Montador RISC-V

Grupo 1 - Gabriel Rodrigues Marques (5097), Alan Araújo dos Reis (5096)

Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa - Campus Florestal (UFV-CAF)
Florestal – MG – Brasil

{gabriel.r.marques, alan.a.reis}@ufv.br

Resumo. *Esta documentação apresenta o desenvolvimento e implementação em de um montador simplificado para a arquitetura RISC-V, capaz de converter instruções em linguagem de máquina.*

Repositório: *https://github.com/gabridulol/RISC-V-Assembler*

1. Introdução

O trabalho consiste no desenvolvimento de uma versão simplificada de um montador para a arquitetura RISC-V. O montador converte uma série de instruções na linguagem RISC-V Assembly para a linguagem de máquina em binário correspondente. A linguagem de programação Python foi escolhida para implementação. As instruções seguem o padrão descrito no livro *Computer Organization and Design RISC-V Edition: The Hardware Software Interface* e no documento *The RISC-V Instruction Set Manual - Volume I: User-Level ISA*.

2. Desenvolvimento

O montador implementado por cada grupo deveria suportar um subconjunto de 7 instruções formado à partir de: lb, lh, lw, sb, sh, sw, add, sub, and, or, xor, addi, andi, ori, sll, srl, bne, beq.

`lb`	`sb`	`add`	`and`	`ori`	`sll`	`bne`

Figura 1. Instruções do Grupo 1

Com o objetivo de tornar o montador mais completo, todas as outras instruções foram implementadas. Além disso, o montador possui suporte para números na base hexadecimal. O conjunto de pseudo-instruções não foi totalmente implementado, suas partes do código estão comentadas.

Cada instrução pode ser classificada de acordo com o seu tipo. Para cada tipo de instrução existe um formato específico. O tamanho é de 32 bits por instrução, variando a disposição dos campos que a compõem para cada tipo. A figura abaixo (Figura 2) foi utilizada como referência para realizar a conversão para linguagem de máquina no montador.

Name (Field size)	Field						Comments
	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits	
R-type	funct7	rs2	rs1	funct3	rd	opcode	Arithmetic instruction format
I-type	immediate[11:0]		rs1	funct3	rd	opcode	Loads & immediate arithmetic
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode	Stores
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode	Conditional branch format
UJ-type	immediate[20,10:1,11,19:12]				rd	opcode	Unconditional jump format
U-type	immediate[31:12]				rd	opcode	Upper immediate format

Figura 2. RISC-V Formato das Instruções

Format	Instruction	Opcode	Funct3	Funct6/7
R-type	add	0110011	000	0000000
	sub	0110011	000	0100000
	sll	0110011	001	0000000
	xor	0110011	100	0000000
	srl	0110011	101	0000000
	sra	0110011	101	0000000
	or	0110011	110	0000000
	and	0110011	111	0000000
	lr.d	0110011	011	0001000
	sc.d	0110011	011	0001100
I-type	lb	0000011	000	n.a.
	lh	0000011	001	n.a.
	lw	0000011	010	n.a.
	ld	0000011	011	n.a.
	lbu	0000011	100	n.a.
	lhu	0000011	101	n.a.
	lwu	0000011	110	n.a.
	addi	0010011	000	n.a.
	slli	0010011	001	000000
	xori	0010011	100	n.a.
	srli	0010011	101	000000
	srai	0010011	101	010000
	ori	0010011	110	n.a.
	andi	0010011	111	n.a.
	jalr	1100111	000	n.a.
S-type	sb	0100011	000	n.a.
	sh	0100011	001	n.a.
	sw	0100011	010	n.a.
	sd	0100011	111	n.a.
SB-type	beq	1100111	000	n.a.
	bne	1100111	001	n.a.
	blt	1100111	100	n.a.
	bge	1100111	101	n.a.
	bltu	1100111	110	n.a.
	bgeu	1100111	111	n.a.
U-type	lui	0110111	n.a.	n.a.
UJ-type	jal	1101111	n.a.	n.a.

