



**Especificação do Projeto**

Processador LAPI DOpaCA LAMBA

Universidade Estadual de Feira de Santana

**Build 2.1**

## Histórico de Revisões

Date	Descrição	Autor(s)
xx/xx/xxxx	<Descrição>	<Autor(es)>
08/12/2015	Teste	Filipe Boaventura
xx/xx/xxxx	<ul style="list-style-type: none"><li>• Exemplo de;</li><li>• Revisões em lista;</li></ul>	<Autor(es)>

## SUMÁRIO

<b>1</b>	<b>Introdução</b>	<b>3</b>
1.1	Propósito . . . . .	3
1.2	Document Outline Description . . . . .	3
1.3	Acronyms and Abbreviations . . . . .	3
<b>2</b>	<b>Visão Geral</b>	<b>4</b>
2.1	Perspectives . . . . .	4
2.2	Características Principais . . . . .	4
2.3	Non-functional Requirements . . . . .	4
<b>3</b>	<b>Conjunto de Instruções</b>	<b>5</b>
3.1	Instruções de Acesso à Memória . . . . .	5
3.2	Instruções com Constantes . . . . .	5
3.3	Instruções ALU . . . . .	6
3.4	Transferência de Controle . . . . .	7
3.5	CPU Load and Store Instructions . . . . .	7
3.6	Computational Instructions . . . . .	8
3.7	Jump/Branch Instructions . . . . .	9
3.8	No Operation Instruction . . . . .	9
3.9	End of Operations . . . . .	9
<b>4</b>	<b>Internal General Purpose Registers</b>	<b>9</b>

## 1. Introdução

### 1.1. Propósito

O objetivo desse documento é definir as especificações para o processador **LAPI DO-paCA LAMBA**, incluindo o seu montador e simulador. Aqui são definidos os parâmetros de implementação do processador que incluem: modos de operação, arquitetura do conjunto de instruções (ISA), e a descrição dos registradores e flags. Este documento está organizado da seguinte forma: Visão Geral na seção 2, Arquitetura do Conjunto de Instruções seção 3, Montador seção 4, Simulador seção 5.

### 1.2. Document Outline Description

Este documento está organizado da seguinte forma:

- Section 2: Apresenta uma visão geral do processador.
- Section 3: Especifica a arquitetura do conjunto de instruções.
- Section 4: Describes the general purpose registers.

### 1.3. Acronyms and Abbreviations

Acronym	Description
RISC	Reduced Instruction Set Computer
GPR	General Purpose Registers
FPGA	Field Gate Programmable Array
GPPU	General Purpose Processing Unit
SPRAM	Single-port RAM
HDL	Hardware Description Language
CPU	Central Processing Unit
ISA	Instruction Set Architecture
ALU	Arithmetic and Logic Unit
PC	Program Counter
RF	Flags Register
CST	Constant Value

## 2. Visão Geral

O LAPI DOpaCA LAMBA é um processador RISC que apresenta um conjunto de 42 instruções organizadas em X grupos funcionais e possui três modos de endereçamento. O processador possui 16 registradores de propósito geral em seu banco de registradores. O LAPI DOpaCA LAMBA suporta operações lógicas e aritméticas básicas, exceto multiplicação e divisão.

### 2.1. Perspectives

The MUSA 32-bit core release targets modern FPGA devices with embedded SPRAM memory, and is intended to be deployed as a GPPU soft IP-core. Its architecture is designed under MIPS original implementation, presenting a slightly reduced ISA.

### 2.2. Características Principais

- Arquitetura de 32 bits;
- 16 registradores de propósito geral;
- Arquitetura de conjunto de instruções (ISA) RISC;
- ISA composta por 42 instruções;
- Organização de dados Big-Endian;
- Três modos de endereçamento: imediato, por registrador, e base-deslocamento;

### 2.3. Non-functional Requirements

- The FPGA prototype should run in a Terasic ALTERA Cyclone III (EP3C25F324) Development platform;
- The design must be described using Verilog-HDL;
- The tests must be written in both Verilog-HDL or SystemVerilog;
- A set of test programs must be provided in order to validate the implementation;

### 3. Conjunto de Instruções

O Processador LAPI DOpaCA LAMBA possui 42 instruções de 32 bits de largura que estão organizadas nos seguintes grupos funcionais:

- Acesso à Memória (load e store);
- Constantes;
- ALU;
- Transferências de controle (jump);
- NOPs;
- Condição de término.

No LAPI DOpaCA LAMBA, as operações são realizadas utilizando operandos localizados no banco de registradores. A memória principal é acessada apenas através das instruções load e store.

#### 3.1. Instruções de Acesso à Memória

Op Código	Mnemônico	Operação
10100	load c, a	$C = \text{Mem}[A]$
10110	store a, b	$\text{Mem}[A] = B$

#### 3.2. Instruções com Constantes

As instruções abaixo carregam constantes para um registrador de propósito geral.

