Prof.: Paulo D. G. da Luz / Douglas P. B. Renaux

LABORATÓRIO 2 – ASSEMBLY, AAPCS E DESEMPENHO

Contexto:

Para aumentar o desempenho de um sistema embarcado, existem algumas técnicas, entre elas vale a pena ressaltar duas. A primeira é a criação de funções escritas em linguagem assembly e chamadas via código em "C/C++", esta técnica gera um ganho de desempenho desde que a função seja bem escrita. A segunda técnica é utilizarmos funções alocadas em memória RAM, esta técnica possibilita ganho de desempenho através da diferença de velocidade leitura entre as memórias FLASH e RAM. Assim este laboratório visa cobrir estas duas técnicas para melhorar o desempenho de um sistema embarcado.

Para auxiliar neste processo de criação da função em linguagem assembly consultar o documento do padrão **AAPCS** e o antigo documento do padrão **ATPCS:**

http://www.elf74.daeln.com.br/Pdfs/AAPCS.pdf

http://www.elf74.daeln.com.br/Pdfs/ATPCS.pdf

Como fonte de consulta o vídeo do prof. Denardin, que explica como rodar um código da **RAM**, detalhe do (++RAM *), bit **T** do PC:

https://youtu.be/KDCu9LzI2q0?si=iHO6K3NM3qT5AryZ

A primeira versão do programa deverá ser escrita em "C/C++". Ou seja, a função **EightBitHistogram_C** deverá ser construída para testar os cálculos do histograma. Assim como todo o restante do programa.
Após tudo testado e funcionando, partir para a codificação da função **EightBitHistogram_C** em sua versão assembly **EightBitHistogram_ASM**. Após testar tudo novamente, partir então para a alocação da função **EightBitHistogram_ASM** na memória RAM, completando as duas técnicas para gerar ganhos de desempenho em sistemas embarcados.

1 Elaborar uma rotina em "C/C++" e sua versão em assembly que será chamada de um programa em C/C++. A rotina deve gerar um histograma de uma imagem em tons de cinza. Desenvolver uma função em "C/C++" e sua versão 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.
- **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 das funções:

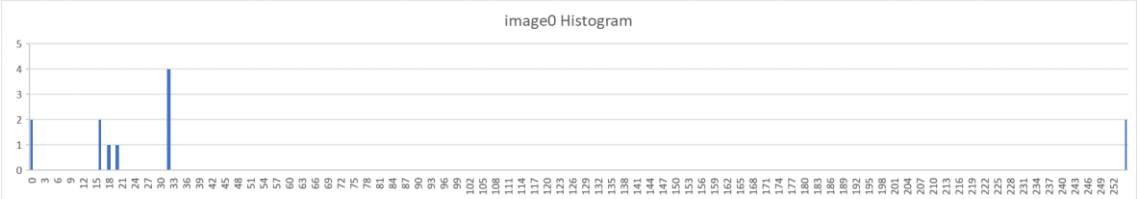
uint16_t EightBitHistogram_C(uint16_t width, uint16_t height, uint8_t * p_image, uint16_t * p_histogram);

uint16_t EightBitHistogram_ASM(uint16_t width, uint16_t height, uint8_t * p_image, uint16_t * p_histogram);

Dois casos de teste devem ser utilizados:

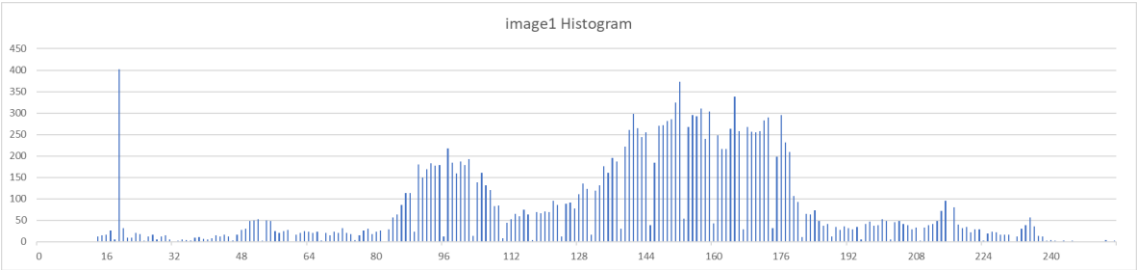
- image0, definida no vetor image0 abaixo é uma imagem de apenas 3 linhas e 4 colunas e pode ser usada para analisar o comportamento da função executando-a passo-a-passo.
- O histograma correspondente está apresentado graficamente abaixo. Seu programa deve apresentar textualmente os valores do histograma na **Uart0** / Serial em formato CSV para gerar o gráfico no Excel. Utilize o programa **sscom32.exe**.
- Deve-se obter como saída do programa um texto no formato (CSV: *comma - separated - values*);
- Verificar o arquivo exemplo para padronizar a saída de texto do programa: **exemplo.scv** / **exemplo.xlsx** no *link* **Labs** em **http://www.elf74.daeln.com.br**;

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



- O segundo caso de teste é uma imagem, disponível no arquivo: **images.c** no *link* **Labs** em **http://www.elf74.daeln.com.br**;

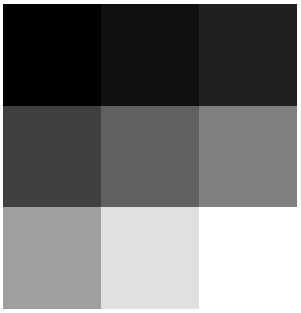
- Esta imagem tem 160 x 120 **pixels**:



Devem estar na pasta do projeto os dois gráficos dos histogramas em um arquivo do Excel chamado: histograma.xlsx.

