NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Arquitetura de Computadores 2023/24

TPC3

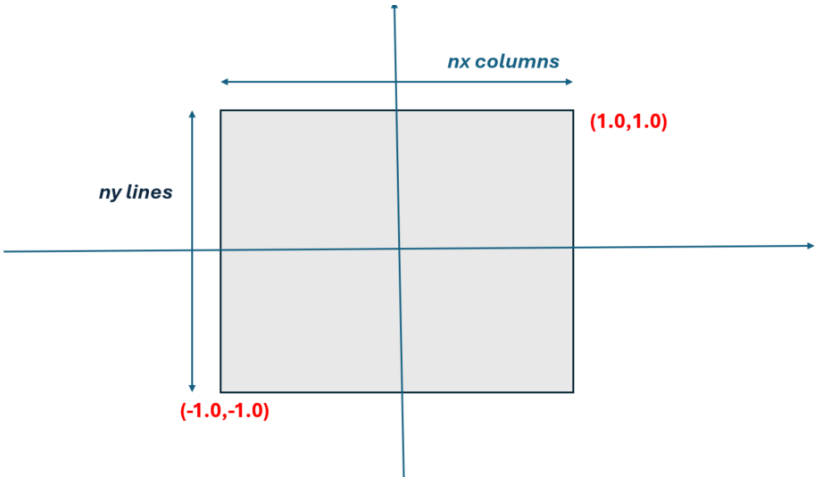
Este trabalho de casa consiste num exercício de programação **a ser realizado em grupo de no máximo dois alunos**. Pode esclarecer dúvidas gerais com colegas, mas a solução e a escrita do código devem ser estritamente realizadas pelos membros do grupo. Todas as resoluções serão comparadas de forma automática e os casos de plágio serão punidos de acordo com os regulamentos em vigor. **Caso use ferramentas como CoPilot ou o ChatGPT, deve incluir no código fonte um comentário a relatar esse uso.**

Data/ Hora limite para a entrega : dia 14/5 (3ª feira) às 10:00.

Conjunto de Mandelbrot

O conjunto de Mandelbrot tem aplicações em várias áreas. Na matemática é usado para estudar sistemas dinâmicos complexos e geometria de fractais. Na Física, é usado para implementar modelos destinados a estudar fenómenos naturais. Imagens derivadas do conjunto de Mandelbrot têm aplicações em arte. O conjunto também já foi usado em compressão de dados, criptografia e outras áreas da Ciência dos Computadores. O que é notável no conjunto de Mandelbrot é que regras muito simples podem produzir resultados infinitamente complicados.

Neste trabalho vamos usar uma combinação de um programa em C com um progrma em assembly x86-64 para produzir um conjunto de Mandelbrot correspondente a uma configuração que se descreve a seguir:



O conjunto de Mandelbrot em causa vai ser guardado numa matriz com *nx* colunas e *ny* linhas. Sendo as coordenadas do vértice inferior esquerdo (*xmin*, *ymin*) e do vértice superior direito (*xmax*, *ymax*) teremos

```
delta_x = (xmax-xmin) / nx
delta_y = (ymax-ymin) / ny
```

Cada um dos nx*ny pontos vai ter as seguintes coordenadas

```
x = xmin + delta_x * i
y = ymin + delta_y * j
```

O valor v correspondente ao ponto com coordenadas *(x,y)* é dado pelo seguinte algoritmo em que *x, y, zi, zr, nr, ni* são números reais (em precisão dupla, neste trabalho):

```
iterations = 0;
zi = 0;
zr = 0;
while ((zr*zr + zi*zi < 4.0) && (iterations < 255)) {
    nr = zr*zr - zi*zi + x;
    ni = 2*zr*zi + y;
    zi = ni;
    zr = nr;
    iterations++;
  }
  return iterations;
```

Programa que calcula o conjunto de Mandelbrot

O programa que calcula este conjunto de Mandelbrot é construído a partir de dois ficheiros fonte:

- *mandel.c* Este ficheiro está disponível no CLIP; está completo e não deve ser modificado. O seu comportamento pode ser descrito pelo seguinte pseudo código (ver o código para mais detalhes):

    1. obter as dimensões da imagem *nx* e *ny* e reservar um vetor de bytes *buffer* com dimensão correspondente a uma matriz com *ny* linhas e *nx* colunas
    2. para cada elemento *buffer[x,y]* da matriz
        a. calcular os números reais *x_value* e *y_value* correspondentes à posição coluna *x*, linha *y*
        b. invocar a função *computePoint* com argumentos *x_value* e *y_value* que retorna um valor inteiro *v* entre 0 e 255
        c. invocar a função *updateImage* que faz *buffer[x,y] = v*
    3. criar um ficheiro em formato PPM com uma imagem que permite visualizar o conteúdo da matriz.

