مجموعة التعليمات Instruction Set

للـ 8086 أنماط للتعليمات6

1. تعليمات نقل البيانات Data Transfer Instructions
2. التعليمات الحسابية Arithmetic Instructions
3. التعليمات المنطقية Logical Instructions
4. تعليمات معالجة السلاسل String manipulation Instructions
5. تعليمات التحكم بالعملية Process Control Instructions
6. تعليمات التحكم بالنقل Control Transfer Instructions

مجموعة التعليمات Instruction Set

1. تعليمات نقل البيانات

تستخدم تعليمات هذا النمط لنقل البيانات /العناوين من وإلى المسجلات ومن وإلى مواقع الذاكرة ومن وإلى بوابات I/O.
Instructions that are used to transfer data/ address in to registers, memory locations and I/O ports.

المستهدف بشكل عام هو حدين two operands: حدّي المصدر Source والهدف نبفس الحجم same size.

المصدر Source: مسجل أو موقع ذاكرة أو بيانات فورية immediate data.
الهدف Destination: مسجل أو موقع ذاكرة.

بحجم بايت أو كلمة byte or a word.

يمكننا نقل بيانات بحجم 8-bit إلى مسجلات بحجم ومواقع ذاكرة بنفس الحجم والبيانات بحجم 16-bit إلى مسجلات وماقع ذاكرة بنفس الحجم.

تعريف المعطيات يمكن تعريف المعطيات في لغة التجميع

تعرف المعطيات في مقطع data segment
يمكن أن تعرف اما DB byte
أو كلمة من Word 16 bit
من 32 bit double Word

- Fivfor initialized data

```
DB   Define Byte  e define directives      ;allocates 1 byte
DW   Define Word          ;allocates 2 bytes
DD   Define Doubleword    ;allocates 4 bytes
DQ   Define Quadword      ;allocates 8 bytes
DT   Define Ten bytes     ;allocates 10 bytes
```

Examples

```
sorted    DB    'y'
value     DW    25159
Total     DD    542803535
float1    DD    1.234
```

تعريف مصفوفة أحادية البعد من 8 عناصر

- Multiple definitions can be cumbersome to initialize data structures such as arrays

Example

To declare and initialize an integer array of 8 elements

```
marks   DW    0,0,0,0,0,0,0,0
```

- What if we want to declare and initialize to zero an array of 200 elements?
  - There is a better way of doing this than repeating zero 200 times in the above statement
    - Assembler provides a directive to do this (DUP directive)

* Examples
  - Previous marks array

```
marks   DW    0,0,0,0,0,0,0,0
```

can be compactly declared as

```
marks   TIMES   8   DW   0
```

Symbol Table

* Assembler builds a symbol table so we can refer to the allocated storage space by the associated label

Example

```
.DATA
value     DW   0
sum       DD   0
marks     DW   10 DUP (?)
message   DB   'The grade is:',0
char1     DB   ?
```

name	offset
value	0
sum	2
marks	6
message	26
char1	40

- Directives for uninitialized data
- Five reserve directives

```
RESB   Reserve a Byte        ;allocates 1 byte
RESW   Reserve a Word        ;allocates 2 bytes
RESD   Reserve a Doubleword  ;allocates 4 bytes
RESQ   Reserve a Quadword    ;allocates 8 bytes
REST   Reserve a Ten bytes   ;allocates 10 bytes
```

Examples

```
response   resb   1
buffer     resw   100
Total      resd   1
```

- Multiple definitions can be abbreviated

Example

```
message   DB   'B'
          DB   'y'
          DB   'e'
          DB   0DH
          DB   0AH
```

can be written as

```
message   DB
'B','y','e',0DH,0AH
```

- **More compactly as**

```
message   DB 'Bye',0DH,0A
```

The mov instruction

* Five types of operand combinations are allowed:

Instruction type	Example
`mov register,register`	`mov DX,CX`
`mov register,immediate`	`mov BL,100`
`mov register,memory`	`mov EBX,[count]`
`mov memory,register`	`mov [count],ESI`
`mov memory,immediate`	`mov [count],23`

مثال :
حدد الأخطاء في البرنامج التالي

```
.data
bVal   BYTE    100
bVal2  BYTE    ?
wVal   WORD    2
dVal   DWORD   5
.code
    mov ds,45
    mov esi,wVal
    mov eip,dVal
    mov 25,bVal
    mov bVal2,bVal
```

