Especificação de Requisitos

Processador Lapidopacalamba

Universidade Estadual de Feira de Santana

Build 1

Histórico de Revisões

Date	Descrição	Autor(s)
21/12/2015	Concepção e estruturação do Documento	Patricia Gomes
21/12/2015	Finalização do Documento	Patricia Gomes
21/12/2015	Revisão do Documento	Fábio Barros
20/01/2016	Correções no Documento	Patricia Gomes

SUMÁRIO

1. Introdução

1.1. Propósito do Documento

O projeto consiste no desenvolvimento um processador capaz de executar 42 instruções. O processador desenvolvido possui 16 registradores de propósito geral, sendo que cada registrador possui a capacidade de armazenamento de 32 bits.

Este documento descreve a lógica de implementação de todos os requisitos do processador.

1.2. Visão Geral do Documento

- **Requisitos funcionais** - lista de todos os requisitos funcionais.
- **Requisitos não funcionais** - lista de todos os requisitos não funcionais.

1.3. Definições

Termo	Descrição
Requisitos Funcionais	Requisitos de hardware que compõem os módulos, descrevendo as ações que o mesmo deve estar apto a executar.
Requisitos Não Funcionais	Requisitos de hardware que compõem os módulos, representando as características que o mesmo deve ter, ou restrições que o mesmo deve operar. Estas características referem-se técnicas, algoritmos, tecnologias e especificidades do Sistema como um todo.

1.4. Acrônimos e Abreviações

Sigla	Descrição
FR	Requisito Funcional
NFR	Requisito Não Funcional
ULA	Unidade de Lógica e Aritmética

2. Requisitos Funcionais

2.1. Requisitos dos módulos

[FR1] Criação de uma Unidade de Lógica e Aritmética

Descrição: Para que o processador atenda ao requisito de ser capaz de executar 42 instruções, é necessário construir uma unidade responsável por executar as operações de lógica e aritmética(ULA). A ULA deve ser capaz de realizar as operações com dois operandos de 32 bits. A mesma deve possuir duas saídas, uma para apresentar o resultado que também será de 32 bits, e outra para apresentar as flags atualizadas de acordo com a operação realizada. Além disso, a ULA deve possuir um sinal de controle responsável por indicar a mesma o código da operação, ou seja, qual operação ela deverá executar no momento.

Nível de Prioridade: **Importante**

[FR2] Criação de um Extensor de sinais

Descrição: Para viabilizar as operações com constantes e de saltos deve ser implementado um extensor de sinais. O mesmo deve ser capaz de extender 12 ou 16 bits para 32 bits conforme o sinal de controle enviado para o extensor. Para realizar a extensão dos sinais, o extensor deve replicar o bit de sinal.

Nível de Prioridade: **Importante**

[FR3] Criação de um registrador de flags

Descrição: Deve ser criado um registrador de flags de forma a armazenar as flags atualizadas de acordo com a operação da ULA. A ULA possui uma saída de flags e para essa saída devem ser enviadas as flags atualizadas na mesma. Um sinal de controle é enviado para permitir que a saída da ULA com as flags atualizadas sejam armazenadas no registrador de flags.

Nível de Prioridade: **Importante**

[FR4] Atualizar Flags

Descrição: A medida que uma operação é realizada algumas flags deverão ser atualizadas. A atualização dessas flags se dará de acordo com a operação realizada pela ULA. As Flags a serem atualizadas são: Overflow, Sinal, Carry e Zero. A flag de sinal é atualizada se o resultado da operação for negativo. Se o resultado for igual a 0, deve-se atualizar a flag zero. No caso da flag de overflow, a mesma deve ser atualizada se o sinal do resultado for diferente do sinal dos operandos.

Nível de Prioridade: **Importante**

[FR5] Criação de um testador de flags

Descrição: Para ser possível a realização de desvios condicionais deve ser implementado um módulo testador de flags que terá como função analisar uma condição e decidir se um jump condicional será ou não realizado. As condições a serem analisadas são apresentadas no documento de arquitetura. Se a condição for satisfeita e for enviando um sinal de jump false o salto não deverá ser tomado, logo para que um salto seja executado, a condição deve ser verdadeira e o sinal de jump deve ser true, ou a condição deve ser falsa e o sinal de jump deve ser false.

Nível de Prioridade: **Importante**

[FR6] Criação de uma memória de dados

Descrição: Deverá ser desenvolvida uma memória responsável por salvar/ler dados do banco de registradores de acordo com as instruções de acesso à memória.

