DEBUG详细用法

目录

[DEBUG详细用法 1](#_Toc407569273)

[DEBUG命令一览 2](#_Toc407569274)

[1.读写寄存器 2](#_Toc407569275)

[2.汇编和反汇编 2](#_Toc407569276)

[3.执行指令 2](#_Toc407569277)

[4.读写内存 2](#_Toc407569278)

[5.读写磁盘 3](#_Toc407569279)

[6.读写I/O端口 3](#_Toc407569280)

[7.十六进制加减 3](#_Toc407569281)

[8.退出DEBUG 3](#_Toc407569282)

[8. 标志寄存器 3](#_Toc407569283)

[示例一:DEBUG环境下建立和汇编程序 3](#_Toc407569284)

[输入并汇编程序 4](#_Toc407569285)

[运行程序 4](#_Toc407569286)

[反汇编程序 4](#_Toc407569287)

[将机器指令程序送到起始地址为200H单元 4](#_Toc407569288)

[执行机器指令程序 4](#_Toc407569289)

[示例二:显示修改寄存器内容 5](#_Toc407569290)

[显示所有寄存器内容 5](#_Toc407569291)

[修改寄存器 5](#_Toc407569292)

[修改标志寄存器 5](#_Toc407569293)

[示例三:汇编,汇编指令 5](#_Toc407569294)

[汇编命令 6](#_Toc407569295)

[反汇编命令 6](#_Toc407569296)

[示例四:执行程序 6](#_Toc407569297)

[单步执行 6](#_Toc407569298)

[多步执行 7](#_Toc407569299)

[说明:T和P的不同 7](#_Toc407569300)

[连续执行 7](#_Toc407569301)

[断点执行 8](#_Toc407569302)

[示例五:修改和显示内存内容 8](#_Toc407569303)

[显示命令 8](#_Toc407569304)

[修改命令 8](#_Toc407569305)

[填充命令 8](#_Toc407569306)

[搜索命令 9](#_Toc407569307)

[移动命令 9](#_Toc407569308)

[比较命令 9](#_Toc407569309)

[示例六:读写磁盘 10](#_Toc407569310)

[指定文件命令 10](#_Toc407569311)

[写入命令 10](#_Toc407569312)

[装入命令 11](#_Toc407569313)

[示例七:I/O端口的输入输出 11](#_Toc407569314)

[输入命令 11](#_Toc407569315)

[输出命令 11](#_Toc407569316)

[示例八:十六进制加减 11](#_Toc407569317)

# DEBUG命令一览

## 1. 读写寄存器

-R 显示所有寄存器内容

-R 寄存器名 显示和修改指定寄存器内容

-RF 显示和修改标志寄存器内容

## 2. 汇编和反汇编

-A [内存地址] 从指定地址开始汇编

-U [内存块] 对指定内存块进行反汇编

## 3. 执行指令

-T [:内存地址] [条数] 单步或多步执行指令

-P [= 内存地址] [条数] 单步或多步执行指令

-G [= 内存地址] 连续执行指令

-G [= 内存地址] 断点地址 设断点执行指令

## 4.读写内存

-D [内存块] 显示指定内存块内容

-E 内存地址 字符或数值串 修改指定内存内容

-F 内存块 字符或数值串 填充指定内存块

-S 内存块 字符串或数值 在指定内存块中查找串

-M 内存块1 内存块2的首地址 复制内存块内容

-C 内存块1 内存块2的首地址 比较两个指定内存块

## 5.读写磁盘

-N [d:][PATH]文件名.扩展名 指定想要读写的磁盘文件

-W 内存地址 将指定内存块写入文件

-L [内存地址] 将文件调如内存

## 6.读写I/O端口

-I 端口地址 读入指定端口的内容

-O 端口地址 数值 将数据写入指定端口

## 7.十六进制加减

-H 数值1 数值2 计算并显示两数之和,两数之差

## 8.退出DEBUG

-Q 退出DEBUG,返回DOS

## 8. 标志寄存器

OV = OVerflow, NV = No oVerflow.

DN = DowN, UP (up).

EI = Enable Interupt, DI = Disable Interupt.

NG = NeGative, PL = PLus;

ZR = ZeRo, NZ = Not Zero.

AC = Auxiliary Carry, NA = Not Auxiliary carry.

PE = Parity Even, PO = Parity Odd.

CY = CarrY, NC = No Carry.

