**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине Архитектура вычислительных систем

Системная архитектура процессоров IA-32. Сегментирование и обработка прерываний в защищенном режиме.

|  |  |  |
| --- | --- | --- |
| Студент  Группа АС-20-2 |  | Зверев А.Г. |
| Руководитель |  | Болдырихин О. В. |

Липецк 2022 г.

**Цель работы**

Изучение сегментирования и обработки прерываний в защищенном режиме процессоров IA-32

**Задание кафедры**

Написать на языке ассемблера программу, выполняющую ввод, обработку и вывод в защищенном режиме.

Как и в предыдущей работе программа должна включать подпрограммы. Хотя бы одна из подпрограмм должна быть дальней.

Главная программа выполняет необходимую инициализацию, в том числе готовит необходимые структуры данных: глобальную таблицу дескрипторов и таблицу дескрипторов прерываний – и переключает процессор в защищенный режим. Далее в защищенном режиме вызывается подпрограмма ввода с клавиатуры, которая последовательно заполняет массив исходных данных в памяти. Главная программа последовательно передает исходные данные подпрограмме преобразования через стек. Подпрограмма проверяет исходные данные на допустимость или корректность, при положительном исходе проверки осуществляет преобразование и возвращает результат главной программе. Главная программа размещает результаты в другой области памяти также в виде массива. Результаты выводятся на дисплей с помощью соответствующей подпрограммы.

Исследовать работу программы в обычном режиме, ввести исходные данные, получить результаты, сделать скриншот.

Исследовать работу программы в отладчике до загрузки регистра таблицы дескрипторов прерываний. По результатам исследования определить поля базы дескрипторов сегментов глобальной таблицы дескрипторов, значения регистров GDTR и IDTR.

По листингу определить моменты переключения процессора в защищенный и реальный режим, перехода в код защищенного режима и возврата в код реального режима.

Выполнить необходимые сравнения, сделать выводы.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Задача, выполняемая программой | Расположение GDT в программе | Расположение обработчика прерываний клавиатуры |
| 28 | Преобразование числа из упакованного двоично-десятичного кода | В сегменте кода защищенного режима | В отдельном сегменте |

**Листинг программы**

Листинг программы main.lst:

Turbo Assembler Version 4.1 12/10/22 12:27:34 Page 1

main.asm

7 0000 data segment

8 0000 00 bin db (?)

9 0001 00 bcd db (?)

10 0002 data ends

11

12 0000 InSeg segment

13 assume cs:InSeg

14 0000 inproc proc far

15 0000 8B EC mov bp, sp

16 0002 8B 76 04 mov si, [bp + 4]

17 0005 B9 0002 mov cx, 2

18 0008 EB 1B 90 jmp input

22 000B beep:

23 000B 51 push cx

26 000C B0 24 mov al, 24h

27 000E E6 42 out 42h, al

28 0010 B0 0E mov al, 0Eh

29 0012 E6 42 out 42h, al

31 0014 E4 61 in al, 61h

32 0016 0C 03 or al, 00000011b

33 0018 E6 61 out 61h, al

35 001A 33 C9 xor cx, cx

36 001C delay:

37 001C E2 FE loop delay

39 001E E4 61 in al, 61h

40 0020 24 FC and al, 11111100b

41 0022 E6 61 out 61h, al

42 0024 59 pop cx

46 0025 input:

47 0025 E4 60 in al, 60h

48 0027 3A C3 cmp al, bl

50

51 0029 74 FA je input

53

54 002B 8A D8 mov bl, al

56 002D 3C 01 cmp al, 01h

57 002F 75 05 jne not\_esc

Turbo Assembler Version 4.1 12/10/22 12:27:34 Page 2

main.asm

58 0031 B2 FF mov dl, -1

59 0033 EB 2B 90 jmp @quit

60 0036 not\_esc:

61 0036 3C 02 cmp al, 02h

62 0038 72 D1 jb beep

63 003A 3C 82 cmp al, 82h

64 003C 73 E7 jae input

65 003E 3C 0B cmp al, 0Bh

66 0040 77 C9 ja beep

67

68 0042 FE C8 dec al

69 0044 3C 0A cmp al, 0Ah

70 0046 75 02 jne not\_zero

71 0048 2C 0A sub al, 0Ah

72 004A not\_zero:

