

Âm thanh và công nghệ xử lý âm thanh

Nguyễn Văn Tỉnh

Các nội dung chương 4

- Mục tiêu chương 4
- Một số khái niệm
- Âm thanh số
- Nén âm thanh
- MIDI vs. Audio
- Tổng kết chương
- Tài liệu tham khảo

Mục tiêu chương 4

- Người học sẽ:
 - + Được trang bị kiến thức âm thanh, biểu diễn, lưu trữ và tạo âm thanh
 - + Được giới thiệu nguyên lý và phương pháp nén, xử lý và truyền âm thanh
- Sau khi kết thúc chương, người học :
 - + Nắm được kiến thức cơ bản về âm thanh
 - + Biết vận dụng một số kỹ thuật, công cụ xử lý âm thanh để tạo và lưu trữ âm thanh trong ứng dụng cụ thể

Âm thanh là gì ?

AUDIO



If a tree falls in the forest and nobody is there to hear it, will it make a sound?

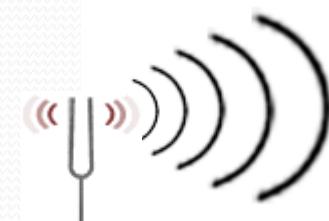
Âm thanh là gì ?

- Âm thanh có thể là
 - + Tiếng nói
 - + Âm nhạc
 - + Tiếng ồn
- Âm thanh là một mối quan hệ phức tạp của các đối tượng sau:

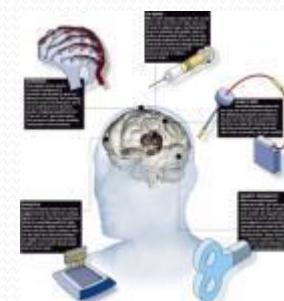
+ Nguồn âm



+ Môi trường truyền (thường là không khí)

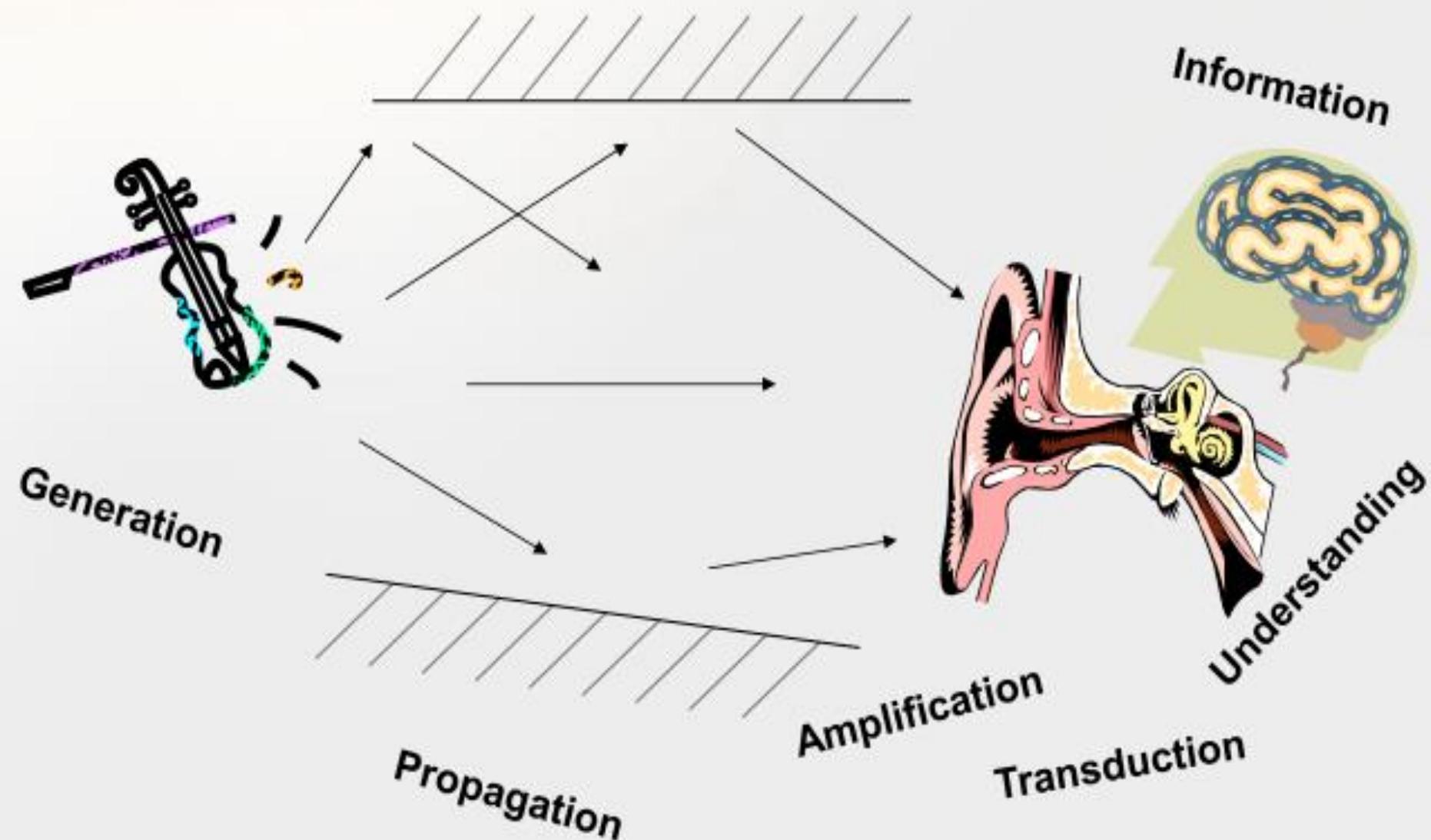


+ Bộ thu nhận âm (tai)

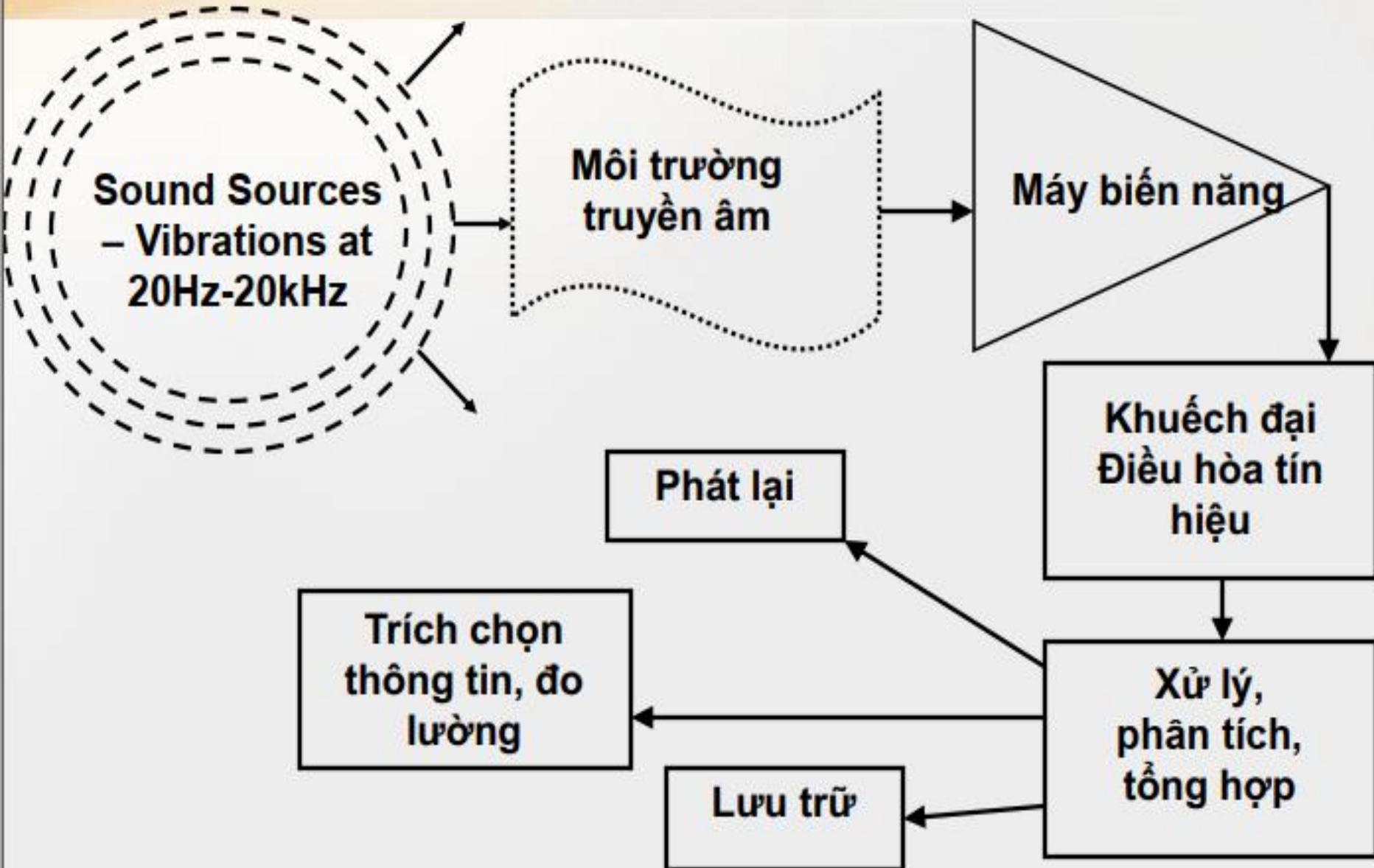


+ Bộ cảm nhận âm (não bộ)

Hệ thống truyền nhận âm tự nhiên



Hệ thống truyền nhận âm điện tử



Sóng âm



Something vibrates
in the air



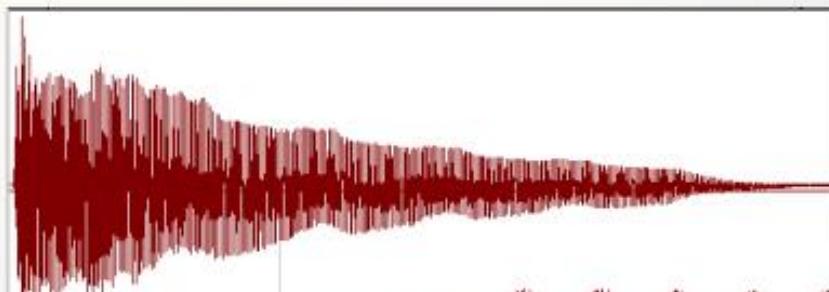
Waves of pressure
Ear drums will translate
these changes in wave
Forms as sound



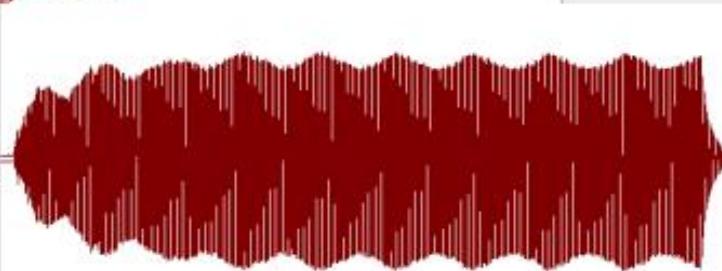
- ➊ Âm thanh được đo bằng decibel → **dB (decibel)**
- ➋ Âm thanh lan truyền trong môi trường: **sóng âm**.