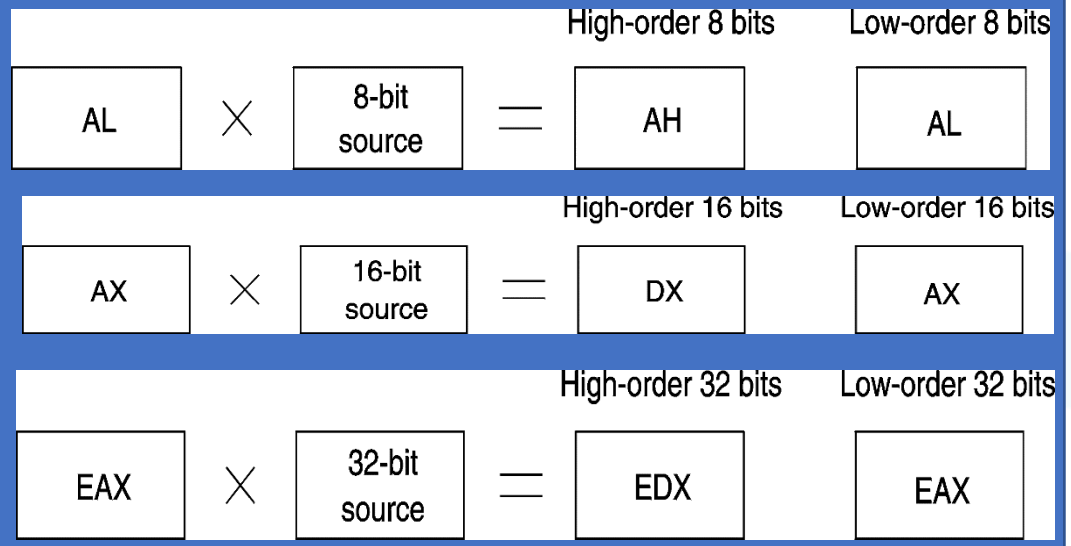
immediate move to DS not permitted
size mismatch
لا يمكن أن يكون المسجل هدف IP or eip
immediate value cannot be destination
memory-to-memory move not permitted

Mnemonics: ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, CMP…

MUL reg/ mem	
MUL reg	For byte : (AX) ← (AL) x (reg8) For word : (DX)(AX) ← (AX) x (reg16)
MUL mem	For byte : (AX) ← (AL) x (mem8) For word : (DX)(AX) ← (AX) x (mem16)
IMUL reg/ mem	
IMUL reg	For byte : (AX) ← (AL) x (reg8) For word : (DX)(AX) ← (AX) x (reg16)
IMUL mem	For byte : (AX) ← (AX) x (mem8) For word : (DX)(AX) ← (AX) x (mem16)

