Equipe 02

Ana Paula Rodrigues Raimundo - 1862197

João Vitor Santos Anacleto - 1802836

Nilton Miguel Guimaraes de Souza - 2237164

ISA

a Xtensa ISA é uma arquitetura "moderna pós-RISC" focada na implementação de sistemas embarcados de performance e baixo consumo, como a própria empresa declara:

1.2 The Xtensa Instruction Set Architecture

The Xtensa Instruction Set Architecture (ISA) is a new post-RISC ISA targeted at embedded, communication, and consumer products. The ISA is designed to provide:

- A high degree of extensibility
- Industry-leading code density
- Optimized low-power implementation
- High performance
- Low-cost implementation

Obtivemos o documento da ISA no repositório: xtensa Instruction Set Architecture (ISA) Reference Manual.pdf

3.8 Instruction Summary

Table 3–11 summarizes the core instructions included in all versions of the Xtensa architecture. The remainder of this section gives an overview of the core instructions.

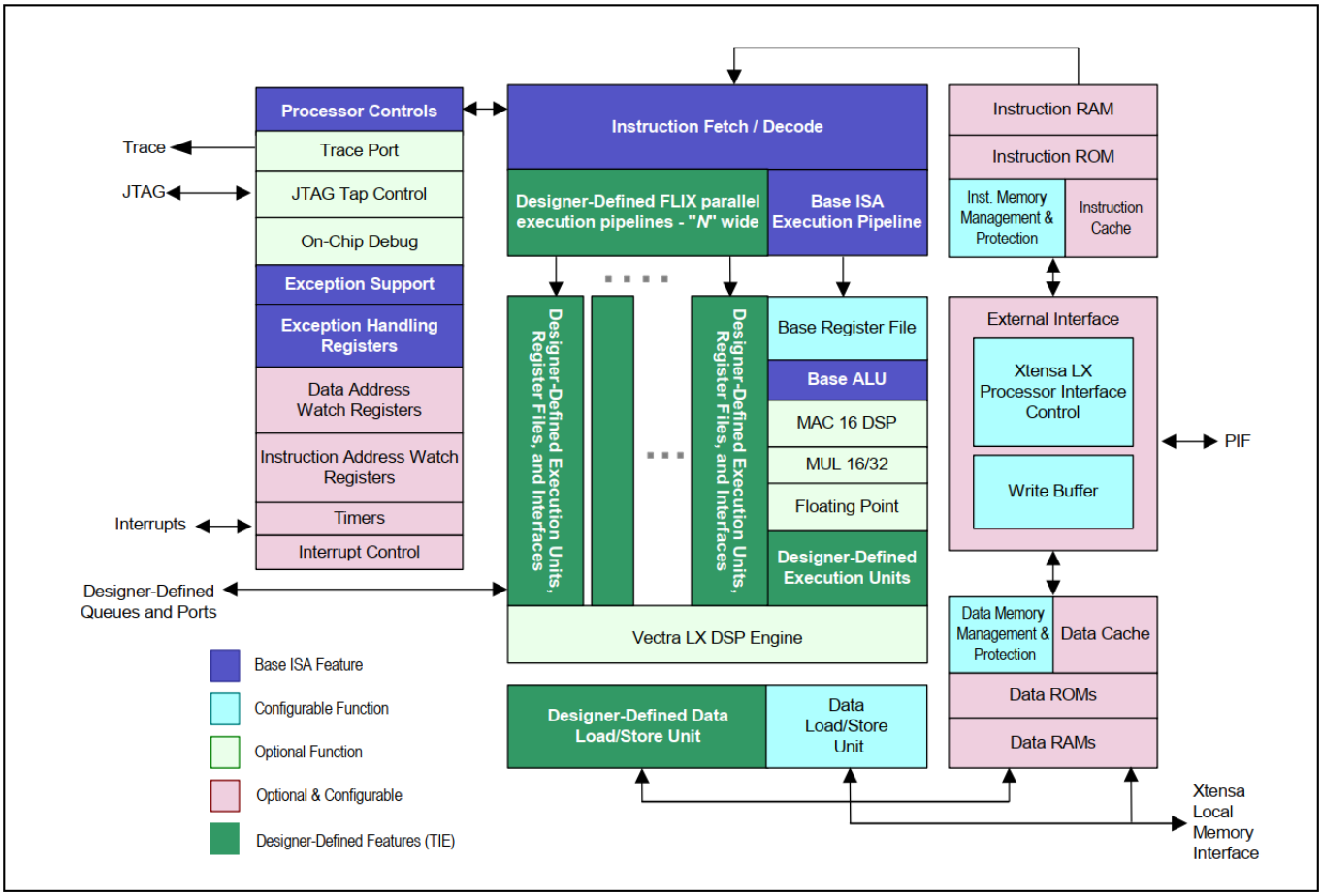
Table 3–11. Core Instruction Summary

Instruction Category	Instructions[1]	Reference
Load	L8UI, L16SI, L16UI, L32I, L32R	"Load Instructions" on page 33
Store	S8I, S16I, S32I	"Store Instructions" on page 36
Memory ordering	MEMW, EXTW	"Memory Access Ordering" on page 39
Jump, Call	CALL0, CALLX0, RET J, JX	"Jump and Call Instructions" on page 40
Conditional branch	BALL, BNALL, BANY, BNONE BBC, BBCI, BBS, BBSI BEQ, BEQI, BEQZ BNE, BNEI, BNEZ BGE, BGEI, BGEU, BGEUI, BGEZ BLT, BLTI, BLTU, BLTUI, BLTZ	"Conditional Branch Instructions" on page 40
Move	MOVI, MOVEQZ, MOVGEZ, MOVLTZ, MOVNEZ	"Move Instructions" on page 42
Arithmetic	ADDI, ADDMI, ADD, ADDX2, ADDX4, ADDX8, SUB, SUBX2, SUBX4, SUBX8, NEG, ABS	"Arithmetic Instructions" on page 43
Bitwise logical	AND, OR, XOR	"Bitwise Logical Instructions" on page 44
Shift	EXTUI, SRLI, SRAI, SLLI SRC, SLL, SRL, SRA SSL, SSR, SSAI, SSA8B, SSA8L	"Shift Instructions" on page 44
Processor control	RSR, WSR, XSR, RUR, WUR, ISYNC, RSYNC, ESYNC, DSYNC, NOP	"Processor Control Instructions" on page 45

1. These instructions are fully described in Chapter 6, "Instruction Descriptions" on page 243.

Na versão mais nuclear da ISA estão poucas instruções, sendo a maioria variações de tamanho dos dados transferidos ou qual o mecanismo de offset e shift.

O diagrama de blocos do Hardware que implemente essa ISA. Percebe-se na legenda da imagem o quanto dessas funções são opcionais, ou configuráveis/ inclusas a critério do fabricante do hardware para uma tarefa específica. A Arquitetura é muito adaptável.



Consumo

Como o foco da ISA é em sistemas embarcados (alimentados a bateria) ela limita a maioria as operações a 32 bit e deixa a possibilidade de extensão a dados maiores para implementações que realmente precisarem.

