

## İKİLİ TABANDA ÇOK BAYTLI ÇARPMA

Aritmetik işlemler onlu sayı sisteminde yapılabileceği gibi diğer sayı sistemleri kullanılarak da yapılabilir. Örneğin sayısal bilgisayarlar, yalnız "1" ve "0" lardan oluşan ikili sayı sistemlerini kullandığından, ancak bu sayı sisteminde verilen sayılar üzerinde aritmetik işlem yapabilirler. Hesap makinelerinde, çarpılacak sayıları kullanıcı ondalık sistemde tuşlamaktadır. Fakat makine bu sayıları uygun bir tuş tarama programıyla ikili sisteme çevirip belleğine yerleştirmektedir. Makinede bu sayılar ikili sistemde çarpıldıktan sonra sonuç yine görüntüleyicide ondalık olarak görüntülenmektedir. Söz konusu çarpma işlemi, ikili tabanda çok byteli çarpma işlemi yapan bir paket programın koşturulması ile gerçekleştirilir. Bu program önceden makinenin ROM belleğine yerleştirilmiştir. O halde esas sorun ikili sistemde çok byteli çarpma işlemi yapacak bir programın çalıştırılmasından ibarettir. Daha sonra da inceleneceği gibi, sayısal bilgisayarlarda yapılan ikili çarpma işlemi, el ile yapılan onlu çarpma işlemine benzer. Tek fark onlu hanelerin yerini ikili hanelerin almasıdır.

Sayısal bilgisayarlar genellikle belirli kelime uzunluklarına sahip olacak şekilde yapılabildiklerinden, çok sayıda bit ile ifade edilebilecek sayıların çarpılması için özel bir çarpma emri kullanamaz. Bunun için, çarpma işlemini gerçekleştiren uygun bir algoritmanın geliştirilmesi zorunlu olmaktadır. Örneğin 8 bitlik bir mikrobilgisayarda, registerler ve hafıza hücreleri 8 bitlik uzunluğa sahip olduğundan, bir adımda ancak 8 bitlik bilgi işlenebilir. 32 bitlik sayıları çarpabilmek için bu 8 bit uzunluğundaki registerlerden 4 adet kullanılarak 32 bitlik bir register oluşturabiliriz. Böylece uygun bir algoritma ile çok byte'lı bir çarpma işlemi gerçekleştirilebilir.

Özel çarpma emri olmayan bir sayısal bilgisayarda 8 bitlik iki sayının çarpılması aşağıdaki örneklerle gösterilmiştir. Örnekte görüldüğü gibi çarpım, çarpılan sayıların iki katı uzunluktadır.

		0 0 1 0 0 1 0 0
	x	0 0 0 1 0 1 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 1 0 0 1 0 0
		0 0 0 0 0 0 0 0
		0 0 1 0 0 1 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
	+	0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0

