Data Compression

Lossless	Lossy
run-length	CS&Q
Huffman	JPEG
delta	MPEG
LZW	

a. Lossless or Lossy

Method	Group size: input	output
CS&Q	fixed	fixed
Huffman	fixed	variable
Arithmetic	variable	variable
run-length, LZW	variable	fixed

b. Fixed or variable group size

Lossless: Retrieved data is identical to original data.

Lossy: Retrieved data is NOT identical to original data, BUT no significant data is lost.

Example: An image file of 600 kbytes of data.

Without compression: *TIFF* → 600 kbyte

Lossless compression: *GIF* → 300 kbyte

Lossy compression: *JPEG* → 50 kbyte

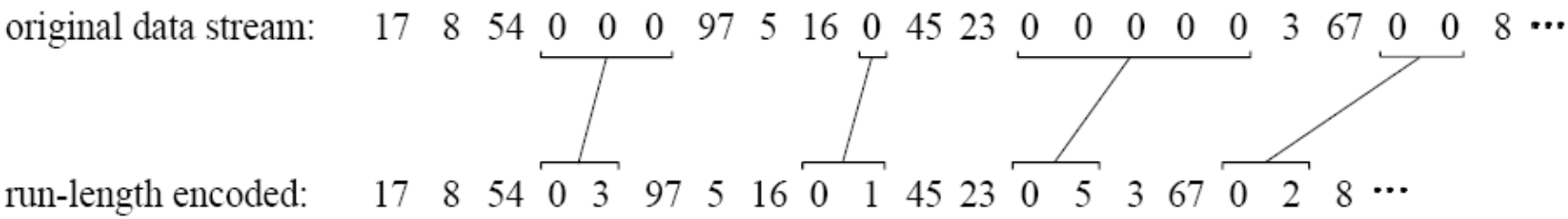
Run-Length Encoding

Concept: Data files often contain same characters repeated in a row.

Text files: multiple spaces to separate sentences.

Digitized signals: run the same value over some time (unchanged signal).

Image: night sky, same black background for long area.



Pack Bits:

Input: 8 bit (0 to 255); Output: 9 bit (-255 to 255)

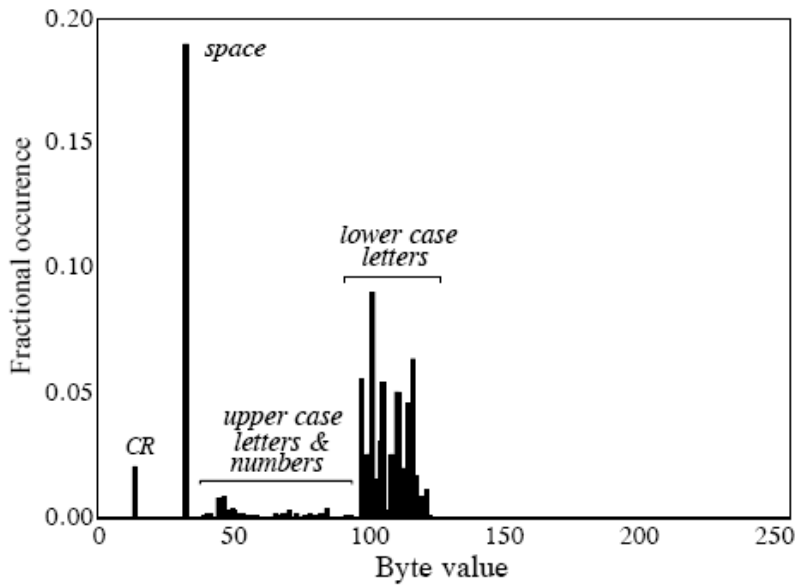
Input Stream: 1, 2, 3, 4, 2, 2, 2, 2, 4

Output Stream: 1, 2, 3, 4, 2, -3, 4

Normally ASCII characters are from 0 to 127; so the 8th bit can be used for (-).

Huffman Encoding - I

Concept: A general ASCII file content:



96%+ : lower case letters, space, comma, carriage return, and period (31 characters).

5 bits.

(00000 - 01111)

4%- : special characters.

11111

(Indicate special char.)

Next 8 bits indicate which of the special char.

Total: 5+8 = 13 bits

Size reduction: 5/8

Frequently used characters: fewer bits.
Seldom used characters: more bits.

Average number of bits required:

$0.96 \times 5 + 0.04 \times 13 = 5.32$.

