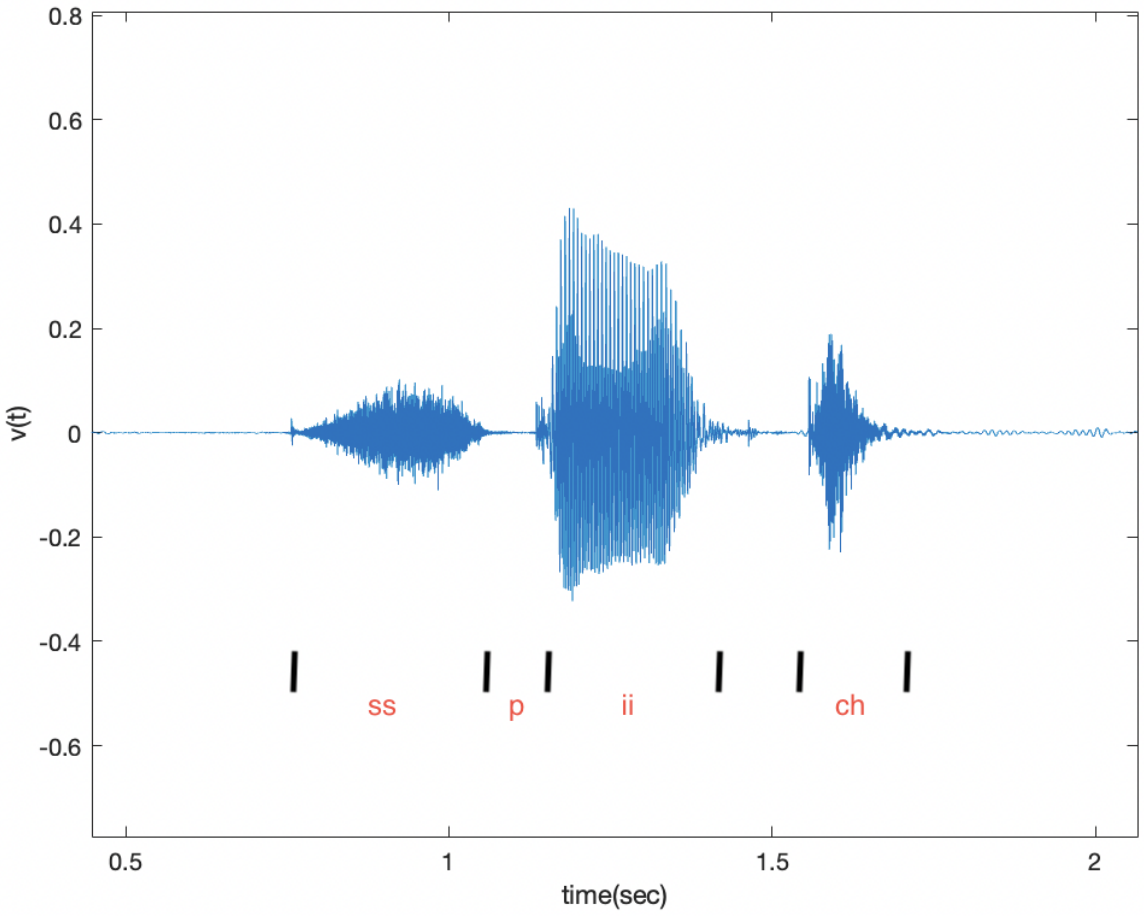
Speech Processing
Lab-3 Report

Rajat Tyagi 180020029

Part 0 :

Recording the phrase “Speech”:
Recording specifications : Fs = 44.1 kHz and Bit resolution = 16.
Recorded audio plot



Part A :

Sampling Rate Study by keeping bit resolution constant (16 bits/sample)

Theory:

The sampling of analog signals is based on the sampling theorem.
The sampling theorem states that if f_m is the maximum frequency component of the signal, then the signal can be represented by its sampled version such that the number of samples taken per second is greater than or equal to twice the maximum frequency component present in the signal. This term samples/second is known as sampling frequency (F_s). According to the sampling theorem, F_s >= 2 f_m.Human speech has frequency components from approximately 20 Hz and 20 kHz. By using the above stated sampling theorem we can say that to gain complete information of the human speech we need to sample it at about 40 kHz.
Normally we take it as 44.1 kHz to allow some guard band over the required 40 kHz.

