به نام خدا

تمرین پنجم درس ریزپردازنده

غزل زمانی‌نژاد

۹۷۵۲۲۱۶۶

1. برای پیاده سازی ماکروها، از کتاب درسی صفحه 163 استفاده شده است. و برای پیاده سازی ماکرو رنگ از اسلایدهای درسی استفاده شده است. بررسی ماکروها:

Cursor: دو عدد سطر و ستون به عنوان ورودی دریافت میکند و نشانگر ماوس را به آن نقطه میبرد.

Display: یک رشته به عنوان ورودی دریافت میکند و آن را در صفحه چاپ میکند.

Show_dot: یک رنگ را دریافت میکند و یک نقطه با آن رنگ را در صفحه چاپ میکند.

```
CURSOR  MACRO ROW,COLUMN
        MOV AH,02H
        MOV BH,00
        MOV DH,ROW
        MOV DL,COLUMN
        INT 10H
        ENDM

DISPLAY MACRO STRING
        MOV AH,09H
        MOV DX,OFFSET STRING ;load string address
        INT 21H
        ENDM

SHOW_DOT   MACRO   COL
        MOV     AH, 09  ;display option
        MOV     BH, 00  ;page 0
        MOV     AL, 09  ;dot ascii code is 9
        MOV     CX, 1   ;repeat
        MOV     BL, COL ;set favorite color to cga
        SHL     BL, 1   ;shift to next color
        ENDM
```

Clear: اطلاعات موجود در صفحه نمایشگر را پاک میکند.

Convert: برای تبدیل اعداد و نمایش آنها در صفحه استفاده میشود.

```
CLEAR   MACRO
        MOV AX,0600H ;scroll the screen
        MOV BH,07    ;normal attribute
        MOV CX,0000  ;from row=00, column=00
        MOV DX,184FH ;to row=18h, column=4fh
        INT 10H      ;invoke interrupt to change mode
CLEAR   ENDM

CONVERT MACRO
        SHR AX,3
        MOV TMP,10
        SUB AH,AH
        DIV TMP
        OR  AX,3030H

CONVERT ENDM
```

در قسمت data segment متغیرهایی که در ادامه برنامه به آنها نیاز داریم را تعریف میکنیم. این متغیرها شامل: دکمه ریست برای پاک کردن نقاط صفحه، رشته mouse position برای چاپ کردن در صفحه، x و y برای ذخیره مختصات نشانگر ماوس، dot_color برای رنگ نقطه ای که چاپ خواهد شد و یک مقدار tmp است.

```
.MODEL  SMALL
.STACK  64
;-------------------------
.DATA

RESET       DB   '*** R E S E T *** ','$'
MOUSE_POS   DB   'MOUSE POSITION: ','$'

X       DB   ?,?,',','$'
Y       DB   ?,?,'$','$'

DOT_COLOR    DB   20H
TMP     DB   00H
```

در قسمت code segment، تابع main عملیات اصلی را انجام میدهد.

ابتدا رنگ نقطه را بر روی قرمز تنظیم میکنیم، صفحه را ریست میکنیم و دکمه ریست را در وسط صفحه نمایش میدهیم. سپس mouse initialization انجام میدهیم. همانطوری که میدانیم در صورتی که کاربر از ماوس استفاده کند، وضعیت آن در bx ذخیره میشود. پس اگر مقدار آن برابر با یک باشد، یعنی کاربر در صفحه کلیک کرده، در غیر این صورت به لیبل initial برمیگردیم و منتظر میمانیم تا کاربر کلیک کند.

```
.CODE
MAIN            PROC    FAR
                MOV     AX, @DATA
                MOV     DS, AX

                MOV     AH, 00
                MOV     AL, 03H
                INT     10H
SHOW_RESET:     MOV     DOT_COLOR,20H ;reset the color
                CURSOR  12,32
                DISPLAY RESET

INITIAL:        MOV     AX, 03  ;mouse initialization
                INT     33H

                CMP     BX, 1   ;BX contains mouse button status
                JZ      POSITION
                JMP     INITIAL
```

