Name, Lastname, Stu # :

1	2	3	4	Toplam
1	1	1	3, 4	PÖÇ

KTU Faculty of Engineering, Dept. of Computer Engineering
COM/BİL 2000 Final Exam 06.06.2022

---

1. How does parallel priority interrupt hardware resolve priority and generate a code corresponding to the highest priority of all interrupt requesting devices. Explain briefly in one sentence. ⇔ Paralel öncelikli kesme donanımı, önceliği nasıl çözer ve tüm kesme talep eden cihazların en yüksek öncelikli olanına karşı düşen bir kodu nasıl üretir. Tek cümle ile kısaca açıklayınız.

Paralel olarak gelen birçok kesmenin önceliğini belirleyen öncelik kodlayıcı (priority encoder), tüm girişlerin oluşturduğu kombinasyonlarda en yüksek öncelikli olanı seçerek karşı düşen kodu üretecek şekilde kombinasyonel devre olarak tasarlandığı için gelen kesmelerden en yüksek öncelikli olanını seçerek kodunu çıkışta üretir.

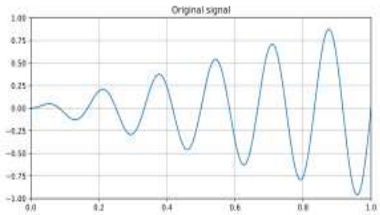
2. Computers have high quality, wide dynamic range (24-bit A/D) stereo audio interfaces (sound cards) which can sample input audio signal from microphones and provide output to speakers at studio quality (96.000 kHz) at the same time. Which I/O mode should be used to minimize audio I/O operation impact on CPU which at the same time has to encode high resolution video from camera with a high efficiency video codec to improve full-duplex online meeting performance? Explain the reason why in one sentence. ⇔ Bilgisayarlar, mikrofonlardan giriş ses işaretini örnekleyebilen ve aynı zamanda stüdyo kalitesinde (96.000 kHz) hoparlörlere çıktı sağlayabilen yüksek kaliteli, geniş dinamik aralığa (24-bit A/D) stereo ses arayüzüne (ses kartı) sahiptir. Her iki yönde görüşme gerçekleştirilen çevrimiçi toplantı performansını iyileştirmek için yüksek verimli video Codec ile kameradan yüksek çözünürlüklü videoyu kodlaması gereken CPU üzerindeki ses G/Ç işlem etkisini en aza indirmek için hangi G/Ç modu kullanılmalıdır? Nedenini bir cümlede açıklayın.

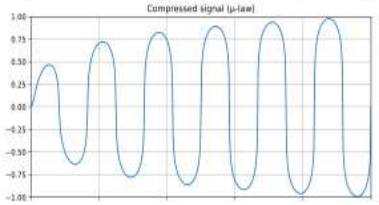
Doğrudan Bellek Erişimi, DMA kullanılmalıdır. Nedeni, 2 (full-duplex) x 2 (stereo) x 96.000 (örnek hızı) x 3 (24-bit / 8) = 1.152.000 Byte/s veriyi programlı veya kesme ile başlatılan programlı G/Ç yapması, CPU'nun saniyede milyonlarca G/Ç emrini yürütme veya kesme servisi vermesini gerektireceği için CPU'yu kullanmayan DMA ile yapılması gerekir.

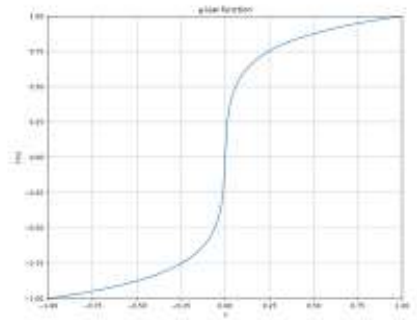
3. Explain briefly how the CPU and the IOP communicate to perform I/O? What would you suggest to improve the communication among many cores of modern CPUs and IOPs with many high speed I/O channels to peripherals. ⇔ CPU ve IOP'nin I/O gerçekleştirmek için nasıl iletişim kurduğunu kısaca açıklayın. Modern CPU'ların birçok çekirdeği ve çevre birimlerine birçok yüksek hızlı I/O kanalı olan IOP'ler arasındaki iletişimi geliştirmek için ne önerirdiniz?

CPU ve IOP arasındaki iletişim her iki birimin ortak eriştiği bellek üzerinden birbirlerine mesaj göndermeleri şeklinde gerçekleştirilmektedir. Paylaşılan belleğe yapılan bu sıralı erişim, özellikle CPU ve/veya IOP sayısı artarsa paylaşılan belleğe erişimi daha da geciktireceği için, grup olarak CPU'lar ve IOP'lerin paylaştığı yerel ortak bellekler ve gruplar arası mesaj iletimi için CPU+IOP grupların paylaştığı global bellek ile çözümlenebilmektedir.

