Lab3 – Prog. Assembly e AAPCS

- **Objetivo:**

Elaborar uma rotina em assembly que será chamada de um programa em C++. A rotina deve gerar um histograma de uma imagem em tons de cinza.

Lab3 – Prog. Assembly e AAPCS

Um projeto bem elaborado deve seguir uma sequencia de atividades:

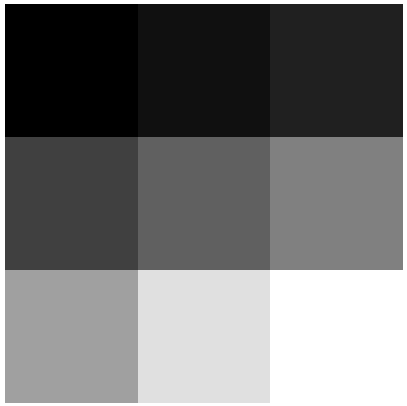
1. Planejamento das fases do processo de desenvolvimento.
2. Definição do problema a ser resolvido.
3. Especificação da solução.
4. Estudo da plataforma de HW (placa Tiva e seu processador).
5. Projeto (design) da solução:
    1. Planejamento das estruturas de dados;
    2. Forma de passagem de parâmetros;
    3. Algoritmo e alocação de variáveis aos registradores.
6. Configuração do projeto na IDE (IAR).
7. Edição do código da solução.
8. Teste e depuração.
9. Entrega dos resultados (sincronizar seu código com o seu GitHub).
   Acrescente na pasta Lab3, um documento em pdf relatando os passos 3 e 5 deste slide. O formato deste relato é livre. Deve ter entre 1 e 5 páginas.

```
#define R3_i  R3
```

LAB 3 - Domínio do Problema

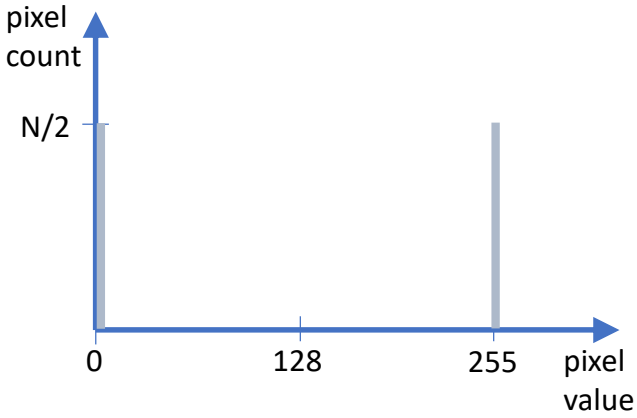
- Um bitmap é composto por pixels dispostos na forma de uma matriz. Numa imagem em tons de cinza, cada pixel é representado por um valor numérico indicando o nível de luminosidade daquele pixel.
- Numa imagem em tons de cinza de 8-bits, cada pixel é representado por um valor de 8-bits, portanto de 0 (preto) até 255 (branco).
- A imagem ao lado, de dimensão 3 x 3, contem os 9 pixels cujos valores estão apresentados.

0 16 32
64 96 128
160 224 255



LAB 3 - Domínio do Problema

- Um histograma é uma representação gráfica da distribuição de tons de uma imagem. O eixo horizontal apresenta os possíveis valores dos pixels (neste caso de 0 a 255) e o eixo vertical indica quantos pixels da imagem tem aquele valor.
- O histograma ao lado corresponde a uma imagem onde metade dos pixels são brancos e a outra metade são pretos.
- Histogramas são muito uteis em processamento de imagem para determinar valores de disparo (trigger), ajuste de brilho e contraste, valores limites em tomadas de decisão, etc.
- A construção de um histograma requer que todos os pixels da imagem sejam analisados; portanto, é interessante o uso de algoritmos otimizados para esta finalidade.



LAB 3 - Definição do Problema

- Desenvolva uma função em assembly que constrói o histograma de uma imagem em tons de cinza com 8-bits por pixel.
- Parâmetros de entrada:
- image width - número de pixels em uma linha da imagem.
- image height - altura da imagem em pixels.
- starting address - endereço do primeiro pixel da imagem.
- histogram - endereço inicial de um vetor de tamanho 256. Cada posição do vetor armazena um inteiro sem sinal de 16-bits. Este vetor não possui dados válidos quando a função é chamada. Ele é usado apenas para o retorno da função.
- Retorno: inteiro sem sinal de 16 bits indicando o número de pixels processados.

