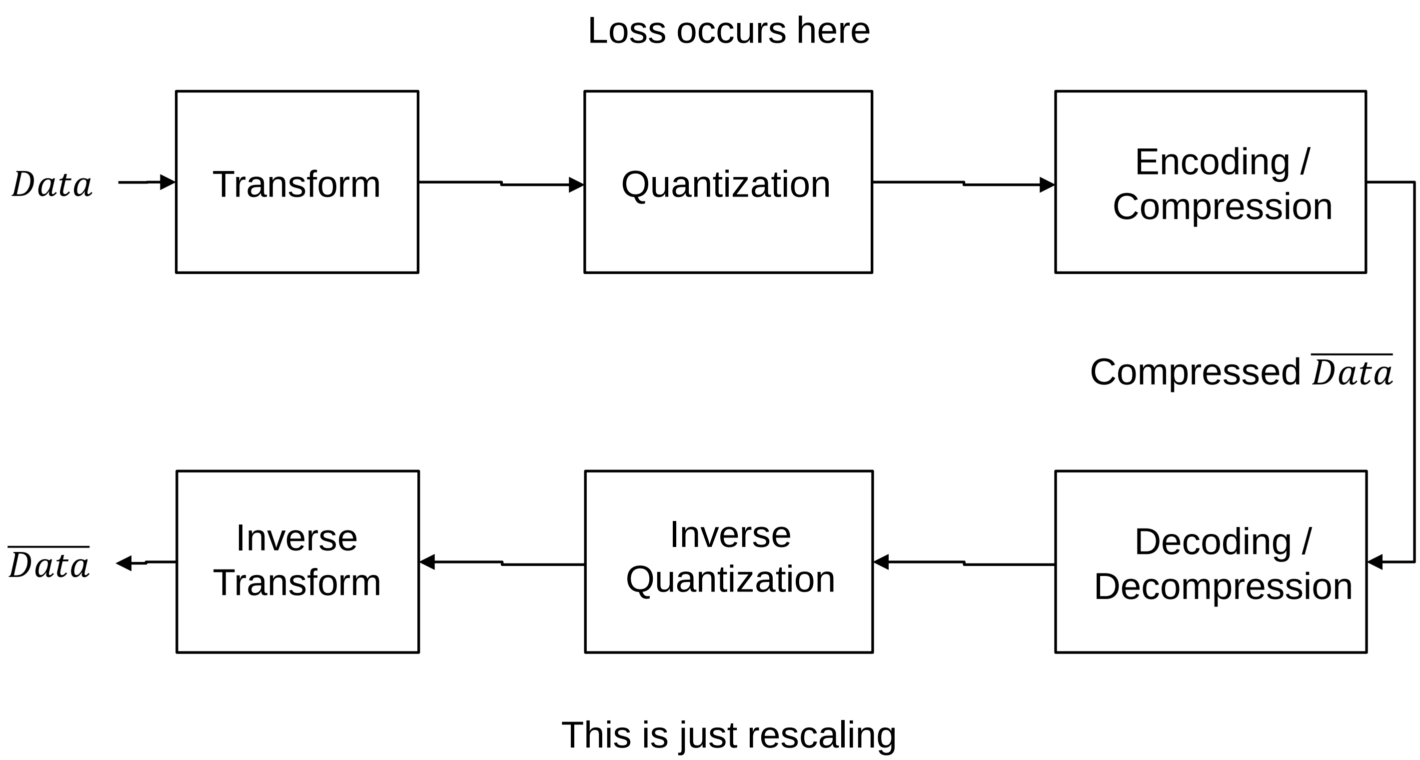
JPEG

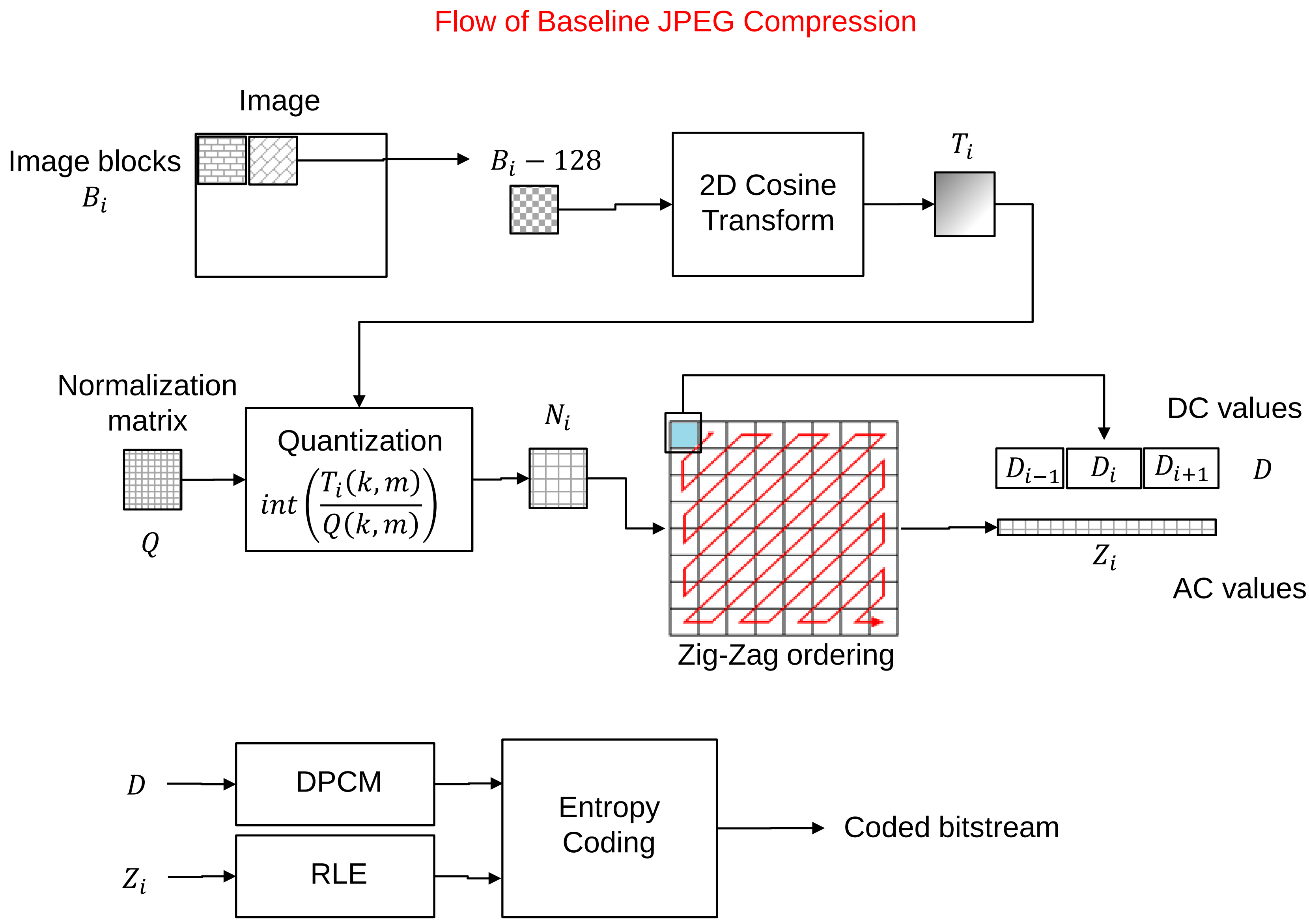
by Erol Seke

For the course "Data Compression"

ESKİŞEHİR OSMANGAZİ UNIVERSITY

General Flow of Transform Coding

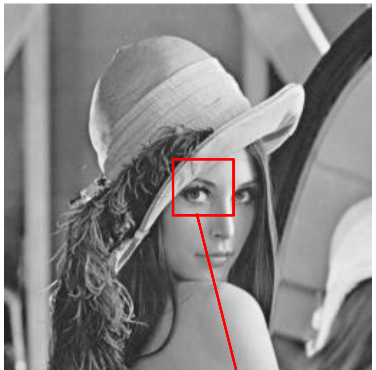




Flow of Baseline JPEG Compression
Image
Image blocks
B_i
$B_i - 128$
2D Cosine Transform
T_i
Normalization matrix
Q
Quantization
$int\left(\frac{T_i(k,m)}{Q(k,m)}\right)$
N_i
DC values
D_{i-1} D_i D_{i+1}
D
Z_i
AC values
Zig-Zag ordering
D
DPCM
Z_i
RLE
Entropy Coding
Coded bitstream