Figura 3. RISC-V Codificação de Instruções

A figura acima (Figura 3) apresenta a classificação e codificação para cada tipo e instrução em RISC-V, apresentando os valores de opcode, funct3 e funct7 que foram utilizados no montador para realizar a conversão.

O montador foi feito utilizando diversas funções para cada campo das instruções, utilizando a estrutura de dicionários. Em Python, um dicionário é uma estrutura de dados que permite armazenar pares de chave-valor. Cada chave é única dentro do dicionário e é associada a um valor correspondente. Isso permite acessar os valores de maneira eficiente usando as chaves.

A seguir estão descritas brevemente as funções principais no código:

instruction: a função recebe uma instrução e retorna o tipo da instrução correspondente.

opcode: a função recebe um tipo de instrução e retorna o opcode correspondente.

funct3: a função recebe uma instrução e retorna a funct3 correspondente.

funct7: a função recebe uma instrução e retorna a funct7 correspondente.

immediate: a função recebe valor decimal ou hexadecimal e converte para binário.

register: a função recebe um registrador e retorna seu valor binário correspondente.

writer: a função lê as instruções de um arquivo de entrada e escreve as instruções montadas em binário em um arquivo de saída.

assemblerWriter: realiza a montagem da instrução de acordo com o seu tipo e a escreve no arquivo de saída.

reader: a função lê as instruções de um arquivo de entrada e exibe as instruções montadas em binário no terminal.

assemblerReader: realiza a montagem da instrução de acordo com o seu tipo e a retorna.

formater: a função recebe uma linha de instrução e remove quaisquer caracteres desnecessários, deixando em um formato processável.

Mais detalhes de implementação podem ser vistos no código fonte do projeto, disponível no repositório.

3. Resultados

O montador pode ser executado de duas maneiras: exibindo os resultados no terminal ou em um arquivo de saída.

(TERMINAL)

python3 src/main.py <filename>.asm

(ARQUIVO DE SAÍDA)

python3 src/main.py <filename>.asm -o <outputFilename>

Os resultados podem ser encontrados nas figuras abaixo (Figura 4, 5, 6).

