NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
“IGOR SIKORSKY KYIV POLYTECHNIC INSTITUTE”

Faculty of Informatics and Computer Engineering

Department of Computer Engineering

Lab Practical Lesson 6 Report

Calculation of arithmetic expressions and transcendental functions Using Coprocessor Commands ix87

Variant 12

Student, group IM-14
(group code)

in the educational and professional program

“Software Engineering For Computer System”

Specialty 121 "Computer Engineering"

Mehmet KULUBECİOGLU

Reviewer Associate Professor, Dr.Ph. Pavlov Valerii
(position, academic degree, academic status, surname and initials)

Kyiv – 2023

Goal of the work

Learn Assembler commands for floating-point arithmetic and gain skills in performing calculations with array elements. Develop a program in Assembly language. The program should calculate an arithmetic formula:

$$\frac{\sqrt{25/c} - d + 2}{b + a - 1}$$

with using Coprocessor commands **ix87** and output a result[1] in user interface.

List of the tasks

1. Please learn arithmetic and logical operations for floating-point arithmetic [1-4].
2. Please, develop a series of 6 test examples[2]:
   - 4 sets for different combinations of numerator and denominator signs;
   - 1 set, which gives you a denominator equal to 0;
   - 1 set that gives the value of the function argument at which it cannot be evaluated (If the function is evaluated for any values, add any similar to clause 1).
3. Please, perform detailed mathematical calculations for each set and place them in the report.
4. Please, implement the same calculations in the form of an assembly language program that should provide:
   - all input datasets must be represented as one-dimensional arrays;
   - calculations should represent a loop that repeats for each input dataset;
   - on each cycle, the results should be displayed in the MessageBox window and should contain:
     - variant number and formula for calculations;
     - values of the input variables of this set, with notations for each;
     - the value of the result of the calculation with notation[3].
5. Please, develop a compute sequence strategy to optimize program.
6. For all tasks, provide for setting the values of the input variables in format **double (DQ)** of the intermediate results of the calculations - in the format **long double (DT)**, and the final - again in **double**.
7. Please make debugging of the program.
8. Please, compare results in the program with the control (mathematical) calculations.
9. Please write a listing of the program in the report for the practical lesson.
10. Please add screenshots with results (MessageBox window) of the calculations as well.
11. Please make a conclusion of the lab practical work. Conclusions should contain an analysis of the results of a comparison of theoretical (mathematical) and practical calculations.

Recommended literature:

1. “Intel® 64 and IA-32 Architectures Software Developer’s Manual”, Volume 1.
2. https://flatassembler.net/docs.php?article=manual#2.1.13
3. https://docs.oracle.com/cd/E19120-01/open.solaris/817-5477/index.html
4. https://www.felixcloutier.com/x86/

2.1 Calculations for Control Set 1 (numerator is positive, denominator is positive):
a = 0.5; b= 5.5; c = 5.0; d = 0.75

Denominator equal b+a-1 = 5.5+0.5-1 = 6.0-1 = 5.0
Numerator equal sqrt(25/c)-d+2 = sqrt(25/5.0)-0.75+2 = sqrt(5)-0.75+2 =2.2360679774997896964091736687313-0.75+2 = 2.2360679774997896964091736687313+1.25 = 3.4860679774997896964091736687313

Result equal = 3.4860679774997896964091736687313/5.0 = 0.69721359549995793928183473374626

2.2 Calculations for Control Set 2 (numerator is positive, denominator is negative):
a = -1.3; b= -2.7; c = 1.5; d = 0.25

Denominator equal b+a-1 = -2.7-1.3-1 = 6.0-1 = -5.0
Numerator equal sqrt(25/c)-d+2 = sqrt(25/1.5)-0.25+2 = sqrt(16.6)-0.25+2 = 4.0824829046386301636621401245098-0.25+2 = 4.0824829046386301636621401245098+1.75= 5.8324829046386301636621401245098

Result equal = 5.8324829046386301636621401245098/-5.0 = -1.166496580927726032732428024902

2.3 Calculations for Control Set 3 (numerator is negative, denominator is positive):
a = 32.5; b= 0.5; c = 26.5; d = 10.5

Denominator equal b+a-1 = 32.5+0.5-1 = 33.0-1 = 32.0
Numerator equal sqrt(25/c)-d+2 = sqrt(25/26.5)-10.5+2 = sqrt(0.94339622641509433962264150943396)-10.5+2 = 0.97128586235726418073560089284883-10.5+2 = 0.97128586235726418073560089284883-8.5 = -7.528714137642735819264399107151

