

جدول
ASCII

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	-	127	7F	[DEL]

كود مرحبا بالعالم

```
.model tiny
.code
org 100h
```

```
main proc near
```

```
mov ah,09h
mov dx,offset message
int 21h
```

```
mov ah,4ch
mov al,00
int 21h
```

```
endp
message db "Hello World $"
```

```
end mai
```

```

01
02
03
04 .model tiny ; com program , Code Data & Stack in one 64K Segment
05 .code ; code segment
06 org 100h ; code starts at offset 100h
07
08
09
10
11 main proc near
12
13 mov ah,09h ; function to display a string
14 mov dx,offset message ; offset of Message string terminating with $
15 int 21h ; dos interrupt
16
17 mov ah,4ch ; function to terminate
18 mov al,00
19 int 21h ; Dos Interrupt
20
21 endp
22 message db "Hello World $" ; Message to be displayed terminating with $
23
24 end main

```

كود اعادة طباعة حرف مدخل

.data

.code

mov ah, 1h
int 21h

mov dl, al
mov ah, 2h
int 21h

end

```

04 .model small
05
06 .data
07
08 .code
09
10
11 mov ah, 1h
12
13 int 21h
14
15 mov dl, al
16 mov ah, 2h
17
18 int 21h
19
20
21
22
23 end

```

كود عرض رقم 2 عن طريق اضافة ما يقابله من جدول ASCII

حيث ان الرمز المقابل

هو رقم

2

.model small

.data

.code

main proc

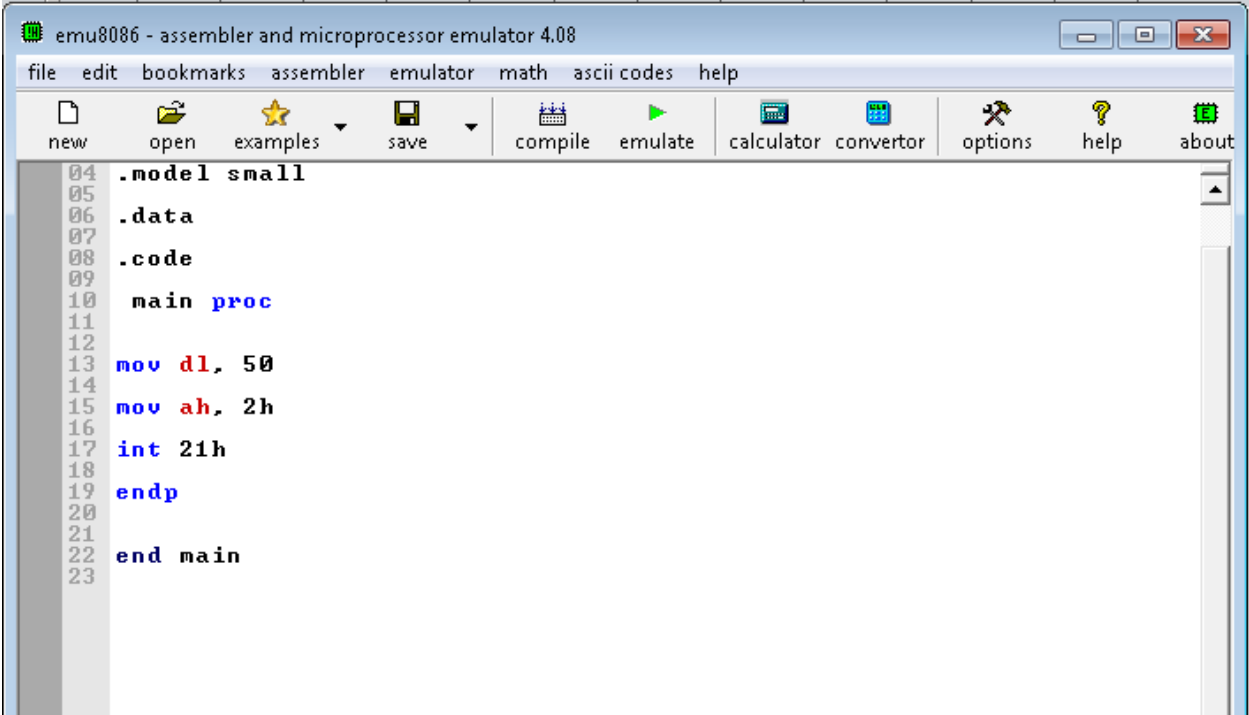
mov dl, 50

mov ah, 2h

int 21h

endp

end main



The screenshot shows the emu8086 - assembler and microprocessor emulator 4.08 window. The interface includes a menu bar (file, edit, bookmarks, assembler, emulator, math, ascii codes, help) and a toolbar with icons for new, open, examples, save, compile, emulate, calculator, convertor, options, help, and about. The main text area displays the following assembly code:

```
04 .model small
05
06 .data
07
08 .code
09
10 main proc
11
12
13 mov dl, 50
14
15 mov ah, 2h
16
17 int 21h
18
19 endp
20
21
22 end main
23
```

