

Montador RISC-V

Versão simplificada

Vitor R. Lacerda, João V. B. Andrade, Mateus H. V. Figueiredo

¹Universidade Federal de Viçosa, campus Florestal (UFV)
Minas Gerais – MG – Brasil

{vitor.lacerda, joao.andrade1, mateus.h.figueiredo}@ufv.br

Abstract. *This document refers to the practical work I, of the subject Computer Organization I, code CCF-252. It was suggested the implementation of a simplified version of a RISC-V assembler; able to read an ASM input file and write a binary output file, compatible with the chosen simulator.*

Resumo. *Este documento refere-se ao trabalho prático I, da matéria de Organização de Computadores I, código CCF-252. Foi sugerida a implementação de uma versão simplificada de um montador RISC-V, capaz de ler um arquivo de entrada do tipo ASM e escrever um arquivo de saída binário, compatível com o simulador escolhido.*

1. Introdução

Este documento se refere ao trabalho prático I, da matéria de Organização de Computadores I, código CCF-252. No trabalho, foi sugerida a implementação de uma versão simplificada de um montador RISC-V, capaz de ler um arquivo de entrada do tipo ASM e escrever um arquivo de saída binário, compatível com o simulador escolhido.

Para o desenvolvimento do trabalho, foram usadas bibliotecas externas, tais como "os" e "sys". A biblioteca "os", é uma biblioteca de comandos do sistema operacional que auxilia em algumas operações dentro do computador. Já a biblioteca "sys" (System-specific parameters and functions) permite acesso à variáveis usadas ou mantidas pelo interpretador e a funções que interagem fortemente com o interpretador.

O programa foi dividido em três pastas, garantindo uma maior organização para os requisitos do trabalho sugerido.

2. Utilizando Python

Para o trabalho foi escolhida a linguagem de programação python, essa escolha foi feita pela simplicidade da linguagem e sua capacidade de resolver problemas complexos.

2.1. Instalando Python

O processo a seguir é utilizado para instalar o Python, que foi usado para a construção do montador, em sua versão mais recente (3.10.4), no sistema operacional Ubuntu. Para instalar Python 3 no Linux verifique se o Python já está instalado com o comando "python3 --version" ou "python --version" (Figura 1).

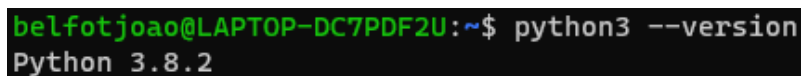
Nota: Se sua distribuição do Linux acompanha Python, poderá ser necessário instalar o pacote de desenvolvedor Python. O pacote de desenvolvedor inclui os cabeçalhos e

as bibliotecas que são necessárias para compilar extensões e instalar o AWS ParallelCluster. Use o gerenciador de pacotes para instalar o pacote de desenvolvedor. Geralmente é chamado de python-dev ou python-devel.

Se Python 2.7 ou posterior não estiver instalado, instale Python com o gerenciador de pacote de distribuição. O comando e o nome do pacote varia de: nos derivados do Debian, como Ubuntu, use apt "sudo apt-get install python3" no Red Hat e derivados, use yum "sudo yum install python3" no SUSE e derivados, use o zypper "sudo zypper install python3"(Figura 2).

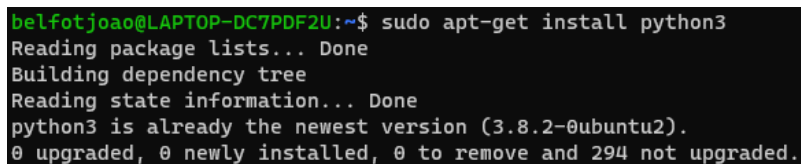
Para verificar se o Python foi instalado corretamente, abra um prompt de comando ou shell e execute o comando a seguir "python3 --version"(Figura 3).