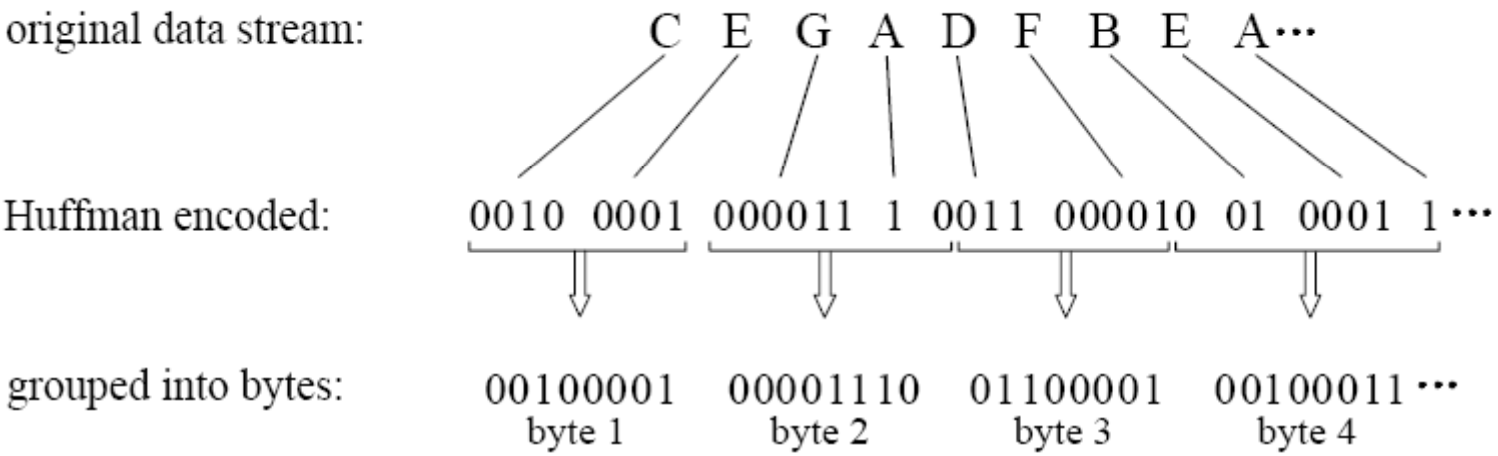
Compression ratio: 8 / 5.32 = 1.5 : 1

Huffman Encoding - II

Higher frequencies (probabilities) → fewer bits.

Example Encoding Table

letter	probability	Huffman code
A	.154	1
B	.110	01
C	.072	0010
D	.063	0011
E	.059	0001
F	.015	000010
G	.011	000011



Decoding: When a '1' is found, check the sequence until '1' in the table. If a match is not found, include the next bit for look up.

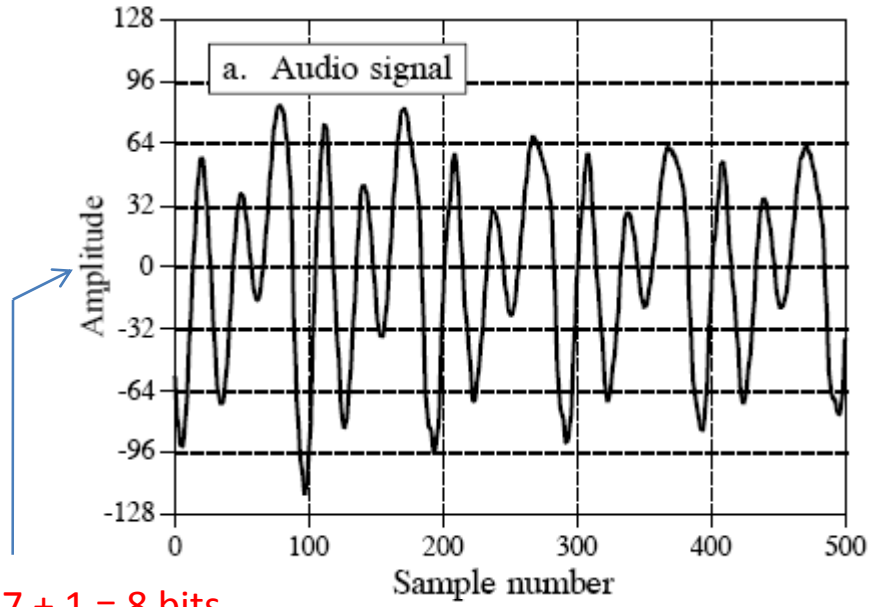
Delta Encoding

original data stream: 17 19 24 24 24 21 15 10 89 95 96 96 96 95 94 94 95 93 90 87 86 86 ...

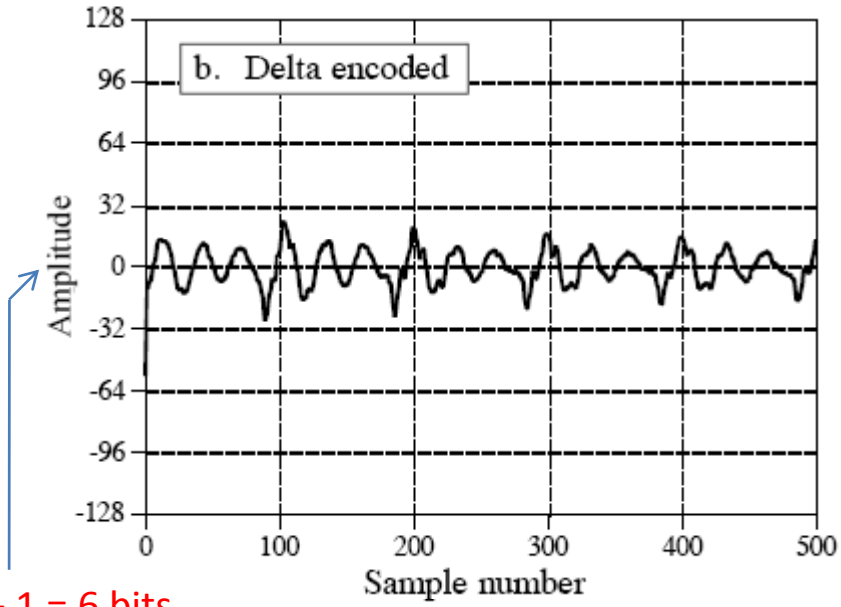
move delta delta delta ...

delta encoded: 17 2 5 0 0 -3 -6 -5 79 6 1 0 0 -1 -1 0 1 -2 -3 -3 -1 0 ...

Can be encoded in fewer bits



7 + 1 = 8 bits.



5 + 1 = 6 bits.

Useful if the difference between adjacent samples is low.

