos
e_pc	output	32	Endereço no contador de programa
e_inst1	output	5	Endereço primeiro operando
e_inst2	output	5	Endereço segundo operando
e_inst3	output	5	Endereço de destino, caso seja tipo R
e_aluop	output	2	Operador da ALU
e_alusrc	output	1	Sinal de controle
e_regdst	output	1	Sinal de controle
e_regwrite	output	1	Sinal de controle
e_memread	output	1	Sinal de controle
e_memtoreg	output	1	Sinal de controle
e_memwrite	output	1	Sinal de controle
e_branch	output	1	Sinal de controle
e_select_execute	output	3	informa o caminho responsável pela execução da instrução
stall	output	1	Sinal de stall
e_num_inst_PRF	output	5	sinal associado a escrita in-order no Reorder Buffer

Tabela 5 - Descrição dos sinais do módulo ISSUE

Como já mencionado, um submódulo denominado ***selectExPipe*** é responsável por informar o caminho responsável pela execução da instrução através dos sinais traduzidos do opcode, ou seja, seleciona entre X0 ou L0/L1 ou M0/M1/M2/M3 ou S0. Esse sinal, ainda, está associado ao quadro de scoreboard proposto, o qual indica o estado de cada unidade funcional. Para que seja possível indicar qual unidade de função escreverá os resultados da operação, um sinal "***num_inst_PRF***" direciona as escritas em ordem.

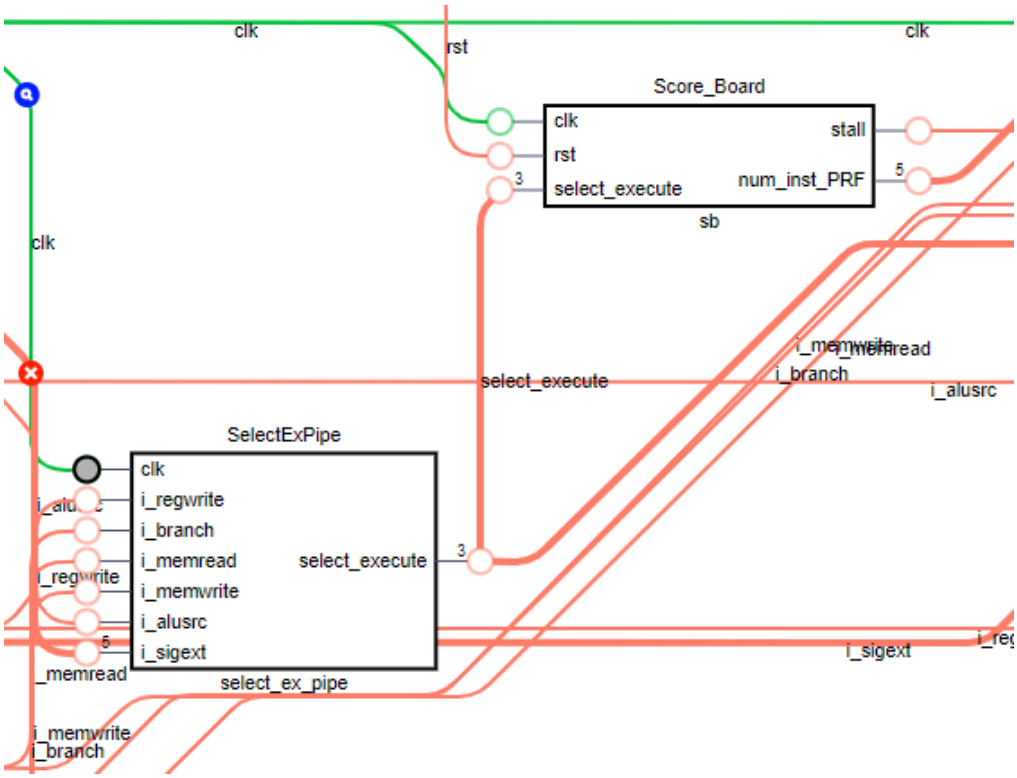

Figura 7 - Submódulo SCOREBOARD e SelectExecute

2.1.1.4 EXECUTE

O conjunto de módulos associados a execução das instruções é apresentado na *Figura 8* e os módulos específicos de X0 (verde), S0 (roxo), L0/L1 (preto), e M0/M1/M2/M3 (azul) estão destacados.

