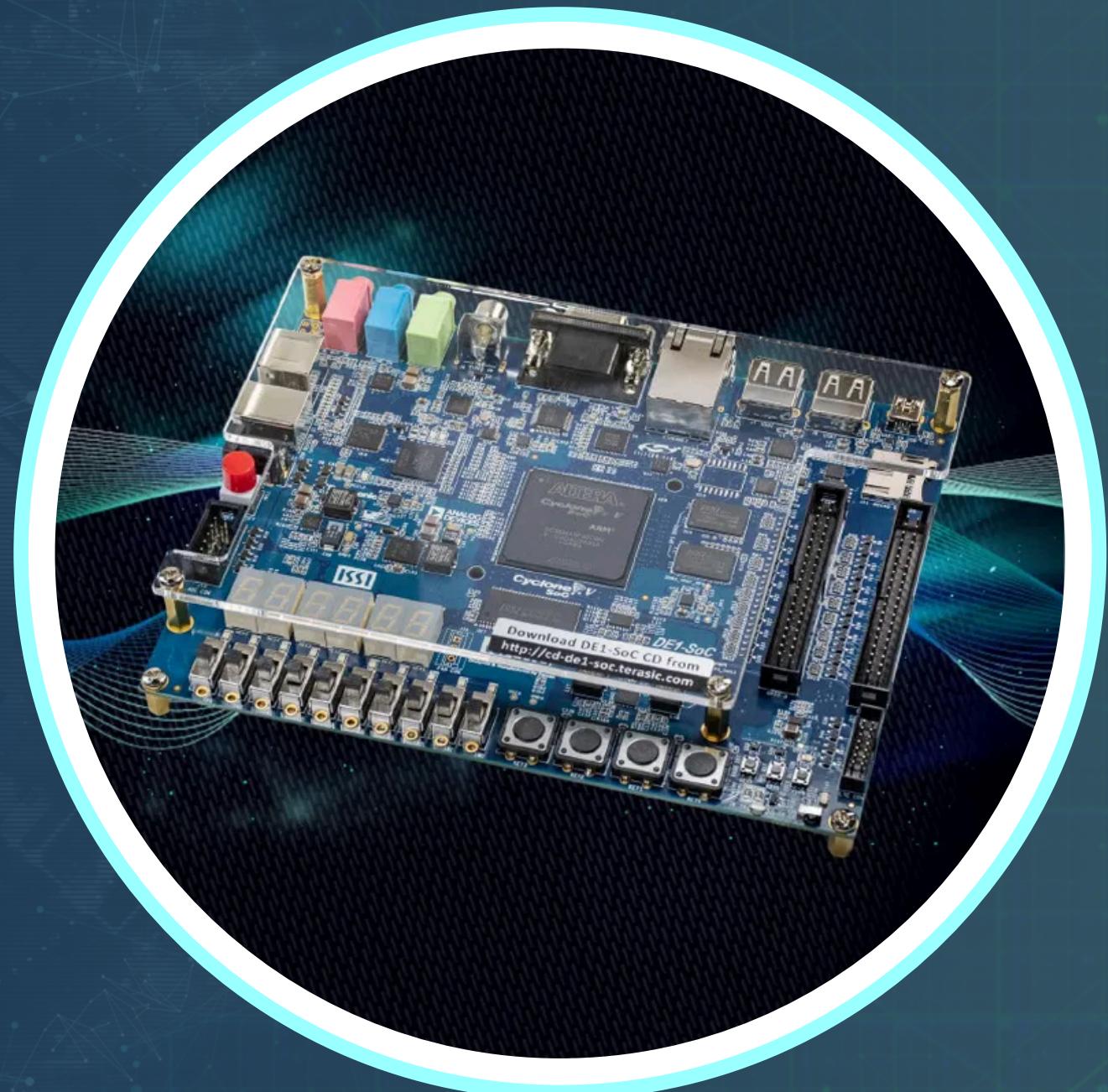




PROGRAMAÇÃO ASSEMBLY E CONSTRUÇÃO DE DRIVER DE SOFTWARE. ETAPA 2

Componentes:
Felipe Bastos
Nathan de Jesus
Jonatas de Jesus



INTRODUÇÃO



Recapitulação da Etapa 1

- Controle dos algoritmos de zoom direto no hardware (chaves e botões)

Problema da Etapa 2

- Criar uma interface Hardware/Software para controle via HPS (Processador ARM)

Objetivos

- Desenvolver uma API (driver) em Assembly ARM
- Permitir que uma aplicação em C controle o hardware da FPGA
- Definir um conjunto de instruções (ISA) para a comunicação



FUNDAMENTAÇÃO TEÓRICA



1. HPS (Hard Process System)

- Unidade de processamento presente em um SoC,
- CPU, controladores e periféricos integrados.
- Comunicação com a FPGA por meio de pontes específicas.

