Multiplication Instructions

Multiplication Instruction Mnemonics

- `mul` for unsigned multiplication
  - Operands treated as unsigned numbers
- `imul` for signed multiplication
  - Operands treated as signed numbers
  - Sign of result can be positive or negative depending on the signs of the operands (standard rule)

Reading Flags in Visual Studio

Flag	Set value
Overflow	`OV = 1`
Direction	`UP = 1`
Interrupt	`EI = 1`
Sign	`PL = 1`
Zero	`ZR = 1`
Auxiliary carry	`AC = 1`
Parity	`PE = 1`
Carry	`CY = 1`

`mul` Instruction Format

- `mul` *source*
- Single operand *source*
- byte, word, doubleword, or quadword
- Source can be register or memory but NOT immediate
- Specifies one factor
- The other factor is implied (the accumulator)
  - AL for byte-size *source*
  - AX for word *source*
  - EAX for doubleword *source*
  - RAX for quadword source

AH,AL,BH,BL,CH,CL..... --->Bytesize registe

AX,BX,CX,DX..... --->16 bits----> 2bytes word size registers

EAX,EBX,ECX.....--->32 bits--->4bytes--->double word size

RAX,RBX,RCX,..... -->64BIT-----
>8bytes---->quad word

AH, AL, AX,EAX,RAX----->Accumulator registers

mul CH --------->CH*AL
mul BX ------->BX*AX
mul ECX -------> ecx*eax
mul RBX --------> RBX*RAX

mul eax
mul var
~~mul 10~~

`mul` Instruction Operation

- When a byte source is multiplied by the value in AL, the product is put in AX.
- When a word source is multiplied by the value in AX, the product is put in DX:AX.
  - The high-order 16 bits in DX and the low-order 16 bits in AX.
- When a doubleword source is multiplied by the value in EAX, the product is put in EDX:EAX.
- Product of two quadwords in RDX:RAX

let us take two hexa decimal values each of byte size

number1= E7
number2= B7

multiply the number1 and number2

E7 * B7 = A521

2 hex-decimal 1-byte

4 hex decimal --2bytes---->word size

byte*byte----> result may be upto word length--->AH,AL
word*word---> result may be up to double word -->DX,AX

FEB7 * 00A4------------->A32D3C
DX--->00A3 AX---->2D3C

`mul` Instruction Operation

Operand length	Other factor	Result Length	Result location
byte	AL	word	AX (AH:AL)
word	AX	double word	DX:AX
double word	EAX	quad word	EDX:EAX
quad word	RAX	double quad word	RDX:RAX

mul bh------>bh*AL---->AH:AL
mul bx ------>bx*AX--->DX:AX
mul ebx------>ebx*eax--->edx:eax
mul rbx------->rbx*rax---->Rdx:rax

AH,DX,EDX,RDX----> HIGH ORDER HALF
AL,AX,EAX AND RAX--->LOW ORDER HALF

Double-Length Product

- The "double-length" product ensures that the result will always fit in the destination location.
- If significant bits of the product actually "spill over" into the high-order half (AH, DX, or EDX), then CF and OF are both
  set to 1.
- If the high-order half is not significant, then CF and OF are both cleared to 0.
  - For unsigned multiplication, this is when the high-order half is all o's.

word*word--->result stored in dx and ax

bx*ax
00A7 *00B1
=7377 ---------->word size

result stored in ax itself and dx is not used ----->AX=7377
CF=0 and OF=0

---

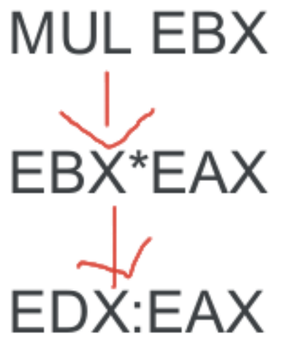
bx*ax
FFA1 * AB71= AB31 6111

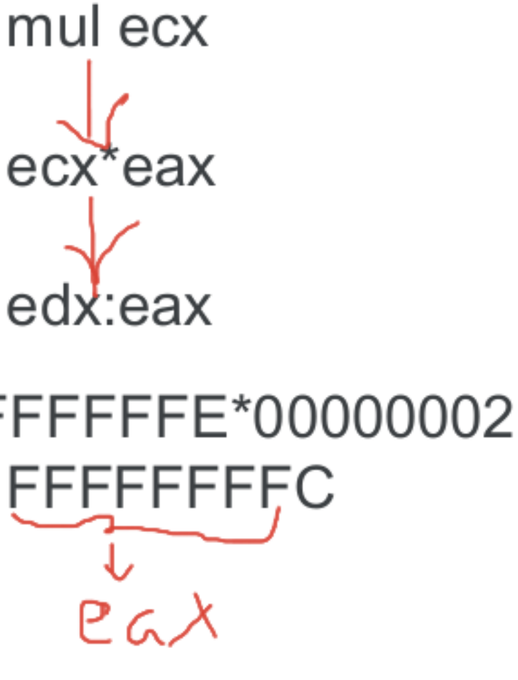
result is more than word size , we need to use DX and AX to store result