Result equal = -7.528714137642735819264399107151/32.0 = -0.23527231680133549435201247209847

2.4 Calculations for Control Set 4 (numerator is negative, denominator is negative):
a = 0.6; b= 0.3; c = 23.5; d = 41.5

Denominator equal b+a-1 = 0.3+0.6-1 = 0.9-1 = -0.1
Numerator equal sqrt(25/c)-d+2 = sqrt(25/23.5)-41.5+2 = sqrt(1.0638297872340425531914893617021)-41.5+2 = 1.0314212462587934072498809734942-41.5+2 = 1.0314212462587934072498809734942 -39.5 = -38.468578753741206592750119026506

Result equal = -38.468578753741206592750119026506/-0.1 = 384.68578753741206592750119026506

2.5 Calculations for Control Set 5: Exception situation – sqrt(-x) is an invalid expression.
a = -0.9; b= 1.6; c = -3.4; d = 2.1

Denominator equal b+a-1 = 1.6-0.9-1 = 0.7-1 = -0.3

Numerator equal sqrt(25/-3.4)-d+2 = sqrt(25/-3.4)-2.1+2 = sqrt(-3529411764705882352941176470588)-2.1+2 = invalid-4.1 = invalid.

Result equal = invalid/-0.3
Calculations cannot be performed. sqrt(-x) is invalid.

2.6 Calculations for Control Set 6: Exception situation – denominator is 0.

a = 0.5; b= 0.5; c = 3.2; d = 1.3

Denominator equal b+a-1 = 0.5+0.5-1 = 1-1 = 0

Numerator equal sqrt(25/3.2)-d+2 = sqrt(7.8125)-1.3+2 = 2.7950849718747371205114670859141-1.3+2 = 3.4950849718747371205114670859141

Result equal 3.4950849718747371205114670859141/0 = invalid
Calculations cannot be performed. A denominator is 0, invalid

3. The following arithmetic operations are used in the calculations:
- addition;
- subtraction;
- multiplication;
- division;
- calculating the Arctangent Function.

To optimize the program and minimize the number of instructions, we use a table that describes these operations:

Function	Function in Assembler	Argument X	Argument Y	Result	Commentary
addition	FADD	ST(0)	ST(i)	ST(0)	ST(0) = ST(0) + ST(i)
		ST(i)	ST(0)	ST(i)	ST(i) = ST(0) + ST(i)
		ST(0)	mem32 or mem64	ST(0)	ST(0) = ST(0) + mem32/64
addition and pop	FADDP	ST(i)	ST(0)	ST(i)	ST(i) = ST(0) + ST(i) and pop
		- {ST(0)}	- {ST(1)}	ST(0)	ST(0) = ST(0) + ST(1) and pop
subtraction	FSUB	ST(0)	ST(i)	ST(0)	ST(0) = ST(0) - ST(i)
		ST(i)	ST(0)	ST(i)	ST(i) = ST(i) - ST(0)
		ST(0)	mem32 or mem64	ST(0)	ST(0) = ST(0) - mem32/64
subtraction and pop	FSUBP	ST(i)	ST(0)	ST(i)	ST(i) = ST(i) - S(0) and pop
		- {ST(0)}	- {ST(1)}	ST(1)	ST(1) = ST(1) - ST(0) and pop
reverse subtraction	FSUBR	ST(0)	ST(i)	ST(0)	ST(0) = ST(i) - ST(0)
		ST(i)	ST(0)	ST(i)	ST(i) = ST(0) - ST(i)
		ST(0)	mem32 or mem64	ST(0)	ST(0) = mem32/64 - ST(0)
multiply	FMUL	ST(0)	ST(i)	ST(0)	ST(0) = ST(0) * ST(i)
		ST(i)	ST(0)	ST(i)	ST(i) = ST(0) * ST(i)
		ST(0)	mem32 or mem64	ST(0)	ST(0) = ST(0) * mem32/64
multiply and pop	FMUL	ST(i)	ST(0)	ST(i)	ST(i) = ST(0) * ST(i) and pop
		- {ST(0)}	- {ST(1)}	ST(1)	ST(1) = ST(0) * ST(1) and pop
division	FDIV	ST(0)	ST(i)	ST(0)	ST(0) = ST(0) / ST(i)
		ST(i)	ST(0)	ST(i)	ST(i) = ST(i) / ST(0)
		ST(0)	mem32 or mem64	ST(0)	ST(0) = ST(0) / mem32/64
reverse division	FDIVR	ST(0)	ST(i)	ST(0)	ST(0) = ST(i) / ST(0)
		ST(i)	ST(0)	ST(i)	ST(i) = ST(0) / ST(i)