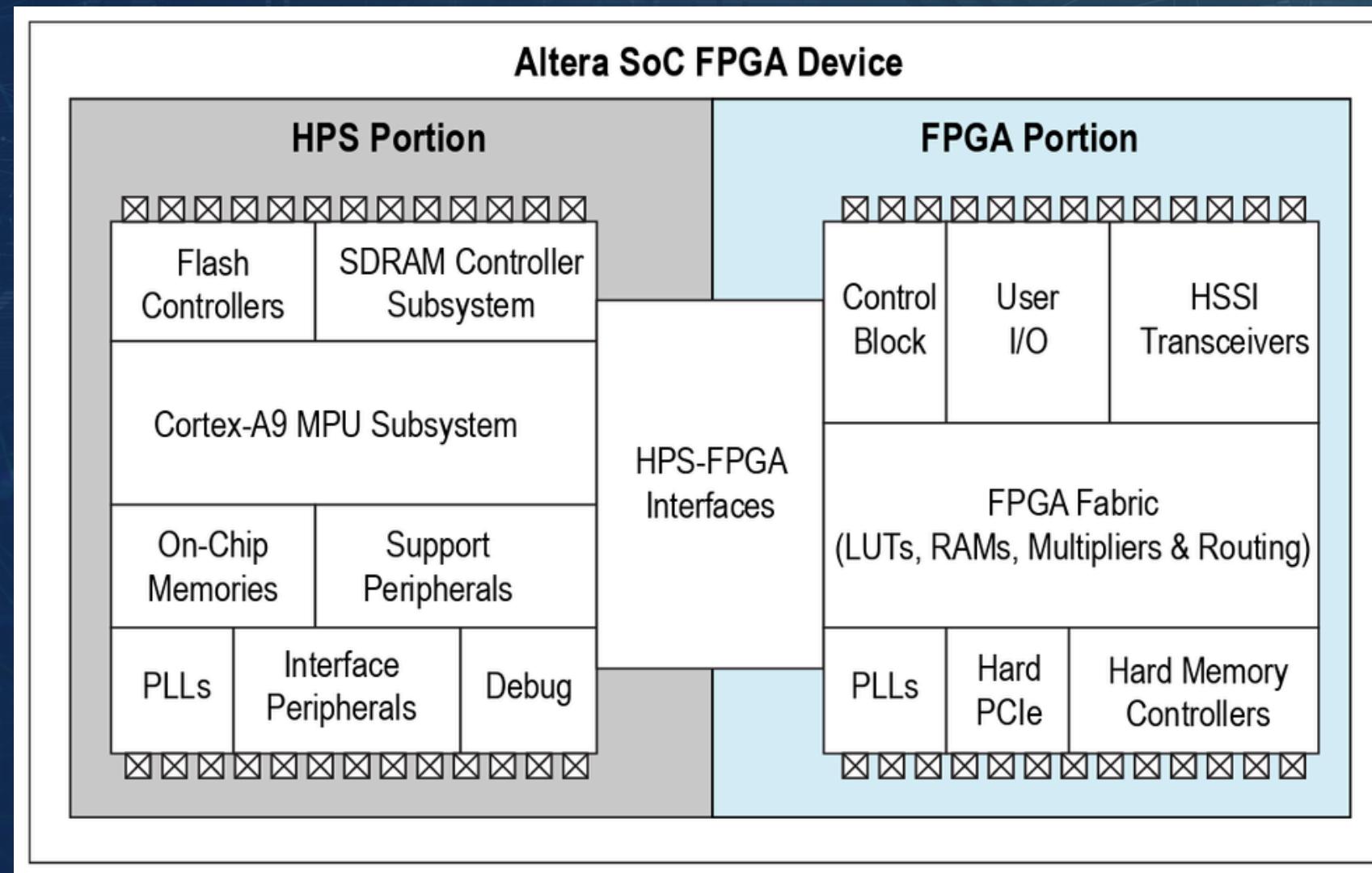


Figura 01 - Exemplo de Diagrama de blocos de dispositivos SoC

FUNDAMENTAÇÃO TEÓRICA ➤➤➤➤➤

2. Lightweight HPS-to-FPGA:

- Interface de comunicação específica dentro de um SoC.
- Otimizada para tráfego de controle de baixa latência

3. AXI (Advanced eXtensible Interface):

- Protocolo de barramento de alto desempenho padrão da arquitetura ARM.
- Alta frequência e baixa latência.
- Protocolo padrão de interconexão

4. Avalon (Avalon-MM):

- Protocolo de interface mapeado em memória
- Conectar periféricos e blocos lógicos dentro da FPGA
- Comunicação por meio de leitura e escrita em endereços.

FUNDAMENTAÇÃO TEÓRICA

5. Barramento

- Sistema de comunicação que transfere dados entre componentes.
- Tipos:
 - **Barramento de endereços** – indica o local do acesso.
 - **Barramento de dados** – carrega os valores lidos/escritos.
 - **Barramento de controle** – define o tipo e o timing da operação

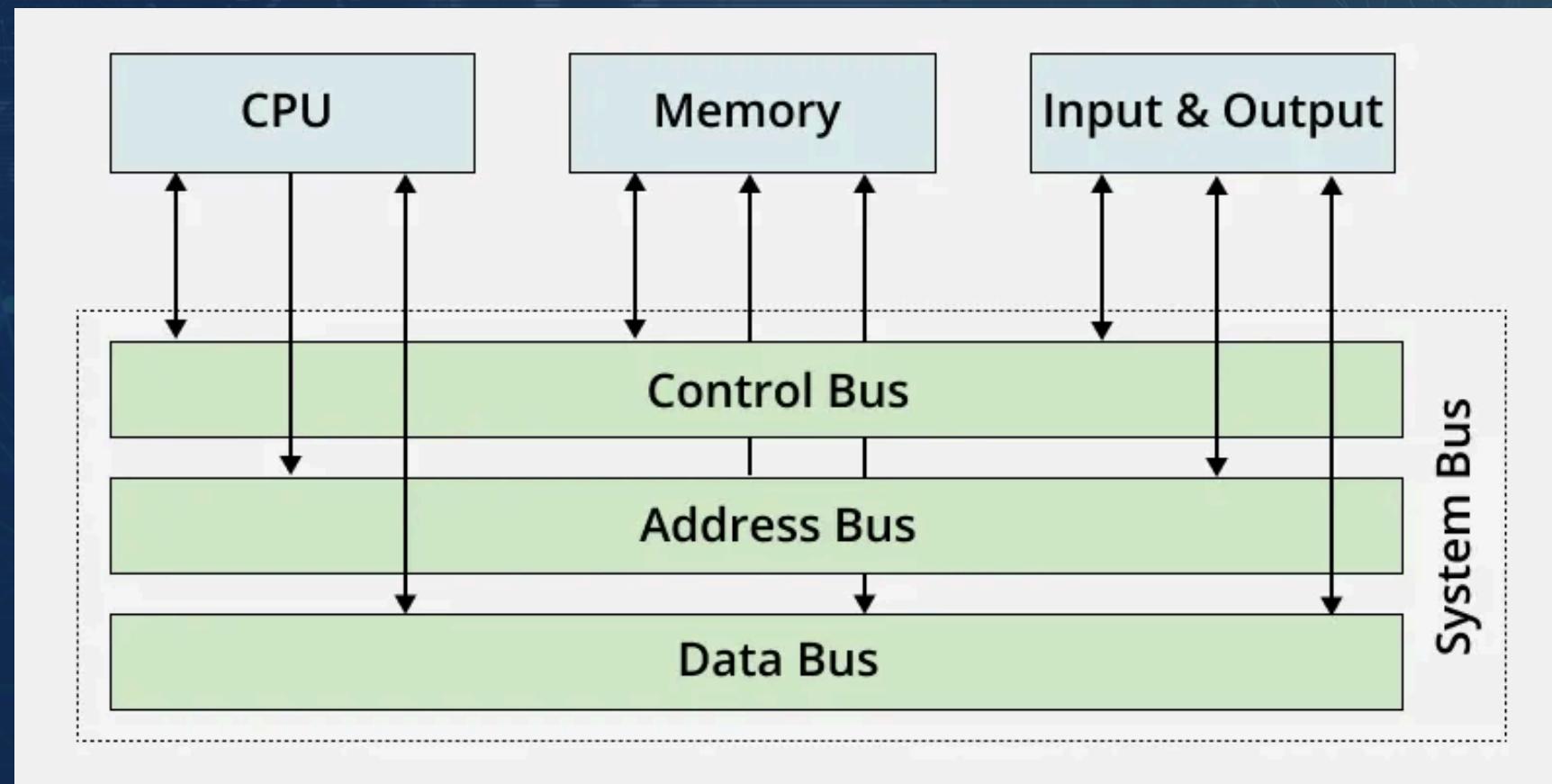


Figura 02 - Diagrama de um sistema de barramento

FUNDAMENTAÇÃO TEÓRICA ➤➤➤➤➤

6. PIO (Programmed Input/Output)

- Método de transmissão de dados,
- Via Entrada/Saída (E/S)
- Entre a CPU e um dispositivo periférico