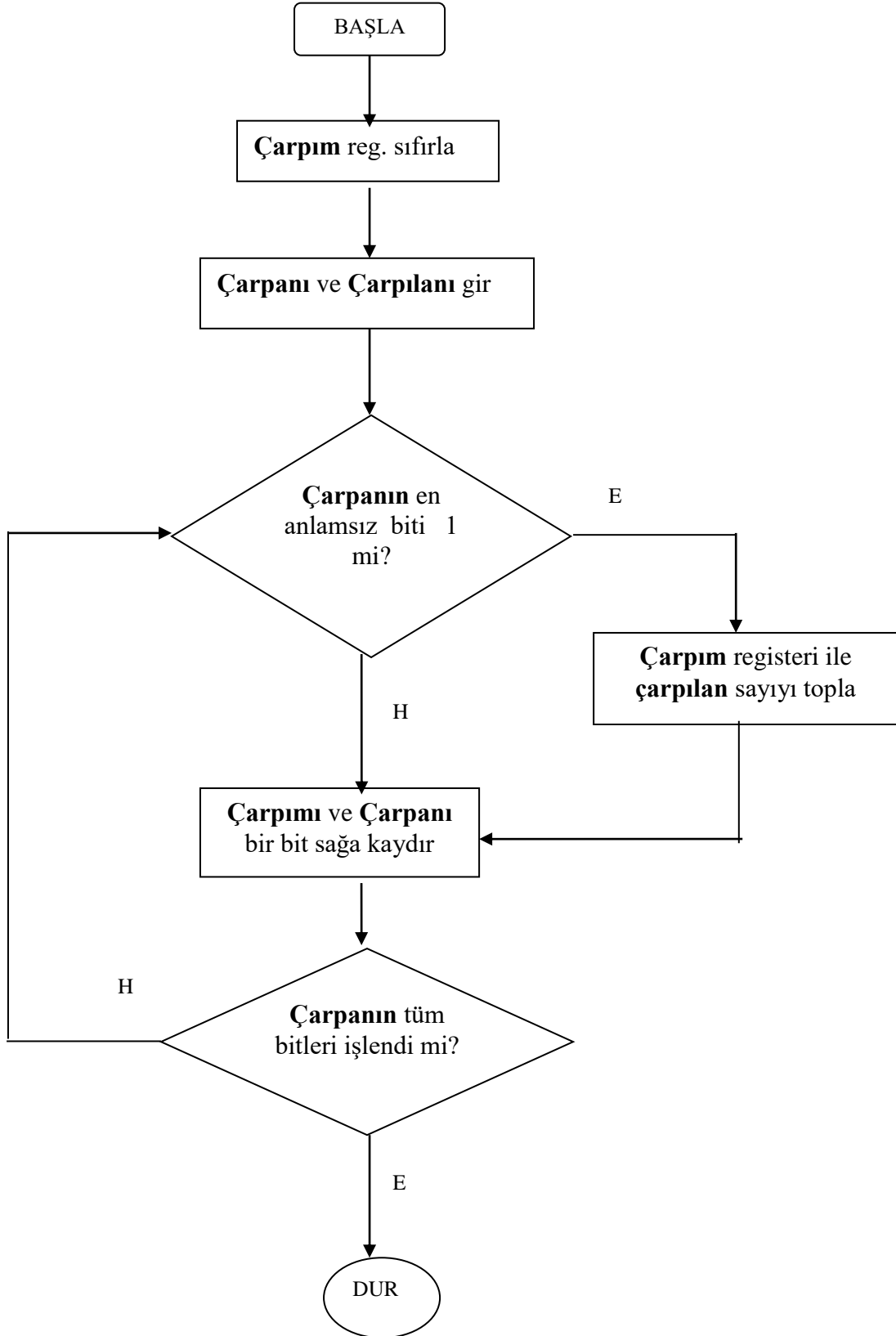
36 çarpılan
x 20 çarpan
720 çarpım

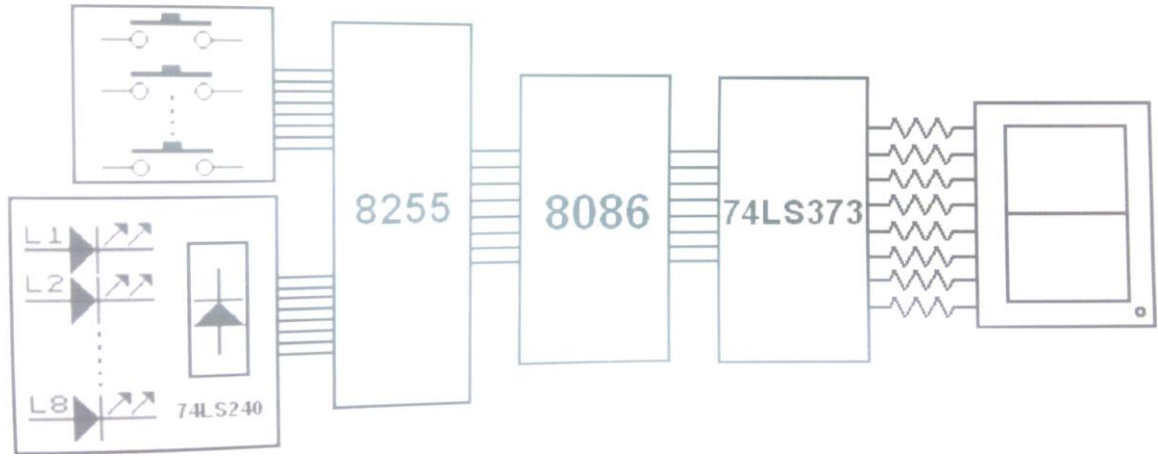
İkili tabanda çarpmaya ilişkin bir algoritma aşağıda verilmiştir.