LZW Encoding

Encoding:

this_is_his_thing

New Byte	Encoded String	New Code	Code Output
t	t	*None*	*None*
h	h	256 (th)	t
i	i	257 (hi)	h
s	s	258 (is)	i
_	_	259 (s_)	s
i	i	260 (_i)	_
s	is	*None*	*None*
_	_	261 (is_)	258 (is)
h	h	262 (_h)	_
i	hi	*None*	*None*
s	s	263 (his)	257 (hi)
_	s_	*None*	*None*
t	t	264 (s_t)	259 (s_)
h	th	*None*	*None*
i	i	265 (thi)	256 (th)
n	n	266 (in)	i
g	g	267 (ng)	n
None	*None*	*None*	g

Decoding:

't' 'h' 'i' 's' '_' 258 '_' 257 259 256 'i' 'n' 'g'

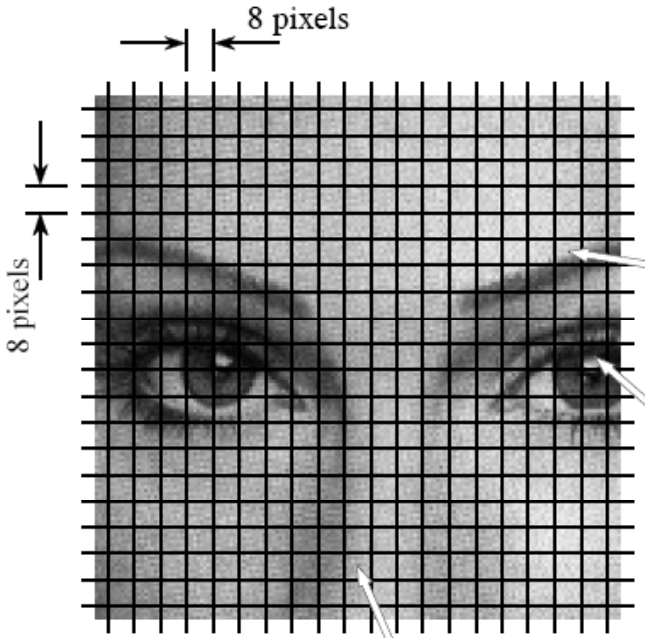
Input Code	Encoded String	Added Code	String Output
t	t	*None*	t
h	h	256 (th)	h
i	i	257 (hi)	i
s	s	258 (is)	s
_	_	259 (s_)	_
258	is	260 (_i)	is
_	_	261 (is_)	_
257	hi	262 (_h)	hi
259	s_	263 (his)	s_
256	th	264 (s_t)	th
i	i	265 (thi)	i
n	n	266 (in)	n
g	g	267 (ng)	g

17 character string is represented by 13 codes.

JPEG - I

Joint Photographers Experts Group

Break the image into 8×8 groups, each containing 64 pixels.



231	224	224	217	217	203	189	196
210	217	203	189	203	224	217	224
196	217	210	224	203	203	196	189
210	203	196	203	182	203	182	189
203	224	203	217	196	175	154	140
182	189	168	161	154	126	119	112
175	154	126	105	140	105	119	84
154	98	105	98	105	63	112	84

42	28	35	28	42	49	35	42
49	49	35	28	35	35	35	42
42	21	21	28	42	35	42	28
21	35	35	42	42	28	28	14
56	70	77	84	91	28	28	21
70	126	133	147	161	91	35	14
126	203	189	182	175	175	35	21
49	189	245	210	182	84	21	35

High number represents white (bits turned into ‘1’).

154	154	175	182	189	168	217	175
154	147	168	154	168	168	196	175
175	154	203	175	189	182	196	182
175	168	168	168	140	175	168	203
133	168	154	196	175	189	203	154
168	161	161	168	154	154	189	189
147	161	175	182	189	175	217	175
175	175	203	175	189	175	175	182

JPEG tries to remove high frequency components (redundant).