Sóng âm

Một số ví dụ về sóng âm

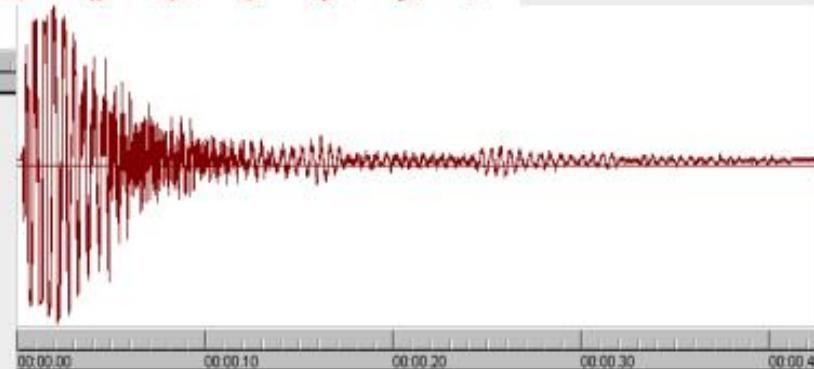


Piano



Pan flute

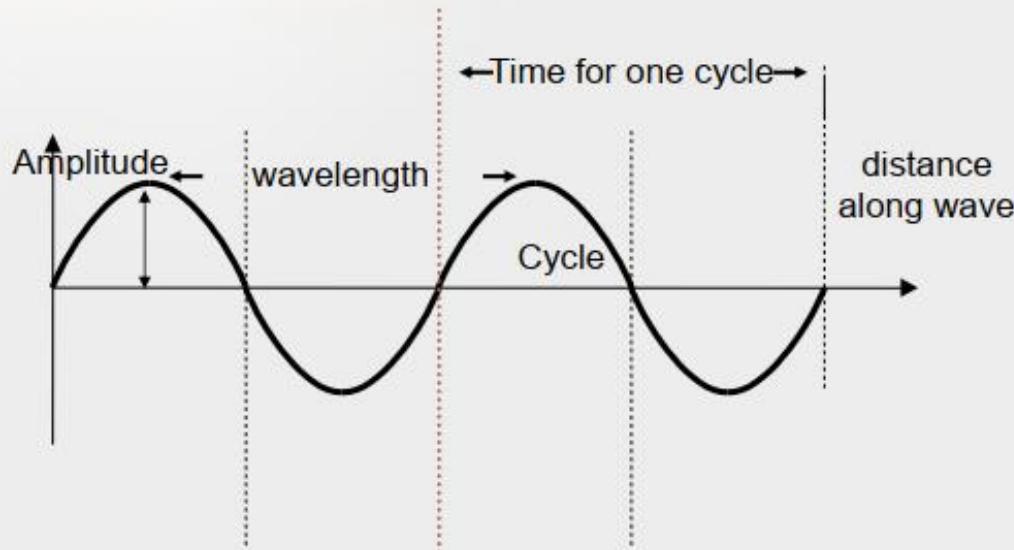
Snare drum



Sóng âm

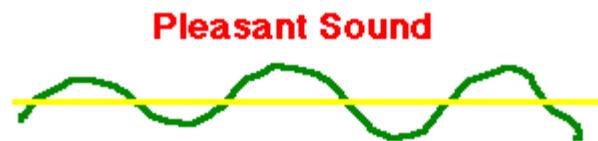
Đặc trưng của sóng âm

- ❖ Âm thanh được mô tả bởi hai đặc trưng sau:
 - ❖ **Tần số Frequency** (or pitch)
 - ❖ **Biên độ Amplitude** (or loudness)



Âm thanh và nhiễu âm

- Một âm thanh tốt có mô hình dạng sóng đều đặn. Các mô hình được lặp đi lặp lại.

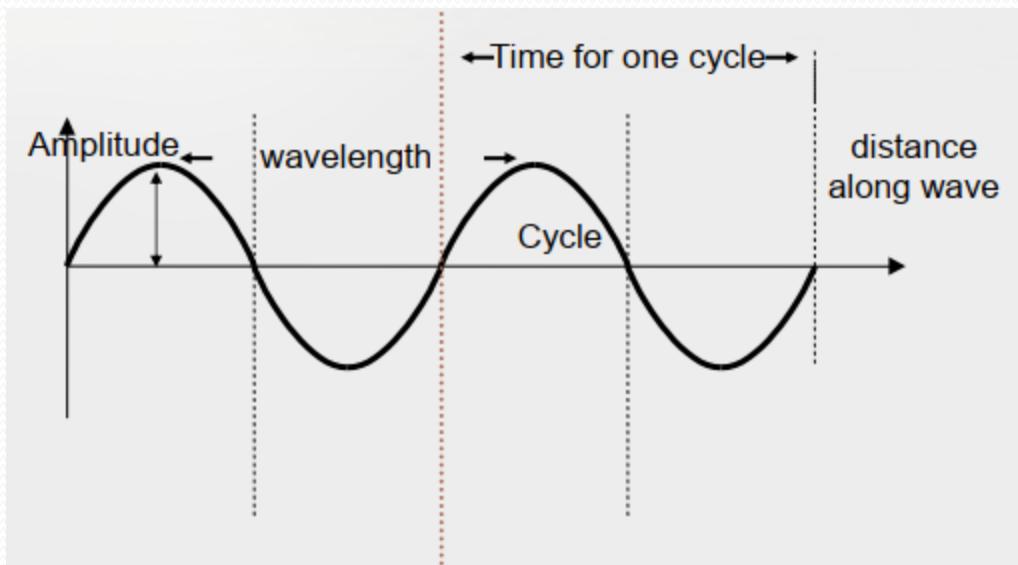


- Mô hình sóng của nhiễu không đều. Chúng không có mô hình được lặp lại.



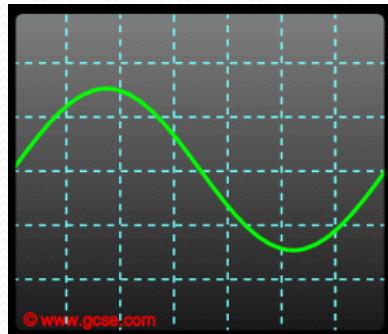
Đặc trưng của sóng âm

- Âm thanh được mô tả bởi hai đặc trưng sau:
 - + **Tần số Frequency** (or pitch)
 - + **Biên độ Amplitude** (or loudness)

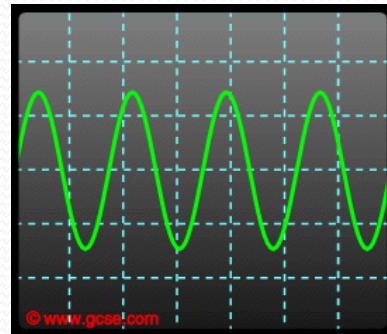


Tần số

- Tần số là một đo lường có bao nhiêu dao động xảy ra trong một giây (second). Đo lường này được đo bằng Hertz (viết tắt Hz) và tương ứng trực tiếp với âm vực (pitch) của âm thanh.
- Rung động xảy ra càng thường xuyên thì âm vực càng cao



Low pitch



High pitch

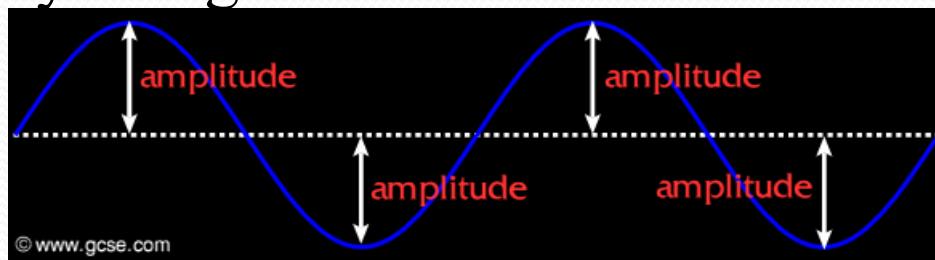
Một cách tối ưu, tai người có thể nghe từ 20 Hz đến 20.000 Hz (20 kHz)

- Âm thanh dưới 20 Hz là vi âm (infrasonic)
- âm thanh trên 20 kHz là siêu âm (ultrasonic)

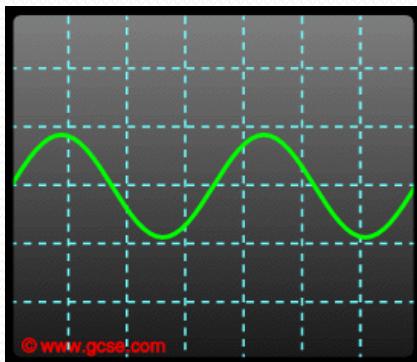
Biên độ âm thanh

- Biên độ là độ dịch chuyển lớn nhất của sóng từ một vị trí cân bằng

Âm thanh càng lớn thì càng có nhiều năng lượng.
Điều này có nghĩa là âm thanh lớn có biên độ lớn



Quiet



Loud



Low amplitude

High Amplitude

Biên độ liên quan đến độ lớn của âm

Âm thanh tương tự và âm thanh số

Âm thanh tương tự

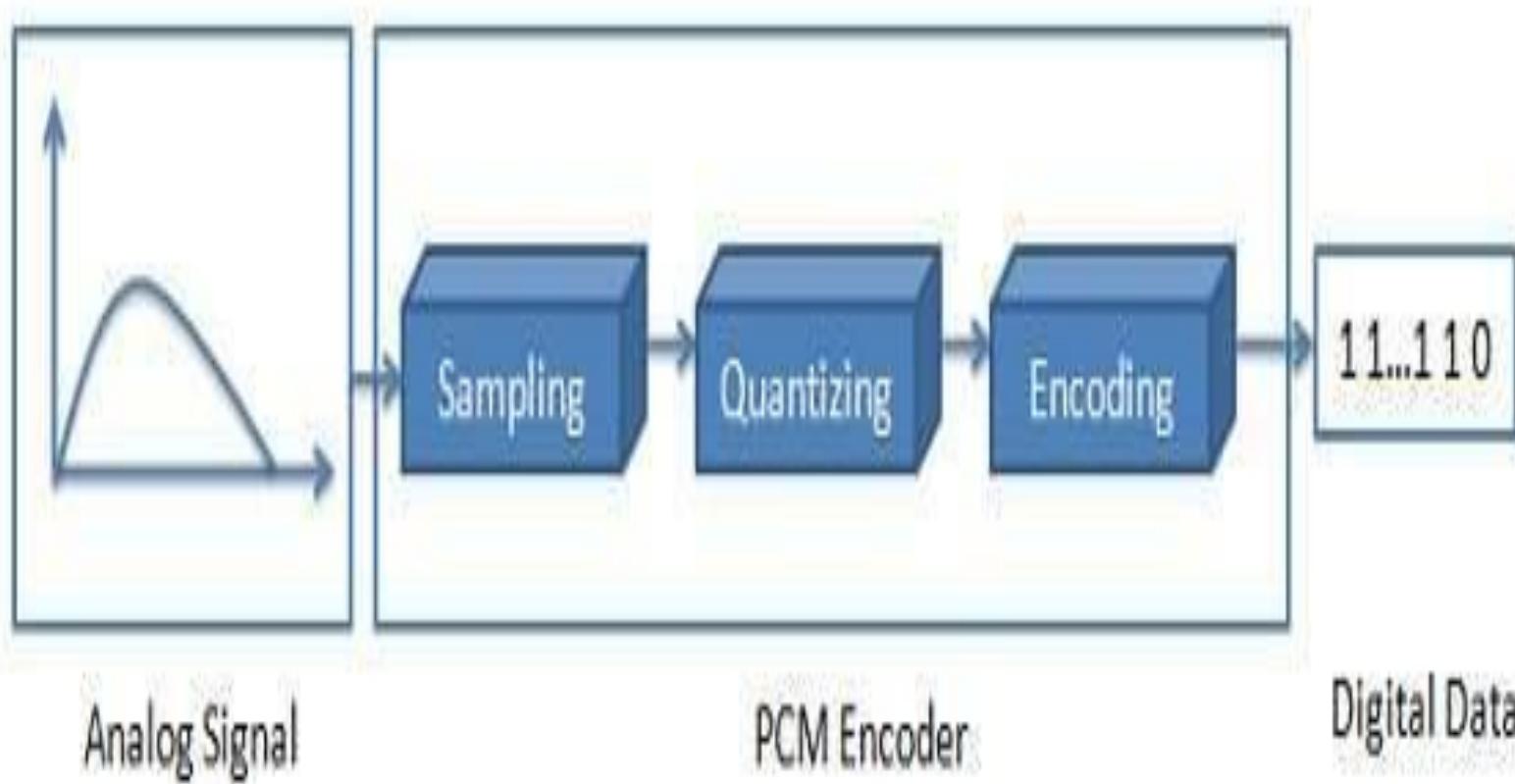
- ➊ Tín hiệu điện mang thông tin về âm thanh như một giá trị điện thế liên tục

Số hóa âm thanh

- ➋ Để sử dụng trong các ứng dụng đa phương tiện, giống như ảnh, video, âm thanh phải được số hóa