1. Çarpan ve çarpılan sayıları kaydet
2. Çarpım registerini sıfırla
3. Çarpan sayının en anlamsız bitinden başlamak üzere, bitlerini tek tek kontrol et.
  - a. Eğer kontrol edilen bit " 0 " ise çarpım registerini bir bit sağa kaydır.
  - b. Eğer kontrol edilen bit " 1 " ise çarpım registerini çarpılan sayı ile topla ve bir bit sağa kaydır.
4. Tüm bitlerin kontrolü bittiğinde, çarpım registerinin içeriği çarpımın sonucudur.

Bu işlemler daha açık bir şekilde akış diyagramında ( sayfa 3 ) gösterilmiştir.

İkili tabanda çok byte'lı çarpma işlemi akış diyagramı aşağıdaki gibi verilebilir.





Donanım-blok şeması

-----8086 KODU-----

*;Bu program AX ve BX registerlerindeki 16 bit uzunluğundaki iki sayıyı ikili tabanda çarparak sonucu DX ve CX registerlerinde saklar. 32 bitlik sonuç, PortA ya bağlı butonlardan button0-3'e basılarak 32 bitlik bu sonucun her biri 8-bitlik PORTB deki LED lerde görüntülenir. Ayrıca 7-segmentli displaye de 32 bitlik sonucun hangi kısmının PORTB de görüntülendiğini belirtmek için 1-4 arasındaki rakamlar yansıtılır.*

```
CNT3      EQU  3FD6H      ; 8255 control word port adresi
APORT3     EQU  3FD0H      ; 8255 PORTA adresi
BPORT3     EQU  3FD2H      ; 8255 PORTB adresi
FND        EQU  3FF0H      ;FND port address (7-segment Display)
```

CODE SEGMENT

```
ASSUME     CS:CODE, DS:CODE
```

```
ORG 0
```

start:

```
MOV  AX,0012h      ;Carpan
MOV  BX,0033h      ;Carpilan
```

```
MOV  DX,0000h      ;Carpim sonucunun ust 16 biti
MOV  CX,0000h      ;Carpim sonucunun alt 16 biti
```

```
MOV  SI,10h        ;dongu degişkeni (16 kez dönmesi için)
```

tekrar:

```
MOV  DI,AX          ;DI=AX
AND  DI,01h         ;DI'nın LSB biti haricindeki bitlerini sıfırla
XOR  DI,01h         ;Carpan'ın en anlamsız biti lojik 1 mi?
JZ   topla_kaydir    ;Evet ise Carpim sonucunu Carpilan ile topla
                        ; ve bir bit sağa kaydır
```

```
CLC
```

devam:

```
RCR  DX,1           ;Capım sonucunun üst 16 bitini 1 bit sağa kaydır.
```

```
RCR  CX,1           ;Capım sonucunun alt 16 bitini 1 bit sağa kaydır.
```

```
SHR  AX,1           ;Carpan'ı bir bit sağa kaydır
```