كود جمع رقمين في

Hex

و عرض الرمز المقابل له

.model small

.data

.code

main proc

mov dl, 2h

add dl, 30h

mov ah, 2h

int 21h

endp

end main

```
04 .model small
05
06 .data
07
08 .code
09
10 main proc
11
12
13 mov dl, 2h
14 add dl, 30h
15
16 mov ah, 2h
17
18 int 21h
19
20 endp
21
22
23 end main
24
```

کود عرض ناتج جمع رقمین

.model small

.data

.code

main proc

mov dl, 2

mov dl, 2

add dl, 48

mov ah, 2h

int 21h

endp

end main

```
13 mov dl, 2
14
15 mov dl, 2
16
17
18 add dl, 48
19
20 mov ah, 2h
21
22 int 21h
23
24 endp
25
26
27 end main
28
```

كود عرض ناتج عمليي طرح

.model small

.data

.code

main proc

mov dl, 2

sub dl, 1

add dl, 48

mov ah, 2h

int 21h

endp

end main

```
01  
02  
03  
04 .model small  
05  
06 .data  
07  
08 .code  
09  
10 main proc  
11  
12  
13 mov dl, 2  
14  
15 sub dl, 1  
16  
17  
18 add dl, 48  
19  
20 mov ah, 2h  
21  
22 int 21h  
23  
24 endp  
25  
26  
27 end main  
28
```

تعريف متغير و اعطائه قيمة

.model small

.data

count1 db 2 ; المتغير

.code

main proc

mov dl, count1

add dl, 48

mov ah, 2h

int 21h

endp

end main

```
01  
02  
03  
04 .model small  
05  
06 .data  
07  
08     count1 db 2 ;  
09 .code  
10  
11 main proc  
12  
13  
14 mov dl, count1  
15  
16  
17 add dl, 48  
18  
19 mov ah, 2h  
20  
21 int 21h  
22  
23 endp  
24  
25  
26 end main  
27
```

كيفية تخزين قسمة اقل الى مسجل ذو قيمة اكبر

.model small

.data

var1 db 1 ;

.code

main proc

mov ax, 0

add al, var1

endp

end main

```
01  
02  
03  
04 .model small  
05  
06 .data  
07  
08     vari db 1 ;  
09 .code  
10  
11 main proc  
12  
13  
14 mov ax, 0  
15  
16  
17 add al, vari  
18  
19 endp  
20  
21  
22 end main  
23
```

زيادة تلقائية بمعدل 1

```
.model small  
.data  
.code  
main proc  
mov dl, 3
```

```
inc dl ; الزيادة
```

```
add dl, 48  
mov ah, 2h  
int 21h
```

```
endp
```

```
end main
```

انقاص تلقائي للقيمة بمعدل 1

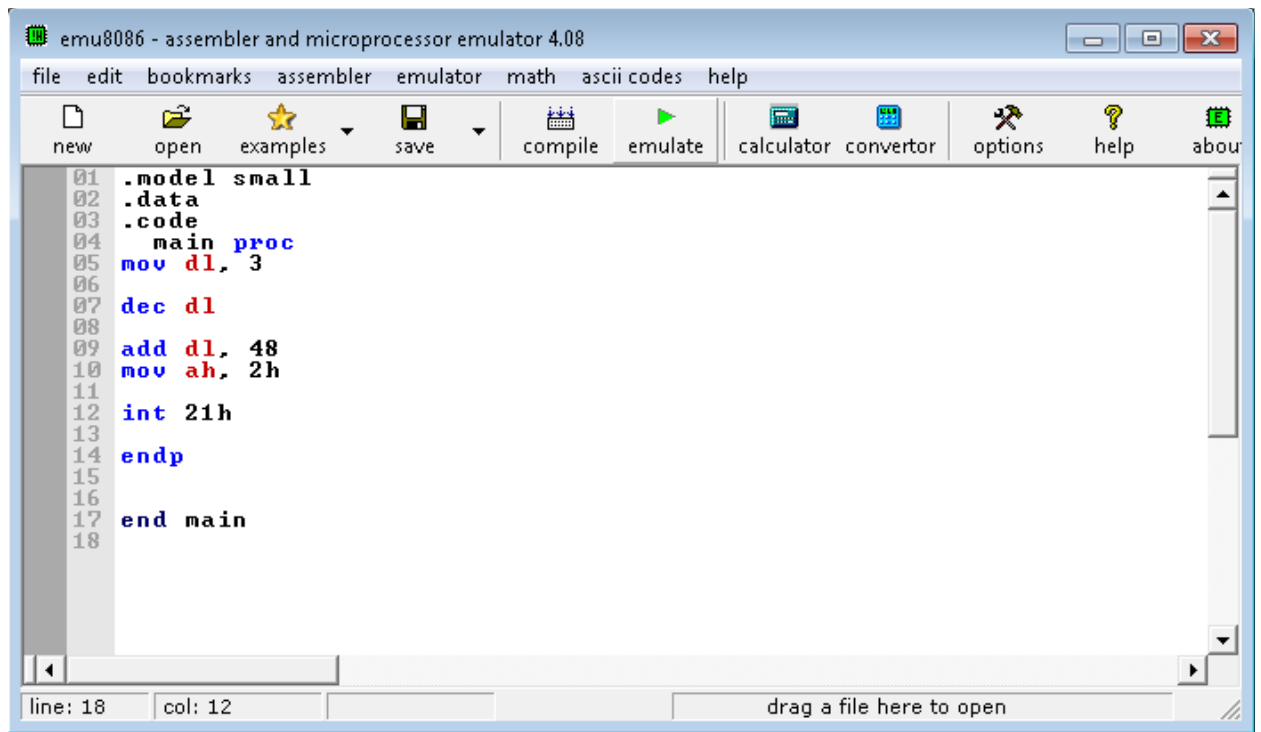
```
.model small  
.data  
.code  
main proc  
mov dl, 3
```

```
dec dl ; الانقاص يحدث هنا
```

```
add dl, 48  
mov ah, 2h  
int 21h
```

```
endp
```

```
end main
```