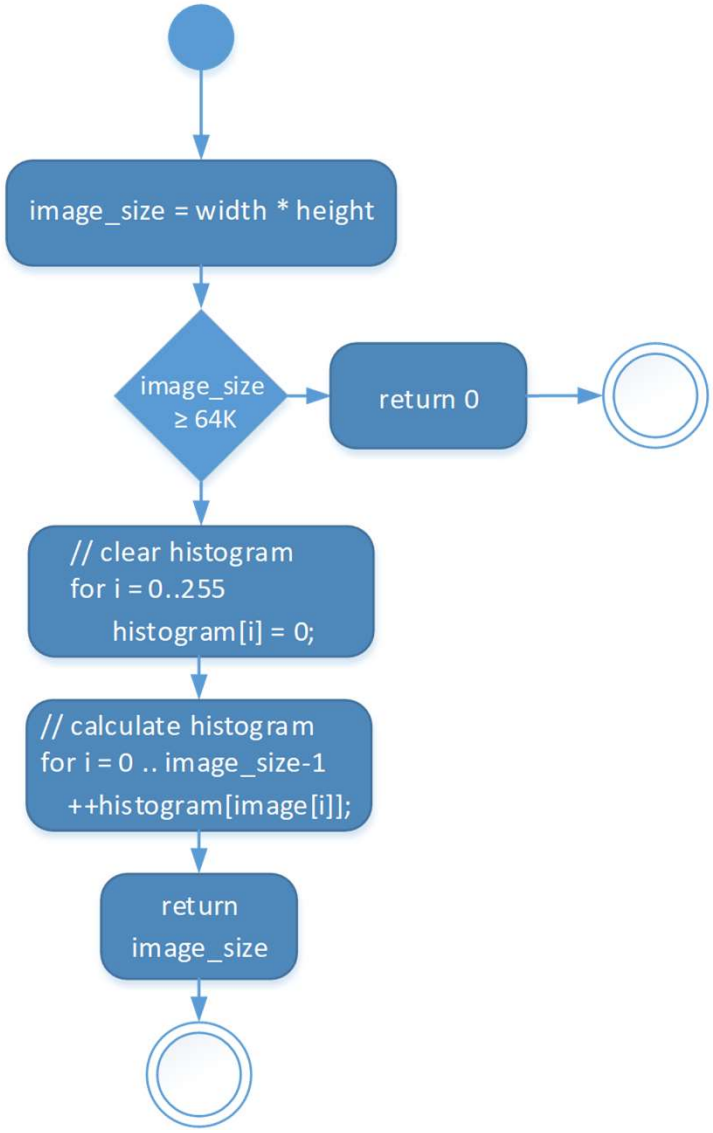
LAB 3 - Definição do Problema

- Restrições:
  O tamanho máximo da imagem é de 64K (65.536) pixels.
- Código de erro de retorno:
  a função retorna o valor 0 se o tamanho da imagem for superior a 64K.
- Declaração da função:

```
uint16_t EightBitHistogram(uint16_t width, uint16_t height, uint8_t * p_image, uint16_t * p_histogram);
```

LAB 3

- Sugestão de Algoritmo
- Uma possível solução está apresentada ao lado como um diagrama de atividades seguindo a notação da UML 2.5.
- Deve ser planejada a alocação das variáveis aos registradores do processador. Se for necessário usar registradores além de R0-R3 e R12 então, pelas regras da ATPCS, devem ser salvos na pilha e recuperados antes do retorno da função.
- Observe como este algoritmo é eficiente, lendo cada pixel uma única vez e usando seu valor para indexar um dos elementos do histograma.
- Apesar da imagem ser uma matriz, a leitura dos pixels é realizada como se fosse um vetor.



LAB 3

- Uma possível organização de arquivos no projeto
- main.cpp -
  este é o programa em C++ responsável por criar o vetor histograma e chamar a função **`EightBitHistogram.`**
  Este arquivo também deve criar os casos de teste e apresentar os resultados pelo Terminal.
- histogram.s - código fonte em assembly da função **`EightBitHistogram`**.
- images.c - uma imagem de teste.

LAB 3

- Possível cabeçalho para o arquivo em assembly:

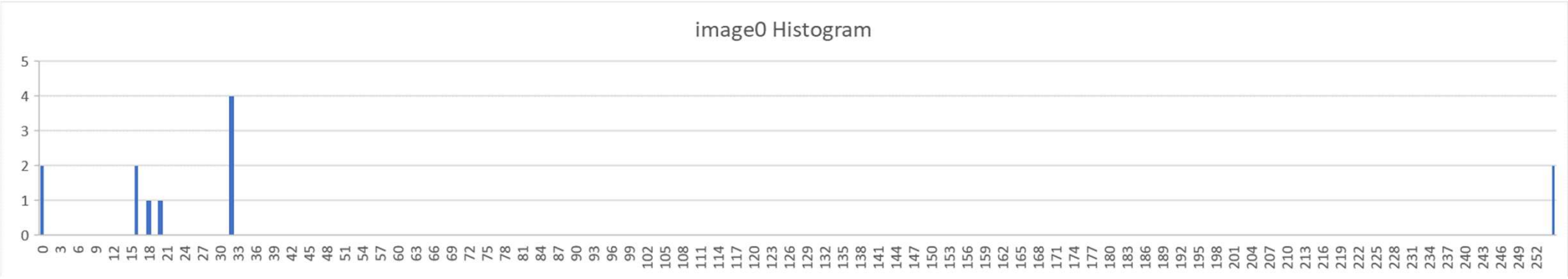
```
PUBLIC EightBitHistogram

SECTION .text : CODE (2)
THUMB
```