73 004A D0 E2 D0 E2 D0 E2 D0+ shl dl, 4

74 E2

75 0052 0A D0 or dl, al

76

77 0054 0D 0730 or ax, 0730h

78 0057 26: 89 05 mov es:[di], ax

79 005A 83 C7 02 add di,2

81 005D 49 dec cx

82 005E 75 C5 jnz input

83 0060 @quit:

84 0060 88 14 mov [si], dl

85 0062 CA 0002 retf 2

86 0065 inproc endp

87 0065 InSeg ends

89 0000 code segment

90 assume cs:code, ds:data

91 0000 main:

92 0000 BA 0000s mov dx, data

93 0003 8E DA mov ds, dx

97 0005 BA B800 mov dx, 0B800h

98 0008 8E C2 mov es, dx

99

100 000A 33 FF xor di, di

108 000C B8 0720 mov ax, 0720h

109 000F B9 07D0 mov cx, 80\*25

111 0012 F3> AB rep stosw

113

114

Turbo Assembler Version 4.1 12/10/22 12:27:34 Page 3

main.asm

115 0014 33 D2 xor dx, dx

116 0016 33 C0 xor ax, ax

117 0018 B3 9C mov bl, 9Ch

119 001A 33 FF xor di, di

120 001C B9 0002 mov cx, 2h

121

122 001F C6 06 0001r FF mov [bcd], -1

123 0024 50 55 8B EC C7 46 02+ push offset bcd

124 0001r 5D

125 002E 9A 00000000sr call inproc

126 0033 80 3E 0001r FF cmp [bcd], -1

127 0038 74 5E je quit

132 003A BB 000A mov bx, 10

133 003D 8A 16 0001r mov dl, [bcd]

134 0041 8A C2 mov al, dl

135 0043 24 F0 and al, 0F0h

136 0045 D0 E8 D0 E8 D0 E8 D0+ shr al, 4

137 E8

138 004D F6 E3 mul bl

139 004F 80 E2 0F and dl, 0Fh

140 0052 02 C2 add al, dl

141 0054 A2 0000r mov [bin], al

149 0057 BF 00A0 mov di, 80\*2

151 005A 8A 16 0000r mov dl, [bin]

152 005E B8 0700 mov ax, 0700h

153 0061 B9 0002 mov cx, 02d

154 0064 output:

155 0064 80 E2 F0 and dl, 0F0h

156 0067 D0 EA D0 EA D0 EA D0+ shr dl, 4

157 EA

158 006F 8A C2 mov al, dl

160 0071 3C 09 cmp al, 09d

161 0073 76 07 jbe next

162 0075 2C 0A sub al, 10

163 0077 04 41 add al, 'A'

164 0079 EB 03 90 jmp print

165 007C next:

166 007C 04 30 add al, '0'

167 007E print:

168 007E 26: 89 05 mov es:[di], ax

169 0081 83 C7 02 add di, 2

170 0084 8A 16 0000r mov dl, [bin]

171 0088 D0 E2 D0 E2 D0 E2 D0+ shl dl, 4

Turbo Assembler Version 4.1 12/10/22 12:27:34 Page 4

main.asm

172 E2

173 0090 E2 D2 loop output

174

178 0092 pause:

179 0092 E4 60 in al, 60h

181 0094 3C 01 cmp al, 01h

182 0096 75 FA jne pause

183 0098 quit:

184 0098 B8 4C00 mov ax, 4C00h

185 009B CD 21 int 21h

186 009D code ends

187 end main

Turbo Assembler Version 4.1 12/10/22 12:27:34 Page 5

Symbol Table

Symbol Name Type Value Cref (defined at #)

??DATE Text "11/27/22"

??FILENAME Text "main "

??TIME Text "12:27:34"

??VERSION Number 040A

@CPU Text 0101H

@CURSEG Text CODE #7 #12 #89

@FILENAME Text MAIN

@QUIT Near INSEG:0060 59 #83

@WORDSIZE Text 2 #7 #12 #89

BCD Byte DATA:0001 #9 122 123 126 133

BEEP Near INSEG:000B #22 62 66

BIN Byte DATA:0000 #8 141 151 170

DELAY Near INSEG:001C #36 37

INPROC Far INSEG:0000 #14 125

INPUT Near INSEG:0025 18 #46 51 64 82

MAIN Near CODE:0000 #91 187

NEXT Near CODE:007C 161 #165

NOT\_ESC Near INSEG:0036 57 #60

NOT\_ZERO Near INSEG:004A 70 #72

OUTPUT Near CODE:0064 #154 173

PAUSE Near CODE:0092 #178 182

PRINT Near CODE:007E 164 #167

QUIT Near CODE:0098 127 #183

Groups & Segments Bit Size Align Combine Class Cref (defined at #)

CODE 16 009D Para none #89 90

DATA 16 0002 Para none #7 90 92

INSEG 16 0065 Para none #12 13

Результаты исследования представлены в таблице 1.

