



Centro de Informática

Universidade Federal da Paraíba - UFPB

Engenharia da computação

Implementação da Unidade de Controle do RISC V multiciclo

Disciplina: Concepção Estruturada de Circuitos Integrados

Professor: Antônio Carlos Cavalcanti

Aluno: Johan Kevin Estevão de Freitas

Matrícula: 20170171741

2020

Resumo

Neste relatório será detalhada a arquitetura do Jogo de Instruções RISC V com os diferentes tipos de instrução e as correspondentes ocupações dos bits nas palavras de instrução. Será descrito também o funcionamento da máquina de estados desenvolvida para a microarquitetura Multiciclo MIPS e implementado o código e resultados do modelo de ouro da máquina de estados abstrata: de S0 a S13 descrevendo a unidade de controle multiciclo adaptada ao RISC V detalhando a "main decoder" e a ULA Control. O código e os resultados do modelo de ouro da ULA Control, do Main Decoder e da unidade de controle multiciclo completa Adaptada ao RISC V estarão também apresentados neste relatório.

Introdução

RISC-V é um conjunto de instruções (ISA) baseado e estabelecido nos princípios RISC (acrônimo de Reduced Instruction Set Computing, em português, “Computação de conjunto de instruções reduzida”).



Figura 01 - Exemplo de Placa de desenvolvimento que integra RISC-V

RISC-V é livre para ser usado para qualquer finalidade, permitindo a qualquer pessoa ou empresa projetar e vender chips e software RISC-V. Embora não seja o primeiro conjunto de instruções livre, ele é importante porque foi projetado com foco para dispositivos computadorizados modernos, como computação em nuvem, aparelhos móveis, sistemas embarcados e internet das coisas. O conjunto também possui uma gama considerável de software de suporte, o que evita um problema usual de novos conjuntos de instruções. O RISC-V foi projetado para implementações de alto desempenho e baixo consumo de energia. Sendo um conjunto limpo e modular, trabalhando com bases de 32, 64 e 128 bits, com várias opções de extensão em ponto flutuante.

Caminho de dados

A arquitetura RISC-V é uma arquitetura load-store, ou seja, a leitura e a escrita de operandos armazenados na memória são feita através de duas instruções dedicadas, lw e sw, respectivamente. Neste tipo de arquitetura, as unidades funcionais não acessam diretamente a memória, mas registradores internos que armazenam temporariamente os seus operandos. Desta forma, o caminho de dados da CPU proposta é composto pelas memórias de instrução e de dados, por uma unidade lógica e aritmética (ALU) de inteiros e outra equivalente para números complexos (CALU), e por um banco de registradores de 32 bits, responsável pelo armazenamento interno (temporário) dos operandos (variáveis) e endereços de memória (ponteiros).

Conjunto de instruções RISC-V

O ISA RISC-V é definido com um conjunto de instruções básicas de inteiros, que deve ser presente em todas as implementações, e uma série de extensões opcionais ao conjunto de instruções (WATERMAN et al., 2014). O conjunto base provê as instruções necessárias para o suporte de ferramentas de desenvolvimento de software e de sistemas operacionais – quando em conjunto com instruções privilegiadas do modo supervisor. Além deste, os modos de máquina, com controle total do processador, e de usuário, limitado pelo sistema operacional, também são suportados pela especificação

Apesar de definir a especificação do conjunto de instruções básico e das ferramentas de software de referência, o ISA RISC-V não define a estrutura de processador, estando a definição da organização a cargo do projetista. Esta característica possibilita o desenvolvimento independente de uma família de processadores RISC-V projetados sob medida às necessidades de desempenho, custo, área e consumo energético das variadas aplicações, todas elas tendo em comum uma base de software compatível, incluindo sistemas operacionais de como Linux/Android.

O conjunto de instruções padrão da arquitetura (WATERMAN et al., 2014) define quatro formatos de instruções básicos : R, I, S, U, mostrados na *Figura 02*.

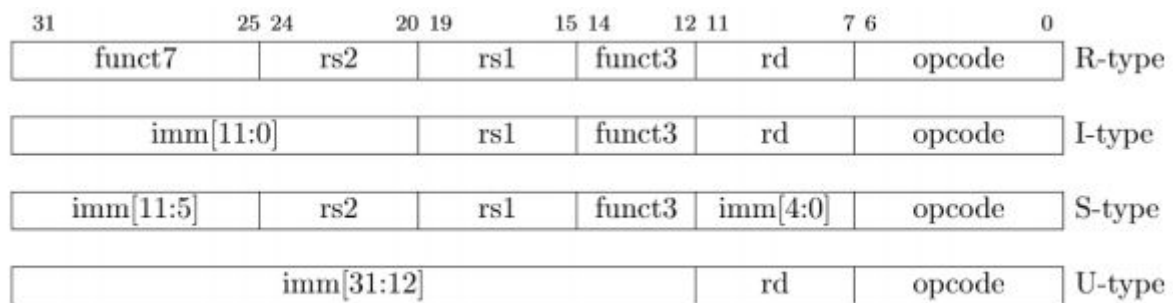


Figura 02 - Formatos das Instruções RISC-V (WATERMAN et al., 2014)

Por ser uma arquitetura load-store, todas as instruções operam sobre valores armazenados nos registradores. Na figura 5.2, os campos rd, rs1 e rs2 selecionam os registradores de destino (rd) , e de origem (rs1, rs2).