تعليمة ضرب عددين بدون إشارة MUL

mul source

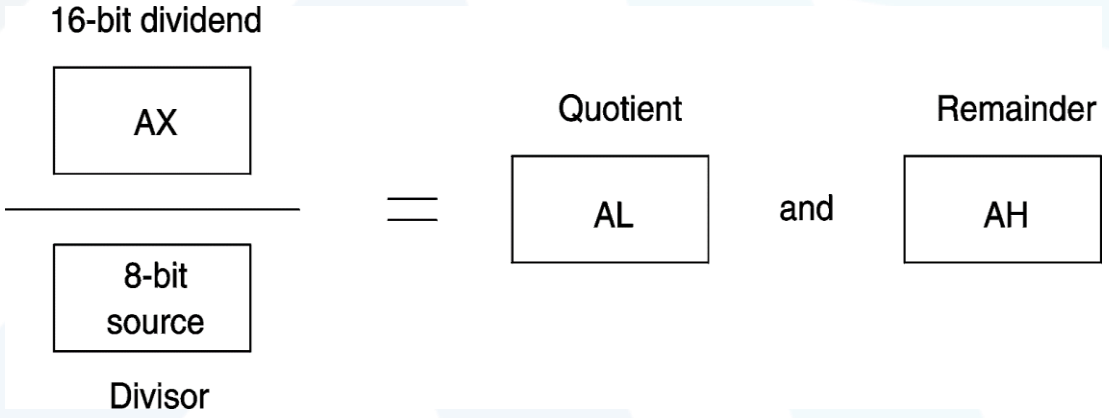


تعليمة ضرب عددين بإشارة IMUL

Mnemonics: ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, CMP…

DIV reg/ mem	
DIV reg	For 16-bit :- 8-bit : (AL) ← (AX) :- (reg8) Quotient (AH) ← (AX) MOD(reg8) Remainder For 32-bit :- 16-bit : (AX) ← (DX)(AX) :- (reg16) Quotient (DX) ← (DX)(AX) MOD(reg16) Remainder
DIV mem	For 16-bit :- 8-bit : (AL) ← (AX) :- (mem8) Quotient (AH) ← (AX) MOD(mem8) Remainder For 32-bit :- 16-bit : (AX) ← (DX)(AX) :- (mem16) Quotient (DX) ← (DX)(AX) MOD(mem16) Remainder

- Dividend is twice the size of the divisor
- Dividend is assumed to be in
  - AX (8-bit divisor)
  - DX:AX (16-bit divisor)
  - EDX:EAX (32-bit divisor)



Mnemonics: ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, CMP…

IDIV reg/ mem	
IDIV reg	For 16-bit :- 8-bit : (AL) ← (AX) :- (reg8) Quotient (AH) ← (AX) MOD(reg8) Remainder For 32-bit :- 16-bit : (AX) ← (DX)(AX) :- (reg16) Quotient (DX) ← (DX)(AX) MOD(reg16) Remainder
IDIV mem	For 16-bit :- 8-bit : (AL) ← (AX) :- (mem8) Quotient (AH) ← (AX) MOD(mem8) Remainder For 32-bit :- 16-bit : (AX) ← (DX)(AX) :- (mem16) Quotient (DX) ← (DX)(AX) MOD(mem16) Remainder

* Some additional related instructions
  - » Sign extension

    `cwde` converts word to doubleword (extends AX into EAX)

- Example

```
mov     AL,-95
cbw                     ; AH = FFH
mov     CL,12
idiv    CL
```

produces –7D in AL and –11D as remainder in AH

- Example

```
mov     AX,-5147
cwd                     ; DX = FFFFH
mov     CX,300
idiv     CX
```

produces –17D in AX and –47D as remainder in DX

Mnemonics: ADD, ADC, SUB, SBB, INC, DEC, MUL, DIV, CMP…

CMP reg2/mem, reg1/ mem	

CMP reg2, reg1	Modify flags ← (reg2) – (reg1)
	
	If (reg2) > (reg1) then CF=0, ZF=0, SF=0
	If (reg2) < (reg1) then CF=1, ZF=0, SF=1
	If (reg2) = (reg1) then CF=0, ZF=1, SF=0

CMP reg2, mem	Modify flags ← (reg2) – (mem)
	
	If (reg2) > (mem) then CF=0, ZF=0, SF=0
	If (reg2) < (mem) then CF=1, ZF=0, SF=1
	If (reg2) = (mem) then CF=0, ZF=1, SF=0

CMP mem, reg1	Modify flags ← (mem) – (reg1)
	
	If (mem) > (reg1) then CF=0, ZF=0, SF=0
	If (mem) < (reg1) then CF=1, ZF=0, SF=1
	If (mem) = (reg1) then CF=0, ZF=1, SF=0

```
mov al,5
cmp al,5                    ; Zero flag set
```

```
mov al,5
cmp al,5                    ; Zero flag set
```

- Example: destination > source

```
mov al,6
cmp al,5                    ; ZF = 0, CF = 0
```

- Example: destination > source

performed with signed integers.

```
mov al,5
cmp al,-2;     Sign flag == Overflow flag
```

CMP reg/mem, data	

CMP reg, data	Modify flags ← (reg) – (data)
	
	If (reg) > data then CF=0, ZF=0, SF=0
	If (reg) < data then CF=1, ZF=0, SF=1
	If (reg) = data then CF=0, ZF=1, SF=0

CMP mem, data	Modify flags ← (mem) – (mem)
	
	If (mem) > data then CF=0, ZF=0, SF=0
	If (mem) < data then CF=1, ZF=0, SF=1
	If (mem) = data then CF=0, ZF=1, SF=0

Logical Instructions التعليمات المنطقية

Mnemonics: AND, OR, XOR, TEST, SHR, SHL, RCR, RCL ...

