

BIL 362 Mikroişlemciler

M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü



Konular

Aritmetik ve Mantık (Arithmetic and Logic) Komutları

- Toplama, Çıkarma ve Karşılaştırma Komutları
- Çarpma ve Bölme Komutları
- Temel Mantık Komutları
- Kaydırma ve Döndürme (Shift and Rotate) Komutları
- String Karşılaştırma Komutları

Toplama

- ADD (Addition) mikroişlemcide birçok şekilde kullanılabilir.
- Register, immediate, memory-to-register, array, increment, addition-with-carry toplama işlemleri yapılabilir.
- Memory-to-memory ve segment register toplama işlemleri yapılamaz.
- Segment register'ları sadece MOV, PUSH veya POP yapılabilir.

Toplama

Register toplama

Assembly Language	Operation
ADD AL,BL	AL = AL + BL
ADD CX,DI	CX = CX + DI
ADD EBP,EAX	EBP = EBP + EAX
ADD CL,44H	CL = CL + 44H
ADD BX,245FH	BX = BX + 245FH
ADD EDX,12345H	EDX = EDX + 12345H
ADD [BX],AL	AL adds to the byte contents of the data segment memory location addressed by BX with the sum stored in the same memory location
ADD CL,[BP]	The byte contents of the stack segment memory location addressed by BP add to CL with the sum stored in CL
ADD AL,[EBX]	The byte contents of the data segment memory location addressed by EBX add to AL with the sum stored in AL
ADD BX,[SI+2]	The word contents of the data segment memory location addressed by SI + 2 add to BX with the sum stored in BX
ADD CL,TEMP	The byte contents of data segment memory location TEMP add to CL with the sum stored in CL
ADD BX,TEMP[DI]	The word contents of the data segment memory location addressed by TEMP + DI add to BX with the sum stored in BX
ADD [BX+DI],DL	DL adds to the byte contents of the data segment memory location addressed by BX + DI with the sum stored in the same memory location
ADD BYTE PTR [DI],3	A 3 adds to the byte contents of the data segment memory location addressed by DI with the sum stored in the same location
ADD BX,[EAX+2*ECX]	The word contents of the data segment memory location addressed by EAX plus 2 times ECX add to BX with the sum stored in BX

Toplama

Örnek

```
0000 03 C3      ADD  AX, BX
0002 03 C1      ADD  AX, CX
0004 03 C2      ADD  AX, DX
```

- Örnekte AX, BX, CX ve DX register'ları toplanır ve 16-bti AX register'ına sonuç saklanır.
- Bir toplama işlemi yapıldığında FLAG register değeri değişir.
- ADD komutu sign, zero, carry, auxiliary carry, parity ve overflow bitleri değişir.

Toplama

Immediate toplama

- Sabit bir değer toplanır.

```
0000 B2 12      MOV  DL, 12H
0002 80 C2 33    ADD  DL, 33H
```

- Önce DL'ye 12H yüklenir.
- Daha sonra 33H ile DL içeriği toplanır.
- Toplama işlemi sonucu 45H DL register'ına saklanır.
- Bu işlem sonucunda
FLAG değişimleri yandaki gibidir.

Z = 0 (result not zero)

C = 0 (no carry)

A = 0 (no half-carry)

S = 0 (result positive)

P = 0 (odd parity)

O = 0 (no overflow)

Toplama

Memory-to-register toplama

- İstenen hafıza alanı bir register'la toplanır.

```
0000 BF 0000 R    MOV  DI,OFFSET NUMB      ;address NUMB
0003 B0 00        MOV  AL,0                ;clear sum
0005 02 05        ADD  AL,[DI]             ;add NUMB
0007 02 45 01     ADD  AL,[DI+1]           ;add NUMB+1
```

- Önce DI'ya NUMB offset adresi aktarılır.
- Daha sonra AL'ye 0 değeri aktarılır.
- Data segment'te DI offset adresinin içeriği AL register'ına eklenir.
- Son olarak AL register'ına data segment'te DI+1 offset adresinin içeriği eklenir.

Toplama

Array toplama

- Hafızada bir alanda bulunan dizi değerleri toplanır.

```
0000 B0 00        MOV  AL,0                ;clear sum
0002 BE 0003      MOV  SI,3                ;address element 3
0005 02 84 0000 R  ADD  AL,ARRAY[SI]       ;add element 3
0009 02 84 0002 R  ADD  AL,ARRAY[SI+2]     ;add element 5
000D 02 84 0004 R  ADD  AL,ARRAY[SI+4]     ;add element 7
```

- Önce AL'ye aktarılır.
- Daha sonra SI register'ına 3 değeri aktarılır.
- ARRAY[SI] operand'ı SI=3 olduğu için dizideki 3. elemanı gösterir.
- Dizinin 3. elemanı, 5. elemanı ve 7. elemanı toplanarak sonuç AL register'ına aktarılır.

Toplama

Increment toplama

- INC komutu bir register'a (segment register'ları hariç) veya hafıza alanına 1 ekler.
- Indirect memory artırımında BYTE PTR, WORD PTR veya DWORD PTR ile data boyutu belirtilmelidir.