DX=AB31, AX=6111
CF=1,OF=1

`mul` Instruction Example

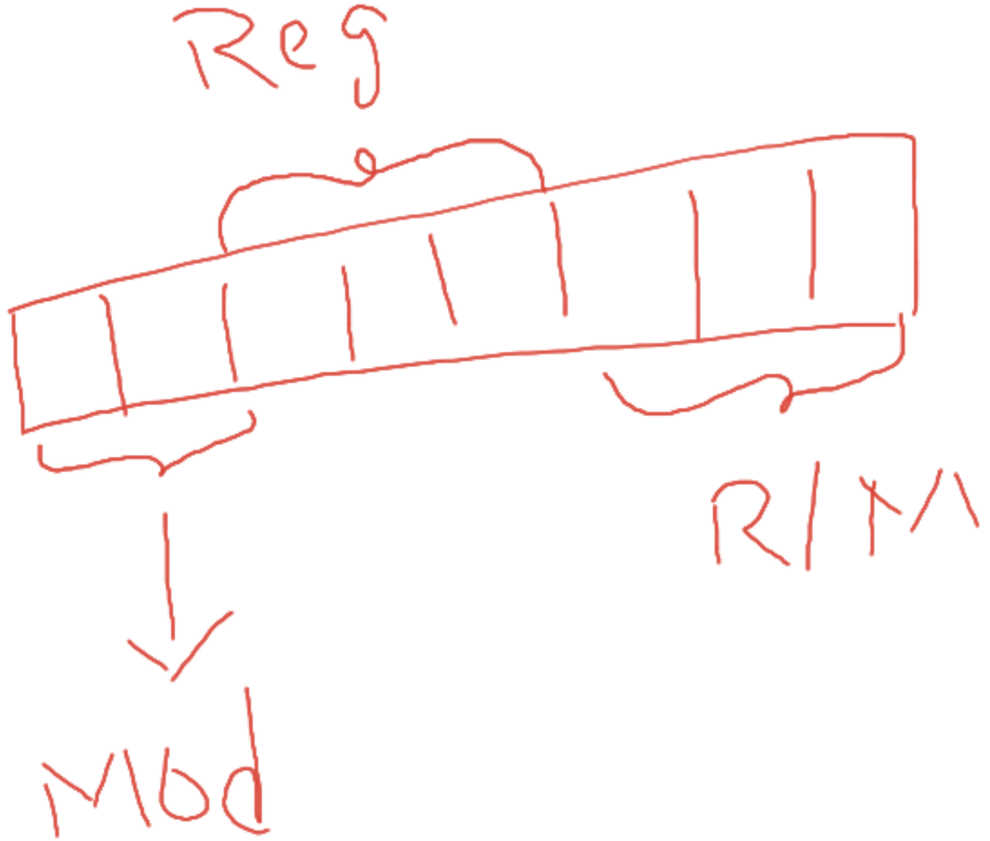
Example: 5 * 2 = 10	Example: 2 * -2 = -4
Before: EAX: 00000005 EBX: 00000002 EDX: ???????? *Instruction:* `mul ebx` *After:* EAX: 0000000A EBX: 00000002 EDX: 00000000 CF=OF=0	*Before:* EAX: 00000002 ECX: FFFFFFFE EDX: ???????? *Instruction:* `mul ecx` *After:* EAX: FFFFFFFC ECX: FFFFFFFE EDX: 00000001 CF=OF=1





ModR/M->Reg Field spec

	000	001	010	011	100	101	110	111
80,81	ADD	OR	ADC	SBB	AND	SUB	XOR	CMP
D0,D1	ROL	ROR	RCL	RCR	SHL	SHR		SAR
F6,F7	TEST		NOT	NEG	MUL	IMUL	DIV	IDIV
FE,FF	INC	DEC					PUSH	



`mul` Instruction encoding (1 op)

Operand	Opcode	Obj Code: Byte Length
register 8	F6	2
register 16	F7	3
register 32	F7	2
register 64	F7	3
Memory byte	F6	2+
Memory word	F7	3+
Memory doubleword	F7	2+
Memory quadword	F7	3+

- Applied Prefix: 66 (word) and 4x (for 64-bit specific).

