**את"מ מעבדה 12:  
מגישים:  
 אורי מלכא- 314862996   
אלן ציפין- 313206062**  
second function:

;

; Lab12 - ori and alan

.MODEL SMALL

.STACK 100h

.DATA

;vars for expPowM2x

sum1 DQ 1.0

func1 DD 0

powx1 DQ 0

fac1 DQ 1.0

sign1 DQ 1.0

tempn1 DW 0

one DQ 1.0

;vars for expPowM2xEps

sum2 DQ 1.0

func2 DD 0

powx2 DQ 0

fac2 DQ 1.0

sign2 DQ 1.0

tempn2 DW 0

product DQ 1.0

.CODE

.386

.387

;extern double expPowM2x(long(\*pf)(int n), double x, int n);

PUBLIC \_expPowM2x

\_expPowM2x PROC NEAR

;[BP+4] -> expPowM2x(long(\*pf)(int n)

;[BP+6] -> double x

;[BP+14] -> int n

;save values

PUSH BP

MOV BP,SP

;our algorithm follow this steps:

;1) call func to get f(n) and store the returned value in func1

;2) calc n! and store it in fac1

;3) calc x^n and store it in powx1

;4) we have a sign "flag" that will be multiplied each time by -1 to change it to addition or subtraction (initialized to 1)

;5) calc product (f(n)/n!\*x^n)

;6) multiply prod1 by the sign "flag"

;7) sum1 + prod1

; we will repeat this algorithm n times

XOR CX,CX

INC CX

SumUptoN:

CMP CX,[BP+14]

JA FinishSum

; step 1

PUSH CX

CALL [BP+4] ;RETURNED VALUE IN DX:AX

POP CX

XOR EBX,EBX

ADD BX,DX

SHL EBX,16

ADD BX,AX

MOV func1,EBX

; step 2

XOR BX,BX

FLD QWORD PTR one

FSTP QWORD PTR fac1

CalcFac:

CMP BX,CX

JE WeHaveFac

INC BX

MOV WORD PTR tempn1,BX

FILD WORD PTR tempn1

FLD QWORD PTR fac1

FMUL

FSTP QWORD PTR fac1

JMP CalcFac

WeHaveFac:

;step 3

XOR BX,BX

INC BX

FLD QWORD PTR [BP+6]

CalcPow:

CMP BX,CX

JE WeHavePow

FMUL QWORD PTR [BP+6] ;ST[0]\*x

INC BX

JMP CalcPow

WeHavePow:

FSTP QWORD PTR powx1

;step 4

FLD QWORD PTR sign1

FCHS

FSTP QWORD PTR sign1

;step 5

FILD DWORD PTR func1

FLD QWORD PTR fac1

FDIV; ST[0]=f(n)/n!

FMUL QWORD PTR powx1

;here we have the product ST[0]f(n)/n!\*x^n

;step6

FMUL QWORD PTR sign1

;step7

FLD QWORD PTR sum1

FADD ;ST[0]=ST[0]+SUM (st[0] could be positive or negative depends on the sign1 specific sign

FSTP QWORD PTR sum1 ;sum1=sum1(+-)st[0]

INC CX

JMP SumUptoN

FinishSum:

FLD QWORD PTR sum1

POP BP

RET

\_expPowM2x ENDP

;extern double expPowM2xEps(long(\*pf)(int n), double x, double eps);

PUBLIC \_expPowM2xEps

\_expPowM2xEps PROC NEAR

;[BP+4] -> long(\*pf)(int n)

;[BP+6] -> double x

;[BP+14] -> double eps

;save values

PUSH BP

MOV BP,SP

XOR CX,CX

INC CX

SumUptoEps:

;We initialized the var product to be 1 beacuse the sum is 1+SOP the first Product when the ORIGINAL SOP starts from n=0, if eps>=1 the answer is 1

FLD QWORD PTR product

FCOM QWORD PTR [BP+14] ;PRODUCT-EPS , if eps>=product then we JBE

FSTSW AX

SAHF

JBE FinishSumEps

; step 1

PUSH CX

CALL [BP+4] ;RETURNED VALUE IN DX:AX

POP CX

XOR EBX,EBX

ADD BX,DX

SHL EBX,16

ADD BX,AX

MOV func2,EBX

; step 2

XOR BX,BX

FLD QWORD PTR one

FSTP QWORD PTR fac2

CalcFac2:

CMP BX,CX

JE WeHaveFac2

INC BX

MOV WORD PTR tempn2,BX

FILD WORD PTR tempn2

FLD QWORD PTR fac2

FMUL

FSTP QWORD PTR fac2

JMP CalcFac2

WeHaveFac2:

;step 3

XOR BX,BX

INC BX

FLD QWORD PTR [BP+6]

CalcPow2:

CMP BX,CX

JE WeHavePow2

FMUL QWORD PTR [BP+6] ;ST[0]\*x

INC BX

JMP CalcPow2

WeHavePow2:

FSTP QWORD PTR powx2

;step 4

FLD QWORD PTR sign2

FCHS

FSTP QWORD PTR sign2

;step 5

FILD DWORD PTR func2

FLD QWORD PTR fac2

FDIV; ST[0]=f(n)/n!

FMUL QWORD PTR powx2

;here we have the product ST[0]f(n)/n!\*x^n

FST QWORD PTR product

;step6

FMUL QWORD PTR sign2

;step7

FLD QWORD PTR sum2

FADD ;ST[0]=ST[0]+SUM (st[0] could be positive or negative depends on the sign1 specific sign

FSTP QWORD PTR sum2 ;sum1=sum1(+-)st[0]

INC CX

JMP SumUptoEps

FinishSumEps:

FLD QWORD PTR sum2

POP BP

RET

\_expPowM2xEps ENDP

END

  
first function:  