Nível de Prioridade: **Importante**

[FR7] Criação de um banco de registradores

Descrição: O banco de registradores deve possuir 16 registradores com capacidade de armazenamento de 32 bits. O banco deve ter como entrada os endereços dos registradores fonte, o endereço do registrador destino, e uma entrada para o dado a ser armazenado. E deve possuir como saída os dados contidos nos registradores fonte.

Nível de Prioridade: **Importante**

[FR8] Criação de uma memória de instrução

Descrição: Todas as instruções devem ser armazenadas na memória uma memória específica para armazenar as instruções que deverão ser realizadas pelo processador.

Nível de Prioridade: **Importante**

2.2. Requisitos das operações

[FR9] Operação de adição

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores e soma os dois operandos bit a bit. A operação de soma deve atualizar todas as flags.

Nível de Prioridade: **Importante**

[FR10] Operação de adição com incremento

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores executa a soma dos dois operandos bit a bit e em seguida soma ao número 1. Esta operação deve atualizar todas as flags.

Nível de Prioridade: **Importante**

[FR11] Operação de incremento

Descrição: A Unidade Lógica e Aritmética recebe um operando inicialmente armazenado no banco de registradores e soma ao número 1. Esta operação deve atualizar todas as flags.

Nível de Prioridade: **Importante**

[FR12] Operação de subtração com decremento

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores e executa a subtração fazendo uma soma bit a bit com o segundo operando em complemento a 2 e em seguida subtrai o número 1. Essa operação deve atualizar todas as flags.

Nível de Prioridade: **Importante**

[FR13] Operação de subtração

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores e executa a subtração fazendo uma soma bit a bit com o segundo operando em complemento a 2. Esta operação deve atualizar todas as flags.

Nível de Prioridade: **Importante**

[FR14] Operação de decremento

Descrição: A Unidade Lógica e Aritmética recebe um operando inicialmente armazenado no banco de registradores e subtrai o número 1. Esta operação deve atualizar todas as flags.

Nível de Prioridade: **Importante**

[FR15] Operação de deslocamento lógico

Descrição: A Unidade Lógica e Aritmética recebe um operando inicialmente armazenado no banco de registradores e desloca seus bits à esquerda. Esta operação deve atualizar as flags de sinal, carry e zero.

Nível de Prioridade: **Importante**

[FR16] Operação de deslocamento aritmético

Descrição: A Unidade Lógica e Aritmética recebe um operando inicialmente armazenado no banco de registradores e desloca seus bits à direita. Esta operação deve atualizar as flags de sinal, carry e zero.

Nível de Prioridade: **Importante**

[FR17] Operação zeros

Descrição: A Unidade Lógica e Aritmética envia zero para saída. Esta operação deve atualizar apenas a flag de zero.

Nível de Prioridade: **Importante**

[FR18] Operação and

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores e executa uma and entre os dois operandos. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR19] Operação and com o primeiro operando negado

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores e faz o complemento a 2 do primeiro operando, em seguida executa uma and entre os dois operandos. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR20] Operação que passa o operando B

Descrição: A Unidade Lógica e Aritmética recebe um operando inicialmente armazenado no banco de registradores e o envia para saída. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR21] Operação and com o segundo operando negado

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores e faz o complemento a 2 do segundo operando, em seguida executa uma and entre os dois operandos. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR22] Operação que passa o operando A

Descrição: A Unidade Lógica e Aritmética recebe um operando inicialmente armazenado no banco de registradores e o envia para saída. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR23] Operação xor

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores e executa uma xor entre os dois operandos. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR24] Operação or

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores e executa uma or entre os dois operandos. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR25] Operação nand

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores, transforma os dois em negativos usando o complemento a 2 e sem seguida executa uma and entre os operandos. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR26] Operação xnor

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores e executa uma xor entre os dois operandos, em seguida transforma o valor obtido em negativo fazendo o complemento a 2. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR27] Operação que passa o operando A negativo

Descrição: A Unidade Lógica e Aritmética recebe um operando inicialmente armazenado no banco de registradores o transforma em negativo fazendo o complemento a 2 e o envia para saída. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR28] Operação or com o primeiro operando negado

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores, transforma o primeiro operando em negativo usando o complemento a 2 e em seguida executa uma or entre os operandos. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR29] Operação que passa o operando B negativo

Descrição: A Unidade Lógica e Aritmética recebe um operando inicialmente armazenado no banco de registradores o transforma em negativo fazendo o complemento a 2 e o envia para saída. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR30] Operação or com o segundo operando negado

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores, transforma o segundo operando em negativo usando o complemento a 2 e sem seguida executa uma or entre os operandos. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR31] Operação nor

Descrição: A Unidade Lógica e Aritmética recebe dois operandos inicialmente armazenados no banco de registradores, transforma os dois em negativos usando o complemento a 2 e sem seguida executa uma or entre os operandos. Esta operação deve atualizar as flags de sinal e zero.

