|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | **ИУК4 «Программное обеспечение ЭВМ,** |
| **информационные технологии»** | |

**ДОМАШНЯЯ РАБОТА**

**«Обработка двухмерных массивов целых чисел»**

**ДИСЦИПЛИНА: «Системное программирование»**



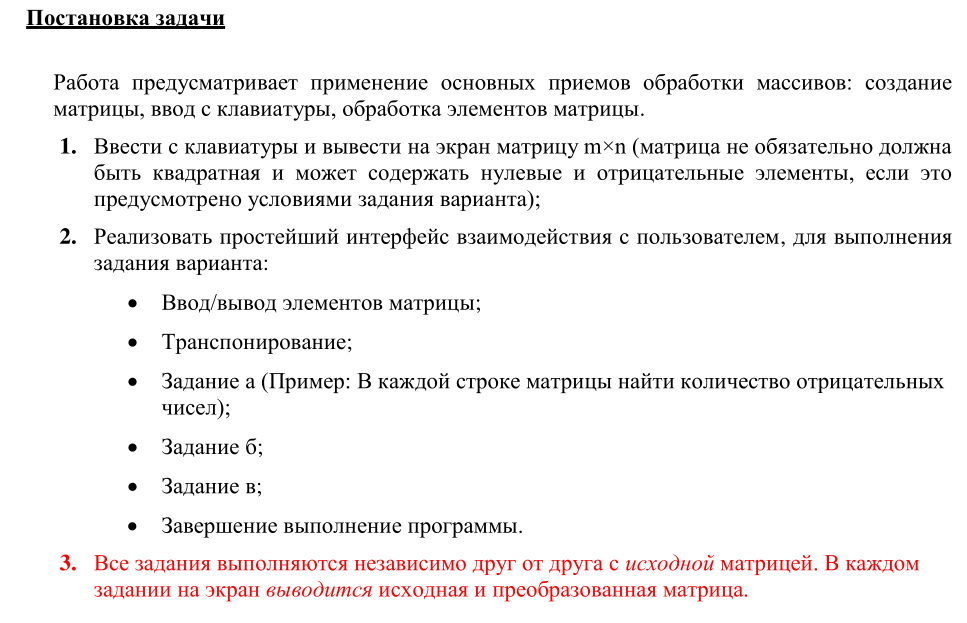
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-31Б | |  |  | ( | Суриков Н. С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Амеличева К. А. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

**Цель:** практическое овладение навыками разработки программного кода на языке Ассемблер. Обработка массивов.

**Задачи:**

1. Изучение основных приемов обработки массивов:
   1. ввод-вывод
   2. доступ к элементам массива
   3. транспонирование
   4. выполнение типовых операций



**Вариант №27**

Дана матрица.

а) В каждой строке матрицы найти максимальный элемент. Найти строку, которая содержит наименьший максимальный элемент.

б) Проверить, состоит ли матрица только из элементов больших введенного значения n и меньших к (к>0).

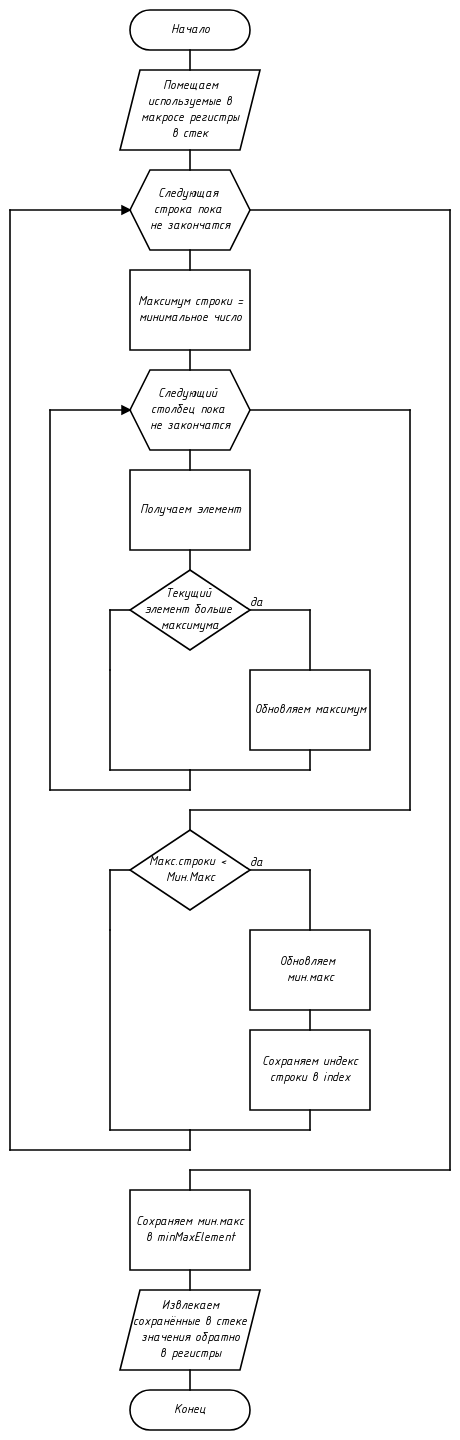
в) Определить сумму отрицательных элементов каждой строки и поместить на место первого элемента.

