BIL 362 Mikroişlemciler

M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Konular

Program Kontrol Komutları

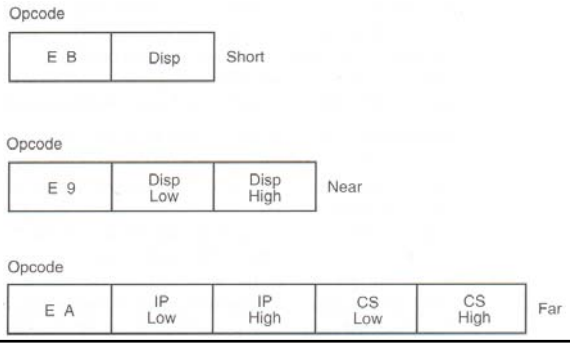
- Şartsız Atlama
- Şartlı Atlama
- Döngüler (Loop)
- Assembly Programı Akış Denetimi
- While Döngüleri
- Repeat-Until Döngüleri
- Prosedürler
- Kesmelere (Interrupts) Giriş
- Diğer Komutlar

Şartsız Atlama

- Kontrol komutları programın bir kısmına atlamayı sağlar.
- Kontrol komutları bayrak bitlerinin durumuna göre şartlı atlamayı gerçekleştir.
- Short jump, near jump ve far jump olmak üzere 3 tür şartsız atlama komutu kullanılabilir.

Şartsız Atlama

- Short jump komutu 2 byte uzunluğundadır ve bulunulan yere göre +127 ve -128 adres aralığına atlama yapar.
- Near jump 3 byte uzunluğundadır ve bulunulan segment içerisinde ±32K aralığına atlama yapar.
- Far jump 5 byte uzunluğundadır ve hafızada herhangi bir alana atlamayı gerçekleştirir.
- Short ve near jump intrasegment, far jump intersegment atlama yapar.



Şartsız Atlama

Short jump

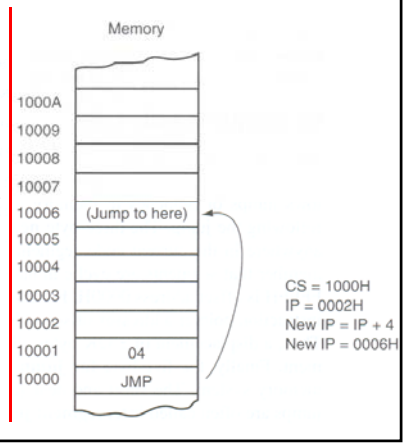
- Bulunulan segment içerisinde göreceli atlama yapar.
- Opcode içinde atlanacak adres yerine bulunulan yere uzaklık verilir.
- **JMP SHORT** NEXT şeklinde kullanılır. NEXT atlanacak yerdir.

```
0000 33 DB                XOR   BX,BX
0002 B8 0001      START:  MOV   AX,1
0005 03 C3                ADD   AX,BX
0007 EB 17                JMP   SHORT NEXT

                  <skipped memory locations>

0020 8B D8        NEXT:   MOV   BX,AX
0022 EB DE                JMP   START
```

0009H + 0017H = 0020H



Şartsız Atlama

Near jump

- Near jump, short jump gibi aynı segment içerisinde atlama yapar ancak short jump'a göre daha uzun aralığa atlar.
- Bulunulan segment içerisinde ±32K aralığına atlama yapar. 80386 ve üstü işlemcilerde protected mode'da ±2G aralığına atlama yapar.

```
0000 33DB                 XOR   BX,BX
0002 B8 0001      START:  MOV   AX,1
0005 03 C3                ADD   AX,BX
0007 E9 0200 R            JMP   NEXT

                  <skipped memory locations>

0200 8B D8        NEXT:   MOV   BX,AX
0202 E9 0002 R            JMP   START
```