مثال على القفز الى نقطة

.model small

.data

.code

main proc

top: ; نقطة العودة

mov dl, 5 ; ادخل الى المسجل

add dl, 48 ; اضافة الى المسجل

mov ah, 2h ; طباعة على الشاشة

int 21h ;

jmp top ; التوجيه الى نقطة العودة

endp

end main

```
01 .model small
02 .data
03 .code
04 main proc
05
06
07 top:
08 mov dl, 5
09
10 add dl, 48
11
12 mov ah, 2h
13
14 int 21h
15
16 jmp top
17
18 endp
19
20
21 end main
22
23
```

كود جملة دواره مع التحكم فى عدد المرات

```
.model small
.data
.code
main proc
    mov cx, 5 ; عدد المرات الذى سيفيذها الدوران هى خمس مرات
    top:
    mov dl, 6
    add dl, 48
    mov ah, 2h
    int 21h
    loop top ; الدوران و العوده الى top
endp
```

```
01 .model small
02 .data
03 .code
04 main proc
05
06     mov cx, 5
07
08     top:
09     mov dl, 6
10
11     add dl, 48
12
13     mov ah, 2h
14
15     int 21h
16
17
18     loop top
19
20
21     endp
22
23
```

مثال على الجمل الدوارة المركبة

```
.model small
.data
```

```
.code
```

```
main proc
```

```
    mov cx, 5 ; تحديد عدد مرات الدوران
```

```
    lop1: ; نقطة بدء الدوران الاول
    mov dl, 6
```

```
    add dl, 48
```

```
    mov ah, 2h
```

```
    int 21h
```

```
    lop2: ; نقطة بدء الدوران الثاني
```

```
    mov dl, 7
```

```
    add dl, 48
```

```
    mov ah, 2h
```

```
    int 21h
```

```
    loop lop2 ; الذهاب الى الدوران الثاني
```

```
    mov cx, 5 ; اعادة تحديد عدد الدوران
```

```
    loop lop1 ; الذهاب الى الدوران الاول
```

```
endp
```

```
end main
```

```

01 .model small
02 .data
03 .code
04     main proc
05
06         mov cx, 5
07
08         lop1:
09     mov dl, 6
10
11     add dl, 48
12
13     mov ah, 2h
14
15     int 21h
16
17     lop2:
18
19         mov dl, 7
20
21     add dl, 48
22
23     mov ah, 2h
24
25     int 21h
26
27
28     loop lop2
29
30
31         mov cx, 5
32
33     loop lop1
34
35     endp
36
37
38     end main
39
40
41

```

مثال على

Push & pop

وظيفة

Pop

هي الحصول على اخر قيمة تم ادخالها بطريق

Push

```

.model small
.data
.code
main proc
    mov ax, 1
    push ax
    pop cx
    mov ah, 2h
    int 21h
    endp
end main

```

```

01 .model small
02 .data
03 .code
04   main proc
05
06   mov ax, 1
07   push ax
08
09   pop cx
10
11   mov ah, 2h
12
13   int 21h
14
15
16
17   endp
18
19
20   end main
21
22

```

مثال على

popf & pushf

```

.model small
.data

```

```

Veribl dw "Hello"

