



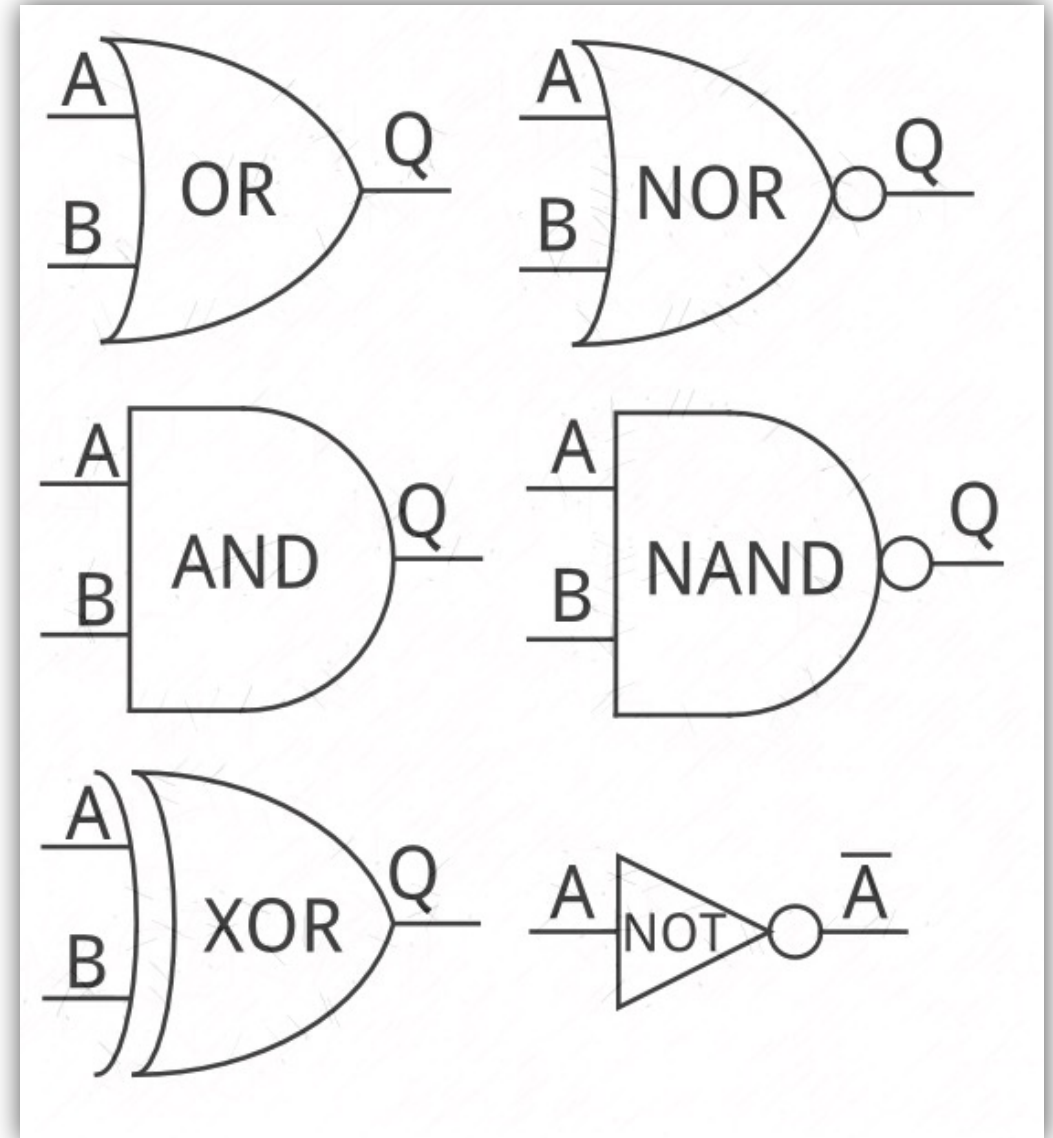
C209 – Computação Gráfica e Multimídia
EC215 – Multimídia

Operações no Domínio do Espaço

Marcelo Vinícius Cysneiros Aragão
marcelovca90@inatel.br

Conteúdo

- Operações
 - Aritméticas
 - Adição
 - Subtração
 - Multiplicação
 - Divisão
 - *Blending*
 - Lógicas
 - AND/NAND
 - OR/NOR
 - XOR/XNOR
 - NOT
 - SHIFT
- Limites Inferior e Superior nas Operações
- Referências



OPERAÇÕES COM IMAGENS

Introdução

- A aritmética de imagens é a forma mais simples de processamento de imagens.
- Os operadores são aplicados de uma forma **pixel-por-pixel**, portanto as imagens devem ser **do mesmo tamanho**.
- Entretanto, uma das imagens de entrada pode ser um valor constante, como por exemplo, na adição de um deslocamento constante a uma imagem.
- A vantagem dos operadores aritméticos é processamento **simples e rápido**.

Introdução

- As imagens processadas são frequentemente tiradas da mesma cena em diferentes instantes:
 - **Redução do ruído** aleatório adicionando imagens sucessivas da mesma cena
 - **Detecção de movimento** subtraindo duas imagens sucessivas.
- Os operadores lógicos são frequentemente usados para combinar duas imagens (principalmente binárias). No caso de imagens inteiras, o operador lógico é normalmente aplicado bit a bit.
 - Utilização de uma máscara binária para seleccionar uma determinada região de uma imagem.

OPERAÇÕES ARITMÉTICAS COM IMAGENS

$+$ $-$ \times \div Σ

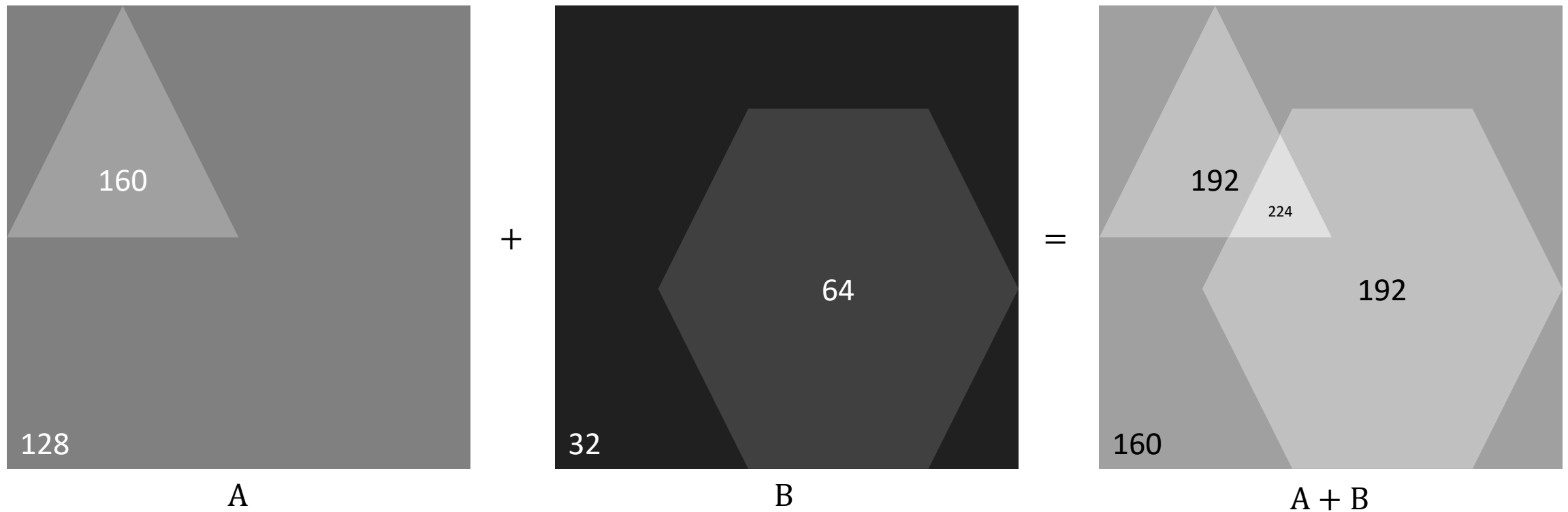
Operações Aritméticas: Adição

- Este operador toma como entrada duas imagens de tamanho idêntico e produz como saída uma terceira imagem do mesmo tamanho das duas primeiras, em que cada valor de pixel é a soma dos valores do pixel correspondente de cada um das imagens de entrada.
- Também é possível usar apenas uma única imagem como entrada e somar um valor constante a todos os pixels.

