Name : CHANDRAWANSHI MANGESH SHIVAJI
Roll Number : 1801CS16
Filename : 1801CS16_Lab3.pdf
Course : CS322 (Computer Architechture Lab)
Date : 15/09/2020

Short Report on Debug Tutorial

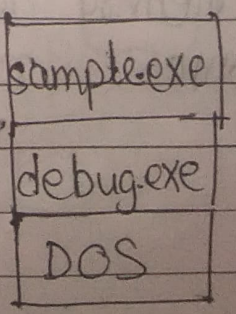
A debugger displays the contents of the memory and lets us view registers and variables as they change. It can be used to test assembler instructions, try out new programming ideas, or to carefully step through your programs.

Functions of a debugger

- Assemble short programs
- View programs source code along with its machine code.
- View the CPU registers and flags
- 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

Assemble program using inst. mnemonics (A).

Execute program in memory (G).

Display contents of registers & flags (R)

Proceed past an instruction/loop (P)

Trace a single instruction (T)

Disassemble memory into mnemonics (U)

Memory Manipulation

Compare memory ranges (C)

Display (Dump) content of memory (D)

Enter bytes into memory (E)

Fill a memory range with single value (F)

Move bytes from one memory range to another (M)

Search a memory range for specific values (S)

Miscellaneous

Perform hexa decimal Add^n and Sub^n (H)

Quit Debug and return to DOS (Q)

Input-Output

Input a byte from port (I)

Send a byte to port (O).

Load: data from disk (L)

Write data from memory to disk (W)

Create a file name for use by the L and W commands (N).

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:\asm\progs\test.com

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

Range eg. format 1: 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 current location
A DS:2000 Assemble at DS:2000h

② compare C range address

Compares bytes between a specified range with the same number of bytes at a target address

eg. C 100 105 200 Bytes betn DS:100 and DS:010
are compared to bytes at DS:0200

③ dump D [range]

displays memory on the screen as single byt
in both hexadecimal and ASCII

eg. D 150 15A dump DS:0150 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 G10 G1 [=address] [addresses]

Execute the program in memory.

eg. G Execute from current location to end of program.

G1=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 hexa-decimal 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 3F8 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 C, starting at logical sector number 0A11.

⑩ move M range address

copies a block of data from one memory location to another

eg. M 100 105 1100 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 execute 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 → 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 number 0.

#P1_1:Write an assembly language program to find sum of 10 random numbers#

```
.MODEL SMALL
.STACK 64
.DATA

SZ   DB 10                          ;size of array
ARR  DB 1,2,12,15,14,5,6,7,8,9      ;array elements
SUM    DB ?                                  ;difference to be stored

      .CODE
MAIN  PROC  FAR         ;this is the program entry point
            MOV   AX,@DATA    ;load the data segment address
            MOV   DS,AX       ;assign value to DS
            LEA SI,ARR      ;load Memory Location of First element into SI
            LEA DI,SZ
            MOV CL,[DI]       ;Initialize CX with size of array (Here,10)
            MOV AL,00         ;Initialize AL to zero

LOOP1:      ADD AL,[SI]       ;Add data into accumulator
            INC SI            ;Increment pointer to array element
            DEC CL                  ;Decrement counter
            JNZ LOOP1         ;Loop again if CX is non zero
            MOV SUM,AL      ;Store the answer in SUM
            HLT
            MOV AH,4CH
            INT   21H               ;return to DOS
MAIN        ENDP
            END MAIN          ;this is the program exit point
```

OUTPUT : SUM = 4FH

#P1_2: Write an assembly language program to find average of the given set of 16 bit number numbers#

```
.MODEL SMALL
.STACK 64
.DATA

SZ   DW 8                         ;size of array
ARR  DW 12,15,14,5,6,7,8,9      ;array elements
AVG    DW ?                              ;difference to be stored

      .CODE
MAIN  PROC  FAR         ;this is the program entry point
            MOV   AX,@DATA    ;load the data segment address
            MOV   DS,AX       ;assign value to DS
            MOV CX,SZ         ;store size in CX as a counter
            MOV BX,CX       ;store size for division
            LEA SI,ARR      ;load address of start of array to SI
            MOV AX,0000       ;initialze AX with zero

LOOP1:      ADD AX,[SI]       ;Add current element to AX
```

```
            INC SI                  ;Increment SI
            INC SI
            DEC CX                  ;Decrement Counter CX
            JNZ LOOP1          ;if CX is non zero go to LOOP

            DIV BX                  ;Divide AX with BX to get average
            MOV AVG,AX         ;Move it to AVG
            HLT

            MOV   AH,4CH            ;set up to
            INT   21H               ;return to DOS
MAIN        ENDP
            END MAIN           ;this is the program exit point
```