Transform of Blocks

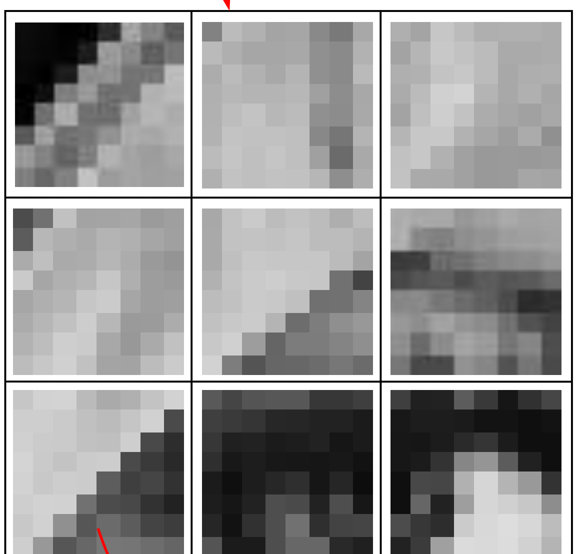
$$y_k = \sum_{n=0}^{N-1} x_n cos\left(\left(n+\frac{1}{2}\right)k\,\pi/N\right)$$

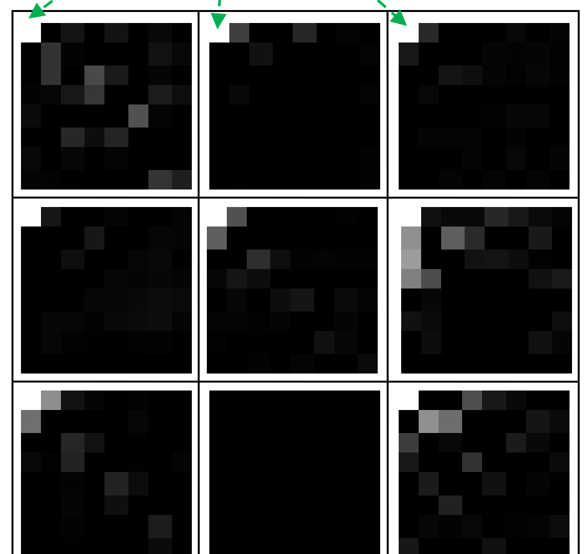


- 128

Consecutive DC components will be DPCM coded because they are large and neighboring DCs are similar

8x8 blocks





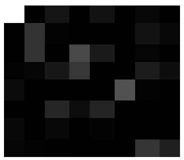
$B_i - 128$

2D Cosine Transform

T_i

$$y_{k,l} = \sum_{m=0}^{M-1} \alpha(m)\left(\sum_{n=0}^{N-1} \alpha(n) x_{m,n} cos\big((2n+1)k\,\pi/2\,N\big)\right) cos\big((2m+1)l\,\pi/2\,M\big)$$

Quantization

 =

344,12	85,935	-27,117	-60,316	51,375	-67,483	5,4144	-0,9025
-51,894	-20,105	24,176	-8,2341	-7,1179	-3,3003	-7,7824	6,6240
-8,8309	-9,0513	-5,9634	-0,4184	-10,926	-7,4485	2,2045	-13,435
-15,807	12,277	-8,7681	-24,146	1,1885	-5,0784	-8,8987	5,5736
-3,6250	-10,469	-17,855	0,4644	-21,875	-0,5659	-0,3162	-2,1043
-4,7567	1,3709	-11,813	-3,8462	-5,8313	-7,7358	-2,0873	-6,2885
-2,7012	-7,8226	-3,7955	-17,866	-3,1863	-13,250	-5,7866	2,7547
-0,7145	1,8562	-1,4233	-7,1339	-5,2303	-5,1786	-2,0782	4,4869

$$N_i(k,m) = int\left(\frac{T_i(k,m)}{Q(k,m)}\right)$$

note: This table is for luminance.
Chrominance table is different

$N_i(k,m)$

21	7	-2	-3	2	-1	0	0
-3	-1	1	0	0	0	0	0
0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$Q(k,m)$