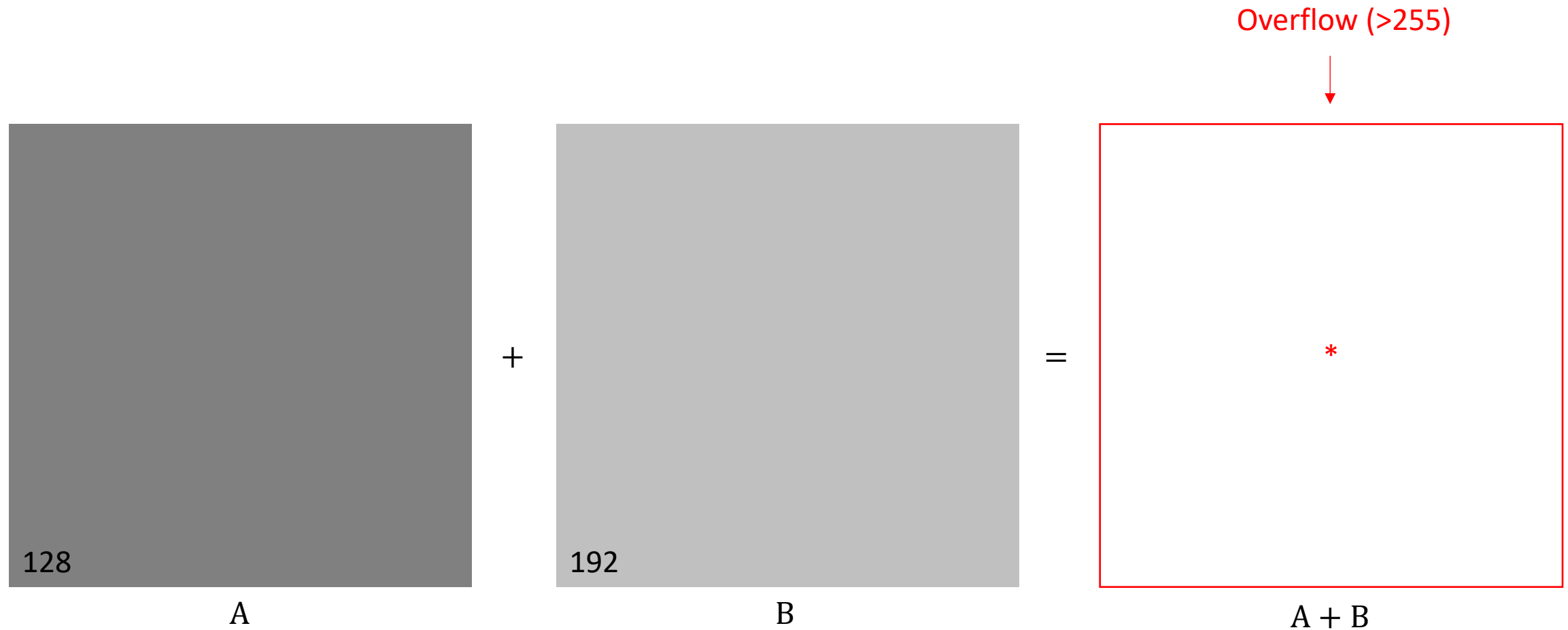
$$Q(i, j) = P_1(i, j) + P_2(i, j)$$

$$Q(i, j) = P_1(i, j) + C$$

Operações Aritméticas: Adição



Operações Aritméticas: Adição

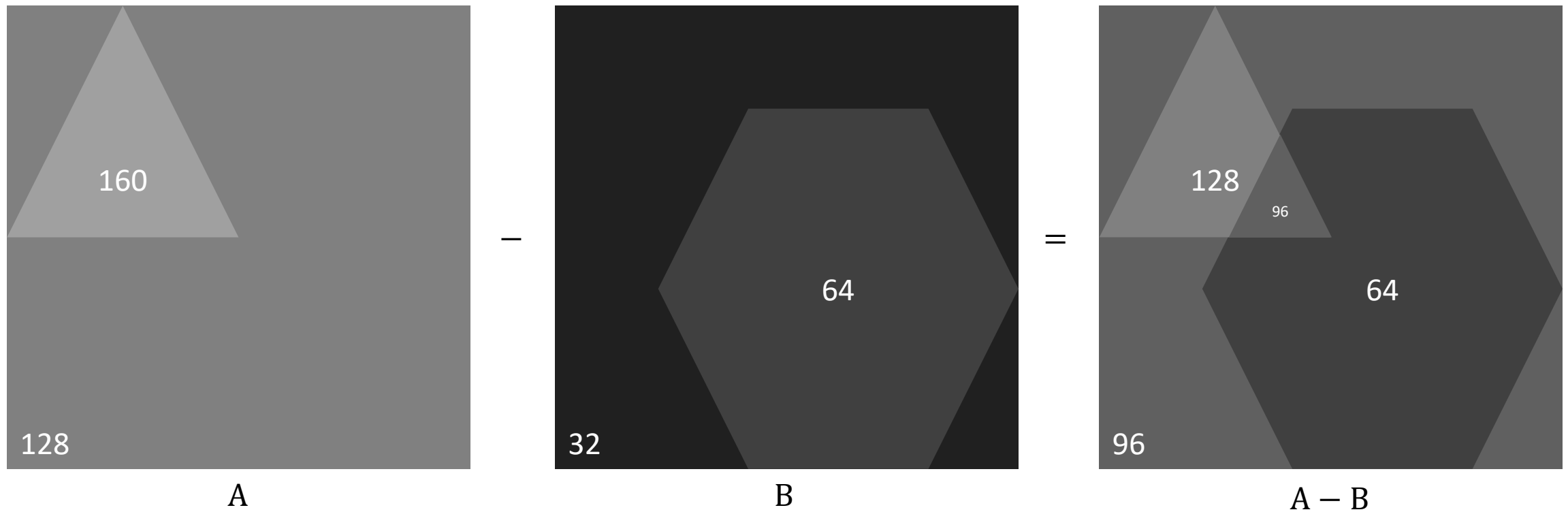


Operações Aritméticas: Subtração

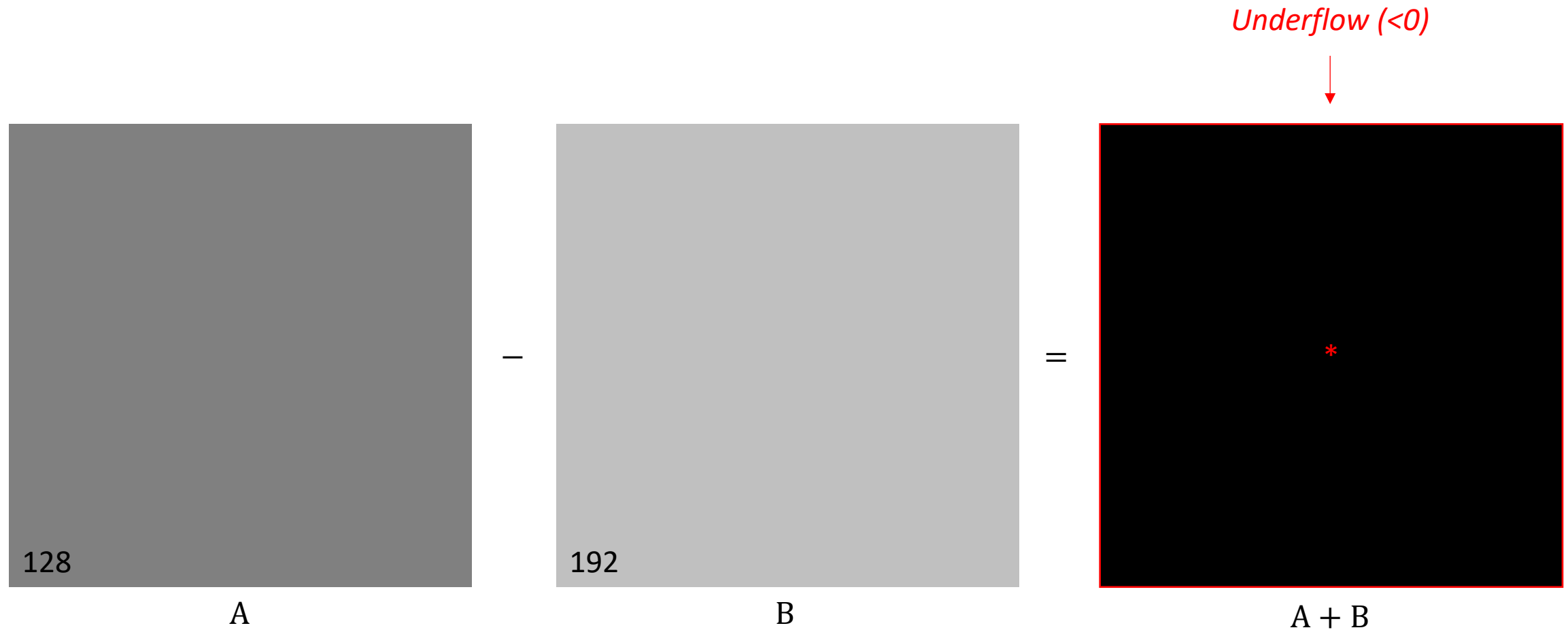
- Este operador toma como entrada duas imagens de tamanho idêntico e produz como saída uma terceira imagem do mesmo tamanho das duas primeiras, em que cada valor de pixel é a subtração dos valores do pixel correspondente de cada um das imagens de entrada.
- Também é possível usar apenas uma única imagem como entrada e subtrair um valor constante de todos os pixels.
- Algumas versões do operador apenas produzirão a diferença absoluta entre os valores de pixel, em vez da saída direta (com sinal).

$$Q(i, j) = P_1(i, j) - P_2(i, j) \qquad Q = |P_1(i, j) - P_2(i, j)| \qquad Q = P_1(i, j) - C$$

Operações Aritméticas: Subtração



Operações Aritméticas: Subtração



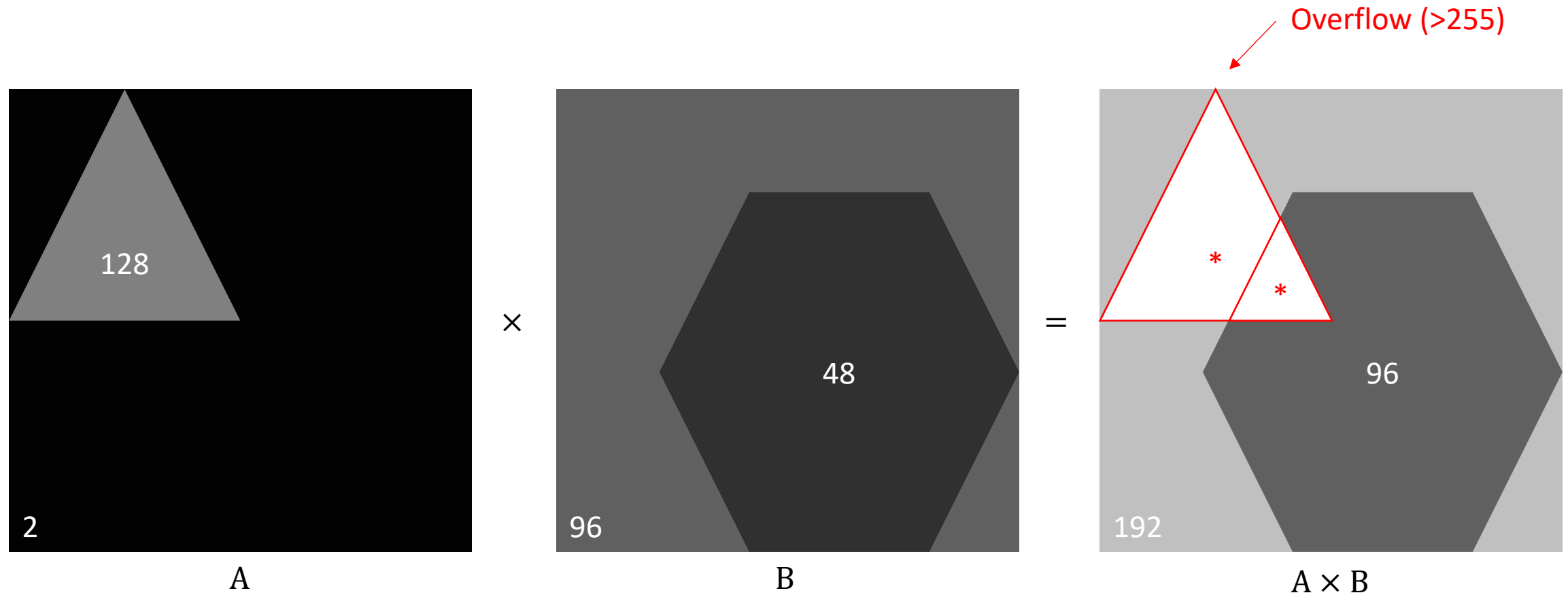
Operações Aritméticas: Multiplicação