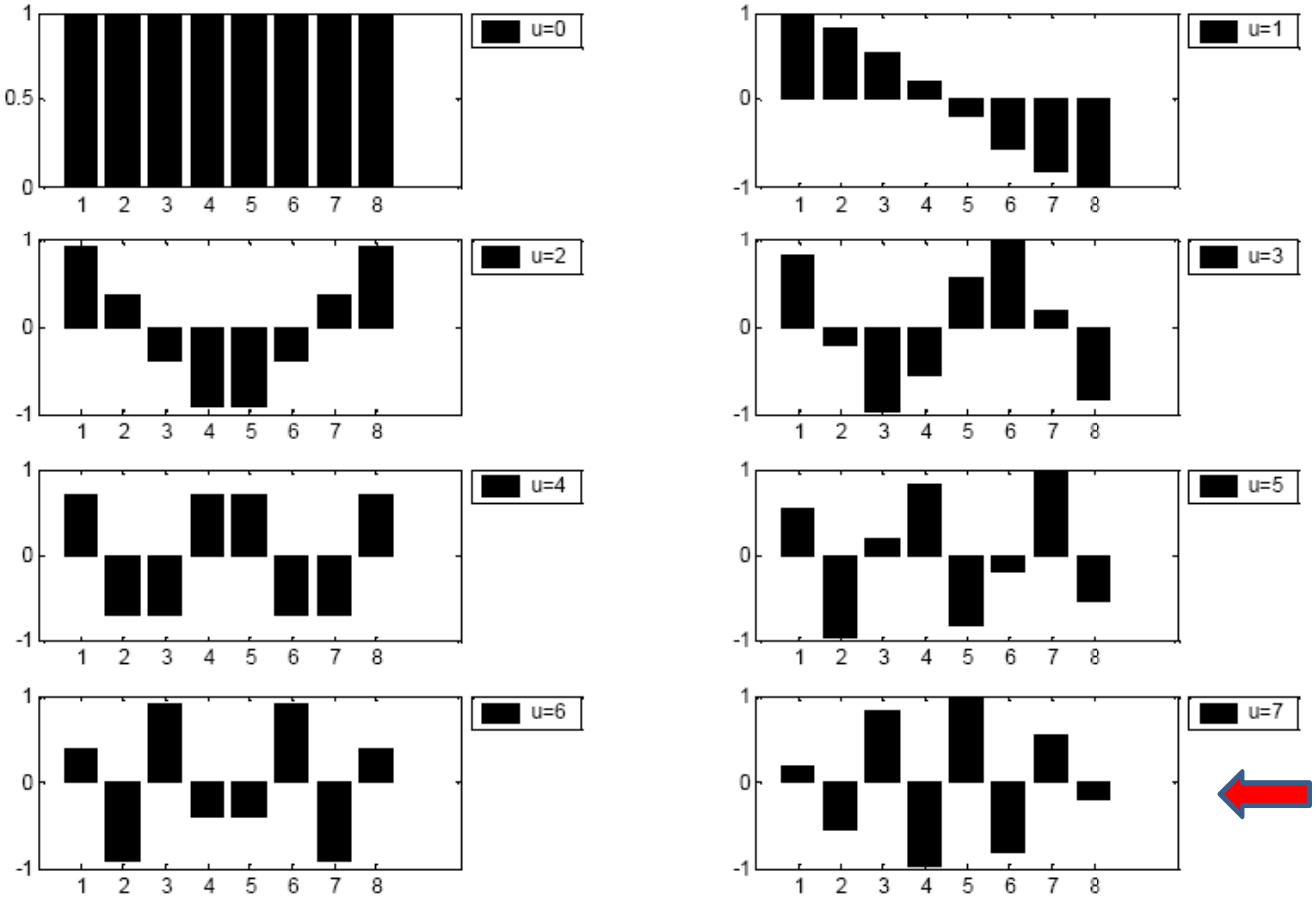
Discrete Cosine Transform (DCT)

One dimensional:

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos\left[\frac{\pi(2x+1)u}{2N}\right]$$

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad u = 0 \\ \sqrt{\frac{2}{N}} & for \quad u \neq 0. \end{cases}$$

DC



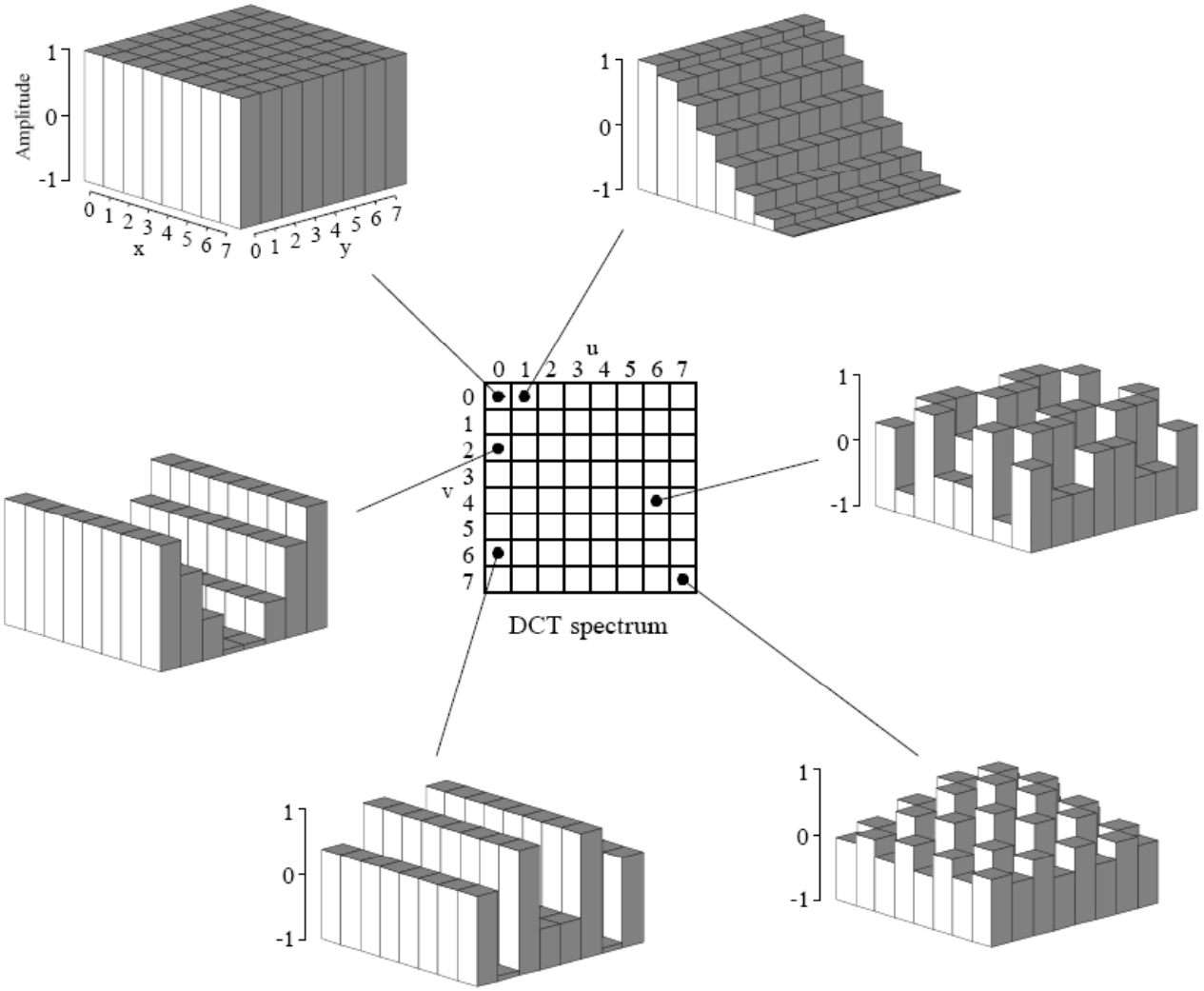
High Freq.

