Multiplication and Division Instructions

Outline

- MUL and IMUL
- Simple application of MUL and IMUL
- DIV and IDIV
- Sign extension of the dividend
- Decimal input and output procedures

Signed and Unsigned Multiplication

- In binary multiplication, signed and unsigned numbers must be treated differently.
- For example, we want to multiply the eight-bit numbers 10000000 and 11111111.
- Interpreted as unsigned numbers, they represent 128 and 255 respectively. The product is 32640 = 0111111110000000b.
- Interpreted as signed numbers, they represent -128 and -1 respectively and the product is 128 = 0000000010000000b.
- Because signed and unsigned multiplication lead to different results there are two multiplication instructions: MUL and IMUL.
- For multiplication of positive numbers MUL and IMUL give the same result.

MUL Instructions

- MUL (multiply) is used for unsigned multiplication.
- Syntax:

  MUL source

- This instruction multiply bytes or words.
- For byte multiplication, one number is contained in the source and the other is assumed to be in AL. The 16-bit product will be in AX. The source may be a byte register or memory byte, but not a constant.

MUL Instructions

- For word multiplication, one number is contained in the source and the other is assumed to be in AX. The most significant 16-bits of the double word product will be in DX, and the least significant 16 bits will be in AX (DX:AX). The source may be a 16-bit register or memory word, but not a constant.
- Effect on status flags:
  - SF, ZF, PF and AF undefined.
  - CF/OF is 0 if the upper half of the result is zero otherwise 1.

IMUL Instructions

- IMUL (integer multiply) is used for signed multiplication.
- Syntax:

  IMUL source

- This instruction multiply bytes or words.
- For byte multiplication, one number is contained in the source and the other is assumed to be in AL. The 16-bit product will be in AX. The source may be a byte register or memory byte, but not a constant.

IMUL Instructions

- For word multiplication, one number is contained in the source and the other is assumed to be in AX. The most significant 16-bits of the double word product will be in DX, and the least significant 16 bits will be in AX (DX:AX). The source may be a 16-bit register or memory word, but not a constant.
- Effect on status flags:
  - SF, ZF, PF and AF undefined.
  - CF/OF is 0 if the upper half of the result is the sign extension of the lower half otherwise 1.

Example

- Suppose AX contains 1 and BX contains FFFFh.

Instruction	Decimal Product	Hex Product	DX	AX	CF/OF
MUL BX	65535	0000FFFF	0000	FFFF	0
IMUL BX	-1	FFFFFFFF	FFFF	FFFF	0

Simple Application of MUL and IMUL

- Translate the high-level language assignment statement A= 5 x A - 12 x B into assembly code. Let A and B be word variables, and suppose there is no overflow. Use IMUL for multiplication.
  - Solution:

```
MOV   AX, 5
IMUL  A
MOV   A, AX
MOV   AX, 12
IMUL  B
SUB   A, AX
```

DIV Instruction

- DIV (divide) is used for unsigned division.
- Syntax:

  DIV divisor

- This instruction divide 8 (or 16)bits into 16 (or 32) bits.
- The quotient and remainder have the same size as the divisor.
- In byte form, the divisor is an 8-bit register or memory byte. The 16-bit dividend is assumed to be in AX. After division, the 8-bit quotient is in AL and the 8-bit remainder is in AH. The divisor may not be a constant.

DIV Instruction

- In word form divisor is a 16-bit register or memory word. The 32-bit dividend is assumed to be in DX:AX. After division, the 16-bit quotient is in AX and the 16 bit remainder is in DX. The divisor may not be a constant.
- The effect of DIV on the flags is that all status flags are undefined.
- It is possible that the quotient will be too big to fit in the specified destination(AL or AX). This can happen if the divisor is much smaller than the dividend. If this happens, the program terminates and the system displays the message "Divide Overflow".

IDIV Instruction

- IDIV (integer divide) is used for signed division.
- Syntax:

```
IDIV    divisor
```

- These instructions divide 8 (or 16)bits into 16 (or 32) bits.
- The quotient and remainder have the same size as the divisor.
- In byte form, the divisor is an 8-bit register or memory byte. The 16-bit dividend is assumed to be in AX. After division, the 8-bit quotient is in AL and the 8-bit remainder is in AH. The divisor may not be a constant.

IDIV Instruction

- In word form divisor is a 16-bit register or memory word. The 32-bit dividend is assumed to be in DX:AX. After division, the 16-bit quotient is in AX and the 16 bit remainder is in DX. The divisor may not he a constant.
- For signed division, the remainder has the same sign as the dividend.
- The effect of IDIV on the flags is that all status flags are undefined.
- It is possible that the quotient will be too big to fit in the specified destination(AL or AX). This can happen if the divisor is much smaller than the dividend. Where this happens, the program terminates and the system displays the message "Divide Overflow".

Example

- Suppose DX contains 0000h, AX contains 0005h, and BX contains 0002h.

Instruction	Decimal Quotient	Decimal Remainder	AX	DX
DIV BX	2	1	0002	0001
IDIV BX	2	1	0002	0001

Sign Extension of the Dividend

- Word Division
  - The dividend is in DX:AX even if the actual dividend will fit in AX. In this case DX should be prepared as follows:
  1. For DIV, DX should be cleared.
  2. For IDIV, DX should be made the sign extension of AX. The instruction CWD (convert word to double word) will do the extension.
- Example: Divide -1250 by 7
- Solution:

```
MOV    AX, -1250
CWD
MOV    BX, 7
IDIV   BX
```

Sign Extension of the Dividend

- Byte Division
  - The dividend is in AX. If the actual dividend is a byte then AH should be prepared as follows:
  1. For DIV, AH should be cleared.
  2. For IDIV, AH should the sign extension of AL. The instruction CBW (convert byte to word) will do the extension.
- Example : Divide the signed value of the byte variable: XBYTE by -7.
- Solution:

```
MOV    AL, XBYTE
CBW
MOV    BL, -7
IDIV BL
```