- Também é possível realizar a multiplicação de duas formas distintas.
- A primeira toma duas imagens de entrada e produz uma imagem de saída na qual os valores de pixel são os da primeira imagem, multiplicados pelos valores dos valores correspondentes na segunda imagem.
- A segunda forma toma uma única imagem de entrada e produz saída na qual cada valor de pixel é multiplicado por uma constante especificada. Esta última forma é mais utilizada e é chamada de escala.

$$Q(i, j) = P_1(i, j) \times P_2(i, j)$$

$$Q(i, j) = P_1(i, j) \times C$$

Operações Aritméticas: Multiplicação



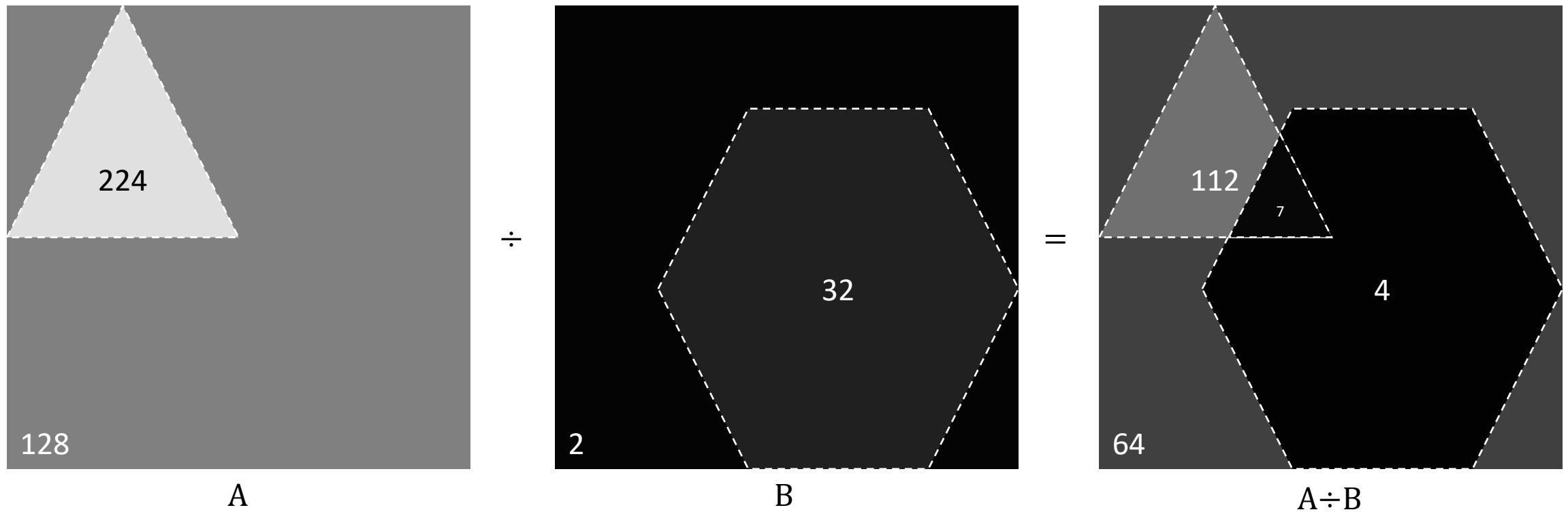
Operações Aritméticas: Divisão

- O operador de divisão de imagem toma normalmente duas imagens como entrada e produz um terceiro cujos valores de pixel são os valores de pixel da primeira imagem divididos pelos valores de pixel correspondentes da segunda imagem.
- Muitas implementações também podem ser usadas com apenas uma única imagem de entrada, caso em que cada valor de pixel na imagem é dividido por uma constante especificada.