16	11	10	16	24	40	51	61
13	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

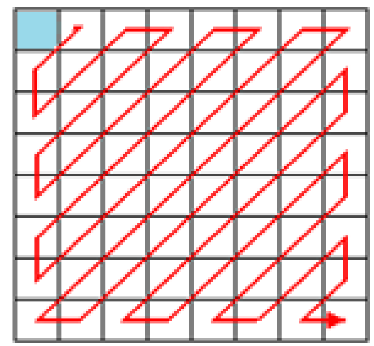
Example normalization matrix

Zig-Zag Ordering

Human eye is less sensitive to high frequency components and therefore these components are quantized coarsely. Low frequency coefficients are generally larger after the quantization. Most HF coefficients are truncated to zero. In order to make them consecutive, a zig-zag ordering is applied.

$N_i(k,m)$

21	7	-2	-3	2	-1	0	0
-3	-1	1	0	0	0	0	0
0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



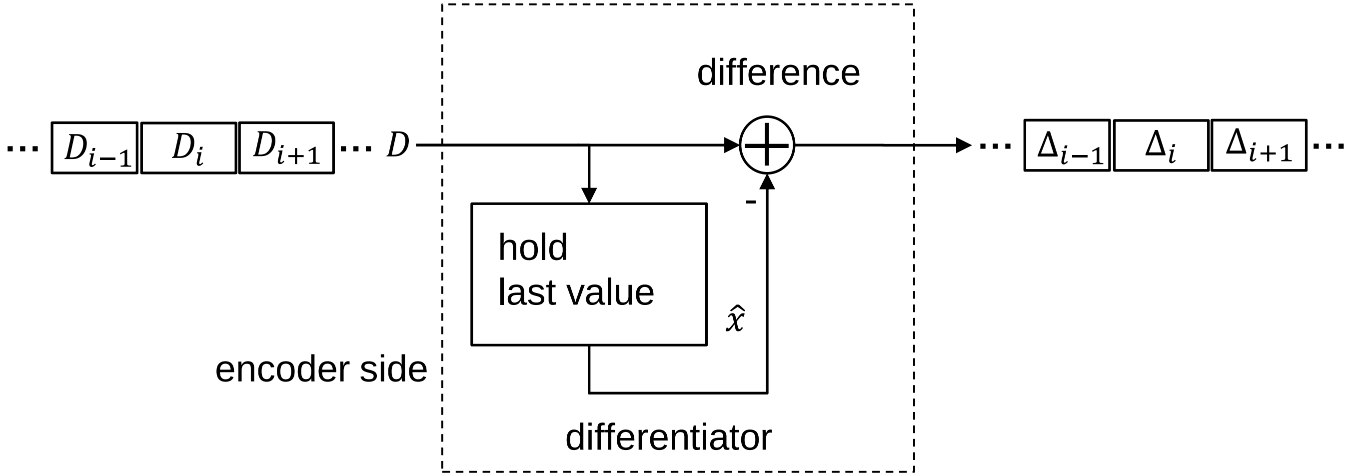
D_i Z_i

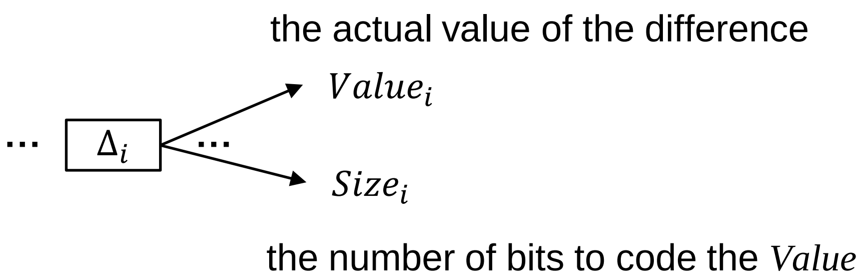
{21, 7, -3, 0, -1, -2, -3, 1, 0, -1, 0, 0, 0, 0, 2, -1, EOB} Serialized coefficients