JPEG - II

Using DCT

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right],$$

DC



DCT basis functions:

JPEG - III

Bit truncation

Original Group

a. Eyebrow

231	224	224	217	217	203	189	196
210	217	203	189	203	224	217	224
196	217	210	224	203	203	196	189
210	203	196	203	182	203	182	189
203	224	203	217	196	175	154	140
182	189	168	161	154	126	119	112
175	154	126	105	140	105	119	84
154	98	105	98	105	63	112	84

b. Eye

42	28	35	28	42	49	35	42
49	49	35	28	35	35	35	42
42	21	21	28	42	35	42	28
21	35	35	42	42	28	28	14
56	70	77	84	91	28	28	21
70	126	133	147	161	91	35	14
126	203	189	182	175	175	35	21
49	189	245	210	182	84	21	35

DCT Spectrum

d. Eyebrow spectrum

174	19	0	3	1	0	-3	1
52	-13	-3	-4	-4	-4	5	-8
-18	-4	8	3	3	2	0	9
5	12	-4	0	0	-5	-1	0
1	2	-2	-1	4	4	2	0
-1	2	1	3	0	0	1	1
-2	5	-5	-5	3	2	-1	-1
3	5	-7	0	0	0	-4	0

e. Eye spectrum

70	24	-28	-4	-2	-10	-1	0
-53	-35	43	13	7	13	1	3
23	9	-10	-8	-7	-6	5	-3
6	2	-2	8	2	-1	0	-1
-10	-2	-1	-12	2	1	-1	4
3	0	0	11	-4	-1	5	6
-3	-5	-5	-4	3	2	-3	5
3	0	4	5	1	2	1	0

Quantization Error

g. Using 10 bits

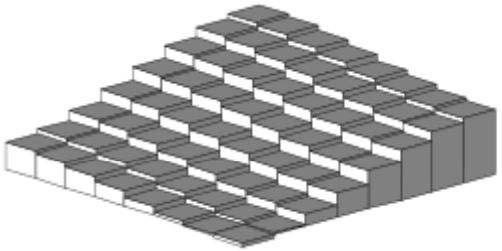
0	0	0	0	-1	0	0	0
-1	0	0	0	0	0	0	-1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

h. Using 8 bits

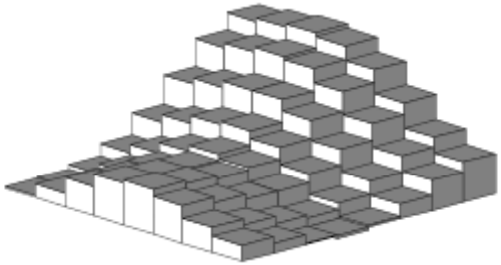
0	-3	-1	-1	1	0	0	-1
1	0	-1	-1	0	0	0	-1
-1	-2	1	0	-2	0	-2	-2
-1	-2	-1	2	0	2	0	1
0	-2	1	0	0	1	0	0
0	-4	-1	0	1	0	0	0
0	-2	0	1	-1	-1	1	-1
-1	-3	1	1	1	-3	-2	-1