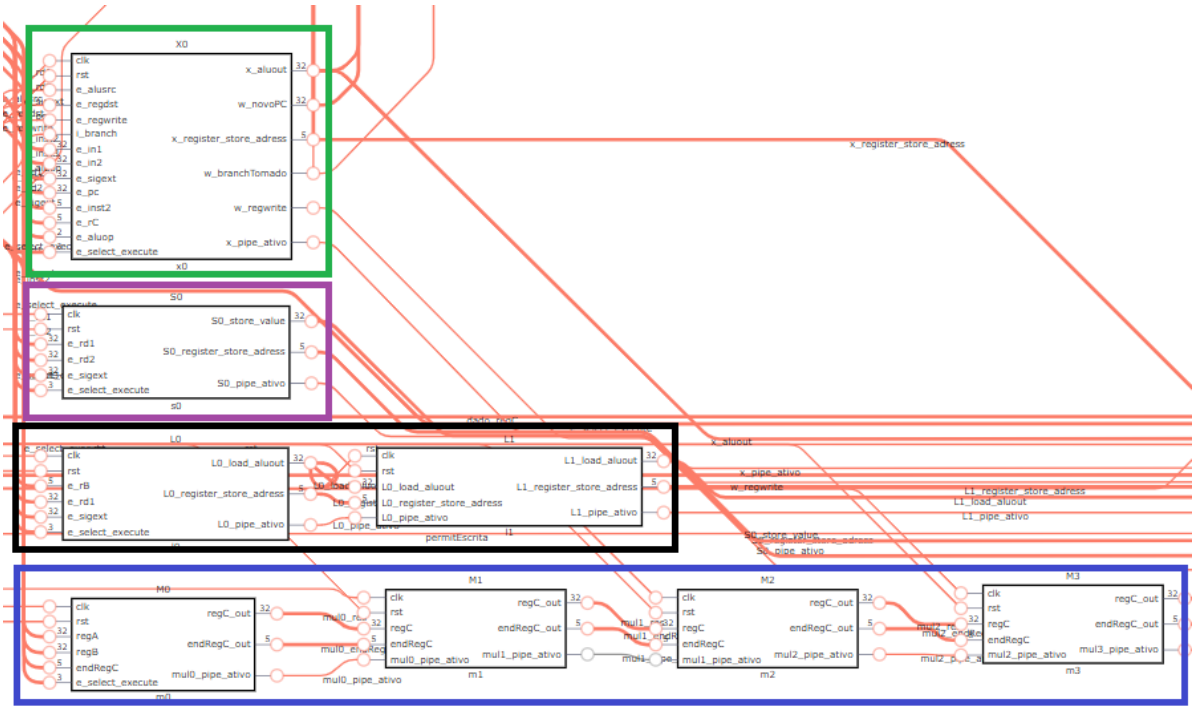

Figura 8 - Módulo EXECUTE e seus respectivos caminhos de execução

Os sinais destes módulos são apresentados na *Tabela 6*.