Assembly Language	Operation
INC BL	BL = BL + 1
INC SP	SP = SP + 1
INC EAX	EAX = EAX + 1
INC BYTE PTR[BX]	Adds 1 to the byte contents of the data segment memory location addressed by BX
INC WORD PTR[SI]	Adds 1 to the word contents of the data segment memory location addressed by SI
INC DWORD PTR[ECX]	Adds 1 to the doubleword contents of the data segment memory location addressed by ECX
INC DATA1	Adds 1 to the contents of data segment memory location DATA1

Toplama

Increment toplama - örnek

```

0000 BF 0000 R    MOV  DI,OFFSET NUMB      ;address NUMB
0003 B0 00        MOV  AL,0                ;clear sum
0005 02 05        ADD  AL,[DI]             ;add NUMB
0007 47          INC  DI                   ;increment DI
0008 02 05        ADD  AL,[DI]             ;add NUMB+1

```

- Önce data segment'te NUMB offset adresinin içeriği AL'ye eklenir.
- Ardından DI adres değeri 1 artırılarak DI+1 offset adresin içeriği AL'ye eklenir.
- Word boyutundaki data için DI değerini 2 artırmak, doubleword data için 4 artırmak gerekir.

Toplama

Carry ile toplama

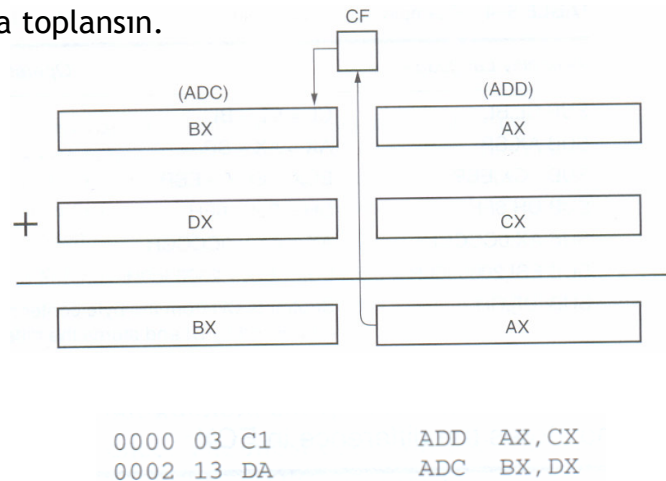
- ADC (addition-with-carry) komutu operand'a carry bitinin değerini ekler.

Assembly Language	Operation
ADC AL,AH	AL = AL + AH + carry
ADC CX,BX	CX = CX + BX + carry
ADC EBX,EDX	EBX = EBX + EDX + carry
ADC DH,[BX]	The byte contents of the data segment memory location addressed by BX add to DH with the sum stored in DH
ADC BX,[BP+2]	The word contents of the stack segment memory location addressed by BP plus 2 add to BX with the sum stored in BX
ADC ECX,[EBX]	The doubleword contents of the data segment memory location addressed by EBX add to ECX with the sum stored in ECX

Toplama

Carry ile toplama - örnek

- AX ve BX te bulunan 32-bit data CX ve DX te bulunan 32-bit datayla toplansın.



Çıkarma

- SUB (subtraction) komutu bir operand'ı diğerinden çıkarmak için kullanılır.
- Özel formu DEC (decrement) komutudur ve 1 çıkarır.

Assembly Language	Operation
SUB CL,BL	CL = CL - BL
SUB AX,SP	AX = AX - SP
SUB ECX,EBP	ECX = ECX - EBP
SUB DH,6FH	DH = DH - 6FH
SUB AX,0CCCCCH	AX = AX - 0CCCCCH
SUB ESI,2000300H	ESI = ESI - 2000300H
SUB [DI],CH	Subtracts CH from the byte contents of the data segment memory addressed by DI and stores the difference in the same memory location
SUB CH,[BP]	Subtracts the byte contents of the stack segment memory location addressed by BP from CH and stores the difference in CH
SUB AH,TEMP	Subtracts the byte contents of memory location TEMP from AH and stores the difference in AH
SUB DI,TEMP[ESI]	Subtracts the word contents of the data segment memory location addressed by TEMP plus ESI from DI and stores the difference in DI
SUB ECX,DATA1	Subtracts the doubleword contents of memory location DATA1 from ECX and stores the difference in ECX

Çıkarma

Register çıkarma

- Bir register'ın değeri diğerinden çıkarılır.

```
0000 2B D9      SUB  BX, CX
0002 2B DA      SUB  BX, DX
```

- Örnekte 16-bit CX ve DX register'ları BX'ten çıkarılır ve sonuç BX'te saklanır.
- Her çıkarmadan sonra FLAG bitleri değiştirilir.

Çıkarma

Immediate çıkarma

- Bir sabit değer bir register'dan çıkarılır. Sabit değer instruction'la birlikte verilir.

```
0000 B5 22      MOV  CH, 22H
0002 80 ED 44    SUB  CH, 44H
```

- Örnekte 8-bit 22H değeri CH register'ına aktarılır.
- Ardından 22H değerinden 44H değeri çıkartılır. Sonuç (0DEH) CH register'ında saklanmaktadır. FLAG durumları aşağıdadır.

Z = 0 (result not zero)

C = 1 (borrow)

A = 1 (half-borrow)

S = 1 (result negative)

P = 1 (even parity)

O = 0 (no overflow)

Çıkarma

Decrement çıkarma

- Bir hafıza alanından veya register'dan 1 çıkarılır.
- DEC komutu hafıza adreslediğinde BYTE PTR, WORD PTR veya DWORD PTR kullanılmalıdır.
- DEC [SI] yerine DEC BYTE PTR [SI], DEC WORD PTR [SI] veya DEC DWORD PTR [SI] kullanılmalıdır.