DEC	SI	<i>;döngü değişkenini bir azalt</i>
CMP	SI,0	<i>;döngü değişkeni sıfır mı?</i>
JNZ	tekrar	<i>;Eğer sıfır değil ise aynı işlemleri tekrarla</i>
JMP	son	<i>;Çarpım sonucunu göster</i>

topla\_kaydır:

ADD	DX,BX	<i>;DX=DX+AX</i>
JMP	devam	

son:

MOV	SI,DX	<i>;DX teki bilgiyi SI registerine yedekle. DX, I/O erişiminde kullanılacak.</i>
MOV	SP,2000H	<i>;stack pointerın başlangıç adresini 2000h olarak belirle</i>
MOV	AX,CS	<i>; DS=CS</i>
MOV	DS,AX	<i>;Data Segment=Code Segment</i>

; < Set 8255 control word register>

MOV	DX,CNT3	<i>;8255 control portunu aktif eder</i>
MOV	AL,90H	<i>;8255 control word registerine yazılacak veri</i>
OUT	DX,AL	<i>; 90h verisini 8255 control portuna gönderir</i>
		<i>; 8255 mode0, portA=Input, portB ve portC=Output</i>

dongu:

MOV	BX,SI	<i>;SI da yedeklenmiş olan bilgiyi BX'e geri yükle</i>
MOV	DX,APORT3	<i>;PORTA nın adresini DX registerine yükle</i>
IN	AL,DX	<i>;PORTA daki buton bilgisini AL'ye yükle</i>
NOT	AL	<i>; Buton bilgisini low dan high'a çevir</i>
CMP	AL,08H	<i>; 4. butona basıldı mı?</i>
JNZ	J1	<i>; Hayır ise J1'e dallan</i>
MOV	AL,BH	<i>; AL=BH , 32 bitlik çarpım sonucunun ilk 8 bitini AL'ye yükle</i>
MOV	AH, 04	<i>;7 segment display verisi</i>
JMP	yazdir	

J1:

CMP	AL,04H	<i>; 3. butona basıldı mı?</i>
JNZ	J2	<i>; Hayır ise J2'e dallan</i>
MOV	AL,BL	<i>;AL=BL , 32 bitlik çarpım sonucunun ikinci 8 bitini AL'ye yükle</i>
MOV	AH, 03	<i>;7 segment display verisi</i>
JMP	yazdir	

J2:

CMP	AL,02H	<i>; 2. butona basıldı mı?</i>
JNZ	J3	<i>; Hayır ise J3'e dallan</i>
MOV	AL,CH	<i>;AL=CH , 32 bitlik çarpım sonucunun üçüncü 8 bitini AL'ye yükle</i>
MOV	AH, 02	<i>;7 segment display verisi</i>
JMP	yazdir	

J3:

MOV	AL,CL	<i>;AL=CL , 32 bitlik çarpım sonucunun dördüncü 8 bitini AL'ye yükle</i>
MOV	AH, 01	<i>;7 segment display verisi</i>

yazdir:

```

MOV DX,BPORT3      ;DX'e BPORT3 ün adresini yükle
OUT  DX,AL          ;sonucu porta gönder

MOV BX,OFFSET FONT  ; Lookup tablosunun başlangıç adresini BX'e yükle
MOV AL,AH           ;AH daki veri AL'ye kopyalanır
XLATB               ; AL nin içindeki veriyi BX ile adreslenen LookUp
                   ; tablosundaki veri ile değiştir
MOV DX,FND          ;7-Segmentli displayi aktif eder
OUT  DX,AL          ; AL deki veriyi 7-segmentli displaye gönder

JMP  dongu

```

```

FONT      ;      Dgfedcba
          DB  11000000B ;0
          DB  11111001B ;1
          DB  10100100B ;2
          DB  10110000B ;3
          DB  10011001B ;4
          DB  10010010B ;5
          DB  10000010B ;6
          DB  11011000B ;7
          DB  10000000B ;8
          DB  10011000B ;9

```

```

CODE ENDS
      END  START

```