Şartsız Atlama

Far jump

- Far jump yeni bir segment ve offset adres alır ve doğrudan istenen adrese atlar.
- Far jump FAR PTR komutuyla veya far label etiketiyle belirtilir.

```
                                EXTRN   UP:FAR

0000 33 DB                      XOR   BX,BX
0002 B8 0001            START:  ADD   AX,1
0005 E9 0200 R                  JMP   NEXT

                        <skipped memory locations>

0200 8B D8              NEXT:   MOV   BX,AX
0202 EA 0002 ---- R             JMP   FAR PTR START

0207 EA 0000 ---- R             JMP   UP
```

Şartlı Atlama

- Şartlı atlama komutları 8086-80286 işlemcilerde short jump yapar.

Assembly Language	*Tested Condition*	*Operation*
JA	Z = 0 and C = 0	Jump if above
JAE	C = 0	Jump if above or equal
JB	C = 1	Jump if below
JBE	Z = 1 or C = 1	Jump if below or equal
JC	C = 1	Jump if carry
JE or JZ	Z = 1	Jump if equal or jump if zero
JG	Z = 0 and S = 0	Jump if greater than
JGE	S = 0	Jump if greater than or equal
JL	S != O	Jump if less than
JLE	Z = 1 or S != O	Jump if less than or equal
JNC	C = 0	Jump if no carry
JNE or JNZ	Z = 0	Jump if not equal or jump if not zero
JNO	O = 0	Jump if no overflow
JNS	S = 0	Jump if no sign (positive)
JNP or JPO	P = 0	Jump if no parity or jump if parity odd
JO	O = 1	Jump if overflow
JP or JPE	P = 1	Jump if parity or jump if parity even
JS	S = 1	Jump if sign (negative)
JCXZ	CX = 0	Jump if CX is zero
JECXZ	ECX = 0	Jump if ECX equals zero

Şartlı Atlama

- +127 ile -128 arasında atlama yapar. MASM v.6X assembler uzaklığı otomatik ayarlar.
- Şartlı atlama komutları S, Z, C, P ve O bayrak bitlerini kontrol eder.
- Test edilen şart doğru (true, 1) ise etiketle belirtilen yere atlanır, yanlış (false, 0) ise sonraki adım çalıştırılır.
- Karşılaştırma işlemi signed (işaretli) ve unsigned (işaretsiz) sayılarda farklı sonuçlar oluşturur.
- İşaretsiz sayılarda FFH (255) sayısı 00H (0) sayısından büyüktür.
- İşaretli sayılarda FFH (-1) sayısı 00H (0) sayısından küçüktür.

Şartlı Atlama

- İşaretli sayılarda karşılaştırma için JG, JL, JGE, JLE, JE ve JNE komutları kullanılır.
- İşaretsiz sayılarda karşılaştırma için JA, JB, JAE, JBE, JE ve JNE komutları kullanılır.
- Diğer şartlı atlama komutları bayrak bitlerini test eder.
- JCXZ ve JECXZ komutları CX ve ECX register'larını test eder ve 0 ise istenen yere atlama yapar.
- Aşağıdaki örnekte 100 byte alan içinde 0AH değerini arar.

```
                ;Instructions that search a table of 100H bytes for 0AH
                ;The offset address of TABLE is assumed to be in SI
                ;
0017 B9 0064          MOV  CX,100          ;load counter
001A B0 0A            MOV  AL,0AH          ;load AL with 0AH
001C FC               CLD                  ;auto-increment
001D F2/AE            REPNE SCASB          ;search for 0AH
001F F9               STC                  ;set carry if found
0020 E3 01            JCXZ NOT_FOUND       ;if not found
0022                NOT_FOUND
```

Döngüler (Loop)

- LOOP komutu CX azaltma ve JNZ şartlı atlama komutlarının birleşimidir.
- Her adımda CX register'ı 1 azaltılır ve CX değeri test edilir.
- 8086-80286 işlemcilerde CX azaltılır ve CX<>0 olduğu sürece belirtilen etikete atlanır. CX=0 ise sonraki adıma geçilir.
- 80386 ve üstü işlemcilerde CX veya ECX kullanılır. LOOPW komutu CX, LOOPD komutu ECX register'ını kullanır.