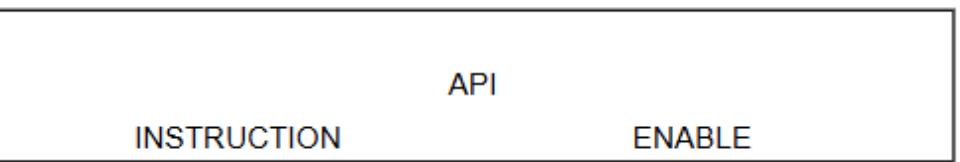
7. Syscall (System Call)

- Programas solicitam serviços do kernel do sistema operacional.

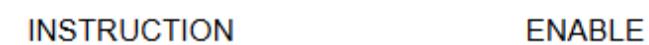
8. ISA (Instruction Set Architecture)

- Como o processador entende e executa comandos
- Inclui tipos de dados, registradores, operações básicas, etc.

HPS



LIGHTWEIGHT HPS-TO-FPGA BRIDGE



GOLDEN HARDWARE REFERENCE DESIGN

FPGA

HPS TO FPGA

Figura 03 - Diagrama da conexão HPS-FPGA

CAMINHO DE DADOS

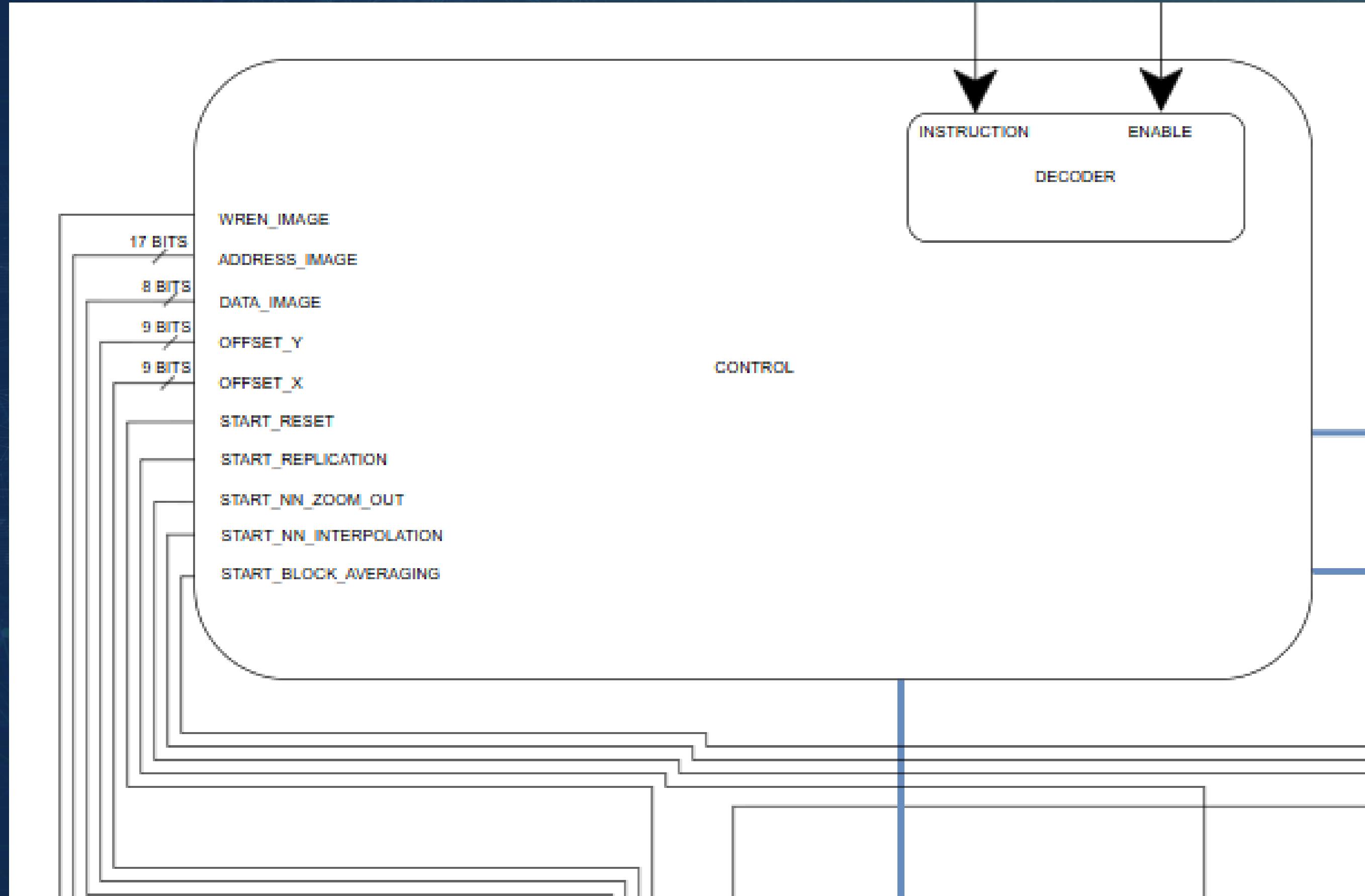


Figura 04 - Diagrama do módulo de controle e o caminho de dados

DECODIFICADOR

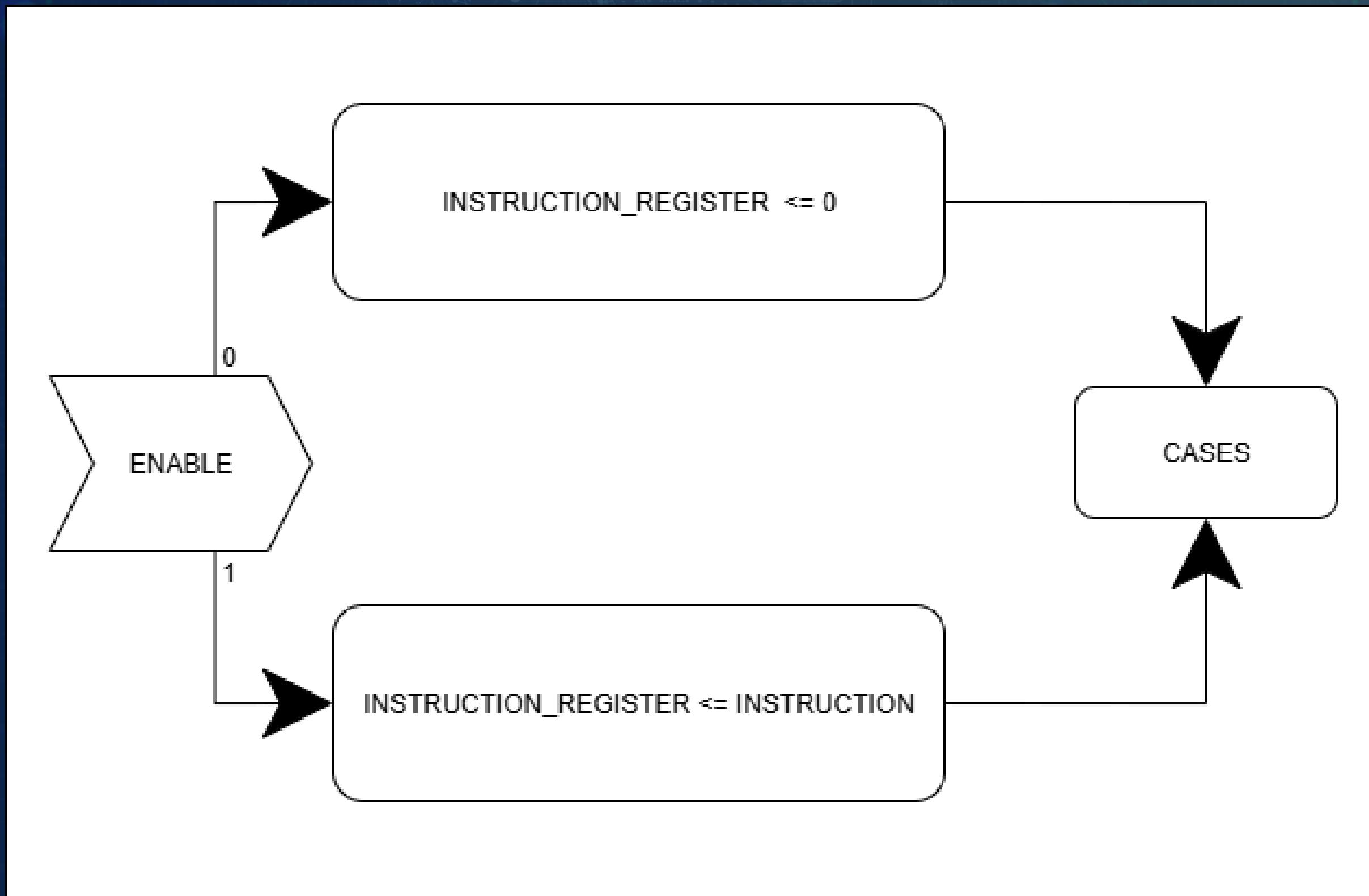