4. From Wikipedia, the Free Encyclopedia; In telecommunication and signal processing, companding (occasionally called compansion) is a method of mitigating the detrimental effects of a channel with limited dynamic range. The name is a portmanteau of the words compressing and expanding, which are the functions of a compander at the transmitting and receiving end respectively. The use of companding allows signals with a large dynamic range to be transmitted over facilities that have a smaller dynamic range capability. Companding is employed in telephony and other audio applications such as professional wireless microphones and analog recording. ⇔ Vikipedi, Özgür Ansiklopedi'den, İletişim ve işaret işlemede, companding (bazen sıkıştırıcı-genişletici olarak ta adlandırılır), sınırlı dinamik aralığa sahip bir kanalın zararlı etkilerini azaltma yöntemidir. İsim, sırasıyla gönderici tarafın işlevi olan sıkıştırma (COMPression) ve alıcı tarafın işlevi olan genişletme (exPANDing, exPANSion) kelimelerinin birleştirilmesi ile oluşturulmuştur. Companding kullanımı, büyük bir dinamik aralığa sahip işaretlerin daha küçük bir dinamik aralık özelliğine sahip dizgeler (işaret akış yolları) üzerinden iletilmesini sağlar. Companding, telefon ve profesyonel kablosuz mikrofonlar ve analog kayıt gibi diğer ses uygulamalarında kullanılır.



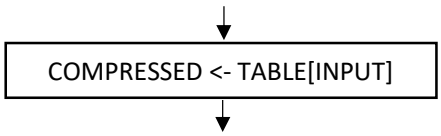


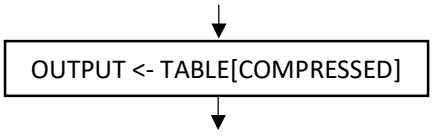


a. Draw the flowchart of the compressor algorithm if your student identification number is odd and expander algorithm if even. Assume an integer corresponding to sound sample is stored at address INPUT, compressed output samples needs to be stored at COMPRESSED (for compressor) and output samples needs to be stored at OUTPUT (for expander). ⇔ Ögrenci kimlik numaranız tek ise sıkıştırıcı algoritmanın, çift ise genişletici algoritmanın akış şemasını çiziniz. Ses örneğine karsılık gelen bir tamsayının INPUT adresinde saklandığını, sıkıştırılmış çıkış örneklerinin COMPRESSED adresinde (kompresör için) ve çıkış değerinin OUTPUT adresinde (genişletici için) saklanması gerektiğini varsayın.

Hint: Most of the non-linear functions (shown here for μ-law) used in companders are compute intensive and not easy to compute numerically in real-time. Instead, a look-up table is used to associate an input value to an output value. For example, if the function is multiplication by a constant (lets say 2), the look-up table would be 1⇔2, 2⇔4, 3⇔6, 4⇔8, …. ⇔ **İpucu:** Sıkıstırıcılarda kullanılan doğrusal olmayan işlevlerin çoğu (burada μ-yasası için gösterilmiştir) hesaplama zorluğu içeren bir işlemdir ve gerçek zamanlı olarak sayısal hesaplanması kolay değildir. Bunun yerine, bir giriş değerini bir çıkış değeriyle ilişkilendirmek için bir bakma-tablosu (karşılık tablosu) kullanılır. Örneğin, fonksiyon bir sabitle çarpma ise (2 diyelim), arama tablosu 1⇔2, 2⇔4, 3⇔6, 4⇔8, …. olur,

Doğrusal olmayan işlevin giriş ve sonuç değerlerinin yer aldığı Nx2 boyutlu bir TABLE'da işlevin argüman değerine eşit olan TABLE[i][1] indisi bulunduktan sonra, TABLE[i][2] dan giriş değerine karşı düşen non-lineer çıkış değeri bulunur. Tabloda **arama** yapmayı gerektirdiği için yavaş çalışacak bu yöntem yerine, **sıralı** giriş değerlerine karşı düşen non-lineer işlev çıkış değerini tutan Nx1 boyutlu TABLE'ı bellekte arama yapmadan doğrudan adresleyerek çıkış değerini elde etmek çok daha verimlidir. Bu durumda arama yapmadan çıkış, M[TABLE+giriş] adresindeki bellekten doğrudan okunabilir. Sıkıştırma ve açma işlemleri için sırası ile akış şemaları:





b. Code your algorithm in any high level language. ⇔ Algoritmasını herhangi bir yüksek seviyeli dil ile kodlayınız.

İndisle adreslenebilen tamsayı dizi veri yapısı kullanarak sırası ile:

```
COMPRESSED=TABLE[INPUT];
```

```
OUTPUT=TABLE[COMPRESSED];
```

c. Code your algorithm in **Assembly** language (not **micro-operations**). ⇔ Algoritmanızı **Assembly** dili ile kodlayınız (**mikro operasyonlar** değil).

TABLE adresinde tutulan tablo içinde indisli adresleme ile R1 kaydedicisindeki değere karşı düşen değerin tablo'dan bakıldığı ve R2'ye atandığı assembly dili kod parçaları sırası ile:

```
MOV R1, INPUT          ; Read linear INPUT
MOV R2, TABLE (R1)     ; Get corresponding non-linear value
MOV COMPRESSED, R2     ; Write non-linear COMPRESSED
```

```
MOV R1, COMPRESSED     ; Read non-linear COMPRESSED
MOV R2, TABLE (R1)     ; Get corresponding linear value
MOV OUTPUT, R2         ; Write linear OUTPUT
```