Exercise: Audio Processing

Nama: Adin Adry
NIM: 12214004
Link Github: https://github.com/AdinAdryTjindarbumi/IF25-4035-Sistem-Teknologi-Multimedia
Mata Kuliah: Sistem Teknologi Multimedia

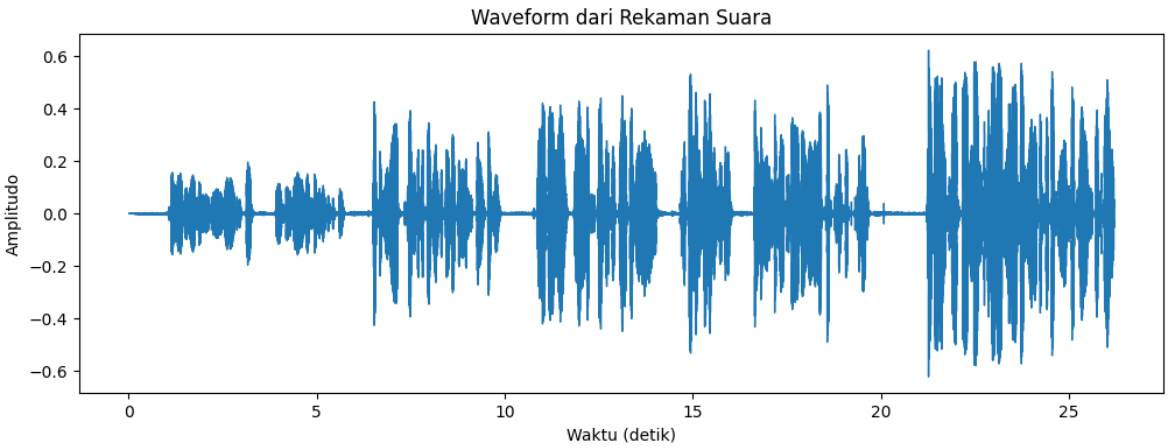
Soal 1: Rekaman dan Analisis Suara Multi-Level

1. Visualisasi waveform berdasarkan audio yang dibuat:

In [2]:
```
import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import Audio
import soundfile as sf
import os

# === 1. Load audio ===
audio_path = os.path.join(os.getcwd(), "audio", "Soal1.wav")
y, sr = librosa.load(audio_path, sr=None)  # sr=None agar sampling rate asli dip

print("Audio Asli:")
display(Audio(y, rate=sr))

# === 2. Visualisasi waveform ===
plt.figure(figsize=(12, 4))
librosa.display.waveshow(y, sr=sr)
plt.title("Waveform dari Rekaman Suara")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.show()
```

```
Audio Asli:
```

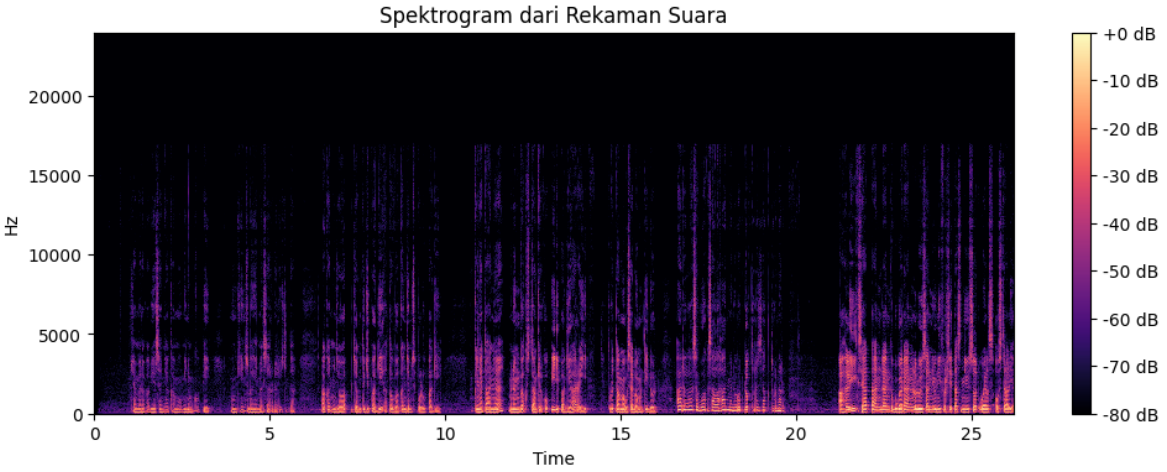
0:00 / 0:26



Analisis: Berdasarkan hasil waveform dari audio yang dibuat, dapat dilihat bahwa setiap 5 detik terdapat perubahan besar energi suara yang terus bertambah, yang dikarenakan perubahan jenis suara yang digunakan.

2. Visualisasi spectogram berdasarkan audio yang dibuat:

In [12]:
```
# === 3. Visualisasi spektrogram ===
D = np.abs(librosa.stft(y))  # Short-Time Fourier Transform
DB = librosa.amplitude_to_db(D, ref=np.max)

plt.figure(figsize=(12, 4))
librosa.display.specshow(DB, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
plt.colorbar(format="%+2.0f dB")
plt.title("Spektrogram dari Rekaman Suara")
plt.show()
```



Analisis: berdasarkan spectogram yang dihasilkan, dapat dilihat bahwa setiap 5 detik terdapat perubahan pada spektogram yaitu dari frekuensi suara yang semakin padat dan intensitas suara yang dihasilkan semakin tinggi.

3. Lakukan resampling pada file audio Anda kemudian bandingkan kualitas dan durasinya

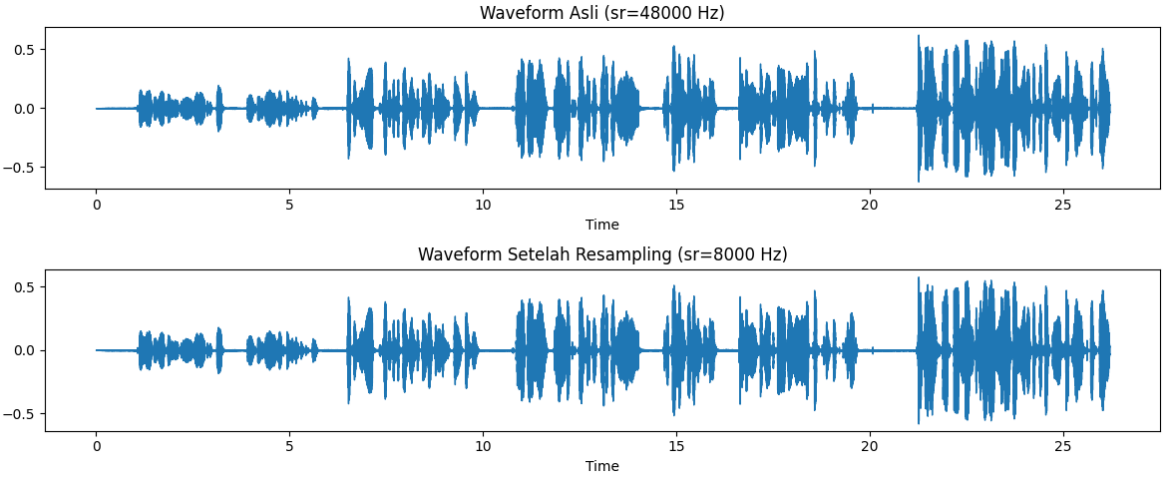
In [19]:
```
# Resampling ke 8000 Hz
new_sr = 8000
y_resampled = librosa.resample(y, orig_sr=sr, target_sr=new_sr)

# Hitung durasi
durasi_asli = len(y) / sr
durasi_resampled = len(y_resampled) / new_sr

print("Audio Setelah Resampling:")
display(Audio(y_resampled, rate=new_sr))

# === 2. Simpan kedua file untuk perbandingan ukuran ===
sf.write("audio_asli.wav", y, sr)
sf.write("audio_resampled.wav", y_resampled, new_sr)

# === 3. Hitung durasi dan ukuran file ===
durasi_asli = len(y) / sr
```

```
durasi_resampled = len(y_resampled) / new_sr

ukuran_asli = os.path.getsize("audio_asli.wav") / 1024  # KB
ukuran_resampled = os.path.getsize("audio_resampled.wav") / 1024  # KB

# === 4. Tampilkan hasil perbandingan ===
print("=== PERBANDINGAN AUDIO SEBELUM DAN SESUDAH RESAMPLING ===")
print(f"Sampling rate asli         : {sr} Hz")
print(f"Sampling rate setelah ubah : {new_sr} Hz")
print(f"Durasi asli                : {durasi_asli:.2f} detik")
print(f"Durasi setelah resampling  : {durasi_resampled:.2f} detik")
print(f"Ukuran file asli           : {ukuran_asli:.2f} KB")
print(f"Ukuran file setelah ubah   : {ukuran_resampled:.2f} KB")

# Visualisasi perbandingan waveform
plt.figure(figsize=(12, 5))

plt.subplot(2, 1, 1)
librosa.display.waveshow(y, sr=sr)
plt.title(f"Waveform Asli (sr={sr} Hz)")

plt.subplot(2, 1, 2)
librosa.display.waveshow(y_resampled, sr=new_sr)
plt.title(f"Waveform Setelah Resampling (sr={new_sr} Hz)")

plt.tight_layout()
plt.show()
```

