آزمایش 10

طاها موسوی 98243058

نیلوفر مرادی جم 97243063

گروه 2

سوالات تحلیلی:

1 - کدهای مختلف آدرس دهی پردازنده 8086 را با ذکر مثال توضیح دهید.

- Register mode: هردو عملوند رجیستر.مثال:
  MOV BX, DX
  MOV ES, AX
- Immediate mode: عملوند مقصد نمی‌تواند immediate باشد و عملوند منبع داده 8 یا 16 بیتی است.مثال:
  MOV DL, 08H
  MOV AX, 0A9FH
- Direct addressing: آدرس موثر مستقیما به عنوان جابجایی داده می‌شود. مثال:
  MOV BX, [1354H]
  MOV BL, [0400H]
- Register indirect addressing: آدرس دهی effective address در SI, DI یا BX است. مثال:
  MOV CX, [BX]
- Based addressing: آدرس موثر برابر مجموع رجیستر base register و آدرس جابجایی است.
  MOV AX, [BX + 08H]
- Indexed addressing: آدرس موثر، برابر است با مجموع index register و جابجایی است.
  MOV CX, [SI + 0A2H]
- Based indexed addressing: آدرس موثر، برابر است با مجموع base register و جابجایی و index register است.
  MOV DX, [BX + SI + 0AH]
- Direct & Indirect port addressing: برای خواندن و نوشتن بر روی port های خارجی است.
  IN AL, [09H]
- String addressing: در این مد بسته به جهت flag، مقادیر SI و DI افزایش یا کاهش می‌یابند. مثال:
  MOV BYTE
- Relative addressing: برای آدرس دهی نسبت به PC استفاده می‌شود.
  JZ 0AH

- Implied addressing: برای حالتی است که نیاز به operand نداریم. مثال:
CLC

2 - خطاهای موجود در هر یک از موارد زیر را بیان کنید

الف) MOV AX 3D	ب) MOV CX, CH	پ) ADD 2, CX
ت) INC AX, 2.	ث) MOV 23, AX	ج) MOVE AX, 1H
چ) ADD 3, 6	ح) MOV DX,CL	خ) MOV BH, AX
د) ADD AL, 2073H	ذ) MOV 7632H, CX	ر) IN BL, 04H

الف) نبودن ویرگول بین عملوند اول و دوم .

ب) ریختن CH(8 بیت) در CX(16 بیت).

پ) عملوند مقصد نباید یک عدد (immediate) باشد.

ت) عدد 2 و نقطه اضافی است.

ث) همانند مورد پ.

ج) باید به جای MOV،MOVE نوشته شود.

چ) عملوند اول نمی‌تواند immediate باشد.

ح) مقدار ۸ بیتی در ۱۶ بیت نباید ریخته شود.

خ) مقدار 16 بیتی در 8 بیت نباید ریخته شود.

د) یک مقدار ۸ بیتی (AL) نمی‌تواند با یک immediate ۱۶ بیتی جمع شود.

ذ) عملوند اول نمیتواند immediate باشد.

ر) عملوند اول دستور IN فقط AX یا AL است.

3 - برنامه زیر چه کاری انجام میدهد؟ علاوه بر پاسخ دادن به این سوال، بنویسید هر خط از کد چه کاری انجام میدهد.

عملکرد برنامه: ریختن 12 حرف اول استرینگ STRING1 در STRING2 به صورت معکوس.

```
DATA SEGMENT
        STRING1 DB 'MICROLAB OF SBU'
        STRING2 DB 15 DUP(0)
DATA ENDS
CODE SEGMENT
        ASSUME CS : CODE, DS : DATA,ES : DATA
START :
        MOV AX, DATA
        MOV DS, AX  ;Initialize DS with DATA
        MOV ES, AX  ;Initialize ES with DATA
        MOV BX, OFFSET STRING1 ;BX point to the start of STRING1
        MOV SI, BX ;move bx to SI( SI is source index)
        MOV DI, OFFSET STRING2 ;DI point to the start of STRING2
        ADD DI, 0CH ; ADD 12 TO DI Tto point to the 12th character
        CLD  ; clear direction flag
        MOV CX, 0CH  ; CX = 12 (Number of loops repeated)
UP :
        MOV AL, [SI]
        MOV ES : [DI] , AL ; ES point to the value of AL
        INC SI ; Increament value of SI(NEXT character of STRING1)
        DEC DI ; decrement value of DI(previous character of STRING2)
        LOOP UP ; back to the start of the loop
        REP MOVSB
        INT 03H
CODE ENDS
END START
ENDS
```

رفرنس های سوالات تحلیلی:

اسلاید های و کلاس درس استاد.

