Name, Lastname, Stu # :

1	2	3	4	Toplam
2	3	2, 3	3, 4	PÖÇ

KTU Faculty of Engineering, Dept. of Computer Engineering
COM/BİL 2000 Mid-Term Exam 09.04.2018

---

1. Which addressing mode should be used to initialize all elements of an N x M matrix to zero. Explain your reason in one sentence **//** N x M boyutlarındaki bir matrisin tüm elemanlarının sıfıra ilklendirilmesi için hangi adresleme modunu kullanması gerekir. Nedeninizi bir cümle ile açıklayınız.

   Erişilmek istenen adresi bir kaydedicide (address-register) saklayan adresleme modu, yani **kaydedici üzerinden dolaylı** (register indirect) adresleme modu kullanılmalıdır. Bu modun seçilme nedeni, matris tipi veri yapısının elemanlarının bellekte ardışıl adreslerde saklanması ve emir kodlarında erişilmek istenen tüm matris elemanlarının saklandığı ardışıl bellek adresini belirtmeye gerek kalmayacak şekilde adı ile belirtilen bir adres kaydedicisinde saklanan başlangıç adresinin artırılarak tüm matris elemanlarına erişilebilmesidir.

2. Write two separate Assembly language code snippets using both Independent (Isolated) and Memory Mapped I/O instructions to input (load) a printer's status from *PRN_STATUS* which indicates Out of Paper if not zero and output (store) to *PRN_DATA* a character code stored in accumulator. **//** Bir yazıcının sıfır olmadığında kağıt bittiğini gösteren durumunu (status) *PRN_STATUS* adresinden giriş (yükleme) yaptıktan sonra durum sıfırdan farklı değilse akümlatördeki bir karakter kodunu *PRN_DATA* adresine çıkış (saklama) yapan hem Bağımsız (İzole) hem de Belleğe Haritalanmış G/Ç emirleri ile gerçekleştiren iki Assembly dili kod parçasını yazınız.

Bağımsız (Independent or Isolated)

```
            in    B, PRN_STATUS
            and   B, B          // bayrakları güncelle
            jnz   kağıt_bitti   (veya bnz)
            out   PRN_DATA, A
kağıt_bitti : izleyen emir
```

Belleğe Haritalı (Memory Mapped)

```
            mov   B, PRN_STATUS
            and   B, B
            jnz   kağıt_bitti   (veya bnz)
            mov   PRN_DATA, A
kağıt_bitti : izleyen emir
```

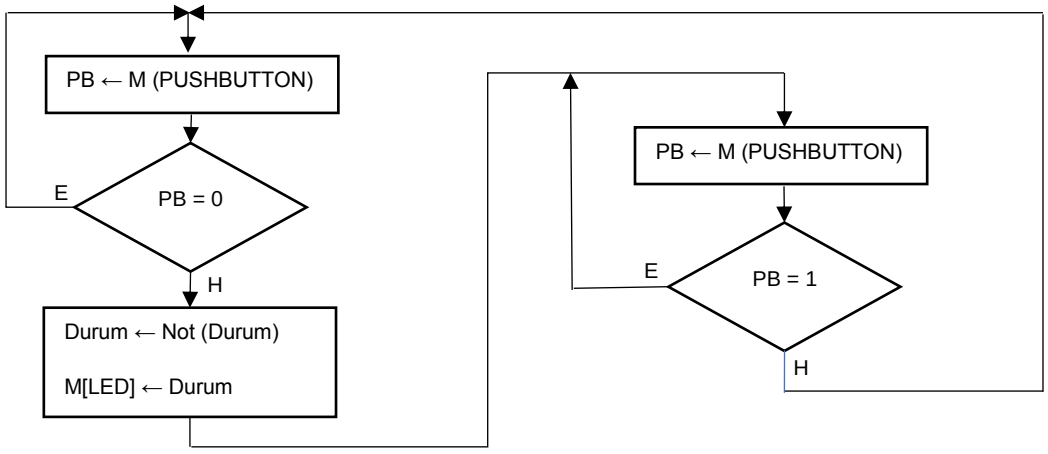
3. Even though subroutine call (**CALL** *ADRS*) and return (**RET**) instructions are defined in Assembly language, they are used in other languages as well. List the sequence of micro-operations required for both **CALL** *ADRS* and **RET** instructions using a stack addressed by Stack Pointer (SP) addressing the Top of Stack defined in memory. **//** Altprogram çağırma (**CALL** *ADRS*) ve geri dönüş (**RET**) emirleri Assembly dilinde tanımlanmış olsalar da diğer dillerde de kullanılmaktadır. Bellekte tanımlı yığını ve yığının en üstteki elemanını (Top of Stack) adresleyen Yığın Göstericiyi (Stack Pointer, SP) kullanarak **CALL** *ADRS* ve **RET** emirlerinin çalıştırılması için gereken mikro-operasyonlar dizisini listeleyiniz.

```
DR ← M[PC]      // bellekten emir kodunu DR'a al
IR ← DR         // emrin kodunu emir kaydediciye al ve çözümle
PC ← PC + 1     // PC'yi artır
DR ← M[PC]      // bellekten Alt Program adresini DR'a al
PC ← PC + 1     // Alt Program dönüşünde gidilecek bir sonraki emri gösterecek şekilde PC'yi artır
SP ← SP + 1     // Yığın Göstericisi'ni Dönüş Adresini saklayacağı boş bir yeri gösterecek şekilde artır
M[SP] ← PC      // Dönüş adresini Yığına it (push)
PC ← DR         // Alt Programa adresini PC'ye aktararak Alt Programa git (call)

DR ← M[PC]      // bellekten emir kodunu DR'a al
IR ← DR         // emrin kodunu emir kaydediciye al ve çözümle
PC ← PC + 1     // PC'yi artır
DR ← M[SP]      // Yığın Göstericisi'nin gösterdiği adresten Dönüş Adresini DR'a al
SP ← SP – 1     // Yığın Göstericisi'ni yığının en üstteki elemanını gösterecek şekilde azalt
PC ← DR         // ana programda call emrini izleyen adrese geri dön (ret)
```