1. Результаты исследования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер команды | Адрес команды | Команда на машинном языке | Команда на языке ассемблер | Cодержание изменившихся регистров и ячеек памяти |
| 1 | 0000 | BA7908 | Mov dx, 0879 | DX = 0879  IP = 0003 |
| 2 | 0003 | 8EDA | MOV DS, DX | DS = 0879 IP = 0005 |
| 3 | 0005 | BA00B8 | MOV DX, B800 | DX = B800  IP = 0008 |
| 4 | 0008 | 8EC2 | MOV ES, DX | ES = B800  IP = 000A |
| 5 | 000A | 33FF | XOR DI, DI | IP = 000C |
| 6 | 000C | B8207 | MOV AX, 0720 | AX = 0720  IP = 000F |
| 7 | 000F | B9D007 | MOV CX, 07D0 | CX = 07D0  IP = 0012 |
| 8 | 0012 | F3AB | REP STOSW | DI = 0FA0  CX = 0000  IP = 0014 |
| 9 | 0014 | 33D2 | XOR DX, DX | DX = 0000  IP = 0016 |
| 10 | 0016 | 33C0 | XOR AX, AX | DX = 0000  IP = 0018 |
| 11 | 0018 | B39C | MOV BL, 9C | BX = 009C  IP = 001A |
| 12 | 001A | 33FF | XOR DI, DI | DI = 0000  IP = 001C |
| 13 | 001C | B90200 | MOV CX, 0002 | CX = 0002  IP = 001F |
| 14 | 001F | EB1B | JMP 003C | IP = 003C |
| 15 (input) | 003C | E460 | IN AL, 60 | AX = 0042  IP = 003E |
| 16 | 003E | 3AC3 | CMP AL, BL | IP = 0040 |
| 17 | 0040 | 74FA | JE 003C | IP = 0042 |
| 18 | 0042 | BAD8 | MOV BL, AL | BX = 0042  IP = 0044 |
| 19 | 0044 | 3C01 | CMP AL, 01 | IP = 0046 |
| 20 | 0046 | 7503 | JNE 004B | IP = 004B |
| 21 | 004B | 3C02 | CMP AL, 02 | IP = 004D |
| 22 | 004D | 72D3 | JB 0022 | IP = 004F |
| 23 | 004F | 3C82 | CMP AL, 82 | IP = 0051 |
| 24 | 0051 | 73E9 | JNB 003C | IP = 0053 |
| 25 | 0053 | 3C0B | CMP AL, 0B | IP = 0055 |
| 26 | 0055 | 77CB | JA 0022 | IP = 0022 |
| 27 (beep) | 0022 | 51 | PUSH CX | SP = FFFE  IP = 0023 |
| 28 | 0023 | B024 | MOV AL, 24 | AX = 0024  IP = 0025 |
| 29 | 0025 | E642 | OUT 42, AL | IP = 0027 |
| 30 | 0027 | B00E | MOV AL, 0E | AX = 0003  IP =0029 |
| 31 | 0029 | E642 | OUT 42, AL | IP = 002B |
| 32 | 002B | E641 | IN AL, 61 | AX = 0000  IP = 002D |
| 33 | 002D | 0C03 | OR AL, 03 | AX = 0003  IP = 002F |
| 34 | 002F | E661 | OUT 61, AL | IP = 0031 |
| 35 | 0031 | 33C9 | XOR CX, CX | CX = 0000  IP = 0033 |
| 36 | 0033 | E2FE | LOOP 0033 | IP = 0035 |
| 37 | 0035 | E461 | IN AL, 61 | AX = 0001  IP = 0037 |
| 38 | 0037 | 24FC | AND AL, FC | AX = 0000  IP = 0039 |
| 39 | 0039 | E661 | OUT 61, AL | IP = 003B |
| 40 (ввод закончен) | 003B | 59 | POP CX | CX = 002  SP = 0000  IP = 003C |
| 41 | 0075 | 88160100 | MOV [0001], DL | AX = 0736  BX = 0007  DX = 0086  DI = 004  DS = 0879  ES = B800  SS = 0878  CS = 087A  IP = 0079 |
| 42 | 0079 | BB0A00 | MOV BX, 000A | BX = 000A  IP = 007C |
| 43 | 007C | 8A160100 | MOV DL, [0001] | IP = 0080 |
| 44 | 0080 | 8AC2 | MOV AL, DL | AX = 0786  IP = 0082 |
| 45 | 0082 | 24F0 | AND AL, F0 | AX = 0780  IP = 0084 |