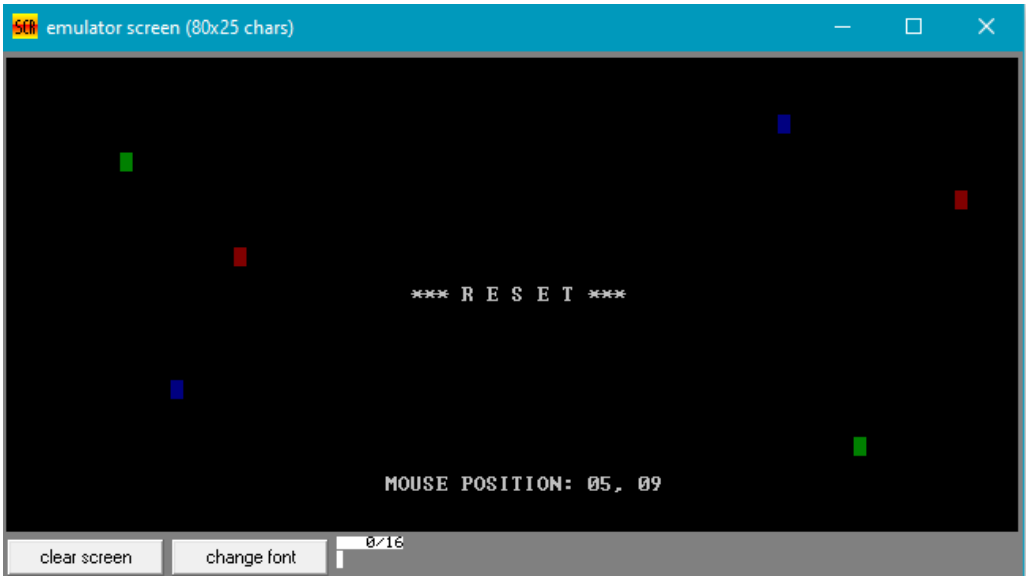
بعد از اینکه کاربر کلیک کرد، مختصات نقطه ای که ماوس به آن اشاره میکرده را از cx و bx تقسیم بر 8 میکنیم. سپس با استفاده از مختصات چک میکنیم که آیا کاربر بر روی دکمه ریست کلیک کرده یا خیر. برای این کار مختصات را با مختصات دکمه چک میکنیم، در صورت برابری به لیبل show_reset میرویم.

```
POSITION:   SHR     CX, 3   ;cx/8 contains horizontal coordinate
            SHR     DX, 3   ;dx/8 contains vertical coordinate


;IF  DX=12 AND 32<CX<48
;THEN CLEAR THE SCREEN AND RESET
ON_RESET_CLK:   CMP     DX,12
                JNE     MOUSE_CLK

                CMP     CX,32
                JB      MOUSE_CLK
                CMP     CX,48
                JA      MOUSE_CLK
                CLEAR

                JMP     SHOW_RESET
```

در غیر این صورت رنگ مربوطه با استفاده از یک حلقه از میان 3 رنگ قرمز و آبی و سبز تنظیم میکنیم. و با استفاده از ماکرو show_dot نقطه را در صفحه نمایش میدهیم. همچنین با استفاده از ماکرو display، مختصات نقطه را در صفحه چاپ میکنیم.

```
MOUSE_CLK:   CURSOR  DL,CL
             SHOW_DOT DOT_COLOR
             CMP     BL, 40H    ;compare the color with red
             JNA     FIND_COLOR ;if the color is in the range print the position
             MOV     BL,10H

FIND_COLOR:  MOV     DOT_COLOR, BL
             INT     10H

             MOV     AX, 03
             INT     33H

             MOV     AX,DX
             CONVERT
             MOV     X,AL
             MOV     X+1,AH

             MOV     AX,CX
             CONVERT
             MOV     Y,AL
             MOV     Y+1,AH

             CURSOR  22,30
             DISPLAY MOUSE_POS
             DISPLAY X
             DISPLAY Y

             JMP     INITIAL

             MOV     AH, 4CH
             INT     21H
MAIN         ENDP

             END MAIN
```