    O programa é invocado com a linha de comando

    ```
    ./mandel <num-columns> <num-lines>
    ```

- *mandel_ASM.s* que deve ser criado de raiz e que define as duas funções *computePoint* e *updateImage*

Para gerar o executável mandel deve ser usado o seguinte *Makefile*

```
CC=gcc
CFLAGS=-g -z noexecstack
ASFLAGS=-gstabs
all: mandel
mandel: mandel.c mandel_ASM.o
	$(CC) $(CFLAGS) -o mandel mandel.c  mandel_ASM.o
mandel_ASM.o: mandel_ASM.s
	as $(ASFLAGS) -o mandel_ASM.o mandel_ASM.s
clean:
	rm -f *.o mandel *~
```

mandel_ASM.s

O ficheiro mandel_ASM.s deverá ter a seguinte estrutura

```
.globl updateImage  #  void updateImage( buffer, x,     y,  val, base)
.globl computePoint #  unsigned char computePoint( double x, double y)
.section .note.GNU-stack,"",@progbits
.text
updateImage:

…
    retq

computePoint:

…
    retq
```

As funções a implementar correspondem às assinaturas existentes no ficheiro *mandel.c*

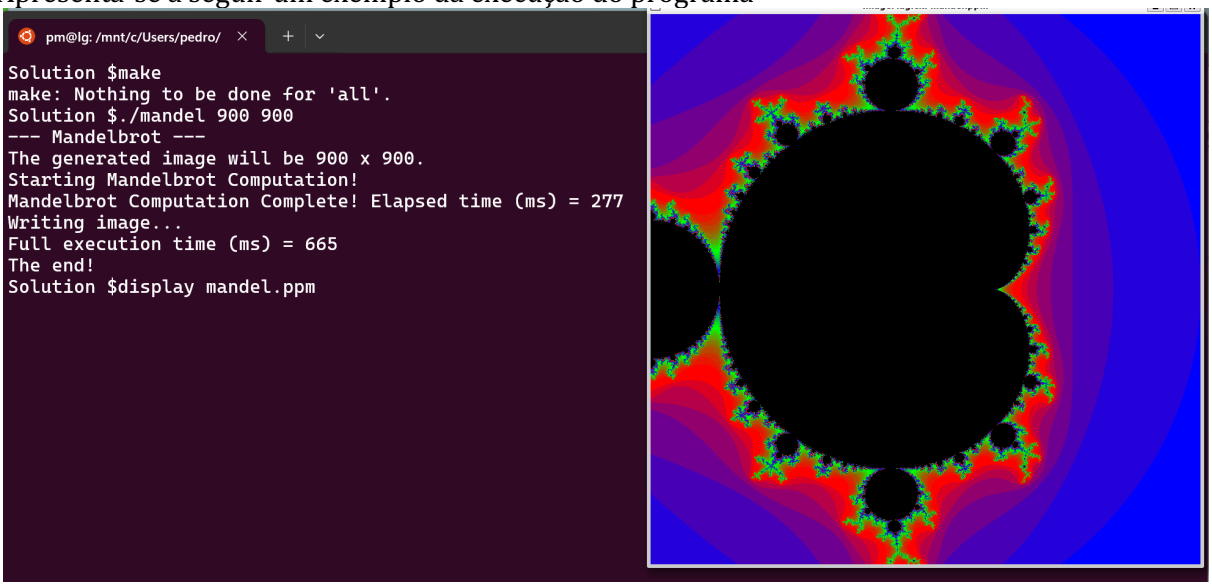
```
unsigned char computePoint( double cr, double ci );
void updateImage( unsigned char *buffer, unsigned int x,
       unsigned int y, unsigned char val, unsigned int base );
```

Na função *computePoint* deve utilizar as instruções máquina e os registos SSE mencionados nas aulas teóricas e descritos no documento *CS:APP2e Web Aside ASM:SSE: SSE-Based Support for Floating Point* de Randal E. Bryant e David R. O'Hallaron disponível em http://csapp.cs.cmu.edu/2e/waside/waside-sse.pdf

A função *output_ppm* produz um ficheiro no formato PPM. Para informação sobre este formato de imagem consultar https://en.wikipedia.org/wiki/Netpbm. Para a imagem ser gerada é preciso que esteja disponível o ficheiro *color.map*.
No Linux, para visualizar a imagem pode ser usado o conjunto de programas disponível se instalarmos o package *imagemagick* que pode ser instalado com o comando `sudo apt install imagemagick`
A visualização do ficheiro pode ser feita a partir da interface gráfica ou a partir do *bash* com

```
display mandel.ppm
```

Apresenta-se a seguir um exemplo da execução do programa



NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Modo de entrega

O ficheiro com as funções em assembly x86-64 deve ter o nome *mandel_ASM.s* Esse ficheiro deve ser incluído numa mensagem de email a enviar para o email do mestre **Pedro Camponês (p.campones@campus.fct.unl.pt)**
O assunto (Subject) da mensagem deve ser AC2024-TPC3 estudantes XXXXX e YYYYY.
XXXXX é o número de estudante do 1º autor e YYYYY é o número do 2º autor. Se o grupo só tiver um elemento YYYYY deve ser 00000.