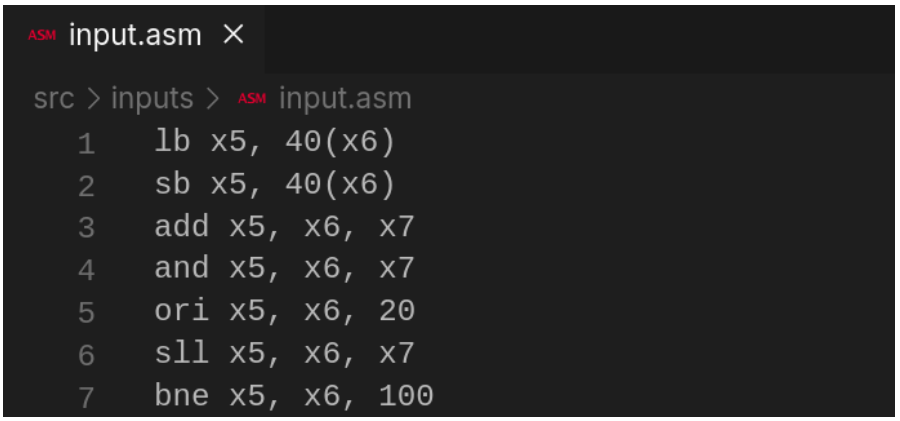

Figura 4. Arquivo de Entrada

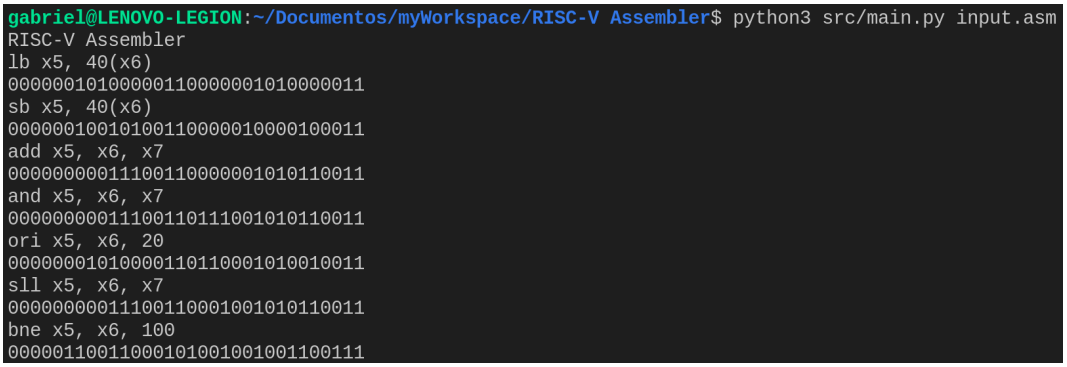

Figura 5. Terminal

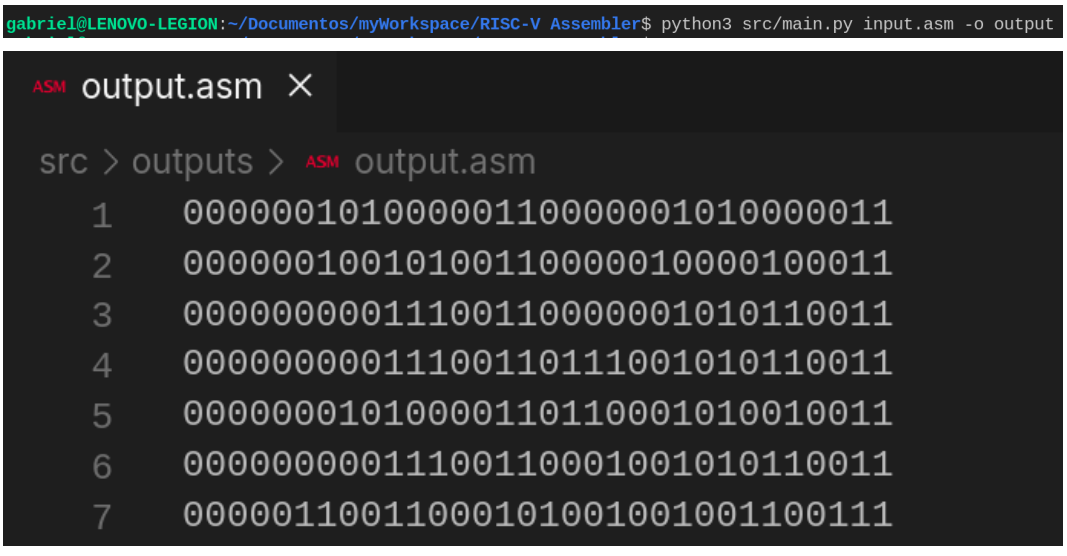

Figura 6. Arquivo de Saída

4. Conclusão

O desenvolvimento do montador simplificado para a arquitetura RISC-V permitiu uma compreensão mais profunda das características do conjunto de instruções RISC-V e das nuances de sua implementação. O montador demonstrou ser capaz de converter instruções em linguagem de máquina com precisão e eficiência, suportando um conjunto selecionado de instruções essenciais. A escolha da linguagem Python para a implementação proporcionou uma abordagem flexível e facilitou a manipulação dos diferentes tipos de instruções. Apesar de algumas limitações, como a ausência completa de suporte para pseudo-instruções, o montador provou ser uma ferramenta valiosa para a conversão de instruções na linguagem RISC-V Assembly, e seu código-fonte está disponível para futuras melhorias e ampliações.

Referências

David A. Patterson and John L. Hennessy. Computer Organization and Design RISC-V Edition: The Hardware Software Interface.

Andrew Waterman and Krste Asanović. The RISC-V Instruction Set Manual - Volume I: User-Level ISA.