Assembly Language	Operation
DEC BH	BH = BH - 1
DEC CX	CX = CX - 1
DEC EDX	EDX = EDX - 1
DEC BYTE PTR[DI]	Subtracts 1 from the byte contents of the data segment memory location addressed by DI
DEC WORD PTR[BP]	Subtracts 1 from the word contents of the stack segment memory location addressed by BP
DEC DWORD PTR[EBX]	Subtracts 1 from the doubleword contents of the data segment memory location addressed by EBX
DEC NUMB	Subtracts 1 from the contents of data segment memory location NUMB

Çıkarma

Ödünç alma ile (subtraction-with-borrow) çıkarma

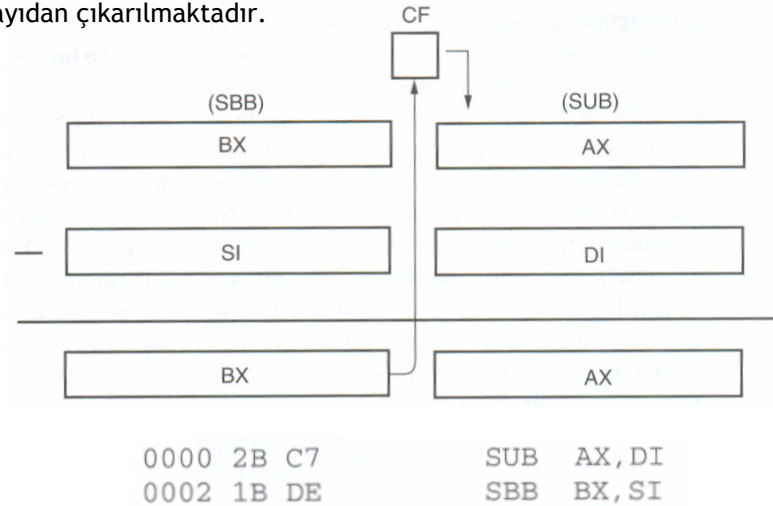
- SBB (subtraction-with-borrow) hedef operand'dan diğer operand ile carry biti çıkarılır.
- SBB komutu hafıza adreslediğinde BYTE PTR, WORD PTR veya DWORD PTR kullanılmalıdır. SBB [SI] yerine SBB BYTE PTR [SI], SBB WORD PTR [SI] veya SBB DWORD PTR [SI] kullanılmalıdır.

Assembly Language	Operation
SBB AH,AL	AH = AH - AL - carry
SBB AX,BX	AX = AX - BX - carry
SBB EAX,ECX	EAX = EAX - ECX - carry
SBB CL,2	CL = CL - 2 - carry
SBB BYTE PTR[DI],3	Both 3 and carry subtract from the data segment memory location addressed by DI
SBB [DI],AL	Both AL and carry subtract from the data segment memory location addressed by DI
SBB DI,[BP+2]	Both carry and the word contents of the stack segment memory location addressed by BP plus 2 subtract from DI
SBB AL,[EBX+ECX]	Both carry and the byte contents of the data segment memory location addressed by EBX plus ECX subtract from AL

Çıkarma

Ödünç alma ile çıkarma - örnek

- BX ve AX register'larında bulunan 32-bit sayı SI ve DI'da bulunan 32-bit sayıdan çıkarılmaktadır.



Karşılaştırma (Comparison)

- **CMP** (compare) komutu çıkarma işlemi yapar, hedef operand değişmez.
- Genellikle bir **JUMP** komutu izler ve **FLAG** ları kontrol eder.
- **Memory-to-memory** ve **segment register** karşılaştırılmaz.

Assembly Language	Operation
CMP CL,BL	CL – BL
CMP AX,SP	AX – SP
CMP EBP,ESI	EBP – ESI
CMP AX,2000H	AX – 2000H
CMP [DI],CH	CH subtracts from the byte contents of the data segment memory location addressed by DI
CMP CL,[BP]	The byte contents of the stack segment memory location addressed by BP subtracts from CL
CMP AH,TEMP	The byte contents of data segment memory location TEMP subtracts from AH
CMP DI,TEMP[BX]	The word contents of the data segment memory location addressed by TEMP plus BX subtracts from DI
CMP AL,[EDI+ESI]	The byte contents of the data segment memory location addressed by EDI plus ESI subtracts from AL

Karşılaştırma (Comparison)

- Aşağıdaki örnekte **AL** register'ı **10H** ile karşılaştırılır.
- Şartlı dallanma (conditional jump) komutları **JA** (jump above) ve **JB** (jump below) karşılaştırmanın sonucuna göre atlamayı gerçekleştirir.
- **JA** komutu eğer **AL** register'ı **10H**'dan büyükse atlamayı gerçekleştirir, **JB** komutu küçükse atlamayı gerçekleştirir.

```

0000 3C 10      CMP  AL,10H      ;compare AL against 10H
0002 73 1C      JAE  SUBER     ;if AL is 10H or above

```

- Örnekte **JAE** (jump above or equal) komutu **AL** register'ı **10H** ise veya **10H**'dan büyükse programı **SUBER** etiketinin olduğu satıra atlatır.
- **JBE** (jump below or equal) komutu **AL** register'ı **10H** ise veya **10H**'dan küçükse ilgili satıra atlatır.

