

Name : Nisarg Amlani .k.

Roll : CE001

ID : 22ceueg082

LAB : 4

AIM: 8087 programs using 8086 and 8087 instruction set

1. Compute x^y . x and y can be integer or real.

=> x = 2.0 , y = 3.5

Code:

data segment

x dd 2.0

y dd 3.5

power dd ?

intpower dd ?

res dd ?

cw dw 07ffh

data ends

code segment

assume cs:code, ds:data

start:

mov ax, data

mov ds, ax

finit

fld y ; Load exponent (y) onto the FPU stack

fld x ; Load base (x) onto the FPU stack

fyl2x ; Calculate $y * \log_2(x)$, result in st(0)

fst power ; Store the calculated power ($y * \log_2(x)$)

fldcw cw ; Load control word for truncation

frndint ; Round st(0) to the nearest integer

fst intpower ; Store the integer part of the power

fld power ; Reload the power value

fxch ; Exchange st(0) and st(1) (intpower and fractional part)

```

fsub      ; Subtract intpower from power to isolate the fractional part
f2xm1     ; Calculate 2^(fractional part) - 1
fld1      ; Load constant 1 onto the FPU stack
fadd      ; Add 1 to calculate 2^(fractional part)

```

```

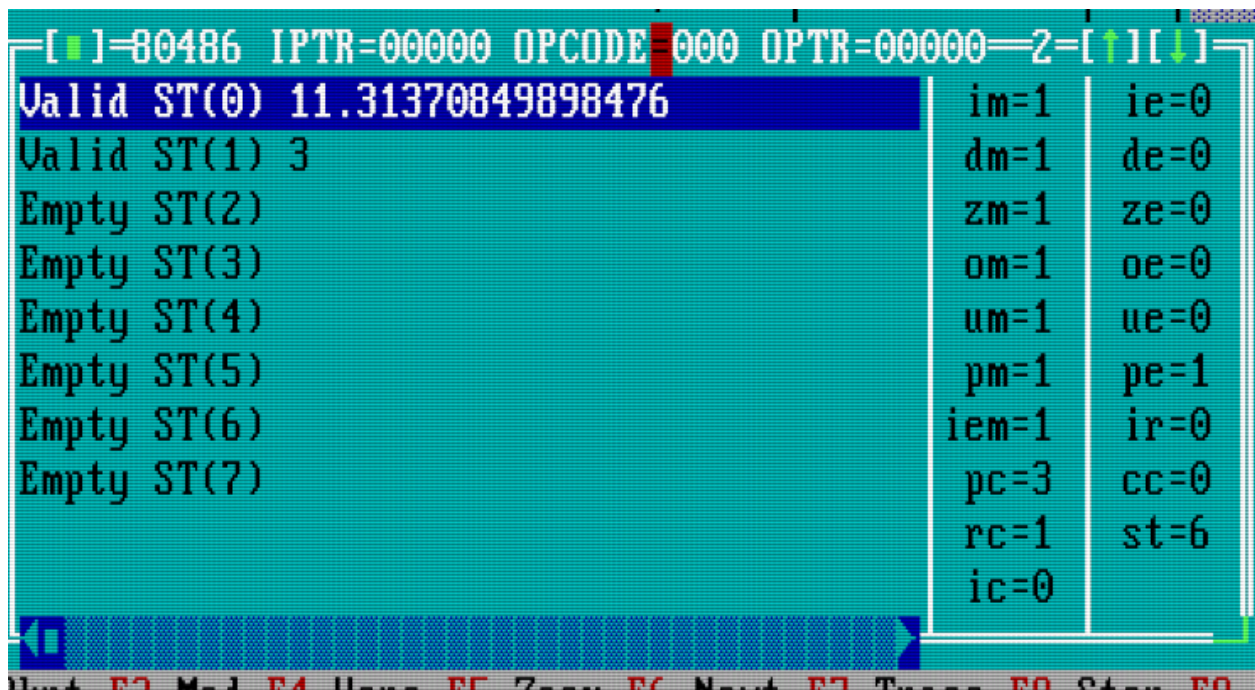
fld intpower ; Reload the integer part of the power
fxch        ; Exchange st(0) and st(1) (fractional and integer parts)
fscale      ; Scale result by 2^intpower (final result in st(0))
fst res     ; Store the final result in memory