Procedure:

1. First record the phrase “Speech” with a sampling frequency of 44.1 kHz and Bit resolution = 16 bits/sample.
2. Then use the Lab3Plots() function which takes the new sampling rate and bit resolution as input and plots the spectrum for each sound in the audio for the given sampling frequency and bit resolution (in this part we keep the bit resolution constant).

Code:

The code consists of three functions namely
(i) Lab3Plots() : for plotting all the spectrums.
(ii) bitsPerSample() : to change the bit resolution. (not useful in this part)
(iii) speechfft() : to compute the fft of the given parts of audio.

```
function [] = Lab3Plots(Fs, bits
   %Original audio Bits/sample = 16 and Fs = 44.1k H
   [y,fs] = audioread('speech.wav')

   %time duration of the given audio file
   time_duration = length(y)/fs;

   %generating the time axis
   t = 0 : 1/Fs : time_duration - 1/Fs;

   %Resample to given Fs
   y = resample(y,Fs,fs);

   %reducing Bits per sample to given bitsPerSample
   y_new = bitsPerSample(y,bits);

   %extracting different parts of the phrase speech
   ss_sound = y_new(0.75*Fs : 1.00*Fs);

   p_sound = y_new(1.06*Fs : 1.15*Fs);

   ii_sound = y_new(1.16*Fs : 1.4*Fs);

   ch_sound = y_new(1.55*Fs : 1.75*Fs);

   %plotting audio in time domain and spectrogram
   figure(1);
   plot(t,y_new);
   title("Audio Plot");
   figure(2);
   spectrogram(y_new,hamming(500),490,[],Fs,'yaxis')
   title("Audio with Bit resolution = " + num2str(bits)+ " and Fs = " + num2str(Fs));

   %fft of different sounds
   speechfft(ss_sound,Fs,"ss sound",bits,3);
   speechfft(p_sound,Fs,"p sound",bits,4);
   speechfft(ii_sound,Fs,"ii sound",bits,5);
   speechfft(ch_sound,Fs,"ch sound",bits,6);

end
```

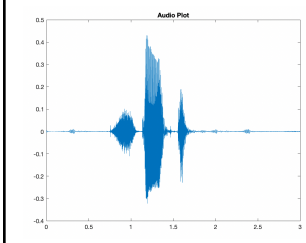
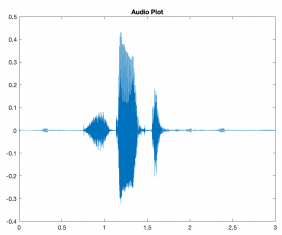
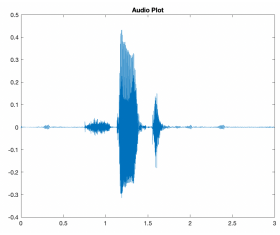
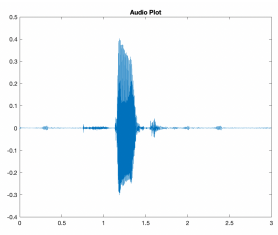
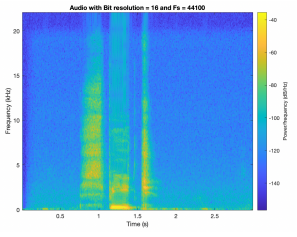
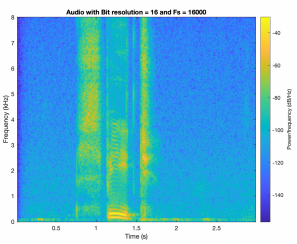
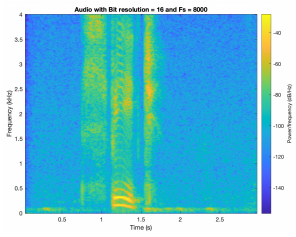
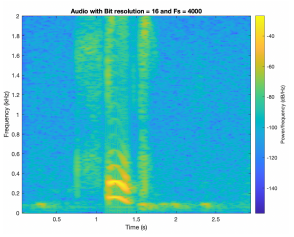
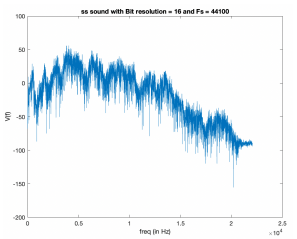
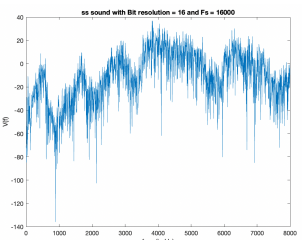
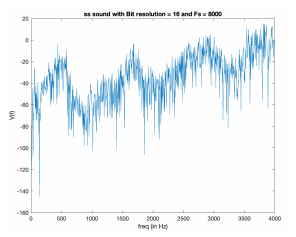
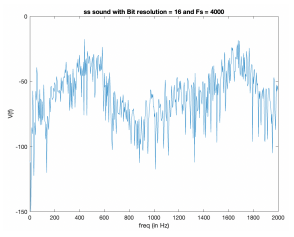
This code is self explanatory, it first reads the saved audio, resamples it to desired Fs, changes it to desired bit resolution, extracts different sounds, and computes the fft of each using speechfft() for plotting.