`mul` Instruction encoding (1 op)

```
00000000  F6 E0          mul AL
00000002  F6 E7          mul BH

00000004  66| F7 E0         mul AX
00000007  66| F7 E1         mul CX

0000000A  F7 E0          mul EAX
0000000C  F7 E2          mul EDX

0000000E  48/ F7 E0          mul RAX
00000011  48/ F7 E3          mul RBX
```

```
00000014  48/ 8D 1D          lea RBX, bytenum
      000001CE R
0000001B  48/ 8D 0D          lea RCX, wnum
      000001CF R
00000022  48/ 8D 35          lea RSI, dwnum
      000001D1 R
00000029  48/ 8D 3D          lea RDI, qwnum
      000001D5 R

00000030  F6 25 000001CE R mul bytenum
00000036  F6 23          mul BYTE PTR [RBX]
00000038  66| F7 25          mul wnum
      000001CF R
0000003F  66| F7 21          mul WORD PTR [RCX]
00000042  F7 25 000001D1 R mul dwnum
00000048  F7 26          mul DWORD PTR [RSI]
0000004A  48/ F7 25          mul qwnum
      000001D5 R
00000051  48/ F7 27          mul QWORD PTR [RDI]
```

`imul` Instruction Formats

- `imul` *`source`*
- `imul` *`register, source`*
- *`imul register, source, immediate`*

`imul` *source*

- "Single-operand format"
- Similar to `mul` *source* except for signed operands
- CF=OF=0 if each bit in the high-order half is the same as the sign bit in the low-order half.
- CF=OF=1 otherwise (the bits in the high-order half are significant)

`imul` Instruction encoding (1 op)

Operand	Opcode	Obj Code: Byte Length
register 8	F6	2
register 16	F7	2
register 32	F7	2
Memory byte	F6	2+
Memory word	F7	2+
Memory doubleword	F7	2+
register 64	F7	2
Memory quadword	F7	2+

- Applied Prefix: 66 (word) and 4x (for 64-bit specific).
- Same as *mul* instruction encoding
- ModR/M: reg is 100 for `mul` and 101 for `imul`

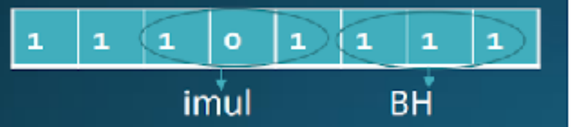
`imul` Instruction encoding (1 op)

```
00000054  F6 E8          imul AL
00000056  F6 EF          imul BH

00000058  66| F7 E8         imul AX
0000005B  66| F7 E9         imul CX

0000005E  F7 E8          imul EAX
00000060  F7 EA          imul EDX

00000062  48/ F7 E8         imul RAX
00000065  48/ F7 EB         imul RBX
```




```
00000068  48/ 8D 1D          lea RBX, bytenum
      000001CE R
0000006F  48/ 8D 0D          lea RCX, wnum
      000001CF R
00000076  48/ 8D 35          lea RSI, dwnum
      000001D1 R
0000007D  48/ 8D 3D          lea RDI, qwnum
      000001D5 R

00000084  F6 2D 000001CE R imul bytenum
0000008A  F6 2B           imul BYTE PTR [RBX]
0000008C  66| F7 2D          imul wnum
      000001CF R
00000093  66| F7 29          imul WORD PTR [RCX]
00000096  F7 2D 000001D1 R imul dwnum
0000009C  F7 2E           imul DWORD PTR [RSI]
0000009E  48/ F7 2D          imul qwnum
      000001D5 R
000000A5  48/ F7 2F          imul QWORD PTR [RDI]
```

- Applied Prefix: 66 (word) and 4x (for 64-bit specific).
- Same as *mul* instruction encoding
- ModR/M: reg is 100 for `mul` and 101 for `imul`

`imul register,source`

- "Two-operand format"
- Source operand can be in a register, in memory, or immediate.
- Register contains other factor, and also specifies the destination
- Both operands must be word-size or doubleword-size, quadword-size NOT byte-size.
- Product must "fit" in destination register.
  - CF and OF are cleared to 0 if result fits.
  - CF and OF are set to 1 if it doesn't fit.