Çarpma ve Bölme

Çarpma (Multiplication)

- Çarpma işlemi byte, word veya doubleword datayla yapılabilir.
- İşaretili (signed) integer (IMUL) veya işaretsiz (unsigned) integer (MUL) komutları kullanılabilir.
- Çarpım sonucu her zaman iki kat genişliğindedir. 8-bit iki sayının çarpım sonucu 16-bit, 16-bit iki sayının çarpımı 32-bit ve 32-bit iki sayının çarpımı 64-bit olur.
- O ve C flag bitleri etkilenir. Eğer 8-bit iki sayının sonucunda soldaki 8-bit 0 ise O=0 ve C=0 olur. C=1 olması sonucun 16-bit olduğunu gösterir.
- İki 16-bit sayının çarpım sonucunda C=0 ve O=0 olursa soldaki 16-bit 0 değerindedir.

Çarpma ve Bölme

8-bit çarpma

- 8-bit çarpmada çarpılan her zaman AL (işaretili/işaretsiz) olur.
- Çarpan 8-bit bir register veya bir hafıza alanı olur.
- Çarpma komutunda 1 operand vardır ve default çarpılan AL register'ıdır.
- MUL BL komutu AL ile BL'yi işaretsiz çarpar. Çarpma sonucu 16-bit AX register'ına kaydedilir.

Assembly Language	Operation
MUL CL	AL is multiplied by CL; the unsigned product is in AX
IMUL DH	AL is multiplied by DH; the signed product is in AX
IMUL BYTE PTR[BX]	AL is multiplied by the byte contents of the data segment memory location addressed by BX; the signed product is in AX
MUL TEMP	AL is multiplied by the byte contents of data segment memory location TEMP; the unsigned product is in AX

Çarpma ve Bölme

8-bit çarpma - örnek

- BL ve CL register'larındaki 8-bit unsigned sayılar çarpılmaktadır.
- 16-bit sonuç DX register'ına kaydedilmektedir.
- Bu işlem AL register'ı kullanılarak yapılacağından bir register'ı AL'ye aktarmak gerekir.
- Signed sayılar için IMUL komutu kullanılır.
- Negatif sayılar için ikinin tümleyeni, pozitif sayılar için binary form kullanılır.

```

0000 B3 05      MOV    BL,5           ;load data
0002 B1 0A      MOV    CL,10        ;load data
0004 8A C1      MOV    AL,CL        ;position data
0006 F6 E3      MUL    BL           ;multiply
0008 8B D0      MOV    DX,AX        ;position product

```

Çarpma ve Bölme

16-bit çarpma

- 16-bit çarpmada çarpılan her zaman AX olur.
- Çarpan 8-bit bir register veya bir hafıza alanı olur.
- Çarpım sonucu DX-AX içerisine saklanır. DX most-significant 16-bit, AX least-significant 16-bit sayıyı saklar.

Assembly Language	Operation
MUL CX	AX is multiplied by CX; the unsigned product is in DX-AX
IMUL DI	AX is multiplied by DI; the signed product is in DX-AX
MUL WORD PTR[SI]	AX is multiplied by the word contents of the data segment memory location addressed by SI; the unsigned product is in DX-AX

Çarpma ve Bölme

32-bit çarpma

- 80386 ve üstü işlemcilerde 32-bit çarpma yapılabilir.
- 32-bit çarpma işaretli (IMUL) veya işaretsiz (MUL) olabilir.
- EAX register'ı komut tarafından belirlenen operand'la çarpılır.
- Sonuç EDX-EAX içerisine saklanır. EDX most-significant 32-bit ve EAX least significant 32-bit kısmı saklar.

Assembly Language	Operation
MUL ECX	EAX is multiplied by ECX; the unsigned product is in EDX-EAX
IMUL EDI	EAX is multiplied by EDI; the signed product is in EDX-EAX
MUL DWORD PTR[ESI]	EAX is multiplied by the doubleword contents of the data segment memory location address by ESI; the unsigned product is in EDX-EAX

Çarpma ve Bölme

Bölme (Division)

- 8086-80286 işlemcilerde bölme 8-bit veya 16-bit sayılarda, 80386-Pentium 4 işlemcilerde 32-bit sayılarda yapılır.
- IDIV işaretli sayılarda, DIV işaretsiz sayılarda bölme yapar.
- Bölünen her zaman iki kat genişliğindedir.
- AX=3000 ise ve 2 ile bölünürse sonuç 1500 olur ve AL registerına sığmaz. (1500 = **10111011100**)
- Bu durumda overflow oluşur ve divide-error-interrupt gerçekleşir.

Çarpma ve Bölme

8-bit bölme

- 8-bit bölmede bölünen AX register'ında bulunur ve 8-bit register veya hafızadaki bir değere bölünür.
- AL register'ına bölüm sonucu, AH register'ına kalan saklanır.
- İşaretsiz sayılar için bölüm negatif veya pozitif olabilir, kalan bölünenin işaretini saklar ve integer sayıdır.
- AX=0010H(+16), BL=FDH(-3) ve IDIV BL komutu çalışırsa AX=01FBH olur. AL=-5 bölüm sonucu ve AH=1 kalan değerdir.