```

```

.code
main proc

```

```

pushf

```

```

pop Veribl
push Veribl
popf

```

```

mov ah, 2h

```

```

int 21h

```

```

endp

```

```

end main

```

انشاء اجراء و استدعاؤه

```

.model small
.data

```

```

.code

```

```

main proc

```

```

call myproc

```

```

endp

```

```

myproc proc
mov dl, 5

```

```
add dl, 48
mov ah, 2h
int 21h
ret
myproc endp
```

```
end main
```

```
01 .model small
02 .data
03
04
05
06 .code
07
08
09 main proc
10     call myproc
11
12     endp
13
14 myproc proc
15     mov dl, 5
16
17     add dl, 48
18
19     mov ah, 2h
20     int 21h
21
22     ret
23
24     myproc endp
25
26
27
28 end main
29
30
31
```

مثال علی
And

```
.model small
.data
```

```
.code
```

```
main proc
```

```
mov ah, 00000101b
mov bh, 00000001b
```

```
and ah, bh
```

```
endp
```

```
end main
```

مثال علی
Or

.model small

.data

.code

main proc

mov ah, 00000100b

mov bh, 00000011b

or ah, bh

endp

end main

\$

مثال على

Xor

.model small

.data

.code

main proc

mov ah, 11111111b

mov bh, 11111110b

xor ah, bh

endp

end main

\$

مثال على

not

.model small

.data

.code

main proc

mov ah, 01111110b

not ah

```
endp
```

```
end main
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

مثال على

test

```
.model small
```

```
.data
```

```
.code
```

```
main proc
```

```
mov ah, 01111110b
```

```
test ah, 01111110b
```

```
endp
```

```
end main
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

مثال على المقارنة بين قيمتين ايهما اكبر و معرفة النتيجة من خلال

Flag

ZF , CF

```
.model small
```

```
.data
```

```
.code
```

```
main proc
```

```
mov ax, 5
```

```
cmp ax, 5
```

```
Endp
```


القفز المشروط

الكلمة المختزلة	المعنى
JC	القفز إذا كان $CF = 1$
JNC	القفز إذا كان $CF = 0$
JO	القفز إذا كان $OF = 1$
JNO	القفز إذا كان $OF = 0$
JS	القفز إذا كان $SF = 1$
JNS	القفز إذا كان $SF = 0$
JCXZ	القفز إذا كان $CX = 0000$
JE/JZ	القفز في حالة التساوي/أو إذا كان الناتج يساوي الصفر
JGE/JNL	القفز إذا كان أكبر أو يساوي/القفز إذا لم يكن أصغر
JA/JNBE	القفز إذا كان فوق/القفز إذا لم يكن تحت أو يساوي
JAE/JNB	القفز إذا كان فوق أو يساوي/القفز إذا لم يكن تحت
JB/JNAE	القفز إذا كان تحت/القفز إذا لم يكن فوق أو يساوي
JBE/JNA	القفز إذا كان تحت أو يساوي/القفز إذا لم يكن فوق
JG/JNLE	القفز إذا كان أكبر/القفز إذا لم يكن أصغر أو يساوي
JLE/JNG	القفز إذا كان أصغر أو يساوي/القفز إذا لم يكن أكبر
JNE/JNZ	القفز إذا لم يكن يساوي/القفز إذا كان الناتج يساوي قيمة غير صفرية
JNB/JBO	القفز إذا كانت خانة Parity غير موجودة/القفز إذا كان $PF = 0$
JP/JPE	القفز في حالة وجود خانة Parity/القفز إذا كان $PF = 1$

مثال على القفز في حالة إذا كان

Flag = fz

عند المقارنة

```
.model small
.data
```

```
.code
```

```
main proc
```

```
mov ax, 5
```

```
cmp ax, 5
```

```
jz lp1
```

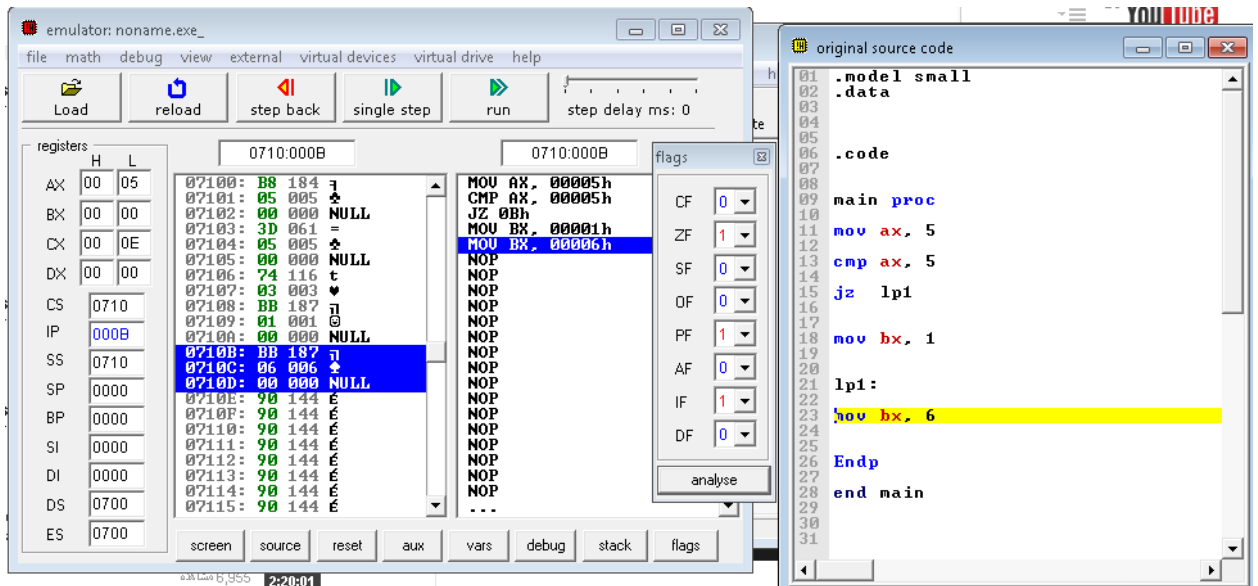
```
mov bx, 1
```

```
lp1:
```

```
mov bx, 6
```

```
Endp
```