Audio Setelah Resampling:

0:00 / 0:26

```
=== PERBANDINGAN AUDIO SEBELUM DAN SESUDAH RESAMPLING ===
Sampling rate asli         : 48000 Hz
Sampling rate setelah ubah : 8000 Hz
Durasi asli                : 26.20 detik
Durasi setelah resampling  : 26.20 detik
Ukuran file asli           : 2456.04 KB
Ukuran file setelah ubah   : 409.38 KB
```



Penjelasan: Disini saya melakukan resampling dari sample rate 48000hz menjadi 8000hz, suara dari audio yang di resampling menjadi kurang jelas/ memendam dibandingkan dengan audio aslinya, kemudian untuk durasi audio tetap sama yaitu 26.20 detik karena hanya sample rate yang diubah, dan untuk ukuran audio setelah resampling lebih kecil

dibanding yang asli.

Soal 2: Noice Reduction dengan Filtering

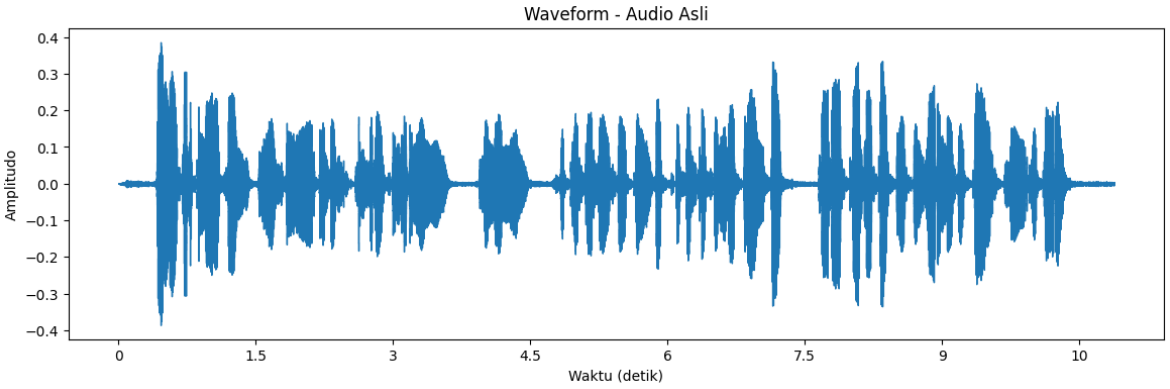
1. Menggunakan filter equalisasi (high-pass, low-pass, dan band-pass) untuk menghilangkan noise pada rekaman yang dibuat

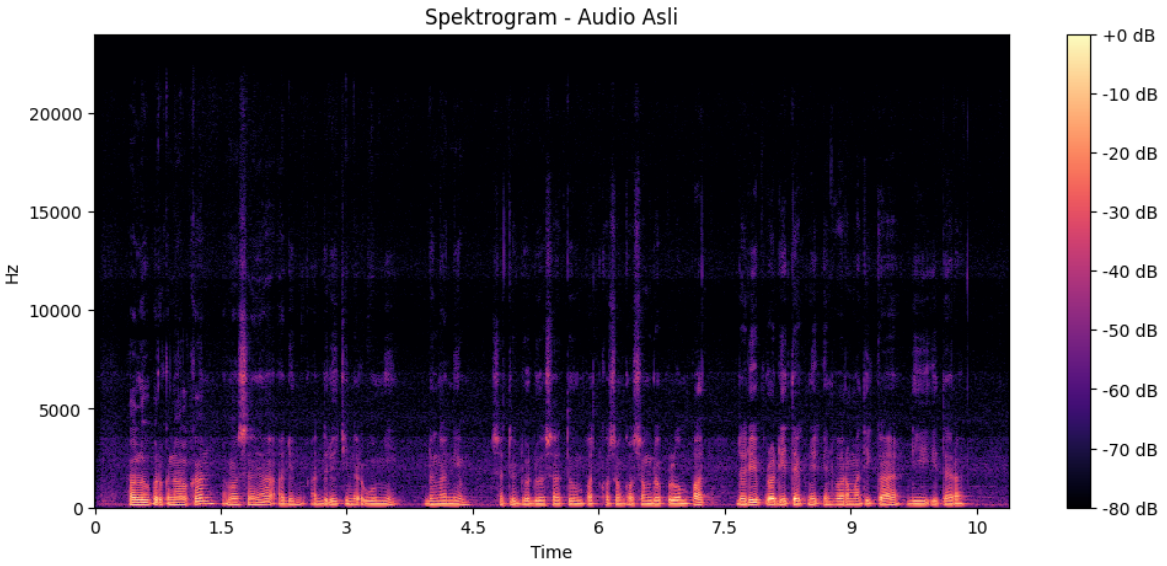
In [200…

```
from scipy.signal import butter, filtfilt
from IPython.display import Audio

# --- Load Audio ---
audio_path = os.path.join(os.getcwd(), "audio", "Soal2.wav")
y, sr = librosa.load(audio_path, sr=None)

print(f"Sampling rate: {sr} Hz")
print(f"Durasi: {len(y)/sr:.2f} detik")

# --- Tampilkan Audio Asli ---
print("Audio Asli:")
display(Audio(data=y, rate=sr))

plt.figure(figsize=(14, 4))
librosa.display.waveshow(y, sr=sr)
plt.title("Waveform - Audio Asli")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.show()

plt.figure(figsize=(12, 5))
D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
plt.colorbar(format="%+2.f dB")
plt.title("Spektrogram - Audio Asli")
plt.show()
```

```
Sampling rate: 48000 Hz
Durasi: 10.88 detik
Audio Asli:
```

0:00 / 0:11





- Membuat fungsi filter (Low-pass, High-pass, dan Band-pass)