JPEG - IV

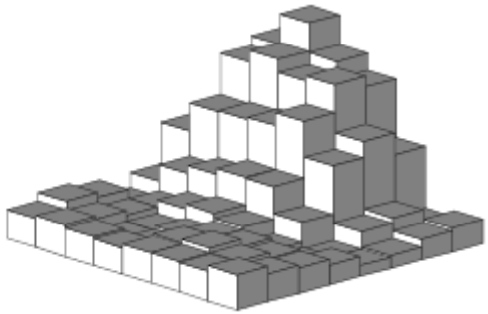
High frequency components removal



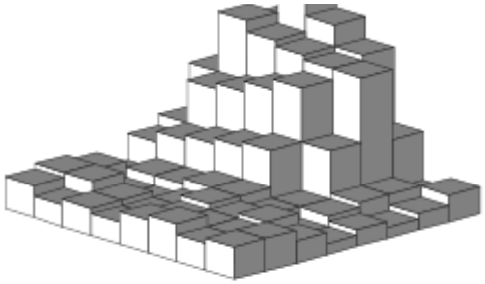
a. 3 coefficients



b. 6 coefficients



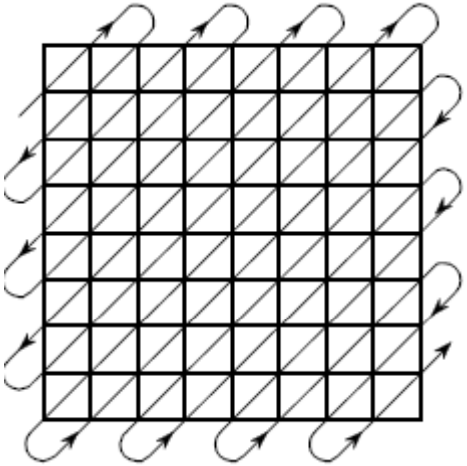
c. 15 coefficients



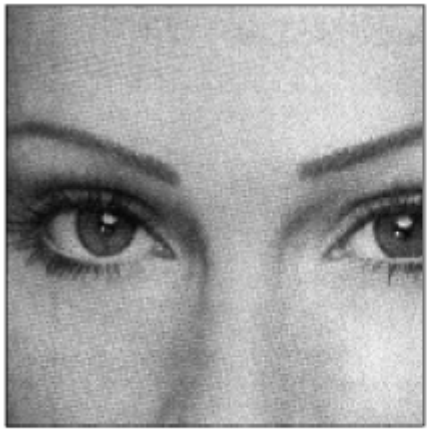
d. 64 coefficients
(correct image)

Steps:

1. Image is broken into 8×8 groups.
2. DCT is taken for each group.
3. Each 8×8 spectrum is compressed by bit truncating and high freq. components removal.
4. Modified spectrum is converted into a linear sequence (eliminated components will be zeros).
5. Compress zero by run-length encoding.
6. Sequence is coded using Huffman coding for example.



Example of JPEG Distortion



a. Original image

b. With 10:1 compression

c. With 45:1 compression

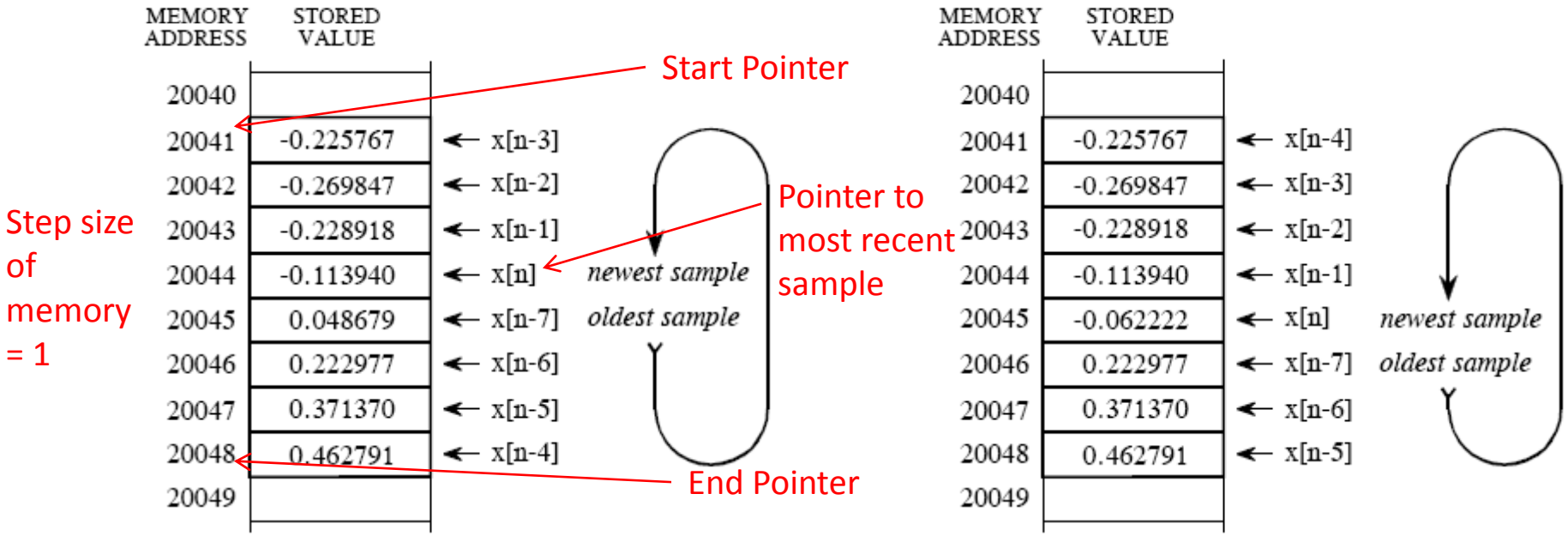
Digital Signal Processors - Introduction

Processors dedicated to DSP.

Off-line processing: all the data needs to be in the memory at the same time. The output is produced after all the input data is in the memory.

On-line processing: the output is produced as the same time the input is coming. No delay or little delay.

Circular Buffer Operation:



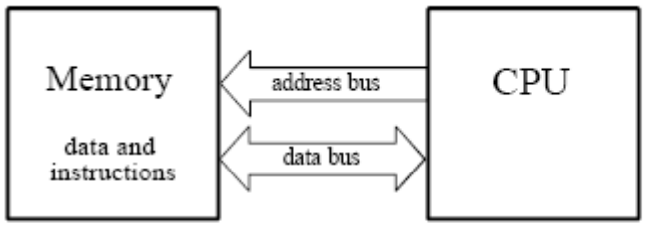
a. Circular buffer at some instant

b. Circular buffer after next sample

Microprocessors Architecture

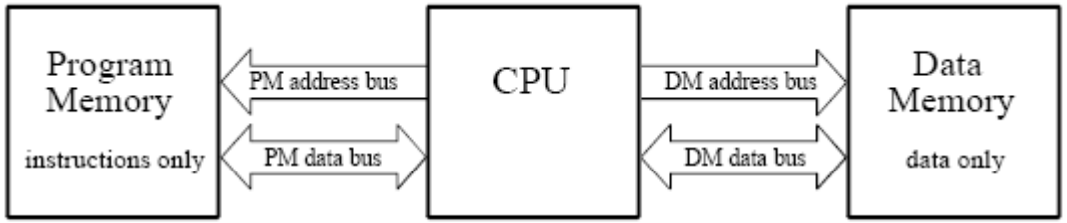
Normal microprocessors

a. Von Neumann Architecture (*single memory*)

