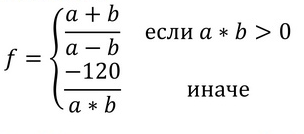
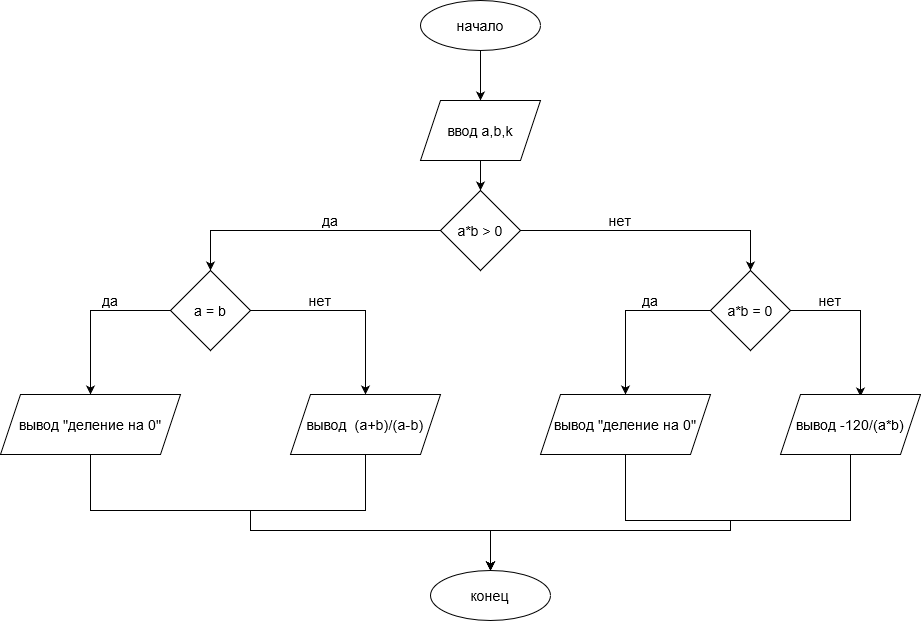
# Задание

Вычилслить целочисленное выражение:

Схема алгоритма

Рисунок 1 — схема алгоритма

Исходный код программы:

; Template for console application

.586

.MODEL flat, stdcall

OPTION CASEMAP:NONE

Include kernel32.inc

Include masm32.inc

IncludeLib kernel32.lib

IncludeLib masm32.lib

.CONST

MsgExit DB "Press Enter to Exit",0AH,0DH,0

MsgZeroMulti DB "Division by zero (a\*b)",0AH,0DH,0

MsgZeroSub DB "Division by zero (a-b)",0AH,0DH,0

MsgResult1 DB "Result f=(a+b)/(a-b)",0AH,0DH,0

MsgResult2 DB "Result f= -120/(a\*b)",0AH,0DH,0

MsgLn DB 0AH,0DH,0

reqA DB 'Input A: ',13,10,0

reqB DB 'Input B: ',13,10,0

.DATA

.DATA?

inbuf DB 100 DUP (?)

buffer DB 10 DUP (?)

outstr DB 10 DUP (?)

A SWORD ?

B SWORD ?

ab SWORD ?

sum\_ab SWORD ?

sub\_ab SWORD ?

f1\_result SWORD ?

f2\_result SWORD ?

.CODE

Start:

;

; Add you statements

;

XOR EAX,EAX

Invoke StdOut,ADDR reqA

Invoke StdIn,ADDR buffer,LengthOf buffer

Invoke StripLF,ADDR buffer

Invoke atol,ADDR buffer ;result in EAX

mov dword ptr A, EAX

Invoke StdOut,ADDR reqB

Invoke StdIn,ADDR buffer,LengthOf buffer

Invoke StripLF,ADDR buffer

Invoke atol,ADDR buffer ;result in EAX

mov dword ptr B, EAX

mov AX, A

imul B ;DX:AX:=AX\*B=A\*B

mov ab, AX

cmp ab, 0

jng negative\_ab

mov AX, A

cmp AX, B ; a=b?