In [124…

```python
def butter_filter(data, sr, cutoff, filter_type, order=5):
    nyquist = 0.5 * sr
    if filter_type == 'band':
        low = cutoff[0] / nyquist
        high = cutoff[1] / nyquist
        b, a = butter(order, [low, high], btype='band')
    else:
        normal_cutoff = cutoff / nyquist
        b, a = butter(order, normal_cutoff, btype=filter_type)
    return filtfilt(b, a, data)
```

- Menerapkan dan juga membandingkan filter

In [158…

```python
cutoffs = [100, 1800, 1900]

filters = {
    'Low-pass': lambda y, c: butter_filter(y, sr, c, 'low'),
    'High-pass': lambda y, c: butter_filter(y, sr, c, 'high'),
    'Band-pass': lambda y, c: butter_filter(y, sr, (300, 4000), 'band')
}

results = {}
for f_name, f_func in filters.items():
    if f_name == 'Band-pass':
        y_filt = f_func(y, None)
    else:
        y_filt = f_func(y, 4000)
    results[f_name] = y_filt

for f_name, y_filt in results.items():
    print(f"=== {f_name} Filter ===")
    display(Audio(y_filt, rate=sr))
```

```
=== Low-pass Filter ===
```

0:00 / 0:11

```
=== High-pass Filter ===
```

0:00 / 0:11

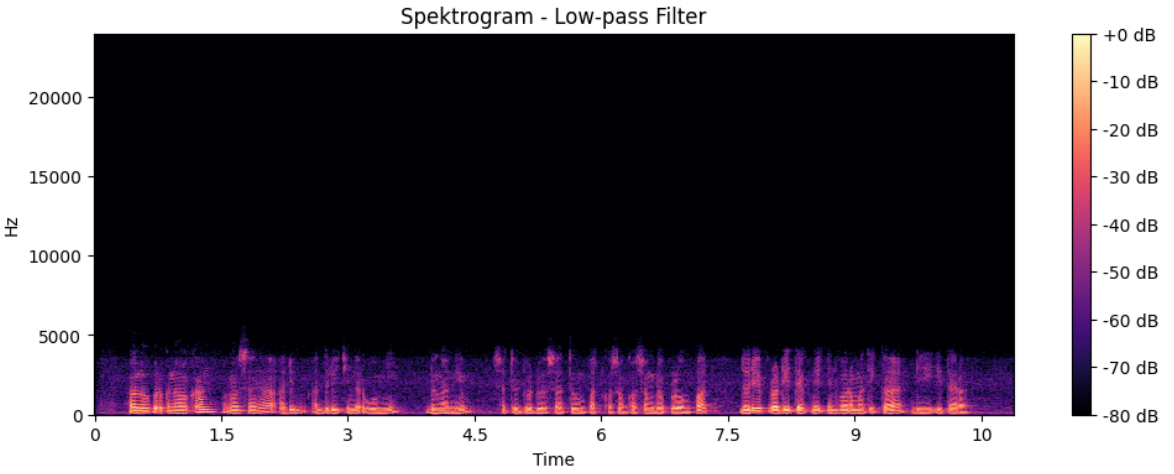
```
=== Band-pass Filter ===
```

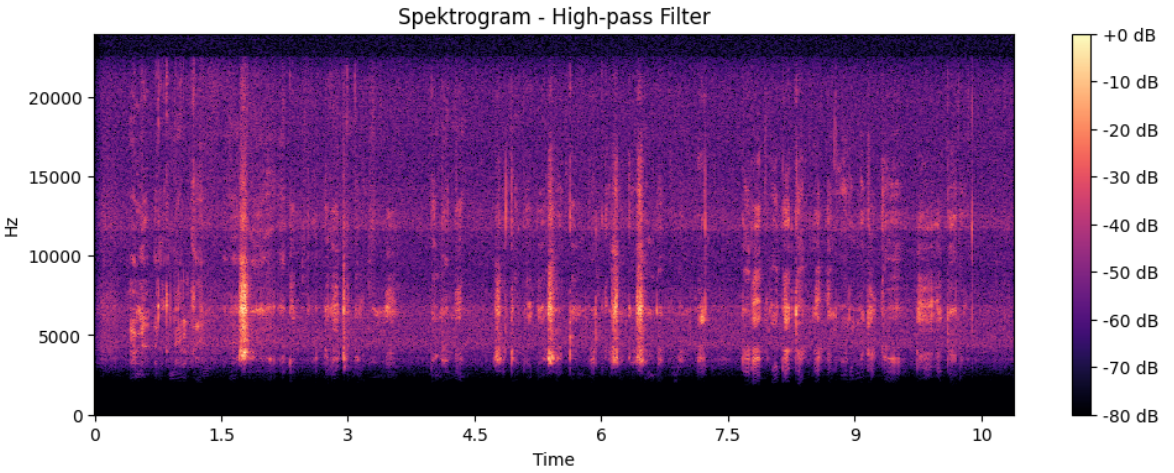
0:00 / 0:11

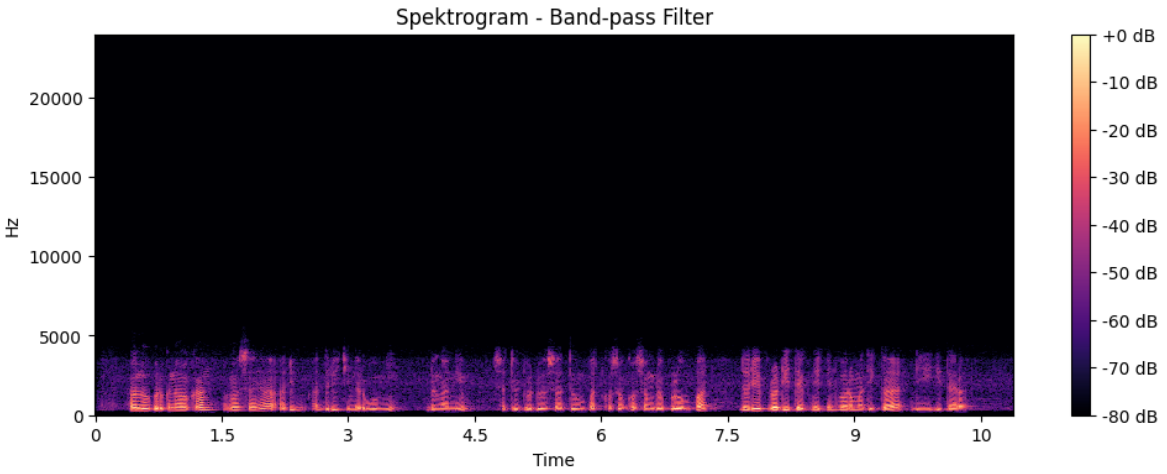
- Visualisasi Spektogram dari setiap filter

In [159…

```python
for f_name, y_filt in results.items():
    plt.figure(figsize=(12, 4))
    D = librosa.amplitude_to_db(np.abs(librosa.stft(y_filt)), ref=np.max)
    librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
    plt.colorbar(format="%+2.f dB")
    plt.title(f"Spektrogram - {f_name} Filter")
    plt.show()
```







Penjelasan:

- Jenis noise yang muncul dari bakcsound rekaman suara saya berasal dari suara kipas angin
- Filter yang paling cocok untuk menghilangkan/meredam noise dari rekaman tersebut adalah low-pass filter yang memotong frekuensi tinggi dari rekaman suara tersebut.
- Nilai Cutoff terbaik yang saya coba adalah 4000 untuk low-pass
- Perubahan kualitas suara yang dihasilkan lumayan terasa, walaupun noise dapat diredam dengan cukup baik, tetapi suara ucapan saya menjadi lumayan terpendam dibanding dengan audio aslinya.