Nível de Prioridade: **Importante**

[FR32] Operação ones

Descrição: A Unidade Lógica e Aritmética envia 1 para saída. Esta operação não deve atualizar nenhuma flag.

Nível de Prioridade: **Importante**

[FR33] Operação loadlit

Descrição: A Unidade Lógica e Aritmética recebe o valor de uma connstante contido em um registrador do banco de registradores e o envia para saída.

Nível de Prioridade: **Importante**

[FR34] Operação lcl

Descrição: Inicialmente é realizada uma operação and entre uma constante de 16 bits e o valor contido no endereço 0xffff0000 da memória, em seguida é feita uma or entre o resultado dessa and e a constante de 16 bits.

Nível de Prioridade: **Importante**

[FR35] Operação lch

Descrição: Inicialmente a constante é deslocada em 16 bits e é realizada uma operação and entre a constante e o valor contido no endereço 0x0000ffff da memória, em seguida é feita uma or entre o resultado dessa and e a constante deslocada.

Nível de Prioridade: **Importante**

[FR36] Operação load

Descrição: o endereço presente no registrador B do banco de registradores deve ser lido da memória, e a saída é escrita no banco de registradores no endereço especificado na pelo registrador A.

Nível de Prioridade: **Importante**

[FR37] Operação store

Descrição: Na instrução Store os dados devem ser lidos do banco de registradores e escritos na memória, sendo registrador A o dado a ser escrito e o registrador B endereço onde será armazenado.

Nível de Prioridade: **Importante**

[FR38] Operação de desvio incondicional

Descrição: Deve realizar um salto para um endereço de instrução definido na memória de instruções, de forma a permitir a execução da instrução para onde ocorreu o salto. No desvio incondicional o valor de PC deve apontar para a instrução seguinte, esse valor da memória de instruções tem o sinal extendido e o resultado vai para a ula e da ula vai um multiplexador cuja seleção dependerá do módulo responsável por testar as flags. No jump incondicional é enviando um sinal que habilita o jump no testador de flags junto com

a condição TRUE da tabela de condições (consta no documento de arquitetura). O testador de Flags envia o sinal de seleção para o mux e o mux manda para sua saída o valor que havia saído da ula e esse valor volta pra PC que indicará qual instrução deverá ser realizada.

Nível de Prioridade: **Importante**

[FR39] Operação de desvios condicionais

Descrição: Deve realizar um salto para um endereço de instrução definido na memória de instruções, de forma a permitir a execução da instrução para onde ocorreu o salto. No desvio incondicional o valor de PC deve apontar para a instrução seguinte, esse valor da memória de instruções tem o sinal extendido e o resultado vai para a ula e da ula vai um multiplexador cuja seleção dependerá do módulo responsável por testar as flags. Se a condição de flags for verdadeira e o jump for true, ou se a condição for falsa e o jump for false, o multiplaxador manda para sua saída o valor que havia saído da ula e esse valor volta pra PC que insicará qual instrução deverá ser realizada. .

Nível de Prioridade: **Importante**

[FR40] Operação de desvios tipo and link

Descrição: O desvio do tipo jump and link é um desvio incondicional onde o valor de PC+1 deve ser armazenado no registrador r15 e o conteúdo do registrador RB armazenado em PC. Esse tipo de jump permite que o valor que PC tinha antes do desvio seja recuperado, logo, após o salto, PC volta a apontar para a mesma instrução que apontava antes do salto.

Nível de Prioridade: **Importante**

[FR41] Operação de jump register

Descrição: Deve armazenar o conteúdo do registrador RB no PC.

Nível de Prioridade: **Importante**

[FR42] Halt

Descrição: Deve realizar um salto incondicional para o endereço atual.

Nível de Prioridade: **Importante**

[FR43] Nop

Descrição: Nessa instrução todos os sinais de controle devem ser zerados, de forma que nada seja registrado na memória ou no banco de registradores.

Nível de Prioridade: **Importante**

3. Requisitos não Funcionais

[NFR1] Armazenamento em memória de forma big-endian

Descrição: O armazenamento na memória deve ser feito de forma big-endian, logo, o bit mais significativo do dado deve ser armazenado na posição menos significativa da memória.

Nível de Prioridade: **Importante**

[NFR2] Unidade de controle hardwired

Descrição: A unidade de controle deve ser hardwire, logo, sinais de controle devemm ser gerados com o uso de técnicas de circuitos lógicos convencionais.

Nível de Prioridade: **Importante**