Figura 05 - Fluxograma do decodificador de instruções

MUDANÇAS NO HARDWARE

- Criação dos PIOs.
- Modificação da primeira memória.
- Implementação de um módulo decoder .
- Ajuste na máquina de estados das imagens.
- Inclusão dos módulos necessários para a comunicação com o HPS.
- Revisão e ajuste das conexões de ativação dos algoritmos.
- Adição dos offsets utilizados pelos algoritmos de Zoom In.



DIAGRAMA COMPLETO

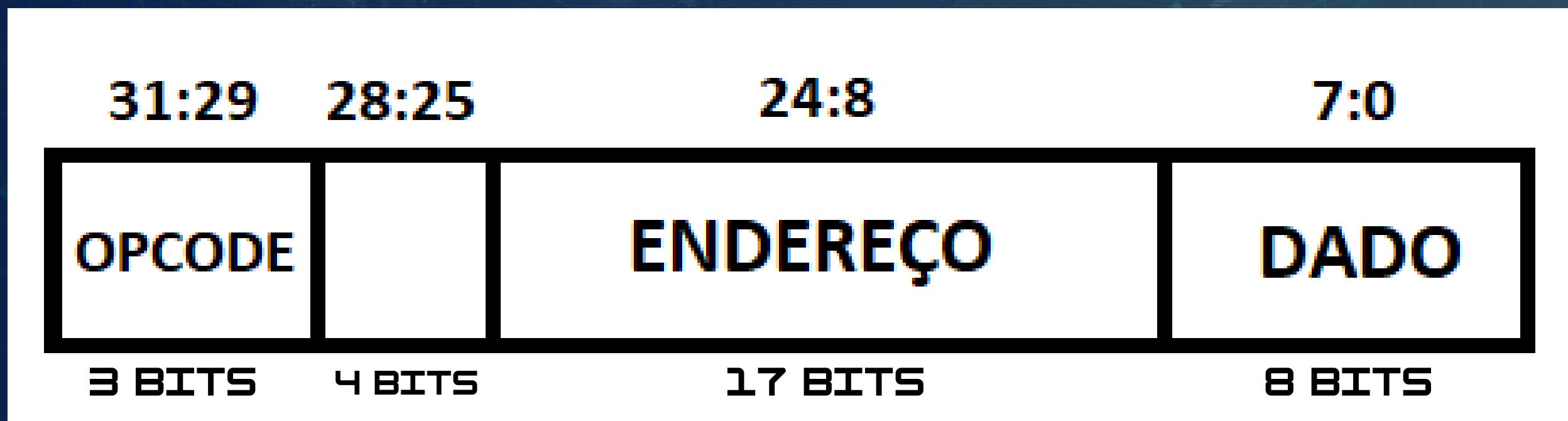
ARQUITETURA DA ISA



OPCODE (3 bits)	Descrição da Operação
001	Carregar Imagem
010	Vizinho mais próximo
011	Replicação
100	Decimação
101	Média de blocos
110	Reset