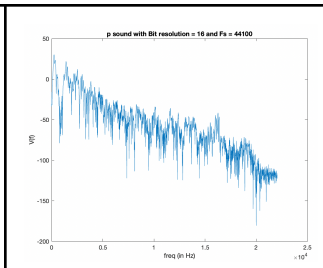
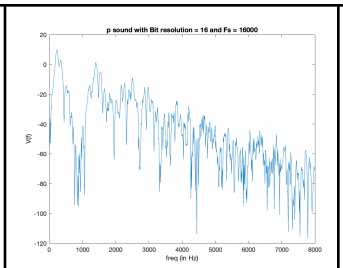
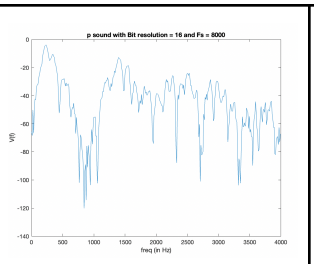
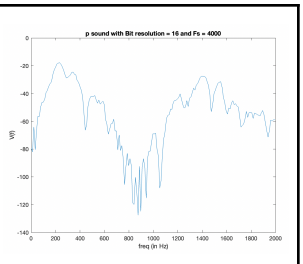
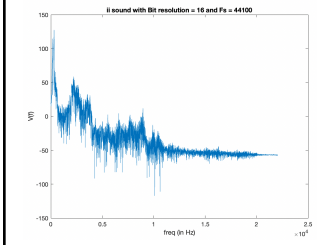
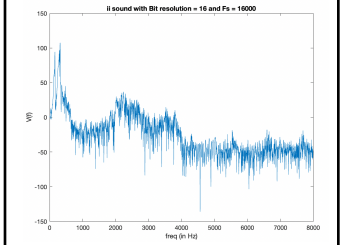
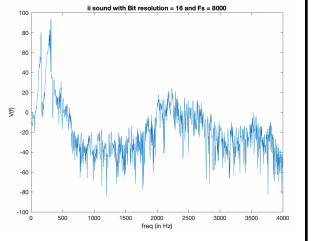
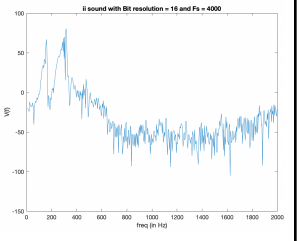
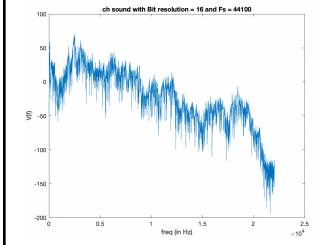
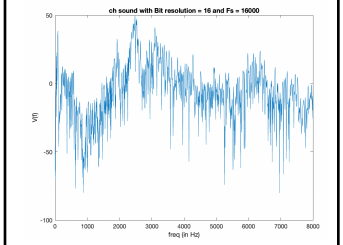
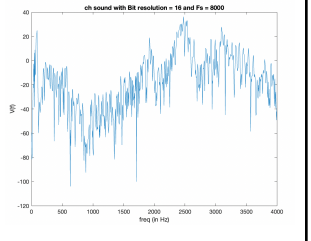
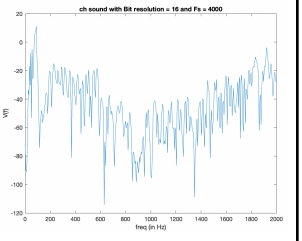
```
function [] = speechfft(x,Fs,text,bits,n)

   fftFull = fft(x);

   %length of the FFT of non stationary signal in frequency domain
   Len_f = length(fftFull);

   %Taking only +ve frequencies
   fftHalf = fftFull(1:round(Len_f/2));

   %converting in DB scale
   fftDB = 20*log(abs(fftHalf));

   %iterating freq from 0 to +len/2
   freq = 0 : 1 : round(Len_f/2) - 1;
```

```
    %converting each term of freq into frequency as for the 'k'th term f = Fs.k/N
    freq = Fs*freq/Len_f;

    %Plotting
    figure(n);
    plot(freq,fftDB);
    title(text + " with Bit resolution = " + num2str(bits) + " and Fs = " + num2str(Fs));
    xlabel("freq (in Hz)");
    ylabel("V(f)");

end
```