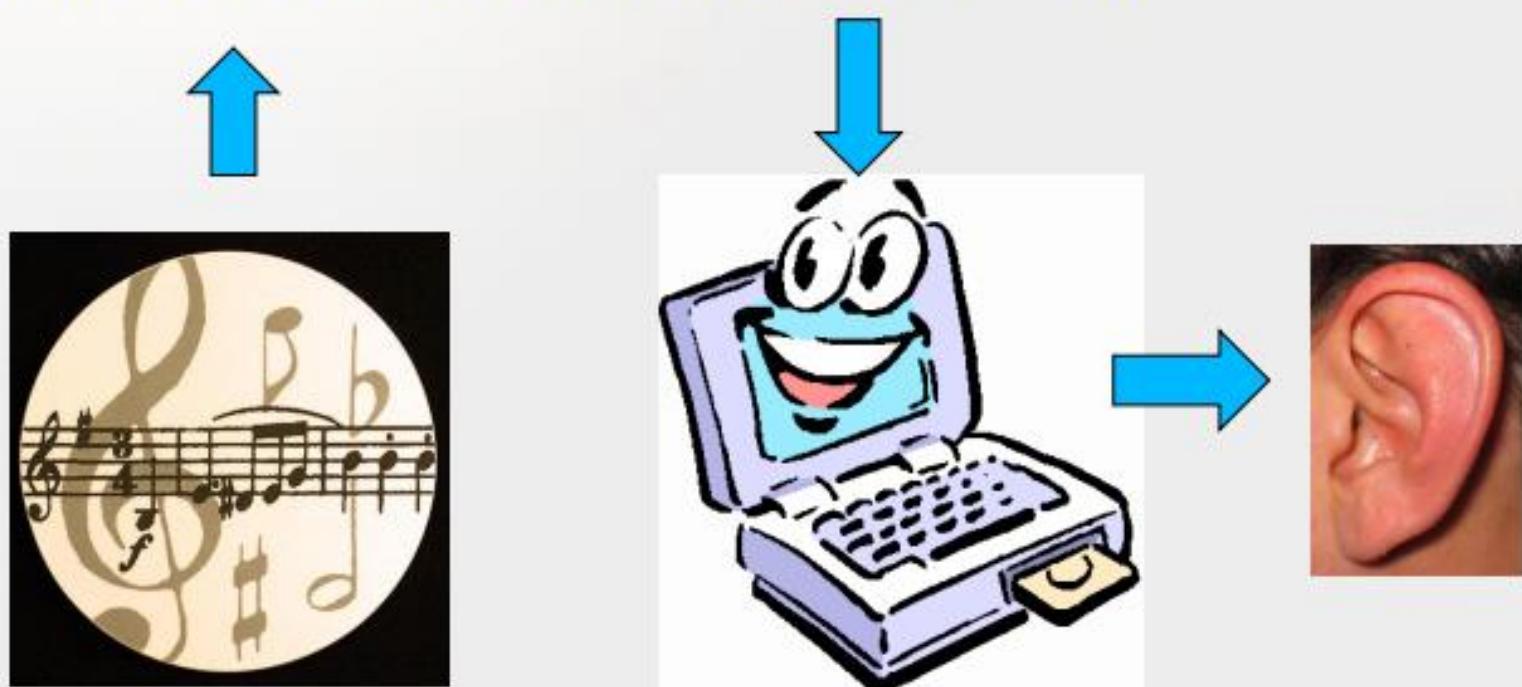


Số hóa âm thanh



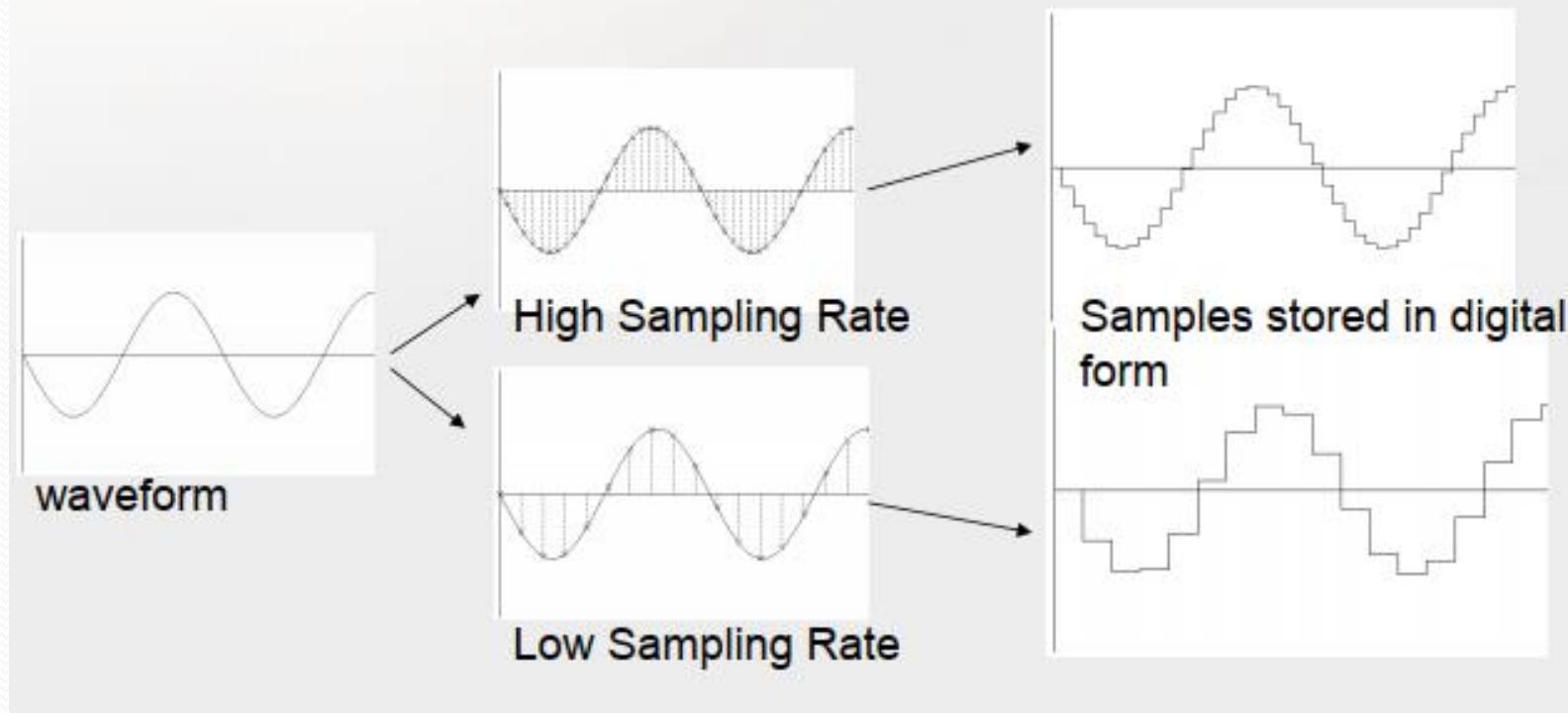
Biểu diễn số của tín hiệu âm thanh

0101010000101010101010011010001010100110100101001
0100011100010101010100101111001001010...



Âm thanh số

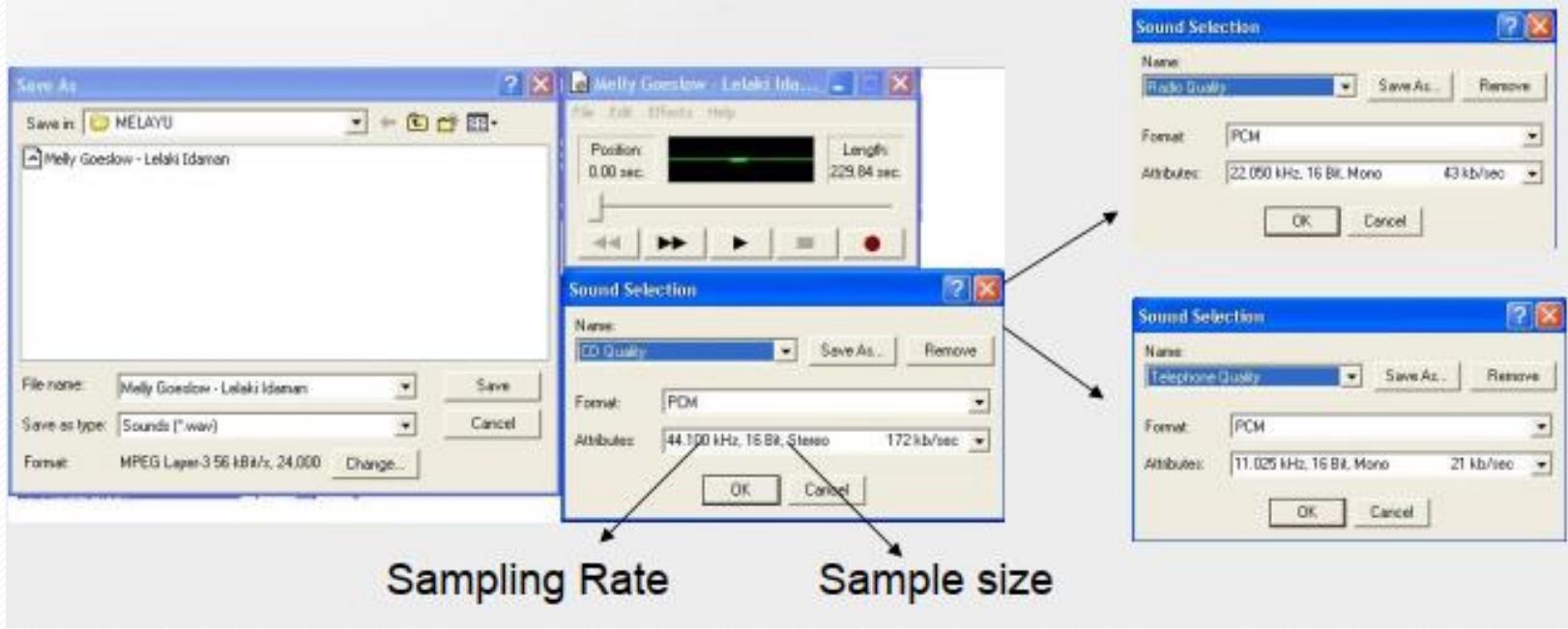
Dữ liệu âm thanh số là biểu diễn của âm thanh, được lưu trữ dưới dạng các điểm mẫu



Âm thanh số

Chất lượng âm thanh số phụ thuộc vào

1. Tốc độ lấy mẫu (sampling rate)
2. Kích thước mẫu (quantization step)



Âm thanh số

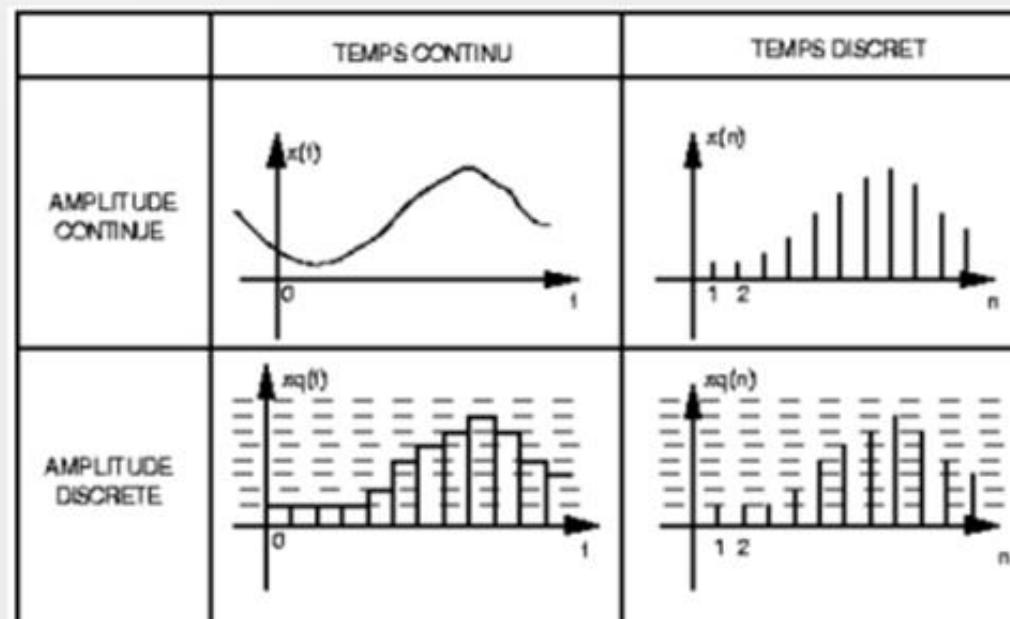
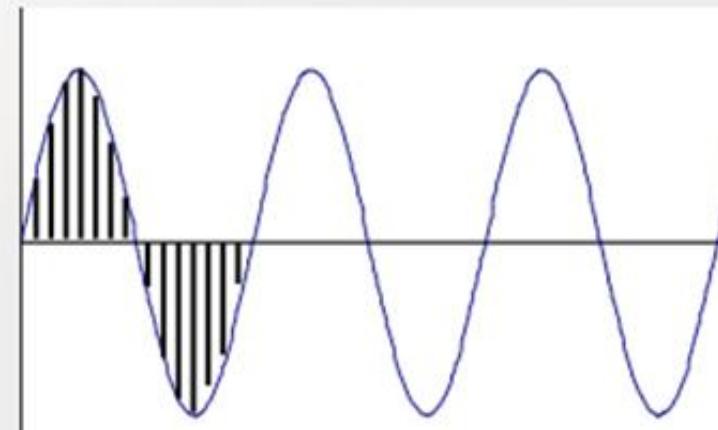
Ngoài ra, chất lượng âm thanh số phụ thuộc vào

- Chất lượng của nguồn âm
- Chất lượng của thiết bị thu và các phần cứng hỗ trợ.
- Các đặc trưng sử dụng để thu âm.
- Khả năng phát lại của môi trường phát âm.