Assembly Language	Operation
DIV CL	AX is divided by CL; the unsigned quotient is in AL and the unsigned remainder is in AH
IDIV BL	AX is divided by BL; the signed quotient is in AL and the signed remainder is in AH
DIV BYTE PTR[BP]	AX is divided by the byte contents of the stack segment memory location addressed by BP; the unsigned quotient is in AL and the unsigned remainder is in AH

Çarpma ve Bölme

8-bit bölme - devam

- 8-bit bölmede sayılar genellikle 8-bit olur. Bölünen sayının AX register'ı için word boyutuna dönüştürülmesi gerekir.
- İşaretsiz sayılarda soldaki 8-bit (AH) 0 yapılır (zero-extended). İşaretsiz sayılarda AH register'ına AL'nin işaret biti aktarılır (sign-extended).
- **CBW** (convert byte to word) komutu işaretli sayı olan AL register'ının içeriğini AX boyutuna dönüştürür.

Çarpma ve Bölme

8-bit bölme - örnek

- Unsigned byte sayı NUMB, unsigned sayı NUMB1'e bölünür.
- Bölüm sonucu ANSQ ve kalan ANSR hafıza alanlarına aktarılır.

```

0000 A0 0000 R      MOV AL,NUMB      ;get NUMB
0003 B4 00          MOV AH,0        ;zero-extend
0005 F6 36 0002 R    DIV NUMB1      ;divide by NUMB1
0009 A2 0003 R      MOV ANSQ,AL     ;save quotient
000C 88 26 0004 R    MOV ANSR,AH    ;save remainder

```

- Aşağıda signed bölme yapılıyor ve sign extent için CBW kullanılmaktadır.

```

MOV AL, NUMB
CBW
IDIV NUMB1
MOV ANSQ, AL
MOV ANSR, AH

```

Çarpma ve Bölme

16-bit bölme

- 16-bit bölmede bölünen 32-bit sayı DX-AX register'ında bulunur ve 16-bit register veya hafızadaki bir değere bölünür.
- AX register'ına bölüm sonucu, DX register'ına kalan saklanır.
- 16-bit unsigned bölünen sayı için DX=0 yapılır (zero-extended) ve signed sayı için DX=sign(AX) yapılır (sign-extended).
- CWD(convert word to doubleword) 16-bit sign extend yapar.

Assembly Language	Operation
DIV CX	DX-AX is divided by CX; the unsigned quotient is in AX and the unsigned remainder is in DX
IDIV SI	DX-AX is divided by SI; the signed quotient is in AX and the signed remainder is in DX
DIV NUMB	DX-AX is divided by the word contents of data segment memory NUMB; the unsigned quotient is in AX and the unsigned remainder is in DX

Çarpma ve Bölme

16-bit bölme - örnek

- Aşağıda iki 16-bit signed sayının bölümü görülmektedir.
- AX = -100, CX = +9 ve CWD komutu AX=-100(16-bit) değerini DX-AX = -100(32-bit) yapar.
- Bölme sonucunda AX=-11(bölüm) ve DX=-1(kalan) değerine sahiptir.

```
0000 B8 FF9C      MOV  AX,-100      ;load a -100
0003 B9 0009      MOV  CX,9         ;load +9
0006 99          CWD                     ;sign-extend
0007 F7 F9      IDIV  CX
```

Çarpma ve Bölme

32-bit bölme

- 80386-Pentium 4 işlemcilerde işaretli veya işaretsiz sayılarda 32-bit bölme yapılır.
- 64-bit bölünen EDX-EAX değeri operand'la belirlenen değere bölünür. EAX 32-bit bölüm sonucunu ve EDX kalanı tutar.
- CDQ (convert doubleword to quadword) 32-bit EAX'i 64-bit EDX-EAX'e dönüştürür.

Assembly Language	Operation
DIV ECX	EDX-EAX is divided by ECX; the unsigned quotient is in EAX and the unsigned remainder is in EDX
IDIV DATA4	EDX-EAX is divided by the doubleword contents in data segment memory location DATA4; the signed quotient is in EAX and the signed remainder is in EDX
DIV DWORD PTR[EDI]	EDX-EAX is divided by the doubleword contents of the data segment memory location addressed by EDI; the unsigned quotient is in EAX and the unsigned remainder is in EDX

Çarpma ve Bölme

32-bit bölme - örnek

- Örnekte AX değeri BL'ye bölünür ve sonuç yuvarlanır (round).
- ADD AH,AH ile kalan ikiyle çarpılır ve BL'den büyükse AL (bölüm sonucu) 1 artırılır.
- 0.5 ten büyük değerler bir üst tamsayıya yuvarlanmış olur.

```

0000 F6 F3          DIV  BL          ;divide
0002 02 E4          ADD  AH,AH       ;double remainder
0004 3A E3          CMP   AH,BL      ;test for rounding
0006 72 02          JB    NEXT       ;if OK
0008 FE C0          INC   AL         ;round
000A                NEXT:

```

Temel Mantık Komutları

- AND, OR, XOR ve NOT temel mantık komutlarıdır.
- TEST ve NEG komutları AND ile NOT komutlarının özel şeklidir.
- Mantık komutlar bitlere set,clear ve complement işlemi yapar.
- Mantık komutlar carry ve overflow bitlerini 0 yapar. Diğer FLAG bitleri işlem sonucuna göre değer alır.
- Mantık komutları yazılımlarda bit seviyesinde kontrol gerçekleştirir.

Temel Mantık Komutları

AND

- AND işlemi mantıksal çarpma yapar. Memory-to-memory ve segment register adresleme yapılamaz. Aşağıdaki doğruluk tablosuna sahiptir.

