به نام خدا

تمرین سوم درس ریزپردازنده

غزل زمانی‌نژاد

۹۷۵۲۲۱۶۶

1. **ابتدا در بخش** data segment **یک آرایه از نمرات نگهداری میکنیم. سپس سایر متغیرهایی که در محاسبات به آنها نیاز داریم (مثل میانه، بیشترین نمره و ...) را تعریف میکنیم.**

```
;place data definitions here
GRADES  DB  15H,13H,09H,08H,16H,06H,11H,17H,18H,04H,02H,10H,12H,17H,10H,15H,13H,09H,08H,16H ;all numbers are packed BCD
LENGTH  DW  14H

MEDIAN  DB  ?
MAX     DB  ?
MAXSCORE    DB  20H
DIFF      DB  ?


HEXSUM1     DW  ?
HEXSUM2     DW  ?
AVG1    DB  ?
AVG2    DB  ?
BCDAVG1 DW  ?
BCDAVG2 DW  ?
```

در بخش code segment **تابع** main **را تعریف میکنیم. در آن** source index **را بر روی شروع آرایه نمرات قرار میدهیم. سپس** sum1 **را صدا میزنیم تا مجموع نمرات محاسبه شود.** cal_avg1 **را صدا میزنیم تا میانگین نمرات محاسبه شود. و با صدا زدن** bcd_avg1 **میانگین را به** packed bcd **تبدیل میکنیم.**

برای محاسبه کردن میانه ابتدا باید آرایه را سورت کنیم. پس از پیدا کردن میانه، با صدا کردن cal_max **بیشترین مقدار موجود در نمرات را می یابیم. سپس** cal_maxscore **را صدا میزنیم تا مقدار نمره ای که در نمودار به افراد اضافه میشود را بیابیم. بعد از پیدا کردن این مقدار، با صدا زدن** grade_increase **نمرات را شیفت میدهیم. در نهایت دوباره مجموع اعداد آرایه را حساب میکنیم و به کمک آن میانگین نمرات (اول به** hexadecimal **و بعد به** packed bcd**) حساب میکنیم.**

```
.CODE
MAIN    PROC    FAR
        MOV     AX,@DATA
        MOV     DS,AX
        ;
        ;place code here
        MOV     SI,OFFSET GRADES ;put offset of array in source index pointer
        CALL    SUM1
        MOV     LENGTH,14H
        CALL    CAL_AVG1
        CALL    BCD_AVG1

        CALL    SORT
        CALL    FIND_MEDIAN

        CALL    CAL_MAX
        CALL    CAL_MAXSCORE
        CALL    GRADE_INCREASE

        CALL    SUM2
        MOV     LENGTH,14H
        CALL    CAL_AVG2
        CALL    BCD_AVG2

        ;

        MOV     AH,4CH
        INT     21H
MAIN    ENDP
```

sort: برای مرتب کردن آرایه استفاده میشود. از دو حلقه تو در تو برای پیاده سازی bubble sort استفاده میکنیم. در حلقه بیرونی یک عدد را انتخاب میکنیم و آن را با تمامی اعداد بعد از خودش در حلقه دوم مقایسه میکنیم. در صورتی که مقدار آن بیشتر باشد، به لیبل change می رویم و جای دو عدد را عوض میکنیم. این کار را تا زمانی ادامه میدهیم که تمامی خانه های آرایه را در حلقه بیرونی دیده باشیم.

```
;------------------------------ SORT--
SORT PROC
     MOV    CX,LENGTH

OUTERLOOP:  MOV SI,0H
            MOV DI,0H
            INC DI
            JMP INNERLOOP
            RET

INNERLOOP:    MOV   AL,[SI]
              MOV   BL,[DI]
              CMP   AL,BL
              JA    CHANGE
              INC   DI
              INC   SI
              CMP   DI,CX
              JL    INNERLOOP
              LOOP OUTERLOOP
              RET

CHANGE: MOV [SI],BL
        MOV [DI],AL
        MOV AL,BL
        INC   DI
        INC   SI
        CMP   DI,CX
        JL    INNERLOOP
        LOOP OUTERLOOP
        RET

SORT    ENDP
```

Find_median: آفست شروع آرایه نمرات را در یک رجیستر ذخیره میکنیم. میدانیم در یک آرایه 20تایی، میانه برابر با نصف مجموع خانه نهم و دهم (در آرایه مرتب شده) است. پس پوینتر به ابتدای آرایه را به میزان 9خانه جلو میبریم. عدد آن خانه را پیدا میکنیم. سپس عدد خانه دهم را پیدا میکنیم. با هم جمع کرده و بر 2 تقسیم میکنیم. خارج قسمت

تقسیم که در al ذخیره میشود را در median می ریزیم. (چون این دو مقدار bcd هستند از دستور daa استفاده میکنیم تا به صورت خودکار مجموع شان تصحیح شود)

```
;------------------------------ MEDIAN
FIND_MEDIAN PROC
            MOV BX,OFFSET GRADES
            ADD BX,09H
            MOV AX,WORD PTR [BX]
            ADD BX,01H
            ADD AX,WORD PTR [BX]
            DAA
            MOV AH,0H
            MOV BL,02H
            DIV BL
            MOV MEDIAN,AL
            RET

FIND_MEDIAN ENDP
```