Nome	Input/Output	Tamanho	Função
clk	Input	1	Sinal de clock global
rst	Input	1	Sinal de reset global
e_in1	Input	32	Dado associado ao primeiro operando
e_in2	Input	32	Dado associado ao segundo operando
e_sigext	Input	32	Valores imediatos/externos
e_pc	Input	32	Endereço no contador de programa
i_inst1	Input	5	Endereço primeiro operando
e_inst2	Input	5	Endereço segundo operando
e_rc	Input	5	Endereço de destino, caso seja tipo R
i_aluop	Input	2	Operador da ALU
e_alusrc	Input	1	Sinal de controle
e_regdst	Input	1	Sinal de controle
e_select_execute	Input	3	informa o caminho pela execução da instrução
x_aluout	output	32	Resultado da operação
w_novoPC	output	32	Novo valor do PC em caso de desvio
x_register_store_adress	output	5	Endereço de destino para salvar resultado
w_branchTomado	output	1	Sinal de controle
w_regwrite	output	1	Sinal de controle
w_pipe_ativo	output	1	Endereço segundo operando
S0_store_value	output	32	Valor a ser armazenado
S0_register_store_adress	output	5	Endereço de memória para salvar resultado
S0_pipe_ativo	output	1	Sinal de controle
Lx_load_aluout	output	32	Valor a ser carregado

Lx_register_store_adress	output	5	Endereço para load
lx_pipe_ativo	output	1	Sinal de controle
regC_out	output	32	Resultado da operação
endRc_out	output	5	Endereço de memória para salvar resultado
mulx_pipe_ativo	output	1	Sinal de controle

Tabela 6 - Sinais associados à etapa de execução

2.1.1.4.1 X0

O módulo X0 é apresentado na *Figura 9*. É responsável pela execução das operações aritméticas, lógicas e desvios condicionais.

Figura 9 - Módulo X0

2.1.1.4.2 S0

O módulo S0 é apresentado na *Figura 10*, responsável pela execução das instruções de armazenamento (*store*).

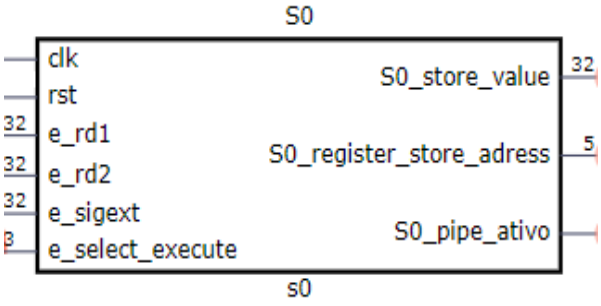

Figura 10 - Módulo S0

2.1.1.4.3 L0 e L1

A seguir, os módulos para a execução das instruções de LOAD (*Figura 11*).

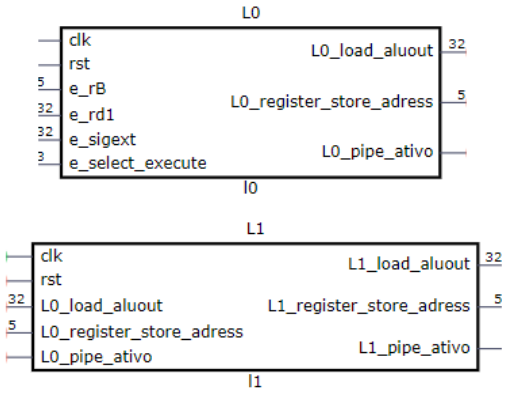

Figura 11 - Módulo LOAD

2.1.1.4.4 MULTIPLICAÇÃO

O caminho responsável pela execução das instruções de multiplicação é apresentado na Figura 12.

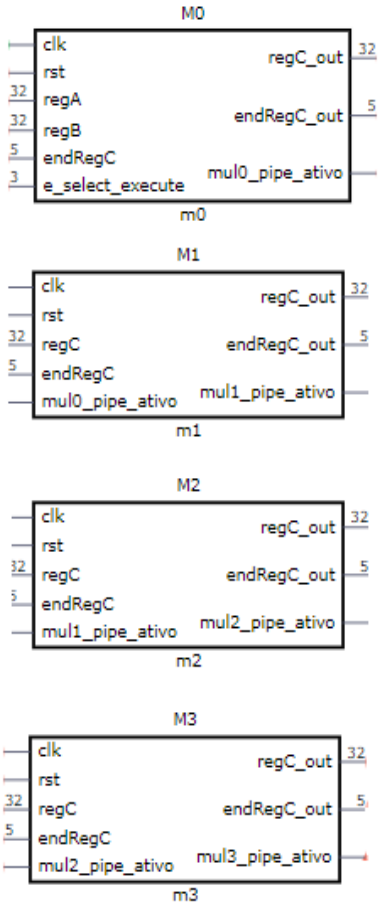

Figura 12 – Módulos da execução da multiplicação

2.1.1.5 WriteBack

O módulo WriteBack é apresentado na *Figura 13*.