Biểu diễn số tín hiệu âm thanh

Tín hiệu âm thanh được

- ⦿ Lấy mẫu
- ⦿ Lượng tử hóa
- ⦿ mã hóa



Lấy mẫu (Sampling)

In order for a computer to work with audio waves, they must be converted from analog to digital form.

This is done through a process called sampling, in which every fraction of a second a sample of the audio is recorded in digital bits

There are two factors that affect the quality of the digitized audio:

- Sample rate
- Sample size

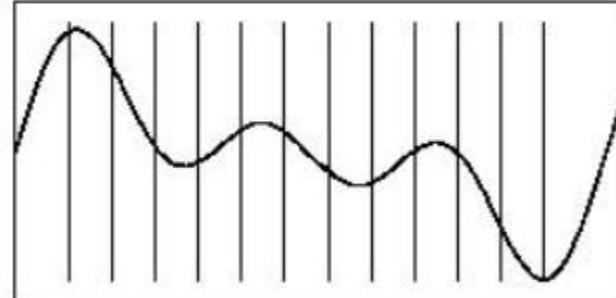
Lấy mẫu (Sampling)

Sample Rate

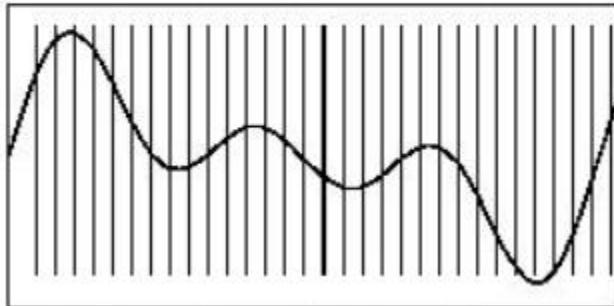
- Sample rate is the **number of times** the sample is taken.
- The three most common sample rates are; 11.025 kHz, 22.05 kHz, and 44.1 kHz.
- The higher the sample rate, the more samples that are taken and, thus, the better the quality of the digitized audio.

Lấy mẫu (Sampling)

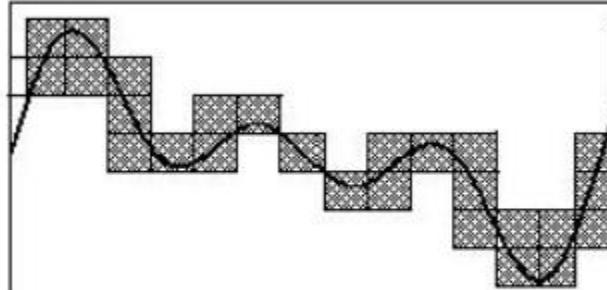
Sample Rate



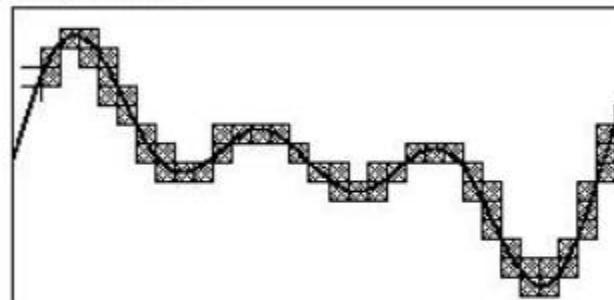
lower sample rates take fewer snapshots
of the waveform



faster sample rates take more
snapshots....



resulting in a rough recreation of the
waveform.



resulting in a smoother and more detailed
recreation of the waveform.

Lấy mẫu (Sampling)

Nyquist Theorem

For lossless digitization, the sampling rate should be at least twice the maximum frequency response.

- ➊ In mathematical terms:

$$f_s > 2*f_m$$

- ➋ where f_s is sampling frequency and f_m is the maximum frequency in the signal

Lấy mẫu (Sampling)

Nyquist Limit

- ➊ max data rate = $2 H \log_2 V$ bits/second, where
 - H = bandwidth (in Hz)
 - V = discrete levels (bits per signal change)
- ➋ Shows the maximum number of bits that can be sent per second on a *noiseless* channel with a bandwidth of H, if V bits are sent per signal
 - ➌ Example: what is the maximum data rate for a 3kHz channel that transmits data using 2 levels (binary) ?
 - ➍ Solution ($2 \times 3,000 \times \ln 2 = 6,000$ bits/second)

Lấy mẫu (Sampling)

Sample Size

- Sample size is the **amount of information** stored about the sample.
- The two most common sample sizes are 8 bit and 16 bit.
- An 8-bit sample allows 256 values that are used to describe audio, whereas a 16-bit sample provides 65,536 values.
- The greater the sample size, the better the quality of the audio.

Lấy mẫu (Sampling)

Sample Size

- Table below shows the file size (in bytes) for 10 seconds of digital audio given various sample rates and bit values.

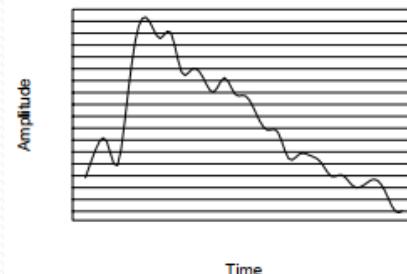
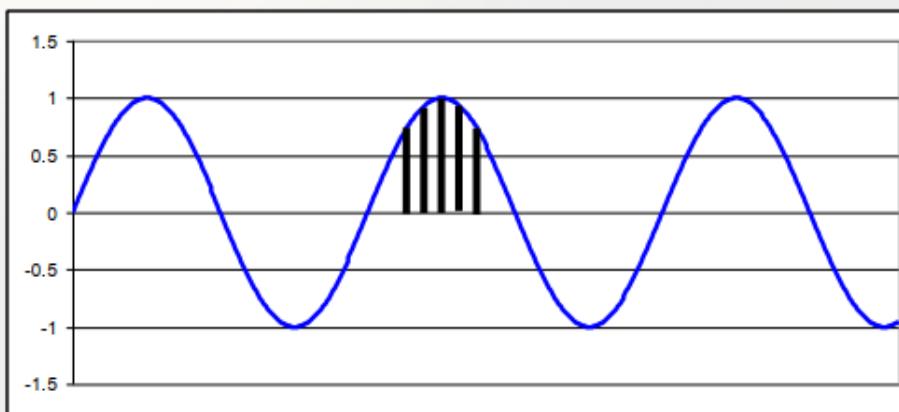
Sample Rate	Bit Value	Size of File
44.1 kHz	16	1.76 MB
44.1 kHz	8	882 KB
22.05 kHz	16	882 KB
22.05 kHz	8	440 KB
11.025 kHz	8	220 KB

Lượng tử hóa

Lượng tử hóa là quá trình rời rạc hóa tín hiệu tương tự về biên độ. Tại mỗi mẫu, biên độ được chia ra các mức gọi là mức lượng tử. Số lượng mức lượng tử N được xác định bởi số bit n , $N=2^n$ mức. Ví dụ: khi $n=4$ bit thì $N=16$ và khi $n=8$ bit thì $N=256$ mức lượng tử.

Như vậy, số bit lượng tử càng lớn thì số mức lượng tử càng nhiều và chuyển đổi A/D

càng chính xác. Khoảng cách giữa hai mức lượng tử liền kề gọi là bước lượng tử Q .



Typically use

- ➊ 8 bits = 256 levels
- ➋ 16 bits = 65,536 levels

How should the levels be distributed?

- ➊ Linearly? (PCM)
- ➋ Perceptually? (u-Law)
- ➌ Differential? (DPCM)
- ➍ Adaptively? (ADPCM)

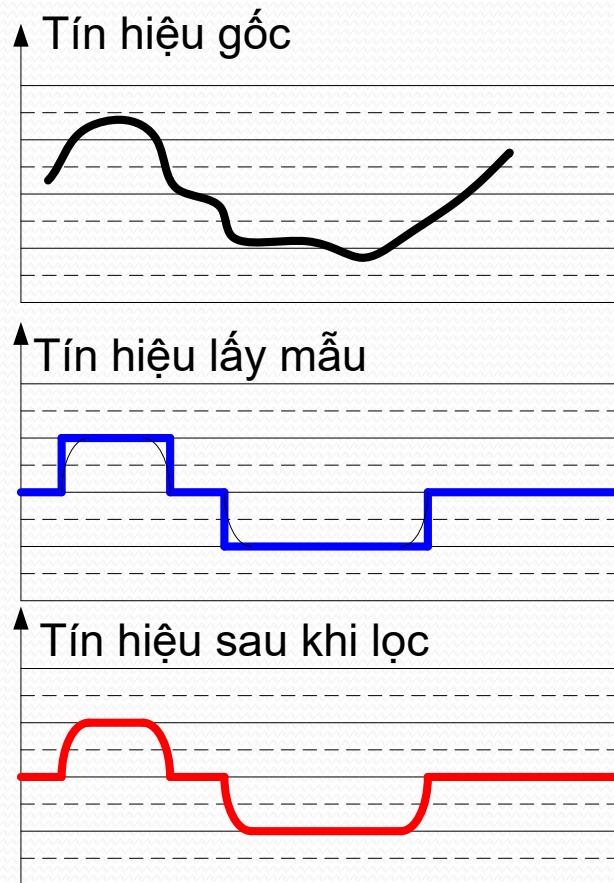
Lượng tử hóa

- Lượng tử hóa là một quá trình để gán một giá trị rời rạc từ một loạt các giá trị có thể có cho mỗi mẫu. Số lượng mẫu hoặc dải giá trị phụ thuộc vào số lượng bit được sử dụng để đại diện cho mỗi mẫu. Kết quả lượng tử hóa tạo ra dạng sóng bước giống như tín hiệu nguồn.
- Lỗi / nhiễu lượng tử hóa - Sự khác biệt giữa mẫu và giá trị được gán cho nó được gọi là lỗi/ nhiễu lượng tử hóa hoặc.
- Tỷ lệ tín hiệu trên nhiễu (SNR) - Tín hiệu trên tỷ lệ đe cập đến chất lượng tín hiệu so với lỗi lượng tử hóa. Tỷ lệ Tín hiệu trên nhiễu càng cao, chất lượng âm thanh càng tốt. Làm việc với các mức rất nhỏ thường gây ra nhiều lỗi hơn. Vì vậy, thay vì lượng tử hóa đồng nhất, lượng tử hóa không đồng nhất được sử dụng như quá trình companding. Companding là một quá trình làm sai lệch tín hiệu tương tự theo cách có kiểm soát bằng cách nén các giá trị lớn ở nguồn và sau đó mở rộng ở đầu nhận trước khi quá trình lượng tử hóa diễn ra.

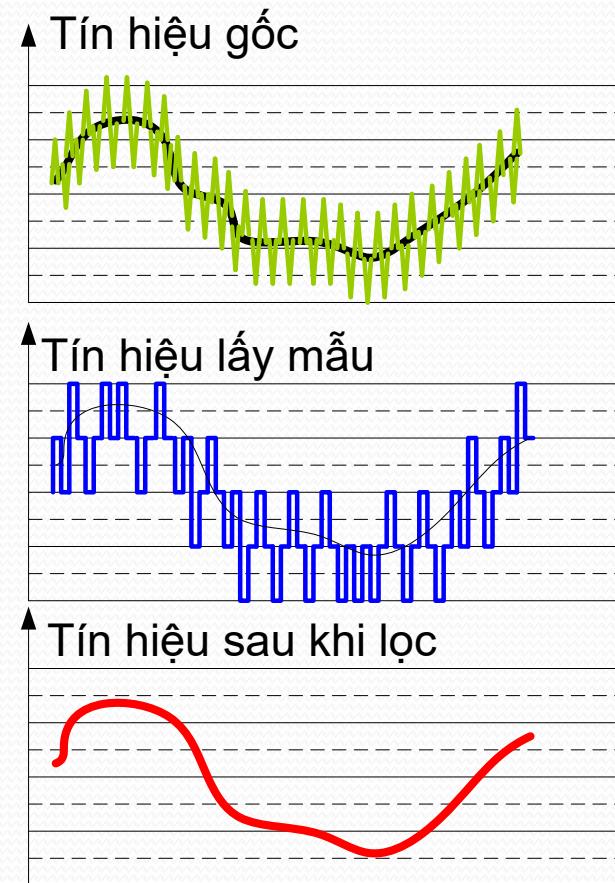
Dither

- *Nguyên nhân:* Lượng tử hóa → méo. Tín hiệu có biên độ càng nhỏ thì méo lượng tử càng cao.
- *Khắc phục:* Cộng âm thanh trước khi lấy mẫu với một tạp âm tương tự → Ngẫu nhiên hóa các ảnh hưởng méo lượng tử để phân phối đều méo lượng tử thành các lối ngẫu nhiên chứ không tập trung nhiều vào phần có biên độ thấp.
- *Khái niệm:* Dither là một nhiễu được cộng vào tín hiệu âm thanh.
- *Mục đích:* Loại bỏ méo lượng tử.
- *Cơ sở:* Dither làm cho tín hiệu âm thanh bị biến đổi giữa các mức lượng tử gần nhau, điều này làm giảm độ tương quan của lượng tử hóa tín hiệu, loại các ảnh hưởng của lỗi và mã hóa các biên độ tín hiệu thấp hơn một mức lượng tử.
- *Nhược điểm:* Cộng nhiễu vào tín hiệu.