**Описание макросов:**

|  |  |  |
| --- | --- | --- |
| Номер | Название | Страница |
| 1 | mReadMatrix | 11 |
| 2 | mWriteMatrix | 11 |
| 3 | mTransposeMatrix | 12 |
| 4 | mFindMinMaxInRows | 13 |
| 5 | mCheckMatrixRange | 13 |
| 6 | mRowNegativeSumsCopy | 14 |
| 7 | mClear | 11 |
| 8 | mWriteStr | 10 |
| 9 | mReadAX | 9 |
| 10 | mWriteAX | 10 |

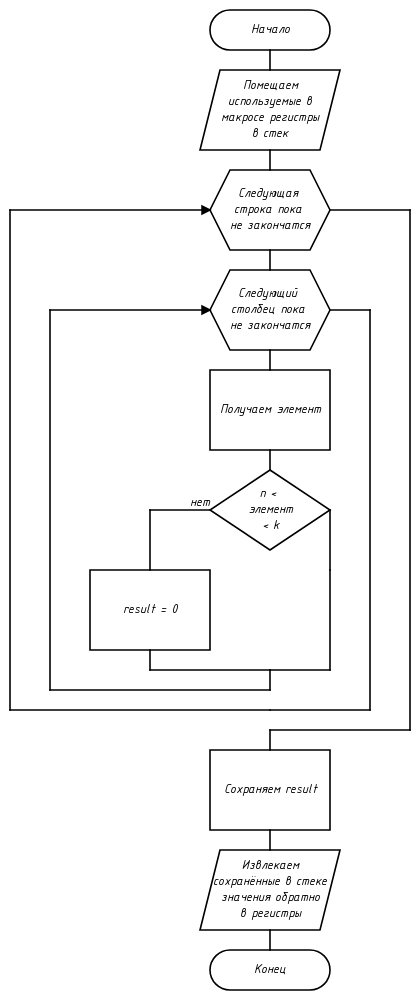
1. mReadMatrix - Макрос для ввода матрицы чисел. В начале помещаем все используемые регистры в стек. Затем обнуляем смещение по строкам и устанавливаем счётчик строк. Внешний цикл проходит по строкам, а внутренний — по столбцам. Для каждого элемента вызываем макрос mReadAX, чтобы считать значение в регистр AX и затем сохраняем его в соответствующую ячейку матрицы. После завершения ввода каждого столбца выводим строку на экран с помощью mWriteStr. Увеличиваем смещение по строкам и повторяем процесс до тех пор, пока не будут обработаны все строки.
2. mWriteMatrix - Макрос для вывода матрицы чисел. Сначала помещаем все используемые регистры в стек. Затем обнуляем смещение по строкам и устанавливаем счётчик строк. Внешний цикл проходит по строкам, а внутренний — по столбцам. Для каждого элемента загружаем значение из матрицы в регистр AX и вызываем макрос mWriteAX для его вывода на экран. Между элементами строки добавляем табуляцию с помощью mWriteStr, а после вывода всей строки — переводим курсор на новую строку. Процесс продолжается до тех пор, пока не будут обработаны все строки.
3. mTransposeMatrix - Макрос для транспонирования матрицы. Начинаем с помещения всех используемых регистров в стек. Обнуляем смещение по строкам и устанавливаем счётчик строк. Внешний цикл проходит по строкам оригинальной матрицы, а внутренний — по столбцам. Для каждого элемента вычисляем его новое положение в результирующей матрице, сохраняя элемент в стеке, а затем перемещаем его в соответствующую ячейку результирующей матрицы. После обработки всех элементов строки переходим к следующей строке. В конце восстанавливаем значения регистров из стека.
4. mFindMinMaxInRows - Макрос для поиска максимального элемента в каждой строке матрицы. Начинаем с помещения всех используемых регистров в стек. Инициализируем переменные для хранения минимального максимума и текущего максимума строки. Внешний цикл проходит по строкам, а внутренний — по столбцам. Для каждого элемента сравниваем его с текущим максимумом строки и обновляем максимум при необходимости. После обработки всех столбцов строки сравниваем найденный максимум с минимальным максимумом и обновляем его, если это необходимо. В конце сохраняем минимальный максимум в указанную переменную.
