P1_0 : SHORT REPORT ON DEBUG TUTORIAL

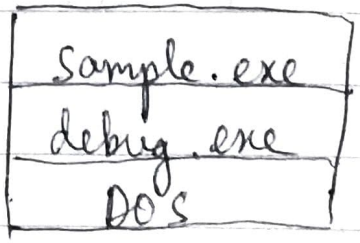
A debugger displays the contents of the memory and lets us view registers and variables as they change. It can be used to test assembler instructions, try out new programming ideas, or to carefully step through your programs.

Functions of a Debugger.

- Assemble short programs
- View program source code along with its machine code.
- Trace or execute a program, watching variables for changes.
- Enter new values into memory.
- Search for Binary and ASCII values in memory. Move a block of memory from one location to another.
- Fill a block of memory.
- Load and write disk files and sectors.

Command to debug a sample program

- debug sample.exe



Debug Commands

Program Creation and Debugging	Memory Manipulation	Miscellaneous	Input - Output
Assemble program using inst. mnemonics (A)	Compare memory ranges (C)	Perform Hex Addition and Subtraction (H)	Input a byte from port (I)
Execute program in memory (G)	Display content of memory (D)	Quit Debug and return to DOS (Q)	Send a byte to port (O)
Display contents of registers & flags (R)	Enter bytes into memory (E)		Load data from memory to disk (L)
Proceed past an instruction or loop (P)	Fill a mem. range with single value (F)		Write data from memory to disk (W)
Trace a Single instruction (T)	Move bytes from one mem. range to another (M)		Create a filename for use by the L and W commands (N)
Disassemble memory into mnemonics (U)	Search a mem. range for specific values (S)		

Default Values

1. All segment registers are set to the bottom of free memory, just above debug exe program.
2. IP is set to 0100H
3. 256 bytes of stack space is reserved at the end of current segment by debug.
4. All of available memory is allocated.
5. BX CX are set to the length of the current program or file.

6. Flags

NV	(Overflow flag clear)
UP	(Direction flag = UP)
EI	(Interrupts enabled)
PL	(Sign flag = Positive)
NZ	(Zero flag clear)
NA	(Auxillary carry flag clear)
PO	(odd parity)
NC	(carry flag clear).

Command Parameters

Address eg. F000:100, DS:200, 0AF5

Filespec eg. file1, C:\arm\progs\test.com

List eg. 10, 20, 30, 40 ; 'A', 'B', 50

Range eg. Format : address, [address] 100, 500

Sector String Value

'COMMAN' eg. 3A, 3A6F

Important Commands.

① assemble A [address]

⇒ Assemble a program into machine Language.

eg A 100 Assemble at CS : 100h
A Assemble at curr. location.
A DS : 2000 Assemble at DS : 2000h

② compare C range address.

⇒ Compares bytes between a specified range with the same no. of bytes at target address.

eg C 100 105 200 Bytes bet^n DS:100 and DS:105 are compared to bytes at DS : 0200.

③ dump D [range]

⇒ displays memory on the screen as single byte in both Hex and ASCII.

eg. D 150 15A dump DS:150 through 015A

④ enter E address [list]

⇒ Place individual bytes in memory on supplying starting memory location.

eg E CS:100 "This is a string"

⑤ fill F range list

⇒ Fills a range of memory with a single value or list of values.

eg F CS:300 CS:1000, FF fill locations CS:300 through CS:1000 with hex FFh.

⑥ GO G [=address] [addresses]

⇒ Execute the program in memory.

eg G Execute from current location to end of program.

G=10 50 Begin execution at CS:10 and stop before the instruction at offset CS:50.

⑦ Hex H value1 value2

⇒ Performs addition and subtraction on two hexadecimal numbers.

eg. H 1A 10

⇒ Display 2A 0A.

⑧ input I port

⇒ inputs a byte from a specified input/output port and displays it in hexadecimal.

eg - I 378 Display 00.

⑨ load L [address] [drive] [firstsector] [number]

⇒ loads a file (or logical disk sectors) into memory at a given address.

eg L 100 2 A 5 load five sectors from drive 2 starting at logical sector number 0Ah.

⑩ move M range address

⇒ Copies a block of data from one memory location to another.

eg M 100 105 110 Move bytes in the range DS: 100-105 to location DS: 110

(11) Name N [pathname] [arglist]

⇒ initialize filename in memory.

eg N b:myfile.dta.

(12) Output O port byte

⇒ Outputs a byte to a specified port.

eg O. 3F8 00

(13) Proceed P [=address] [number]

⇒ executes one or more instructions / subroutines

eg P=150 6 executes 6 instructions starting at CS:0150

(14) Quit Q

quits debug and return to DOS.