Dicas:

- 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 exemplo, de dimensão 3 x 3, contém os 9 pixels cujos valores estão apresentados.



0 16 32

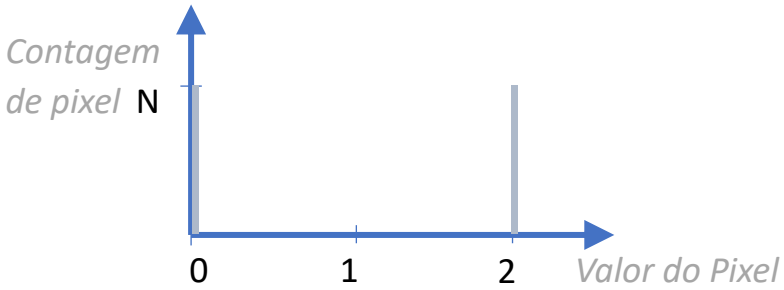
64 96 128

160 224 255

- Um histograma é uma representação gráfica da distribuição de tons de uma imagem. O eixo horizontal apresenta os possíveis valores dos *pixels* (neste

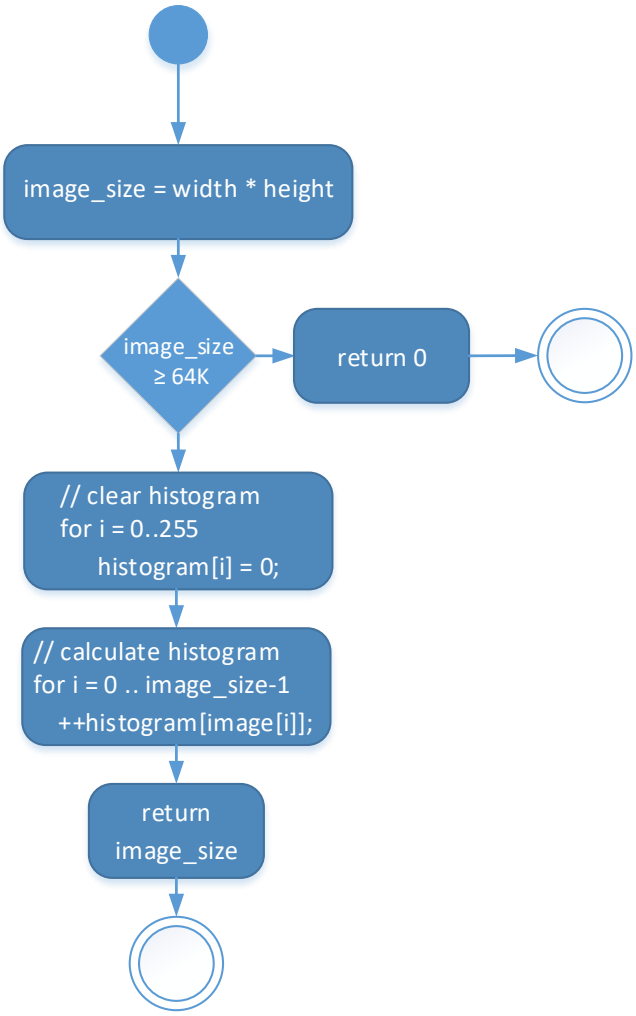
caso de 0 a 255) e o eixo vertical indica quantos *pixels* da imagem tem aquele valor.

- O histograma exemplo 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.



Sugestão de Algoritmo:

- Uma possível solução está apresentada 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 AAPCS, 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.



Conteúdo ao arquivo (readme.txt):

```
Disciplina: ELF74
Laboratório: 2
Equipe: Nome do aluno 1, RA
        Nome do aluno 2, RA
Data: XX/YY/ZZZZ
```

Configuração de Diretórios:

Manter os projetos de todos os laboratórios no mesmo nível de diretório que o diretório da "*TivaWare*", também manter o nome do diretório original: **"TivaWare_C_Series-2.2.0.295"**. Isto serve para que consiga compilar os exemplos do professor e vice-versa, quando entregar os laboratórios para avaliação.

Para entregar o Laboratório enviar um *email* para o professor com o diretório do projeto compactado em um arquivo .zip.

Email: garcez@professores.utfpr.edu.br

Título: **ELF74 – Laboratório 2**

Corpo do email:

Equipe: Aluno1, RA

Aluno2, RA

Anexo: **lab2.zip**

Composição da nota exclusivamente para este Laboratório:

1) Versão do Código 100% escrito em "C/C++": **90%**
2) Versão do Código escrito em "Assembly" e "C/C++": **+10%**
3) Versão do Código escrito em "Assembly" e "C/C++" com a função rodando em RAM: **+20%**