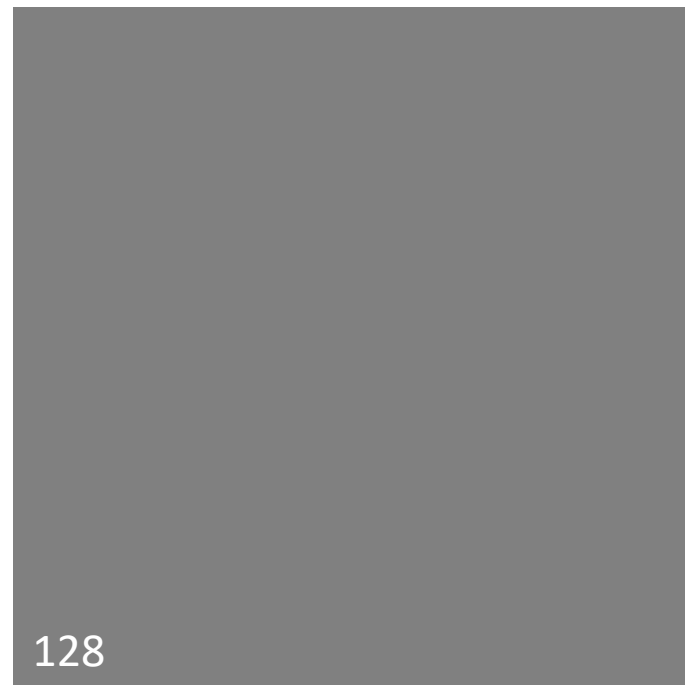
$$Q(i, j) = P_1(i, j) \div P_2(i, j)$$

$$Q(i, j) = P_1(i, j) \div C$$

Operações Aritméticas: Divisão

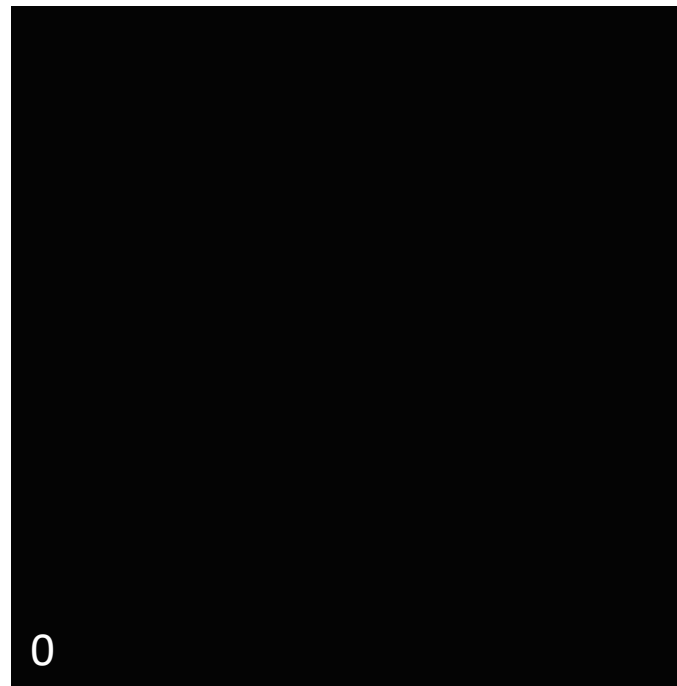


Operações Aritméticas: Divisão



A

÷



B

=

A problem has been detected and windows has been shut down to prevent damage to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** STOP: 0x00000001 (0x0000000C, 0x00000002, 0x00000000, 0xF86B5A89)

*** gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further assistance.

Divisão por zero

A ÷ B

Operações Aritméticas: *Blending*

- Este operador forma uma mistura de duas imagens de entrada do mesmo tamanho.
- Semelhante à adição de pixel, o valor de cada pixel na imagem de saída é uma combinação linear dos valores de pixel correspondentes nas imagens de entrada.
- Os coeficientes da combinação linear são especificados pelo usuário e definem a razão pela qual a escala de cada imagem antes de combiná-los. Estas proporções são aplicadas de modo a que os valores de pixel de saída não excedam o valor máximo de pixel.

$$Q(i, j) = X \times P_1(i, j) + (1 - X) \times P_2(i, j)$$

Operações Aritméticas: *Blending*

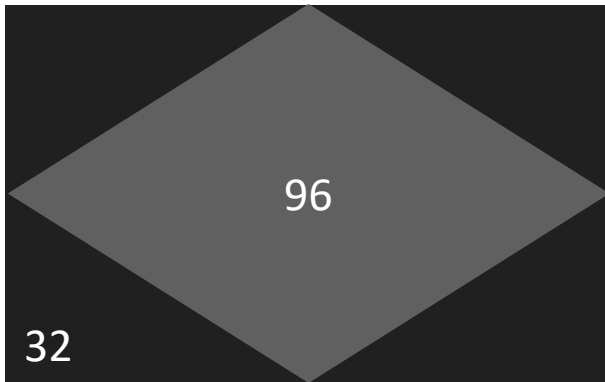


Imagem original

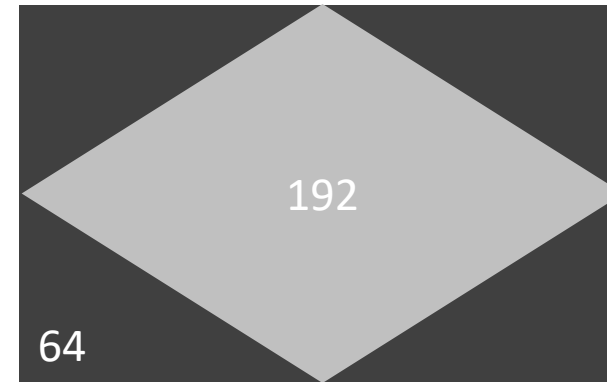
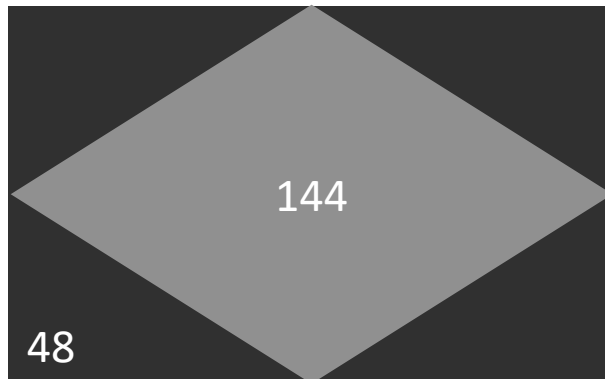
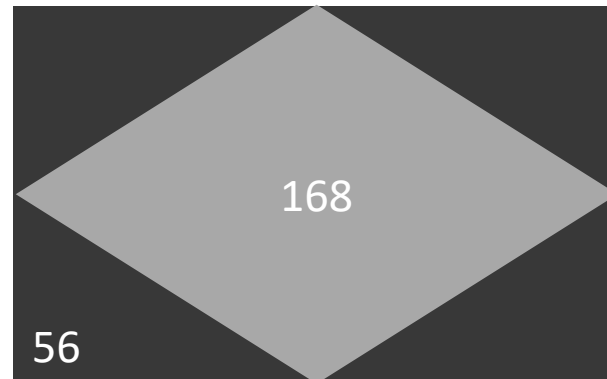