Şartlı LOOP

- LOOPE (Loop while equal) ve LOOPNE (Loop while not equal) komutları CX<>0 ve verilen şart geçerli olduğu sürece döngüyü tekrarlar.
- 80386 ve üstü işlemcilerde CX veya ECX kullanılabilir. LOOPEW/LOOPNEW ve LOOPED/LOOPNED komutları CX ve ECX register'larını kullanır.

Döngüler (Loop)

- Örnekte BLOCK1 de word boyutundaki data BLOCK2 ye eklenir.

```
                        ;A program that sums the contents of BLOCK1 and BLOCK2
                        ;and stores the results on top of the data in BLOCK2.
                        ;
                        .MODEL SMALL                    ;select SMALL model
0000                    .DATA                           ;start data segment
0000 0064 [             BLOCK1 DW  100 DUP(?)           ;100 words for BLOCK1
          0000
             ]
00C8 0064 [             BLOCK2 DW  100 DUP(?)           ;100 words for BLOCK2
          0000
             ]
0000                    .CODE                           ;start code segment
                        .STARTUP                        ;start program
0017 8C D8                      MOV AX,DS               ;overlap DS and ES
0019 8E C0                      MOV ES,AX
001B FC                         CLD                     ;select auto-increment
001C B9 0064                    MOV CX,100              ;load counter
001F BE 0000 R                  MOV SI,OFFSET BLOCK1    ;address BLOCK1
0022 BF 00C8 R                  MOV DI,OFFSET BLOCK2    ;address BLOCK2
0025 AD                 L1:     LODSW                   ;load AX with BLOCK1
0026 26:03 05                   ADD AX,ES:[DI]          ;add BLOCK2
0029 AB                         STOSW                   ;save answer
002A E2 F9                      LOOP L1                 ;repeat 100 times
                        .EXIT
                        END
```

Assembly Programı Akış Denetimi

- MASM 6.X assembler .IF, .ELSE, .ELSEIF, .ENDIF, .REPEAT, .UNTIL, .WHILE ve .ENDW gibi komutları sağlar.
- Yandaki örnekte .IF ve .ENDIF ile yapılan işlemin assembly komutlarıyla gerçekleştirilmesi görülmektedir.

```
.IF AL >= 'A'  &&  AL <= 'F'
        SUB   AL,7
.ENDIF
SUB   AL,30H
```

```
char temp;
_asm{
        mov   al,temp
        cmp   al,41h
        jb    Later
        cmp   al,46h
        ja    Later
        sub   al,7
Later:
        sub   al,30h
        mov   temp,al

}
```

Assembly Programı Akış Denetimi

- Aşağıda .IF komutuyla kullanılabilecek ilişkisel operatörler görülmektedir.

Operator	*Function*
==	Equal or the same as
!=	Not equal
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
&	Bit test
!	Logical inversion
&&	Logical AND
||	Logical OR
|	OR

Örnek

Assembler üretir

```
                        ;A DOS program that reads the keyboard and converts all
                        ;lowercase data to uppercase before displaying it.
                        ;
                        ;This program is terminated with a control-C
                        ;
                        .MODEL TINY                     ;select tiny model
                        .LISTALL                        ;list all statements
0000                    .CODE                           ;start code segment
                        .STARTUP                        ;start program
0100                  * @Startup
0100 B4 06              MAIN1: MOV  AH,6                ;read key without echo
0102 B2 FF                     MOV  DL,0FFH
0104 CD 21                     INT  21H
0106 74 F8                     JE   MAIN1               ;if no key
0108 3C 03                     CMP  AL,3                ;test for control-C
010A 74 10                     JE   MAIN2               ;if control-C

                               .IF  AL >= 'a' && AL <= 'z'

010C 3C 61                   *      cmp  al,'a'
010E 72 06                   *      jb   @C0001
0110 3C 7A                   *      cmp  al,'z'
0112 77 02                   *      ja   @C0001
0114 2C 20                          SUB  AL,20H

                               .ENDIF

0116                  * @C0001:
0116 8A D0                     MOV  DL,AL               ;echo character to display
0118 CD 21                     INT  21H
011A EB E4                     JMP  MAIN1               ;repeat
011C                    MAIN2:
                        .EXIT
011C B4 4C              *      MOV  AH,4CH
011E CD 21              *      INT  21H
                        END
```