A	B	T
0	0	0
0	1	0
1	0	0
1	1	1



- AND işlemi binary sayıda bir bit değerini 0 yapmak için kullanılır. (masking)

(a) (b)

x x x x	x x x x	Unknown number
• 0 0 0 0	1 1 1 1	Mask
0 0 0 0	x x x x	Result

```

0000 BB 3135      MOV  BX,3135H      ;load ASCII
0003 81 E3 0F0F    AND   BX,0F0FH      ;mask BX
  
```

BX mask yapılır

Temel Mantık Komutları

AND - devam

- AND işlemleri aşağıdaki tabloda görülmektedir. Memory-to-register veya register-to-register veya immediate adresleme

Assembly Language	Operation
AND AL,BL	AL = AL and BL
AND CX,DX	CX = CX and DX
AND ECX,EDI	ECX = ECX and EDI
AND CL,33H	CL = CL and 33H
AND DI,4FFFH	DI = DI and 4FFFH
AND ESI,34H	ESI = ESI and 34H
AND AX,[DI]	The word contents of the data segment memory location addressed by DI are ANDed with AX
AND ARRAY[SI],AL	The byte contents of the data segment memory location addressed by ARRAY plus SI are ANDed with AL
AND [EAX],CL	CL is ANDed with the byte contents of the data segment memory location addressed by ECX

Temel Mantık Komutları

OR

- OR işlemi mantıksal toplama yapar. Memory-to-memory ve segment register adresleme yapılamaz. Aşağıdaki doğruluk tablosuna sahiptir.

A	B	T
0	0	0
0	1	1
1	0	1
1	1	1



- OR işlemi binary sayıda bir bit değerini 0 yapmak için kullanılır. (masking)

$$\begin{array}{r}
 \text{x x x x x x x x} \quad \text{Unknown number} \\
 + 00001111 \quad \text{Mask} \\
 \hline
 \text{x x x x 1111} \quad \text{Result}
 \end{array}$$

```

0000 B0 05      MOV AL,5      ;load data
0002 B3 07      MOV BL,7
0004 F6 E3      MUL BL
0006 D4 0A      AAM          ;adjust
0008 0D 3030    OR AX,3030H   ;convert to ASCII
  
```

Temel Mantık Komutları

OR - devam

- OR işlemleri aşağıdaki tabloda görülmektedir. Memory-to-register veya register-to-register veya immediate adresleme yapılabilir.

Assembly Language	Operation
OR AL,BL	AL = AL or BL
OR SI,DX	SI = SI or DX
OR EAX,EBX	EAX = EAX or EBX
OR DH,0A3H	DH = DH or 0A3H
OR SP,990DH	SP = SP or 990DH
OR EBP,10	EBP = EBP or 10
OR DX,[BX]	DX is ORed with the word contents of data segment memory location addressed by BX
OR DATES[DI+2],AL	The byte contents of the data segment memory location addressed by DI plus 2 are ORed with AL

Temel Mantık Komutları

Exclusive-OR (XOR)

- Exclusive-OR işleminde OR işleminden farklı olarak 1,1 girişleri için sonuç 1 değil 0 olur. Memory-to-memory ve segment register adresleme yapılamaz. Aşağıdaki doğruluk tablosuna sahiptir.

A	B	T
0	0	0
0	1	1
1	0	1
1	1	0



- OR işlemi binary sayıda bir bit değerini 0 yapmak için kullanılır. (masking)

(a)	(b)
x x x x x x x x	Unknown number
⊕ 0 0 0 0 1 1 1 1	Mask
x x x x $\bar{x}$ $\bar{x}$ $\bar{x}$ $\bar{x}$	Result

```

0000 81 C9 0600      OR   CX,0600H      ;set bits 9 and 10
0004 83 E1 FC          AND   CX,0FFFCH    ;clear bits 0 and 1
0007 81 F1 1000      XOR   CX,1000H    ;invert bit 12

```

Temel Mantık Komutları

XOR - devam

- XOR işlemleri aşağıdaki tabloda görülmektedir. Memory-to-register veya register-to-register veya immediate adresleme yapılabilir.

Assembly Language	Operation
XOR CH,DL	CH = CH xor DL
XOR SI,BX	SI = SI xor BX
XOR EBX,EDI	EBX = EBX xor EDI
XOR AH,0EEH	AH = AH xor 0EEH
XOR DI,00DDH	DI = DI xor 00DDH
XOR ESI,100	ESI = ESI xor 100
XOR DX,[SI]	DX is Exclusive-ORed with the word contents of the data segment memory location addressed by SI
XOR DEAL[BP+2],AH	AH is Exclusive-ORed with the byte contents of the stack segment memory location addressed by BP plus 2

Temel Mantık Komutları

Test ve Bit Test Komutları

- Test komutu AND işlemi yapar. TEST komutu hedef operandı değiştirmez, AND değiştirir.
- TEST işlemi FLAG bitlerini değiştirir ve CMP komutuna benzer şekilde çalışır. CMP bir byte veya word üzerinde işlem yapar TEST bit üzerinde işlem yapar.
- Eğer test edilen bit 0 ise Z = 1 ve 1 ise Z = 0 olur. (Z -> Zero Flag)