		ST(0)	mem32 or mem64	ST(0)	ST(0) = mem32/64 / ST(0)
Square root	FSQRT	ST(0)	ST(0)	ST(0)	ST(0 = SQRT(ST(0)
Status word to Register	FSTSW	AX	x	SW	Move SW to AX
AH to F	SAHF	AH	X	Flag	From AH to Flag register
Const 25	FLD	tfive	x	ST(0)	ST(0) = 25.0
Load	FLD	mem32/mem64	x	ST(0)	ST(0) = mem32/mem64
Store and pop	FSTP	mem32/mem64	x	mem	mem32/mem64 = ST(0) and pop
Const 1	FLD1	-	x	ST(0)	ST(0) = "1.0"

Using Stack Registers in Calculations

Stack Register Flow

Formulae : $$\frac{\sqrt{25/c} - d + 2}{b + a - 1}$$

$$\frac{\sqrt{st(0)/st(1)} - d + 2}{b + a - 1}$$

$$\Downarrow$$

$$\frac{\sqrt{st(0)} - d + 2}{b + a - 1}$$

$$\Downarrow$$

$$\frac{st(0) - d + 2}{b + a - 1}$$

$$\Downarrow$$

$$\frac{st(1) - st(0)}{b + a - 1}$$

$$\Downarrow$$

$$\frac{st(0)}{b + a - 1}$$

$$\Downarrow$$

$$\frac{st(1)}{st(0)} = st(0) = \frac{\sqrt{25/c} - d + 2}{b + a - 1}$$

4. Listing of program code

```
.386                              ; all instructions of the proper processor are settled
.model flat, stdcall              ; the model of memory is set
option casemap :none              ; the register of characters is recognized
```

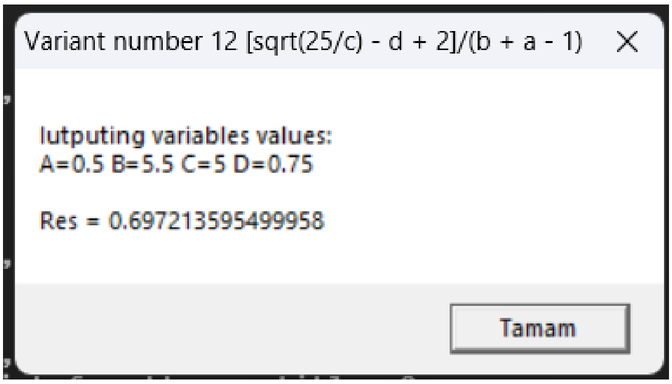
```
includelib \masm32\lib\kernel32.lib     ; connect external library kernel32.lib
includelib \masm32\lib\user32.lib       ; connect external library user32.lib
includelib \masm32\lib\masm32.lib       ; connect external library masm32.lib
includelib \masm32\lib\fpu.lib          ; connect external library fpu.lib
include \masm32\include\windows.inc     ; for a management by the modes of Window
include \masm32\include\user32.inc      ; for using MessageBox
include \masm32\include\kernel32.inc    ; for using ExitProcess
include \masm32\include\masm32.inc      ; for using FloatToStr
include \masm32\include\fpu.inc         ; for using FPU instructions


.data
msg_title db "12 [sqrt(25/c) - d + 2]/(b + a - 1)", 0                ; text variable for title of window

msg_message db "Iutputing variables values:", 10, "A=%s ", "B=%s ","C=%s ","D=%s ",10, 10, 0
msg_messageres db "Res = %s", 0
msg_message1 db "Exception situation - Denominator equal to 0",0
msg_message2 db "Exception situation - sqrt value is negative as C is negative",0


A dq  0.5,  -1.3,  32.5,  0.6,  -0.9,  0.5
B dq  5.5,  -2.7,  0.5, 0.3, 1.6,  0.5
C0 dq 5.0,  1.5, 26.5,  23.5,  -3.4,  3.2
D dq  0.75, 0.25,  10.5, 41.5,  2.1,  1.3


tfive dq 25.0
two dq 2.0

.data?

RES dq ?
digbuf db 512 dup(?)
inbuf db 64 dup(?)
finbuf db 64 dup(?)
buffA db 32 dup(?)
buffB db 32 dup(?)
```

```
buffC db 32 dup(?)
buffD db 32 dup(?)
buffRES db 32 dup(?)
.code

main:
finit

mov esi, 0
.WHILE esi <= 5

invoke FloatToStr, A[esi*8], addr buffA
invoke FloatToStr, B[esi*8], addr buffB
invoke FloatToStr, C0[esi*8], addr buffC
invoke FloatToStr, D[esi*8], addr buffD

invoke wsprintf, addr digbuf, addr msg_message, addr buffA, addr buffB, addr buffC, addr buffD
invoke szCatStr, addr digbuf, addr inbuf


; Load values into FPU registers

FLD C0[esi*8]
FTST                    ; Compares the value in the ST(0) register with 0.0
FSTSW AX                    ; Stores the current value of the x87 FPU status word in AX registry
SAHF                    ; Loads the SF, ZF, AF, PF, and CF flags of the EFLAGS register
FSTP RES
jb Error_2

; numerator

fld C0[ESI*8]
fld tfive
fdiv st(0),st(1)                ; ST(0) = ST(0)/ST(1)
fsqrt                   ; ST(0) = sqrt(ST(0))
fld D[ESI*8]                    ; ST(0) = D, ST(1) = sqrt(ST(0))
fsubp st(1),st(0)
```

