

بسمه تعالی



گزارش کار آزمایش ۱ ریزپردازنده و اسمبلی

بخش A

غزاله طالبیان، ۹۸۲۴۳۰۳۶

محسن کربلائی امینی، ۹۸۲۴۲۱۲۸

۶ اسفند ۱۴۰۱

بخش ۱۴:

برای محاسبه مربعات هر یک از اعداد از رجیستر AX استفاده شده به این صورت که ۸ بیت هر عدد داخل AL ذخیره و سپس با استفاده از دستور زیر مربع عدد در خود AL ذخیره می‌شود.

$MUL\ AL$

از آنجایی که مربع هر عدد n بیتی یک عدد $2n$ بیتی خواهد بود، حاصل در متغیر $answer$ ذخیره می‌شود که از نوع $word$ تعریف شده‌است.

برای انجام مقایسه بین مربعات نیازی به MOV نمودن حاصل مربع دوم نیست و AL و $answer$ مستقیماً به عنوان مربعات اعداد باهم مقایسه می‌شوند که در صورت منفی شدن مقایسه مقدار \bullet در $answer$ ذخیره خواهد شد.

کد و نتیجه شبیه‌سازی:

The image shows two screenshots of a debugger window, likely Visual Studio, displaying assembly code and variable values. The top screenshot shows the initial state of the program, and the bottom screenshot shows the state after some execution.

Top Screenshot:

```
01 .model small
02
03 .stack 64
04
05 .data
06 data1 db 3
07 data2 db 4
08 answer dw ?
09
10 .code
11
12 main proc far
13 mov ax,@data
14 mov ds,ax
```

Variables Panel (Top):

Variable	Value
DATA1	03h
DATA2	04h
ANSWER	0000h

Bottom Screenshot:

```
01 .model small
02
03 .stack 64
04
05 .data
06 data1 db 4
07 data2 db 3
08 answer dw ?
09
10 .code
11
12 main proc far
13 mov ax,@data
```

Variables Panel (Bottom):

Variable	Value
DATA1	04h
DATA2	03h
ANSWER	0007h

```

1 .model small
2
3 .stack 64
4
5
6
7 data1 db 3
8 data2 db 4
9 answer dw ?
10
11 .data
12 data1 db 3
13 data2 db 4
14 answer dw ?
15
16 .code
17
18 main proc far
19     mov ax,@data
20     mov ds,ax
21
22
23     mov al,data1
24     mul al
25
26     mov answer,ax
27
28
29     mov al,data2
30     mul al
31
32     sub answer,ax
33
34
35     cmp answer,0
36     jg end
37
38     mov answer,0
39
40
41     end:
42     mov ah,4ch
43     int 21h
44
45 main endp
46
47 end main
48
49 ~
50 ~

```

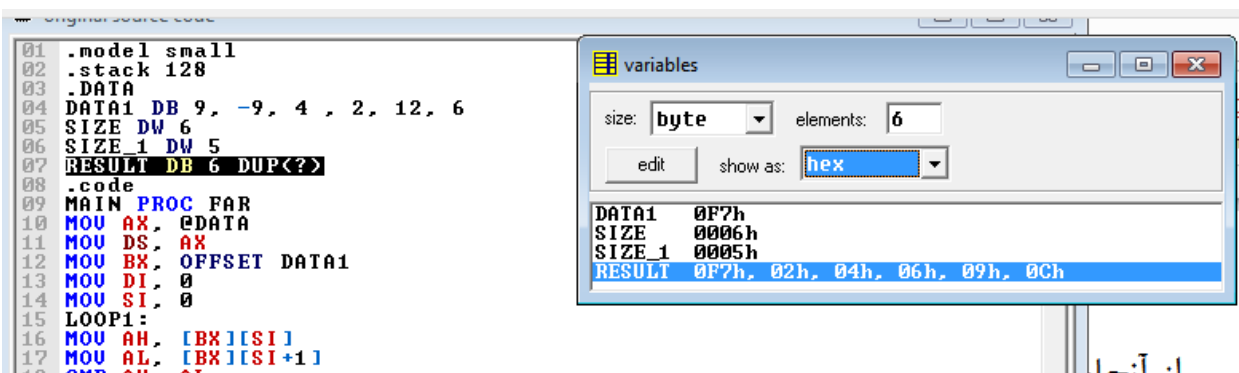
بخش A ۲:

با استفاده از الگوریتم مرتب سازی *Bubble sort* اعداد وارد شده به صورت صعودی مرتب و در حافظه *result* ذخیره می‌شوند.

در این بخش از ۳ حلقه استفاده شده است. ۲ حلقه اول که در واقع هر دو ذیل *LOOP ۱* تعریف شده‌اند و وظیفه مرتب سازی ورودی‌ها و ذخیره آنها در *BX* را دارند و دو شمارنده *SI* و *DI* شروط ورود به حلقه را تعریف میکنند.

حلقه سوم وظیفه انتقال داده‌های مرتب شده از *BX* به *result* را بر عهده دارد.

کد و نتیجه شبیه‌سازی:



```
01 .model small
02 .stack 128
03 .DATA
04 DATA1 DB 9, -9, 4, 2, 12, 6
05 SIZE DW 6
06 SIZE_1 DW 5
07 RESULT DB 6 DUP{?}
08 .code
09 MAIN PROC FAR
10 MOV AX, @DATA
11 MOV DS, AX
12 MOV BX, OFFSET DATA1
13 MOV DI, 0
14 MOV SI, 0
15 LOOP1:
16 MOV AH, [BX][SI]
17 MOV AL, [BX][SI+1]
18 CMP AH, AL
```

variables

size: byte elements: 6

edit show as: hex

DATA1	0F7h
SIZE	0006h
SIZE_1	0005h
RESULT	0F7h, 02h, 04h, 06h, 09h, 0Ch

```

.model small
.stack 128
.DATA
    DATA1 DB 9, -9, 4, 2, 12, 6
    SIZE DW 6
    SIZE_1 DW 5
    RESULT DB 6 DUP(?)
.code
10 MAIN PROC FAR
11     MOV AX, @DATA
12     MOV DS, AX
13     MOV BX, OFFSET DATA1
14     MOV DI, 0
15     MOV SI, 0
16     LOOP1:
17         MOV AH, [BX][SI]
18         MOV AL, [BX][SI+1]
19         CMP AH, AL
20         JLE NOSWAP
21         JMP SWAP
22     NOSWAP:
23         MOV [BX][SI], AH
24         MOV [BX][SI+1], AL
25         ;MOV AH,AL ; omit the minimum to make place for the new value to compare
26         JMP CONTINUE_1
27     SWAP:
28         MOV [BX][SI], AL
29         MOV [BX][SI+1], AH
30         JMP CONTINUE_1
31     CONTINUE_1:
32     INC SI
33     CMP SI, SIZE_1
34     JL LOOP1
35     ; MOV AL, [BX][SI+1]
36     INC DI
37     MOV SI, 0
38     CMP DI, SIZE_1
39     JLE LOOP1
40     CALCULATE:
41     MOV DI, 0
42     LOOP2: ; calculate final result
43         CMP DI, SIZE_1
44         JZ END
45         MOV AX, [BX][DI]
46         MOV BX, OFFSET RESULT
47         MOV [BX][DI], AX
48         MOV BX, OFFSET DATA1
49         INC DI
50         JMP LOOP2
51     END:
52     MOV AH, 4CH
53     INT 21H
54     ENDP
55     END MAIN

```