(15) R (Register) R [register]

⇒ display register and flag contents, allowing them to be changed.

eg R display contents of all registers.

R F display all flags and prompt for a new flag value.

(16) Search S range list

⇒ Searches a range of addresses for a sequence of one or more bytes.

eg S 100 1000 0D Search DS:0100 to DS:1000 for the value of 0Dh

(17) trace T [:address] value

eg T=105 10 (10 → hex) Trace 16 instructions starting at CS:105

(18) Unassemble U [range]

⇒ translate memory into assembly language mnemonics.

eg U 100 108 disassemble bytes from CS:100 to CS:108.

(19) Write W [address] [drive] [firstsector] [number]

⇒ Write a block of memory to a file or individual disk sectors.

eg W 100 0 02 Write two sectors to drive A from location CS 0100 starting at logical sector no. 0.

```
; Q1: SUM OF ARRAY ELEMENTS
.MODEL SMALL
.STACK 100H
.DATA
      ARR DB 5,3,1,7,9,2,6,4,8,10   ;ARRAY ELEMENTS
      LEN DW $-ARR                  ;LENGTH OF ARRAY
      SUM DW ?                ;SUM
.CODE
START:      MOV AX, @DATA
      MOV DS, AX
      MOV SI, 0
      MOV AX, 0
      MOV CX, LEN
REPEAT: MOV BL, ARR[SI]
      MOV BH, 0
      ADD AX, BX
      INC SI
      DEC CX
      JNZ REPEAT
      MOV SUM, AX
      MOV AH, 4CH
      INT 21H
      END START
.END
```

```
; Q2. AVERAGE OF ARRAY ELEMENTS
.MODEL SMALL
.STACK 100H
.DATA
      ARR DB 7,8,6,5,7,3,6    ;ARRAY ELEMENTS
      LEN DW $-ARR            ;LENGTH OF ARRAY
      AVG DW ?          ;AVERAGE
.CODE
START:      MOV AX, @DATA
      MOV DS, AX
      MOV SI, 0         ;LOAD OFFSET TO SI
      MOV AX, 0         ;INITIALIZE SUM=0
      MOV CX, LEN       ;LOOP VARIABLE
REPEAT: MOV BL, ARR[SI]       ;8 BIT NUMBER
      MOV BH, 0         ;FIRST 8 BITS=0
      ADD AX, BX        ;ADDITION
      INC SI                  ;INCREMENT OF OFFSET ADDRESS
      DEC CX                  ;DECREMENT OF COUNT
      JNZ REPEAT        ;IF ZF=0 REPEAT
      MOV DX, LEN
      DIV DL                  ;DIVIDING FOR AVERAGE
      MOV AVG, AX       ;STORING AVERAGE
      MOV AH, 4CH
      INT 21H
      END START
.END
```

```
; Q3. MINIMUM AND MAXIMUM OF ARRAY ELEMENTS
.MODEL SMALL
.STACK 100H
.DATA
      ARR DB 7,8,6,5,7,3,6    ;ARRAY ELEMENTS
      LEN DW $-ARR            ;LENGTH OF ARRAY
      MIN DB ?
      MAX DB ?
.CODE
START:      MOV AX, @DATA
      MOV DS, AX
      MOV SI, 0         ;LOAD OFFSET TO SI
      MOV AL, ARR[SI]         ;INITIALIZE MIN=FIRST ELEMENT
      MOV MIN,AL        ;INITIALIZE MIN
      MOV MAX,AL        ;INITIALIZE MAX
      MOV CX, LEN       ;LOOP VARIABLE
      INC SI
      DEC CX
REPEAT: MOV BL, ARR[SI]       ;8 BIT NUMBER
      INC SI                  ;INCREMENT OF OFFSET ADDRESS
      DEC CX                  ;DECREMENT OF COUNT
      CMP MIN, AL       ;COMPARE MIN and CURRENT NO.
      JLE SKIP
      MOV MIN, AL       ;[MIN]=[AL]
SKIP:       CMP MAX, AL       ;COMPARE MAX AND CUURENT NO.
      JGE NEXT
      MOV MAX, AL       ;[MAX]=[AL]
NEXT: JNZ REPEAT        ;IF ZF=0 REPEAT
      MOV AH, 4CH
      INT 21H
      END START
.END
```

```
;SWAP 2 NUMBERS
.MODEL SMALL
.STACK 100H
.DATA
      DATA1 DB 52H            ;FIRST NUMBER
      DATA2 DB 29H            ;SECOND NUMBER
.CODE
START:      MOV AX, @DATA
      MOV DS, AX
      MOV AL, DATA1           ;COPYING FIRST NUMBER
      MOV AH, DATA2           ;COPYING SECOND NUMBER