Formato	Mnemônico	Operação
I	$C = \text{Constante}$	loadlit c, Const
II	$C = \text{Const8} \mid (C \& 0\text{xff}00)$	lcl c, Const8
II	$C = (\text{Const8} \ll 8) \mid (C \& 0\text{x}00\text{ff})$	lch c, Const8

R	Operação
0	lcl
1	lch

### 3.3. Instruções ALU

O LAPI DOpaCA LAMBA realiza operações lógicas e aritméticas de 32-bits usando dois ou três operandos. As instruções ALU seguem o seguinte formato:

Código	Mnemônico	Operação	Flags Afetadas
00000	zeros c	$C = 0$	Zero
00001	and c,a,b	$C = A \& B$	neg e zero
00010	andnota c,a,b	$C = (\tilde{A}) \& B$	neg e zero
00011	passa c,a	$C = A$	neg e zero
00100	xor c,a,b	$C = A \text{ xor } B$	neg e zero
00101	or c,a,b	$C = A   B$	neg e zero
00110	nor c,a,b	$C = !(A   B)$	neg e zero
00111	xnor c,a,b	$C = !(A \text{ xor } B)$	neg e zero
01000	passnota c,a	$C = \tilde{A}$	neg e zero
01001	ornotb c,a,b	$C = A   \tilde{B}$	neg e zero
01010	nand c,a,b	$C = !(A \& B)$	neg e zero
01011	ones c	$C = 1$	nenhuma
01100	add c,a,b	$C = A + B$	rc = ra   rb
01101	addinc c,a,b	$C = A + B + 1$	todas
01110	inca c,a	$C = A + 1$	todas
01111	subdec c,a,b	$C = A - B - 1$	todas
10000	sub c,a,b	$C = A - B$	todas
10001	deca c,a	$C = A - 1$	todas
10010	lsl c,a	$C = \text{shift lógico para a esquerda}$	neg, zero e carry
10011	lsr c,a	$C = \text{shift lógico para a direita}$	neg e zero
10100	asr c,a	$C = \text{shift aritmético para a direita}$	neg e zero
10101	asl c,a	$C = \text{shift aritmético para a esquerda}$	NÃO SEI

### 3.4. Transferência de Controle

O LAPI DOpaCA LAMBA possui 5 instruções para transferência de controle. Desvios condicionais com base no estado de uma das flags, desvio para um endereço contido

num registrador, desvio para um endereço referente a um label e uma instrução para chamada de funções.

Formato	Op Código	Mnemônico	Operação
I	00	Jump False	jf.cond Destino
I	01	Jump True	jt.cond Destino
II	10	Jump Incondicional	j Destino
III	11	Jump and Link	jal b
VI	11	Jump Register	jr b

R	Operação
0	Jump and Link
1	Jump Register

Op Código	Mnemônico	Condição
10100	.neg	Resultado ALU negativo
10110	.zero	Resultado ALU zero
10100	.carry	Carry da ALU
10110	.negzero	Resultado ALU negativo ou zero
10100	.true	Resultado ALU diferente de zero
10110	.overflow	Resultado ALU overflow

### 3.5. CPU Load and Store Instructions

Mnemonic	Operands	Realization	Description
LW	$RD, RS1, I_{16}$	$RD \leftarrow MEM[RS1 + I_{16}]$	Load word from data memory.
SW	$RS1, RS2, I_{16}$	$MEM[RS1 + I_{16}] \leftarrow RS2$	Store word into data memory.

### 3.6. Computational Instructions

The MUSA provides 32-bit arithmetic operations. The computational instructions uses ALU two-operand instructions. These are signed versions of the following operations:

- Add



- Subtract
- Multiply
- Divide

Mnemonic	Operands	Realization	Description
ADD	$RD, RS1, RS2$	$RD \leftarrow RS1 + RS2$	Add two words.
SUB	$RD, RS1, RS2$	$RD \leftarrow RS1 - RS2$	Subtract two words.
MUL	$RD, RS1, RS2$	$RD \leftarrow RS1 * RS2$	Multiply two words.
DIV	$RD, RS1, RS2$	$RD \leftarrow RS1 / RS2$	Divide two words.
AND	$RD, RS1, RS2$	$RD \leftarrow RS1 \odot RS2$	Logical AND.
OR	$RD, RS1$	$RD \leftarrow RS1 \oplus RS2$	Logical OR.
ADDI	$RD, RS1, I_{16}$	$RD \leftarrow RS1 + I_{16}$	Add a stored word and an immediate value.
SUBI	$RD, RS1, I_{16}$	$RD \leftarrow RS1 - I_{16}$	Subtract a stored word and an immediate value.
ANDI	$RD, RS1, I_{16}$	$RD \leftarrow RS1 \odot I_{16}$	Logical AND of a stored word and an immediate value
ORI	$RD, RS1, I_{16}$	$RD \leftarrow RS1 \oplus I_{16}$	Logical OR of a stored word and an immediate value.
CMP	$RD, RS1$	–	Compares $RD$ and $RS1$ and set the flags.
NOT	$RD$	$RD \leftarrow \sim RD$	Logical NOT.

### 3.7. Jump/Branch Instructions

Mnemonic	Operands	Characteristic	Description
JR	$R$	Unconditional	Jump to destination.
JPC	$I_{28}$	Unconditional	Jump to destination PC-relative.
BRFL	$RF, CST$	Conditional	Jump to destination if $RF == CST$ .
CALL	$R$	Unconditional	Subroutine call.
RET	–	Unconditional	Subroutine return

### 3.8. No Operation Instruction

The *No Operation* instruction (NOP) is used to control the instruction flow or to insert delays (stalls) into the datapath, such as when computing the result of a jump/branch instruction. When using a NOP instruction after a branch/jump instruction it is so named a **branch delay slot**.

### 3.9. End of Operations

The HALT instruction (system stop) must be implemented as a  $L: j L$  (a unconditional branch to the current address)

## 4. Internal General Purpose Registers

The current MUSA architecture provides 32 fixed-point general purpose registers of 32 bits:  $R_0$  to  $R_{31}$ . The  $R_0$  have special use for the hardware always returning zero, no matter what software attempts to store to it.