AND A, data AND AL, data8	(AL) ← (AL) & data8
AND AX, data16	(AX) ← (AX) & data16

AND reg/mem, data AND reg, data	(reg) ← (reg) & data
AND mem, data	(mem) ← (mem) & data

```
mov al,'a' ;            AL = 01100001b
and al,11011111b;       AL = 01000001b
```

Logical Instructions التعليمات المنطقية

Mnemonics: AND, OR, XOR, TEST, SHR, SHL, RCR, RCL ...

OR reg2/mem, reg1/mem	
OR reg2, reg1	(reg2) ← (reg2) | (reg1)
OR reg2, mem	(reg2) ← (reg2) | (mem)
OR mem, reg1	(mem) ← (mem) | (reg1)

```
mov al,6 ;              AL = 00000110b
or  al,00110000b;       AL = 00110110b
```

OR reg/mem, data	
OR reg, data	(reg) ← (reg) | data
OR mem, data	(mem) ← (mem) | data

OR A, data	
OR AL, data8	(AL) ← (AL) | data8
OR AX, data16	(AX) ← (AX) | data16

Logical Instructions التعليمات المنطقية

Mnemonics: AND, OR, XOR, TEST, SHR, SHL, RCR, RCL ...

XOR reg2/mem, reg1/mem	
XOR reg2, reg1	(reg2) ← (reg2) ^ (reg1)
XOR reg2, mem	(reg2) ← (reg2) ^ (mem)
XOR mem, reg1	(mem) ← (mem) ^ (reg1)

XOR reg/mem, data	
XOR reg, data	(reg) ← (reg) ^ data
XOR mem, data	(mem) ← (mem) ^ data

XOR A, data	
XOR AL, data8	(AL) ← (AL) ^ data8
XOR AX, data16	(AX) ← (AX) ^ data16

SHR reg/mem

SHR reg

i) SHR reg, 1

ii) SHR reg, CL

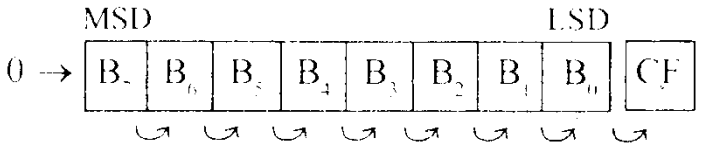
SHR mem

i) SHR mem, 1

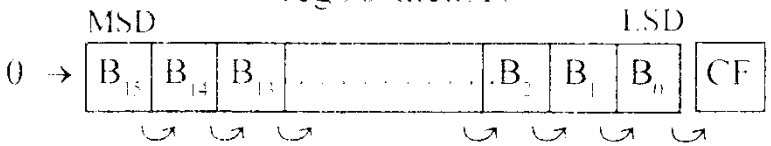
ii) SHR mem, CL

$$CF \leftarrow B_{LSD} \; ; \; B_n \leftarrow B_{n+1} \; ; \; B_{MSD} \leftarrow 0$$

reg 8 / mem 8



reg 16 / mem 16



SHL reg/mem or SAL reg/mem

SHL reg or SAL reg

i) SHL reg, 1 or SAL reg, 1

ii) SHL reg, CL or SAL reg, CL

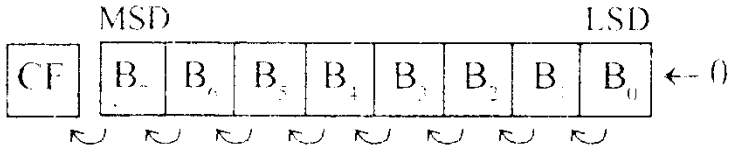
SHL mem or SAL mem

i) SHL mem, 1 or SAL mem, 1

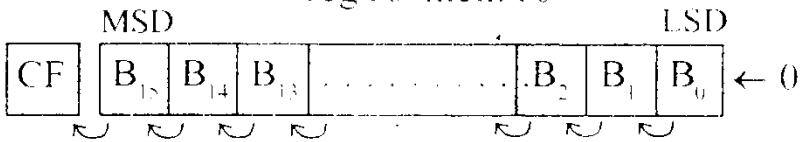
ii) SHL mem, CL or SAL mem, CL

$$CF \leftarrow B_{MSD} \; ; \; B_{n+1} \leftarrow B_n \; ; \; B_{LSD} \leftarrow 0$$