jne f1\_branch

Invoke StdOut,ADDR MsgZeroSub

jmp right\_merge

f1\_branch:

mov AX, A

add AX, B

mov sum\_ab, Ax

mov AX, A

sub AX, B

mov sub\_ab, AX

mov AX, sum\_ab

cwd ;DX:AX = AX

idiv sub\_ab ;AX:=(DX:AX):sub\_ab

mov f1\_result, AX

Invoke StdOut,ADDR MsgResult1

Invoke dwtoa,f1\_result,ADDR outstr

Invoke StdOut,ADDR outstr

Invoke StdOut,ADDR MsgLn

right\_merge:

jmp main\_merge

negative\_ab:

cmp ab, 0

jne f2\_branch

Invoke StdOut,ADDR MsgZeroMulti

jmp left\_merge

f2\_branch:

mov AX, -120

cwd ;DX:AX = AX

idiv ab ;AX:=(DX:AX):(a\*b)

mov f2\_result, AX

Invoke StdOut,ADDR MsgResult2

Invoke dwtoa,f2\_result,ADDR outstr

Invoke StdOut,ADDR outstr

Invoke StdOut,ADDR MsgLn

left\_merge:

main\_merge:

Invoke StdOut,ADDR MsgExit

Invoke StdIn,ADDR inbuf,LengthOf inbuf

Invoke ExitProcess,0

End Start

Таблица 1 — тесты

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат |
| 5 5 | „Division by zero (a-b)“ | „Division by zero (a-b)“ |
| 7 0 | „Division by zero (a\*b)“ | „Division by zero (a\*b)“ |
| 0 9 | „Division by zero (a\*b)“ | „Division by zero (a\*b)“ |
| 8 6 | Result f=(a+b)/(a-b)  7 | Result f=(a+b)/(a-b)  7 |
| -6 2 | Result f= -120/(a\*b)  10 | Result f= -120/(a\*b)  10 |

# Контрольные вопросы

1. Какие машинные команды используют при программировании ветвлений и циклов?

Для программирования ветвлений и итерационных циклов используются команды безусловной(jmp) и условной(je, jne, jg, jng, jl, jnl) передачи управления.

Для программирования счетного цикла используется команда loop.

2. Выделите в своей программе фрагмент, реализующий ветвление. Каково назначение каждой машинной команды фрагмента?

mov AX, A ; запись значения А в регистр АХ

cmp AX, B ; сравнение АХ(A) и B

jne f1\_branch ; если A != B перейти к вычислениям

… ; обработка случая A = B

jmp right\_merge ; перейти к точке слияния ветвей

f1\_branch: ; ветвь вычислений

…

right\_merge: ; точка слияния ветвей

…

3. Чем вызвана необходимость использования команд безусловной передачи управления?

Так как без команд передачи управления все команды в программе выполняются последовательно, то без использования команды безусловной передачи управления могут быть выполнены обе ветви алгоритма.

4. Поясните последовательность команд, выполняющих операции ввода-вывода в вашей программе. Чем вызвана сложность преобразований данных при выполнении операций ввода-вывода?

Invoke StdOut,ADDR reqA ; вывод запроса на ввод А

Invoke StdIn,ADDR buffer,LengthOf buffer ; ввод строки

Invoke StripLF,ADDR buffer ; замена символов конца строки нулем

Invoke atol,ADDR buffer ; преобразование строки в число

mov dword ptr A, EAX ; копирование числа из регистра EAX в переменую А

Сложность преобразования данных обусловлена тем, что считанную строку необходимо сначала проебразовав, заменив код конца строки нулем, затем конвертировать в число и затем скопировать в переменную из регистра EAX, учитывая возможную разницу в типе(размере) переменной и размере регистра EAX.

**Вывод:** в ходе данной лабораторной были изучены команды передачи управления в языке ассемблера, а также реализация с их помощью ветвлений и циклов.