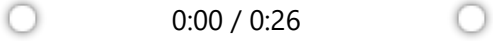
Soal 3: Pitch Shifting dan Audio Manipulation

- Lakukan pitch shifting pada rekaman suara Soal 1 untuk membuat suara terdengar seperti chipmunk (dengan mengubah pitch ke atas).

In [3]:
```
# --- Load Audio ---
audio_path = os.path.join(os.getcwd(), "audio", "Soal1.wav")
y, sr = librosa.load(audio_path, sr=None)

print(f"Sampling rate: {sr} Hz")
print(f"Durasi: {len(y)/sr:.2f} detik")

# --- Tampilkan audio asli ---
print("Audio Asli:")
Audio(y, rate=sr)
```

```
Sampling rate: 48000 Hz
Durasi: 26.20 detik
Audio Asli:
```

Out[3]:



- Melakukan pitchshifting dengan nilai +7 dan +12

In [ ]:
```
# Pitch shifting ke atas sebesar +7 dan +12 semitone
```

```
y_pitch7 = librosa.effects.pitch_shift(y, sr=sr, n_steps=7)
y_pitch12 = librosa.effects.pitch_shift(y, sr=sr, n_steps=12)

# Tampilkan hasil audio
print("Pitch Shift +7:")
display(Audio(y_pitch7, rate=sr))

print("Pitch Shift +12:")
display(Audio(y_pitch12, rate=sr))

sf.write("Soal3_pitch7.wav", y_pitch7, sr)
sf.write("Soal3_pitch12.wav", y_pitch12, sr)
```

```
Pitch Shift +7:
```

0:00 / 0:26

```
Pitch Shift +12:
```

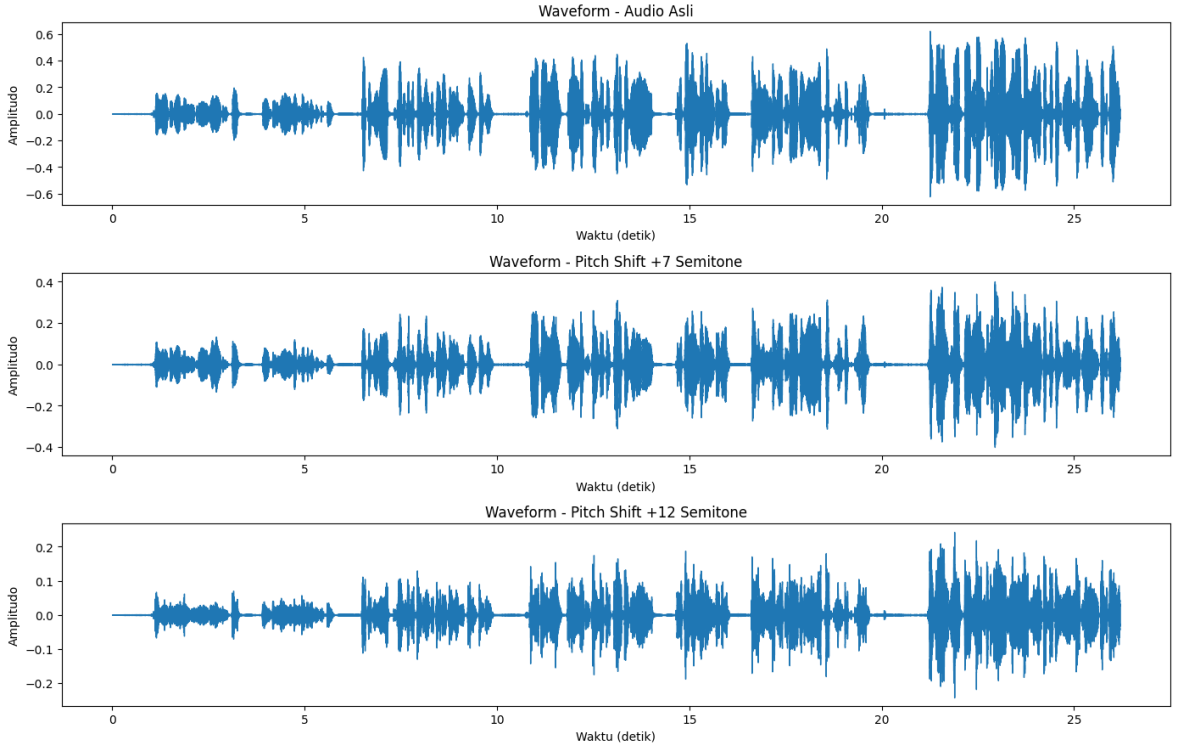
0:00 / 0:26

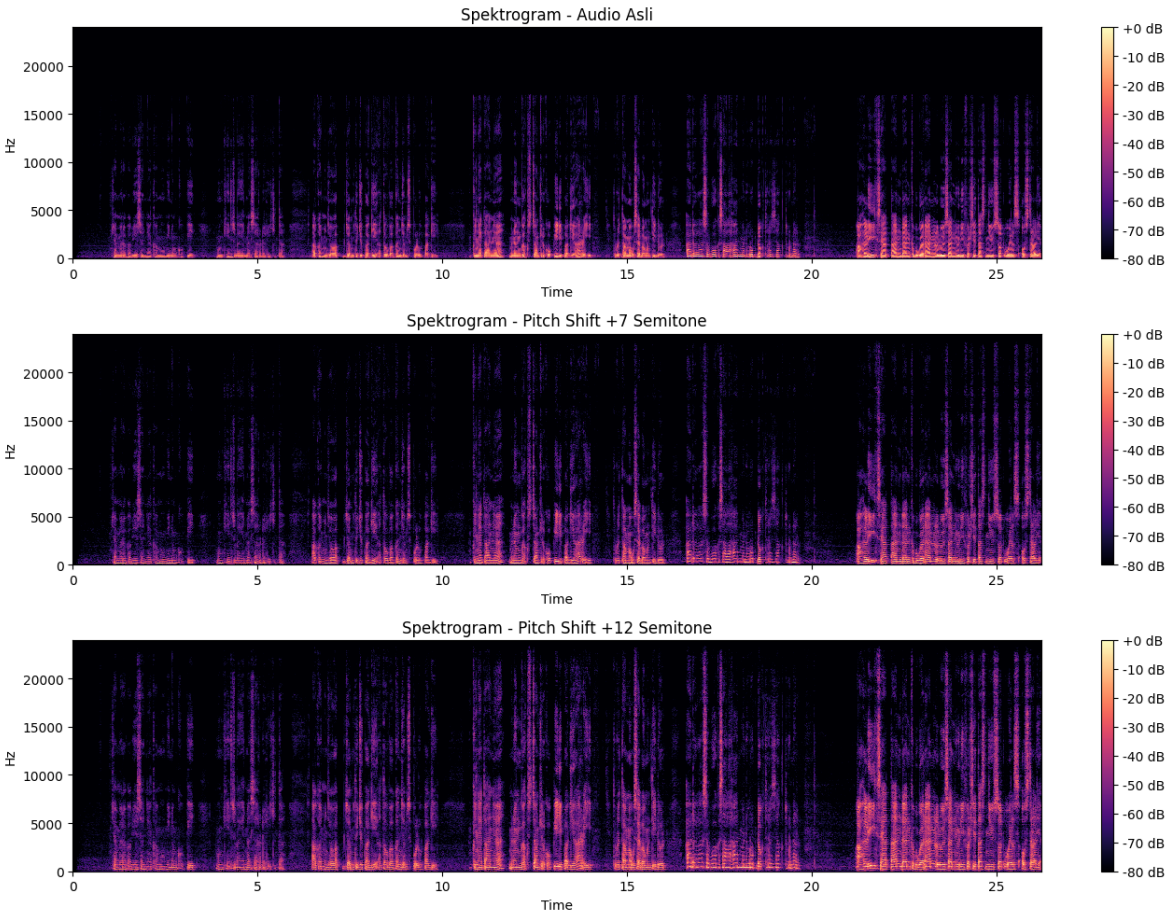
- Perbandingan Waveform dan Spektogram sebelum dan sesudah pitch shifting