```

```

int 3
code ends
end start

```



2. Compute $P \times (1 + r/100)^n$. Example: $P = 500.0$, $r = 12\%$ and $n = 4.0$

Code:

```

data segment
r dd 12.0
n dd 4.0

```

```
p dd 500.0
divider dd 100.0
power dd ?
intpower dd ?
res dd ?
cw dw 07ffh
data ends
```

```
code segment
assume cs:code, ds:data
```

```
start:
```

```
    mov ax, data
    mov ds, ax
    finit
```

```
    fld n          ; Load n (exponent) onto the FPU stack
    fld r          ; Load r (interest rate) onto the FPU stack
    fld divider    ; Load the divider value (100.0) onto the FPU stack
    fdiv           ; Divide st(1) by st(0) (r / divider), result in st(1)
    fld1           ; Load constant 1.0 onto the FPU stack
    fadd           ; Add st(0) and st(1) (r/divider + 1), result in st(0)
    fyl2x          ; Compute n * log2(st(0)) (exponentiation via logarithm), result in st(0)
    fst power      ; Store the calculated power (n * log2(1 + r/100))
```

```
    fldcw cw       ; Load control word to configure rounding mode
    frndint         ; Round st(0) to the nearest integer
    fst intpower    ; Store the integer part of the power in memory
```

```
    fld power       ; Reload the full power value (n * log2(1 + r/100))
    fxch           ; Exchange st(0) and st(1) (swap full power and integer power)
    fsub           ; Subtract integer part from full power to isolate the fractional part
    f2xm1          ; Compute 2^(fractional part) - 1, result in st(0)
    fld1           ; Load constant 1.0 onto the FPU stack
    fadd           ; Add 1 to compute 2^(fractional part)
```

```
    fld intpower    ; Reload the integer part of the power
    fxch           ; Exchange st(0) and st(1) (swap fractional part result and integer part)
    fscale         ; Scale st(0) by 2^st(1) (compute full 2^power)
    fmul p         ; Multiply the result by principal amount (p)
    fst res        ; Store the final result (res) in memory
```

```
    int 3
code ends
end start
```

[]=80486 IPTR=00000 OPCode=000 OPTR=00000=2=[↑][↓]		
Valid ST(0)	786.75967385840547	im=1 ie=0
Valid ST(1)	0	dm=1 de=0
Empty ST(2)		zm=1 ze=0
Empty ST(3)		om=1 oe=0
Empty ST(4)		um=1 ue=0
Empty ST(5)		pm=1 pe=1
Empty ST(6)		iem=1 ir=0
Empty ST(7)		pc=3 cc=8
		rc=1 st=6
		ic=0

3. Compute roots of quadratic equation. If roots are imaginary display message "Roots are imaginary".

Example: $x^2 + 7x + 10 = 0$

Code:

data segment

a dd 1.0

b dd 7.0

c dd 15.0

val dd 4.0

val1 dd 2.0

temp dw ?

alpha dd ?

ans dd ?

ans1 dd ?

str1 db "Roots are imaginary\$"; Message for imaginary roots

data ends

code segment

Assume cs:code, ds:data

start:

mov ax, data

mov ds, ax

finit

```
; Calculate the discriminant:  $b^2 - 4ac$ 
fld b          ; Load coefficient b onto the FPU stack
fmul b         ; Compute  $b^2$ 
fld a          ; Load coefficient a onto the FPU stack
fmul c         ; Compute  $a * c$ 
fmul val       ; Compute  $4 * a * c$ 
fsub          ; Subtract  $4 * a * c$  from  $b^2$ 
fsqrt         ; Compute the square root of the discriminant
fstsw temp     ; Store the FPU status word in temp
mov ax, temp   ; Move the status word to AX
and ax, 0001h  ; Check for invalid operation (e.g., sqrt of a negative number)
cmp ax, 0001h  ; Compare the result with 1 (indicating imaginary roots)
jz imaginary   ; If discriminant is negative, jump to the "imaginary" label
fstp alpha     ; Store the square root of the discriminant in alpha
```

```
; Calculate the first root:  $(-b + \sqrt{\text{discriminant}}) / (2 * a)$ 
fld b          ; Load coefficient b onto the FPU stack
fchs          ; Negate b (-b)
fadd alpha     ; Add the square root of the discriminant
fld a          ; Load coefficient a onto the FPU stack
fmul val1      ; Multiply a by 2
fdiv          ; Divide  $(-b + \sqrt{\text{discriminant}})$  by  $(2 * a)$ 
fstp ans       ; Store the result in ans (first root)
```

```
; Calculate the second root:  $(-b - \sqrt{\text{discriminant}}) / (2 * a)$ 
fld b          ; Load coefficient b onto the FPU stack
fchs          ; Negate b (-b)
fsub alpha     ; Subtract the square root of the discriminant
fld a          ; Load coefficient a onto the FPU stack
fmul val1      ; Multiply a by 2
fdiv          ; Divide  $(-b - \sqrt{\text{discriminant}})$  by  $(2 * a)$ 
fstp ans1      ; Store the result in ans1 (second root)
```

```
jmp exit       ; Jump to exit
```