خروجی:



در صورت کلیک بر روی دکمه reset **در وسط صفحه، تمامی نقاط پاک میشوند و رنگ روی قرمز ست میشود.**

2. برای پیاده سازی برج هانوی، از راه حل ویکی پدیا استفاده میکنیم. طبق این راه، باید به تعداد دیسک هایی که داریم رقم برای دانستن وضعیت هر دیسک داشته باشیم. باید در هر مرحله شماره دیسکی که باید جا به جا شود، برجی که از آن جا به جا میشود و برجی که به آن میرود را محاسبه کنیم.

دو ماکرو display و convert را مطابق سوال قبل داریم.

در ماکرو movement محاسبات اصلی را انجام میدهیم. در قسمت اول باید شماره دیسک را محاسبه کنیم. برای اینکار باید چک کنیم عدد m چند بار بر 2 قابل تقسیم است. در یک لوپ آن را مرتب به 2 تقسیم میکنیم و مقدار آن را در AX ذخیره میکنیم. سپس آن را CONVERT میکنیم و در متغیر DISK_NUM می ریزیم. و به همراه یک پیام در صفحه نمایش میدهیم.

برای محاسبه شماره برج ها از فرمول های زیر استفاده میکنیم.

peg `(m & m - 1) % 3` to peg `((m | m - 1) + 1) % 3`

در قسمت دوم محاسبات مربوط به برج مبدا را انجام میدهیم. باید توجه داشته باشیم در تقسیم 16 بیتی، باقی مانده تقسیم در DX ذخیره میشود. سپس آن را در T1 می ریزیم و در صفحه نمایش می دهیم.

در قسمت سوم نیز مشابه قسمت دوم عمل میکنیم. محاسبات را انجام داده، در T3 می ریزیم و در صفحه نمایش می دهیم.

```
MOVEMENT    MACRO M
            LOCAL DISK_ID
            ;calculate number of times (m) can be divided by 2
            MOV TMP,2
            MOV AX,M
            MOV CX,0
DISK_ID:    INC CX
            MOV DX, 0
            DIV TMP
            CMP DX, 0
            JE  DISK_ID
            MOV AX, CX

            CONVERT
            MOV DISK_NUM, AL
            MOV DISK_NUM+1,AH
            DISPLAY MSG1
            DISPLAY DISK_NUM

            ;(m&m-1) % 3
            MOV TMP,M
            DEC TMP
            AND TMP,M
            MOV AX,TMP
            MOV CX,3
            MOV DX,0    ;in 16bit mode, remainder is stored in DX
            DIV CX
            OR  DL,30H
            MOV T1,DL
            DISPLAY MSG2
            DISPLAY T1

            ;(m|m-1)+1 % 3
            MOV TMP,M
            DEC TMP
            OR  TMP,M
            INC TMP
            MOV AX,TMP
            MOV CX,3
            MOV DX,0    ;in 16bit mode, remainder is stored in DX
            DIV CX
            OR  DL,30H
            MOV T3,DL
            DISPLAY MSG3
            DISPLAY T3
            DISPLAY NEW_LINE
            ENDM
```

در ماکرو HANOI، ماکرو قسمت قبل را به دفعات نیاز صدا میزنیم.

```
HANOI   MACRO
        LOCAL LOOP1
        LOCAL FINISH
        MOV BX, 1
LOOP1:  CMP BX, CALL_COUNT
        JE  FINISH
        MOVEMENT BX
        INC BX
        JMP LOOP1

FINISH:
        ENDM
```