While Döngüleri

- Yüksek seviyeli dillerde olduğu gibi MASM 6.X WHILE döngü yapısı içermektedir.
- .WHILE deyimi şart kısmını, .ENDW ise döngü bitişini gösterir.
- .WHILE deyimindeki şart kısmı doğru olduğu sürece döngü tekrarlanır. Her döngüde şart tekrar kontrol edilir.
- Şart doğru olmadığında .ENDW deyiminden sonraki adıma geçilir.
- Sonraki sayfadaki örnekte klavyeden 0DH (Enter) girilene kadar tüm girişler ekstra segment içerisinde BUF adlı diziye kaydedilir.
- While döngülerinde .BREAK ve .CONTINUE deyimleri kullanılır.
- .BREAK .IF AL == 0DH komutu AL==0 ise döngüden çıkış yapar.
- .CONTINUE .IF AL==15 komutu AL==15 ise döngü sonuna kadarki kısmı atlar ve sonraki döngüye devam eder.

```
                        ;A DOS program that reads a character string from the
                        ;keyboard and then displays it again.
                        ;
                        .MODEL SMALL                    ;select small model
0000                    .DATA                           ;start data segment
0000 0D 0A              MES     DB  13,10               ;return and line feed
0002 0100 [             BUF     DB 256 DUP(?)           ;character string buffer
          00
            ]
0000                    .CODE                           ;start code segment
                        .STARTUP                        ;start program
0017 8C D8                      MOV  AX,DX              ;overlap DS with ES
0019 8C C0                      MOV  ES,AX
001B FC                         CLD                     ;select auto-increment
001C BF 0002 R                  MOV  DI,OFFSET BUF      ;address buffer

                                .WHILE AL != 0DH        ;loop while not enter
001F EB 05              *       jmp  @C0001
0021                    * @C0002:
0021 B4 01                      MOV  AH,1               ;read key
0023 CD 21                      INT  21H
0025 AA                         STOSB                   ;store key code
                                .ENDW
0026                    * @C0001:
0026 3C 0D              *       cmp  al,0dh
0028 75 F7              *       jne  @C0002
002A C6 45 FF 24                MOV  BYTE PTR[DI-1]'&'
002E BA 0000 R                  MOV  DX,OFFSET MES
0031 B4 09                      MOV  AH,9
0033 CD 21                      INT  21H                ;display MES
                        .EXIT
                        END
```