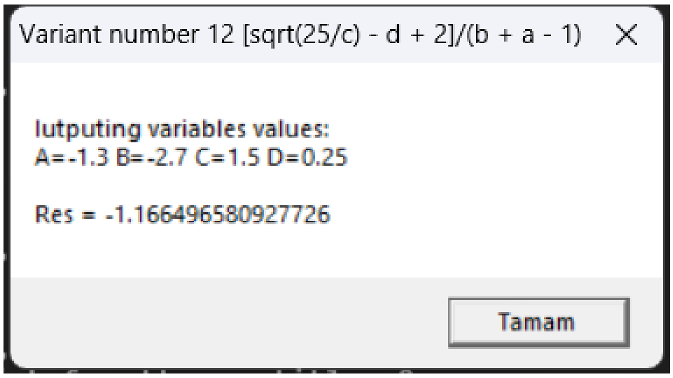
```
fadd two

; denaminator

fld B[ESI*8]                    ; ST(0) = B
fadd A[ESI*8]                   ; ST(0) = B+A
fld1                    ; ST(0) = 1, ST(1) = b+a
fsubp st(1),st(0)               ; ST(1) = b+a-1 and POP ST(0)

FTST                                    ; Compares the value in the ST(0) register with 0.0
FSTSW AX                                ; Stores the current value of the x87 FPU status word in AX
registry
SAHF                                    ; Loads the SF, ZF, AF, PF, and CF flags of the EFLAGS
register
jz Error_1
fdiv                    ; ST(0) = sqrt(25/c)-d+2/b+1-1
fstp  RES                       ; STORE the result and pop ST(0)

invoke FloatToStr2, RES, addr buffRES
;Print texts on the screen
invoke wsprintf, addr finbuf, addr msg_messageres, addr buffRES
Jmp Final

Error_1:
invoke wsprintf, addr finbuf, addr msg_message1
Jmp Final

Error_2:
invoke wsprintf, addr finbuf, addr msg_message2

Final:
invoke szCatStr, addr digbuf, addr finbuf
invoke MessageBox, 0, addr digbuf, addr msg_title, 0
;add esi, 1
inc esi
.endw
invoke ExitProcess, 0                   ; exit of process
end main
```