Dither



Không dither



Dither

Mã hóa

- Là quá trình chuyển các mức rời rạc thành một chuỗi các mẫu số nhị phân (hoặc các hệ đếm khác) theo một quy luật nhất định.
- Sau mã hóa nhị phân, ta được tín hiệu điều xung mã PCM.
- Mã hóa âm thanh được sử dụng để thu được các biểu diễn số nhỏ gọn của tín hiệu âm thanh có độ trung thực cao nhằm mục đích truyền tín hiệu hoặc lưu trữ tín hiệu hiệu quả. Mục tiêu trong mã hóa âm thanh là thể hiện tín hiệu với số bit tối thiểu trong khi vẫn đạt được khả năng tái tạo tín hiệu âm thanh trong suốt mà đầu ra không thể phân biệt được với đầu vào ban đầu.

Mã hóa

Các phương pháp mã hóa:

- Điều chế mã xung- Pulse code modulation (PCM)
- Mã hóa vi sai (Differential Coding)
- Mã hóa dự đoán không tổn thất (Lossless predictive coding)
- Điều chế mã xung vi sai -Differential Pulse Code Modulation, DPCM
- Điều chế mã xung vi sai thích ứng -Adaptive Differential Pulse Code Modulation, ADPCM

Mã hóa

- Điều chế mã xung- Pulse code modulation (PCM)

PCM là một phương pháp hoặc kỹ thuật để truyền dữ liệu tương tự theo cách số và nhị phân độc lập với độ phức tạp của dạng sóng tương tự. Tất cả các loại dữ liệu tương tự như video, thoại; nhạc, v.v. có thể được chuyển bằng PCM. Nó là dạng tiêu chuẩn cho âm thanh kỹ thuật số trong máy tính.

Nó được sử dụng bởi các định dạng Blu-ray, DVD và Compact Disc và các hệ thống khác như hệ thống điện thoại kỹ thuật số. PCM bao gồm ba bước để số hóa một tín hiệu tương tự:

- + Lấy mẫu
- + Lượng tử hóa
- + Mã hóa nhị phân

Mã hóa

- Mã hóa vi sai (Differential Coding)

Mã hóa vi sai hoạt động bằng cách làm cho các số nhỏ đi. Các số nhỏ hơn yêu cầu ít thông tin để mã hóa hơn các số lớn hơn. Sẽ chính xác hơn nếu khẳng định rằng các số tương tự yêu cầu ít thông tin hơn để mã.

Giả sử ta đang làm việc với một chuỗi byte. Số byte nằm trong khoảng từ 0..255. Đây là một chuỗi:

150 152 148 150 153 149 152

Trên phạm vi của 0 .. 255, những con số đó khá lớn.

Trong khi chúng khá giống nhau. Sự khác biệt được lấy và mã hóa thay vì số hoàn chỉnh. Thông thường, số đầu tiên được lấy và sau đó sự khác biệt được tính toán

Mã hóa

- Mã hóa vi sai (Differential Coding)

150

$152 - 150 = 2$

$148 - 152 = -4$

$150 - 148 = 2$

$153 - 150 = 3$

$149 - 153 = -4$

$152 - 149 = 3$

Vì vậy, chuỗi số mã hóa là

150 2 -4 2 3 -4 3

Khi giải mã chuỗi số này, bắt đầu với số đầu tiên và bắt đầu cộng các vi sai để có các số còn lại:

150

$150 + 2 = 152$

$152 - 4 = 148$

$148 + 2 = 150$

$150 + 3 = 153$

$153 - 4 = 149$

$149 + 3 = 152$

Do đó, âm thanh không được lưu trữ trong PCM đơn giản mà ở dạng khai thác sự khác biệt.

Mã hóa

Lossless Predictive Coding

- **Predictive coding:** simply means transmitting differences — predict the next sample as being equal to the current sample; send not the sample itself but the difference between previous and next.
 - (a) Predictive coding consists of finding differences, and transmitting these using a PCM system.
 - (b) Note that differences of integers will be integers. Denote the integer input signal as the set of values f_n . Then we **predict** values \widehat{f}_n as simply the previous value, and define the error e_n as the difference between the actual and the predicted signal:

$$\widehat{f}_n = f_{n-1}$$

$$e_n = f_n - \widehat{f}_n$$

Mã hóa

Lossless Predictive Coding

- (c) But it is often the case that some **function** of a few of the previous values, f_{n-1} , f_{n-2} , f_{n-3} , etc., provides a better prediction. Typically, a linear **predictor** function is used:

$$\widehat{f}_n = \sum_{k=1}^{\text{2 to 4}} a_{n-k} f_{n-k}$$

Mã hóa

Lossless Predictive Coding

- The idea of forming differences is to make the histogram of sample values more peaked.
 - (a) For example, Fig.6.15(a) plots 1 second of sampled speech at 8 kHz, with magnitude resolution of 8 bits per sample.
 - (b) A histogram of these values is actually centered around zero, as in Fig. 6.15(b).
 - (c) Fig. 6.15(c) shows the histogram for corresponding speech signal **differences**: difference values are much more clustered around zero than are sample values themselves.
 - (d) As a result, a method that assigns short codewords to frequently occurring symbols will assign a short code to zero and do rather well: such a coding scheme will much more efficiently code sample differences than samples themselves.

Mã hóa

Lossless Predictive Coding

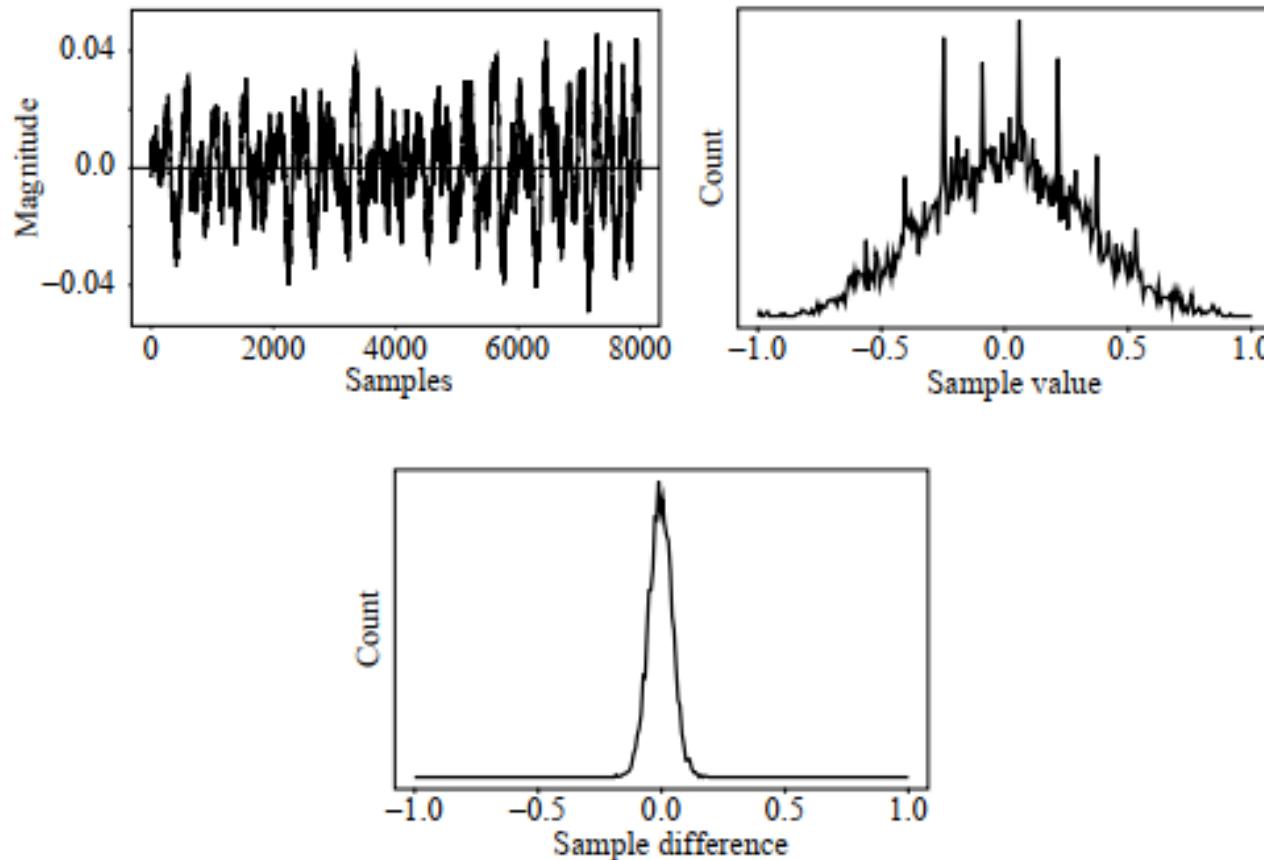


Fig. 6.15: Differencing concentrates the histogram. (a): Digital speech signal. (b): Histogram of digital speech signal values. (c): Histogram of digital speech signal differences.

Mã hóa

Lossless Predictive Coding

- One problem: suppose our integer sample values are in the range 0..255. Then differences could be as much as -255..255 — we've increased our **dynamic range** (ratio of maximum to minimum) by a factor of two → need more bits to transmit some differences.
 - (a) A clever solution for this: define two new codes, denoted SU and SD, standing for Shift-Up and Shift-Down. Some special code values will be reserved for these.
 - (b) Then we can use codewords for only a limited set of signal differences, say only the range -15..16. Differences which lie in the limited range can be coded as is, but with the extra two values for SU, SD, a value outside the range -15..16 can be transmitted as a series of shifts, followed by a value that is indeed inside the range -15..16.
 - (c) For example, 100 is transmitted as: SU, SU, SU, 4, where (the codes for) SU and for 4 are what are transmitted (or stored).

Mã hóa

Lossless Predictive Coding

- *Lossless predictive coding* — the decoder produces the same signals as the original. As a simple example, suppose we devise a predictor for \widehat{f}_n as follows:

$$\widehat{f}_n = \lfloor \frac{1}{2}(f_{n-1} + f_{n-2}) \rfloor$$

$$e_n = f_n - \widehat{f}_n$$

Mã hóa

Lossless Predictive Coding

- Let's consider an explicit example. Suppose we wish to code the sequence $f_1, f_2, f_3, f_4, f_5 = 21, 22, 27, 25, 22$. For the purposes of the predictor, we'll invent an extra signal value f_0 , equal to $f_1 = 21$, and first transmit this initial value, uncoded:

$$\widehat{f}_2 = 21, e_2 = 22 - 21 = 1;$$

$$\begin{aligned}\widehat{f}_3 &= \lfloor \frac{1}{2}(f_2 + f_1) \rfloor = \lfloor \frac{1}{2}(22 + 21) \rfloor = 21, \\ e_3 &= 27 - 21 = 6;\end{aligned}$$

$$\begin{aligned}\widehat{f}_4 &= \lfloor \frac{1}{2}(f_3 + f_2) \rfloor = \lfloor \frac{1}{2}(27 + 22) \rfloor = 24, \\ e_4 &= 25 - 24 = 1;\end{aligned}$$

$$\widehat{f}_5 = \lfloor \frac{1}{2}(f_4 + f_3) \rfloor = \lfloor \frac{1}{2}(25 + 27) \rfloor = 26,$$

$$e_5 = 22 - 26 = -4$$

Mã hóa

Lossless Predictive Coding

- The error does center around zero, we see, and coding (assigning bit-string codewords) will be efficient. Fig. 6.16 shows a typical schematic diagram used to encapsulate this type of system:

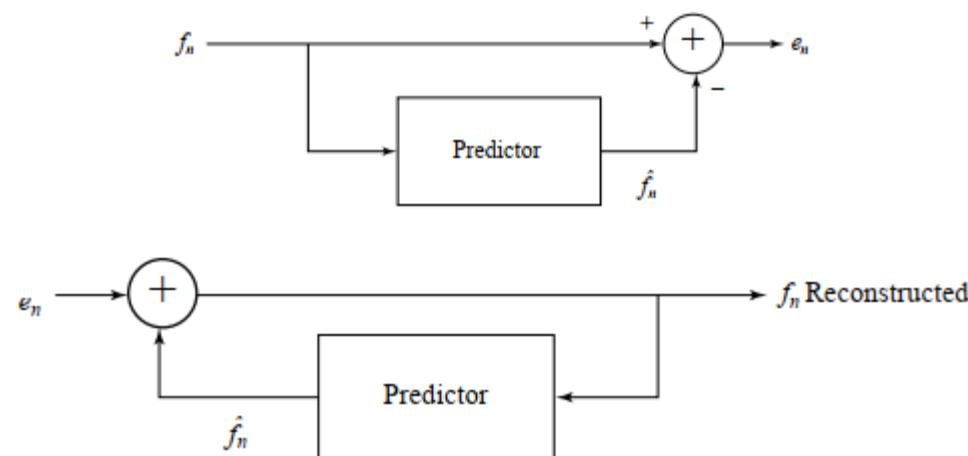


Fig. 6.16: Schematic diagram for Predictive Coding encoder and decoder.

