Atividade de Laboratório 6a

Objetivos

O objetivo desta atividade é exercitar o uso de instruções para manipulação de vetores e matrizes utilizando o conjunto de instruções da arquitetura RISC-V.

Descrição

Neste laboratório, você deve fazer um programa em linguagem de montagem do RISC-V que leia uma imagem em formato pgm e a exiba na tela utilizando chamadas de sistema para o periférico canvas.

Entrada

Seu programa deve ler um arquivo chamado “imagem.pgm” que estará no mesmo diretório do executável. Você deve utilizar a chamada de sistema open para abrir o arquivo (veja um exemplo no final do enunciado).

Você pode obter exemplos de imagens pgm neste endereço:
PGMB Files - Binary Portable Gray Map Graphics Files

Observações

- Para testes no simulador, você deve carregar seu programa (com extensão .s) e o arquivo de imagem (renomeie-o para “imagem.pgm” antes de selecionar) **simultaneamente**.
    - Note que ao carregar novos arquivos no simulador, os anteriores são apagados de sua memória (portanto, você tem que selecionar o código **E** a imagem **juntos** toda vez).
- Para utilizar o Canvas, você deve habilitá-lo no simulador. Para fazer isso, vá na aba “Hardware” -> tabela “External Devices” -> Ícone de “+” na linha Canvas. Uma nova aba aparecerá para você visualizar o canvas.
- Nesse laboratório utilizaremos vários números de syscall. Esses valores serão sempre gravados no registrador a7, e a função ecall tem um comportamento diferente para cada valor. Para consultar o syscall de uma funcionalidade específica, confira a tabela no simulador (note que as syscalls relacionadas a external devices, como o Canvas, não aparecem nessa tabela se não forem habilitados).
- Você não receberá imagens maiores do que 512x512 (que tipicamente tomam 262159 bytes).
- Em todas as imagens, *Maxval* será 255.
- O canvas é indexado começando em 0.

Saída

Seu programa deve exibir a imagem na tela utilizando chamadas de sistema para o periférico canvas. Você deve ajustar o canvas para ter o tamanho da imagem. As chamadas disponíveis são:

Syscall	Input	Descrição
setPixel	a0: coordenada x do pixel a1: coordenada y do pixel a2: Cor do pixel concatenadas: R|G|B|A ● A2[31..24]: Red ● A2[23..16]: Green ● A2[15..8]: Blue ● A2[7..0]: Alpha a7: 2200 (número da syscall)	Define a cor de um certo pixel do canvas. Para escala cinza, utilize o mesmo valor para as cores (R = B = G) e alpha = 255.
setCanvasSize	a0: largura do canvas (valor entre 0 e 512) a1: altura do canvas (valor entre 0 e 512) a7: 2201 (número da syscall)	Reseta e define o tamanho do canvas.
setScaling	a0: Scaling horizontal a1: Scaling vertical a7: 2202 (número da syscall)	Atualiza o scaling do canvas

Exemplos de chamada de syscall

Exemplo de *setPixel*:

```
    li a0, 100 # coordenada x = 100
    li a1, 200 # coordenada y = 200
    li a2, 0xFFFFFFFF # pixel branco
    li a7, 2200 # syscall setGSPixel (2200)
    ecall
```

Exemplo de *open*:

A chamada open retorna o file descriptor (fd) do arquivo em a0.

```
    la a0, input_file    # endereço do caminho para o arquivo
    li a1, 0             # flags (0: rdonly, 1: wronly, 2: rdwr)
    li a2, 0             # modo
    li a7, 1024          # syscall open
    ecall

input_file: .asciz "imagem.pgm"
```

Exemplo de *read*:

```
    li a0, 0 # file descriptor = 0 (stdin)
    la a1, input_adress #  buffer
    li a2, 1 # size (lendo apenas 1 byte, mas tamanho é variável)
    li a7, 63 # syscall read (63)
    ecall

input_adress: .skip 0x10  # buffer
```

Exemplo de *write*:

```
    li a0, 1            # file descriptor = 1 (stdout)
    la a1, string       # buffer
    li a2, 19           # size
    li a7, 64           # syscall write (64)
    ecall

string:  .asciz "Hello! It works!!!\n"
```