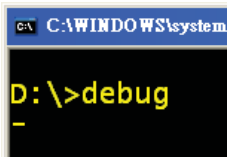
開始→執行→cmd

```
D:\>debug
-
```

開始→執行→cmd



AX, BX, CX, DX = registers 寄存器
AX=AH+AL, …, DX=DH+DL

```
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0B42  ES=0B42  SS=0B42  CS=0B42  IP=0100   NV UP EI PL NZ NA PO NC
```

Instruction Pointer
下一執行句

```
-u
0B42:0100  B401      MOV     AH,01
0B42:0102  CD21      INT     21
0B42:0104  CD20      INT     20
0B42:0106  CC        INT     3
0B42:0107  F6C702    TEST    BH,02
0B42:010A  7548      JNZ     0154
```

低階語言 Low Level Languages

記憶體地址 address　機器碼 machine code　匯編語言 Assembly Language

```
// getche()
// 輸入字符
mov ah,01
int 21
int 20
```

AX, BX, CX, DX = registers 寄存器

```
D:\>DEBUG
-u        -a
          0C8A:0100 mov ax,0200
          0C8A:0103 mov dx,0041
          0C8A:0106 int 21
          0C8A:0108 int 20
          0C8A:010A

          -h 010A 0100
          020A  000A

          -n test.com

-r        -r cx
          CX 0000
          :000A

          -w
          Writing 0000A bytes
          -q
```

AX=AH+AL

41(hex)
65(dec)
DX='A'

INT 21, AH=02
putc(DX); 輸出

結束
return(0);

計算
010A+0100 = 020A
010A–0100 = A(10)

檔名

更改
CX=10

存檔

quit

```
D:\>test.com
A
```



```
D:\>debug
-a
0B38:0100       mov ah,09
0B38:0102       mov dx,0109
0B38:0105       int 21
0B38:0107       int 20
0B38:0109       db 'How are you?$'
0B38:0116

-g
How are you?

-a
0B38:0100       MOV CX,09
0B38:0103       MOV DL,41
0B38:0105       MOV AH,02
0B38:0107       INT 21
0B38:0109       INC DL
0B38:010B       LOOP 0105
0B38:010D       INT 20
0B38:010F

-g
ABCDEFGHI
```

輸出文字串
puts(dx)

define byte
文字串

重複09次

41(hex)=65(dec)
DL='A'

INT 21, AH=02
putc(DX); 輸出

DL++

-e 100 輸入資料 enter hex or char data
-e cs:100 "This is a string."

-u 列出指令
translates memory into mnemonics.

-d 100 顯示 dump memory
-g 執行 go/execute

? (Help)
A (Assemble)
C (Compare)
D (Dump)
E (Enter)
F (Fill)
G (Go)
H (Hex arith)
I (Input)
L (Load)

Status Register 狀態寄存器

Set (1)	***Clear (0)***
OV = Overflow	NV = No Overflow
DN = Direction Down	UP = Direction Up
EI = Interrupts Enabled	DI = Interrupts Disabled
NG = Sign Flag negative(-)	PL = Sign Flag positive(+)
ZR = Zero	NZ = Not Zero
AC = Auxiliary Carry	NA = No Auxiliary Carry
PO = Odd Parity	PE = Even Parity
CY = Carry	NC = No Carry

M (Move)
N (Name)
P (Ptrace)
Q (Quit)
R (Register)
S (Search)
T (Trace)
U (Unassemble)
W (Write)

開始→執行→cmd

```
D:\>debug
-
```

文字輸出

```
0100 MOV AH,09
0102 MOV DX,0113     ; puts(message)
0105 INT 21
```

輸入字符

```
0107 MOV AH,07       ; AL=getch()
0109 INT 21
```

AL=1B

```
010B CMP AL,1B       ; AL==ESC(1B=27)?
010D JZ  0111        ; 相同 jump zero 0111
010F JMP 0107        ; 不同 jump 0107
0111 INT 20          ; stop
```

JNZ 0107

```
0113 DB "press a key, esc to quit $"
```

```
printf("press a key, esc to quit ");
do{
        c = getch();
}while(c!=ESC);
```

C:\>debug

int 10

```
0100 MOV AH,02--| set cursor position
0103 MOV DH,06--| row(y) 6,
0105 MOV DL,40--| col(x) 40(=64)
0107 MOV BH,00--| page no.
0109 INT 10
```

```
// clear
// screen
mov ax,600
mov cx,0
mov dx,184f
mov bh,07
int 10
int 20
```

```
0100 MOV AH,09--| print char
0102 MOV AL,41--| letter A(41)
0104 MOV BH,00--| page no.
0106 MOV BL,FC--| attribute color
0108 MOV CX,05--| display 5 times
010B INT 10
010D INT 20
```

http://www.ablmcc.edu.hk/~scy/home/javascript/text-color.htm

char s[50]="\nThis is my ... \n";

```
-a 100
xxxx:0100 jmp 126          ; Jump to 0126
xxxx:0102 db 0d,0a,"This is my first DEBUG program"
xxxx:0123 db 0d,0a,"$"
xxxx:0126 mov ah,9         ; puts 輸出文字
xxxx:0128 mov dx,102       ; address of data(string)
xxxx:012B int 21           ; execute 執行
xxxx:012D mov ah,0
xxxx:012F int 21           ; Terminate 結束 Program.
xxxx:0131
-g=100
```