5. mCheckMatrixRange - Макрос для проверки, соответствуют ли все элементы матрицы заданному диапазону. Начинаем с помещения всех используемых регистров в стек. Инициализируем переменную результата, предполагая, что матрица соответствует условию. Внешний цикл проходит по строкам, а внутренний — по столбцам. Для каждого элемента проверяем, находится ли он в заданном диапазоне. Если хотя бы один элемент не соответствует условию, устанавливаем результат в 0. В конце сохраняем результат в указанную переменную.
6. mRowNegativeSumsCopy - Макрос для копирования исходной матрицы и вычисления сумм отрицательных элементов в каждой строке. Сначала помещаем все используемые регистры в стек. Копируем элементы исходной матрицы в результирующую. Затем обрабатываем каждую строку: для каждого элемента проверяем, является ли он отрицательным, и если да, добавляем его к сумме. Если сумма отрицательных элементов не равна нулю, записываем её в начало строки результирующей матрицы. В конце восстанавливаем значения регистров из стека.
7. mClear - Макрос для очистки экрана. В начале помещаем все используемые регистры в стек. Затем устанавливаем режим видеоадаптера, чтобы использовать 16 цветов. Подготавливаем код для очистки экрана, задавая цвет фона и шрифта, а также указывая количество строк для очистки и адрес видеопамяти. Вызываем прерывание BIOS для выполнения очистки экрана. После этого устанавливаем курсор в позицию (0, 0). В конце восстанавливаем значения регистров из стека.
8. mWriteStr - Макрос для вывода строки на экран. В начале помещаем регистры AX и DX в стек для сохранения их значений. Устанавливаем функцию прерывания DOS для вывода строки, используя регистр DX, который содержит адрес строки, переданной в макрос. Вызываем прерывание 21h для вывода строки на экран. После завершения вывода восстанавливаем значения регистров из стека.
9. mReadAX - Макрос для обработки ввода десятичного числа. Для начала помещаем все используемые регистры в стек. Затем вводим число, которое считываем как массив символов, и переходим на новую строку. Если число отрицательное, смещаем указатель по массиву на 1. Затем переводим каждый символ массива в соответствующее число, отняв от него 30h, и, умножив предыдущею сумму на 10, прибавить число к этой сумме. Выполняем предыдущие действии пока не достигнем конца массива. Если число отрицательное, инвертируем его. В конце вытаскиваем из стека значения регистров.
10. mWriteAX - макрос для вывода десятичного числа. Для начала помещаем все используемые регистры в стек. Затем проверяем число на знак. Если число отрицательное, выводим знак минус. Затем переводим число в строку следующим образом: делим число на десять, переводим остаток в символ и помещаем полученный символ в стек. Выполняем предыдущее действие, пока число не станет равным 0. После этого, достаем из стека символ и выводим его. Повторяем столько раз, сколько делили число на 10. В конце макроса вытаскиваем из стека значения регистров.