Imagem a ser “*blended*”



Blending com fator 0.5
(retângulo: $0.5 \cdot 32 + 0.5 \cdot 64 = 48$)
(losango: $0.5 \cdot 96 + 0.5 \cdot 192 = 144$)



Blending com fator 0.25
(retângulo: $0.25 \cdot 32 + 0.75 \cdot 64 = 56$)
(losango: $0.25 \cdot 96 + 0.75 \cdot 192 = 168$)

OPERAÇÕES LÓGICAS COM IMAGENS

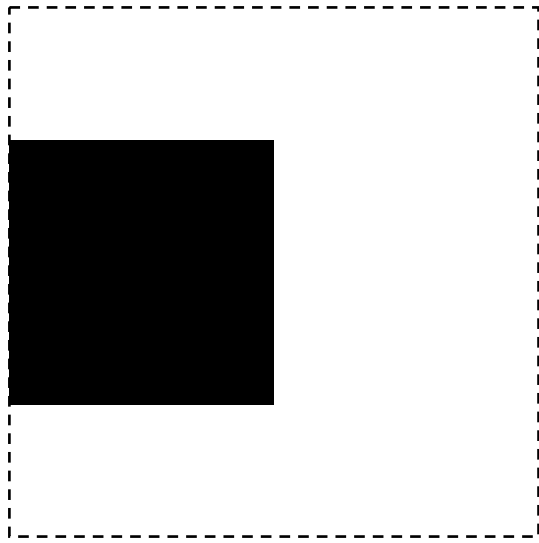
$\wedge \vee \oplus \neg \ll \gg$

Operações Lógicas: AND/NAND

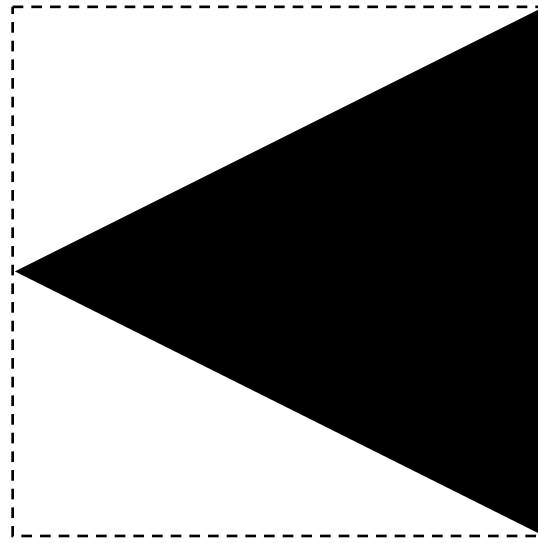
A	B	Q	A	B	Q
0	0	0	0	0	1
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	0
AND			NAND		

- AND e NAND são exemplos de operadores lógicos com as tabelas-verdade mostradas acima, à direita.
- Como pode ser visto, os valores de saída de NAND são simplesmente o inverso dos valores de saída correspondentes de AND.
- O operador AND (e similarmente o NAND) toma duas imagens binárias como entrada, e retorna uma terceira imagem cujos valores de pixel são apenas aqueles da primeira imagem, após realizar uma operação AND com os pixels correspondentes a partir da segunda.
- **Em imagens vistas na forma binária, os objetos (tom preto) são representados pelo valor lógico 1, e o fundo (tom branco) por 0.**
- Uma variação desse operador leva apenas uma única imagem de entrada e executa AND/NAND pixel a pixel com um valor constante especificado.

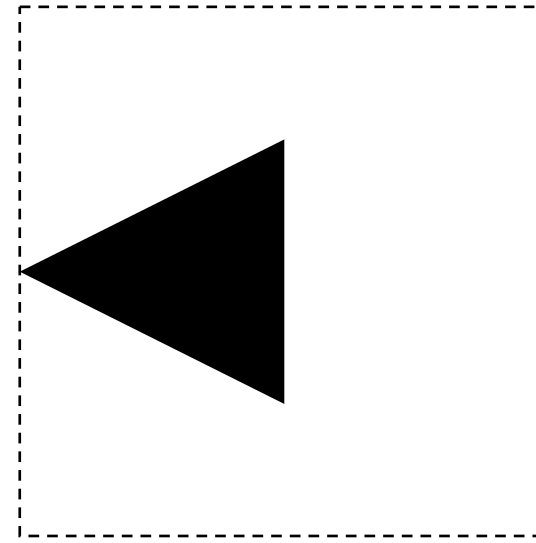
Operações Lógicas: AND/NAND



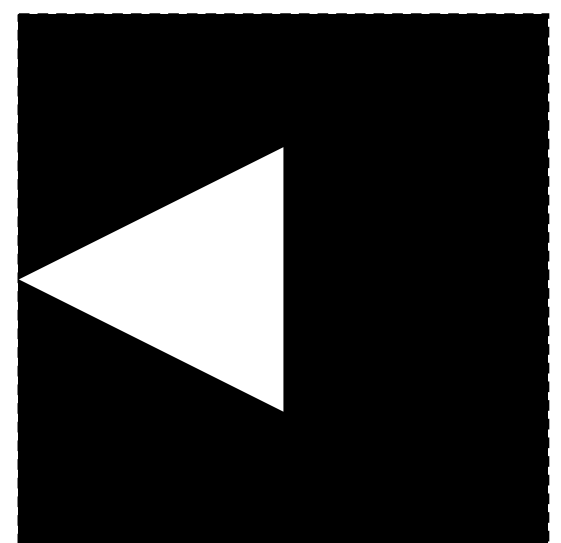
A



B



A AND B



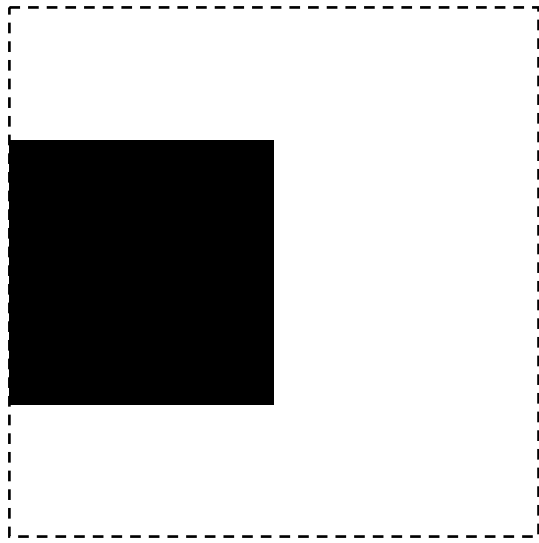
A NAND B

Operações Lógicas: OR/NOR

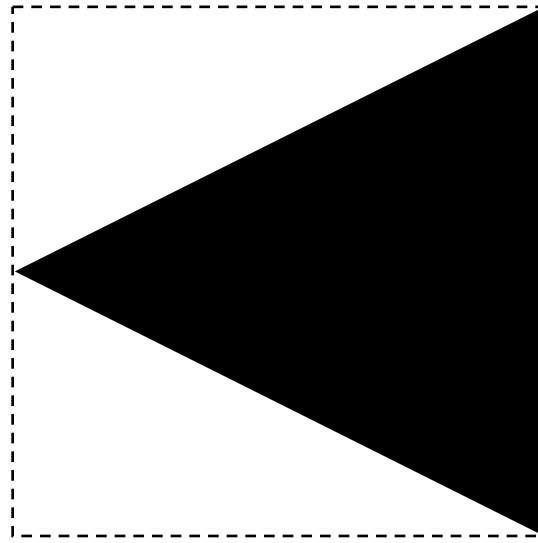
A	B	Q	A	B	Q
0	0	0	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	1	1	1	0
OR			NOR		

- OR e NOR são exemplos de operadores lógicos com as tabelas-verdade acima, à direita.
- Como pode ser visto, os valores de saída de NOR são simplesmente os inversos dos valores de saída correspondentes de OR.
- O operador OR (e similarmente o NOR) tipicamente toma duas imagens binárias como entrada, e retorna uma terceira imagem cujos valores de pixel são apenas aqueles da primeira imagem, após realizar uma operação OR com os pixels correspondentes a partir da segunda.
- Em imagens vistas na forma binária, os objetos (tom preto) são representados pelo valor lógico 1, e o fundo (tom branco) por 0.
- Uma variação desse operador leva apenas uma única imagem de entrada e executa OR/NOR pixel a pixel com um valor constante especificado.