| 46 | 0084 | D0E8 | SHR AL, 1 | AX = 0740  IP = 0086 |
| 47 | 0086 | D0E8 | SHR AL, 1 | AX = 0720  IP = 0088 |
| 48 | 0088 | D0E8 | SHR AL, 1 | AX = 0710  IP = 008A |
| 49 | 008A | D0E8 | SHR AL, 1 | AX = 0708  IP = 008C |
| 50 | 008C | F6E3 | MUL BL | AX = 0050  IP = 008E |
| 51 | 008E | 80E20F | AND DL, 0F | DX = 0006  IP = 0091 |
| 52 | 0091 | 02C2 | ADD AL, DL | AX = 0056  IP = 0093 |
| 53 | 0093 | A20000 | MOV [00000], AL | IP = 0096 |
| 54 | 0096 | BFA000 | MOV DI, 00A0 | DI = 00A0  IP = 0099 |
| 55 | 0099 | 8A160000 | MOV DL, [0000] | DX = 0056  IP = 009D |
| 56 | 009D | B80007 | MOV AX, 0700 | AX = 0700  IP = 00A0 |
| 57 | 00A0 | B90200 | MOV CX, 0002 | CX = 0002  IP = 00A3 |
| 58 (output) | 00A3 | 80EE2F0 | AND DL, F0 | DX = 0050  IP = 00A6 |
| 59 | 00A6 | D0EA | SHR DL, 1 |  |
| 60 | 00A8 | D0EA | SHR DL, 1 |  |
| 61 | 00AA | D0EA | SHR DL, 1 | DX = 000A  IP = 00AC |
| 62 | 00AC | D0EA | SHR DL, 1 | DX = 0005  IP = 00AE |
| 63 | 00AE | 8AC2 | MOV AL, DL | AX = 0705  IP = 00B0 |
| 64 | 00B0 | 3C09 | CMP AL, 09 | IP = 00B2 |
| 65 | 00B2 | 7607 | JBE 00BB | IP = 00BB |
| 66 | 00BB | 0430 | ADD AL, 30 | AX = 0735  IP = 00BD |
| 67 (print) | 00BD | 268905 | MOV ES:[DI], AX | IP = 00C0 |
| 68 | 00C0 | 83C702 | ADD DI, 0002 | DI = 00A2  IP = 00C3 |
| 69 | 00C3 | 8A160000 | MOV DL, [0000] | DX = 0056  IP = 00C7 |
| 70 | 00C7 | D0EA | SHR DL, 1 | DX = 00AC  IP = 00C9 |
| 71 | 00C9 | D0EA | SHR DL, 1 | DX = 0058  IP = 00CB |
| 72 | 00CB | D0EA | SHR DL, 1 | DX = 00B0  IP = 00CD |
| 73 | 00CD | D0EA | SHR DL, 1 | DX = 0060  IP = 00CF |
| 74 (вывод закончен) | 00CF | E2D2 | LOOP 00A3 | CX = 0001  IP = 00A3 |
| 75 | 00DF | B8004C | MOV AX, 4C00 | AX = 4C00  IP = 00DA |
| 76 | 00DA | CD21 | INT 21 |  |

**Вывод**

При выполнении лабораторной работы я написал на языке ассемблера программу, выполняющую преобразование числа из упакованного двоично-десятичного кода в защищенном режиме. Исходные данные вводятся в память с клавиатуры, результаты выводятся на дисплей в отдельном сегменте. Вывод результатов на экран выполняет дальняя подпрограмма.

В результате выполнения лабораторной работы я узнала следующее:

Надежность системы, функционирующей в защищенном режиме, обусловлена следующим:

1) возможность задания необходимого размера сегмента и контролем адресации памяти вне пределов сегментов.

2) байт доступа дескриптора сегмента кода содержит бит разрешения чтения сегмента (бит 1). Если этот бит установлен в 1, программа может считывать содержимое сегмента кода. В противном случае процессор может только выполнять этот код, т. е. программа не может модифицировать сегмент кода. Это означает невозможность создания самомодифицирующихся программ для защищенного режима.