reg 8 / mem 8



reg 16 / mem 16



RCR reg/mem

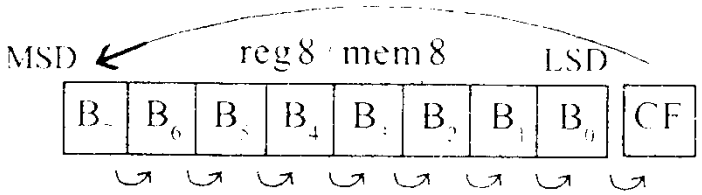
RCR reg

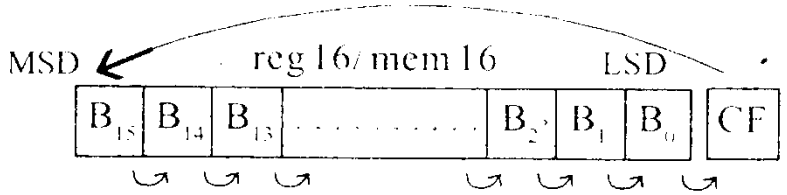
i) RCR reg, 1
ii) RCR reg, CL

RCR mem

i) RCR mem, 1
ii) RCR mem, CL

$B_n \leftarrow B_{n-1}$; $B_{MSD} \leftarrow CF$; $CF \leftarrow B_{LSD}$





ROL reg/mem

ROL reg

i) ROL reg, 1

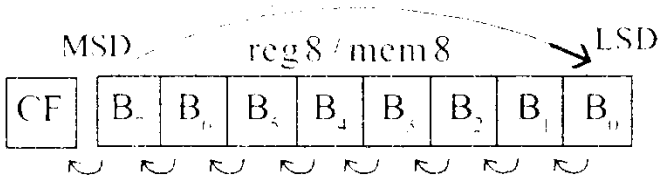
ii) ROL reg, CL

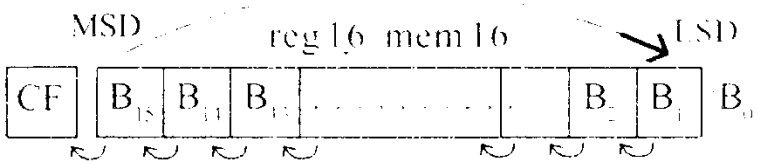
ROL mem

i) ROL mem, 1

ii) ROL mem, CL

$$B_{n+1} \leftarrow B_n \; ; \; CF \leftarrow B_{MSD} \; ; \; B_{LSD} \leftarrow B_{MSD}$$





Instruction Set مجموعة التعليمات

Processor Control Instructions تعليمات التحكم بالعملية

Mnemonics	Explanation
STC	Set CF ← 1
CLC	Clear CF ← 0
CMC	Complement carry CF ← CF/
STD	Set direction flag DF ← 1
CLD	Clear direction flag DF ← 0
STI	Set interrupt enable flag IF ← 1
CLI	Clear interrupt enable flag IF ← 0
NOP	No operation
HLT	Halt after interrupt is set
WAIT	Wait for TEST pin active
ESC opcode mem/ reg	Used to pass instruction to a coprocessor which shares the address and data bus with the 8086
LOCK	Lock bus during next instruction

Control Transfer Instructions تعليمات نقل التحكم

- Transfer the control to a specific destination or target instruction تنقل التحكم إلى تعليمة هدف محدد أو مصدر محدد
- Do not affect flags. لا تؤثر على الأعلام

❑ 8086 Unconditional transfers نقل غير مشروط

Mnemonics	Explanation
CALL reg/ mem/ disp16	Call subroutine
RET	Return from subroutine
JMP reg/ mem/ disp8/ disp16	Unconditional jump

Control Transfer Instructions تعليمات نقل التحكم

- ❑ تعليمات التفريع المشروط مع الإشارة
- ❑ 8086 signed conditional branch instructions

- ❑ تعليمات التفريع المشروط دون الإشارة
- ❑ 8086 signed conditional branch instructions

- اختبار الأعلام Checks flags
- إذا كان الشروط محققة يجري نقل التحكم بالبرنامج إلى موقع جديد في الذاكرة في نفس القطاع من خلال تعديل محتوى IP.
- If conditions are true, the program control is transferred to the new memory location in the same segment by modifying the content of IP

