Relatório do Trabalho de ISA e Microarquitetura

1. Especificação da ISA

1.1 Introdução

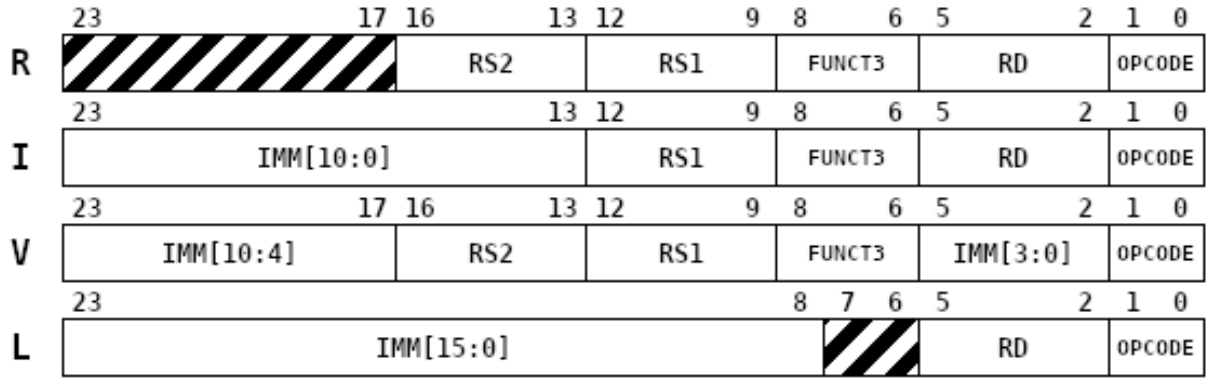
O conjunto de instruções desenvolvido neste trabalho foi fortemente inspirado nas ISAs RV32I e RV32M da arquitetura RISC-V, adaptadas para instruções de 24 bits. Por causa da limitação de tamanho de 24 bits para a instrução, apenas as operações mais essenciais foram incluídas. Ao todo, o conjunto de instruções contém 23 operações.

O objetivo foi manter a arquitetura o mais simples possível, sem comprometer a praticidade de uso. O conjunto de instruções definido cobre operações aritméticas, lógicas, controle de fluxo e acesso à memória, proporcionando suporte completo às estruturas básicas de programação e chamadas de função.

1.2 Formatos de instrução

A ISA definida possui quatro formatos de instrução, identificados unicamente pelo `opcode`:

- **Formato R:** Utilizado para operações lógicas e aritméticas entre registradores. Contém dois registradores fonte (`rs1` e `rs2`) e um registrador destino (`rd`).
- **Formato I:** Usado para operações que envolvem um registrador fonte (`rs1`), um valor imediato (`imm`) e um registrador destino (`rd`). Além de operações lógicas e aritméticas, é usado para operações de salto e acesso à memória.
- **Formato V:** Usado para operações que envolvem dois registradores fonte (`rs1, rs2`) e um valor imediato (`imm`), como instruções de *branch* e *store*.
- **Formato L:** Exclusivo para a operação `li (load immediate)`, usada para carregar um valor imediato de 16 bits (`imm`) em algum registrador (`rd`).



Todos os formatos incluem 2 bits de `opcode`, suficientes para determinar o tipo da instrução. Além disso, os formatos R, I e V possuem 3 bits adicionais (`funct3`), permitindo 8 instruções únicas para cada formato. O formato L não utiliza `funct3`, liberando espaço para um imediato maior.

Os formatos R e L possuem bits excedentes que poderiam ser usados para expandir o conjunto de instruções, mas optou-se por mantê-los inutilizados para simplificar a implementação.

1.3 Registradores

O banco de registradores inclui 16 registradores de propósito geral, cada um com 32 bits, organizados conforme a seguinte convenção de uso:

- `r0 (zero)`: constante zero, imutável
- `r1 (rad)`: endereço de retorno

- `r2 (rbp)`: ponteiro para a base da pilha (base pointer)
- `r3 (rsp)`: ponteiro para o topo da pilha (stack pointer)
- `r4..r7 (rt0..rt3])`: registradores temporários
- `r8..r11 (rs0..rs3)`: registradores salvos
- `r12 (ra0)`: valor de retorno / primeiro argumento de função
- `r13..r15 (ra1..ra3)`: argumentos de função

Há também um registrador especial de 8 bits reservado para o controle do fluxo de execução: o registrador `ip` (`instruction pointer`). Ele é incrementado automaticamente após a execução de cada instrução, exceto em casos de desvios e saltos, onde o valor é alterado diretamente.

1.4 Conjunto de instruções

1.4.1 Instruções lógicas e aritméticas

Formato R:
- `add`: soma.
- `sub`: subtração.
- `mul`: multiplicação.
- `div`: divisão inteira.
- `mod`: resto da divisão inteira.
- `and`, `or`, `xor`: operações lógicas.

Formato I:
- `addi`: soma com valor imediato.
- `modi`: resto da divisão inteira com valor imediato.
- `slli`: *shift* lógico para a esquerda com valor imediato.
- `and`, `or`, `xor`: operações lógicas com valor imediato.

1.4.2 Instruções de acesso à memória

Formato I:
- `lw`: Carrega uma palavra de 32 bits da memória para o registrador.

Formato V:
- `sw`: Armazena uma palavra de 32 bits na memória a partir de um registrador.

As operações de acesso à memória utilizam a notação '`op rx, imm(ry)`', onde `ry` é o registrador que contém o endereço base e `imm` é um valor de deslocamento.

1.4.3 Instruções de salto / desvio de fluxo

Formato I:
- `jalr`: Salto incondicional para o endereço em `rs1` deslocado de `imm`; `rd` recebe o endereço de retorno.

Formato V:
- `beq`, `bne`: Desvio condicional para igualdade e diferença. Se a condição for verdadeira, salta para `ip` + `imm`.
- `bge`, `blt`: Desvio condicional para maior ou igual e menor.
- `halt`: para a execução do programa, impedindo `ip` de ser incrementado.

É importante notar que as instruções de *branch* fazem um desvio relativo à posição atual no programa, enquanto as instruções de *jump* fazem um desvio absoluto, sem considerar a posição atual no programa.

1.4.4 Instruções de carregamento

Formato L:

- `li`: Carrega um imediato de 16 bits (`imm`) na parte baixa do registrador `rd`.