2020/1 Assembly Dersi Ödev 1		
İlan Tarihi	*Son Teslim Tarihi/Saati*	*Teslim Şekli*
28/12/2020	*31/12/2020 – 23:59*	*Cevaplarınızı word dosyası olarak online.yildiz.edu.tr'den yükleyiniz*

(Soru 1-20: 2p, Soru 21: 10p, Soru 22: 50p)

Ad: MHD Besher AL KURDI NO: 19011923

Soru 1: SP, BX, DI, BP, SI için varsayılan kesim yazmaçları hangileridir?

SP: SS **BX:** DS **DI:** DS **BP:** SS **SI:** DS

Soru 2: 8B07H makine kodu üreten assembly komutunu bulunuz.

```
MOV AX, [BX]
```

Soru 3: 8B9E004CH makine kodu üreten assembly komutunu bulunuz.

```
MOV BX, [BP]+4C00h
```

Soru 4: MOV SI, [BX+2] komutunun makine kodu karşılığını bulunuz.

```
8B7702
```

Soru 5: MOV CS, AL komutundaki hatayı bulunuz.

CS doğrudan değiştirilemez. Değiştirilebilir olsa dahi WORD boyutlu olan CS'e BYTE boyutunda bir değişken yazdırılamaz.

Soru 6: MOV DI, NUMBER ve LEA DI, NUMBER komutları arasındaki farkı açıklayınız.

MOV DI, NUMBER Doğrudan sayı değerini aktarırken LEA DI, NUMBER komutu 16-bitlik bir offset sonrasındaki değeri (yani değişkenin adresini) aktarır.

Soru 7: ES'nin gösterdiği kesim alanı içinde BX ile işaret edilen hafıza bölgesinin içeriğini AH yazmacına taşıyan komutu yazınız.

```
MOV AH, ES:[BX]
```

Soru 8: BX'i AX'e toplayan ADD komutunu yazınız.

```
MOV AX, BX
```

Soru 9: 12H değerini AL'ye toplayan ADD komutunu yazınız.

```
ADD AL, 12H
```

Soru 10: DI'yı BP'ye toplayan ADD komutunu yazınız.

```
ADD BP, DI
```

Soru 11: 22H değerini CX'e toplayan ADD komutunu yazınız.

```
ADD CX, 22H
```

Soru 12: SI ile gösterilen adresin içeriğini AL'ye toplayan ADD komutunu yazınız.

```
ADD AL, [SI]
```

Soru 13: CX'ı DENEME ile gösterilen adres içeriğine toplayan ADD komutunu yazınız.

```
ADD WORD PTR [DENEME], CX
```

DENEME 'nin offset değerini gösterdiği kabul edilmiştir. Adres olması için yapılan işlemi tek komutta ifade edemedim.

Soru 14: ADD CX, AH komutundaki hata nedir?

BYTE boyutundaki yazmaç değerinin içeriğe WORD boyutundaki CX'e yazılamaz.

Soru 15: AL, AH, BL ve CL'yi toplayıp sonucu DX'te saklayan komutları yazınız.

Saklanan sayılar işaretli ise:

```
XOR DX, DX
MOV BH, AH
CBW
ADD DX, AX
MOV AL, BH
CBW
ADD DX, AX
MOV AL, BL
CBW
ADD DX, AX
MOV AL, CL
CBW
ADD DX, AX
```

Saklanan sayılar işaretsiz ise:

```
XOR DX, DX
ADD DL, AL
ADD DL, AH
ADC DH, 0
ADD DL, BL
```

```
ADC DH, 0
ADD DL, CL
ADC DH, 0
```

Soru 16: INC [BX] komutundaki hatayı bulunuz.

BYTE PTR ile cast yapılmadığından çalışmaz

Soru 17: SUB ve CMP komutları arasındaki farkı açıklayınız.

SUB komutu verilen ilk operand'dan ikinciyi çıkarıp sonuca ilk operanda yazdırırken CMP komutu operandlar üzerinde değişiklik yapmadan sadece flag değerlerini değiştirir.

Soru 18: IMUL ve MUL komutları arasındaki fark nedir?

IMUL İşaretli sayılar için çarpma yaparken MUL verilen sayının işaretsiz olduğunu kabul ederek çarpma işlemini gerçekleştirir.

Soru 19: DIV işlemi sonucunda kalan nerede saklanır?

Verilen operand WORD boyutunda ise kalan DX yazmacına yazılırken operand BYTE boyutunda ise kalan AH yazmacına yazılır

Soru 20: AX=1001H ve DX=20FFH iken ADD AX, DX komutu çalıştırıldıktan sonra C, A, S, Z ve O bayrakları nasıl değişir?

CF clear olur AF set olur SF clear olur ZF clear olur OF clear olur

Soru 21: DL'de tutulan sayının küpünü bulan assembly komutları yazınız (10p).

Verilen sayının işaretli sayı olduğu varsayılırsa:

```
MOV AL, DL
CBW
MOV BX, AX
IMUL BX
IMUL BX ;sonuc DX:AX'te sakli
```

Soru 22: Random sayı üretmek için kullanılan yöntemlerden biri de "linear congruential generation" yöntemidir. Bu yöntemde bir sonraki random sayı (x_{t+1}), önceki random sayıya (x_t) göre

$$x_{t+1} = (a \cdot x_t + c) \pmod{m}$$

formülüne göre hesaplanır. x_0 değeri ise seed olarak isimlendirilir.

$m=8191$, $a=884$ ve $c=1$ değerleri için verilen yöntemle 0-255 arasında bir random sayı üretmeniz istenmektedir. Kod segmentte 23 ofsetinden itibaren okuyacağınız sayıyı

seed olarak kabul ederek bir random sayı üreten kodu tam bir EXE programı olarak assembly dili ile yazın. m, a ve c değerlerini data segmentte uygun şekilde tanımlayın, üretilen random sayıyı yine data segmentte saklayınız (50p).

İşlemi gerçekleştirmek için `rndn` isimli özyinelemeli yordam tanımlanmıştır. Fonksiyona parametre aktarımı CX,DX,BX yazmaçları üzerinden gerçekleşmekte ve DX üzerinden değer döndürülmekte olup programın tamamı bu işlev için yazıldığından AX yazmacındaki değer **korunmamaktadır**.

```
datasg  SEGMENT PARA 'data'
    m DW 8191
    a DW 884
    c DW 1
    rand DW ?
datasg  ENDS
stacksg SEGMENT PARA STACK 'yigin'
     DW 40 DUP(?)
stacksg ENDS
codesg  SEGMENT PARA 'code'
     ASSUME CS:codesg, SS:stacksg, DS:datasg
rndn    PROC NEAR
    MOV AX, DX
    MUL CX
    ADD AX, c
    DIV BX
    MOV AX, 256
    CMP AX, DX
    JA  finish
    CALL rndn
finish:
    RET
rndn    ENDP
MAIN  PROC FAR
     PUSH DS
     XOR AX, AX
     PUSH AX
     MOV AX, datasg
     MOV DS, AX ;EXE tipi son
     MOV CX, a
     MOV DX, CS:[23]
     MOV BX, m
     CALL rndn
     MOV rand, DX
    RETF
    MAIN ENDP
     codesg ENDS
```

```
      END MAIN
```