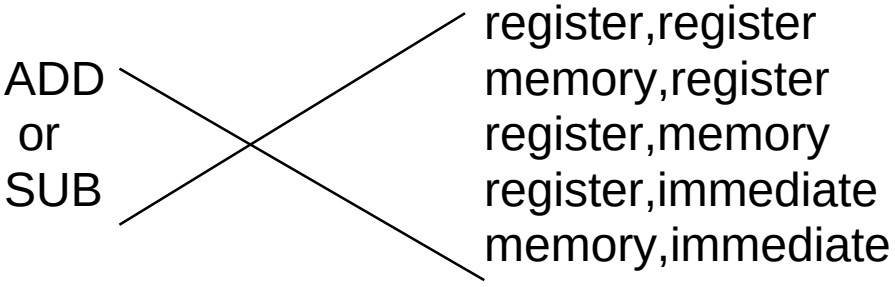
Lecture 13

Data Manipulation: Binary Data

Text:
4th Edition: Chapter 13
5th Edition: Chapter 12

Processing Binary Numbers

Addition and Subtraction



Examples: (All variables are words)

Java	Assembly

`I=J+K;`	`MOV AX,J`
	`ADD AX,K`
	`MOV I,AX`

`P=A+B-C;`	`MOV AX,A`
	`ADD AX,B`
	`SUB AX,C`
	`MOV P,AX`

`M=(N+P)-(R+S)`	`MOV AX,N`
	`ADD AX,P`
	`MOV TEMP1,AX`
	`MOV AX,R`
	`ADD AX,S`
	`MOV TEMP2,AX`
	`MOV AX,TEMP1`
	`SUB AX,TEMP2`
	`MOV M,AX`

Addition of Doublewords on a 16-bit machine

```
      0043 89C3
+     0008 C42E
-------------------
      004C 4DF1
```

32-bit addition

Definition of doublewords:

```
FIRSTH   DW   0043h
FIRSTL   DW   89C3h

SECNDH   DW   0008h
SECNDL   DW   C42Eh

THIRDH   DW   ?
THIRDL   DW   ?
```

4300 C389

0800 2EC4

???? ????

Code for addition:

```
    MOV   AX,FIRSTL
    ADD   AX,SECNDL
    (the carry flag is set to 1)
    MOV   THIRDL,AX
```

AX
89C3
C42E
4DF1

???? F14D

carry → 1

```
    MOV   AX,FIRSTH
    ADC   AX,SECNDH
    MOV   THIRDH,AX
```

AX
0043
0008
004C

4C00 F14D

MULTIPLICATION

MUL	Unsigned data
IMUL	Signed data

Byte times Byte

- The multiplicand is in the AL register
- The multiplier is in a byte (memory or register)
- The product is a WORD in the AX register

Word times Word

- The multiplicand is in the AX register
- The multiplier is in a word (memory or register)
- The product is a DOUBLEWORD
  high order bits in the DX register
  low order bits in the AX register

Doubleword times Doubleword

- The multiplicand is in the EAX register
- The multiplier a doubleword (memory or register)
- The product is a QUADWORD
  high order bits in the EDX register
  low order bits in the EAX register

Sign of Result	Second Operand	
First Operand	+	-
+	+	-
-	-	+

EXAMPLES:

Original values:

DX	AX	CX	“CAT”	“DOG”
0255	0054	0003	FF04	0021

After the independent instructions:

	DX	AX	CX	“CAT”	“DOG”
MUL CL	0255	**00FC**	0003	FF04	0021
*AL is multiplied by CL. The result is a word in AX. 54h*03h=00FCh*					
MUL DOG	**0000**	**0AD4**	0003	FF04	0021
*AX is multiplied by 0021h. The result is a doubleword in DX:AX. 0054h*0021h=00000AD4*					
MUL CAT	**0053**	**AD50**	0003	FF04	0021
*AX is multiplied by the **unsigned** value FF04h. 0054h*FF04h=84*65,284=5,483,856=0053AD50h*					
IMUL CAT	**FFFF**	**AD50**	0003	FF04	0021
*AX is multiplied by the **signed** FF04h. 0054h*FF04h=84*-252=-21,168=FFFFAD50h*					