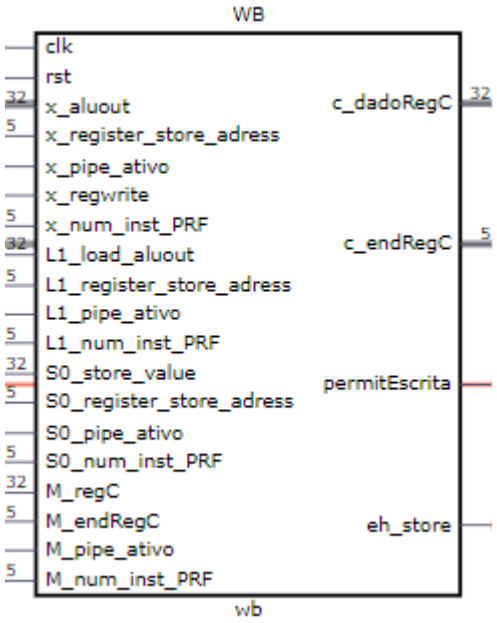

Figura 13 - Módulo WriteBack

Os respectivos sinais são descritos na *Tabela 7*.

Nome	Input/Output	Tamanho	Função
clk	Input	1	Sinal de clock global
rst	Input	1	Sinal de reset global
x_aluout	Input	32	Resultado da operação de X0
x_register_store_adress	Input	5	Endereço de destino para salvar resultado deX0
x_pipe_ativo	Input	1	Sinal de controle
Lx_load_aluout	Input	32	Valor a ser carregado
Lx_register_store_adress	Input	5	Endereço para load
lx_pipe_ativo	Input	1	Sinal de controle
S0_store_value	Input	32	Valor a ser armazenado
S0_register_store_adress	Input	5	Endereço de memória para salvar resultado
S0_pipe_ativo	Input	1	Sinal de controle
M_regC	Input	32	Resultado da operação de Multiplicação
M_endRegC	Input	5	Endereço de memória para salvar resultado de M
M_pipe_ativo	Input	1	Sinal de controle
M_num_inst_PRF	input	1	Sinal de controle
dado_regC	output	32	Valor a ser escrito na memória -> Commit
end_RegC	output	5	Endereço a ser escrito na memória -> Commit
permiteEscrita	output	1	Sinal de controle
eh_store	output	1	Indica escrita na memoria de dados, caso seja store

Tabela 7 - Sinais associados ao módulo de WriteBack

O PRF foi implementado dentro do Reorder Buffer, o qual, por sua vez, fora implementado dentro do WriteBack. Para esse projeto do I2OI, as instruções são disparadas em ordem e, nesse momento, associamos uma numeração a cada uma dessas instruções.

Quando as instruções chegam no WriteBack, mesmo fora de ordem, são sempre armazenadas no PRF usando como índice a numeração dada ao disparar. Assim, o Reorder Buffer pode verificar a disponibilidade dos índices em ordem e, caso a próxima instrução esteja disponível, ela é enviada para o COMMIT, realizando assim uma escrita final em ordem.

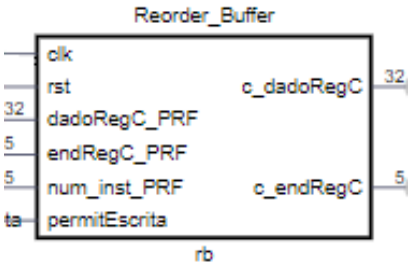

Figura 14 - Reorder Buffer

2.2 Processador IO2I

A *Figura 15* apresenta a arquitetura básica do processador superescalar IO2I.