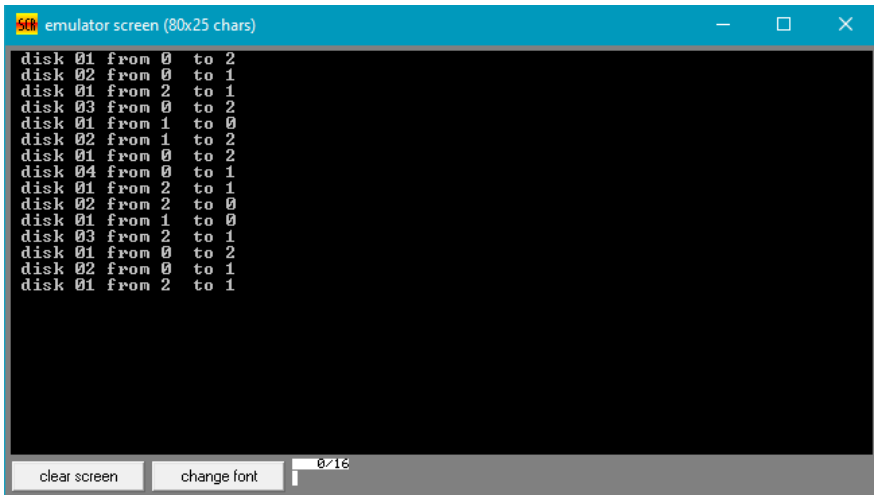
در data segment متغیرهای مورد نیاز را تعریف میکنیم. این متغیرها شامل: تعدادی پیام برای نمایش در صفحه، شماره برج مبدا و مقصد، شماره دیسک فعلی، و یک مقدار tmp است.

```
.MODEL SMALL
.STACK 64

.DATA

MSG1 DB ' disk ','$'
MSG2 DB ' from ','$'
MSG3 DB ' to ','$'
NEW_LINE DB 13,10, "$"

T1 DB ?,?,'$'
T3 DB ?,?,'$'
DISK_NUM DB ?,?,'$'
CALL_COUNT DW 0
TMP  DW 0
```

در code segment باید ابتدا یک عدد n بیتی بسازیم. برای این کار از تعداد دیسک ها یکی کم میکنیم و در cx می ریزیم. سپس در یک حلقه 2 را به توان n می رسانیم. و در نهایت ماکرو Hanoi را استفاده میکنیم.

```
;-------------------------
.CODE
MAIN    PROC FAR
        MOV AX,@DATA
        MOV DS,AX

        MOV AH,00
        MOV AL,03
        INT 10H

        ;CX = DISKS - 1
        MOV AX,2
        MOV BX,2
        MOV CX,3
BITS_NEEDE: MUL BX
            LOOP BITS_NEEDE

        MOV CALL_COUNT,AX
        HANOI

        MOV AH,4CH
        INT 21H
        MAIN ENDP

END MAIN
```

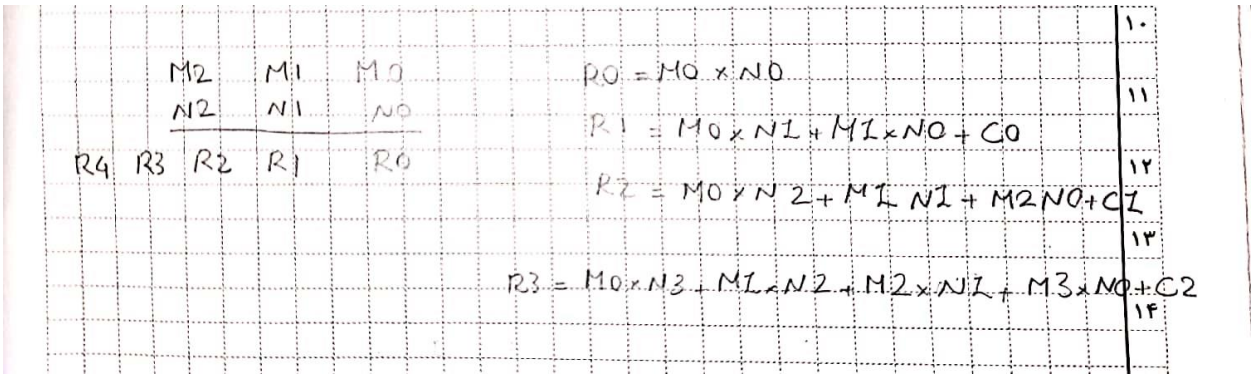
توجه: برای محاسبات 10 دیسک، لازم است 1023 خط در صفحه چاپ شود که این مقدار بسیار زیاد است. اما با ریختن 9 داخل cx میتوانیم این محاسبات را انجام دهیم. در اینجا نتیجه نهایی برای 4 دیسک (15 خط محاسبات) مشاهده میشود.