end main



مثال على القفز في حالة التساوي Je

```
.model small  
.data
```

```
.code
```

```
main proc
```

```
mov ax, 100
```

```
cmp ax, 100
```

```
je lp1
```

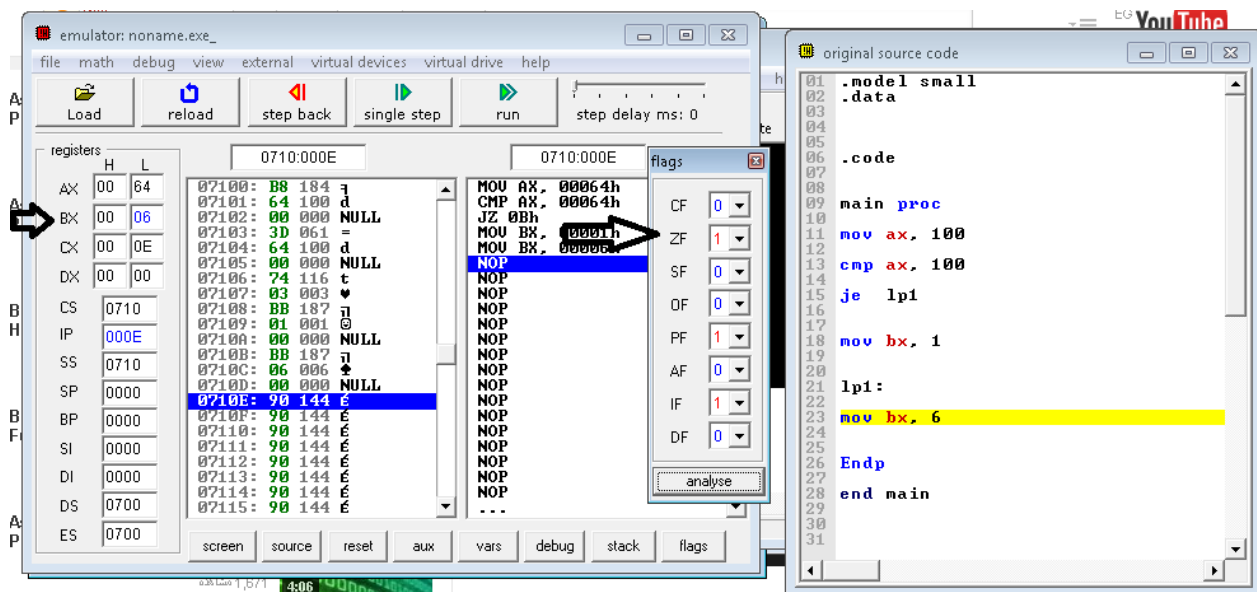
```
mov bx, 1
```

```
lp1:
```

```
mov bx, 6
```

```
Endp
```

```
end main
```