Repeat-Until Döngüleri

- Yüksek seviyeli dillerde olduğu gibi MASM 6.X REPEAT-UNTIL döngü yapısı içermektedir.
- .UNTIL deyimi şart kısmını, .REPEAT ise döngü başlangıcını gösterir.
- Döngü içindeki deyimler şart kısmı doğru olduğu sürece tekrarlanır. Her döngüde şart tekrar kontrol edilir.
- Şart doğru olmadığında .UNTIL deyiminden sonraki adıma geçilir.
- Sonraki sayfadaki örnekte klavyeden 0DH (Enter) girilene kadar tüm girişler ekstra segment içerisinde BUF adlı diziye kaydedilir.
- Repeat döngülerinde .UNTILCXZ deyimi kullanılır ve her döngüde CX register'ını bir azaltarak kontrol eder.

```
                        ;A DOS program that reads a character string from the
                        ;keyboard and then displays it again.
                        ;
                        .MODEL SMALL                    ;select small model
0000                    .DATA                           ;start data segment
0000 0D 0A              MES     DB  13,10               ;return and line feed
0002 0100[              BUF     DB 256 DUP(?)           ;character string buffer
          00
            ]
0000                    .CODE                           ;start code segment
                        .STARTUP                        ;start program
0017 8C D8                      MOV  AX,DS              ;overlap DS with ES
0019 8C C0                      MOV  ES,AX
001B FC                         CLD                     ;select auto-increment
001C BF 0002 R                  MOV  DI,OFFSET BUF      ;address buffer

                        .REPEAT                         ;repeat until enter

001F                  * @C0001:
001F B4 01                      MOV  AH,1               ;read key
0021 CD 21                      INT  21H
0023 AA                         STOSB                   ;store key code

                        .UNTIL AL == 0DH

0025 3C 0D            *         cmp  al,0dh
0027 75 F7            *         jne  @C0001
0028 C6 45 FF 24                MOV  BYTE PTR[DI-1]'&'
002C BA 0000 R                  MOV  DX,OFFSET MES
002E B4 09                      MOV  AH,9
0031 CD 21                      INT  21H                ;display MES
                        .EXIT
                        END
```

Repeat-Until Döngüleri

- Aşağıdaki örnekte CX register'ı her döngüde bir azaltılarak kontrol edilir.
- CX <> 0 olduğu sürece döngü tekrarlanır.

```
012C B9 0064                    MOV  CX,100             ;set count
012F BF 00C8 R                  MOV  DI,OFFSET THREE    ;address arrays
0132 BE 0000 R                  MOV  SI,OFFSET ONE
0135 BB 0064 R                  MOV  BX,OFFSET TWO

                        .REPEAT

0138                  * @C0001:
0138 AC                         LODSB
0139 02 07                      ADD  AL,[BX]
013B AA                         STOSB
013C 43                         INC  BX

                        .UNTILCXZ

013D E2 F9            *         LOOP  @C0001
```

Prosedürler

- Prosedürler programların yeniden kullanılabilir olan ve genellikle sadece bir fonksiyonu gerçekleştiren kısımlarıdır.
- CALL komutu bir prosedüre geçiş için, RET komutu ise geri dönüş için kullanılır.
- Bir prosedür çağrıldığında stack geri dönüş adresini saklar. CALL komutu stack'a bir sonraki adresi (return) push eder.
- RET komutu stack'tan bir adresi geri dönüş adresi olarak alır.
- Bir prosedür PROC deyimiyle başlar ve ENDP ile biter. Her ikisinden önce prosedürün adı belirtilir.
- PROC deyiminden sonra NEAR (intrasegment) ve FAR (intersegment) olarak tür belirtilir.
- NEAR ve FAR deyimlerinden sonra USE deyimiyle hangi register'ların kullanılacağı belirtilebilir. Belirtilen register'lar stack'a otomatik olarak push ve pop edilir.

Prosedürler

Örnek

```
0000                    SUMS    PROC NEAR
0000 03 C3                      ADD  AX,BX
0002 03 C1                      ADD  AX,CX
0004 03 C2                      ADD  AX,DX
0006 C3                         RET
0007                    SUMS    ENDP
0007                    SUMS1   PROC FAR
0007 03 C3                      ADD  AX,BX
0009 03 C1                      ADD  AX,CX
000B 03 C2                      ADD  AX,DX
000D CB                         RET
000E                    SUMS1   ENDP

000E                    SUMS3   PROC NEAR  USE  BX CX DX
0011 03 C3                      ADD  AX,BX
0013 03 C1                      ADD  AX,CX
0015 03 C2                      ADD  AX,DX
                                RET
001B                    SUMS    ENDP
```