Mã hóa

Differential Pulse Code Modulation, DPCM

- Differential PCM is exactly the same as Predictive Coding, except that it incorporates a *quantizer* step.
 - (a) One scheme for analytically determining the best set of quantizer steps, for a non-uniform quantizer, is the **Lloyd-Max** quantizer, which is based on a least-squares minimization of the error term.
 - (b) Our nomenclature: signal values: f_n – the original signal, \hat{f}_n – the predicted signal, and \tilde{f}_n the quantized, reconstructed signal.

Mã hóa

Differential Pulse Code Modulation, DPCM

- (c) **DPCM:** form the prediction; form an error e_n by subtracting the prediction from the actual signal; then quantize the error to a quantized version, \tilde{e}_n .

The set of equations that describe DPCM are as follows:

$$\hat{f}_n = \text{function_of}(\tilde{f}_{n-1}, \tilde{f}_{n-2}, \tilde{f}_{n-3}, \dots),$$

$$e_n = f_n - \hat{f}_n,$$

$$\tilde{e}_n = Q[e_n],$$

transmit *codeword*(\tilde{e}_n) ,

reconstruct: $\tilde{f}_n = \hat{f}_n + \tilde{e}_n$.

Then *codewords* for quantized error values \tilde{e}_n are produced using entropy coding, e.g. Huffman coding

Mã hóa

Differential Pulse Code Modulation, DPCM

- (d) The main effect of the coder-decoder process is to produce reconstructed, quantized signal values $\tilde{f}_n = \hat{f}_n + \tilde{e}_n$.

The **distortion** is the average squared error $[\sum_{n=1}^N (\tilde{f}_n - f_n)^2]/N$; one often plots distortion versus the number of bit-levels used. A Lloyd-Max quantizer will do better (have less distortion) than a uniform quantizer.

- For speech, we could modify quantization steps adaptively by estimating the mean and variance of a patch of signal values, and shifting quantization steps accordingly, for every block of signal values. That is, starting at time i we could take a block of N values f_n and try to minimize the quantization error:

$$\min \sum_{n=i}^{i+N-1} (f_n - Q[f_n])^2$$

Mã hóa

Differential Pulse Code Modulation, DPCM

- Since signal **differences** are very peaked, we could model them using a Laplacian probability distribution function, which is strongly peaked at zero: it looks like

$$l(x) = (1/\sqrt{2\sigma^2}) \exp(-\sqrt{2}|x|/\sigma)$$

for variance σ^2 .

- So typically one assigns quantization steps for a quantizer with nonuniform steps by assuming signal differences, d_n are drawn from such a distribution and then choosing steps to minimize

$$\min \sum_{n=i}^{i+N-1} (d_n - Q[d_n])^2 l(d_n).$$

Mã hóa

Differential Pulse Code Modulation, DPCM

- This is a least-squares problem, and can be solved iteratively == the Lloyd-Max quantizer.
- Schematic diagram for DPCM:

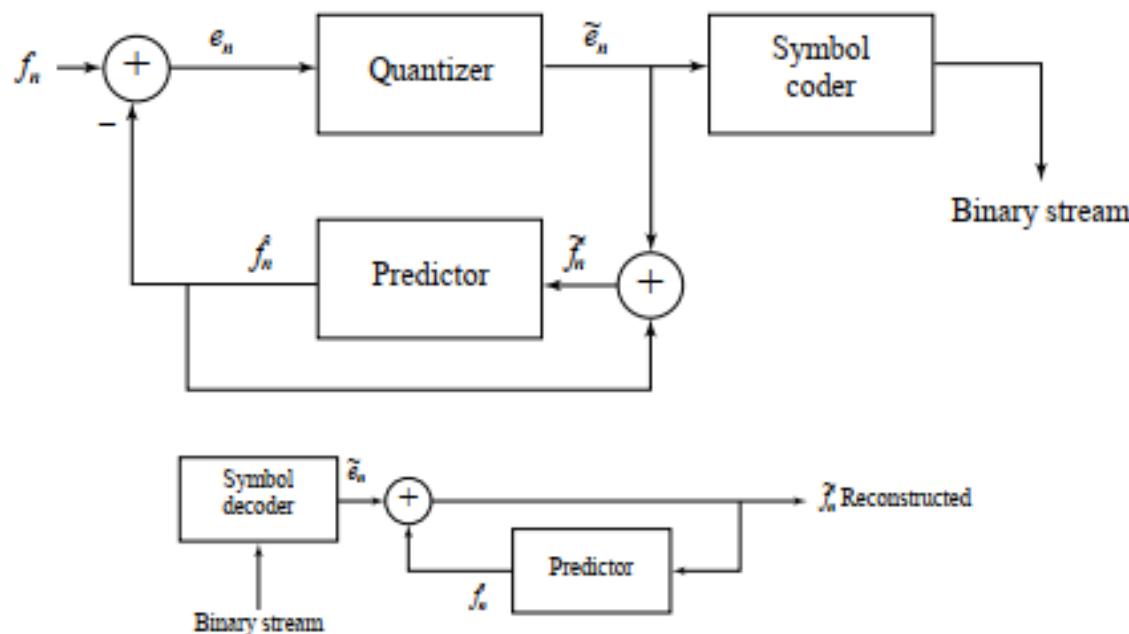


Fig. 6.17: Schematic diagram for DPCM encoder and decoder

Mã hóa

Differential Pulse Code Modulation, DPCM

- Notice that the quantization noise, $f_n - \tilde{f}_n$, is equal to the quantization effect on the error term, $e_n - \tilde{e}_n$.
- Let's look at actual numbers: Suppose we adopt the particular predictor below:

$$\hat{f}_n = \text{trunc}((\tilde{f}_{n-1} + \tilde{f}_{n-2}) / 2)$$

so that $e_n = f_n - \hat{f}_n$ is an integer.

- As well, use the quantization scheme:

$$\tilde{e}_n = Q[e_n] = 16 * \text{trunc}((255 + e_n) / 16) - 256 + 8$$

$$\tilde{f}_n = \hat{f}_n + \tilde{e}_n$$

Mã hóa

Differential Pulse Code Modulation, DPCM

- First, we note that the error is in the range $-255..255$, i.e., there are 511 possible levels for the error term. The quantizer simply divides the error range into 32 patches of about 16 levels each. It also makes the representative reconstructed value for each patch equal to the midway point for each group of 16 levels.
 - Table 6.7 gives output values for any of the input codes: 4-bit codes are mapped to 32 reconstruction levels in a staircase fashion.

Table 6.7 DPCM quantizer reconstruction levels.

e_n in range	Quantized to value
-255 .. -240	-248
-239 .. -224	-232
.	.
.	.
-31 .. -16	-24
-15 .. 0	-8
1 .. 16	8
17 .. 32	24
.	.
.	.
225 .. 240	232
241 .. 255	248

Mã hóa

Differential Pulse Code Modulation, DPCM

- As an example stream of signal values, consider the set of values:

$$\begin{array}{ccccc} f_1 & f_2 & f_3 & f_4 & f_5 \\ 130 & 150 & 140 & 200 & 230 \end{array} .$$

- Prepend extra values $f = 130$ to replicate the first value, f_1 . Initialize with quantized error $\tilde{e}_1 \equiv 0$, so that the first reconstructed value is exact: $\tilde{f}_1 = 130$. Then the rest of the values calculated are as follows (with prepended values in a box):

$$\begin{aligned} \hat{f} &= \boxed{130}, 130, 142, 144, 167 \\ e &= \boxed{0}, 20, -2, 56, 63 \\ \tilde{e} &= \boxed{0}, 24, -8, 56, 56 \\ \tilde{f} &= \boxed{130}, 154, 134, 200, 223 \end{aligned}$$

- On the decoder side, we again assume extra values \tilde{f} equal to the correct value \tilde{f}_1 , so that the first reconstructed value \tilde{f}_1 is correct. What is received is \tilde{e}_n , and the reconstructed \tilde{f}_n is identical to that on the encoder side, provided we use exactly the same prediction rule.

Mã hóa

ADPCM (Adaptive DPCM)

- **ADPCM** (Adaptive DPCM) takes the idea of adapting the coder to suit the input much farther. The two pieces that make up a DPCM coder: the quantizer and the predictor.
 1. In Adaptive DM, adapt the quantizer step size to suit the input. In DPCM, we can change the step size as well as decision boundaries, using a non-uniform quantizer. We can carry this out in two ways:
 - (a) **Forward adaptive quantization**: use the properties of the input signal.
 - (b) **Backward adaptive quantization**: use the properties of the quantized output. If quantized errors become too large, we should change the non-uniform quantizer.

Mã hóa

ADPCM (Adaptive DPCM)

2. We can also **adapt the predictor**, again using forward or backward adaptation. Making the predictor coefficients adaptive is called *Adaptive Predictive Coding* (APC):
 - (a) Recall that the predictor is usually taken to be a linear function of previous reconstructed quantized values, \tilde{f}_n .
 - (b) The number of previous values used is called the "order" of the predictor. For example, if we use M previous values, we need M coefficients a_i , $i = 1..M$ in a predictor

$$\hat{f}_n = \sum_{i=1}^M a_i \tilde{f}_{n-i}$$

Mã hóa

ADPCM (Adaptive DPCM)

- However we can get into a difficult situation if we try to change the prediction coefficients, that multiply previous quantized values, because that makes a complicated set of equations to solve for these coefficients:
 - (a) Suppose we decide to use a least-squares approach to solving a minimization trying to find the best values of the a_i :

$$\min \sum_{n=1}^N (f_n - \hat{f}_n)^2$$

- (b) Here we would sum over a large number of samples f_n , for the current patch of speech, say. But because \hat{f}_n depends on the quantization we have a difficult problem to solve. As well, we should really be changing the fineness of the quantization at the same time, to suit the signal's changing nature; this makes things problematical.

Mã hóa

ADPCM (Adaptive DPCM)

- (c) Instead, one usually resorts to solving the simpler problem that results from using not \tilde{f}_n in the prediction, but instead simply the signal f_n itself. Explicitly writing in terms of the coefficients a_i , we wish to solve:

$$\min \sum_{n=1}^N (f_n - \sum_{i=1}^M a_i f_{n-i})^2$$

Differentiation with respect to each of the a_i , and setting to zero, produces a linear system of M equations that is easy to solve. (The set of equations is called the Wiener-Hopf equations.)