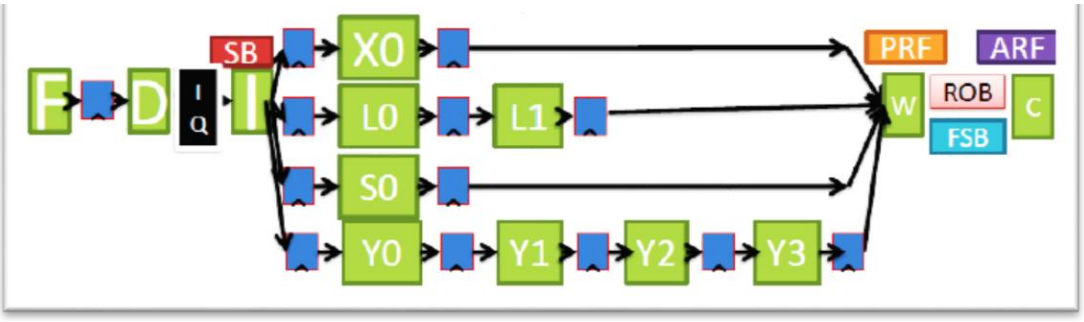

Figura 15 - Processador IO2I (teórico)

A grande diferença entre este processador e o I2OI, é a ordem de disparo das instruções. O IO2I utiliza de uma unidade funcional denominada Issue Queue, responsável pelo disparo fora de ordem das instruções (*out-of-order*). Optamos por utilizar o IssueQueue no formato centralizado, devido a facilidade da implementação, pois já tinhamos um módulo Issue único no processador anterior.

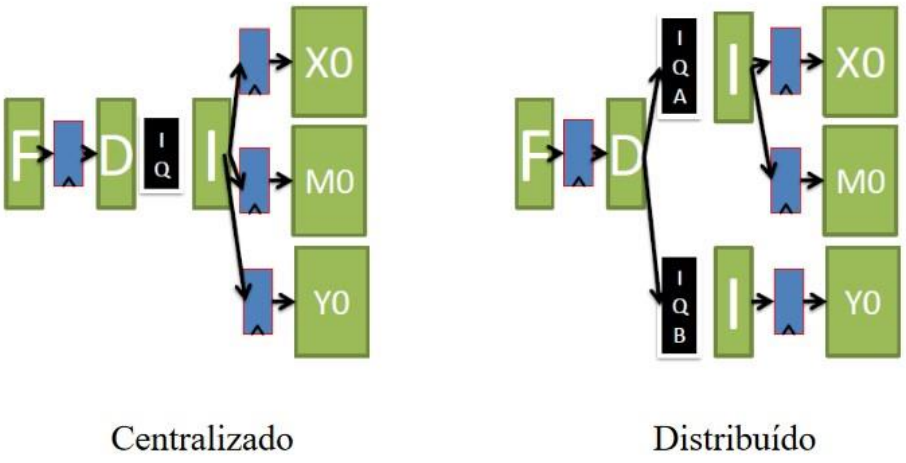

Figura 16 – Tipos de Issue Queue

Conseguimos construir o Issue Queue como descrito acima, no entanto, ao integrá-lo ao pipeline principal do processador não obtivemos erro ao executá-lo.

REFERÊNCIAS

Wolf, Marilyn. *High performance embedded computing: architectures, applications, and methodologies*, 2nd edition. Elsevier Inc., 2014. ISBN 978-0-12-410511-9.

Notas de aula: DCC007 – Organização de Computadores. Prof. Omar Paranaiba Vilela Neto. UFMG, Departamento de Ciência da Computação.