In [164…

```
# --- Waveform Comparison ---
plt.figure(figsize=(14, 9))

plt.subplot(3, 1, 1)
librosa.display.waveshow(y, sr=sr)
plt.title("Waveform - Audio Asli")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

plt.subplot(3, 1, 2)
librosa.display.waveshow(y_pitch7, sr=sr)
plt.title("Waveform - Pitch Shift +7 Semitone")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

plt.subplot(3, 1, 3)
librosa.display.waveshow(y_pitch12, sr=sr)
plt.title("Waveform - Pitch Shift +12 Semitone")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

plt.tight_layout()
plt.show()

# --- Spektrogram Comparison ---
plt.figure(figsize=(14, 10))

# Asli
plt.subplot(3, 1, 1)
D_original = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
librosa.display.specshow(D_original, sr=sr, x_axis='time', y_axis='hz', cmap='ma
plt.title("Spektrogram - Audio Asli")
plt.colorbar(format="%+2.0f dB")

# Pitch +7
```

```
plt.subplot(3, 1, 2)
D_pitch7 = librosa.amplitude_to_db(np.abs(librosa.stft(y_pitch7)), ref=np.max)
librosa.display.specshow(D_pitch7, sr=sr, x_axis='time', y_axis='hz', cmap='magm
plt.title("Spektrogram - Pitch Shift +7 Semitone")
plt.colorbar(format="%+2.0f dB")

# Pitch +12
plt.subplot(3, 1, 3)
D_pitch12 = librosa.amplitude_to_db(np.abs(librosa.stft(y_pitch12)), ref=np.max)
librosa.display.specshow(D_pitch12, sr=sr, x_axis='time', y_axis='hz', cmap='mag
plt.title("Spektrogram - Pitch Shift +12 Semitone")
plt.colorbar(format="%+2.0f dB")

plt.tight_layout()
plt.show()
```





In [ ]:

Penjelasan proses pitch shifting yang dilakukan:

- Parameter yang digunakan adalah n_steps(semitone) berupa 7 dan 12 agar pitch suara naik sehingga menghasilkan suara chipmunk yang diinginkan, dengan menggunakan library librosa yaitu pich_shifft()
- Perbedaan visual dari sebelum dan sesudah melakukan pitch shifting, dapat dilihat di spektogram bahwa frekuensi suara yang dihasilkan semakin tinggi sebanding dengan besar angka pitch (semitone) yang dimasukkan.
- Perubahan pitch yang semakin tinggi membuat suara dari audio menjadi cempreng seperti "chipmunk" untuk audio dengan pitch shift +7 suara walaupun cempreng tetapi masih bisa diketahui apa yang diomongkan sedangkan pada +12 suara sudah tidak terlalu jelas.

Menggabungkan kedua rekaman yang telah di-pitch shift ke dalam satu file audio.

In [165…

```
# Gabungkan hasil pitch +7 dan +12 jadi satu file
combined = np.concatenate((y_pitch7, y_pitch12))

print("Gabungan Pitch +7 dan +12:")
Audio(combined, rate=sr)
```

```
Gabungan Pitch +7 dan +12:
```

Out[165…

0:00 / 0:52

Soal 4: Audio Processing Chain

1. MeLakukan processing pada rekaman yang sudah di-pitch shift pada Soal 3 dengan tahapan:

- Equalizer

In [186…
```
import pyloudnorm as pyln

# --- Load file hasil pitch shift sebelumnya ---
y, sr = librosa.load("Soal3_pitch7.wav", sr=None)

# --- Equalizer (High-pass filter untuk hilangkan hum rendah) ---
from scipy.signal import butter, filtfilt

def butter_filter(y, sr, cutoff, filter_type):
    nyq = 0.5 * sr
    norm_cutoff = np.array(cutoff) / nyq if isinstance(cutoff, (list, tuple)) el
    b, a = butter(4, norm_cutoff, btype=filter_type)
    return filtfilt(b, a, y)

y_eq = butter_filter(y, sr, 100, 'high')  # hilangkan noise di bawah 100 Hz
```

- Gain/Fade

In [187…
```
# --- Gain / Fade ---
# contoh: menaikkan gain sebesar 3 dB
gain_db = 3
y_gain = y_eq * (10 ** (gain_db / 20))
```

- Normalization

In [189…
```
y_norm_peak = y_gain / np.max(np.abs(y_gain))
```

- Compression

In [190…
```
y_compress = np.tanh(2 * y_norm_peak)
```

- Noise gate

In [191…
```
threshold = 0.02
y_gate = np.where(np.abs(y_compress) < threshold, 0, y_compress)
```

- Silence Trimming

In [192…
```
y_trimmed, _ = librosa.effects.trim(y_gate, top_db=30)
```

- Hasil dari Normalisasi Normal (Peak Normalization)

In [198…
```
display(Audio(y_trimmed, rate=sr))
sf.write("processed_final.wav", y_trimmed, sr)
```

0:00 / 0:25

- Atur Loudness ke -16 LUFS

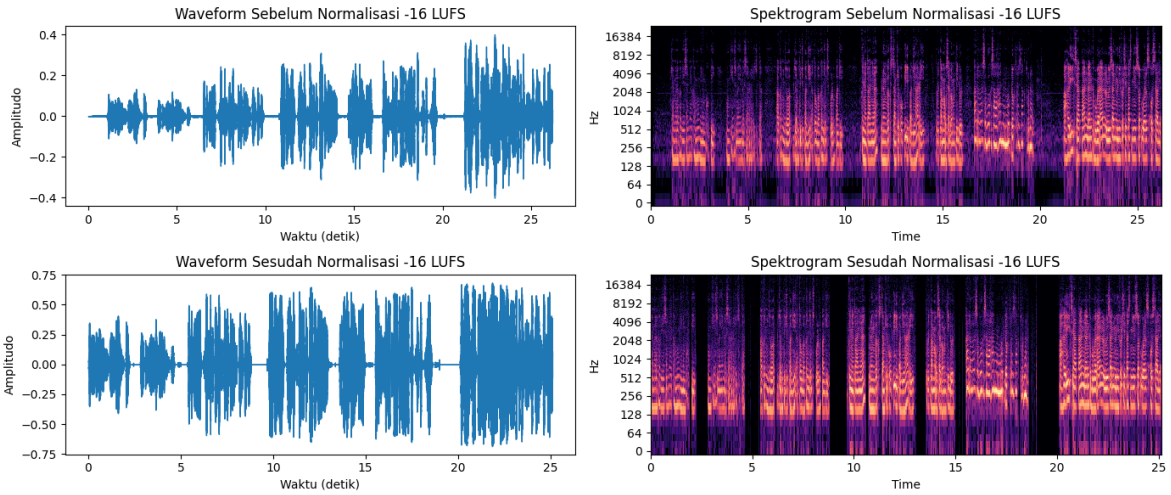
In [194…
```
import pyloudnorm as pyln

# Hitung dan ubah ke target loudness -16 LUFS
meter = pyln.Meter(sr)
loudness = meter.integrated_loudness(y_trimmed)
y_loud = pyln.normalize.loudness(y_trimmed, loudness, -16.0)

display(Audio(y_loud, rate=sr))
sf.write("normalized_LUFS_-16.wav", y_loud, sr)
```