DC components are DPCM coded

AC components are RLE coded

DPCM on DC Values





Value range	Size
0	0
−1, 1	1
−3, −2, 2, 3	2
−7, ..., −4, 4, ..., 7	3
−15, ..., −8, 8, ..., 15	4
−31, ..., −16, 16, ..., 31	5
−63, ..., −32, 32, ..., 63	6
−127, ..., −64, 64, ..., 127	7
−255, ..., −128, 128, ..., 255	8
−511, ..., −256, 256, ..., 511	9
−1023, ..., −512, 512, ..., 1023	A
−2047, ..., −1024, 1024, ..., 2047	B
−4095, ..., −2048, 2048, ..., 4095	C
−8191, ..., −4096, 4096, ..., 8191	D
−16383, ..., −8192, 8192, ..., 16383	E
−32767, ..., −16384, 16384, ..., 32767	F

Value in range → code

–A, …, -B, B, …, A → 0, …, 2^{Size}-1

Example Difference Value : -14

-14 belongs to the {-15, -14, -13, -12, -11, -10, -9, -8, 8, 9, 10, 11, 12, 13, 14, 15} range

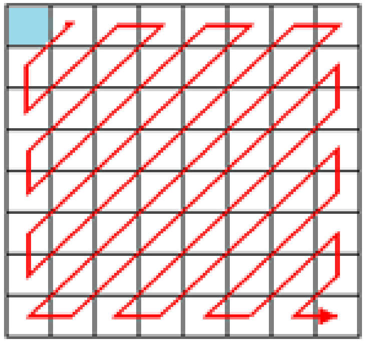
which are coded as {0000, 0001, 0010, 0011, …, 1101, 1110, 1111} with *Size*=4

Size is coded using Huffman-codes given in the table

Size	Code
0	00
1	010
2	011
3	100
=> 4	101
5	110
6	1110
7	11110
8	111110
9	1111110
10	11111110
11	111111110

Therefore, {SizeCode, ValueCode} output pair is {101, 0001}
These binary codes are sent to file/bitstream.

RLE on AC components



Z_i

$\{dc, Z_{0,1}, Z_{1,0}, Z_{2,0}, Z_{1,1}, Z_{0,2}, \ldots, \mathrm{EOB}\}$ Serialized coefficients

D_i

(we have seen how to code DC sequence)

AC components are RLE coded

$\{\ldots, a, 0, 0, \ldots, 0, 0, b, \ldots , \mathrm{EOB}\}$

number of bits required to code value b is Nbits

Nz : number of zeros to the next nonzero values 0-15

([Nz/Nbits], b) pairs are constructed

Value	Nbits
0	0
−1, 1	1
−3, −2, 2, 3	2
−7, …, −4, 4, …, 7	3
−15, …, −8, 8, …, 15	4
−31, …, −16, 16, …, 31	5
−63, …, −32, 32, …, 63	6
−127, …, −64, 64, …, 127	7
−255, …, −128, 128, …, 255	8
−511, …, −256, 256, …, 511	9
−1023, …, −512, 512, …, 1023	A
−2047, …, −1024, 1024, …, 2047	B
−4095, …, −2048, 2048, …, 4095	C
−8191, …, −4096, 4096, …, 8191	D
−16383, …, −8192, 8192, …, 16383	E
−32767, …, −16384, 16384, …, 32767	F

Value in range → code

$-A, \ldots, -B, B, \ldots, A \rightarrow 0, \ldots, 2^{Size}-1$

Find value-code for the b value using this (similar to the DC value codes)