5. Screenshots of the results of all control examples

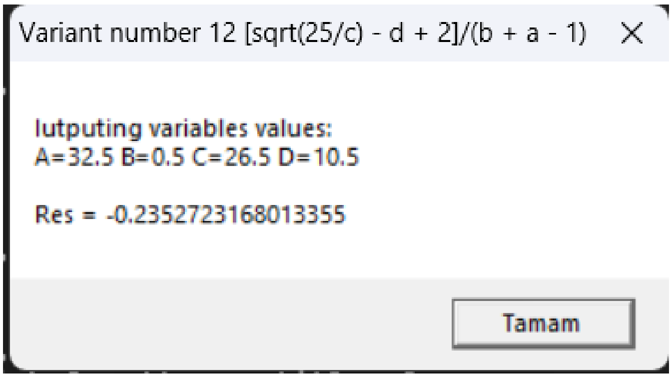
Data set 1



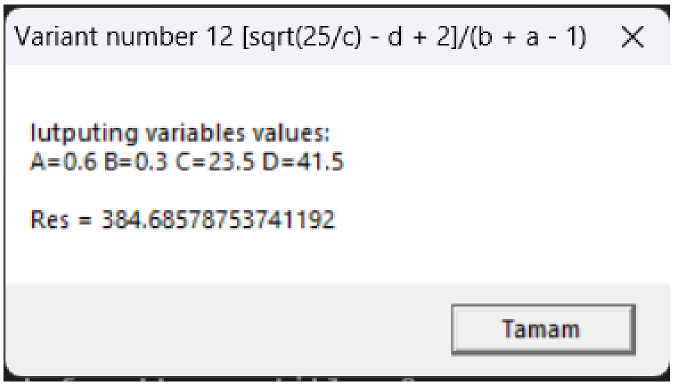
Data set 2



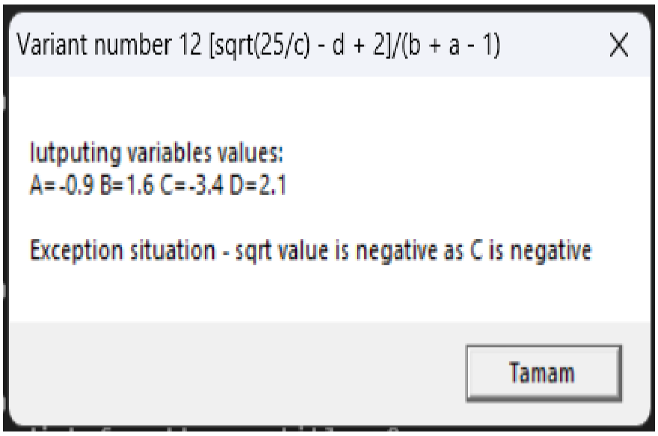
Data set 3



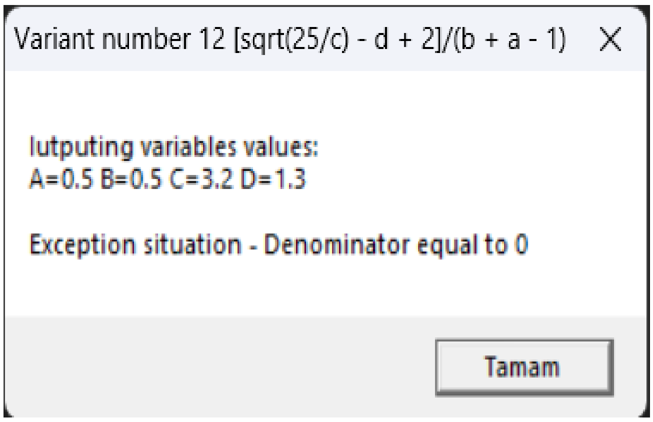
Data set 4



Data set 5



Data set 6



Comparison the results

Control sets	Results of manually calculations	Results of program calculations
Control set 1 a = 0.5; b= 5.5; c = 5.0; d = 0.75	0.69721359549995793928183473374626	0.697213595499958
Control set 2 a = -1.3; b= -2.7; c = 1.5; d = 0.25	-1.166496580927726032732428024902	-1.166496580927726
Control set 3 a = 32.5; b= 0.5; c = 26.5; d = 10.5	-0.23527231680133549435201247209847	-0.2352723168013355
Control set 4 a = 0.6; b= 0.3; c = 23.5; d = 41.5	384.68578753741206592750119026506	384.68578753741192
Control set 5 a = -0.9; b= 1.6; c = -3.4; d = 2.1	Calculations cannot be performed.	Exception situation – sqrt value is negative as C is negative
Control set 6 a = 0.5; b= 0.5; c = 3.2; d = 1.3	Calculations cannot be performed.	Exception situation - Denominator equal to 0

6. Conclusions on the work

The x86 assembly program successfully evaluates the following mathematical equation for the given constants. The FPU instructions are used for floating-point arithmetic; division by zero mistakes needs special attention. The choice to include FPU instructions allows for the efficient handling of floating-point operations. The application assumes that the supplied value is already in radians. The precision of computers that use floating-point representation is poor. The 80-bit extended precision is the default setting for the x86 FPU (Floating Point Unit); nevertheless, values stored in memory may be truncated or rounded. This could lead to a small alteration in the computed result and the actual mathematical result. Using the x86 assembly fsqrt instruction, the arctangent is calculated by a series expansion or other approximation methods. These techniques introduce a certain amount of inaccuracy, and the results may not match those of a very accurate mathematical library. Finally, the program demonstrates how to perform complex floating-point mathematical calculations using x86 assembly code. It serves as a foundation for further investigation into assembly language programming for numerical computing.