As imagens a seguir exemplificam os comandos aqui citados:



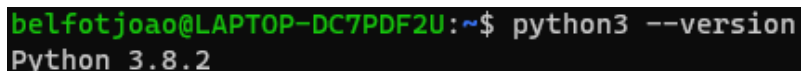
```
beltofjoao@LAPTOP-DC7PDF2U:~$ python3 --version
Python 3.8.2
```

Figura 1. verificando se o Python já está instalado



```
beltofjoao@LAPTOP-DC7PDF2U:~$ sudo apt-get install python3
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.8.2-0ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 294 not upgraded.
```

Figura 2. instalando o python no ubuntu



```
beltofjoao@LAPTOP-DC7PDF2U:~$ python3 --version
Python 3.8.2
```

Figura 3. verificando versão do python no ubuntu

2.2. Rodando o Código em Python

O primeiro passo abrir o terminal do Linux, na mesma pasta onde se encontra o arquivo main.py, para isso é necessário utilizar os comandos "pwd", para ver qual é o diretório atual que estamos navegando, ls para listar os arquivos do diretório atual, cd para navegar entre os diretórios.

Após chegar no diretório é necessário abrir o interpretador do python pois a linguagem é interpretada, sendo necessário colocar "python3", como primeiro argumento, o interpretador exemplo: "python3 src/main.py input.asm", para escrever o resultado no terminal e "python3 src/main.py input_2.asm -o output_2", para retornar o resultado em um arquivo bin .

Nota: A seguir estão alguns exemplos de uso destes comandos, como não nenhum integrante do grupo tem o sistema ubuntu somente o WSL os diretórios não são correlacionados com o trabalho.

```

belfotjoao@LAPTOP-DC7PDF2U:/$ cd home
belfotjoao@LAPTOP-DC7PDF2U:/home$ ls
belfotjoao
belfotjoao@LAPTOP-DC7PDF2U:/home$ pwd
/home

```

Figura 4. exemplo de navegação por diretórios

```

belfotjoao@LAPTOP-DC7PDF2U:/home$ python3
Python 3.8.2 (default, Mar 13 2020, 10:14:16)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.

```

Figura 5. abrindo o interpretador python3

```

>>> exec(open('main.py').read())

```

Figura 6. executando programa main.py

3. Desenvolvimento

No tabalho, foram feitas subdivisões em arquivos para maior organização das funções. Os arquivos que contém funções são os seguintes:

3.1. "converter.py"

Neste arquivo, são estão contidas as funções: "convert_instruction_to_binary", "get_register_binary_number", "check_invalid_values", "get_immediate_binary_12bits", "get_immediate_binary_20bits", "overflow_if_true", "overflow_or_underflow_if_true", "convert_immediate_to_decimal".

- "convert_instruction_to_binary": Função utilizada para, caso as instruções inseridas sejam validas, converter os valores das instruções para binário.

- "get_register_binary_number": Função utilizada para converter um registrador para seu correspondente binário de 5 bits.

- "check_invalid_values": Função utilizada para checar os parâmetros da instrução.

- "get_immediate_binary_12bits": Função que converte um imediato para o seu correspondente binário em uma base de 12 bits.

- "get_immediate_binary_20bits": Função que converte um imediato para o seu correspondente binário em uma base de 20 bits.

- "overflow_if_true": Função que checa se os valores estão "overflowed", se verdadeiro, a função executa a operação de overflow.

- "overflow_or_underflow_if_true": Função que checa se os valores estão "overflowed" ou "underflowed", se verdadeiro, a função executa a operação de overflow ou underflow.

- "convert_immediate_to_decimal": Função que converte um imediato para o seu correspondente binário em uma base de 10 bits

3.2. file.py

Neste arquivo, são estão contidas as funções: "read_file_and_generate_output", "read_file_and_print", "pre_check_line".

- "read_file_and_generate_output": Função utilizada para ler dados de um arquivo e escreve-los em um arquivo de saída (output).

- "read_file_and_print": Função utilizada para ler dados de um arquivo e escreve-los no console.

- "pre_check_line": Função utilizada para checar se a linha é um comentário e/ou se a linha está vazia.