LAB 3 - Casos de Teste

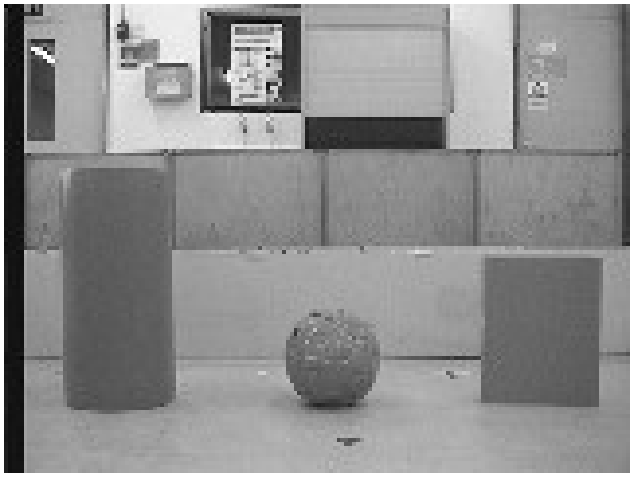
- Dois casos de teste devem ser utilizados:
- image0 (à direita) é uma imagem de apenas 3 linhas e 4 colunas e pode ser usada para analisar o comportamento da função executando-a passo-a-passo.
- O histograma correspondente está apresentado graficamente abaixo. Seu programa deve apresentar textualmente os valores do histograma no Terminal. Utilize cout para display no Terminal.

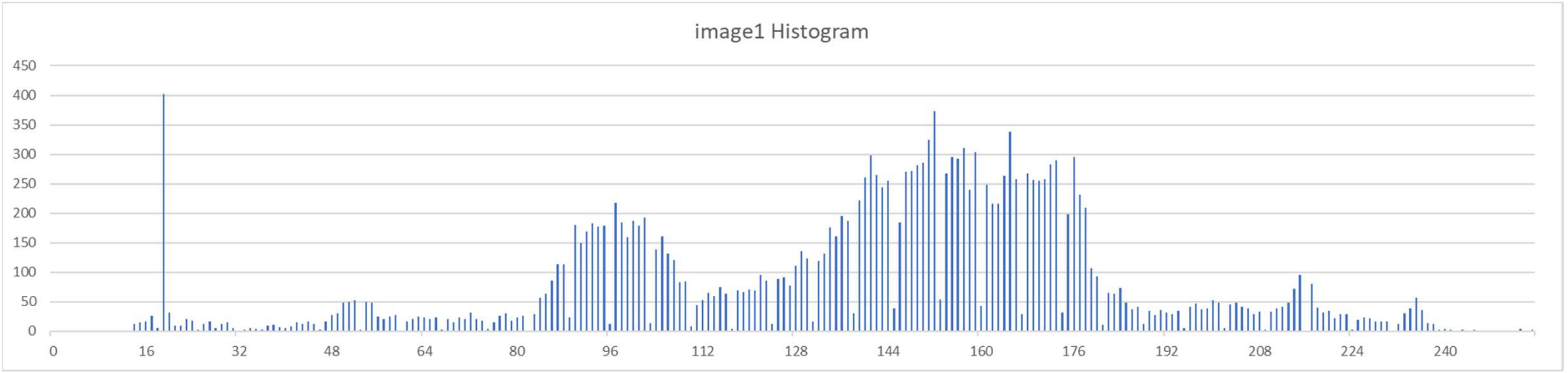
```
#define WIDTH0  4
#define HEIGTH0 3
const uint8_t image0[HEIGTH0][WIDTH0] = {
    { 20,  16, 16, 18},
    {255, 255,  0,  0},
    { 32,  32, 32, 32}
};
```



LAB 3 - Casos de Teste

- O segundo caso de teste é a imagem à direita, disponível no arquivo entregue no Lab3: images.c
- Esta imagem tem 160 x 120 pixels. Seu histograma está apresentado abaixo:





LAB 3 - Entregáveis

- Os arquivos da sua solução devem ser inseridos no seu github na pasta Lab3. No classroom, coloque o link para seu github.
- Inclua nesta pasta o arquivo pdf solicitado no segundo slide deste enunciado.
- Uma forma de validar sua função em assembly, é implementar a mesma função em C++ e comparar os resultados gerados por uma função e por outra. Não há obrigatoriedade de seguir esta sugestão.