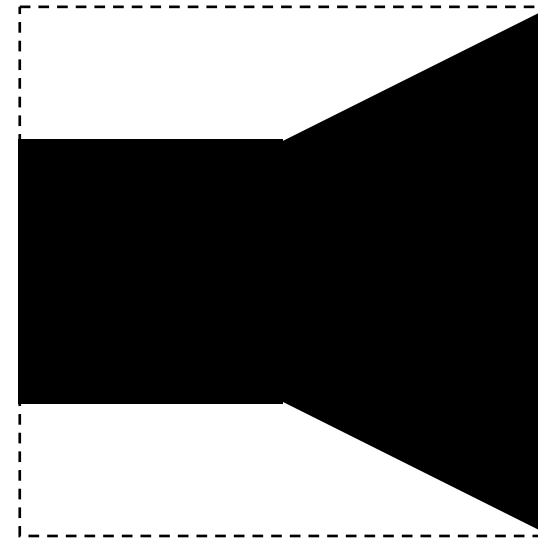
Operações Lógicas: OR/NOR



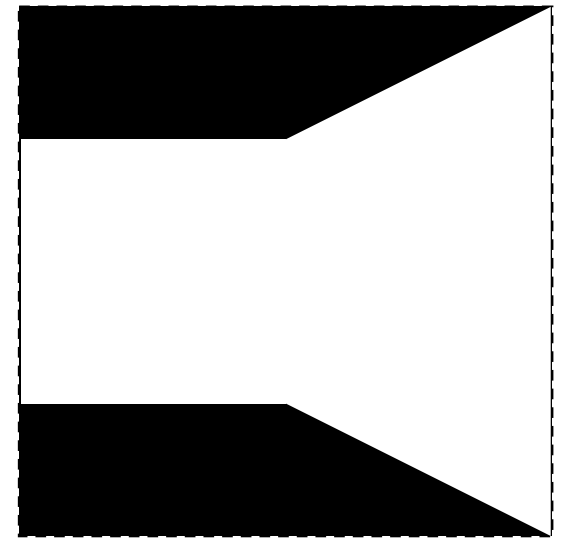
A



B



A OR B



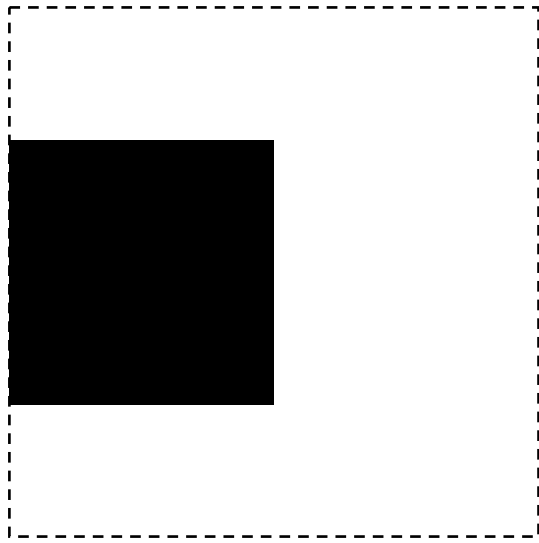
A NOR B

Operações Lógicas: XOR/XNOR

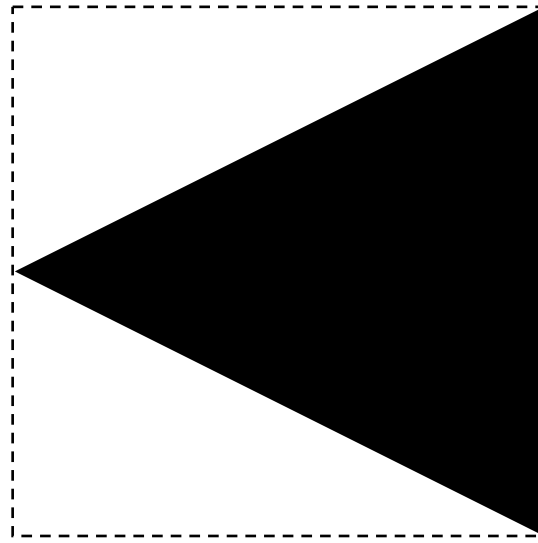
A	B	Q	A	B	Q
0	0	0	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	1
XOR			XNOR		

- XOR e XNOR são exemplos de operadores lógicos com as tabelas-verdade acima, à direita. A função XOR é apenas verdadeira se apenas um (e apenas um, ou seja, exclusivamente) dos valores de entrada for verdadeiro, e falso caso contrário.
- Como pode ser visto, os valores de saída de XNOR são simplesmente o inverso dos valores de saída correspondentes de XOR.
- O operador XOR (e similarmente o XNOR) tipicamente toma duas imagens binárias como entrada, e retorna uma terceira imagem cujos valores de pixel são apenas aqueles da primeira imagem submetidos à uma operação XOR com os pixels correspondentes da segunda.
- Em imagens vistas na forma binária, os objetos (tom preto) são representados pelo valor lógico 1, e o fundo (tom branco) por 0.
- Uma variação deste operador toma uma única imagem de entrada e executa XOR/XNOR pixel a pixel com um valor constante especificado.