4. From Wikipedia, the free encyclopedia: In some contexts, particularly computing, a toggle switch, or the action of toggling, is understood in the different sense of a mechanical or software switch that alternates between two states each time it is activated, regardless of mechanical construction. For example, the caps lock key on a computer causes all letters to be generated in capitals after it is pressed once; pressing it again reverts to lower-case letters. Design a microprocessor based toggle switch activated by a push button like a ring button to power on and off of a computer controlled system. You may assume that push button data which you can input (load) from address PUSHBUTTON is already De-Bounced by a S-R latch. Your algorithm should turn a LED on by output (store) a 1 at address POWER indicating “power on” and output a 0 indicating “power off” which turns off the LED. State all other assumptions you have to make in order to design the toggle switch. **//** Vikipedi, özgür ansiklopedi’den: Bazı bağlamlarda, özellikle bilişimde, bir devirmeli (toggle) anahtar veya devirme (toggle etme), mekanik yapısından bağımsız olarak, her uyarıldığında iki durumun birinden diğerine geçen mekanik veya yazılımla gerçekleştirilen farklı bir anlamda anlaşılır. Örneğin, bilgisayarlardaki büyük harfe kilitleme (caps lock) tuşu, bir kez basıldığında yazılan bütün harflerin büyük harf olarak üretilmesini ve bir kez daha basıldığında yeniden küçük harflere döndürülmesini sağlar. Bilgisayar denetimli bir sistemin açık (power on) ve kapalı (power off) duruma getirilmesi için zil düğmesi gibi basmalı anahtar ile çalıştırılan mikroişlemci tabanlı devirmeli (toggle) anahtar sistemi tasarlayınız. PUSHBUTTON adresinden giriş (yükleme) yapabileceğiniz düğme işaretinin bir S-R kilit ile anahtar sıçramasının giderilmiş olduğunu varsayabilirsiniz. Algoritmanız “sistem açık” olduğunu gösteren LED’i süren POWER adresine 1 değeri çıkarmalı (saklamalı) ve “sistem kapalı” olduğunu gösteren 0 değerini çıkararak (saklayarak) LED’i söndürmelidir. Tasarlarken yapmak zorunda kaldığınız diğer tüm varsayımları belirtmeniz gerekmektedir.

a. Draw the flowchart of the toggle switch algorithm. **//** Devirmeli (toggle) anahtar algoritmasının akış çizgesini çiziniz.



b. Code toggle switch algorithm in any high level language. **//** Devirmeli (toggle) anahtar algoritmasını herhangi bir yüksek seviyeli dil ile kodlayınız.

```
while (true) {
   do {
     PB = inport(PUSHBUTTON);
   } while (PB == 0);      // Tuşa basılmasını bekle
   durum = ! durum;        // durumu tümle (toggle)
   outport(LED,durum); // LED’e çıkar
   do {
     PB = inport(PUSHBUTTON);
   } while (PB == 0);      // Tuşun bırakılmasını bekle
}
```

c. Code toggle switch algorithm in Assembly language. // Devirmeli (toggle) anahtar algoritmasını Assembly dili ile kodlayınız.

```
döngü :   in A, PUSHBUTTON
          and A, A
          jz döngü          // Tuşa basılmasını bekle
          not DURUM         // durumu tümle (toggle)
          mov A, DURUM
          out LED, A        // LED’e çıkar
bekle :   in A, PUSHBUTTON
          and A, A
          jnz bekle         // Tuşun bırakılmasını bekle
          j döngü
```