      MOV DATA1, AH           ;COPYING 2ND TO 1ST LOCATION
      MOV DATA2, AL           ;COPYING 1ST TO 2ND LOCATION
      MOV AH, 4CH
      INT 21H
      END START
.END
```

```
; Converting BCD to Hexadecimal
.MODEL SMALL
.STACK 100H
.DATA
      DATA1 DB 99H            ;BCD NUMBER IN HEX FORM
      HEX   DB ?
.CODE
START:      MOV AX, @DATA
      MOV DS, AX
      MOV AL, DATA1           ;COPYING NUMBER TO AL
      MOV BL, DATA1           ;COPYING NUMBER TO BL
      AND AL, 0F0H            ;MASKING LAST 4 BITS
      AND BL, 0FH       ;MASKING FIRST 4 BITS
      MOV CL, 04H       ;COUNT ROTATION
      ROR AL, CL        ;ROTATING RIGHT BY 4
      MOV DL, 0AH       ;STORING DL=10 IN DECIMAL
      MUL DL                  ;MULTIPLICATION OF AL AND DL
      ADD AL, BL        ;ADDING AL AND BL
      MOV HEX, AL       ;STORING HEXADECIMAL VALUE
      MOV AH, 4CH
      INT 21H
      END START
.END
```

```
; Adding 2 4 digits BCD numbers
.MODEL SMALL
.STACK 100H
.DATA
      DATA1 DB 45H            ;FIRST NUMBER
      DATA2 DB 56H            ;SECOND NUMBER
      DATA3 DB ?        ;NEW NUMBER AFTER ADDITION
      CARRY DB ?        ;CARRY AFTER ADDITION
.CODE
START:      MOV AX, @DATA
      MOV DS, AX
      MOV AL, DATA1           ;STORING FIRST NUMBER
      MOV BL, DATA2           ;STORING SECOND NUMBER
      ADD AL, BL        ;ADDITION
      DAA               ;DECIMAL ADJUSTMENT
      MOV DATA3, AL           ;STORING ANSWER
      MOV AL, 00H       ;AL=0
      ADC AL, AL        ;ADDING CARRY TO AL
      MOV CARRY, AL           ;STORING CARRY
      MOV AH, 4CH
      INT 21H
      END START
.END
```

```
; Sum of 2 digit Hexadecimal No.
.MODEL SMALL
.STACK 100H
.DATA
      DATA1 DB 99H            ;STORING THE HEX NUMBER
      SUM   DB ?        ;SUM OF DIGITS
.CODE
START:      MOV AX, @DATA
      MOV DS, AX
      MOV AL, DATA1           ;COPYING THE NUMBER
      MOV AH, DATA1           ;COPYING THE NUMBER
      AND AL, 0FH       ;MASKING FIRST 4 BITS
      AND AH, 0F0H            ;MASKING LAST 4 BITS
      MOV CL, 04H       ;COUNT FOR ROTATION
      ROR AH, CL        ;ROTATING RIGHT BY 4 BITS
      ADD AL, AH        ;ADDING BOTH THE DIGITS
      MOV SUM,AL        ;STORING SUM
      MOV AH, 4CH
      INT 21H
      END START
.END
```

```
; Binary to Gray Code Convertor
.MODEL SMALL
.STACK 100H
.DATA
      BIN  DB 07H       ;BINARY NUMBER
      GRAY DB ?         ;GRAY CODE
.CODE
START:      MOV AX, @DATA
      MOV DS, AX
      MOV AL, BIN       ;COPYING BINARY NUMBER
      MOV BL, BIN       ;COPYING BINARY NUMBER
      SHR BL, 01        ;SHIFTING BY 1 AND MSB=0
      XOR AL, BL        ;XOR TO GET GRAY CODE
      MOV GRAY,AL       ;STORING GRAY CODE
      MOV AH, 4CH
      INT 21H
      END START
.END
```

```
;Count the number of set bits
.MODEL SMALL
.STACK 100H
.DATA
      DATA1 DB 99H            ;ORIGINAL NUMBER
      SET   DB ?        ;NUMBER OF SET BITS
.CODE
START:      MOV AX, @DATA
      MOV DS, AX
      MOV AL, DATA1           ;COOYING FIRST DATA
      MOV BL, 00        ;INITIALIZING TO 0
      MOV CX, 0008H           ;COUNT=8 FOR LOOP
REPEAT:     RCR AL, 01        ;ROTATING RIGHT THROUGH CARRY
      JNC SKIP          ;IF CARRY=0 SKIP NEXT STEP
      INC BL                  ;INCREMENT COUNT
SKIP: LOOP REPEAT       ;WHILE CX>0 LOOP CONTINUES
      MOV SET, BL       ;STORING NO. OF SET BIT
      MOV AH, 4CH
      INT 21H
      END START
.END
```