Prosedürler

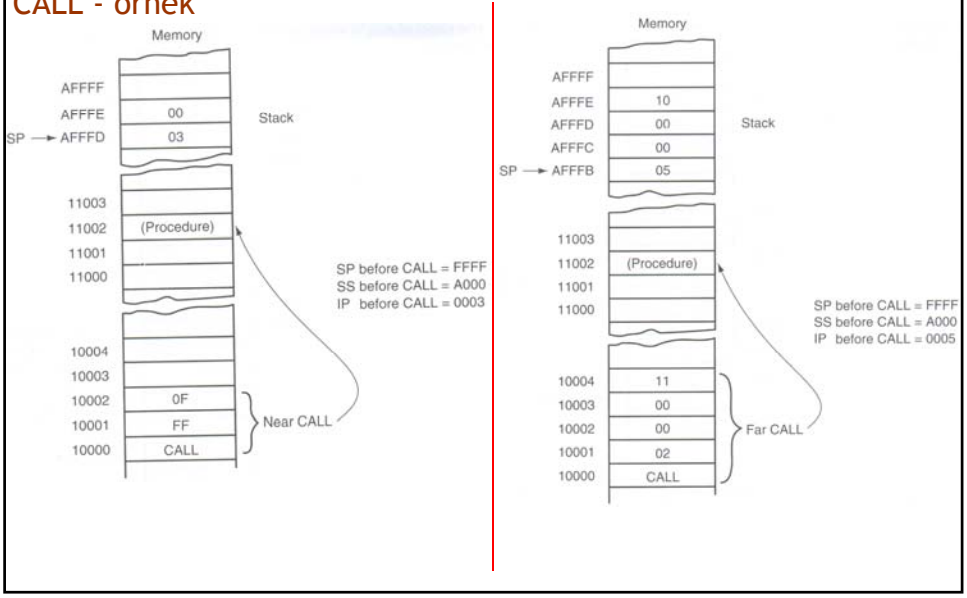
- Near return komutu stack'tan 16-bit değer alır ve IP'ye dönüş adresi olarak atar.
- Far return komutu stack'tan 32-bit değer alır ve IP ile CS'ye dönüş adresi olarak atar.

CALL

- CALL komutu adı verilen prosedürü çağırır ve dönüş adresini stack'a saklar.
- Near CALL komutu 3 byte boyutundadır. İlk byte opcode, ikinci ve üçüncü byte dönüş adresidir (±32K).
- 80386 ve üstü işlemcilerde protected mode'da dönüş adresi 4 byte (±2G) olur.
- Far CALL komutu 5 byte boyutundadır. İlk byte opcode, 2-3 IP ve 4-5 CS' nin yeni değeridir.
- CALL komutu IP ve CS değerlerini stack'a push eder.

Prosedürler

CALL - örnek



Prosedürler

CALL – register ile çağırma

- CALL komutları register operand kullanabilir (CALL BX).
- IP stack'a push edilir ve aynı segmentte BX offset adresine geçilir.
- Aşağıdaki örnekte CALL BX ile DISP prosedürü çağırılmaktadır.
- Aynı işlem CALL DISP şeklinde yapılabilir.

```
                        ;A DOS program that displays OK using the DISP procedure.
                        ;
                        .MODEL TINY                     ;select tiny model
0000                    .CODE                           ;start code segment
                        .STARTUP                        ;start program
0100 BB 0110 R                MOV  BX,OFFSET DISP       ;load BX with offset DISP
0103 B2 4F                    MOV  DL,'O'               ;display O
0105 FF D3                    CALL BX
0107 B2 4B                    MOV  DL,'K'               ;display K
0109 FF D3                    CALL BX
                        .EXIT
                        ;
                        ;Procedure that displays the ASCII character in DL
                        ;
0110                    DISP    PROC   NEAR
0110 B4 02                    MOV  AH,2                 ;select function 2
0112 CD 21                    INT  21H                  ;execute DOS function 2
0114 C3                       RET
0115                    DISP    ENDP
                        END
```