Assembly Language	Operation
TEST DL,DH	DL is ANDed with DH
TEST CX,BX	CX is ANDed with BX
TEST EDX,ECX	EDX is ANDed with ECX
TEST AH,4	AH is ANDed with 4
TEST EAX,256	EAX is ANDed with 256

Temel Mantık Komutları

Test ve Bit Test Komutları - devam

- Test komutundan hemen sonra genellikle JZ (Jump Zero) veya JNZ (Jump Not Zero) komutları kullanılır.
- Hedef operand bir immediate değerle test edilebilir. Immediate değer 1 ise en sağdaki bit, 2 ise sağdan ikinci bit test edilir.
- Aşağıdaki örnekte önce AL nin en sağdaki biti ardından en soldaki biti test edilir. Sonuca göre RIGHT veya LEFT etiketlerinin olduğu kısma atlama yapılır.

```
TEST AL,1          ;test right bit
JNZ RIGHT          ;if set
TEST AL,128        ;test left bit
JNZ LEFT           ;if set
```

Temel Mantık Komutları

Test ve Bit Test Komutları - devam

- 80386 ve üstü işlemciler bir bit testi için kullanılan komutlara sahiptir.
- Test işleminin sonucu Carry Flag bitine aktarılır. BT AX,4 komutu AX register'ının 4.bitini test eder. 1 ise C=1, 0 ise C=0 yapılır.
- Diğer üç komut test işleminden hemen sonra değer değiştirir.

Assembly Language	Operation
BT	Tests a bit in the destination operand specified by the source operand
BTC	Tests and complements a bit in the destination operand specified by the source operand
BTR	Tests and resets a bit in the destination operand specified by the source operand
BTS	Tests and sets a bit in the destination operand specified by the source operand

Temel Mantık Komutları

Test ve Bit Test Komutları - devam

- 80386 ve üstü işlemciler bir bit testi için kullanılan komutlara sahiptir.
- Test işleminin sonucu Carry Flag bitine aktarılır. BT AX,4 komutu AX register'ının 4.bitini test eder. 1 ise C=1, 0 ise C=0 yapılır.
- Diğer üç komut test işleminden hemen sonra değer değiştirir.

```

BTS  CX,9           ;set bit 9
BTS  CX,10          ;set bit 10
BTR  CX,0           ;clear bit 0
BTR  CX,1           ;clear bit 1
BTC  CX,12          ;complement bit 12

```

Temel Mantık Komutları

NOT ve NEG komutları

- NOT mantıksal tersleme veya 1 tümleyen (**one's complement**), NEG aritmetik işaret tersleme veya 2 tümleyen (**two's complement**) işlemi yapar.
- NOT tüm bitleri tersler, NEG işareti tersler.

Assembly Language	Operation
NOT CH	CH is one's complemented
NEG CH	CH is two's complemented
NEG AX	AX is two's complemented
NOT EBX	EBX is one's complemented
NEG ECX	ECX is two's complemented
NOT TEMP	The contents of data segment memory location TEMP are one's complemented
NOT BYTE PTR[BX]	The byte contents of the data segment memory location addressed by BX are one's complemented

SHIFT ve ROTATE Komutları

SHIFT

- SHIFT komutu bir register veya hafıza içeriğini sağa veya sola kaydırma işlemi yapar. 2 nin katları şeklinde çarpma (sola kaydırma) veya bölme (sağa kaydırma) yapar.
- Mantıksal kaydırmada taşan biti atılır. İşaretsiz sayılarda kullanılır.
- Aritmetik kaydırmada değerin işaret biti sabit tutulur. İşaretlili sayılarda kullanılır.

Assembly Language	Operation
SHL AX,1	AX is logically shifted left 1 place
SHR BX,12	BX is logically shifted right 12 places
SHR ECX,10	ECX is logically shifted right 10 places
SAL DATA1,CL	The contents of data segment memory location DATA1 are arithmetically shifted left the number of spaces specified by CL
SAR SI,2	SI is arithmetically shifted right 2 places
SAR EDX,14	EDX is arithmetically shifted right 14 places

SHIFT ve ROTATE Komutları

SHIFT

- SHL ve SHR mantıksal kaydırma yapar.
- SAL ve SAR aritmetik kaydırma yapar.

Örnek

```

0000 C1 E2 0E      SHL  DX,14
                    or
0003 B1 0E          MOV  CL,14
0005 D3 E2          SHL  DX,CL

```

Target register or memory

SHL: Left shift, carry flag (C) is 0.

SAL: Arithmetic left shift, carry flag (C) is 0.

SHR: Right shift, carry flag (C) is 0.

SAR: Arithmetic right shift, sign bit is replicated.

SHIFT ve ROTATE Komutları

SHIFT - örnek

- İlk işlemcilerde kaydırma ile yapılan çarpma MUL komutundan daha hızlı çalışır.

```

;Multiply AX by 10 (1010)
;
    SHL  AX,1          ;AX times 2
    MOV  BX,AX
    SHL  AX,2          ;AX times 8
    ADD  AX,BX         ;AX times 10
;
;Multiply AX by 18 (10010)
;
    SHL  AX,1          ;AX times 2
    MOV  BX,AX
    SHL  AX,3          ;AX times 16
    ADD  AX,BX         ;AX times 18
;
;Multiply AX by 5 (101)
;
    MOV  BX,AX
    SHL  AX,2          ;AX times 4
    ADD  AX,BX         ;AX times 5

```

SHIFT ve ROTATE Komutları

ROTATE

- ROTATE komutları register veya hafızadaki bir alanın carry flag kullanılarak kaydırılması için kullanılır.
- Büyük boyuttaki sayıların kaydırılması için kullanılır.