0:00 / 0:25

- Visualisasi sebelum dan sesudah melakukan proses normalisasi

In [197…
```
y_raw, sr = librosa.load("Soal3_pitch7.wav", sr=None)

def plot_waveform_and_spectrogram(y_before, y_after, sr, title):
    plt.figure(figsize=(14, 6))

    # Waveform sebelum
    plt.subplot(2, 2, 1)
    librosa.display.waveshow(y_before, sr=sr)
    plt.title('Waveform Sebelum ' + title)
    plt.xlabel("Waktu (detik)")
    plt.ylabel("Amplitudo")

    # Spektrogram sebelum
    plt.subplot(2, 2, 2)
    spec_before = librosa.amplitude_to_db(np.abs(librosa.stft(y_before)), ref=np
    librosa.display.specshow(spec_before, sr=sr, x_axis='time', y_axis='log', cm
    plt.title('Spektrogram Sebelum ' + title)

    # Waveform sesudah
    plt.subplot(2, 2, 3)
    librosa.display.waveshow(y_after, sr=sr)
    plt.title('Waveform Sesudah ' + title)
    plt.xlabel("Waktu (detik)")
    plt.ylabel("Amplitudo")

    # Spektrogram sesudah
    plt.subplot(2, 2, 4)
    spec_after = librosa.amplitude_to_db(np.abs(librosa.stft(y_after)), ref=np.m
    librosa.display.specshow(spec_after, sr=sr, x_axis='time', y_axis='log', cma
    plt.title('Spektrogram Sesudah ' + title)

    plt.tight_layout()
    plt.show()
```

```
plot_waveform_and_spectrogram(y_raw, y_loud, sr, "Normalisasi -16 LUFS")
```



Penjelasan:

- Perubahan dinamika, dari yang saya amati, suara bagian awal yang berbisik diperkeras, kemudian untuk suara yang memiliki volume yang terlalu keras (saat suara keras & berteriak) itu dibuat menjadi seperti menjadi suara normal.
- Perbedaan dari peak normalization dengan LUFS normalization, dari yang saya dengar, terdapat perbedaan konsistensi volume suara, dengan LUFS normalization menurut saya memiliki volume suara yang lebih seimbang dibanding dengan peak normalization
- Suara yang dihasilkan setelah melakukan poses normalisasi peak dan LUFS menjadi lebih jelas dan seimbang dibanding dengan sebelum melakukan proses normalisasi.
- Kelebihan dari Loudness normalization yaitu suara yang dihasilkan lebih seimbang, dengan kekurangannya suara dapat menjadi terlalu datar karena suara terlalu seimbang.

Soal 5: Music Analysis dan Remix

- Memuat file audio lagu 1 dan lagu 2 yang digunakan

In [8]:
```
# Tentukan path file Anda
file_lagu1 = os.path.join(os.getcwd(), "lagu", "lagu_sedih.wav")
file_lagu2 = os.path.join(os.getcwd(), "lagu", "lagu_senang.wav")

# Muat file audio
# Kita tentukan sr (sampling rate) agar konsisten
TARGET_SR = 44100
y1, sr1 = librosa.load(file_lagu1, sr=TARGET_SR)
y2, sr2 = librosa.load(file_lagu2, sr=TARGET_SR)

print(f"Lagu 1 ('{file_lagu1}') dimuat dengan sr={sr1}")
display(Audio(y1, rate=sr1))
print(f"Lagu 2 ('{file_lagu2}') dimuat dengan sr={sr2}")
display(Audio(y2, rate=sr2))
```

```
Lagu 1 ('g:\My Drive\SEMESTER 7\MULTIMEDIA\LATIHAN\Latihan 4\lagu\lagu_sedih.wa
v') dimuat dengan sr=44100
```

0:00 / 1:00

```
Lagu 2 ('g:\My Drive\SEMESTER 7\MULTIMEDIA\LATIHAN\Latihan 4\lagu\lagu_senang.wa
v') dimuat dengan sr=44100
```

0:00 / 1:00

- Deteksi Tempo (BPM) dan kunci (Key)

In [13]:
```python
# 1. Deteksi Tempo (BPM)
tempo1 = librosa.feature.rhythm.tempo(y=y1, sr=sr1)[0]
tempo2 = librosa.feature.rhythm.tempo(y=y2, sr=sr2)[0]

# 2. Estimasi Kunci (Key)
# Kita akan menggunakan chromagram untuk melihat pitch class yang dominan
chroma1 = librosa.feature.chroma_stft(y=y1, sr=sr1)
chroma2 = librosa.feature.chroma_stft(y=y2, sr=sr2)

# Rata-ratakan chromagram di sepanjang waktu untuk menemukan kunci dominan
key_vector1 = np.sum(chroma1, axis=1)
key_vector2 = np.sum(chroma2, axis=1)

# Dapatkan indeks dari nada yang paling dominan
key_idx1 = np.argmax(key_vector1)
key_idx2 = np.argmax(key_vector2)

# Konversi indeks ke nama nada
notes = ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B']
key1 = notes[key_idx1]
key2 = notes[key_idx2]

# Tampilkan hasil analisis
print("--- Analisis Lagu 1 (Sedih, Lambat) ---")
print(f"Estimasi Tempo (BPM): {tempo1:.2f}")
print(f"Estimasi Kunci Nada: {key1}")

print("\n--- Analisis Lagu 2 (Ceria, Cepat) ---")
print(f"Estimasi Tempo (BPM): {tempo2:.2f}")
print(f"Estimasi Kunci Nada: {key2}")
```

```
--- Analisis Lagu 1 (Sedih, Lambat) ---
Estimasi Tempo (BPM): 123.05
Estimasi Kunci Nada: G

--- Analisis Lagu 2 (Ceria, Cepat) ---
Estimasi Tempo (BPM): 109.96
Estimasi Kunci Nada: B
```

Analisis: Berdasarkan hasil deteksi tempo dengan menggunakan fitur dari library librosa, lagu 1 yang harusnya lambat malah memiliki BPM/Tempo yang lebih banyak dibanding dengan lagu 2, padahal lagu 2 yang saya pilih memiliki tempo yang cepat jika didengar secara langsung. Kemudian untuk Estimasi Kunci Nada yang dimiliki oleh lagu 1 (sedih, lambat) adalah "G" sedangkan lagu 2 (ceria, cepat) adalah "B".