دستور کار:

سوال اول :

برای این سوال دو procedure نوشاه ایم به نام ها palindrome(برای چک کردن پالیندروم بودن استرینگ) و CountY(برای شمردن تعداد حروف y در استرینگ.

ابتدا استرینگ های مورد نیاز را تعریف می‌کنیم. استرینگی که قرار است تست شود، استرینگ ها نتیجه نهایی و حرف y.

سپس دو procedure تعریف میکنیم. و بعد کال کردن آنها برنامه خاتمه پیدا می‌کند.

```
.MODEL SMALL
.STACK 100H
.DATA

; The string to be printed
TEST_STRING DB 'tryyourbest', '$'
STRING1 DB 'String is palindrome', '$'
STRING2 DB 'String is not palindrome', '$'
Y DB 'y', '$'

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


CALL Palindrome
CALL CountY

;interrupt to exit
MOV AH, 4CH
INT 21H
MAIN ENDP
Palindrome PROC
```

CountY:

SI را به ابتدای استرینگ اشاره می‌کند و DI به ابتدای y. و مقدار DX را برابر 0 قرار می‌دهیم. و سپس در لوپی چک می‌کنیم که آیا به انتهای استرینگ رسیده ایم یا نه. اگر رسیدیم، به لیبل FINISH، جامپ می-کنیم در غیر اینصورت ادامه برنامه. در ادامه برنامه مقادیر موجود در ایندکس های SI و DI را در AX و BX می‌ریزیم. و سپس با هم مقایسه می‌کنیم. اگر برابر هم بود یعنی به تعداد y ها باید یکی اضافه شود که در لیبل EQUAL این کار انجام می‌شود، در غیر این صورت ادامه برنامه. و تکرار لوپ.

```
CountY PROC
    ; load the starting address
    ; of the string and y
    MOV SI,OFFSET TEST_STRING
    MOV DI,OFFSET Y
    MOV DX, 0H    ;Set DX to 0
LOOP3 :
    MOV AX, [SI]
    CMP AL, '$' ;Check end of string, if yes, jump to finish
    JE FINISH
    MOV AX,[SI]
    MOV BX, [DI]
    CMP AL, BL ; Check if AL = Bl (y)
    JE EQUAL ; If equal, jump to EQUAL
    INC SI
    JMP LOOP3
FINISH:
    RET
EQUAL:
    INC DL ; Increase the number of 'y'
    INC SI
    JMP LOOP3
CountY ENDP
```

Palindrome:

در این procedure هم تا حدودی مثل قبل عمل می‌کنیم.
ولی ابتدا SI را در یکی لوپ، به انتهای استرینگ می‌بریم. و DI را به ابتدای آن.

```
Palindrome PROC

; load the starting address
; of the string
MOV SI,OFFSET TEST_STRING

; traverse to the end of;
;the string
LOOP1 :
    MOV AX, [SI]
    CMP AL, '$'
    JE LABEL1
    INC SI
    JMP LOOP1

;load the starting address;
;of the string
LABEL1 :
    MOV DI,OFFSET TEST_STRING
    DEC SI

    ; check if the string is palindrome;
    ;or not
    LOOP2 :
    CMP SI, DI
    JL OUTPUT1
    MOV AX,[SI]
    MOV BX, [DI]
    CMP AL, BL
    JNE OUTPUT2

    DEC SI
    INC DI
    JMP LOOP2
```

و مانند قبلی مقادیر موجود در آن ایندکس ها را با هم چک می‌کنیم. اگر برابر بود ادامه برنامه. به شرطی که همچنان مقدار SI بزرگتر مساوی DI باشد. در غیر اینصورت اگر کوچیکتر شد. یعنی استرینگ پالیندروم است و به لیبلی جامپ می‌کند تا پیام مناسب را چاپ کند.

```
    ;or not
    LOOP2 :
    CMP SI, DI
    JL OUTPUT1
    MOV AX,[SI]
    MOV BX, [DI]
    CMP AL, BL
    JNE OUTPUT2

    DEC SI
    INC DI
    JMP LOOP2

OUTPUT1:
    ;load address of the string
    LEA DX,STRING1

    ; output the string;
    ;loaded in dx
    MOV AH, 09H
    INT 21H
    RET
```

اگر هم دو حرف با هم برابر نبودن به لیبل دیگری جامپ می‌کند و پیام مناسب را چاپ می‌کند.

```
OUTPUT2:
    ;load address of the string
    LEA DX,STRING2

    ; output the string
    ; loaded in dx
    MOV AH,09H
    INT 21H
    RET

Palindrome ENDP
```

```
Palindrome PROC

; load the starting address
; of the string
MOV SI,OFFSET TEST_STRING

; traverse to the end of;
;the string
LOOP1 :
    MOV AX, [SI]
    CMP AL, '$'
    JE LABEL1
    INC SI
    JMP LOOP1

;load the starting address;
;of the string
LABEL1 :
    MOV DI,OFFSET TEST_STRING
    DEC SI

    ; check if the string is palindrome;
    ;or not
    LOOP2 :
    CMP SI, DI
    JL OUTPUT1
    MOV AX,[SI]
    MOV BX, [DI]
    CMP AL, BL
    JNE OUTPUT2

    DEC SI
    INC DI
    JMP LOOP2
```