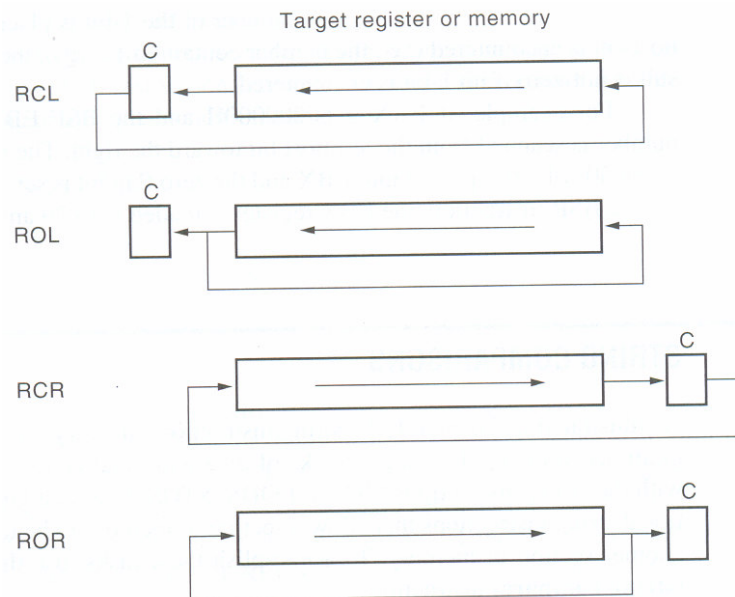
Assembly Language	Operation
ROL SI,14	SI rotates left 14 places
RCL BL,6	BL rotates left through carry 6 places
ROL ECX,18	ECX rotates left 18 places
RCR AH,CL	AH rotates right through carry the number of places specified by CL
ROR WORD PTR[BP],2	The word contents of the stack segment memory location addressed by BP rotate right 2 places

- Aşağıdaki örnekte 48-bit DX,BX,AX sayısı sola kaydırılmıştır.

0000 D1 E0	SHL AX, 1
0002 D1 D3	RCL BX, 1
0004 D1 D2	RCL DX, 1

SHIFT ve ROTATE Komutları

ROTATE



String Karşılaştırma Komutları

- Blok data işlemleri MOVS, LODS, STOS, INS ve OUTS komutlarıyla yapılmaktadır.
- Hafızadaki bir bloğun sabit bir değerle veya başka bir hafıza bloğuyla karşılaştırılması için SCAS (Scan String) ve CMPS (Compare String) komutları kullanılır.

SCAS

- SCAS komutu AL register'ını hafızadaki bir byte blokla, AX register'ını bir word blokla veya EAX register'ını bir doubleword blokla karşılaştırır.
- SCAS komutu hafızadaki bloğu AL, AX veya EAX register'larından çıkartır.
- Byte karşılaştırması için SCASB, word için SCASW ve doubleword için SCASD komutu kullanılır.
- Hafızada karşılaştırılan DI offset adresine sahip blok extra segment içerisindedir.

String Karşılaştırma Komutları

SCAS - devam

- BLOCK isimli 100 byte'lık bir hafıza bloğu olsun. Bu blok içerisinde 00H olup olmadığı aşağıdaki örnekte test edilmektedir.
- REPNE (Repeat While Not Equal) ve REPE (Repeat While Equal) deyimleri CX=0 oluncaya kadar veya karşılaştırma sonucu eşit oluncaya (REPNE)/eşit olmayıncaya (REPE) kadar döngüyü tekrarlar.

```
MOV DI, OFFSET BLOCK ;address data
CLD ;auto-increment
MOV CX, 100 ;load counter
XOR AL, AL ;clear AL
REPNE SCASB
```

```
CLD ;auto-increment
MOV CX, 256 ;load counter
MOV AL, 20H ;get space
REPE SCASB
```

String Karşılaştırma Komutları

CMPS

- CMPS komutu hafızaki iki bloğu byte (CMP SB), word (CMP SW) veya doubleword (CMP SD) olarak karşılaştırır.
- Data segment içerisinde SI ile belirtilen alan extra segment içerisinde DI ile belirtilen alanla karşılaştırılır.
- CMPS, DI ve SI register'larını otomatik artırır veya azaltır.
- REPE, REPNE, REPZ (Repeat While Zero), REPNZ (Repeat While Not Zero) ön belirteçlerle(prefix) kullanılır.
- Aşağıdaki örnekte 10 byte'lık bir blok karşılaştırılır. CX>0 olduğu sürece veya eşitlik olduğu sürece döngü devam eder.

```
MOV SI,OFFSET LINE      ;address LINE
MOV DI,OFFSET TABLE    ;address TABLE
CLD                     ;auto-increment
MOV CX,10                ;load counter
REPE CMPSB               ;search
```

Ödev

- 100 elemanlı bir dizinin her bir elemanının yerine kendisi ve kendisinden önceki sayıların toplamını hesaplayıp kaydeden bir program yazınız.
- Sayıların içinde en büyük sayı BX register'ına ve en küçük sayı DX register'ına kaydedilecektir.
- Program sayıların aritmetik ortalamasını hesaplayıp AX register'ına saklayacaktır.