OUTPUT : AVG = 09H

#P1_3:Write a program to find largest and smallest among an array of numbers and print the difference between them#

```
.MODEL SMALL
.STACK 64
.DATA

SZ   DB 8                        ;size of array
ARR  DB 12,15,14,5,6,7,8,9       ;array elements
DIFF DB ?                                ;difference to be stored

      .CODE
MAIN  PROC  FAR          ;this is the program entry point
            MOV   AX,@DATA    ;load the data segment address
            MOV   DS,AX       ;assign value to DS
            MOV CL,SZ         ;store size in CL
            LEA SI,ARR      ;load address of start of array to SI
            MOV AL,[SI]       ;Move First element to AL
            MOV BL,[SI]       ;Move First element to AL

MIN :   CMP AL,[SI]           ;Compare AL with current element
            JC MAX                  ;if AL is smaller go to MAX
            MOV AL,[SI]       ;else mov that element to AL

MAX : CMP BL,[SI]       ;Compare BL with current element
            JNC LOOP1         ;if BL is larger go to LOOP1
            MOV BL,[SI]       ;elese mov that element to BL

LOOP1 : INC SI                ;Increment SI to position of next element in the
array
            DEC CX                  ;Decrement counter(CX)
            JNZ MIN           ;If CX is non zero go to MIN
            SUB BL,AL         ;subtract smallest from largest
            MOV DIFF,BL       ;move it to DIFF
            HLT

            MOV   AH,4CH            ;set up to
```

```
            INT   21H               ;return to DOS
MAIN        ENDP
            END MAIN          ;this is the program exit point
```

OUTPUT : DIFF = 0AH

#P1_4a:Program to count number of 1's in a byte#

```
.MODEL SMALL
.STACK 64
.DATA

NUM  DB 28       ;Given number
CNT    DB ?        ;count to be stored

      .CODE
MAIN  PROC  FAR         ;this is the program entry point
            MOV   AX,@DATA    ;load the data segment address
            MOV   DS,AX       ;assign value to DS
            SUB BL,BL         ;clear BL to keep number of 1s
            MOV DL,8          ;counter to rotate total 8 times
            MOV AL,NUM      ;store NUM value in AL

LOOP1 : ROL AL,1        ;rotate it once
            JNC GO          ;if Carry Flag = 0, go to GO
            INC BL                  ;if Carry Flag = 1, add one to count
GO :  DEC DL                ;Decrease counter
            JNZ LOOP1         ;if counter(DL) is non zero loop again
            MOV CNT,BL        ;Move count of 1's to CNT
            HLT

            MOV   AH,4CH            ;set up to
            INT   21H               ;return to DOS
MAIN        ENDP
            END MAIN          ;this is the program exit point
```