3. در بخش data segment، دو عدد و همچنین حاصل ضرب را initial میکنیم.

```
.MODEL SMALL
.STACK 64

.DATA

;store numbers as big endian
M   DW   2300H, 1874H, 0230H ;023018742300
N   DW   7091H, 0234H, 4719H ;479102347091
RESULT     DW   0H, 0H, 0H, 0H, 0H, 0H
```

برای ضرب کردن دو عدد 48 بیتی، ابتدا آن دو را به 3 عدد 16 بیتی تبدیل میکنیم. سپس برای محاسبه نتیجه مطابق زیر عمل میکنیم:



در قسمت code segment فرمول های بالا را پیاده سازی میکنیم. برای هرکدام، عدد اول را در ax و عدد دوم را در bx می ریزیم. سپس با دستور mul bx دو عدد را ضرب میکنیم. بعد از این دستور مقدار ضرب را از ax و مقدار بیت carry را از dx بدست می آوریم. Ax را به همان قسمت result و bx را به بعدی اضافه میکنیم.

```
.CODE
MAIN        PROC FAR
            MOV AX, @DATA
            MOV DS, AX

            ;M word computation
            ;r0 = n0 * m0
            MOV AX,[M]
            MOV BX,[N]
            MUL BX
            MOV [RESULT],AX
            MOV [RESULT+2],DX

            ;N word computation
            ;r1 = n1 * m0 + n0 * m1 + c0
            MOV AX,[M]
            MOV BX,[N+2]
            MUL BX
            ADD [RESULT+2],AX
            ADC DX,0
            ADD [RESULT+4],DX

            MOV AX,[M+2]
            MOV BX,[N]
            MUL BX
            ADC DX,0
            ADD [RESULT+2],AX
            ADD [RESULT+4],DX
```

```
                ;third word computation
                ;r2 = n2 * m0 + n1 * m1 + n0 * m2 + c1
                MOV AX,[M]
                MOV BX,[N+4]
                MUL BX
                ADD [RESULT+4],AX
                ADC DX,0
                MOV [RESULT+6],DX

                MOV AX,[M+2]
                MOV BX,[N+2]
                MUL BX
                ADD [RESULT+4],AX
                ADC DX,0
                ADD [RESULT+6],DX

                MOV AX,[M+4]
                MOV BX,[N]
                MUL BX
                ADD [RESULT+4],AX
                ADC DX,0
                ADD [RESULT+6],DX


                ;forth word computation
                ;r3 = n3 * m0 + n2 * m1 + n1 * m2 + n0 * m3 + c2
                MOV AX,[M]
                MOV BX,[N+6]
                MUL BX
                ADD [RESULT+6],AX
                ADC DX,0
                ADD [RESULT+8],DX

                MOV AX,[M+2]
                MOV BX,[N+4]
                MUL BX
                ADD [RESULT+6],AX
                ADC DX,0
                ADD [RESULT+8],DX

                MOV AX,[M+4]
                MOV BX,[N+2]
                MUL BX
                ADD [RESULT+6],AX
                ADC DX,0
                ADD [RESULT+8],DX

                MOV AX,[M+6]
                MOV BX,[N]
                MUL BX
                ADD [RESULT+6],AX
                ADC DX,0
                ADD [RESULT+8],DX

EXIT:           MOV AH, 4CH
                INT 21H
MAIN            ENDP
                END MAIN
```

نتیجه پس از اجرا:

```
M       2300h, 1874h, 0230h
N       7091h, 0234h, 4719h
RESULT  0D300h, 0C517h, 92CEh, 9E19h, 5527h
```