Prosedürler

CALL – dolaylı hafıza adresiyle çağırma

- CALL komutları indirect adres kullanabilir (CALL [BX]).
- Genellikle bir lookup tablodan seçilen farklı prosedürlerin kullanımında faydalıdır.
- Aşağıdaki örnekte EBX değerine göre ZERO, ONE veya TWO prosedürlerine geçiş yapılır.
- CALL FAR PTR [4*EBX] komutu far çağırma yapar veya eğer tablo doubleword tanımlı ise (DD) CALL TABLE[4*EBX] komutu far çağırma yapar.

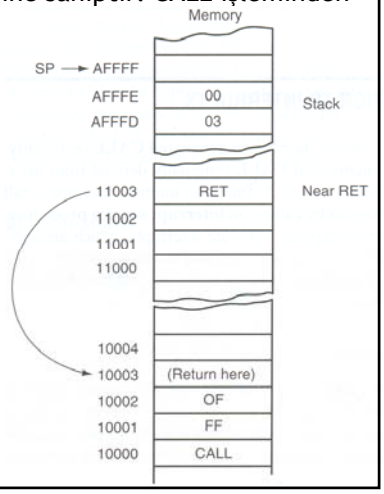
```
        ;Instruction that calls procedure ZERO, ONE, or TWO
        ;depending on the value in EBX
        ;
TABLE   DW    ZERO                  ;address of procedure ZERO
        DW    ONE                   ;address of procedure ONE
        DW    TWO                   ;address of procedure TWO


        CALL  TABLE[2*EBX]
```

Prosedürler

RET

- RET komutu near return için 16-bit sayıyı stack'tan alır ve IP'ye atar. Far return için 32-bit sayıyı stack'tan alır ve IP ve CS'ye atar.
- Aşağıdaki örnekte SS=A000 ve CS=1000 değerine sahiptir. CALL işleminden önce SP=FFFF ve IP=0003 değerindedir.
- CALL'dan sonra SP=FFFD, IP=0FFF.
- RET işleminden sonra SP=FFFF, IP=0003 ve Efektif adres=11003 olur.



Prosedürler

RET

- RET komutu dönüş adresini stack'tan sildikten sonra SP değerine belirli bir değer ekleyebilir. Böylece, prosedür çağrıldığında stack'a push edilen parametreler kadar değer eklenerek bu parametrelerin atılması sağlanır.
- Örnekte AX ve BX (4 byte) prosedür çağrılmadan önce stack'a push ediliyor. RET 4 ile dönüş adresinden sonra SP'e 4 ekleyerek AX,BX'i atar.

```
0000 B8 001E              MOV  AX,30
0003 BB 0028              MOV  BX,40
0006 50                   PUSH AX              ;stack parameter 1
0007 53                   PUSH BX              ;stack parameter 2
0008 E8 0066              CALL ADDM            ;add stack parameters


0071              ADDM    PROC NEAR
0071 55                   PUSH BP              ;save BP
0072 8B EC                MOV  BP,SP           ;address stack with BP
0074 8B 46 04             MOV  AX,[BP+4]       ;get parameter 1
0077 03 46 06             ADD  AX,[BP+6]       ;add parameter 2
007A 5D                   POP  BP              ;restore BP
007B C2 0004              RET  4               ;return, dump parameters
007E              ADDM    ENDP
```

Ödev

- Hafızada bir alanda bulunan ve herbirisi word boyutunda olan 100 adet sayıyı küçükten büyüğe doğru sıralanmış olarak hafızadaki ikinci bir diziye aktaran program yazınız.
- İstenen sıralama algoritması kullanılabilir.
- Program emu8086 ile yazılacaktır.
- Ödevin hem çıktısı teslim edilecek hemde kaynak kodu akcayol@gazi.edu.tr adresine e-posta ile gönderilecektir.