Two-operand Example

- Example: 10 * 10 = 100
- *Before*
  EBX: 0000000A
- *Instruction*
  `imul ebx, 10`
- *After*
  EBX: 00000064
  CF=OF=0

`imul` Instruction encoding (2 op)

Destination	Source	Opcode	Obj Code: Byte Length
register 16	register 16	0F AF	4
register 32	register 32	0F AF	3
register 16	Memory word	0F AF	4+
register 32	Memory doubleword	0F AF	3+
register 16	Immediate byte	6B	4
register 16	Immediate word	69	5
register 32	Immediate byte	6B	3
register 32	Immediate doubleword	69	6
register 64	register 64	0F AF	4
register 64	Memory quadword	0F AF	4+
register 64	Immediate byte	6B	4
register 64	Immediate doubleword	69	7

`imul` Instruction encoding

- Applied Prefix: 66 (word) and 4x (for 64-bit specific).
- Immediate operand is sign extended before multiplication

```
000000A8  66| 0F AF C1      imul AX, CX
000000AC  0F AF D9      imul EBX, ECX
000000AF  66| 0F AF 0D      imul CX, wnum
      000001CF R
000000B7  0F AF 15      imul EDX, dwnum
      000001D1 R
000000BE  66| 6B D2 36      imul DX, 54
000000C2  66| 69 DB 0806    imul BX, 2054
000000C7  6B D2 36      imul EDX, 54
000000CA  69 C9 009C9626    imul ECX, 10262054
```

```
000000D0  48/ 0F AF C1      imul RAX, RCX
000000D4  48/ 0F AF 1D      imul RBX, qwnum
      000001D5 R
000000DC  48/ 6B D2 36      imul RDX, 54
000000E0  48/ 69 D2         imul RDX, 2054
      00000806
000000E7  48/ 69 D2         imul RDX, 10262054
      009C9626


000000EE  48/ 8D 3D         lea RDI, qwnum
      000001D5 R
000000F5  48/ 0F AF 1F      imul RBX, QWORD PTR [RDI]
```

imul register,source,immediate

- "Three-operand format"
- The two factors are given by *source* (register or memory) and the immediate value.
- The first operand, a register, specifies the destination for the product.
- Operands *register* and *source* are the same size, both 16-bit or both 32-bit or 64-bit BUT not 8-bit.
- If the product will fit in the destination register, then CF and OF are cleared to 0; if not, they are set to 1.

imul bx, val , 1000

bx:A007
val:1234

result 2C 72E5 BDE0

bx:BDE0

as we are not able to fit result into register--->cf=1 and of=1

imul Instruction encoding (3 op)

Reg Dest	Source	Imm Operand	Opcode	Obj Code: Byte Length
register 16	register 16	byte	6B	4
register 16	register 16	word	69	5
register 16	memory word	byte	6B	4+
register 16	memory word	word	69	5+
register 32	register 32	byte	6B	3
register 32	register 32	doubleword	69	6
register 32	memory doubleword	byte	6B	3+
register 32	memory doubleword	doubleword	69	6+
register 64	register 64	byte	6B	4
register 64	register 64	doubleword	69	7
register 64	Memory quadword	byte	6B	4+
register 64	Memory quadword	doubleword	69	7+

imul Instruction encoding

- Applied Prefix: 66 (word) and 4x (for 64-bit specific).
- Immediate operand is sign extended before multiplication

```
000000F9  48/ 8D 2D          lea RBP, wnum
          000001CF R
00000100  48/ 8D 35          lea RSI, dwnum
          000001D1 R
00000107  48/ 8D 3D          lea RDI, qwnum
          000001D5 R

0000010E  66| 6B C1 1A       imul AX, CX, 26
00000112  66| 69 C1 0A88     imul AX, CX, 2696
00000117  66| 6B 0D          imul CX, wnum, 26
          000001CF R 1A
0000011F  66| 6B 4D 00       imul CX, WORD PTR [RBP], 26
          1A
00000124  66| 69 0D          imul CX, wnum, 2096
          000001CF R
          0830
0000012D  66| 69 4D 00       imul CX, WORD PTR [RBP], 2096
          0830
```

```
00000133  6B D9 1A       imul EBX, ECX, 26
00000136  69 D9 00000830   imul EBX, ECX, 2096
0000013C  6B 15 000001D1 R imul EDX, dwnum, 26
          1A
00000143  6B 16 1A       imul EDX, DWORD PTR [RSI], 26
00000146  69 15 000001D1 R imul EDX, dwnum, 20987654
          01403F06
00000150  69 16 01403F06   imul EDX, DWORD PTR [RSI], 20987654

00000156  48/ 6B C1 1A     imul RAX, RCX, 26
0000015A  48/ 69 C1        imul RAX, RCX, 2096
          00000830
00000161  48/ 6B 1D        imul RBX, qwnum, 26
          000001D5 R 1A
00000169  48/ 6B 1F 1A     imul RBX, QWORD PTR [RDI], 26
0000016D  48/ 69 1D        imul RBX, qwnum, 2096986
          000001D5 R
          001FFF5A
00000178  48/ 69 1F        imul RBX, QWORD PTR [RDI], 2096986
          001FFF5A
```