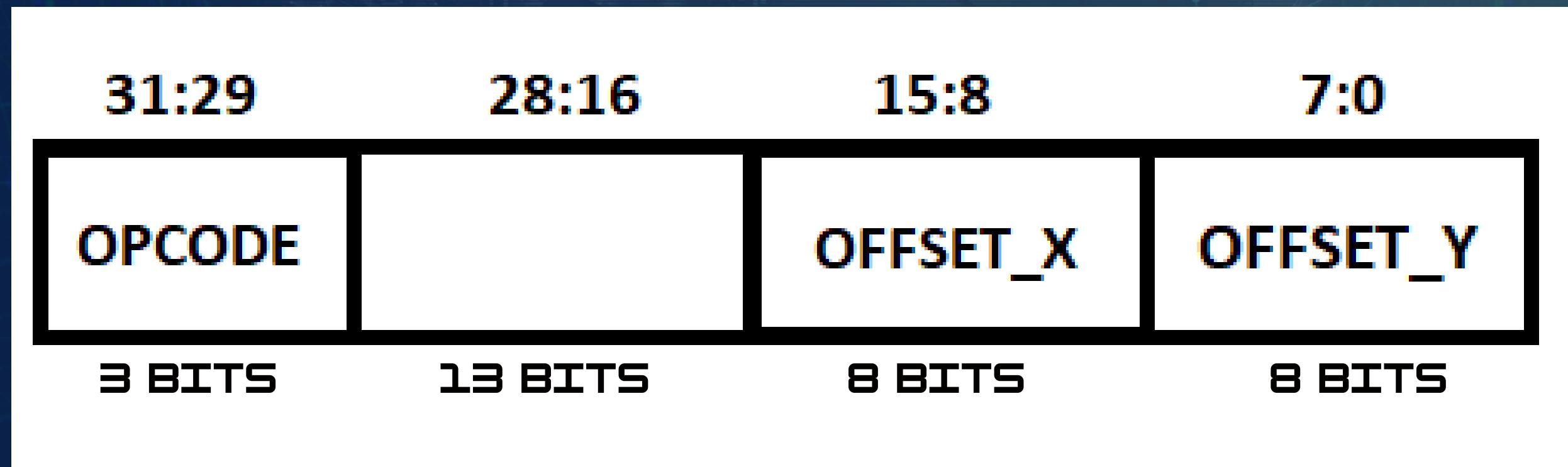
O OPCODE é o responsável por passar o código de operação que o decodificador irá ler e acionar a ação correspondente.

INSTRUÇÕES DO TIPO LOAD





INSTRUÇÃO DO TIPO ZOOM IN



INSTRUÇÃO DO TIPO ZOOM OUT E RESET

31:29

28:0

OPCODE

3 BITS

CHAMADAS DE SISTEMA

Syscalls utilizados	Descrição da Operação
3	Read
5	Open
6	Close
91	mmap
192	munmap

FUNÇÃO DO SYSCALL:

- Read: Ler dados do arquivo.
- Open: Utilizado para abrir arquivos, no nosso caso é o DEV/MEM e a imagem.
- Close: Usado para fechar arquivos, sendo esse o dev/mem.
- mmap: Utilizado para mapear a memória, estabelecendo a ponte de comunicação com o hardware.
- munmap: Desfaz o mapeamento da memória.

API

Função	Ação	Parâmetro
Initialization	Abre o dev/mem e faz o mapeamento da memória.	N/A
Open_image	Lê a imagem para um buffer e em seguida carrega ela para memória.	Imagen
nearest_neighbor	Aplica o algoritmo de vizinho mais próximo.	OFFSET X e Y
pixel_replication	Aplica o algoritmo de replicação.	OFFSET X e Y
pixel_decimation	Aplica o algoritmo de decimação	N/A
block_average	Aplica o algoritmo de média de blocos.	N/A
finalization	Desfaz o mapeamento e fecha o dev/mem.	N/A



INICIALIZAÇÃO

- Utiliza a chamada de sistema open para abrir o DEV/MEM.
- O mmap é chamado para mapear o endereço físico da ponte Lightweight para o espaço de endereçamento do programa.
- Por último, esses endereços retornados pelo mmap, são armazenados em ponteiros.
- Esses passos estabelecem o Memory-Mapped I/O (MMIO), que permite que as instruções em assembly se comuniquem com o hardware.

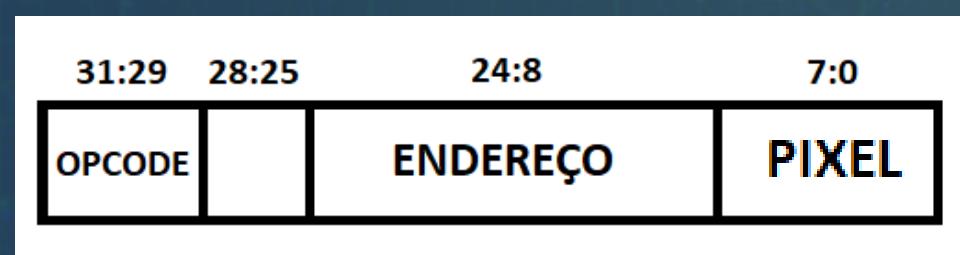
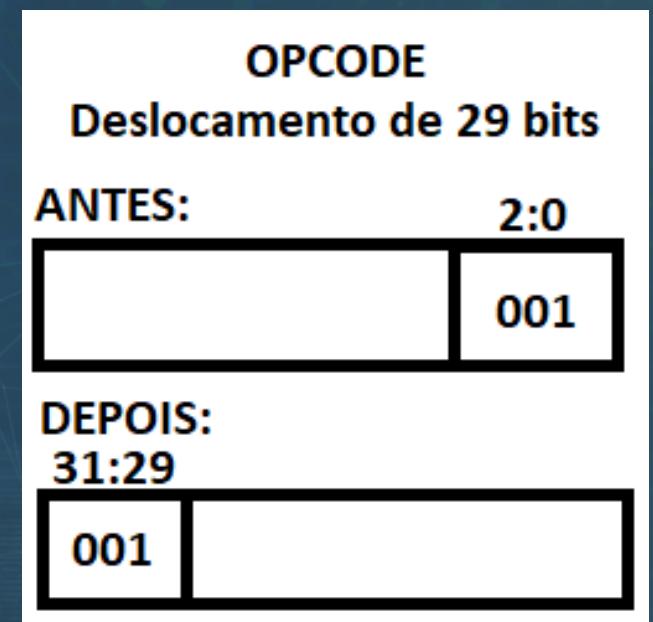
LEITURA DA IMAGEM

- É utilizado um buffer de 76800 bytes para conter todos os dados.
- Para leitura da imagem, é utilizado a chamada de sistema open para abrir a imagem.
- Em seguida, utilizamos a syscall read para ler o cabeçalho do arquivo e salva no buffer.
- Ocorre outra leitura para sobrescrever o buffer com os pixels da imagem.
- Ao final, o buffer contém todos os pixels que serão utilizados posteriormente.