❑ تعليمات التفريع المشروط دون الإشارة

❑ 8086 signed conditional branch instructions

Name	Alternate name
JE disp8 Jump if equal	JZ disp8 Jump if result is 0
JNE disp8 Jump if not equal	JNZ disp8 Jump if not zero
JA disp8 Jump if above	JNBE disp8 Jump if not below or equal
JAE disp8 Jump if above or equal	JNB disp8 Jump if not below
JB disp8 Jump if below	JNAE disp8 Jump if not above or equal
JBE disp8 Jump if below or equal	JNA disp8 Jump if not above

❑ تعليمات التفريع المشروط مع الإشارة

❑ 8086 signed conditional branch instructions

Name	Alternate name
JE disp8 Jump if equal	JZ disp8 Jump if result is 0
JNE disp8 Jump if not equal	JNZ disp8 Jump if not zero
JG disp8 Jump if greater	JNLE disp8 Jump if not less or equal
JGE disp8 Jump if greater than or equal	JNL disp8 Jump if not less
JL disp8 Jump if less than	JNGE disp8 Jump if not greater than or equal
JLE disp8 Jump if less than or equal	JNG disp8 Jump if not greater

```
mov  al,48
mov  bl,4
imul bl              ; AX = 00C0h, OF=1
```

```
mov ax,8760h
mov bx,100h
imul bx
```

DX = FF87h, AX = 6000h, OF = 1

```
mov eax,00128765h
mov ecx,10000h
mul ecx
```

EDX = 00000012h, EAX = 87650000h, CF = 1

DIV Examples

Divide 8003h by 100h, using 16-bit operands:

```
mov dx,0            ; clear dividend, high
mov ax,8003h        ; dividend, low
mov cx,100h         ; divisor
div cx              ; AX = 0080h, DX = 3
```

مثال : ماهو مجتوى المسجلين Ax , dx بعد بنقيذ عملية القسمة

```
mov dx,0087h
mov ax,6000h
mov bx,100h
div bx
```

DX = 0000h, AX = 8760h

CBW, CWD, CDQ Instructions

- **The CBW, CWD, and CDQ instructions provide important sign-extension operations:**
  - CBW (convert byte to word) extends AL into AH
  - CWD (convert word to doubleword) extends AX into DX
  - CDQ (convert doubleword to quadword) extends EAX into EDX
- Example:

```
mov eax,0FFFFFF9Bh        ; (-101)
cdq              ; EDX:EAX = FFFFFFFFFFFFFF9Bh
```

Example: 16-bit division of –48 by 5

```
mov  ax,-48
cwd                ; extend AX into DX
mov  bx,5
idiv bx            ; AX = -9,  DX = -3
```

Example: 8-bit division of –48 by 5

```
mov  al,-48
cbw                ; extend AL into AH
mov  bl,5
idiv bl            ; AL = -9,  AH = -3
```

Example: eax = (-var1 * var2) + var3

```
mov   eax,var1
neg   eax
imul  var2
jo    TooBig          ; check for overflow
add   eax,var3
jo    TooBig          ; check for overflow
```

Example: var4 = (var1 + var2) * var3

```
; Assume unsigned operands
mov   eax,var1
add   eax,var2        ; EAX = var1 + var2
mul   var3            ; EAX = EAX * var3
jc    TooBig          ; check for carry
mov   var4,eax        ; save product
```

Example: var4 = (var1 * 5) / (var2 - 3)

```
mov   eax,var1        ; left side
mov   ebx,5
imul  ebx             ; EDX:EAX = product
mov   ebx,var2        ; right side
sub   ebx,3
idiv  ebx             ; EAX = quotient
mov   var4,eax
```

مثال اكتب مقطع برمجي لتحقيق المعادلة التالية لأعداد ب اشارة من 32 bit

```
eax = (ebx * 20) / ecx
```

```
mov eax,20
imul ebx
idiv ecx
```

Example: `var4 = (var1 * -5) / (-var2 % var3);`

```
mov   eax,var2          ; begin right side
neg   eax
cdq                     ; sign-extend dividend
idiv  var3              ; EDX = remainder
mov   ebx,edx           ; EBX = right side
mov   eax,-5            ; begin left side
imul  var1              ; EDX:EAX = left side
idiv  ebx               ; final division
mov   var4,eax          ; quotient
```