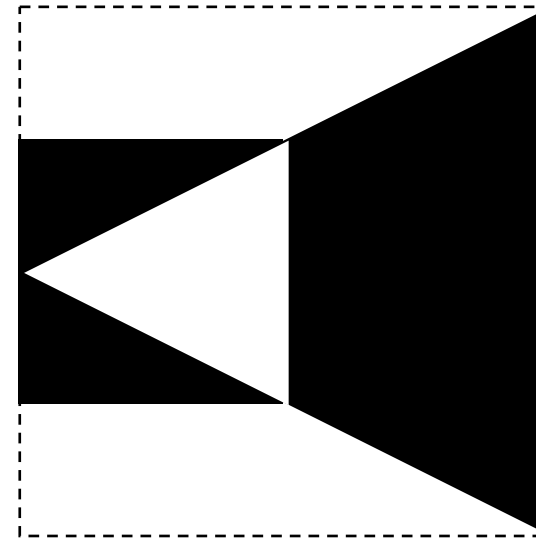
Operações Lógicas: XOR/XNOR



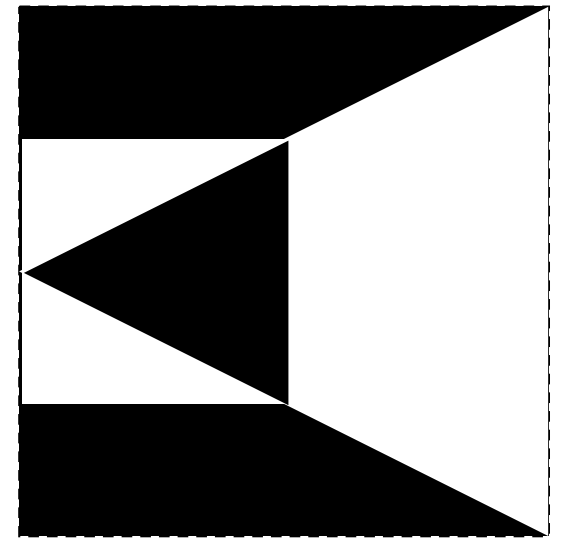
A



B



A XOR B



A XNOR B

A	Q
0	1
1	0

NOT

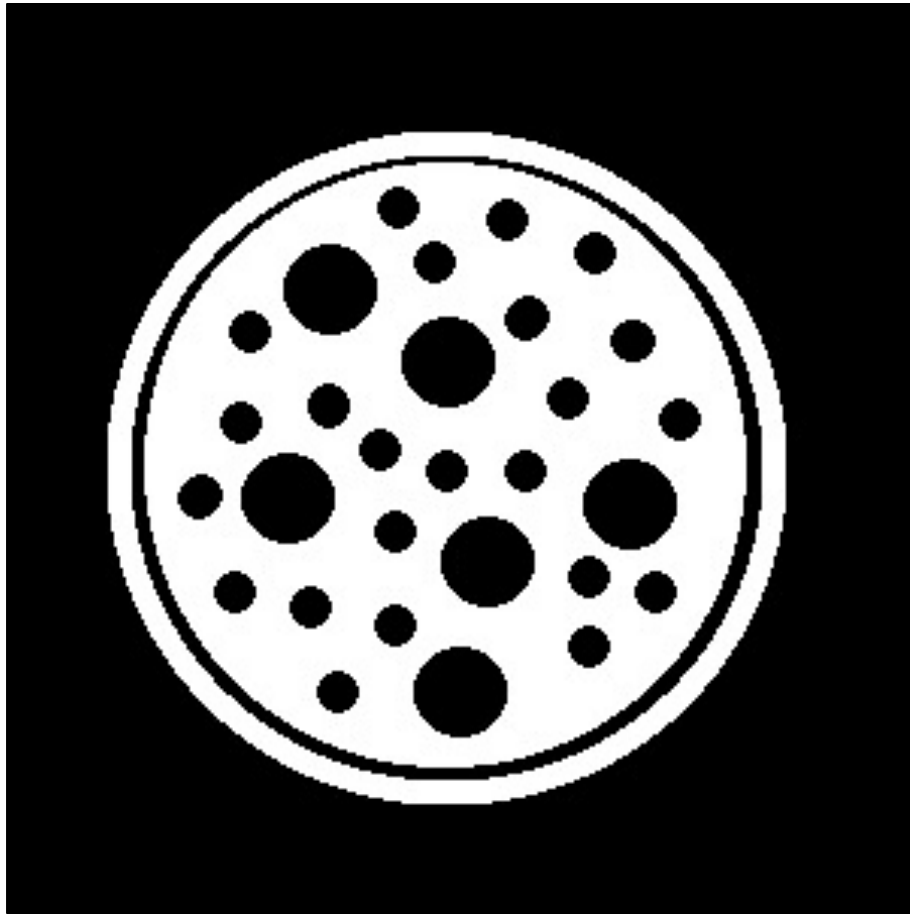
Operações Lógicas: NOT

- O operador lógico NOT (ou inversor) toma uma imagem binária ou em escala de cinza como entrada e produz seu negativo fotográfico, ou seja, áreas escuras na imagem de entrada tornam-se claras e áreas claras ficam escuras.

