Universidade Federal de Viçosa

Trabalho Prático 02 – Montador RISC-V
CCF 252 – Organização de Computadores I
Ciência da Computação – Campus UFV-Florestal
Prof. José Augusto Miranda Nacif

Montador RISC-V (versão simplificada)

O trabalho consiste na implementação de uma versão simplificada de um montador RISC-V de acordo com as seguintes especificações:

1. Arquivo de entrada: Deve seguir o padrão de codificação definido no livro texto e possuir a extensão asm.
2. Arquivo de saída: Deve ser compatível com um simulador previamente escolhido.

O simulador deve suportar as seguintes instruções:

`ADD, SUB, AND, OR, XOR, ADDI, ANDI, ORI, SLL, SRL.`

O montador pode ser implementado em qualquer linguagem de programação e em qualquer paradigma, e será testado numa máquina com Ubuntu 20.

Pontuação extra pode ser dada para:

- Implementação de pseudo instruções;
- Suporte a outras bases numéricas no arquivo .asm;
- Implementação de instruções de load/store ou desvios (devidamente com o cálculo/tradução dos rótulos em constantes binárias para as instruções).

O trabalho prático pode ser feito em grupos de **ATÉ** 3 alunos.

Cópias de trabalhos práticos serão exemplarmente punidas. A punição será a mesma para quem copiou e para quem forneceu o trabalho prático.

Forma de entrega: Github para o código, Classroom para a documentação em PDF e para o vídeo em formato MP4.

O que deve ser entregue:

- Documentação simplificada do trabalho prático em PDF:
    - Modelo SBC (o mesmo do trabalho anterior);
    - Caso o usuário use uma linguagem diferente de C ou C++, a documentação deve conter instruções com os comandos sobre como instalar o compilador ou o interpretador necessário no Ubuntu.
    - A documentação deve conter instruções sobre como executar o montador, contendo capturas de tela dos resultados.
    - É muito interessante explicar partes importantes do código e mostrá-las na documentação. Evite transliterar o código, mas sim explicá-los em linguagem natural.

- Arquivos fontes do trabalho:
    - Contendo uma makefile que execute a compilação do programa, caso seja necessário (para a geração do binário de linguagens compiladas como C, C++) ou para a geração de bytecodes de linguagens interpretadas que precisam disso (java ou golang);
    - O montador deve ter duas formas de exibir o resultado:
        - No terminal: caso o cliente passe por linha de comando somente o arquivo .asm, o programa deverá exibir na tela do terminal todas as instruções em binário, uma instrução por linha;
        - No arquivo: caso o cliente passe por linha de comando uma entrada convencional de compiladores (./binario_programa entrada.asm -o saida), o programa deverá salvar o código binário, uma instrução por linha, no arquivo depois do parâmetro -o (de output, saída). Caso a linguagem escolhida seja interpretada, não há problemas em ter a necessidade de colocar, como primeiro argumento, o interpretador (Exemplo: python3 meuMontador entrada.asm -o saida). desde que devidamente documentado.

- Vídeo com a apresentação do trabalho em MP4, com o grupo narrando o que foi feito, os diferenciais, e mostrando o trabalho sendo executado com os arquivos de entrada anexados.

Exemplo de entrada:

entrada.asm

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

Exemplo de saída:

arquivo ***saída*** ou escrito no terminal

```
00000000000100000000000100110011
00000000001000010001000010110011
00000000000100010110000100110011
00000001000000000111000100010011
11110000110100010000000110010011
```