The above function computes and plots the fft using the inbuilt fft() function in matlab. Here we plot the frequency in DB scale, for that we just take the log of the fft array and multiply it by 20.

Plots:

	44.1 kHz (Original)	16 kHz	8 kHz	4 kHz
Audio Plot				
Spectrogram				
ss Spectrum				

p Spectrum				
ii Spectrum				
ch Spectrum				

Observations:

1. We observe that as we decrease the sampling frequency the ‘ss’ and ‘ch’ part of the audio in the audio plot become smaller and smaller.
2. We can also observe that frequency information that we have is from 0 to Fs/2.
3. On listening to the resampled audio there is hardly any difference between the audios with sampling rate 44.1 kHz and 16 kHz. 8 kHz sampled audio is also quite clear but not as much 16 or 44.1 kHz.
4. The audio with sampling frequency 4 kHz is a little bit different as ‘ss’ sound is not significant but still perceivable to me.
5. The spectrogram of the resampled audios are also very different, the formants of the periodic sounds are easy to observe in low sampling frequency.
6. The ‘ii’ and ‘p’ sounds have lower frequency components hence they are more intelligible than the fricatives ‘ss’ and ‘ch’.

Inference and conclusion:
We see a reduction in ‘ss’ and ‘ch’ in the audio plot as we decrease the sampling frequency because they are fricatives and have high frequency components and as the sampling frequency decreases the high frequency components get removed from their spectrum.

The audio quality for the sampling frequencies 44.1 kHz, 16 kHz and 8 kHz are almost same because as seen in the spectrum plots of all the most of the power is concentrated in the first 8 kHz of the signal, there are some fricatives which have some power beyond 8 kHz but it is not that significant, Hence we 2x8 kHz is the optimal sampling frequency as per sampling theorem.

In telecommunications bandwidth is too precious hence a lower sampling frequency is used. The telephone bandwidth is 8 kHz and is used as it gives a fairly good audio quality. In real world applications it is passed through an anti aliasing filter to cut off frequencies above 4 kHz.

Part B:

Bit Resolution Study by keeping Sampling Rate constant (16 kHz)

Theory:
In the above part we focussed on sampling the time axis, Bit resolution is a way to sample the amplitude of the signal.
The number of bits used for storing each sample of speech is termed as bit resolution. The number of bits/sample depends on the number of quantization levels used during analog to digital conversion. More the number of quantization levels, finer will be the quantization step and hence better will be the information preserved in the digitized form. The drawback is that if we need more precision we will have to store a higher number of bits per sample. Hence it is a trade off between the number of bits and information representation.

In the case of 16 bits/sample we can have 2^16 = 65536 quantization levels, whereas In the case of 8 bits/sample we can have 2^8 = 256 quantization levels.
As we can see the number of quantization levels decrease significantly as we reduce bit resolution from 16 to 8.
16 bits per sample is considered to be an optimal bit resolution to store any audio signal.

Procedure:

1. First of all we set a fixed sampling rate that is 16 kHz.
2. Now we use the same Lab3plot() function for plotting the output spectrums of each sound in the phrase “Speech”.
3. Lab3plot() takes the desired bit resolution as input and inturn uses another function to convert the bit resolution of the audio array that is bitsPerSample().

Code:

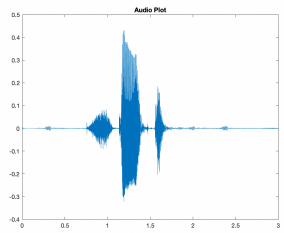
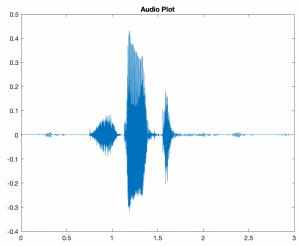
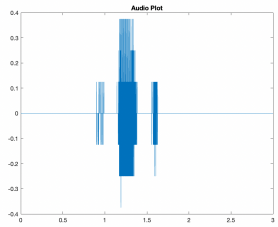
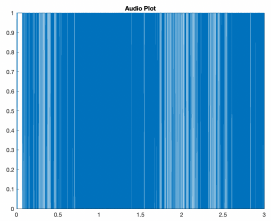
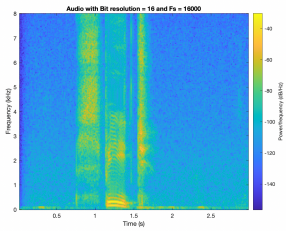
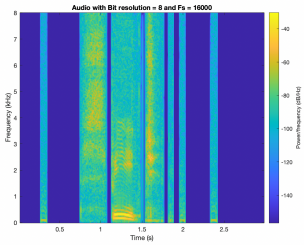
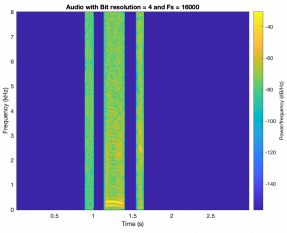
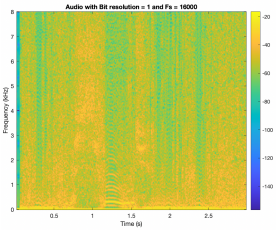
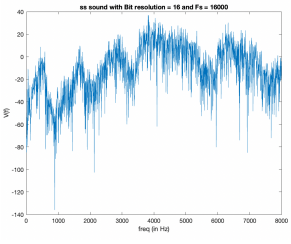
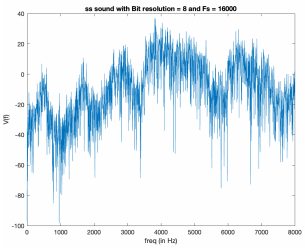
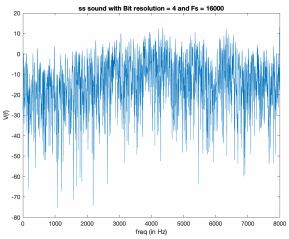
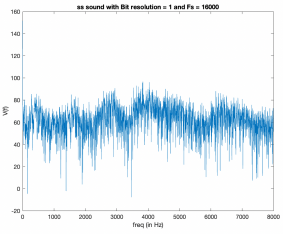
The Lab3plot() function is the same shown in part A.

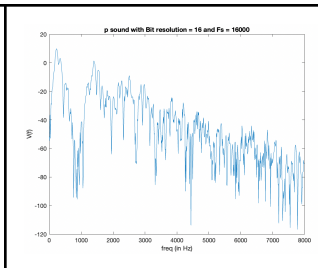
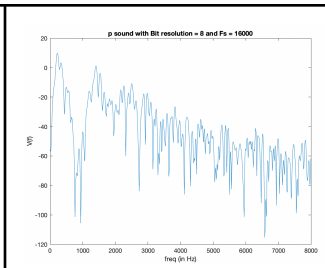
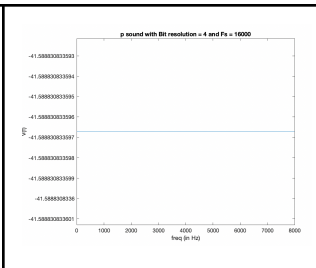
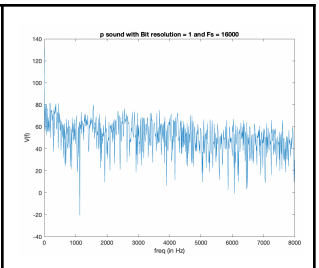
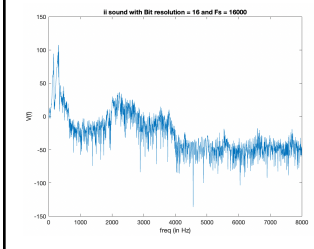
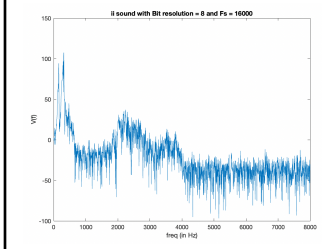
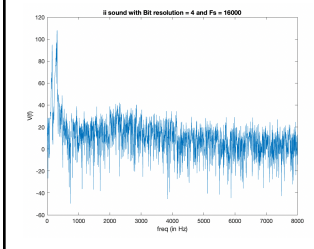
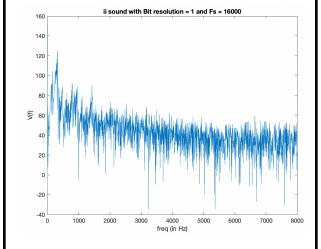
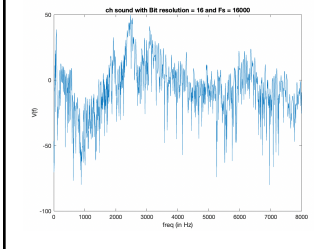
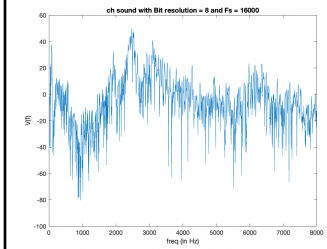
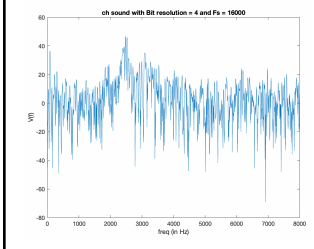
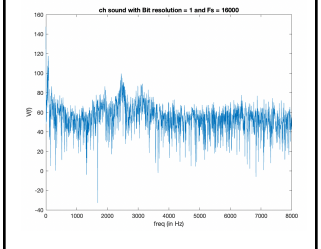
The additional function used in this experiment is the bitsPerSample() which converts the bit resolution of an audio to the desired value.

```
function [ahat] = bitsPerSample(a,B)

   %Special case when B =
   if(B == 1)

       %digitizing the given signal
       ahat = round((sign(a) + 1)/2);

   else

       %Reserving one bit as sign bit hence available bits for storing the value = B - 1;
       B = B - 1;

       %Number of quantization levels
       quantLevels = 2^B;

       % quantized array computed using rounding method.
       ahat = round(a.*quantLevels)./quantLevels;

   end

end
```

