|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | **3** |

|  |  |
| --- | --- |
| **Название:** | Программирование ветвлений и итерационных циклов |
| **Дисциплина:** | Машинно-зависимые языки и основы компиляции |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-42Б |  |  | Д.В. Сулейманов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | М.В. Широкова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

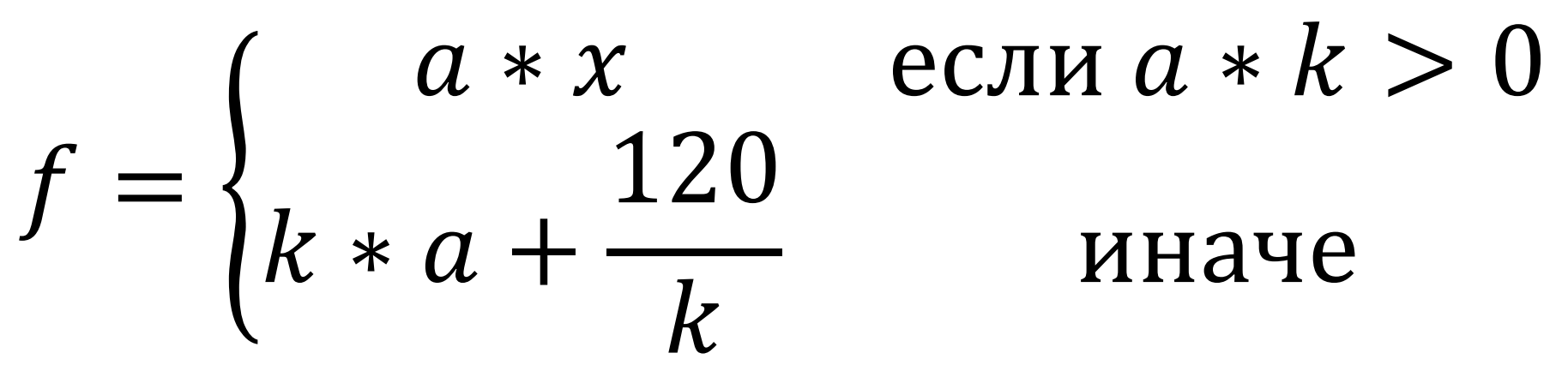
Москва, 2022

**Вариант 2.24**

*Цель работы:*

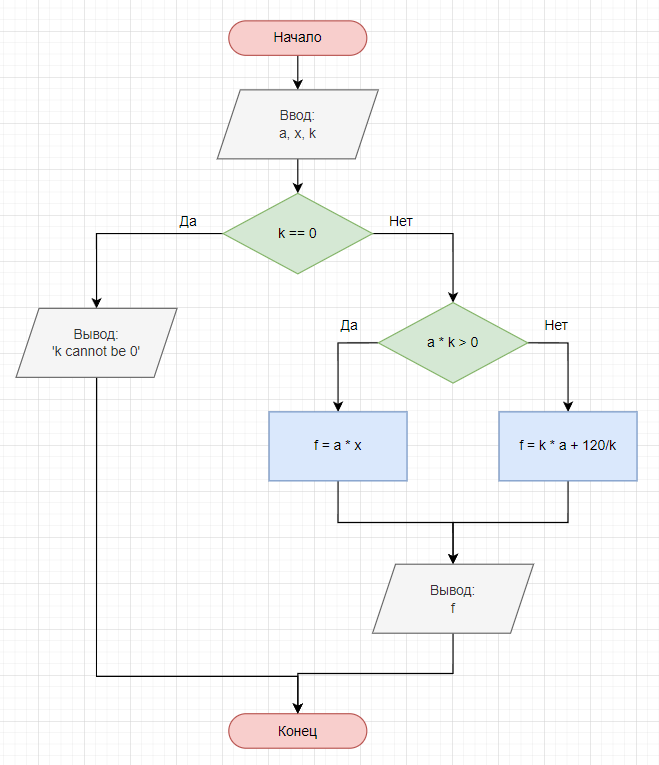
Изучение средств и приёмов программирования ветвлений и итерационных циклов на языке ассемблера.

*Задание:* вычислить целочисленное выражение:



*Ход работы:*

Составим общую схему алгоритма решения данного задания (смотри рисунок 1).



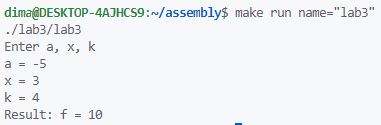
*Рисунок 1. - схема алгоритма программы*

Реализуем схему алгоритма на ассемблере (смотри листинг 1.1).