Mã hóa

ADPCM (Adaptive DPCM)

- Fig. 6.18 shows a schematic diagram for the ADPCM coder and decoder:

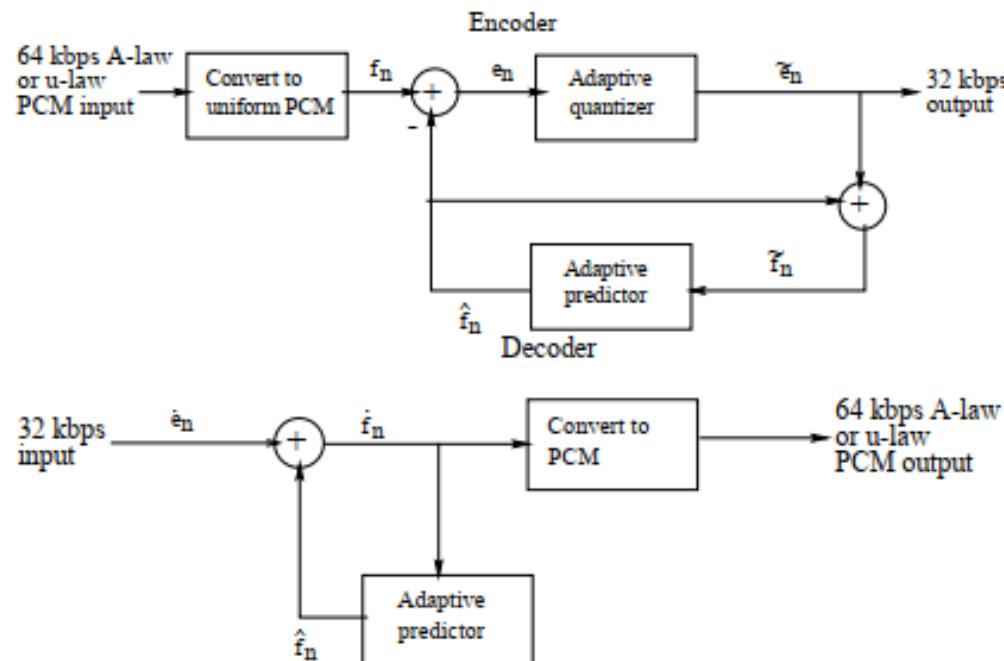


Fig. 6.18: Schematic diagram for ADPCM encoder and decoder

Mã hóa kênh

- Tín hiệu PCM không thích hợp để lưu trữ hoặc truyền dẫn vì vẫn còn tồn tại thành phần một chiều → mã hóa kênh.
- Mã hóa kênh
 - biến đổi dữ liệu với mục đích đạt được mật độ bit cao trong giới hạn băng thông của kênh truyền.
 - Giảm sự tổn hao trong khi truyền hoặc lưu trữ.
 - Cải thiện dải thông, dữ liệu truyền dẫn có đặc tính tối ưu.
 - Làm cho phổ tín hiệu âm thanh số ít méo.

Kênh (Channel)

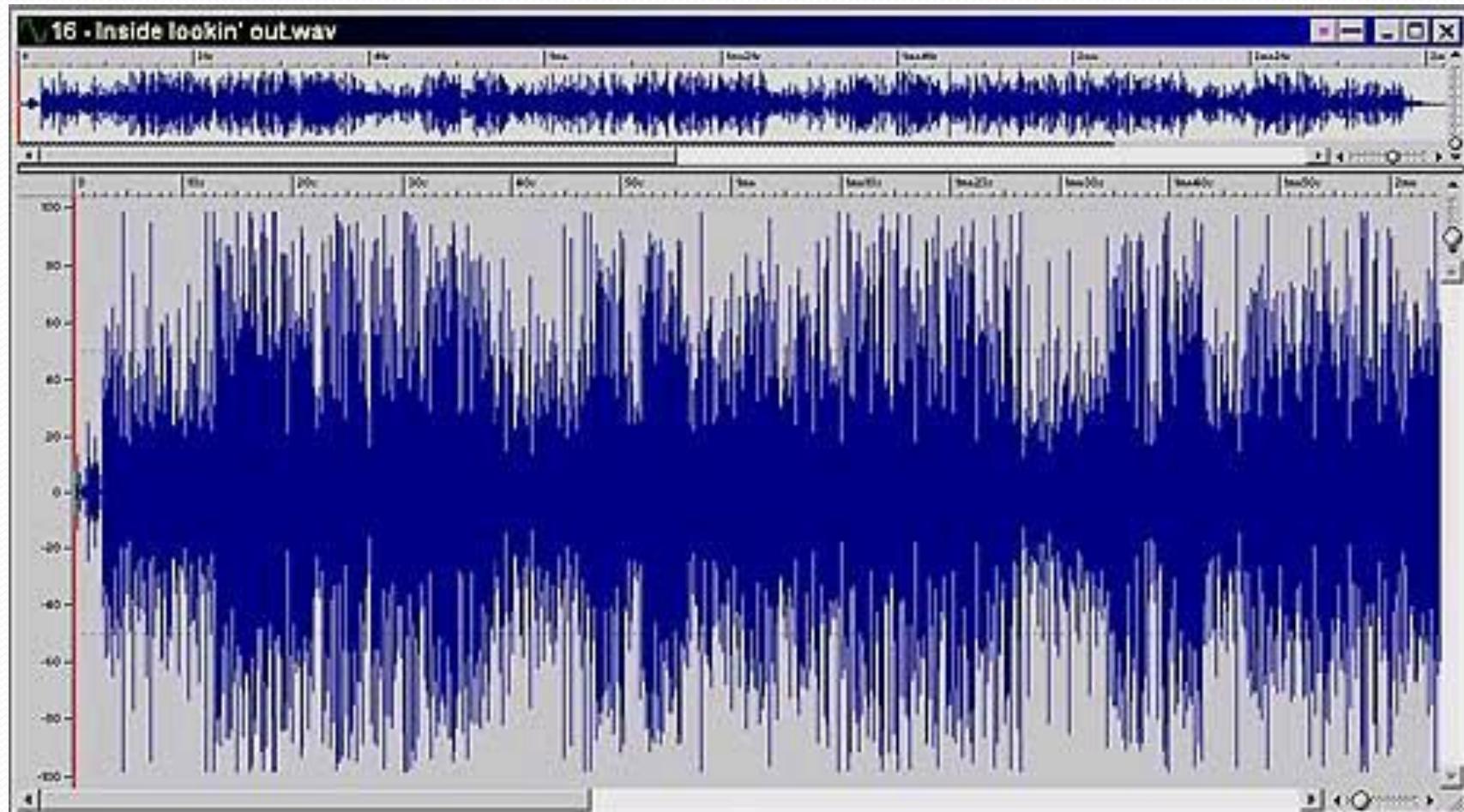
- Two types:
 - Monophonic
 - Stereophonic

Kênh (Channel)

Channel - Monophonic

- Commonly called *mono sound*, *mono*, or *non-stereo sound*, this early sound system used a single channel of audio for sound output.
- Monophonic sound is the most basic format of sound output.
- Mono (monophonic, or monaural) is sound from a single source.
- All speakers in a mono system (like an intercom) will carry the same signal.

Kênh (Channel)

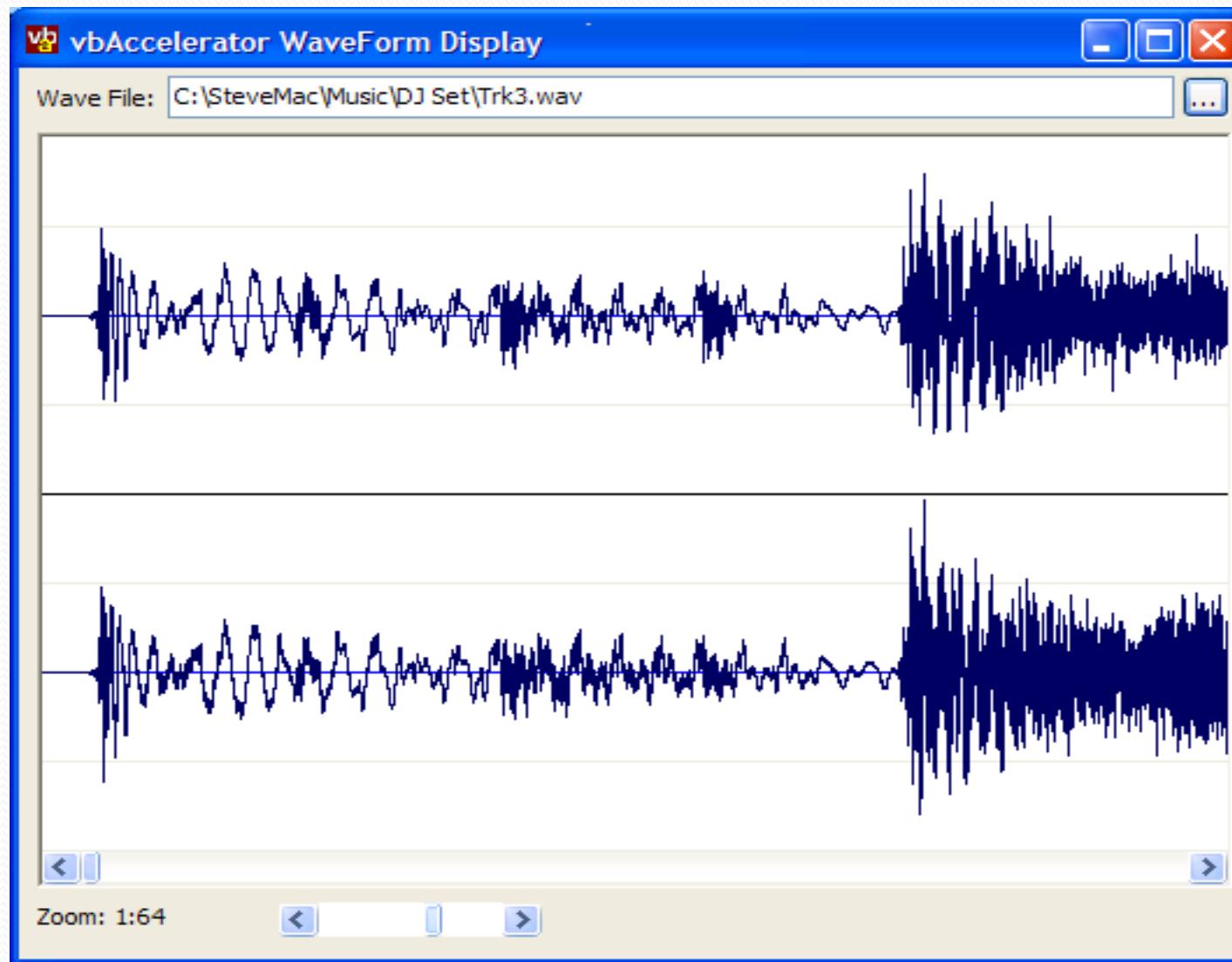


Kênh (Channel)

Channel - Stereophonic

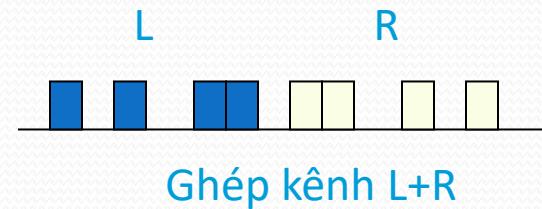
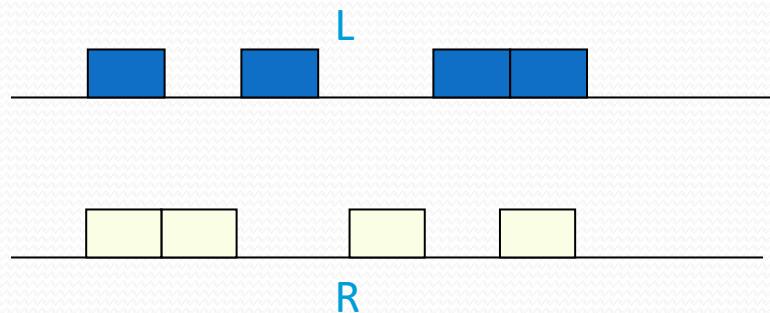
- Commonly called *stereo sound or just stereo*, stereophonic sound divides sounds across two channels (recorded on two separate sources) then the recorded sounds are mixed so that some elements are channeled to the left and others to the right.
- Stereo (stereophonic) is sound from two sources, ideally spaced apart, and reproduces sound the way we hear it naturally, with two ears.

Kênh (Channel)



Ghép kênh

- Tín hiệu âm thanh số thường bao gồm nhiều kênh, ví dụ hệ thống âm thanh 5.1 gồm các kênh trái, phải, trung tâm, trái vòm, phải vòm và siêu trầm, ngoài ra còn có các tín hiệu mã phụ, mã đồng bộ...
- Nguyên lý ghép kênh có thường được áp dụng đó là ghép kênh phân chia theo thời gian, mỗi kênh sử dụng một khe thời gian được ấn định trước.



Tính kích thước file âm thanh

Sound File Size = **Sample rate x sample size x channel x duration**

Example 1:

Calculate how much storage space is needed to record a **16-bit, 44.1khz, stereo** music for a duration of **30 seconds**.

$$\begin{aligned}\text{Sound file size} &= 44100 \times 2 \times 2 \times 30 \\ &= 5292000 \text{ bytes}\end{aligned}$$

Tính kích thước file âm thanh

Sound File Size = **Sample rate x sample size x channel x duration**

Example 2:

Calculate how much storage space is needed to record a **8-bit, 11khz, mono** sound for a duration of **10 seconds**.

$$\begin{aligned}\text{Sound file size} &= 11000 \times 1 \times 1 \times 10 \\ &= 110000 \text{ bytes}\end{aligned}$$

Guidelines

- Consider the appropriateness of using sound.
- Start with the highest-quality sound available and reduce the file size by converting the audio file to a compressed format.
- Consider using sound and still images as an alternative to video to reduce file sizes.
- If appropriate, provide a way to give the user some control over the audio.