Proses Remix:

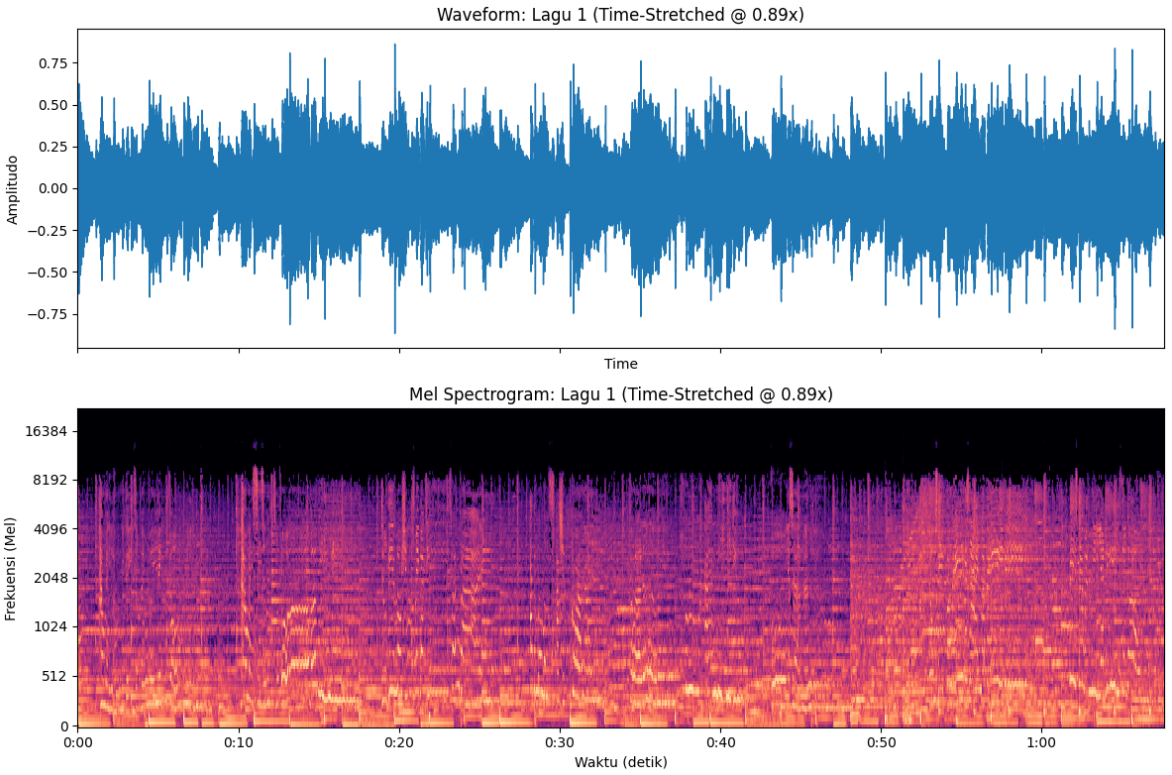
- Time Stretch

In [17]:
```
stretch_rate = tempo2 / tempo1
y1_stretched = librosa.effects.time_stretch(y1, rate=stretch_rate)

print(f"Lagu 1 di-time-stretch dengan faktor: {stretch_rate:.2f}")
display(Audio(y1_stretched, rate=sr1))
```

```
Lagu 1 di-time-stretch dengan faktor: 0.89
```

0:00 / 1:08

In [23]:
```
fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(12, 8), sharex=True)

# 2a. Plot Waveform
librosa.display.waveshow(y1_stretched, sr=sr1, ax=ax[0])
ax[0].set_title(f'Waveform: Lagu 1 (Time-Stretched @ {stretch_rate:.2f}x)')
ax[0].set_ylabel('Amplitudo')

# 2b. Plot Mel Spectrogram
S_stretched = librosa.feature.melspectrogram(y=y1_stretched, sr=sr1, n_mels=128)
S_db_stretched = librosa.power_to_db(S_stretched, ref=np.max)
librosa.display.specshow(S_db_stretched, sr=sr1, x_axis='time', y_axis='mel', ax
ax[1].set_title(f'Mel Spectrogram: Lagu 1 (Time-Stretched @ {stretch_rate:.2f}x)
ax[1].set_ylabel('Frekuensi (Mel)')
ax[1].set_xlabel('Waktu (detik)')

plt.tight_layout()
plt.show()
```



Penjelasan: Pada remix time stretch ini, menggunakan parameter tempo1 dan tempo2 yang didapat dari BPM lagu 1 dan lagu 2, kemudian dibagi untuk mendapatkan stretch rate lalu diproses dengan menggunakan fungsi time_stretch yang ada di library librosa,

untuk lagu yang diremix dengan time stretch adalah lagu 1.

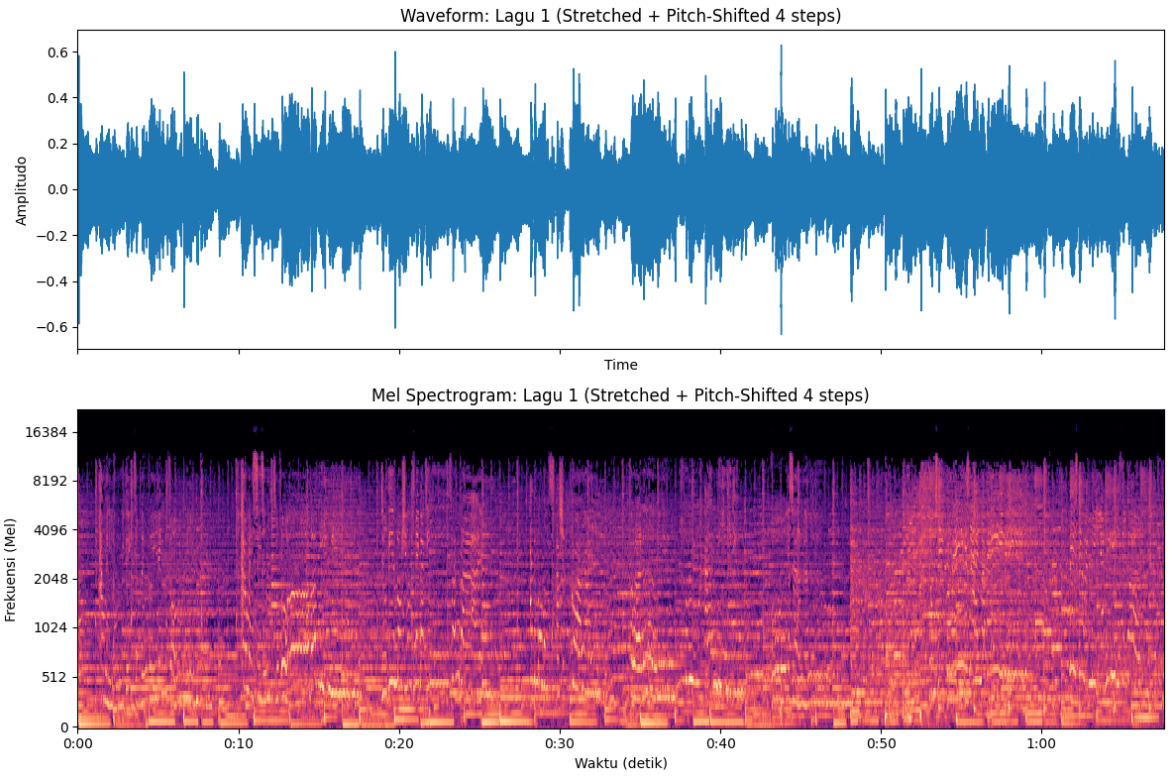
- Pitch Shift Lagu 1 (yang sudah di-stretch)

In [16]:
```
diff = key_idx2 - key_idx1

# Pilih jalur terpendek (misal: dari A ke C lebih dekat naik 3 semitone daripada
if diff > 6:
    n_steps = diff - 12  # Turun nada
elif diff < -6:
    n_steps = diff + 12  # Naik nada
else:
    n_steps = diff

y1_remix = librosa.effects.pitch_shift(y1_stretched, sr=TARGET_SR, n_steps=float

print(f"Lagu 1 di-pitch-shift sebanyak: {n_steps} semitone (dari {key1} ke {key2
display(Audio(y1_remix, rate=TARGET_SR))
```

```
Lagu 1 di-pitch-shift sebanyak: 4 semitone (dari G ke B)
```