مثال على القفز

jcxz

.model small

.data

.code

main proc

mov cx, 1

mov ax, 1

sub cx, ax

jcxz lp1

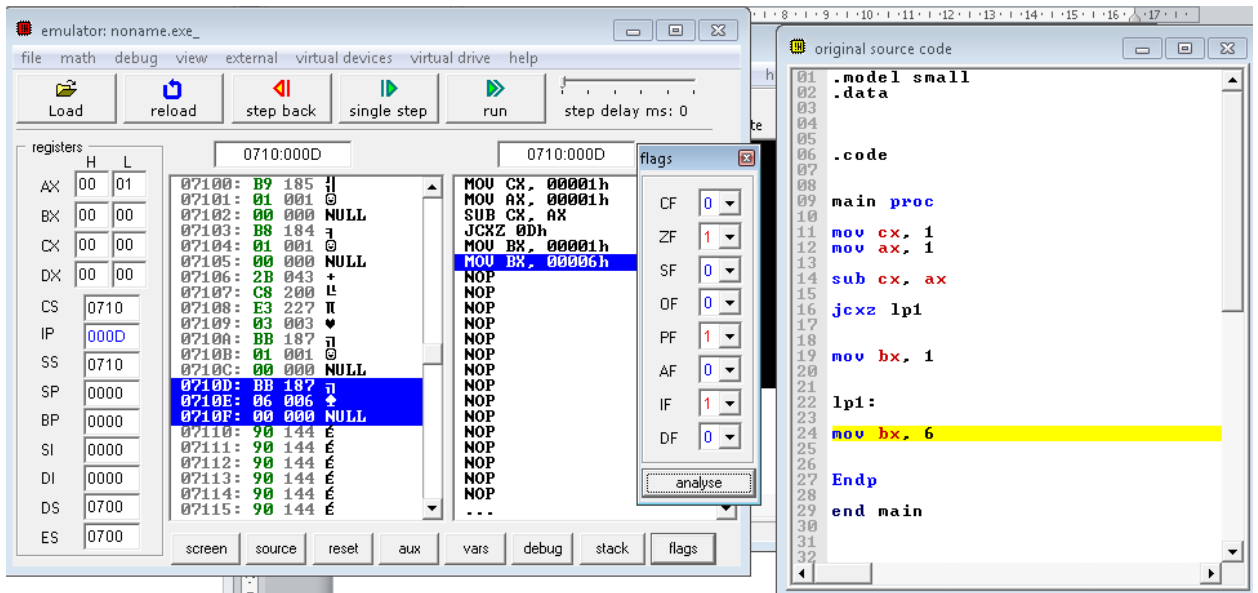
mov bx, 1

lp1:

mov bx, 6

Endp

end main



مثال على القفز

Ja

model small.
data.

code.