Unidade Lógica e Aritmética

A unidade central do caminho de dados é a unidade lógica e aritmética (ALU). A ALU é o conjunto de circuitos de computação responsável pela execução das instruções aritméticas (adição, subtração, comparação, multiplicação e divisão), de lógica booleana (OR, XOR, AND) e manipulação de bits (deslocamentos), além de realizar os cálculos dos endereços nas 57 instruções de acesso à memória (LW, LB, LH, SW, SB, SH) e da instrução de desvio relativo à registrador JALR. A ALU executa as instruções RISC-V, sendo todos os seus operandos vindos do banco de registradores ou de valores imediatos, constantes passadas pelo compilador (programador) por meio de instruções específicas. A escolha da operação a ser executada sobre os operandos A e B é feita através da linha de controle op, de 5 bits de largura. As seguintes operações são executadas pela ALU: Adição, Subtração, Deslocamento à esquerda, Deslocamento à direita (lógico e aritmético), OR, AND, XOR, Comparação, “menor que”, Multiplicação Divisão (Quociente e resto).

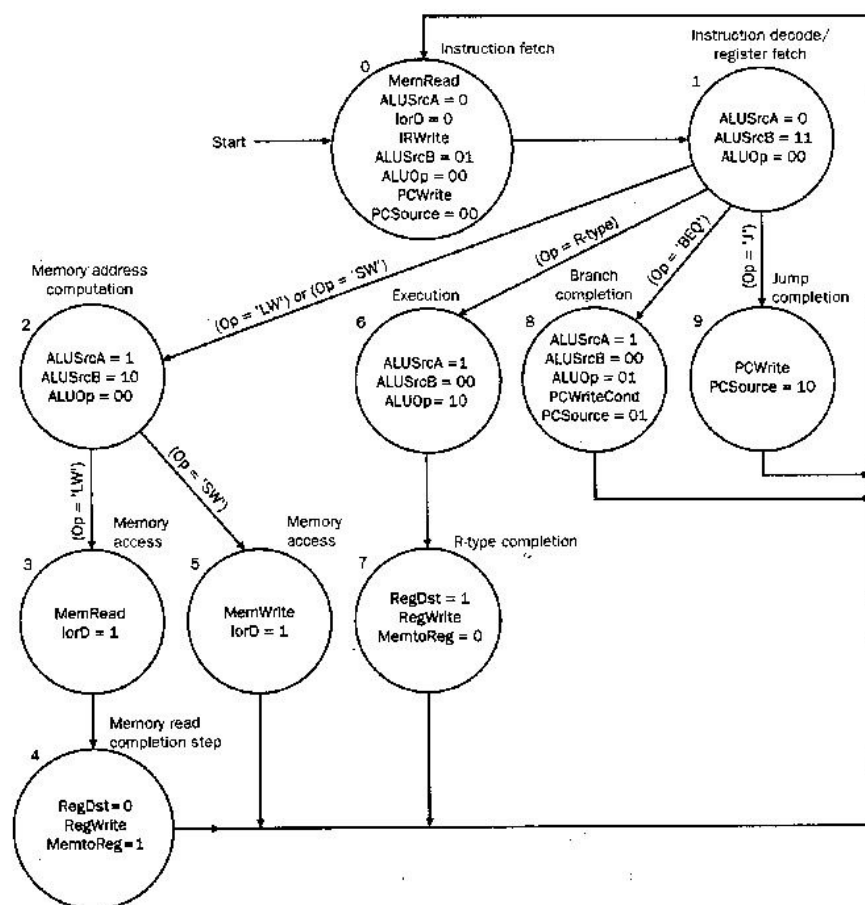
Implementação:

Prezando a organização do relatório, anexados junto a este relatório, estão os arquivos de implementação contendo o código em linguagem C da ULA, o

arquivo de entrada, o arquivo .tv gerado na saída e fotos ordenadas do código para uma melhor visualização.