0:00 / 1:08

In [24]:
```
fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(12, 8), sharex=True)

# 2a. Plot Waveform
librosa.display.waveshow(y1_remix, sr=TARGET_SR, ax=ax[0])
ax[0].set_title(f'Waveform: Lagu 1 (Stretched + Pitch-Shifted {n_steps} steps)')
ax[0].set_ylabel('Amplitudo')

# 2b. Plot Mel Spectrogram
S_remix = librosa.feature.melspectrogram(y=y1_remix, sr=TARGET_SR, n_mels=128)
S_db_remix = librosa.power_to_db(S_remix, ref=np.max)
librosa.display.specshow(S_db_remix, sr=TARGET_SR, x_axis='time', y_axis='mel',
ax[1].set_title(f'Mel Spectrogram: Lagu 1 (Stretched + Pitch-Shifted {n_steps} s
ax[1].set_ylabel('Frekuensi (Mel)')
ax[1].set_xlabel('Waktu (detik)')

plt.tight_layout()
plt.show()
```



Penjelasan: Remix pitch shift, menggunakan parameter n_steps (semitone) dengan proses apabila perbedaan antara key dari lagu 1 dan 2 lebih dari 6 maka nada akan dikurangi, sedangkan apabila kurang dari 6 maka nada akan ditambah, untuk menyesuaikan pitch yang ada dari kedua lagu agar menjadi senada dan tidak fals

- Crossfading kedua lagu

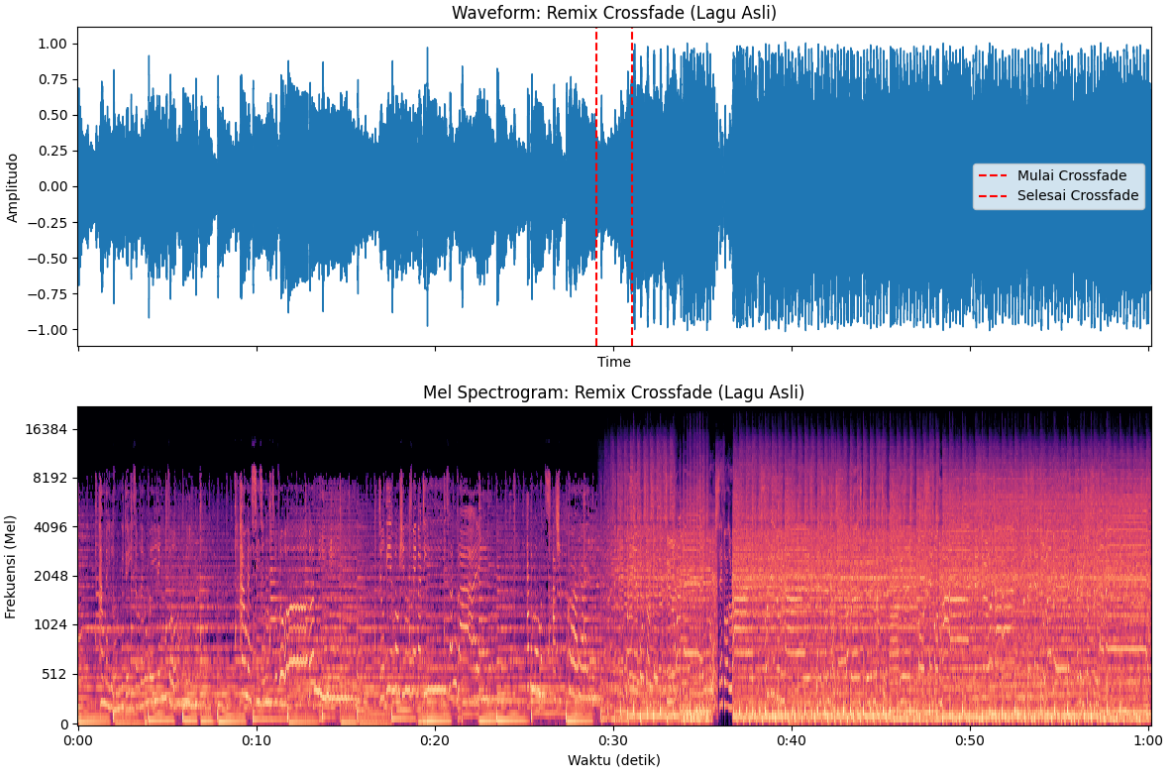
In [22]:
```
# Samakan panjang kedua track asli agar bisa digabung
# Kita potong track yang lebih panjang agar sama dengan yang lebih pendek
len1 = len(y1)  # Menggunakan y1 asli
len2 = len(y2)  # Menggunakan y2 asli
min_len = min(len1, len2)

y1_final = y1[:min_len]
y2_final = y2[:min_len]

# Kita akan buat transisi 2 detik di tengah-tengah
fade_duration_sec = 2  # Durasi crossfade 2 detik
fade_samples = librosa.time_to_samples(fade_duration_sec, sr=TARGET_SR)

# Tentukan titik tengah untuk transisi
mid_point = min_len // 2
start_fade = mid_point - (fade_samples // 2)
end_fade = mid_point + (fade_samples // 2)

# Buat kurva fade in dan fade out
fade_out_curve = np.linspace(1, 0, fade_samples)
fade_in_curve = np.linspace(0, 1, fade_samples)

# Buat track remix kosong
y_remix = np.zeros(min_len)

# Bagian 1: Hanya Lagu 1 (Asli)
```

```
y_remix[:start_fade] = y1_final[:start_fade]

# Bagian 2: Crossfade
y_remix[start_fade:end_fade] = \
    (y1_final[start_fade:end_fade] * fade_out_curve) + \
    (y2_final[start_fade:end_fade] * fade_in_curve)

# Bagian 3: Hanya Lagu 2 (Asli)
y_remix[end_fade:] = y2_final[end_fade:]

# Simpan hasil remix ke file baru
sf.write('remix_crossfade_asli.wav', y_remix, TARGET_SR)
display(Audio(y_remix, rate=TARGET_SR))
print(f"Remix (Crossfade Asli) selesai! Disimpan sebagai 'remix_crossfade_asli.w
```

0:00 / 1:00

```
Remix (Crossfade Asli) selesai! Disimpan sebagai 'remix_crossfade_asli.wav'
```

In [25]:
```
fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(12, 8), sharex=True)

# 2a. Plot Waveform
librosa.display.waveshow(y_remix, sr=TARGET_SR, ax=ax[0])
ax[0].set_title('Waveform: Remix Crossfade (Lagu Asli)')
ax[0].set_ylabel('Amplitudo')
# Tambahkan garis vertikal untuk menandai area crossfade
ax[0].axvline(x=librosa.samples_to_time(start_fade, sr=TARGET_SR), color='r', li
ax[0].axvline(x=librosa.samples_to_time(end_fade, sr=TARGET_SR), color='r', line
ax[0].legend()

# 2b. Plot Mel Spectrogram
S_remix = librosa.feature.melspectrogram(y=y_remix, sr=TARGET_SR, n_mels=128)
S_db_remix = librosa.power_to_db(S_remix, ref=np.max)
librosa.display.specshow(S_db_remix, sr=TARGET_SR, x_axis='time', y_axis='mel',
ax[1].set_title('Mel Spectrogram: Remix Crossfade (Lagu Asli)')
ax[1].set_ylabel('Frekuensi (Mel)')
ax[1].set_xlabel('Waktu (detik)')

plt.tight_layout()
plt.show()
```



Penjelasan: Remix ini menggunakan parameter titik tengah dari kedua lagu untuk dijadikan awalan dan akhiran dari crossfading dimana di area tersebut akan dilakukan faading sebagai transisi pergantian dari kedua lagu

Referensi

1. Chat GPT: https://chatgpt.com/share/68f22a1d-b57c-8003-8e75-b6613c790a23
2. Gemini: https://gemini.google.com/share/22bec61c32a2