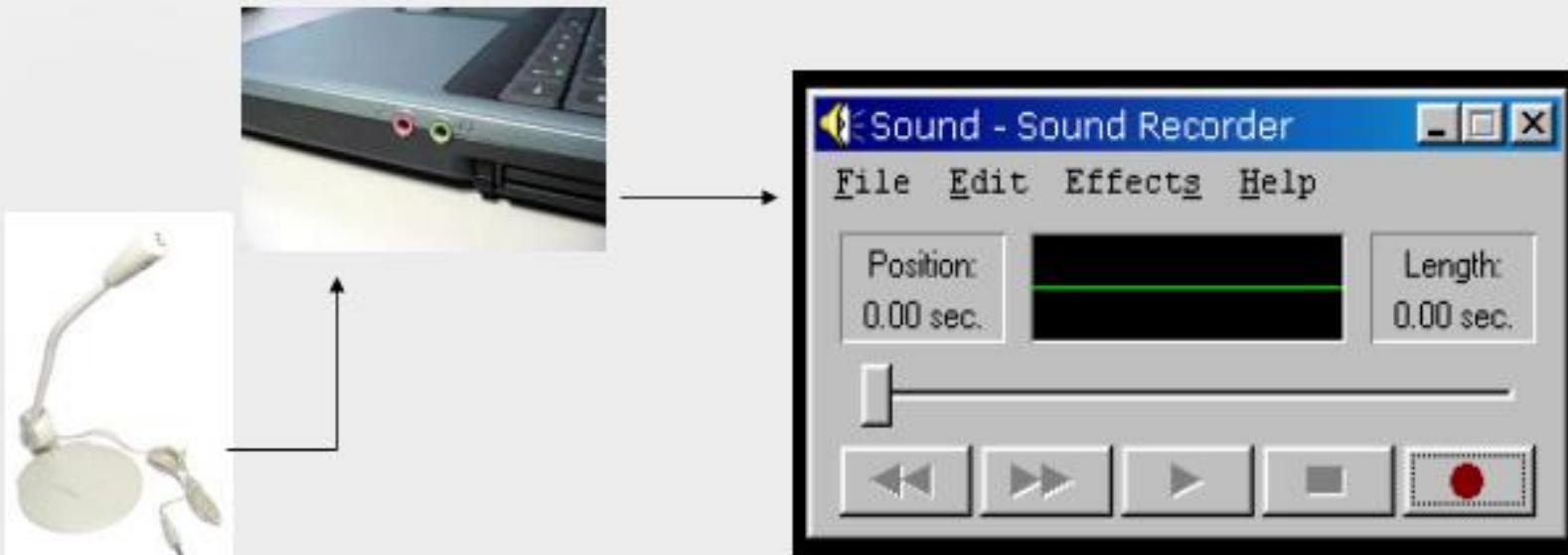
Ghi file âm thanh

Recording Audio Files on the pc

Uses either:

i. Microphone

- connect microphone to the microphone port and record using sound recorder



ii. CD-ROM Drive

- Move music files from CD to hard drive or;
- Play the cd and then record using the sound recorder.



iii. Line-in

- pressing play on the audio source, which is connected to the computer's audio line-in socket.
Record using the sound recorder.



Audio cable



Line in port
on the pc

Midi Audio

❖ **Musical Instrument Digital Interface**

- ❖ Before there was a wide use of mp3 and high bandwidth network, MIDI format audio is popular when an audio is required to be put on a website.
- ❖ Provides a standardized and efficient means of conveying musical performance information as electronic data.
- ❖ Is a easiest and quickest way to compose our own score.
 - (provided we have knowledge of musical instrument and composing)
- ❖ It is in the form of music score and not samples or recording.



Midi Audio: Requirements

- To make MIDI score, we need:
 1. Midi keyboard / Midi keyboard software
 2. Sequencer software
 3. Sound synthesizer (built-in in to sound card)

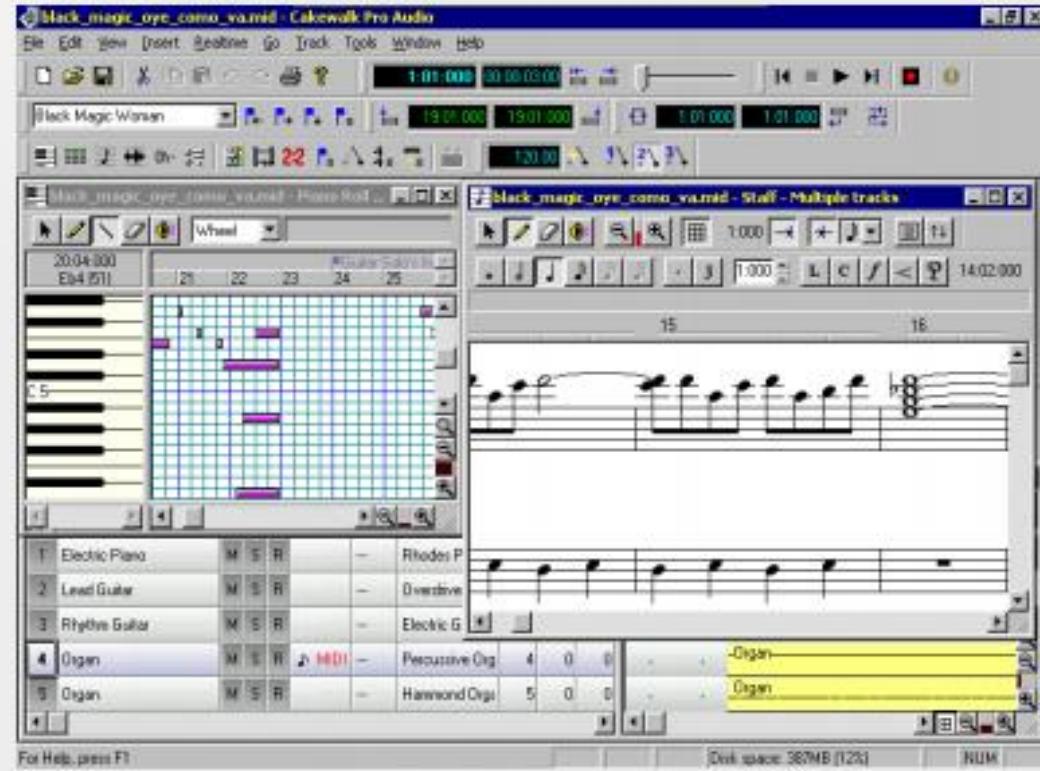
Midi Keyboard

- ❖ MIDI keyboard is used to simplify the creation of music scores (MIDI information)
 - ❖ MIDI information is transmitted in "MIDI messages", which can be thought of as instructions which tell a music synthesizer how to play a piece of music.
 - ❖ The synthesizer receiving the MIDI data must generate the actual sounds.



Midi Sequencer

- A **MIDI sequencer** software lets us to record and edit MIDI data like a word processor
 - Cut and paste
 - Insert / delete



Midi Audio Facts

- Since they are small, MIDI files embedded in web pages load and play.
- Length of a MIDI file can be changed without affecting the pitch of the music or degrading audio quality.
- Working with MIDI requires knowledge of music theory.



Recording MIDI Files

Recording MIDI Files

- ➊ MIDI files can be generated:
 - ➊ by recording the MIDI data from a MIDI instrument (electronic keyboard) as it is played.
 - ➊ by using a MIDI sequencer software application.

File Formats

- Wav audio (.wav)
- Sound (.snd)
- Real audio (.ra, rm)
- Audio File Format (.aif)
- MIDI (.mid)
- MP3 audio (.mp3)
- Windows Media (.wma)

MIDI versus Digital Audio

- ❖ Advantages of MIDI over digital audio:
 - ❖ MIDI files smaller than digital audio files.
 - ❖ Because small file, MIDI files embedded in web pages load and play more quickly.
 - ❖ If MIDI sound source are high quality – sound better.
 - ❖ Can change the length of MIDI files without changing the pitch of the music or degrading the audio quality.

MIDI versus Digital Audio

- ➊ Disadvantages of MIDI over digital audio:
 - ➊ Because MIDI data does not represent the sound but musical instruments, playback will be accurate only if the MIDI playback (instrument) is identical to the device used in the production.
 - ➊ Higher cost and requires skill to edit.
 - ➊ Cannot emulate voice, other effects.

Nén âm thanh

● Tại sao phải nén âm thanh ?

So Many Bits, So Little Time (Space)

- CD audio rate: $2 * 2 * 8 * 44100 = 1,411,200$ bps
- CD audio storage: 10,584,000 bytes / minute
- A CD holds only about 70 minutes of audio
- An ISDN line can only carry 128,000 bps

Security: *Best compressor removes all that is recognizable about the original sound*

Graphics people eat up all the space

Nén âm thanh

⦿ Nguyên lý của nén âm ?

Take advantage of

- Redundancy/Correlation
- Statistics (Local / Global)
- Assumptions / Models

*Problem: Much of this doesn't work
directly on sound waveform data*

Nén âm thanh

- ❖ Sound is difficult to compress using lossless methods, except for special cases.
- ❖ Some compression of audio can be obtained by run-length encoding samples that fall below a threshold that can be considered to represent silence.
- ❖ Companding uses non-linear quantization to compress speech.
- ❖ μ -law and A-law companding are used for telephony.

KỸ THUẬT NÉN AUDIO

- Cơ sở

Âm thanh trung thực và chất lượng dịch vụ thoả mãn thì tốc độ dòng dữ liệu phải lớn.

Ví dụ : Hệ thống âm thanh đa kênh mã hoá 16 bits, tần số lấy mẫu 48kHz (6 kênh) sẽ có tốc độ: $48 \times 16 \times 6 = 4.5 \text{ Mbps}$.

Tốc độ cao → Khó khăn lưu trữ, truyền dẫn và giá thành thiết bị. → Nén.

- Nén không tổn hao

Khôi phục đúng thông tin ban đầu sau khi giải nén.

Cơ sở: Loại bỏ dữ thừa thông kê, các thông tin xuất hiện trong tín hiệu mà có thể dự báo trước.

Tỷ số nén thấp, khoảng 2:1, phụ thuộc vào độ phức tạp của tín hiệu nguồn.

Thường sử dụng kỹ thuật mã hoá dự đoán trong miền thời gian.

KỸ THUẬT NÉN AUDIO

- **Nén tổn hao**

Hệ thống thính giác của con người không thể phân biệt các thành phần phổ có biên độ nhỏ giữa các thành phần phổ có biên độ lớn.

Hệ số nén lớn, khoảng 20:1 phụ thuộc vào quá trình nén và giải nén và chất lượng audio yêu cầu.

- **Các kỹ thuật được sử dụng:**

- Kỹ thuật che (masking) đối với các thành phần tín hiệu trong miền thời gian và tần số.

- Che mức tạp âm lượng tử cho từng âm độ của tín hiệu âm thanh bằng cách chỉ định số bit vừa đủ để chắc chắn rằng mức nhiễu lượng tử luôn nằm dưới mức giá trị cần che.

- Mã hoá ghép: Khai thác độ dư thừa trong hệ thống audio đa kênh với các thành phần số liệu trong các kênh giống nhau. Mã hoá một phần số liệu chung trên một kênh và chỉ định cho bộ giải mã lắp lại tín hiệu đó trên các kênh còn lại.

Chuẩn nén audio

- **MP3** (MPEG 1 layer 3): ra đời năm 1980 từ viện nghiên cứu Fraunhoufer Institute (Đức).
- **ACC**: Ra đời năm 1997 từ Fraunhofer Institutue (Đức) kết hợp với một số công ty như AT&T, Sony, Dolby, là định dạng cải tiến của MP3.
- **OGG**: Là định dạng nguồn mở được Xiph.org Foundation đề xuất năm 1993, nén tốt và có chất lượng ở tốc độ bit thấp.

Chuẩn nén audio

- **Realaudio:** Định dạng của công ty RealNetworks, chủ yếu dùng cho phát nhạc trực tuyến, định dạng đầu tiên ra đời năm 1995, đến nay đã có RealAudio 10
- **WMA:** Định dạng âm thanh của Microsoft, ra mắt năm 1999, trên lý thuyết có thể nén 96 kbps với chất lượng của MP3 128 kbps. WMA cũng phổ biến trong thế giới âm thanh phát trực tuyến.

Các giải thuật nén âm thanh

Nén không tổn thất

- Mã hóa Huffman
- Mã hóa Huffman sửa đổi
- Mã hóa số học
- Giải thuật Lempel – Ziv – Welch (LZW)

Các giải thuật nén âm thanh

Nén có tổn thất

- Các phương pháp nén âm thanh đơn giản:
LCP(Linear Predictive Coding)
CELP (Code Excited Linear Predictor)
- Nén âm thanh dùng mô hình âm – tâm lý (Psychoacoustics):
Hệ thống nghe và phát âm của con người
Che tần số
Băng giới hạn
Che nhất thời
- Nén âm thanh MPEG