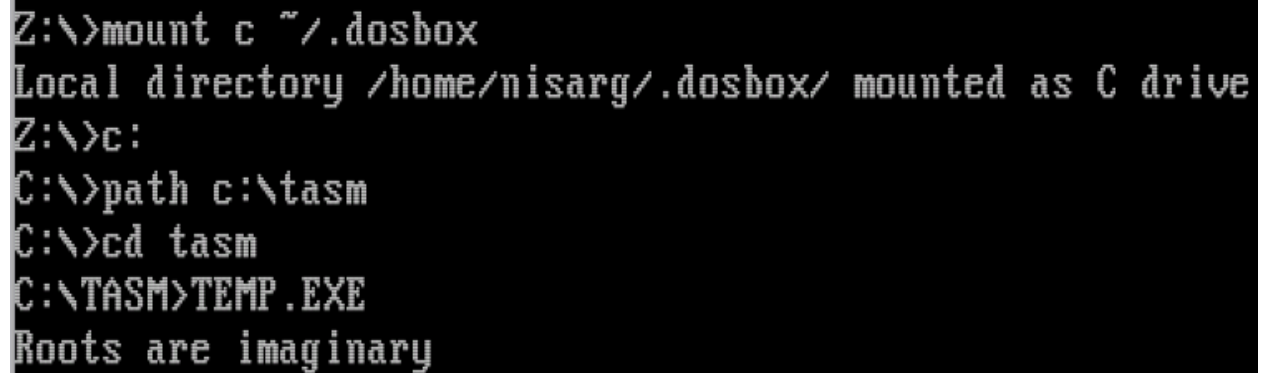
imaginary:

```
; Print the message "Roots are imaginary"
mov ah, 09h    ; DOS interrupt for string output
```

```
    lea dx, str1      ; Load the address of the message into DX
    int 21h           ; Execute the interrupt
    jmp exit          ; Jump to exit
```

exit:

```
    int 3             ; Terminate the program
code ends
end start
```



A screenshot of a DOSBox terminal window with a black background and white text. The terminal shows the following sequence of commands and output:

```
Z:\>mount c ~/.dosbox
Local directory /home/nisarg/.dosbox/ mounted as C drive
Z:\>c:
C:\>path c:\tasm
C:\>cd tasm
C:\TASM>TEMP.EXE
Roots are imaginary
```

4. Compute $\sec(x)$, $\operatorname{cosec}(x)$ and $\cot(x)$.

Code:

=> $\cot(x)$

```
data segment
th dd 60.0
d dd 180.0
pi dd 3.1428
res dd ?
data ends
```

code segment

assume cs:code , ds:data

start:

mov ax, data

mov ds, ax

finit

fld th

fldpi

fmul

fdiv d

fptan

fxch

fdiv

fst res

int 03h

code ends

end start

[EIP]=80486 IPTR=000000 UPCODE=000 OPTR=000000==2=[↑][↓]=		
Valid ST(0) 0.57735026918962576	im=1	ie=0
Empty ST(1)	dm=1	de=0
Empty ST(2)	zm=1	ze=0
Empty ST(3)	om=1	oe=0
Empty ST(4)	um=1	ue=0
Empty ST(5)	pm=1	pe=1
Empty ST(6)	iem=0	ir=0
Empty ST(7)	pc=3	cc=2
	rc=0	st=7
	ic=0	

=> Sec

data segment

th dd 60.0

d dd 180.0

res dd ?
data ends

code segment

assume cs:code , ds:data

start:

mov ax, data

mov ds, ax

finit

fld th

fldpi

fmul

fdiv d

fptan

fxch

fst res

fmul res

fadd

fsqrt

int 03h

code ends

end start

[]=80486 IPTR=000000 OPCODE=000 OPTR=000000==2=[↑][↓]=		
Valid ST(0)	1.9999999865388263	im=1 ie=0
Empty ST(1)		dm=1 de=0
Empty ST(2)		zm=1 ze=0
Empty ST(3)		om=1 oe=0
Empty ST(4)		um=1 ue=0
Empty ST(5)		pm=1 pe=1
Empty ST(6)		iem=0 ir=0
Empty ST(7)		pc=3 cc=2
		rc=0 st=7
		ic=0

=> **cosec**(x)

data segment

th dd 60.0

d dd 180.0

pi dd 3.142

one dd 1.0

res dd ?

data ends

code segment

assume cs:code , ds:data

start:

mov ax, data

mov ds, ax

finit

fld th

fldpi

fmul

fdiv d

fptan

fxch

fdiv

fst res

fmul res

fld one
fadd
fsqrt
fst res

int 03h
code ends
end start

[]=80486 IPTR=000000 UPCODE=000 OPTR=000000=Z=[↑][↓]=		
Valid ST(0)	1.1547005357886475	im=1 ie=0
Empty ST(1)		dm=1 de=0
Empty ST(2)		zm=1 ze=0
Empty ST(3)		om=1 oe=0
Empty ST(4)		um=1 ue=0
Empty ST(5)		pm=1 pe=1
Empty ST(6)		iem=0 ir=0
Empty ST(7)		pc=3 cc=8
		rc=0 st=7
		ic=0