Cal_max: پوینتر را به ابتدای آرایه اشاره میدهیم. در یک لوپ بررسی میکنیم که آیا عدد خانه فعلی آرایه از بیشترین مقدار تا کنون بیشتر است یا خیر. اگر بیشتر باشد مقدارش را در max میریزیم.

```
;------------------------------ MAX --
CAL_MAX PROC
        MOV     SI,0H    ;set source p
        MOV     CX,LENGTH  ;loop cond

        MOV     MAX,0H

LOOP2:  MOV     AL,[SI]
        CMP     AL,MAX
        JA      GREATER
RTN_POINT:      INC     SI
        LOOP    LOOP2
        RET

GREATER:MOV     MAX,AL
        JMP     RTN_POINT

CAL_MAX ENDP
```

Cal_maxscore: چون دو عددی که میخواهیم از هم کم کنیم bcd هستند، نمیتوانیم آنها را به صورت معمول تفریق کنیم. پس در یک لوپ، تک تک رقم های عدد max را از رقم متناظر آن در 20 کم میکنیم. دستور das را اجرا میکنیم. نتیجه را در diff میریزیم.

```
;------------------------------ MAXSCORE --
;compute the difference between 20 and ma
CAL_MAXSCORE PROC
             MOV CX,01
             MOV BX,00
             CLC
BACK:        MOV AL,BYTE PTR MAXSCORE[BX]
             SBB AL,BYTE PTR MAX[BX]
             DAS
             MOV BYTE PTR DIFF[BX],AL
             INC BX
             LOOP BACK
             RET
CAL_MAXSCORE ENDP
```

Grade_increase: در یک لوپ بررسی میکنیم که آیا عدد موجود در آرایه از میانه بیشتر است یا خیر (با دستور cmp). در صورتی که بیشتر باشد، شیفت میخورد. پس به لیبل increase می رویم. آن را با مقدار diff که از قبل حساب کرده بودیم جمع میکنیم.

```
;------------------------------ GRADE_INCREASE -
GRADE_INCREASE  PROC
                MOV SI,0H
                MOV CX,LENGTH

LOOP3:          MOV AL,[SI]
                CMP AL,MEDIAN
                JA  INCREASE
ITERATE:        INC SI
                LOOP LOOP3
                RET

INCREASE:       ADD AL,DIFF
                DAA
                MOV BYTE PTR GRADES[SI],AL
                JMP ITERATE

GRADE_INCREASE  ENDP
```

Sum: برای محاسبه مجموع اعداد، در یک حلقه ابتدا آنها را از bcd به hexadecimal تبدیل میکنیم. سپس مقدارش را به hexsum اضافه میکنیم. متغیر hexsum را به صورت word تعریف میکنیم تا بیت carry از دست نرود.

```
;-------------------    BCD SUM TO HEX -
SUM1    PROC

        MOV SI,0H
        MOV AH,0H

LOOPSUM1:
        MOV BL,[SI]
        AND BL,0FH
        MOV AL,[SI]
        AND AL,11110000B
        MOV CL,04
        ROR AL,CL
        MOV DL,0AH
        MUL DL
        ADD AL,BL

        ADD HEXSUM1,AX
        INC SI
        DEC LENGTH
        CMP LENGTH,0H
        JNZ LOOPSUM1

        RET

SUM1    ENDP
```

Avg: برای محاسبه میانگین، مجموع را بر طول آرایه تقسیم میکنیم. نتیجه را از al داخل avg می ریزیم.

```
;------------------------------  AVG1 -
CAL_AVG1    PROC
            MOV AX,HEXSUM1
            XOR DX,DX
            DIV LENGTH
            MOV AVG1,AL
            RET
CAL_AVG1    ENDP
```

Bcd_avg: در پایان باید اعدادی که محاسبات آن را به hex انجام داده بودیم به bcd برگردانیم.

```
;------------------------------ BCD_AVG1
BCD_AVG1     PROC
             MOV AL,AVG1
             MOV AH,0H
             MOV CL,64H
             DIV CL
             MOV BH,AL
             MOV AL,AH
             MOV AH,0H
             MOV CL,0AH
             DIV CL
             MOV CL,04H
             ROL AL,CL
             ADD AL,AH
             MOV BL,AL
             MOV BCDAVG1,BX
             RET
BCD_AVG1     ENDP
```

نتیجه پس از اجرا:

سطر اول، آرایه مرتب شده. سطر سوم میانه.