DIVISION

DIV	Unsigned data
IDIV	Signed data

Word ÷ Byte

- The dividend is in the AX register
- The divisor is a byte (memory or register)
- The quotient is placed in AL
- The remainder is placed in AH

Doubleword ÷ Word

- The dividend is in the DX:AX register pair
- The divisor is a word (memory or register)
- The quotient is placed in AX
- The remainder is placed in DX

Quadword ÷ Doubleword

- The dividend is in the EDX:EAX register pair
- The divisor is a doubleword (memory or register)
- The quotient is placed in EAX
- The remainder is placed in EDX

Sign of Quot. ; rem.	Second Operand	
First Operand	+	-
+	+;+	-;+
-	-;-	+;-

Original values:

DX	AX	CX	“CAT”	“DOG”
0000	0056	0003	FFF4	0021

After the independent instructions:

	DX	AX	CX	“CAT”	“DOG”
DIV CL	0000	**021C**	0003	FFF4	0021
AX is Divided by CL. The quotient goes in AL, the remainder in AH. *0056h÷03h=86÷3=28 r. 2= 1Ch r. 02h*					
DIV DOG	**0014**	**0002**	0003	FFF4	0021
DX:AX is Divided by 0021h. The quotient goes in AX, the remainder in DX. *0000:0056h÷0021h=86÷33=2 r. 20=0002h r. 0014*					
DIV CAT	**0056**	**0000**	0003	FFF4	0021
*DX:AX is Divided by the **unsigned** value FF04h.* *0000:0056h÷FFF4h=* *86÷65,524=0 r. 86=0000h r.0056h*					
IDIV CAT	**0002**	**FFF9**	0003	FFF4	0021
*DX:AX is Divided by the **signed** FFF4h.* *0000:0056h÷FFF4h=* *86÷-12=-7 r. 2 =FFF9h r. 0002h*					

A negative dividend in a register pair

Suppose you want to do the calculation (in decimal):

-86÷12 = -7 r. -2

The number -86 must be a doubleword in the DX:AX register pair.

Clearly, AX should contain FFAAh, which is the hex value for -86.

DX	AX
	FFAA

DX, however, must not contain leading zeros as before (when the value in AX was positive):

This makes the value in the register pair 0000FFAA, which is positive!

DX	AX
0000	FFAA

The DX register needs to be filled with leading 1's (sign bits):

DX	AX
FFFF	FFAA

CBW (Convert Byte to Word)

Extend the sign bit in the AL register through the AH register.

CWD (Convert Word to Doubleword)

Extend the sign bit in the AX register through the DX register.

OVERFLOW and Division

It is possible for the quotient to be too large to be placed in the receiving location.

Example:

```
ONE   DW     001h

      MOV    DX,0043h
      MOV    AX,1544h
      DIV    ONE
```

0043	1544	÷	0001

00431544 ÷ 0001 = 431544 R. 0

The quotient is too large to be placed in AX!

Rule:

> The divisor must be greater than the left half of the dividend.

0021 4C62 ÷ 0054 = **657B** r. 0006

035B ÷ **04** = D6 R. 3

0092 300A ÷ **0091** = 10218 r. 0072

Exercises - Lecture 13

1. Fill in the results of each instruction in the table below. Do each one independently, using the original values for each calculation.

```
MinusThree      DW      -3
Seven           DW      7
Two             DB      2
```

	DX	AX	BX
	0000	**0025**	**0008**
`mul bx`			
`div bx`			
`mul two`			
`mul MinusThree`			
`imul two`			
`imul MinusThree`			
`div Two`			
`div Seven`			
`div bx`			
`div bl`			
`idiv MinusThree`			

2. Which register values are illegal for `div BX`

DX	AX	BX	legal	illegal
0000	0004	0003		
0000	FFF6	0002		
0042	8AC3	009A		
0042	8AC3	0004		
FFFF	FFF2	0002		