When bit resolution is set to 1 the function maps +ve values to 1 and -ve values to 0. The method used to do that is first using the sign() function of matlab to get a sign of the element of the array, then adding 1 and dividing by 2 to map -1 to 0 and 1 to 1.

For the case when bit resolution is set to a value greater than 1 we first compute the number of possible quantization levels, and then use the rounding technique.
In the rounding technique we first multiply the element of the array with the number of quantization levels then round it up to the nearest integer, then divide it again by the number of quantization levels.

	16 Bits/sample (Original)	8 Bits/sample	4 Bits/sample	1 Bits/sample
Audio plot				
Spectrogram				
ss Spectrum				

p Spectrum				
ii Spectrum				
ch Spectrum				

Observations:

1. As we decrease the bit resolution we can see that the audio plot becomes more and more discontinuous (blurry)
2. There is hardly any difference in the audio quality for the 16 bit resolution and the 8 bit resolution, and they can be perceived as almost the same.
3. In the case of 4 bits/sample there is significant noise but still the sounds ‘ii’ and ‘ch’ are properly perceivable, on the other hand the ‘ss’ sound is not very clear.
4. In the case of 1 bit/sample we observe a very noisy plot and no audio can be seen in it visually, the audio is very noisy but surprisingly still perceivable. The ‘ii’ sound can be heard in this resolution, but is not too clear, ‘ch’ sound is also present but is very short.
5. The spectrum of all the different sounds is almost the same in the case of 1 bit/Sample.

6. ‘p’ sound has a zero spectrum for 4 bits/sample but non zero in case of 1 bit/sample.

Inference and Conclusion:

1. The reason that the audio for both 16 and 8 bits/sample is the same is because all the amplitudes of the audio are below 1, and having 256 quantization levels in from - 1 to 1 is enough.
2. The spectrum ‘ii’ sound has its peak for all the bit resolution because periodicity is maintained even after quantizing the ‘ii’ sound with very small resolution.
3. In the case of 1 bit resolution the noise gets amplified, and hence all the sounds have similar spectrum for this resolution.
4. ‘p’ sound is absent in 4 bit resolution because all its elements are quantized to 0, whereas in the case of 1 bit resolution, even a slight noise having amplitude above 0 is quantized as 1 hence it has some spectrum for 1 bit resolution and not for 4 bit resolution.

After listening to the 1 bit resolution we can conclude that the information of a speech signal lies in the sequence and not essentially on the amplitudes. The 16 bit resolution is optimal and is generally used in audio processing, but still 8 bit resolution is a good alternative to reduce bit rate and still get good quality.