Além de outras escolhas de design para baixo consumo há a possibilidade de desativar a lógica do processador enquanto se espera por uma interrupção com uma instrução privilegiada.

1.2.6 Low-Power

The Xtensa ISA has several energy-efficient attributes that enhance battery-operated systems. The core ISA is built on 32-bit operations; some embedded processors of similar performance have 64-bit base operations, which consumes additional power, often unnecessarily. (TIE does allow 64-bit or greater computations to be added to the processor for those algorithms that require it, but these can be used selectively to achieve a balance between performance and power consumption.)

The core ISA uses a register file with only two read ports and one write port, a configuration that requires fewer transistors and less power than architectures with more ports.

The Xtensa Windowed Registers Option saves power by reducing the number of dynamic data-memory references and increasing the opportunities for variables to reside in registers, where accesses require less power than memory accesses.

The `WAITI` (Wait for Interrupt) instruction, which is a part of the Interrupt Option, saves power by setting the current interrupt level, powering down the processor's logic, and waiting for an interrupt.

Description

`WAITI` sets the interrupt level in `PS.INTLEVEL` to `imm4` and then, on some Xtensa ISA implementations, suspends processor operation until an interrupt occurs. `WAITI` is typically used in an idle loop to reduce power consumption. `CCOUNT` continues to increment during suspended operation, and a `CCOMPARE` interrupt will wake the processor.

When an interrupt is taken during suspended operation, `EPC[i]` will have the address of the instruction following `WAITI`. An implementation is not required to enter suspended operation and may leave suspended operation and continue execution at the following instruction at any time. Usually, therefore, the `WAITI` instruction should be within a loop.

Pipelines

Sim, a arquitetura permite pipelines com grau de flexibilidade. A quantidade de estágios varia com a implementação, mas a ISA nos garante suporte a load-store em 5 estágios e até 7 estágios a depender da aplicação.

1.2.8 Pipelines

The Xtensa ISA can be implemented using a variety of pipelines. A 5-stage load-store oriented pipeline, such as is used in many RISC processors, is supported by Xtensa implementations and illustrated in Figure 1–2. Many other variations are possible. A 7-stage load-store oriented pipeline is supported by some Xtensa implementations. Instructions can also have computation in later pipe stages so that the computation can use memory data loaded by the same instruction.