4. Arquivos de entrada e saída

4.1. input_files:

Os arquivo de input utilizados pelo código nos testes, estão presentes na sub-pasta "input_files". Os arquivos podem ser produzidos pelo usuário, estes arquivo gerados devem se encontrar na sub-pasta previamente informada e possuir o fomato de um código em RISC-V. O nome e caminho destes devem ser informado por meio do terminal. Exemplos de código:

```
add x2, x0, x1
sll x1, x2, x2
or x2, x2, x1
andi x2, x1, 16
addi x3, x2, -243
```

Figura 7. arquivo de input exemplo 1

```
add x2, x0, x1
add x6, x0, x2
sub x3, x6, x2
xor x4, x2, x3
sr1 x0, x2, x2
```

Figura 8. arquivo de input exemplo 2

4.2. output_files

Os arquivos de saída gerados nos testes do código se encontram na sub-pasta "output_files". Como o sugerido pelo trabalho prático, os arquivos de saída apresentam o código traduzido de RISC-V para o formato de linguagem de máquina em binário. Exemplos de arquivos de saída:

```

00000000001000000000100110011
00000000001000010001000010110011
0000000000100010110000100110011
00000001000000001111000100010011
11110000110100010000000110010011

```

Figura 9. arquivo de output para o input exemplo 1

```

00000000001000000000100110011
0000000000100000000001100110011
01000000001000110000000110110011
00000000001100010100001000110011
0000000000100001010100000110011

```

Figura 10. arquivo de output para o input exemplo 2

4.3. Output do terminal

Abaixo, podemos encontrar um exemplo da saída de um terminal após a execução do código.

```

belfotjoao@LAPTOP-DC7PDF2U:~$ ls
go tp01-4675-4694-4707-main
belfotjoao@LAPTOP-DC7PDF2U:~$ cd tp01-4675-4694-4707-main/
belfotjoao@LAPTOP-DC7PDF2U:~/tp01-4675-4694-4707-main$ ls
README.md command-instruction-types.txt input_files output_files src
belfotjoao@LAPTOP-DC7PDF2U:~/tp01-4675-4694-4707-main$ python3 src/main.py input_2.asm -o output_2
Output file generated at: /home/belfotjoao/tp01-4675-4694-4707-main/output_files/output_2.bin.
belfotjoao@LAPTOP-DC7PDF2U:~/tp01-4675-4694-4707-main$ python3 src/main.py input.asm
Translated instructions from 'input.asm':
00000000001000000000100110011
00000000001000010001000010110011
0000000000100010110000100110011
00000001000000001111000100010011
11110000110100010000000110010011

```

Figura 11. arquivo de output para o arquivo de input "input.asm"

5. Conclusão

De forma geral, foram implementados códigos, de forma organizada, comentada e de fácil entendimento para o problema solicitado pelo trabalho, foi decidida uma abordagem mais interativa com o usuário, que ao fim foi uma ótima opção abstraindo o código, e o deixando mais fácil de ser visualizado e compreendido pelo usuário. Os resultados saíram assim como o esperado, sendo que algumas implementações feitas, apesar de deixarem o código mais extenso, facilitam seu uso e entendimento, sendo assim o trabalho, de forma geral atende as exigências passadas pelo Professor do curso de OC-2022/1.

6. Referências

Como instalar o Python 3 no OS Ubuntu [1]

Dividindo strings [2]

[1] PhoenixNAP, How to Install Python 3 on Ubuntu 18.04 or 20.04, 12 de Dezembro 2019, Disponível em: <https://phoenixnap.com/kb/how-to-install-python-3-ubuntu>, Ultimo acesso em: 04 de junho de 2022.

[2] StackOverflow, Split Strings into words with multiple word boundary delimiters, 29 de Junho de 2009, Disponível em: <https://stackoverflow.com/questions/1059559/split-strings-into-words-with-multiple-word-boundary-delimiters>, ultimo acesso em 03 de Junho de 2022.