main proc

mov ax, 11
mov bx, 12

cmp bx, ax

ja lp1

mov bx, 1

:lp1

mov bx, 6

Endp

end main

end main

\$

إزاحة رياضية/إزاحة منطقية و كلاهما نحو اليسار : SAL/SHL

مثال

.model small

.data

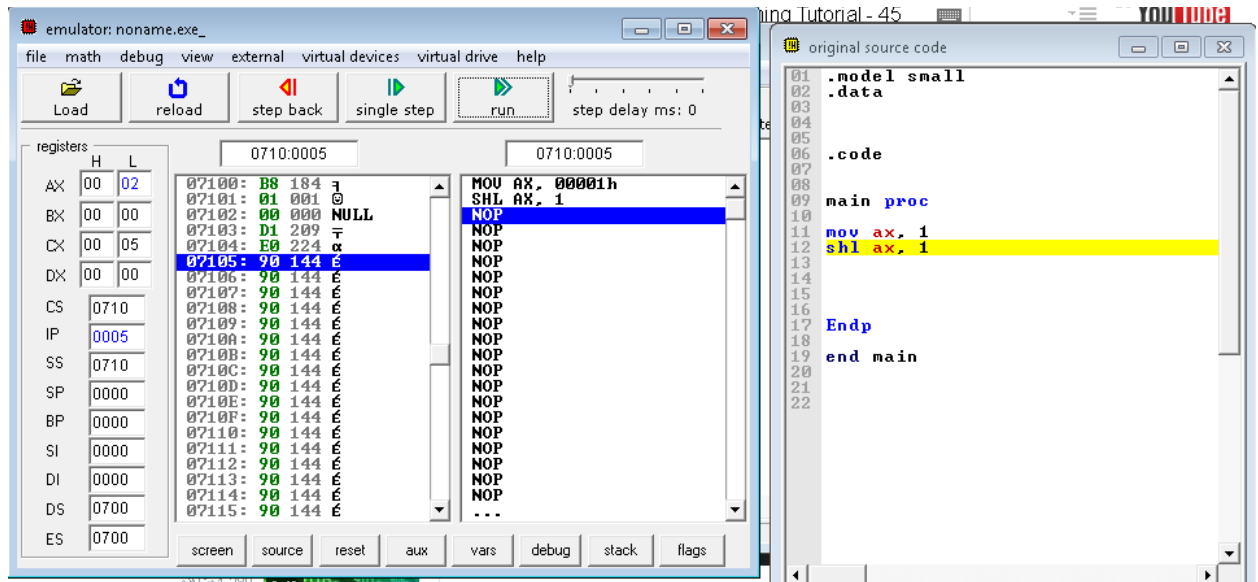
.code

main proc

mov ax, 1
shl ax, 1

Endp

end main



إزاحة رياضية/إزاحة منطقية و كلاهما نحو اليمين : SAR/SHR

مثال

.model small

.data

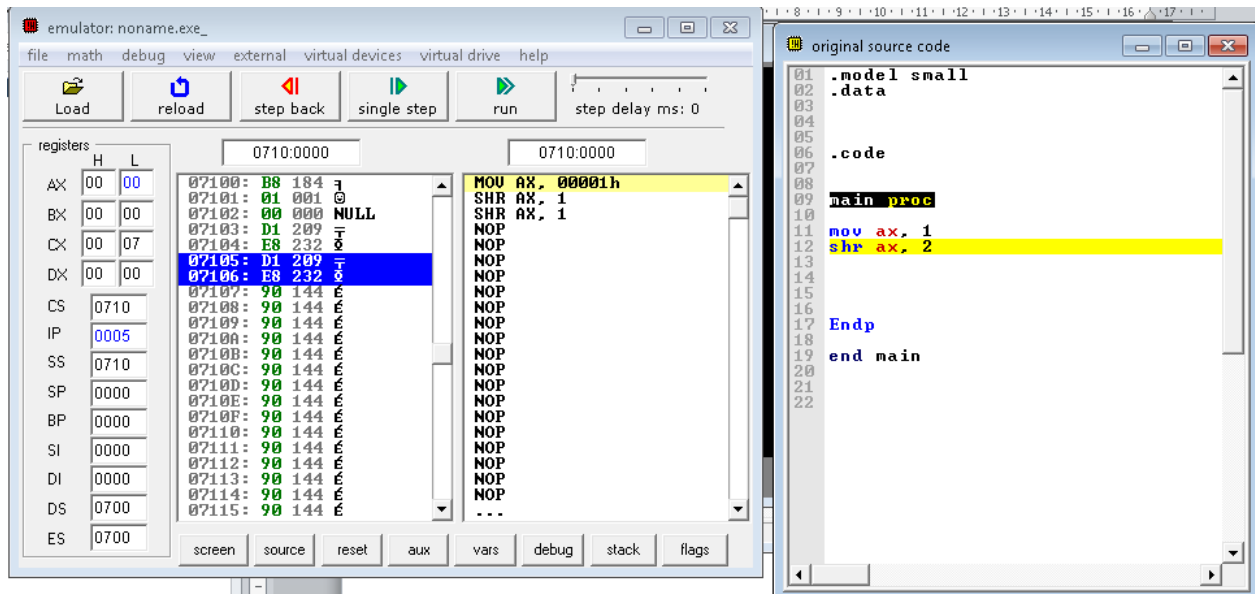
.code

main proc

mov ax, 1
shr ax, 2

Endp

end main



\$

MUL : ضرب بدن إشارة

DIV : تقسيم بدون إشارة

.model small

.data

.code

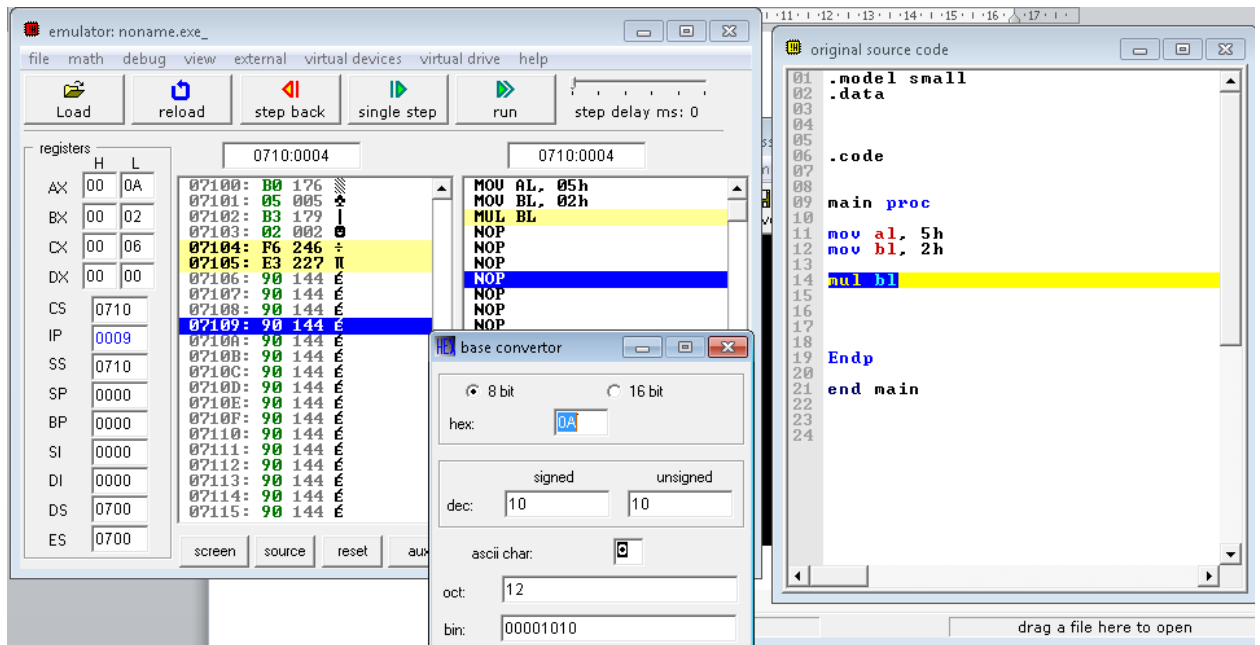
main proc

mov al, 5h
mov bl, 2h

mul bl

Endp

end main



مثال على

Div

.model small

.data

.code

main proc

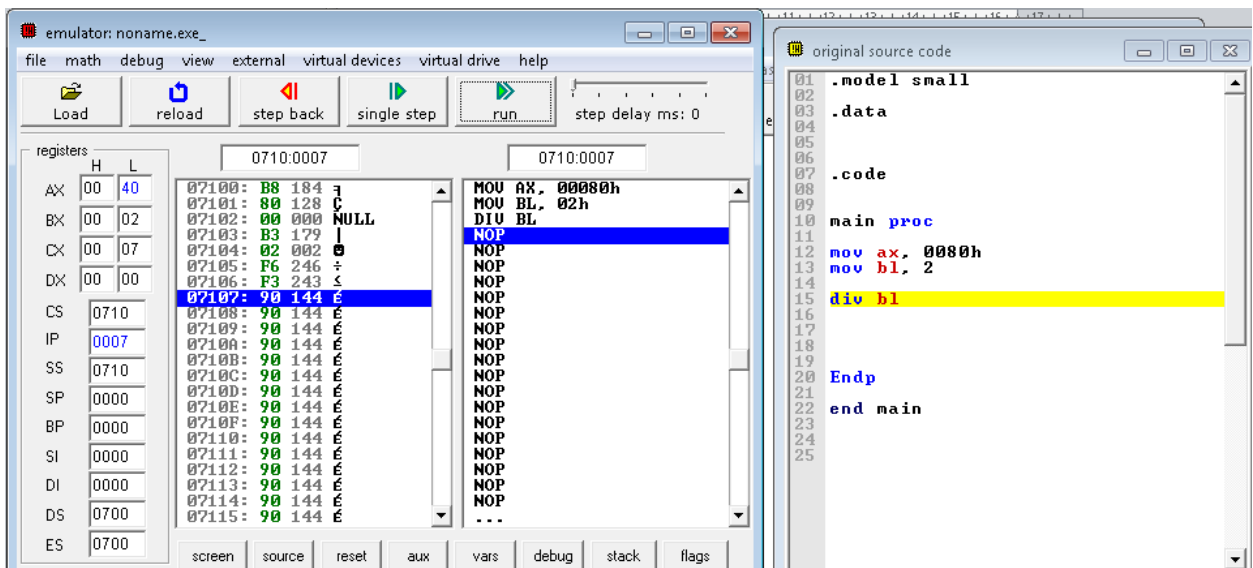
mov ax, 0080h

mov bl, 2

div bl

Endp

end main



مثال بسيط على الرسم

name "vga"

; this program draws a tiny rectangle in vga mode.

org 100h

jmp code

; dimensions of the rectangle:

; width: 10 pixels

; height: 5 pixels

w equ 10

h equ 5

; set video mode 13h - 320x200

code: mov ah, 0

mov al, 13h

int 10h

; draw upper line:

mov cx, 100+w ; column

mov dx, 20 ; row

mov al, 15 ; white

u1: mov ah, 0ch ; put pixel

int 10h

dec cx

cmp cx, 100

jae u1

; draw bottom line:

mov cx, 100+w ; column

mov dx, 20+h ; row

mov al, 15 ; white

```
u2: mov ah, 0ch ; put pixel
int 10h

dec cx
cmp cx, 100
ja u2
```

; draw left line:

```
mov cx, 100 ; column
mov dx, 20+h ; row
mov al, 15 ; white
u3: mov ah, 0ch ; put pixel
int 10h

dec dx
cmp dx, 20
ja u3
```

; draw right line:

```
mov cx, 100+w ; column
mov dx, 20+h ; row
mov al, 15 ; white
u4: mov ah, 0ch ; put pixel
int 10h

dec dx
cmp dx, 20
ja u4
```

; pause the screen for dos compatibility:

```
;wait for keypress
mov ah,00
int 16h
```

```
; return to text mode:
mov ah,00
mov al,03 ;text mode 3
int 10h
```

ret

\$

مرحبا بالعالم بكود اصغر

org 100h

jmp start

msg: db "Hello, World!", 0Dh,0Ah, 24h

```
start: mov dx, msg
       mov ah, 09h
       int 21h

       mov ah, 0
       int 16h

ret
```

(محمد احمد عبد الغنى ابراهيم) ترجمان عربى

Mo.am86@yahoo.com

<https://www.facebook.com/mohuomha.mo>

للحصول على برامج مجانية

<http://arabicturgeman.blogspot.com/>