سطر دوم از پایین میانگین نمرات قبل از شیفت. سطر اول از پایین میانگین نمرات پس از شیفت.

```
GRADES  02h, 04h, 06h, 08h, 08h, 09h, 09h, 10h, 10h, 11h, 14h, 15h, 15h, 17h, 17h, 18h, 18h, 19h, 19h, 20h
LENGTH  0014h
MEDIAN  11h
MAX     18h
MAXSCORE        20h
DIFF    02h
HEXSUM1 00E5h
HEXSUM2 00F9h
AVG1    0Bh
AVG2    0Ch
BCDAVG1 0011h
BCDAVG2 0012h
```

2. یک آرایه از اسامی افراد تشکیل میدهیم. برای اینکه انتهای هر رشته مشخص باشد، آخر هر اسم null (دارای کد اسکی 0) اضافه میکنیم. و همچنین یک آرایه برای نگهداری تعداد تکرار هر اسم میسازیم.

```
;--------------------------------
 DATA
COUNT DB 10 DUP (?)
NAMES DB    "SA",0H,"UR",0H,"AB",0H,"GH",0H,"ABC",0H,"AB",0H,  "SA",0H,"OP",0H,"SA",0H,"XV",0H ;strings are all null terminated
LENGTH  DB  0AH

TMP     DW  ?
```

در بخش main ابتدا di را به ابتدای آرایه count اشاره میدهیم. سپس si را به ابتدای آرایه اسامی اشاره میدهیم. در حلقه بیرونی در CX آدرس ابتدای رشته اولی که میخواهیم آن را با سایر رشته ها مقایسه کنیم میریزیم. در DI آدرس ابتدای رشته دوم را میریزیم. در لوپ درونی باید تک تک کاراکترهای دو رشته مد نظر را با هم مقایسه کنیم. برای این کار CHECK را صدا میزنیم. پس از بررسی یکسان بودن دو رشته، باید DI را جلو ببریم تا رشته اول را با سایر رشته ها

مقایسه کنیم. وقتی این مقایسه تمام شد، به لوپ خارجی میرویم و نام بعدی را برای مقایسه انتخاب میکنیم. SI را حلو میبریم زمانی که به رشته های بعدی برسیم (تا زمانی که کاراکتر NULL را ندیده ایم این کار را انجام میدهیم).

```
;-------------------------------
.CODE
MAIN    PROC    FAR
        MOV     AX,@DATA
        MOV     DS,AX
        ;
        ;place code here
        MOV     AH,LENGTH
        MOV     SI,OFFSET NAMES
        MOV     DI,OFFSET COUNT
        MOV     DX,DI
        DEC     DX

OUTERLOOP:      MOV CX,SI
                MOV DI,OFFSET NAMES
                INC DX

INNERLOOP:      MOV SI,CX
                CALL CHECK
                INC DI
                CMP [DI],0H
                JNZ INNERLOOP

                INC SI
                DEC AH
                CMP AH,0
                JNZ OUTERLOOP


        MOV     AH,4CH
        INT     21H
MAIN    ENDP
```

CHECK: برای مقایسه دو رشته در یک حلقه تک تک کاراکترهایشان را مقایسه میکنیم. با دستور LODSB جایی که SI اشاره میکند را در AL لود میکنیم و SI را یکی زیاد میکنیم. بعد دو کاراکتر را باهم مقایسه میکنیم. اگر برابر نباشند به لیبل DIFFERENT می رویم. اگر برابر بودند، ابتدا چک میکنیم که آیا به انتهای نام رسیده ایم یا خیر. اگر نرسیده باشیم به لیبل LABEL1 برمی گردیم تا سایر کاراکترها را نیز بررسی کنیم. اما اگر به انتهای رشته رسیده باشیم، یعنی دو رشته باهم برابر بوده اند. پس باید در آرایه COUNT یکی به مقدار آن نام اضافه کنیم. در دو لیبل ENDOFSTR نیز باید مقدار SI و DI را زیاد کنیم تا به انتهای هر دو رشته برسیم.

```
;-------------------------------------
CHECK PROC
        DEC DI
LABEL1: INC DI
        LODSB   ;automatically increases si
        CMP [DI],AL
        JNE DIFFERENT

        CMP AL,0H   ;check whether reached end of string
        JNE LABEL1  ;loop, till the end of the string

        MOV TMP,SI
        MOV SI,DX
        MOV BL,BYTE PTR COUNT[SI]
        INC BL
        MOV BYTE PTR COUNT[SI],BL
        MOV DX,SI
        MOV SI,TMP
        RET

DIFFERENT:  DEC SI

ENDOFSTR1:  INC SI
            CMP [SI], 0H
            JNZ ENDOFSTR1
            DEC DI  ;if reached to the last character,decreament di

ENDOFSTR2:  INC DI
            CMP [DI],0H
            JNZ ENDOFSTR2
            RET

CHECK ENDP
```

خروجی پس از یک بار اجرا کردن:

```
COUNT   03h, 01h, 02h, 01h, 01h, 02h, 03h, 01h, 03h, 01h
NAME1   53h
NAME2   55h
NAME3   41h
NAME4   47h
NAME5   41h
NAME6   41h
NAME7   53h
NAME8   4Fh
NAME9   53h
NAME10  58h
TMP     0029h
```