CARREGAMENTO DA IMAGEM

Dentro do loop, podemos dividir em:

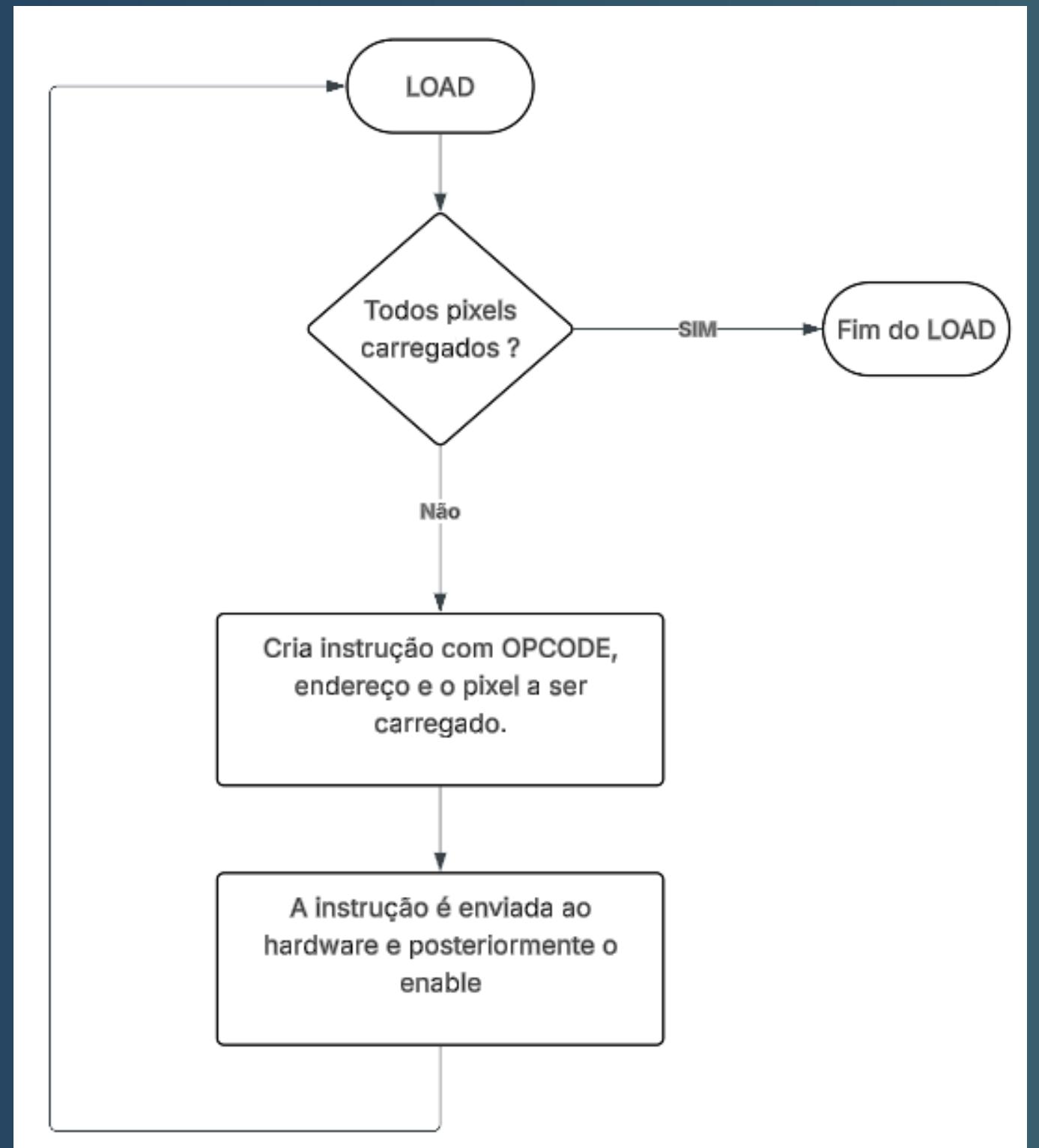
- O Opcode 001 é guardado em um registrador e deslocado 29 bits para esquerda.
- O byte do pixel é lido do buffer e coloca nos 8 bits menos significativos de um outro registrador.
- O contador do loop é utilizado como endereço e é deslocado 8 bits para esquerda em um registrador distinto dos outros dois.
- Todos esses registradores concatenados por meio de uma porta OR em uma única instrução.
- A instrução é enviada para o hardware.