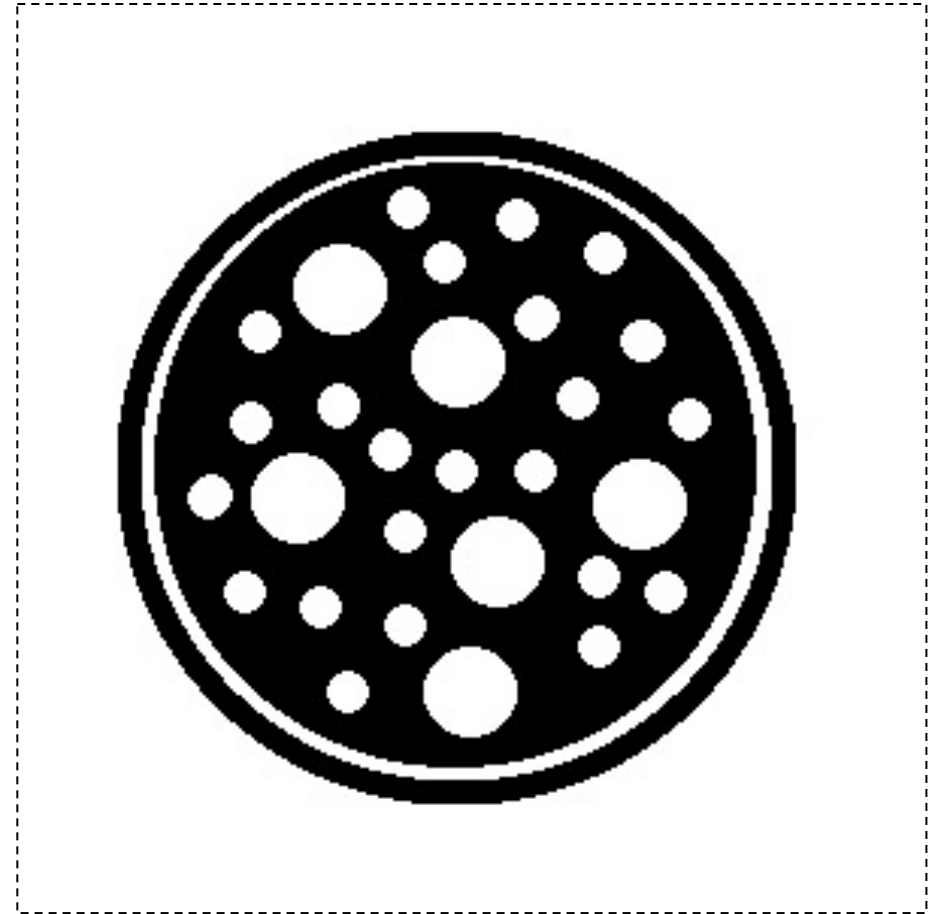
$$Q(i, j) = 255 - P(i, j)$$

$$Q(i, j) = -P(i, j)$$

Operações Lógicas: NOT

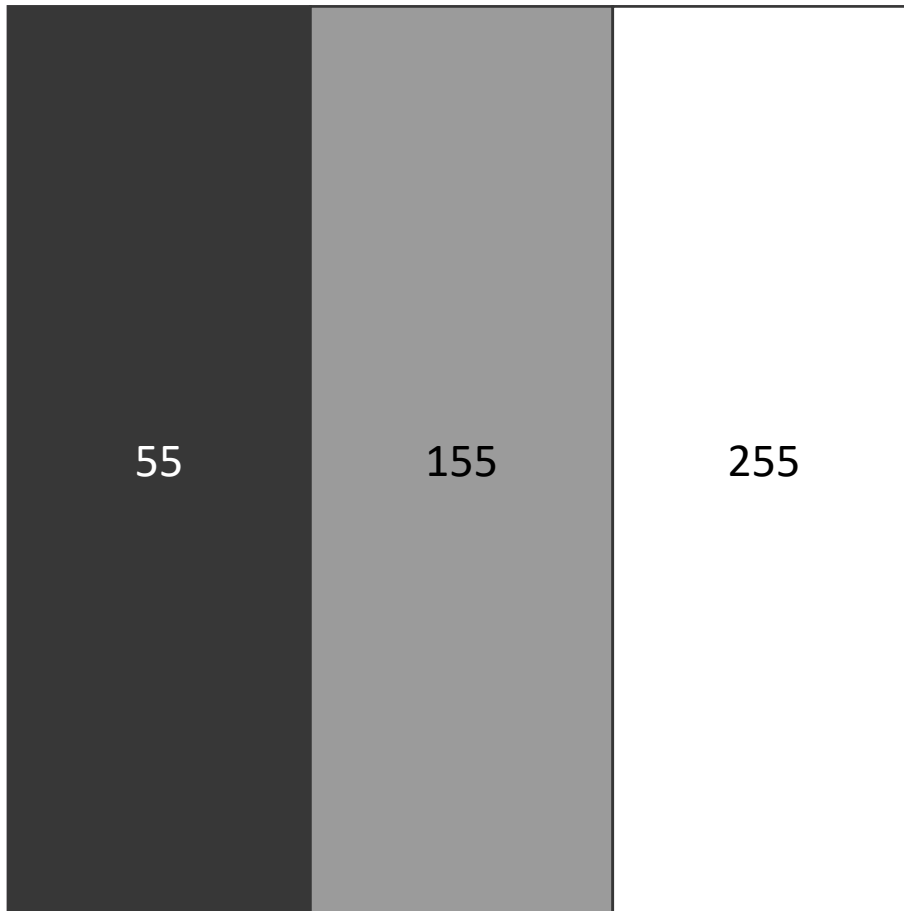


A

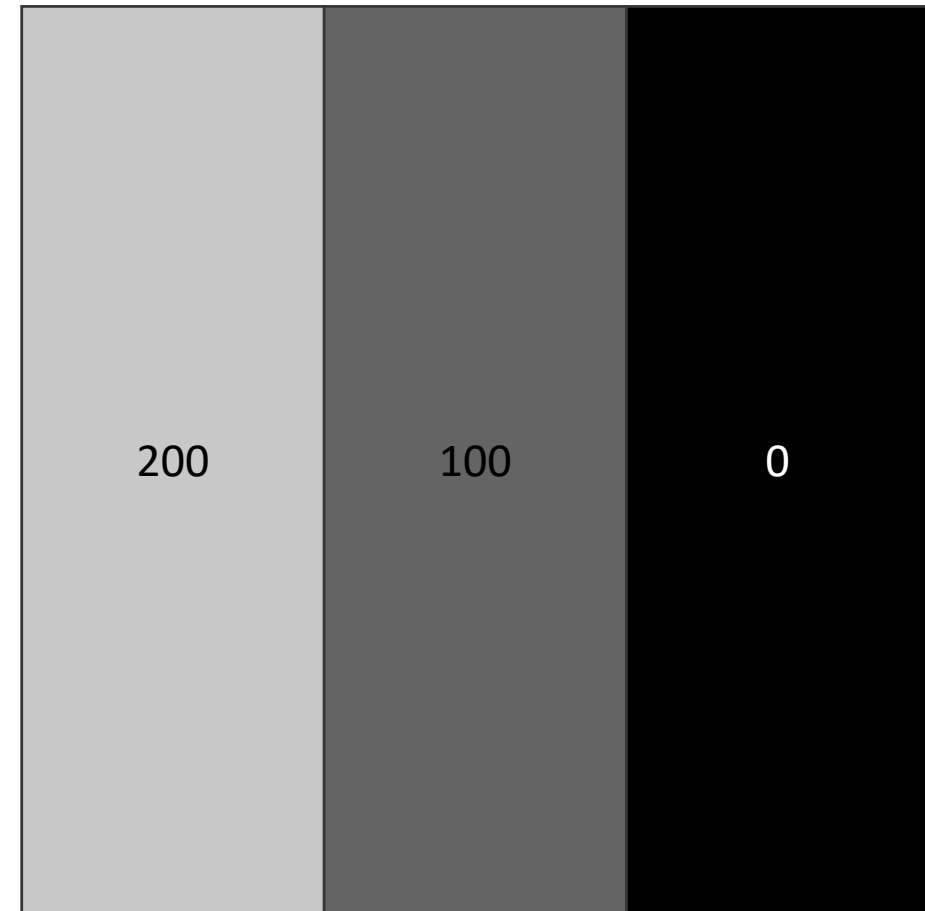


NOT A

Operações Lógicas: NOT



A



NOT A

Operações Lógicas: SHIFT

- O operador de shift trabalha em imagens representadas em formato de byte de pixel ou inteiros, onde cada valor de pixel é armazenado como um número binário com uma quantidade fixa de bits.
- O operador desloca a representação binária de cada pixel para a esquerda ou para a direita por um número predefinido de posições.
- Deslocar um número binário por um bit é equivalente a multiplicar (quando se desloca para a esquerda) ou dividir (quando se desloca para a direita) o número por 2.

Shifting i bits to the right $\Leftrightarrow Q(i, j) = P(i, j) \div 2^i$

Shifting i bits to the left $\Leftrightarrow Q(i, j) = P(i, j) \times 2^i$

Operações Lógicas: SHIFT

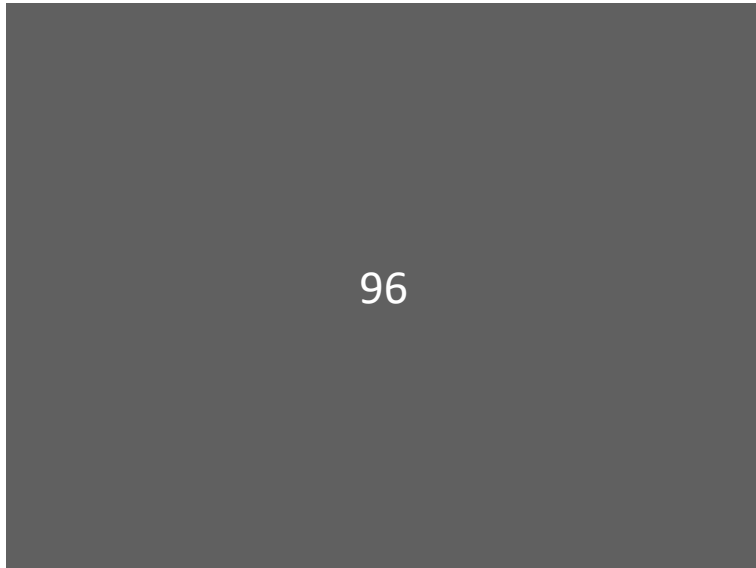
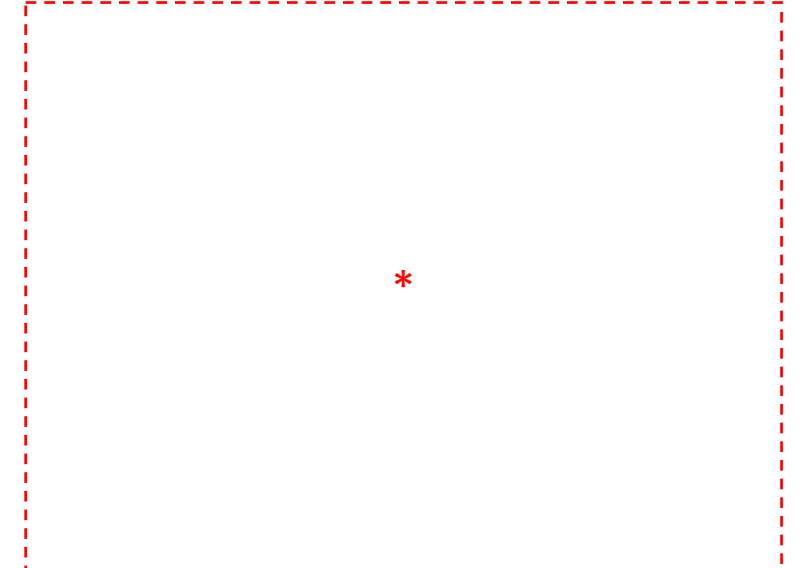


Imagem original



Deslocamento de 1 bit
para a esquerda



Deslocamento de 2 bits
para a esquerda (*overflow*)

Limites Inferior e Superior nas Operações

- Ao executar operações nas imagens, em alguns casos os resultados encontram-se fora dos limites de tom da imagem.
- Estes são os chamados *underflow* e *overflow* do resultado.
- Nestes casos, há duas alternativas:
 - Truncar os valores que excedem os limites, transformando em 255 tudo que exceda 255 e em 0 qualquer valor negativo (ou o contrário, para enfatizar).
 - Prever a possibilidade da representação de números negativos e maiores que 255 na memória e, após a realização da operação, proceder a uma mudança de escala (reescalonamento) dos valores.

Referências

- Image Processing Learning Resources. Robert Fisher, Simon Perkins, Ashley Walker, Erik Wolfart. Disponível em <http://homepages.inf.ed.ac.uk/rbf/HIPR2/arthops.htm>
- AZEVEDO, Eduardo; CONCI, Aura; LETA, F. Computação Gráfica-Teoria e Prática—3ª edição. **Rio de Janeiro: Elsevier, 2003.**