# 示例一 DEBUG环境下建立和汇编程序

## 输入并汇编程序

-A 100

0B70:0100 MOV DL, 65

0B70:0102 MOV AH, 2

0B70:0104 INT 21H

0B70:0106 INT 20H

0B70:0108

## 运行程序

-G=100

e

Program terminated normally

## 反汇编程序

-U 100, 108

0B70:0100 B265 MOV DL,65

0B70:0102 B402 MOV AH,02

0B70:0104 CD21 INT 21

0B70:0106 CD20 INT 20

0B70:0108 F8 CLC

## 将机器指令程序送到起始地址为200H单元

-E 200 B2, 65, B4, 02, CD, 21, CD, 20

## 执行机器指令程序

-G=200

e

Program terminated normally

# 示例二:显示修改寄存器内容

## 显示所有寄存器内容

-R

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0B76 ES=0B76 SS=0B76 CS=0B76 IP=0100 NV UP EI PL NZ NA PO NC

0B76:0100 B003 MOV AL,03

## 修改寄存器

-R AX

AX 0000

:1234

-R

AX=1234 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0B76 ES=0B76 SS=0B76 CS=0B76 IP=0100 NV UP EI PL NZ NA PO NC

0B76:0100 B003 MOV AL,03

## 修改标志寄存器

-RF

NV UP EI PL NZ NA PO NC -DN CY

-RF

NV DN EI PL NZ NA PO CY -

# 示例三:汇编,汇编指令

## 汇编命令

-A 1000

0B76:1000 MOV AH, 2

0B76:1002 MOV DL, 61

0B76:1004 INT 21

0B76:1006 INT 20

0B76:1008

## 反汇编命令

-U 0B76:1000 L8

0B76:1000 B402 MOV AH,02

0B76:1002 B261 MOV DL,61

0B76:1004 CD21 INT 21

0B76:1006 CD20 INT 20

# 示例四:执行程序

## 单步执行

-T=0B76:1000

AX=0234 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0B76 ES=0B76 SS=0B76 CS=0B76 IP=1002 NV DN EI PL NZ NA PO CY

0B76:1002 B261 MOV DL,61

-P=0B76:1000

AX=0234 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0B76 ES=0B76 SS=0B76 CS=0B76 IP=1002 NV DN EI PL NZ NA PO CY

0B76:1002 B261 MOV DL,61

## 多步执行

-T=0B76:1000 2

AX=0234 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0B76 ES=0B76 SS=0B76 CS=0B76 IP=1002 NV DN EI PL NZ NA PO CY

0B76:1002 B261 MOV DL,61

AX=0234 BX=0000 CX=0000 DX=0061 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0B76 ES=0B76 SS=0B76 CS=0B76 IP=1004 NV DN EI PL NZ NA PO CY

0B76:1004 CD21 INT 21

-P=0B76:1000 2

AX=0234 BX=0000 CX=0000 DX=0061 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0B76 ES=0B76 SS=0B76 CS=0B76 IP=1002 NV DN EI PL NZ NA PO CY

0B76:1002 B261 MOV DL,61

AX=0234 BX=0000 CX=0000 DX=0061 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0B76 ES=0B76 SS=0B76 CS=0B76 IP=1004 NV DN EI PL NZ NA PO CY

0B76:1004 CD21 INT 21

## 说明:T和P的不同

1.带重复前缀REP/REPNZ/REPZ的串操作指令和LOOP/LOOPZ/LOOPN等循环指令与CX有关,T命令执行是,每执行一次CX减1,就停下来返回DEBUG;而P命令执行是,一直到CX==0,计算机要执行其后的下一条指令时,才停下来返回DEBUG;

2.调用CALL和INT是,T命令在子函数入口处停下来返回DEBUG,可对子程序进行调试,P命令会跳过子函数,直到执行RET或者IRET后才停下来返回DEBUG;

## 连续执行

-G=0B76:1000

a

Program terminated normally

## 断点执行

-G=0B76:1000 1004

AX=0234 BX=0000 CX=0000 DX=0061 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0B76 ES=0B76 SS=0B76 CS=0B76 IP=1004 NV DN EI PL NZ NA PO CY

0B76:1004 CD21 INT 21

# 示例五:修改和显示内存内容

## 显示命令

-D 0B76:1000 L30

0B76:1000 B4 02 B2 61 CD 21 CD 20-44 07 80 74 14 8A 44 0A ...a.!. D..t..D.

0B76:1010 88 87 70 92 43 83 FB 40-75 03 E8 7C 00 FE CE 75 ..p.C..@u..|...u

0B76:1020 EC 80 7C 08 00 74 0D 38-4C 08 73 08 2A 4C 08 8A ..|..t.8L.s.\*L..

