

# **Universidade de Aveiro**

**Departamento de Electrónica, Telecomunicações e Informática**

**Informação e Codificação (2021/22)**

**Projeto #02**

**Entropy coding of audio and images**

Equipa:

João Moraes - 93288

Pedro Coutinho - 93278

Repositório:

[github.com/antoniojoao10/Project2IC](https://github.com/antoniojoao10/Project2IC)



universidade  
de aveiro

# Parte A

## Golomb (Ex3 e Ex4)

### Comandos:

```
$ ./Golomb
```

### Descrição da implementação:

#### The Golomb encoder

1.  $k \leftarrow \lceil \log_2(m) \rceil$ .
2.  $r \leftarrow s \bmod m$ .
3.  $t \leftarrow 2^k - m$ .
4. Output  $(s \div m)$  using an unary code.
5. If  $r < t$ :
  - a. Output the integer encoded in the  $k - 1$  least significant bits of  $r$  using a binary code.
6. Else:
  - a.  $r \leftarrow r + t$ .
  - b. Output the integer encoded in the  $k$  least significant bits of  $r$  using a binary code.

#### The Golomb decoder

1.  $k \leftarrow \lceil \log_2(m) \rceil$ .
2.  $t \leftarrow 2^k - m$ .
3. Let  $s \leftarrow$  the number of consecutive ones in the input (we stop when we read a 0).
4. Let  $x \leftarrow$  the next  $k - 1$  bits in the input.
5. If  $x < t$ :
  - a.  $s \leftarrow s \times m + x$ .
6. Else:
  - a.  $x \leftarrow x \times 2 +$  next input bit.
  - b.  $s \leftarrow s \times m + x - t$ .

### Referencia

[https://w3.ual.es/~vruiz/Docencia/Apuntes/Coding/Text/03-symbol\\_encoding/09-Golomb\\_coding/index.html](https://w3.ual.es/~vruiz/Docencia/Apuntes/Coding/Text/03-symbol_encoding/09-Golomb_coding/index.html)

### Demonstração:

```
joao@joao-Predator-PH315-52:~/Documents/UA/IC/Project2IC$ ./Golomb 2 15
111111101
15
```

# Parte B

## Lossless

### Comandos:

#### Encode:

```
$ g++ -o main ExBComp.cpp -lsndfile
```

```
$ ./main <input.wav> <output.bin>
```

#### Decode:

```
$ g++ -o main ExBDecomp.cpp -lsndfile
```

```
$ ./main <output.wav> <input.bin>
```

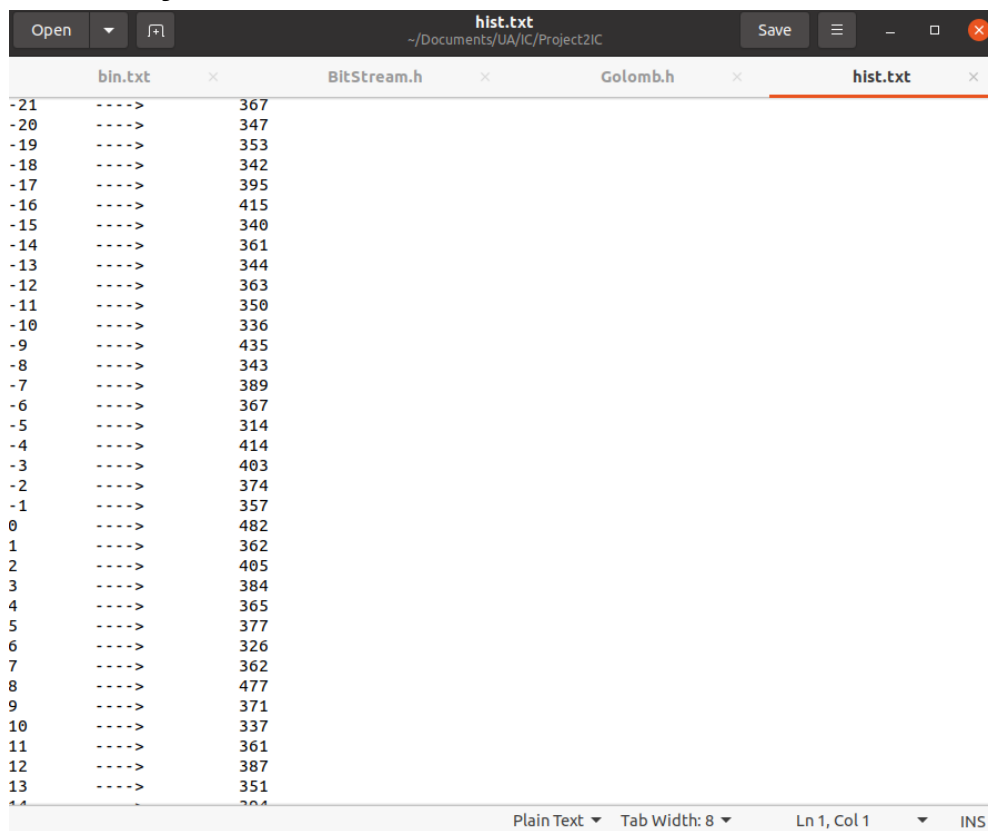
### Descrição da implementação:

No início, o ficheiro é lido através da library libsndfile. Com os dados do áudio num vetor, é usado o preditor (simple 1D polynomial predictor ) e o residual é calculado. É também calculado um histograma com os residuais.

Os residuais são depois computados com o Golomb encoder e escrito num ficheiro bin.

No final o ficheiro comprimido sai muito grande porque não fomos capazes de fazer um bitstream funcional. Sendo que o problema deve estar presente na escrita do ficheiro.

### Demonstração:



The screenshot shows a text editor window with the title bar 'hist.txt' and the path '~/Documents/UA/IC/Project2IC'. The editor contains a histogram of residuals from bin.txt. The data is as follows:

Residual	Count
-21	367
-20	347
-19	353
-18	342
-17	395
-16	415
-15	340
-14	361
-13	344
-12	363
-11	350
-10	336
-9	435
-8	343
-7	389
-6	367
-5	314
-4	414
-3	403
-2	374
-1	357
0	482
1	362
2	405
3	384
4	365
5	377
6	326
7	362
8	477
9	371
10	337
11	361
12	387
13	351
14	364

The editor status bar at the bottom shows 'Plain Text', 'Tab Width: 8', 'Ln 1, Col 1', and 'INS'.

## Lossy

### Comandos:

#### Encode:

```
$ g++ -o main ExBCompLossy.cpp -lsndfile  
$ ./main <input.wav> <output.bin>
```

#### Decode:

```
$ g++ -o main ExBDecomp.cpp -lsndfile  
$ ./main <output.wav> <input.bin>
```

### Descrição da implementação:

Igual ao lossless, mas antes de enviar para o Golomb encoder é feita quantização do valor residual. De modo a reduzir o número de bits, cada valor do array teve os seus primeiros 12 bits a 0. Ou seja, para cada valor é dado, primeiro um *shift right* de 12 bits, e depois um *shift left* de 12 bits. Assim é reduzida a qualidade de som do ficheiro wav de entrada.

# Parte C

## Lossless

### Comandos:

```
$ g++ ExC1.cpp -o testoutput -std=c++11 `pkg-config --cflags --libs opencv`  
$ ./testoutput <input.ppm>
```

### Descrição da implementação:

Após o ficheiro ser lido, o mesmo vai ser modificado para o formato YUV 4:2:0. A partir deste novo ficheiro os canais Y, U e V são separados e codificados em diferentes ficheiros utilizando código baseado em JPEG-LS. Infelizmente, como não foi desenvolvido um BitStream completamente funcional, não é possível decodificar os mesmos com sucesso, no entanto o código de descodificação está presente. Caso a descodificação fosse um sucesso, o ficheiro ia ser novamente modificado para BGR e guardado numa nova imagem intitulada de "new\_image.ppm"

## Lossy

### Comandos:

```
$ g++ ExC2.cpp -o testoutput -std=c++11 `pkg-config --cflags --libs opencv`  
$ ./testoutput <input.ppm>
```

### Descrição da implementação:

Semelhante ao lossless, no entanto é realizado um shift à direita do número de bits indicados a todos os canais da imagem original e depois um shift à esquerda do número de bits indicados a todos os canais da imagem final.