OUTPUT : CNT = 03H

#P1_4b:Program to reverse given array at same location#

```
.MODEL SMALL
.STACK 64
.DATA

ARR  DB 12,15,14,5,6,7,8,9      ;array elements

      .CODE
MAIN  PROC  FAR         ;this is the program entry point
            MOV   AX,@DATA    ;load the data segment address
            MOV   DS,AX       ;assign value to DS
            MOV CX,0004       ;store half the size in CX
            LEA SI,ARR      ;load address of start of array to SI
```

```
            MOV DI,SI
            ADD DI,0007       ;Store memory location of last element in DI (Just
ADD size of array - 1 to SI)

LOOP1:   MOV AL,[SI]          ;Move current element to AL
            XCHG AL,[DI]      ;swap element at [DL] and in AL
            MOV [SI],AL       ;Move swapped element to [SI]
            INC SI                  ;Increment SI
            DEC DI                  ;Decrement DI
            DEC CX                  ;Decrement counter
            JNZ LOOP1
            HLT

            MOV   AH,4CH            ;set up to
            INT   21H               ;return to DOS
MAIN        ENDP
            END MAIN          ;this is the program exit point

OUTPUT : 09H, 08H, 07H, 06H, 05H, EH, FH, CH


#P1_4c:Program to find Factorial of a number#

.MODEL SMALL
.STACK 64
.DATA

NUM DW 0005H

.CODE
MAIN  PROC  FAR         ;this is the program entry point
            MOV   AX,@DATA    ;load the data segment address
            MOV   DS,AX       ;assign value to DS
          LEA SI,NUM          ;load address of num into SI
          LEA DI,NUM+100H     ;load memory location NUM+100H into DI
          MOV CX,[SI];  ;Move value at SI, given number to CX
            MOV AX,0001;      ;Initialize AX with 1
            MOV DX,0000;      ;Initialize DX with 0

LOOP1 :     MUL CX                  ;Multiply AX with current value of CX
            DEC CX                  ;Decrement CX
            JNZ LOOP1         ;If CX is non zero loop again

            MOV [DI],AX;      ;Move value in AX to memory location stored in DI
            MOV [DI+1],DX;    ;Move value in DX to DI+1
            HLT

            MOV   AH,4CH            ;set up to
            INT   21H               ;return to DOS
MAIN        ENDP
            END MAIN          ;this is the program exit point

OUTPUT : 78H (Factorial of 0005H)


#P1_4d:Program To find the no of even & odd nos. from given array of nos.#

.MODEL SMALL
.STACK 64
```

```
.DATA

SZ       DB 8                        ;size of array
ARR      DB 12,13,14,5,6,7,8,9       ;array elements
CNTODD       DB ?                              ;cnt of odd numbers
CNTEVEN      DB ?                              ;cnt of even numbers

.CODE
MAIN  PROC  FAR         ;this is the program entry point
            MOV   AX,@DATA    ;load the data segment address
            MOV   DS,AX       ;assign value to DS
          LEA SI,ARR          ;load address of num into SI
            MOV BX,0000       ;Initialize BX with 0
            LEA DI,SZ
            MOV CL,[DI]       ;Initialize CX with size of array

LOOP1:  MOV AL,[SI]     ;Load data into Accumulator
        AND AL,01H      ;AND with 01H
        JZ  GO          ;If AND is 0, Jump to EVEN
        INC BL          ;Increment BL (stores cnt of odd)
        JMP NEXT        ;Jump to next
GO    :   INC BH          ;Increment BH (stores snt of even)
NEXT:   INC SI          ;Increment SI
        DEC CL               ;Decrement CL
        JNZ LOOP1       ;Loop until CL is non zero
            HLT

            MOV   AH,4CH            ;set up to
            INT   21H               ;return to DOS
MAIN        ENDP
            END MAIN          ;this is the program exit point

OUTPUT : CNTODD = 04H, CNTEVEN = 04H


#P1_4e:Program To check for a Palindrome (single letter)#

.MODEL SMALL
.STACK 64
.DATA

SZ       DB 7                                ;size of array
ARR      DB 'A','B','C','D','C','B','A'     ;array elements

.CODE
MAIN  PROC  FAR         ;this is the program entry point
            MOV AX,@DATA    ;Load Data in temp register
        MOV DS,AX       ;Load data into data into Data Segment
        LEA SI,ARR      ;Load address of first element into SI
        LEA DI,ARR+06H  ;Load address of last element into DI (i.e ARR + SZ-1)
        MOV CL,03H      ;Load half the size of ARR into CL
        MOV CH,00H           ;Initialize CH to 0

LOOP1:  MOV AH,[SI]     ;Load data into AH
        MOV BH,[DI]     ;Load data into BH
        CMP AH,BH       ;Compare AH and BH
        JNZ GO          ;If not zero skip
        INC SI          ;Increment pointer
        DEC DI          ;Decrement pointer
```

```
        DEC CL          ;Decrement pointer
        JNZ LOOP1       ;If not zero, jump to back
        INC CH          ;Increment CH
GO:     HLT

            MOV   AH,4CH            ;set up to
            INT   21H               ;return to DOS
MAIN        ENDP
            END MAIN          ;this is the program exit point

; In this if finally, CH = 01H then given array is a palindrome

OUTPUT : CH = 01H;

#P1_4e:Addition of two 16-bit nos#

.MODEL SMALL
.STACK 64
.DATA

NUM1  DW 8514H
NUM2  DW 5362H
SUM   DW ?
CARRY DB 00H

      .CODE
MAIN  PROC  FAR         ;this is the program entry point
            MOV   AX,@DATA    ;load the data segment address
            MOV   DS,AX       ;assign value to DS
            MOV AX,NUM1     ;Move from NUM1 into accumulator
          ADD AX,NUM2     ;Add NUM2 to AX
          JNC SKIP        ;If there is no carry, skip
          INC CARRY       ;else Increment carry

SKIP:   MOV SUM, AX     ;Store the answer
            HLT

            INT   21H               ;return to DOS
MAIN        ENDP
            END MAIN          ;this is the program exit point


OUTPUT : SUM = D876H and CARRY = 00H
```