| | | | | |
|---|---------|------------------|-----------------|-------|
|  | 01 ULA | 14/08/2020 00:12 | Arquivo JPG | 45 KB |
|  | 02 ULA | 14/08/2020 00:13 | Arquivo JPG | 40 KB |
|  | 03 ULA | 14/08/2020 00:13 | Arquivo JPG | 72 KB |
|  | 04 ULA | 14/08/2020 00:14 | Arquivo JPG | 70 KB |
|  | 05 ULA | 14/08/2020 00:14 | Arquivo JPG | 83 KB |
|  | 06 ULA | 14/08/2020 00:15 | Arquivo JPG | 79 KB |
|  | 07 ULA | 14/08/2020 00:15 | Arquivo JPG | 78 KB |
|  | 08 ULA | 14/08/2020 00:16 | Arquivo JPG | 81 KB |
|  | 09 ULA | 14/08/2020 00:16 | Arquivo JPG | 44 KB |
|  | entrada | 14/08/2020 15:07 | Arquivo TV | 1 KB |
|  | saida | 14/08/2020 15:07 | Arquivo TV | 4 KB |
|  | ULA | 14/08/2020 00:16 | C++ source file | 10 KB |

Máquina de Estados



A máquina de estados finita é um modelo matemático usado para representar programas de computadores ou circuitos lógicos. O conceito é concebido como uma máquina abstrata que deve estar em um de um número finito de estados. Podemos adiantar que tomando por definição o Main Controller tem como entrada o Opcode da instrução é implementado como uma máquina de estados finitos (FSM - Finite State Machine em inglês), onde cada estado é um ciclo de alguma instrução. Apenas ele fornece a maioria dos sinais de controle que vai para o caminho de dados, com exceção apenas do ALUControl que dita a operação da unidade lógica aritmética (ULA ou ALU - Arithmetic Logic Unit em inglês).

Começaremos a implementação da máquina de estados apenas com os estados e suas transições, sem se preocupar com os sinais de controle. A implementação segue a máquina descrita na figura acima. adicionamos as constantes para representar os estados S0..S12, constantes que armazenam o opcode das instruções, e também constantes que armazenam os functs das operações do tipo R.

Para gerar os vetores de teste, foi criada uma função que recebe um arquivo de entrada (entrada.tv) com o sinal de reset, e todos os opcodes que englobam as operações suportadas por essa implementação. Essa função gera um arquivo com os vetores de teste, em que a cada transição do clock é gravado os sinais de clock, reset, o opcode de entrada e o estado esperado naquele momento.

Prezando pela organização do relatório, anexados junto a este relatório, estão os arquivos de implementação contendo o código em linguagem C da MEF, o arquivo de entrada, o arquivo .tv gerado na saída e fotos ordenadas do código para uma melhor visualização.

| | | | |
|---|------------------|-----------------|-------|
|  MEF (1) | 13/08/2020 23:55 | Arquivo JPG | 64 KB |
|  MEF (2) | 13/08/2020 23:44 | Arquivo JPG | 71 KB |
|  MEF (3) | 13/08/2020 23:44 | Arquivo JPG | 79 KB |
|  MEF (4) | 13/08/2020 23:45 | Arquivo JPG | 71 KB |
|  MEF (5) | 13/08/2020 23:45 | Arquivo JPG | 64 KB |
|  MEF (6) | 13/08/2020 23:46 | Arquivo JPG | 68 KB |
|  MEF (7) | 13/08/2020 23:47 | Arquivo JPG | 83 KB |
|  MEF (8) | 13/08/2020 23:47 | Arquivo JPG | 29 KB |
|  entrada | 14/08/2020 15:00 | Arquivo TV | 1 KB |
|  saida | 14/08/2020 15:01 | Arquivo TV | 6 KB |
|  MEF | 13/08/2020 23:56 | C++ source file | 11 KB |

Considerações Finais

A abordagem teórica e implementação prática da arquitetura RISC é de grande aprendizado pois Desenvolver uma nova arquitetura de CPU requer um esforço conjunto de experts de diversas áreas, o que torna a criação de uma arquitetura aberta viável extremamente complicada. O sucesso do RISC-V foi possível graças ao trabalho de diversos especialistas e voluntários, o que de acordo com seus contribuidores o torna um projeto provindo do esforço comunitário e poder implementar e abordar mesmo que basicamente essa tecnologia agrega um conhecimento indispensável no sucesso da carreira de qualquer engenheiro da computação.

Referências bibliográficas

- Material Fornecido nas aulas
- [https://www.embarcados.com.br/fe310g-microcontrolador-open-source-estrutura-basica-risc-v/#:~:text=RISC%2DV%20%C3%A9%20uma%20ISA,e%20o%20MIPS16e\)%20al%C3%A9m%20de](https://www.embarcados.com.br/fe310g-microcontrolador-open-source-estrutura-basica-risc-v/#:~:text=RISC%2DV%20%C3%A9%20uma%20ISA,e%20o%20MIPS16e)%20al%C3%A9m%20de)
- <https://www.cin.ufpe.br/~if674/arquivos/2018.2/Aulas/arquitetura-mips-risc-v-cap2.pdf>
- <https://pt.wikipedia.org/wiki/RISC-V>
- <https://pt.slideshare.net/embarcados/webinar-uma-introducao-a-isa-riscv-e-seu-ecossistema>