Find Nbits from this table using b value

Huffman table for Nz/Nbits

Nz/Nbits	Code	
0/0	**1010 (=EOB)**	**4**
0/1	00	3
0/2	01	4
0/3	100	6
0/4	1011	8
0/5	11010	10
0/6	111000	12
0/7	1111000	14
0/8	1111110110	18
0/9	1111111110000010	25
0/A	1111111110000011	26
1/1	1100	5
1/2	111001	8
1/3	1111001	10
1/4	111110110	13
1/5	11111110110	16
1/6	1111111110000100	22
1/7	1111111110000101	23
1/8	1111111110000110	24
1/9	1111111110000111	25
1/A	1111111110001000	26
2/1	11011	6
2/2	11111000	10
2/3	1111110111	13
2/4	1111111110001001	20
2/5	1111111110001010	21
2/6	1111111110001011	22
2/7	1111111110001100	23

Nz/Nbits	Code	
8/1	11111010	9
8/2	111111111000000	17
8/3	1111111110110111	19
8/4	1111111110111000	20
8/5	1111111110111001	21
8/6	1111111110111010	22
8/7	1111111110111011	23
8/8	1111111110111100	24
8/9	1111111110111101	25
8/A	1111111110111110	26
9/1	111111000	10
9/2	1111111110111111	18
9/3	1111111111000000	19
9/4	1111111111000001	20
9/5	1111111111000010	21
9/6	1111111111000011	22
9/7	1111111111000100	23
9/8	1111111111000101	24
9/9	1111111111000110	25
9/A	1111111111000111	26
A/1	111111001	10
A/2	1111111111001000	18
A/3	1111111111001001	19
A/4	1111111111001010	20
A/5	1111111111001011	21
A/6	1111111111001100	22
A/7	1111111111001101	23

Nz/Nbits	Code	
2/8	1111111110001101	24
2/9	1111111110001110	25
2/A	1111111110001111	26
3/1	111010	7
3/2	111110111	11
3/3	11111110111	14
3/4	1111111110010000	20
3/5	1111111110010001	21
3/6	1111111110010010	22
3/7	1111111110010011	23
3/8	1111111110010100	24
3/9	1111111110010101	25
3/A	1111111110010110	26
4/1	111011	7
4/2	1111111000	12
4/3	1111111110010111	19
4/4	1111111110011000	20
4/5	1111111110011001	21
4/6	1111111110011010	22
4/7	1111111110011011	23
4/8	1111111110011100	24
4/9	1111111110011101	25
4/A	1111111110011110	26

Nz/Nbits	Code	
A/8	1111111111001110	24
A/9	1111111111001111	25
A/A	1111111111010000	26
B/1	111111010	10
B/2	1111111111010001	18
B/3	1111111111010010	19
B/4	1111111111010011	20
B/5	1111111111010100	21
B/6	1111111111010101	22
B/7	1111111111010110	23
B/8	1111111111010111	24
B/9	1111111111011000	25
B/A	1111111111011001	26
C/1	1111111010	11
C/2	1111111111011010	18
C/3	1111111111011011	19
C/4	1111111111011100	20
C/5	1111111111011101	21
C/6	1111111111011110	22
C/7	1111111111011111	23
C/8	1111111111100000	24
C/9	1111111111100001	25
C/A	1111111111100010	26

Find code for Nz/Nbits from this table (Nz is the number of zeros before the b value)

Output Nz/Nbits code followed by b-value code

Example Code for AC stream { 7, -3, 0, -1, -2, -3, 1, 0, -1, 0, 0, 0, 0, 2, -1, EOB}

#bits to code value

#zeros=0

nonzero value following zeros

(0 zeros, 7) → (0/3)7 → 100111 (100: code for 0/3, 111: code for 7 (from value-size table))

(0 zeros, -3) → (0/2)-3 → 0100 (01: code for 0/2, 00: code for -3 (from value-size table))

(1 zeros, -1) → (1/1)-1 → 11000

(0 zeros, -2) → (0/2)-2 → 0101

(0 zeros, -3) → (0/2)-3 → 0100

(0 zeros, 1) → (0/1)1 → 001

(1 zeros, -1) → (1/1)-1 → 11000

(4 zeros, 2) → (4/2)2 → 1111111000010

(0 zeros, -1) → (0/1)-1 → 000

EOB→ (0/0) → 1010 (special code to indicate that the remaining values are all zeros)

END