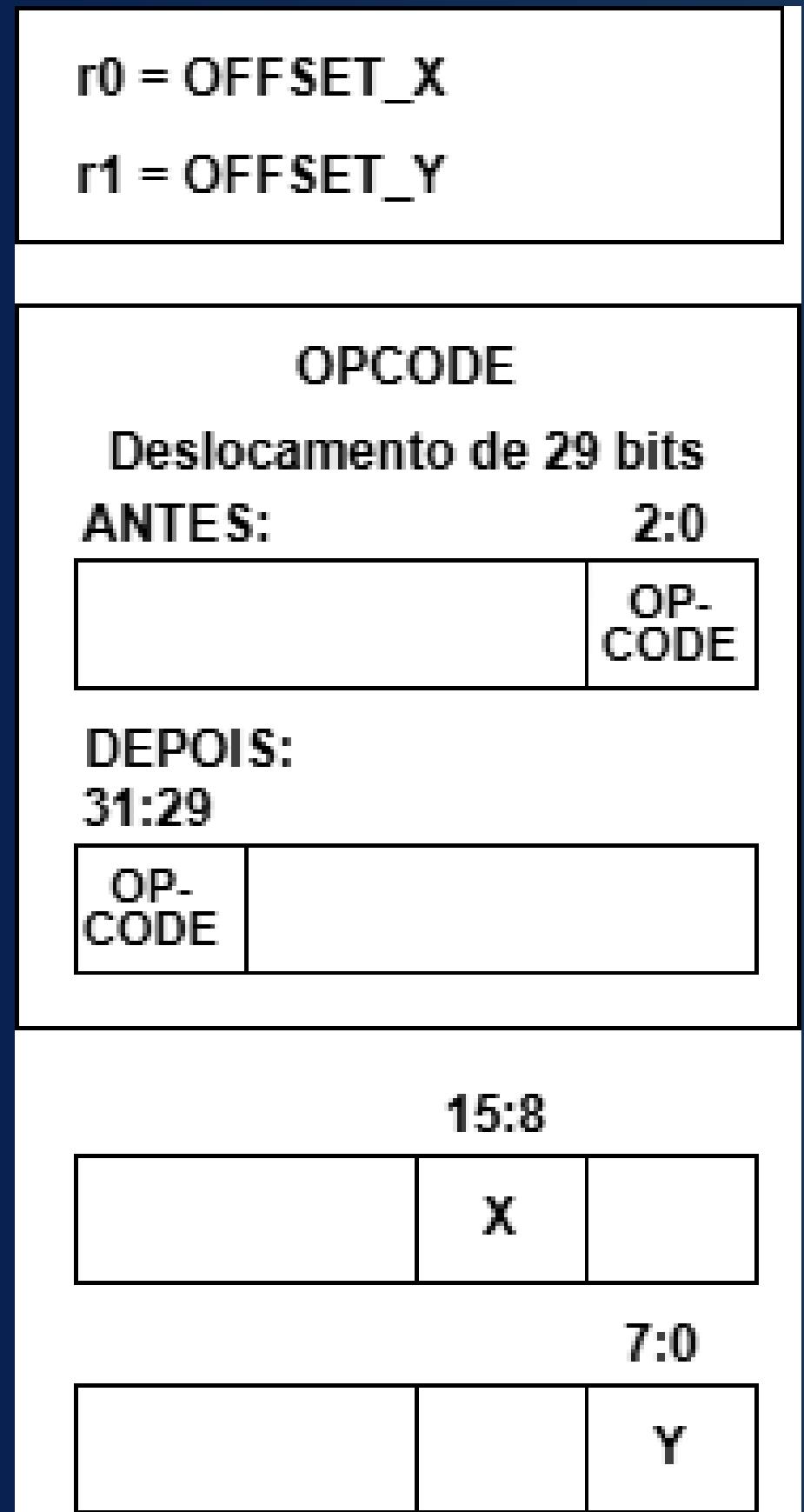


FIM DO CARREGAMENTO

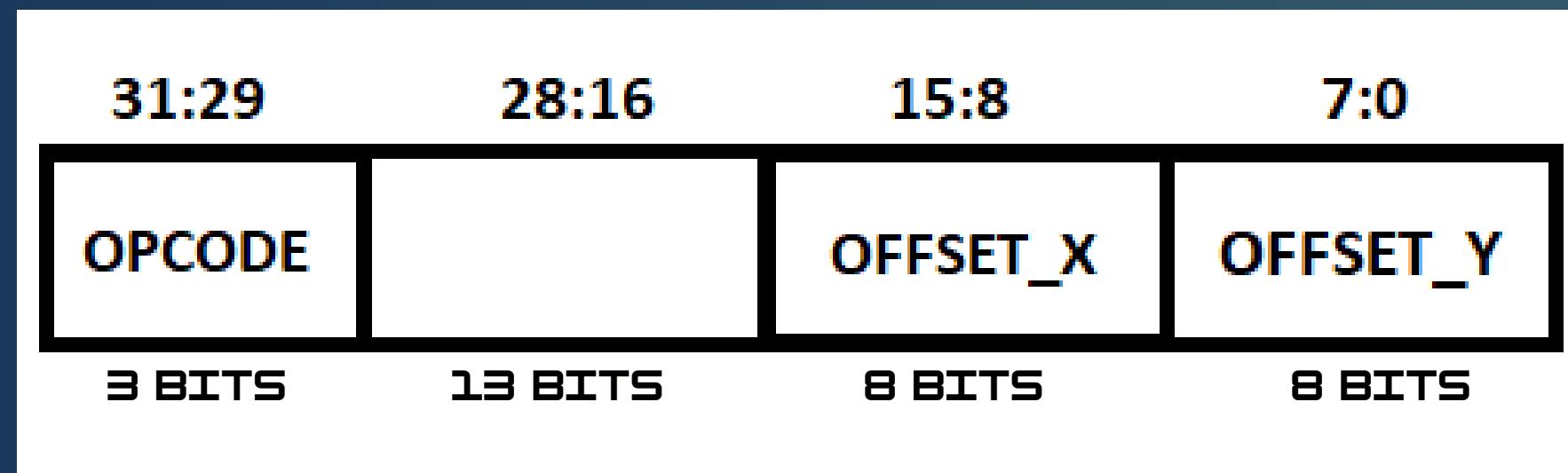
FLUXOGRAMA

- Emite a instrução de reset.
- Responsável por copiar os dados carregados da primeira memória para segunda memória.





INSTRUÇÃO DE ZOOM IN



CÁLCULO DE ENDEREÇO COM OFFSET NO ZOOM IN

ANTES:

ENDEREÇO DE LEITURA = $Y * 320 + X$

DEPOIS:

ENDEREÇO DE LEITURA =
 $(Y + \text{OFFSET_}Y) * 320 + (X + \text{OFFSET_}X)$

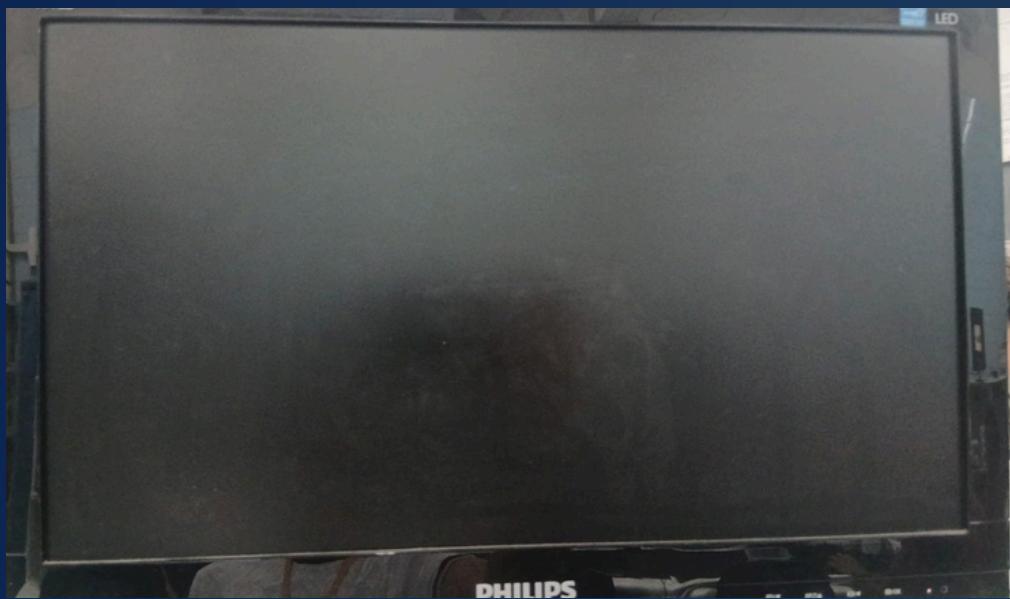
FINALIZAÇÃO

- É utilizado o suscall 91 para realizar o munmap.
- Usado o syscall 6 para fechar o dev/mem.
- Por fim, o programa é finalizado.