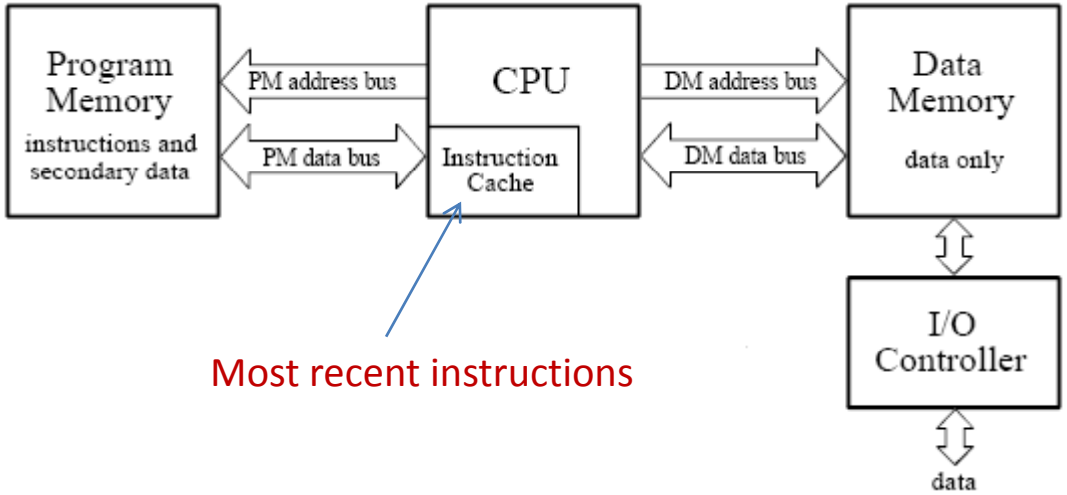


DSP uses these ideas

b. Harvard Architecture (*dual memory*)

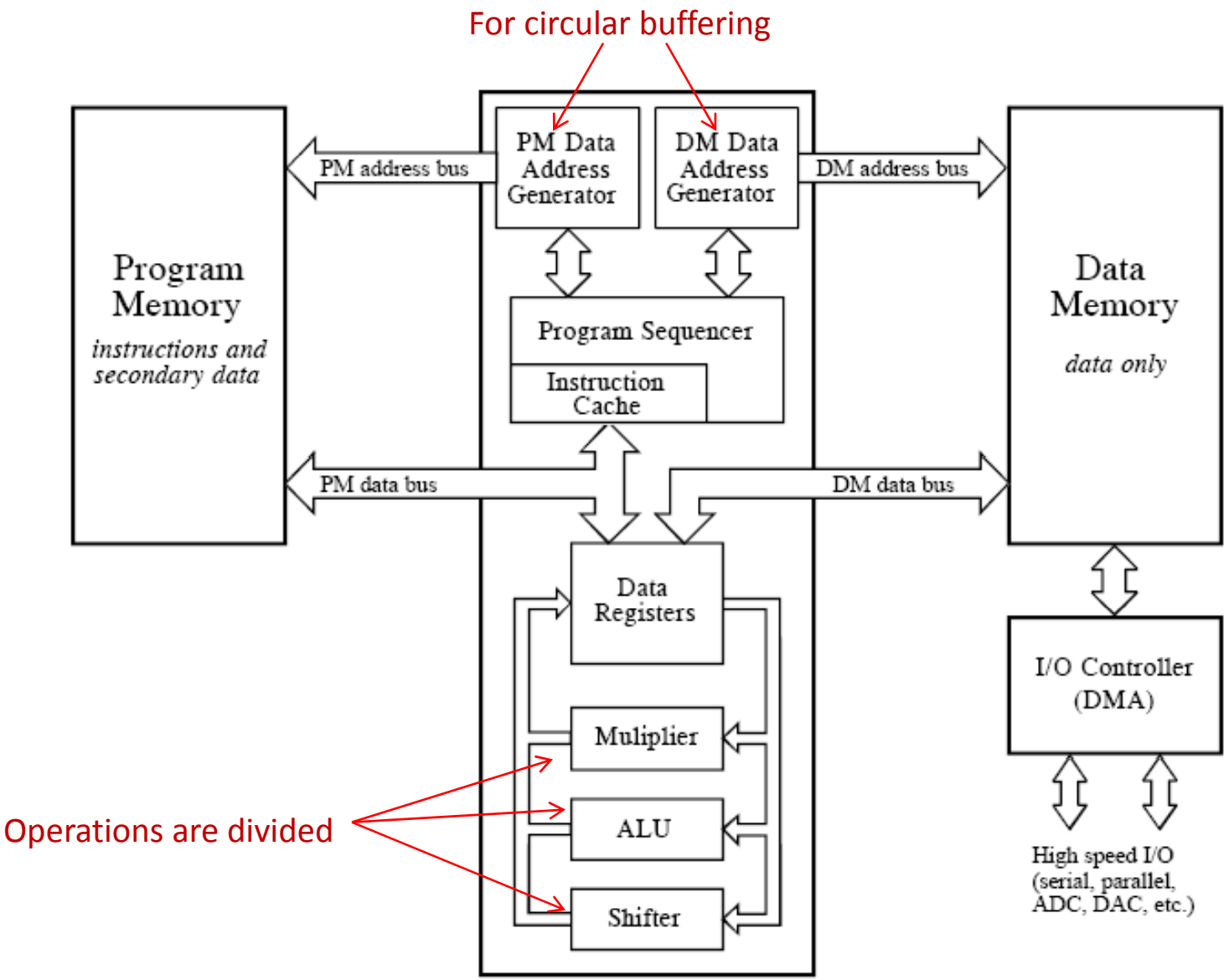


c. Super Harvard Architecture (*dual memory, instruction cache, I/O controller*)