## 修改命令

-E DS:1003 'B'

-D DS:1000 L4

0B76:1000 B4 02 B2 42 ...B

-E DS:1003

0B76:1003 42.A

-D DS:1003 L1

0B76:1000 0A

## 填充命令

-F DS:2000 2100 'Hello world'

-D DS:2000 L100

0B76:2000 48 65 6C 6C 6F 20 77 6F-72 6C 64 48 65 6C 6C 6F Hello worldHello

0B76:2010 20 77 6F 72 6C 64 48 65-6C 6C 6F 20 77 6F 72 6C worldHello worl

0B76:2020 64 48 65 6C 6C 6F 20 77-6F 72 6C 64 48 65 6C 6C dHello worldHell

0B76:2030 6F 20 77 6F 72 6C 64 48-65 6C 6C 6F 20 77 6F 72 o worldHello wor

0B76:2040 6C 64 48 65 6C 6C 6F 20-77 6F 72 6C 64 48 65 6C ldHello worldHel

0B76:2050 6C 6F 20 77 6F 72 6C 64-48 65 6C 6C 6F 20 77 6F lo worldHello wo

0B76:2060 72 6C 64 48 65 6C 6C 6F-20 77 6F 72 6C 64 48 65 rldHello worldHe

0B76:2070 6C 6C 6F 20 77 6F 72 6C-64 48 65 6C 6C 6F 20 77 llo worldHello w

0B76:2080 6F 72 6C 64 48 65 6C 6C-6F 20 77 6F 72 6C 64 48 orldHello worldH

0B76:2090 65 6C 6C 6F 20 77 6F 72-6C 64 48 65 6C 6C 6F 20 ello worldHello

0B76:20A0 77 6F 72 6C 64 48 65 6C-6C 6F 20 77 6F 72 6C 64 worldHello world

0B76:20B0 48 65 6C 6C 6F 20 77 6F-72 6C 64 48 65 6C 6C 6F Hello worldHello

0B76:20C0 20 77 6F 72 6C 64 48 65-6C 6C 6F 20 77 6F 72 6C worldHello worl

0B76:20D0 64 48 65 6C 6C 6F 20 77-6F 72 6C 64 48 65 6C 6C dHello worldHell

0B76:20E0 6F 20 77 6F 72 6C 64 48-65 6C 6C 6F 20 77 6F 72 o worldHello wor

0B76:20F0 6C 64 48 65 6C 6C 6F 20-77 6F 72 6C 64 48 65 6C ldHello worldHel

## 搜索命令

-S DS:2000 L30 'Hello'

0B76:2000

0B76:200B

0B76:2016

0B76:2021

## 移动命令

-M DS:2000 L20 DS:3000

-D DS:3000

0B76:3000 48 65 6C 6C 6F 20 77 6F-72 6C 64 48 65 6C 6C 6F Hello worldHello

0B76:3010 20 77 6F 72 6C 64 48 65-6C 6C 6F 20 77 6F 72 6C worldHello worl

## 比较命令

-C DS:2000 L20 DS:3000

-C DS:2000 L30 DS:3000

0B76:2020 64 20 0B76:3020

0B76:2021 48 77 0B76:3021

0B76:2022 65 69 0B76:3022

0B76:2023 6C 74 0B76:3023

0B76:2024 6C 68 0B76:3024

0B76:2025 6F 20 0B76:3025

0B76:2026 20 6E 0B76:3026

0B76:2027 77 6F 0B76:3027

0B76:2028 6F 20 0B76:3028

0B76:2029 72 70 0B76:3029

0B76:202A 6C 61 0B76:302A

0B76:202B 64 72 0B76:302B

0B76:202C 48 61 0B76:302C

0B76:202D 65 6D 0B76:302D

0B76:202E 6C 65 0B76:302E

0B76:202F 6C 74 0B76:302F

# 示例六:读写磁盘

## 指定文件命令

-N C:TEST.TXT

## 写入命令

-R

AX=0000 BX=0000 CX=0000 DX=0061 SP=FFEE BP=0000 SI=0000 DI=0000

DS=0B76 ES=0B76 SS=0B76 CS=0B76 IP=1004 NV DN EI PL NZ NA PO CY

0B76:1004 CD21 INT 21

-R BX

BX 0000

:

-R CX

CX 0000

:20

-W DS:2000

Writing 00020 bytes

注释:写入的大小有BX,CX决定,BX为高字节,CX为低字节

## 装入命令

-L DS:4000

-D DS:4000 L30

0B76:4000 48 65 6C 6C 6F 20 77 6F-72 6C 64 48 65 6C 6C 6F Hello worldHello

0B76:4010 20 77 6F 72 6C 64 48 65-6C 6C 6F 20 77 6F 72 6C worldHello worl

0B76:4020 4C 03 C8 24 91 8D 02 43-44 03 80 2D 09 8D 05 43 L..$...CD..-...C

注释:如果是.exe文件或.com文件,L后不跟参数,exe装入CS:0000处,com装入CS:0100处

# 示例七:I/O端口的输入输出

## 输入命令

-I 219

FF

## 输出命令

-O 21B 82

# 示例八:十六进制加减

-H 45 36

007B 000F

段寄存器的隐含原则

存储器的操作类型 正常使用段 可使用的其他段 相对位移

取指令 CS 无 IP

堆栈操作 SS 无 SP

变量(以下三中情况除外) DS CS, ES, SS 有效地址EA

源数据串 DS CS, ES, SS SI

目的数据串 ES 无 DI

基址用BP的变量 SS CS, ES 有效地址EA