Casos de Teste

1º CASO:

Antes de abrir imagem



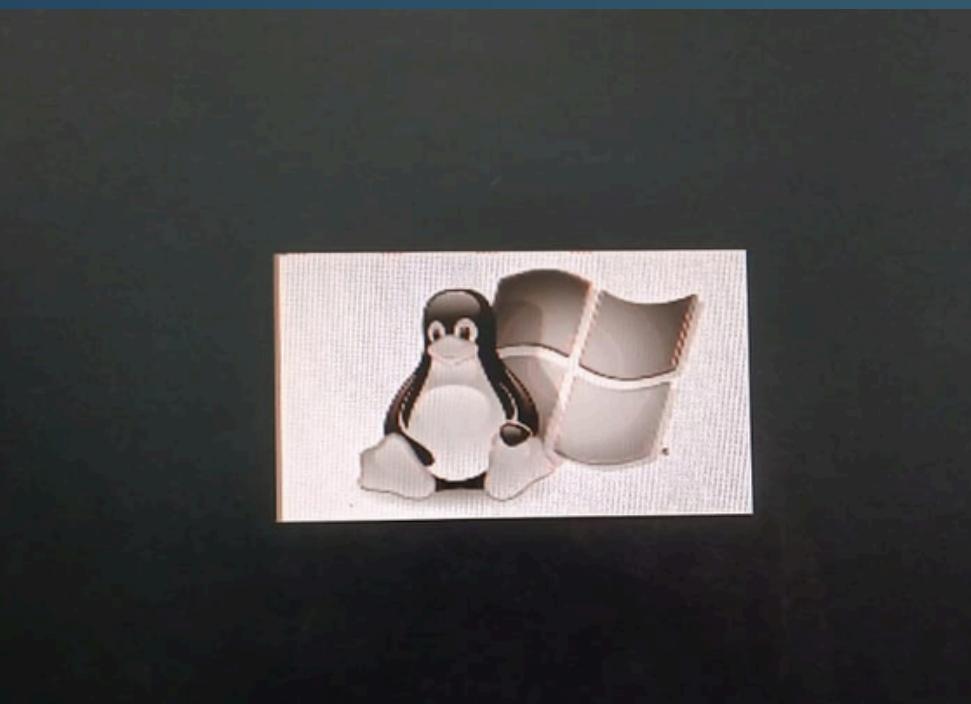
Após carregar imagem



2º CASO: Imagem Original



ZOOM OUT



Resultado

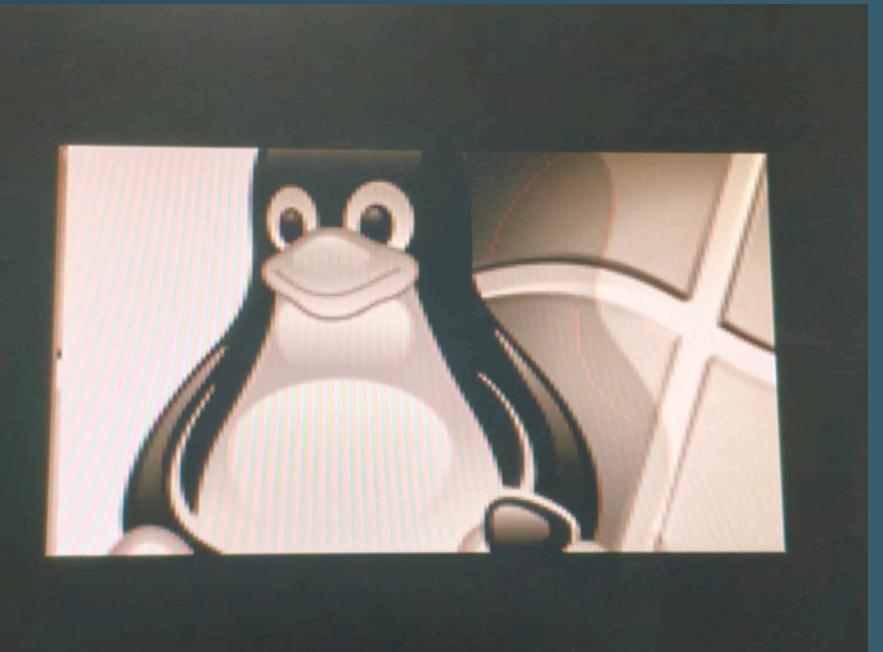
Casos de Teste

3º CASO:

Imagen Original



Resultado



Zoom In com os parâmetros

```
=====
 MENU DE PROCESSAMENTO DE IMAGEM
=====
[1] - Vizinho Mais Próximo (com navegação)
[2] - Replicação de Pixel (com navegação)
[3] - Decimação
[4] - Média de Blocos
[5] - Abrir Imagem
[0] - Sair

Escolha uma opção: 1

== Vizinho Mais Próximo ==
Digite a coordenada X inicial (0-160): 50
Digite a coordenada Y inicial (0-120): 50

Iniciando na posição X=50, Y=50
Use WASD para navegar, Q para sair
```

REFERÊNCIAS



- **Instruction Set Architecture (ISA).** Disponível em: <https://www.arm.com/glossary/isa>. Acesso em: 18 nov. 2025.
- **What is a Computer Bus?** Disponível em: <https://www.geeksforgeeks.org/computer-organization-architecture/what-is-a-computer-bus/>. Acesso em: 18 nov. 2025.
- **HPS – Overview.** Disponível em: <https://insper.github.io/Embarcados-Avancados/Tutorial-HPS/>. Acesso em: 18 nov. 2025.
- **Designing with Avalon® and AXI Interfaces.** Disponível em: <https://www.intel.com/content/www/us/en/docs/programmable/683609/21-3/designing-with-and-axi-interfaces.html>. Acesso em: 18 nov. 2025.
- **FPGA academy.** Disponível em: <https://fpgacademy.org>





FTM!