APÊNDICE A — Diagrama completo dos Processadores

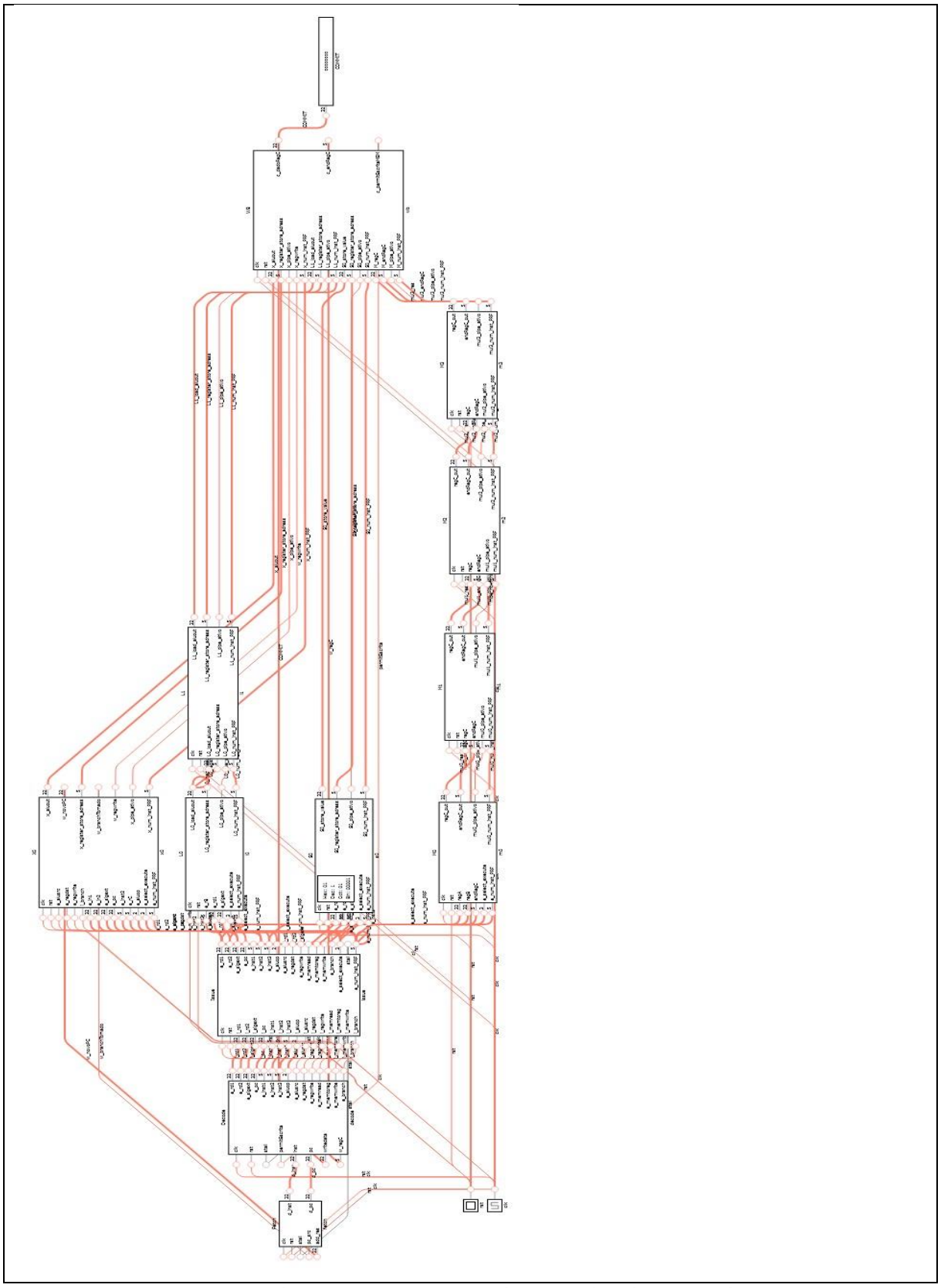

Figura 17 - Processador I2OI implementado