Most recent instructions

Typical DSP Architecture

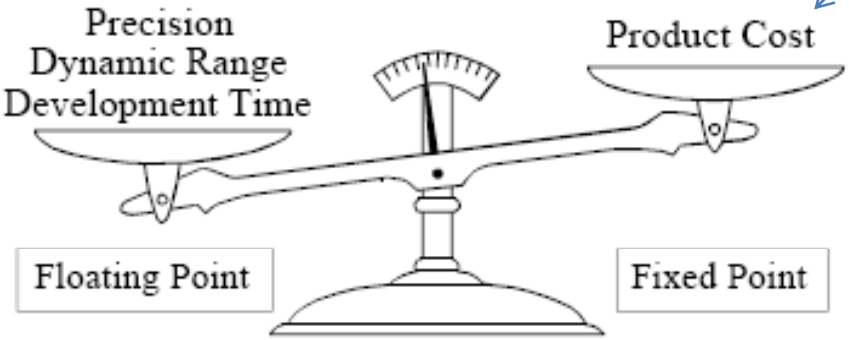


Fixed verses Floating Point

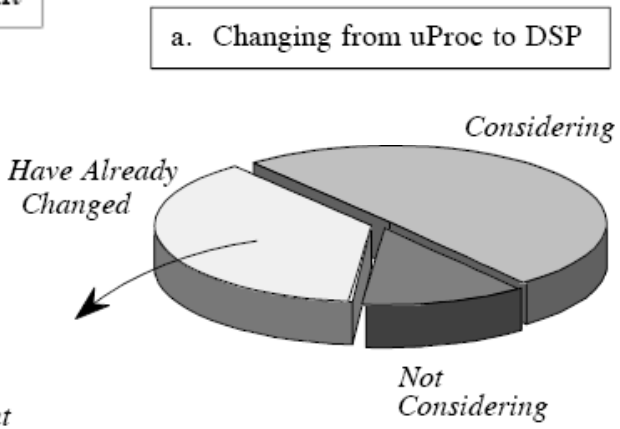
Fixed Point: Low number of bits (minimum 16).

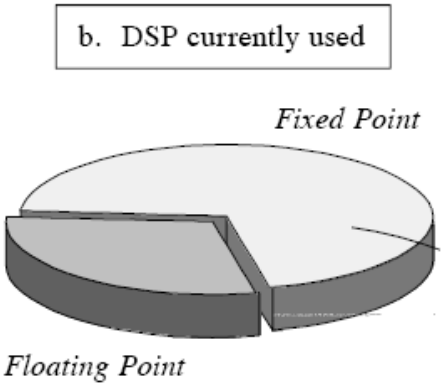
Floating Point: High number of bits (min 32).

All the data buses and registers are only 16 bit wide.



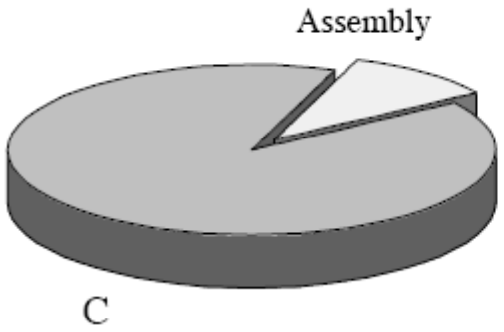
No need to handle underflow, overflow, etc.



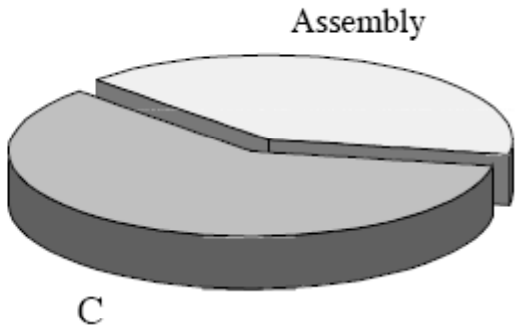


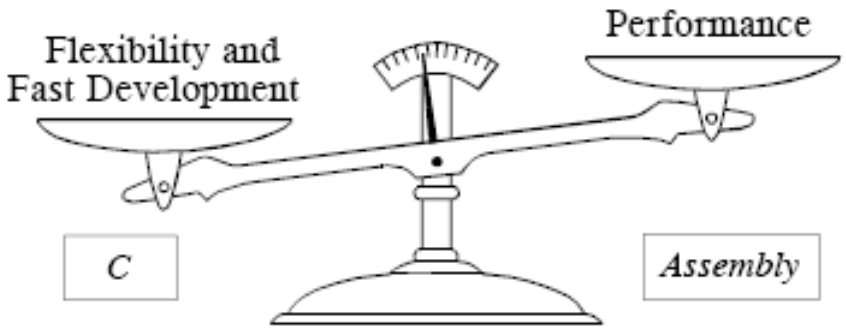
C verses Assembly

a. Traditional Programmers



b. DSP Programmers





c. DSP Revenue