*Листинг 1.1 – текст программы*

1. %include "./lib64.asm"
3. %macro write\_string 2
4. ; вывод
5. ; 1 - адрес строки, 2 - длина строки
6. mov     rax, 1          ; системная функция 1 (write)
7. mov     rdi, 1          ; дескриптор файла stdout=1
8. mov     rsi, %1         ; адрес выводимой строки
9. mov     rdx, %2         ; длина строки
10. syscall                 ; вызов системной функции
11. %endmacro
13. %macro read\_string 2
14. ; ввод
15. ; 1 - буфер ввода, 2 - длина буфера ввода
16. mov     rax, 0          ; системная функция 0 (read)
17. mov     rdi, 0          ; дескриптор файла stdin=0
18. mov     rsi, %1         ; адрес вводимой строки
19. mov     rdx, %2         ; длина строки
20. syscall                 ; вызов системной функции
21. %endmacro
23. %macro StrToInt 1
24. ; перевод string в integer
25. ; rsi должен содержать адрес строки для преобразования
26. call    StrToInt64          ; вызов процедуры
27. cmp     rbx, 0              ; сравнение кода возврата
28. jne     StrToInt64.Error    ; обработка ошибки
29. mov     %1, eax
30. %endmacro
32. %macro IntToStr 2
33. ; перевод integer в string
34. mov     rsi, %2
35. mov     eax, %1             ; получение числа из памяти
36. cwde
37. call    IntToStr64          ; вызов процедуры
38. cmp     rbx, 0              ; сравнение кода возврата
39. jne     StrToInt64.Error    ; обработка ошибки
40. %endmacro
42. section .data
43. InputMsg db "Enter a, x, k", 10
44. lenInput equ $-InputMsg
45. aInput db "a = "
46. aLen equ $-aInput
47. kInput db "k = "
48. kLen equ $-kInput
49. xInput db "x = "
50. xLen equ $-xInput
51. OutputMsg db "Result: f = "
52. lenOutput equ $-OutputMsg
53. ErrorMsg db "k cannot be 0", 10
54. lenError equ $-ErrorMsg
56. section .bss
57. InBuf resb 10
58. lenIn equ $-InBuf
59. OutBuf resb 10
60. lenOut equ $-OutBuf
61. a resw 1
62. x resw 1
63. k resw 1
64. f resd 1
66. section .text
67. global \_start
69. \_start:
70. ; ввод
72. write\_string InputMsg, lenInput
74. write\_string aInput, aLen
75. read\_string InBuf, lenIn
76. StrToInt [a]
78. write\_string xInput, xLen
79. read\_string InBuf, lenIn
80. StrToInt [x]
82. write\_string kInput, kLen
83. read\_string InBuf, lenIn
84. StrToInt [k]
86. mov ax, [k]
87. cmp ax, 0
88. je catch
90. ; вычисления
92. mov ax, [a]
93. mov bx, [k]
94. imul bx
95. cmp dx, 0
96. jge greater
97. less:
98. mov ax, [a]
99. imul bx
100. mov cx, dx
101. shl ecx, 16
102. mov cx, ax
103. mov ax, 120
104. xor dx, dx
105. idiv bx
106. add cx, ax
107. mov eax, ecx
108. jmp continue
109. greater:
110. cmp ax, 0
111. je less
112. mov ax, [a]
113. mov bx, [x]
114. imul bx
115. shl edx, 16
116. mov dx, ax
117. mov eax, edx
118. continue:
119. mov [f], eax
121. ; вывод
123. xor rbx, rbx
124. write\_string OutputMsg, lenOutput
125. IntToStr [f], OutBuf
126. mov RBX, RAX
127. write\_string OutBuf, RBX
128. jmp exit
130. catch:
131. write\_string ErrorMsg, lenError
133. exit:
134. mov     rax, 60         ; системная функция 60 (exit)
135. xor     rdi, rdi        ; return code 0
136. syscall                 ; вызов системной функции

Проверим программу на произвольных данных. Пример выполнения программы показан на рисунке 2. Результаты тестирования программы представлены в таблице 1.



*Рисунок 2 – пример выполнения программы*

*Таблица 1 – результаты тестирования*

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Полученный результат** |
| a = -5, x=3, k = 4 | f = 10 | f = 10 |
| a = 0, x=1, k = 10 | f = 12 | f = 12 |
| a = 7, x=3, k = 1 | f = 21 | f = 21 |
| a = 1, x=1, k = 0 | k cannot be 0 | k cannot be 0 |

*Контрольные вопросы*

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

Ветвление осуществляется про помощи команды проверки *cmp* и команд перехода по условию: je, jnl, jge и других. Принцип их работы заключается в считывании регистра флагов и переходе по указанной метке.

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

Ветвление реализовано на строках 95-119 листинга 1. Здесь при помощи команды *cmp* сравнивается содержимое регистра DX (регистра, в котором находятся старшие байты, полученные при вычислении операции a \* x)

1. Чем вызвана необходимость использования команд безусловной передачи управления?

Безусловная передача управления используется, чтобы перейти на команды, следующие за ветвлением, а не на те, которые следует пропустить.

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

При чтении пользовательского ввода строки сначала попадают в буферную переменную в памяти, из которой они сразу же извлекаются и преобразовываются в числа. Сложность вызвана именно тем, что для ввода и вывода используются символьные обозначения цифр, а для вычислений применяются внутренние представления чисел.

***Вывод:***в рамках лабораторной работы были изучены программирование ветвлений на языке ассемблера и их применение при решении вычислительных задач.