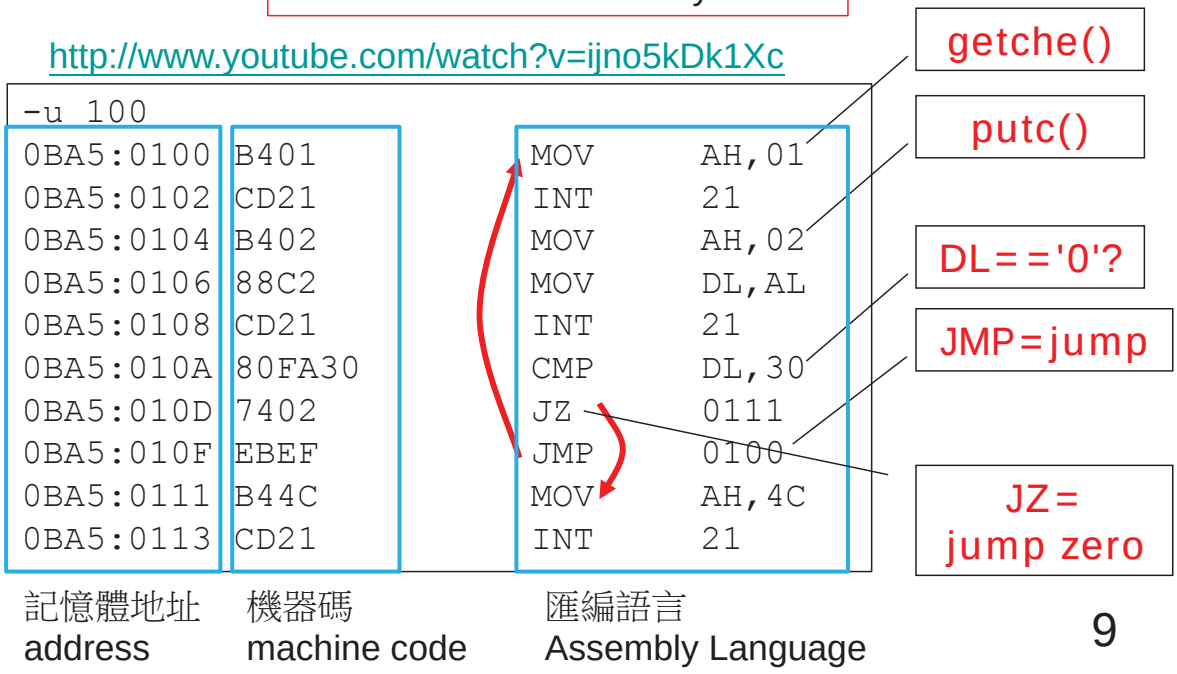
```
_
This is my first DEBUG program!
_
```

int 21 (debug)

http://www.ctyme.com/intr/int.htm

AH=01 AL=getche() 輸入	AH=02 DL=char 輸出 putc (DL)
AH=07 AL=getch() 輸入	AH=09 DX=addr 輸出 puts (DX) db "abc" define byte

http://www.youtube.com/watch?v=ijno5kDk1Xc



記憶體地址 address　機器碼 machine code　匯編語言 Assembly Language

Thus, a value of AH=02h and AL=00h can be expressed as AX=0200h.

INT 21h,2h
Description: 輸出 outputs character to STDOUT
Inputs:
  AH = 02h
  DL = char value
Outputs: none

INT 20h
Description: 結束 program terminate

SP is the stack pointer,
IP is the instruction pointer (PC – program counter)
(next instruction下一指令 to be executed)

參考
http://illegalargumentexception.blogspot.com/2008/05/assembler-using-debugexe-to-write-dos.html
http://teaching.idallen.com/dat2343/00f/using_dos_debug.htm
http://kipirvine.com/asm/debug/Debug_Tutorial.pdf
http://www.computerhope.com/rdebug.htm
http://home.educities.edu.tw/wanker742126/asm/ch01.html
http://home.educities.edu.tw/wanker742126/asm/ch36.html
http://www.armory.com/~rstevew/Public/Tutor/Debug/debug8.htm

The 8088 instruction set (41 instructions)

Data transfer instructions	Arithmetic instructions
MOV--------move	ADD----------- addition
PUSH, POP--stack operation	INC----------- increment
XCHG-------exchange	SUB------------subtract
IN,OUT-----input/output	DEC------------decrement
	NEG------------negate (two's comp)
	CMP------------compare
	MUL------------multiply
	DIV------------divide

IRQ0	INT 8	Time of day tick count
IRQ1	INT 9	Keyboard
IRQ2	INT A	Color graphic's adapter
IRQ3	INT B	Secondary serial adapter
IRQ4	INT C	Primary serial adapter
IRQ5	INT D	Hard drive (XT)
IRQ6	INT E	Floppy drive
IRQ7	INT F	Printer

Logical instructions	String instructions
NOT-------complement	MOVS----- move string
AND-------and	CMPS----- compare string
OR--------inclusive or	SCAS----- scan string
XOR-------exclusive or	LODS----- load from a string
TEST------test bits	STOS----- store into string
SHL,SHR---shift left/right	
ROL,ROR---rotate left/right	

Transfer of control instructions	PROCESSOR CONTROL
CALL------goto a sub-routine	CLC,STC---clear/set flags
RET-------return from a sub-routine	HLT-------halt CPU
JMP-------jump	
JZ,JNZ----conditional jumps	
LOOP------iteration	
LOOPNE----conditional iteration	
INT-------interrupt	
IRET------return from interrupt	