نتیجه :

```
C:\>q1.EXE
String is not palindrome
C:\>s_
```

قبل دیباگ:

```
-t
AX=074A BX=0000 CX=009E DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=074E CS=0744 IP=0003 NV UP EI PL ZR NA PE NC
0744:0003 8ED8              MOV     DS,AX
-s
```

بعد دیباگ:

```
AX=4C24 BX=2479 CX=00A8 DX=0004 SP=0100 BP=0000 SI=0017 DI=0046
DS=074A ES=0734 SS=074F CS=0744 IP=000D NV UP EI PL ZR NA PE NC
0744:000D CD21              INT     21
-T

Program terminated normally (0024)
-T
```

منبع سوال 1:
https://www.geeksforgeeks.org/8086-program-to-check-whether-a-string-is-palindrome-or-not/

سوال 2:

ضرب: در این ضرب 32x32 که استفاده کرده ایم، حاصل 32 بیت کم ارزش جواب ضرب در رجیستر EAX ذخیره می‌شود و 32 بیت پر ارزش تر در EDX .
سپس در ادامه یک Prepare_for_print کال شده که در آن EAX و EDX را در Result که یک دیتای 64 بیتی است ذخیره می‌کنیم. و سپس با دوبار کال کردن تابع پرینت 32 بیتی و استفاده از اینستراکشن مربوط به چاپ در کنسول توسط DOS نتایج را چاپ می‌کنیم.

```
MULTIPLE_NUMS PROC
        MOV EAX, DATA1
        IMUL DATA2
        CALL Prepare_for_print
        LEA SI, OUT_STR
        LEA BX, Result
        CALL Print_NUM
        LEA DX,OUT_STR
    MOV AH,9
    INT 21H
LEA SI, OUT_STR
        LEA BX, Result+4
        CALL Print_NUM
        LEA DX,OUT_STR
    MOV AH,9
    INT 21H

        RET
MULTIPLE_NUMS ENDP
```

پرینت:

```
Print_NUM PROC
for_first_16_bit:
        MOV CX,0
    push CX
        PUSH BX
        ADD BX, 2
        JMP third
for_second_16_bit:
        pop BX
        pop cx
        ADD CX,1
        CMP CX,2
        JE END_OF
        push cx
        push bx
        JMP third
END_OF:
   RET
third:
          MOV AX, [BX]
          MOV CX,4
          MOV BX, 16
first_loop:
          MOV DX,0
      DIV BX
          CMP DL, 09H
          JG zero_nine
      ADD DL,30H
back_from_A_F:
          PUSH DX
          LOOP first_loop
          JMP CC
zero_nine:
          ADD DL, 37H
          JMP back_from_A_F
CC:
          MOV CX, 4
LOOP2:
          POP AX
          JMP handle_put_value_in_buffer
back_from_handle:
      LOOP LOOP2
          JMP for_second_16_bit
handle_put_value_in_buffer:
       MOV [SI],AL
       INC SI
       JMP back_from_handle
Print_NUM ENDP
```

در این یک دیتای 32 بیتی داریم که در ابتدا 16 بیت پرارزش تر و سپس 16 بیت کم ارزش تر را چاپ می‌کنیم.
ابتدا در رجیستر BX 16 بیت پرارزش تر را می‌گذاریم و مقدار قبلی آن را در استک پوش می‌کنیم. و سپس یک سری روتین مربوط به تبدیل 16 بیتی digit به کاراکتر را انجام می‌دهیم. و در OUT_STR که در دیتا تعریف کردیم ذخیره می‌کنیم. و سپس BX ای که در استک پ.ش کرده بودیم را پاپ می‌کنیم و

برای 16 بیت کم ارزش تر همین روال را دوباره انجام می‌دهیم و در ادامه OUT_STR چاپ می‌شود. و در نهایت در آن یک 32 بیت کاراکتر داریم.

تقسیم:

```
div_NUMS PROC
MOV EAX,DATA1
MOV EBX,DATA2
DIV EBX
div_Nums ENDP
```

در تقسیم EAX مقسوم و EBXمقسوم علیه و EDX خارج قسمت می باشد.

نتیجه :

نتیجه زیر نتیجه دو ضرب دیتاست که توسط تابع پرینت، چاپ شده است.

```
C:\>test.exe
0000000000000007
```