**Блок-схемы:**

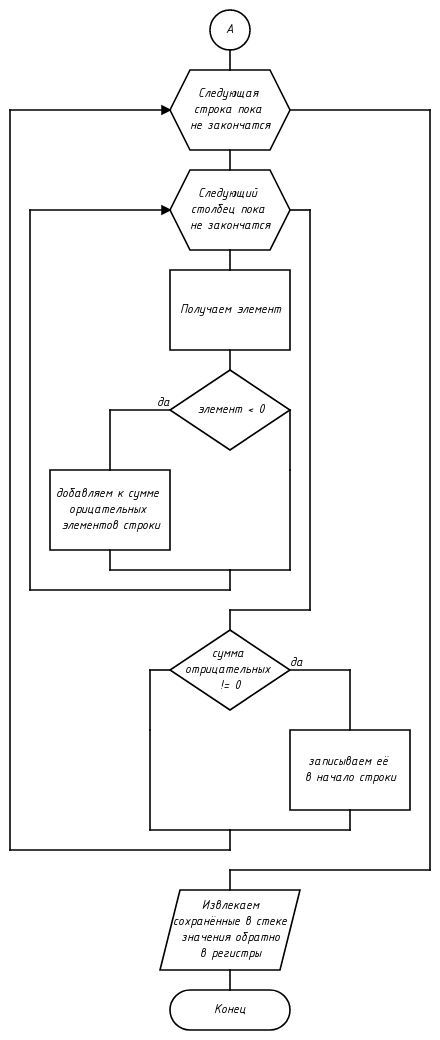
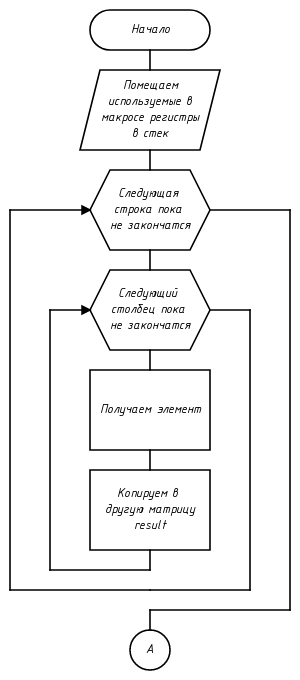
****

*mFindMinMaxInRows*

*mCheckMatrixRange*

**

*mRowNegativeSumsCopy*

**

**Листинг программы:**

1 include macroses.asm *; Подключение файла с макросами*

2

3 .model small

4 .stack 100h

5 .486 *; Включает сборку инструкций для процессора 80386*

6

7

8 *;Дана матрица.*

9 *;а) В каждой строке матрицы найти максимальный элемент. Найти строку, которая содержит наименьший максимальный элемент.*

10 *;б) Проверить, состоит ли матрица только из элементов больших введенного значения n и меньших к (к>0).*

11 *;в) Определить сумму отрицательных элементов каждой строки и поместить на место первого элемента.*

12

13 *; Макрос ввода 10-чного числа в регистр АХ*

14 mReadAX macro buffer, sizee

15 local input, startOfConvert, endOfConvert

16 push bx *; Данные в стек*

17 push cx

18 push dx

19 input:

20 mov [buffer], sizee *; Задаём размер буфера*

21 mov dx, offset [buffer] *; Поместить в регистр dx строку по адресу buffer*

22 mov ah, 0Ah *; Чтение строки из консоли*

23 int 21h *; Прерывание DOS*

24

25 mov ah, 02h *; Вывод символа на экран*

26 mov dl, 0Dh *; Перевод каретки на новую строку*

27 int 21h *; Прерывание DOS*

28

29 mov ah, 02h *; Вывод символа на экран*

30 mov dl, 0Ah *; Чтение строки из консоли*

31 int 21h *; Прерывание DOS*

32

33 xor ah, ah *; Очистка регистра ah*

34 cmp al, [buffer][1] *; Проверка на пустую строку*

35 jz input *; Переход, если строка пустая*

36

37 xor cx, cx *; Очистка регистра cx*

38 mov cl, [buffer][1] *; инициализация переменной-счётчика*

39

40 xor ax, ax *; Очистка регистра ax*

41 xor bx, bx *; Очистка регистра bx*

42 xor dx, dx *; Очистка регистра dx*

43

44 mov bx, offset [buffer][2] *; Поместить начало строки в регистр bx*

45 cmp [buffer][2], '-' *; Проверка на знак числа*

46 jne startOfConvert *; Переход, если число неотрицательное*

47 inc bx *; Инкремент регистра bx*

48 dec cl *; Декремент регистра-счетчика cl*

49 startOfConvert:

50 mov dx, 10 *; Поместить в регистр ax число 10*

51 mul dx *; Умножение на 10 перед сложением с младшим разрядом*

52 cmp ax, 8000h *; Проверка числа на выход за границы*

53 jae input *; Переход, если число выходит за границы*

54 mov dl, [bx] *; Поместить в регистр dl следующий символ*

55 sub dl, '0' *; Перевод его в числовой формат*

56 add ax, dx *; Прибавляем его к конечному результату*

57 cmp ax, 8000h *; Проверка числа на выход за границы*

58 jae input *; Переход, если число выходит за границы*

59 inc bx *; Переход к следующему символу*

60 loop startOfConvert *; Цикл*

61 cmp [buffer][2], '-' *; Проверка на знак числа*

62 jne endOfConvert *; Переход, если число неотрицательное*

63 neg ax *; Инвертирование числа*

64 endOfConvert:

65 pop dx *; Данные из стека*

66 pop cx

67 pop bx

68 endm

69

70 *; Макрос вывода 10-чного числа из регистра AX*

71 mWriteAX macro

72 local convert, write

73 push ax *; Данные в стек*

74 push bx

75 push cx

76 push dx

77 push di

78

79 mov cx, 10 *; cx - основание системы счисления*

80 xor di, di *; di - количество цифр в числе*

81 or ax, ax *; Проверка числа на ноль*

82 jns convert *; Переход, если число положительное*

83 push ax *; Регистр ax в стек*

84 mov dx, '-' *; Поместить в регистр dx символ '-'*

85 mov ah, 02h *; Вывод символа на экран*

86 int 21h *; Прерывание DOS*

87 pop ax *; Регистр ax из стека*

88 neg ax *; Инвертирование отрицательного числа*

89 convert:

90 xor dx, dx *; Очистка регистра dx*

91 div cx *; После деления dl = остатку от деления ax на cx*

92 add dl, '0' *; Перевод в символьный формат*

93 inc di *; Увеличение количества цифр в числе на 1*

94 push dx *; Регистр dx в стек*

95 or ax, ax *; Проверка числа на ноль*

96 jnz convert *; Переход, если число не равно нулю*

97 write:

98 pop dx *; dl = очередной символ*

99 mov ah, 02h *; Вывод символа на экран*

100 int 21h *; Прерывание DOS*

101 dec di *; Повторение, пока di != 0*

102 jnz write

103

104 pop di *; Данные из стека*

105 pop dx

106 pop cx

107 pop bx

108 pop ax

109 endm

110

111 *; Макрос вывода строки*

112 mWriteStr macro string

113 push ax

114 push dx

115

116 mov ah, 09h

117 mov dx, offset string

118 int 21h

119

120 pop dx

121 pop ax

122 endm

123

124 mClear macro start *;Макрос очистки экрана*

125 push ax

126 push bx

127 push cx

128 push dx

129

130 mov ah, 10h

131 mov al, 3h

132 int 10h *; Включение режима видеоадаптора с 16-ю цветами*

133

134 mov ax,0600h *; Подготавливает код для очистки экрана (функция 0).*

135 mov bh, 5Ah *; Устанавливает цвет фона и шрифта*

136 mov cx, start *; Указывает количество строк для очистки (все).*

137 mov dx, 184FH *; Указывает адрес экрана (184FH — адрес видеопамяти).*

138 int 10h *; Вызывает прерывание BIOS для выполнения очистки экрана.*

139

140 mov dx, 0 *; dh - строка, dl - столбец*

141 mov bh, 0 *; Номер видео-страницы*

142 mov ah, 02h *; 02h - функция установки позиции курсора*

143 int 10h *; Устанавливаем курсор на позицию (0, 0)*

144

145 pop dx

146 pop cx

147 pop bx

148 pop ax

149 endm

150

151 mReadMatrix macro matrix, row, col

152 local rowLoop, colLoop

153 JUMPS *; Директива, делающая возможным большие прыжки*

154 push bx *; Сохранение регистров, используемых в макросе, в стек*

155 push cx

156 push si

157

158 xor bx, bx *; Обнуляем смещение по строкам*

159 mov cx, row

160

161 rowLoop: *; Внешний цикл, проходящий по строкам*

162 push cx

163 xor si, si *; Обнуляем смещение по столбцам*

164 mov cx, col

165 colLoop: *; Внутренний цикл, проходящий по столбцам*

166 mReadAX buffer 4 *; Макрос ввода значения регистра AX с клавиатуры*

167 mov matrix[bx][si], ax

168 add si, 2 *; Переходим к следующему элементу (размером в слово)*

169 loop colLoop

170 mWriteStr endl *; Макрос вывода строки на экран*

171 add bx, col *; Увеличиваем смещение по строкам*

172 add bx, col *; (дважды, так как размер каждого элемента - слово)*

173 pop cx

174 loop rowLoop

175

176 pop si *; Перенос сохранённых значений обратно в регистры*

177 pop cx

178 pop bx

179 NOJUMPS *; Прекращение действия директивы JUMPS*

180 endm

181

182

183 mWriteMatrix macro matrix, row, col

184 local rowLoop, colLoop

185 push ax *; Сохранение регистров, используемых в макросе, в стек*

186 push bx

187 push cx

188 push si

189

190 xor bx, bx *; Обнуляем смещение по строкам*

191 mov cx, row

192 rowLoop: *; Внешний цикл, проходящий по строкам*

193 push cx

194 xor si, si *; Обнуляем смещение по столбцам*

195 mov cx, col

196

197 colLoop: *; Внутренний цикл, проходящий по столбцам*

198 mov ax, matrix[bx][si] *; bx - смещение по строкам, si - по столбцам*

199 mWriteAX *; Макрос вывода значения регистра AX на экран*

200 *; Вывод текущего элемента матрицы*

201 xor ax, ax

202 mWriteStr tab *; Макрос вывода строки на экран*

203 *; Вывод на экран табуляции, разделяющей элементы строки*

204 add si, 2 *; Переходим к следующему элементу (размером в слово)*

205 loop colLoop

206

207 mWriteStr endl *; Макрос вывода строки на экран*

208 *; Перенос курсора и каретки на следующую строку*

209 add bx, col *; Увеличиваем смещение по строкам*

210 add bx, col *; (дважды, так как размер каждого элемента - слово)*

211 pop cx

212 loop rowLoop

213

214 pop si *; Перенос сохранённых значений обратно в регистры*

215 pop cx

216 pop bx

217 pop ax

218 endm

219

220 mTransposeMatrix macro matrix, row, col, resMatrix

221 local rowLoop, colLoop

222 push ax *; Сохранение регистров, используемых в макросе, в стек*

223 push bx

224 push cx

225 push di

226 push si

227 push dx

228 xor di, di *; Обнуляем смещение по строкам*

229 mov cx, row

230 rowLoop: *; Внешний цикл, проходящий по строкам*

231 push cx

232 xor si, si *; Обнуляем смещение по столбцам*

233 mov cx, col

234 colLoop: *; Внутренний цикл, проходящий по столбцам*

235 mov ax, col

236 mul di *; Устанавливаем смещение по строкам*

237 add ax, si *; Устанавливаем смешение по столбцам*

238 mov bx, ax

239 mov ax, matrix[bx]

240 push ax *; Заносим текущий элемент в стек*

241 mov ax, row

242 mul si *; Устанавливаем смещение по строкам*

243 add ax, di *; Устанавливаем смешение по столбцам*

244 *; (смещения по строкам и столбцам меняются*

245 *; местами по сравнению с оригинальной матрицей)*

246 mov bx, ax

247 pop ax

248 mov resMatrix[bx], ax *; Заносим в новую матрицу элемент, сохранённый в стеке*

249 add si, 2 *; Переходим к следующему элементу (размером в слово)*

250 loop colLoop

251 add di, 2 *; Переходим к следующей строке*

252 pop cx

253 loop rowLoop

254 pop dx *; Перенос сохранённых значений обратно в регистры*

255 pop si

256 pop di

257 pop cx

258 pop bx

259 pop ax

260 endm

261

262 mFindMinMaxInRows macro matrix, rows, cols, index, minMaxElement

263 local rowLoop, colLoop, skipUpdate, nextRow

264

265 push ax

266 push bx

267 push cx

268 push dx

269 push si

270 push di

271

272 mov di, 7FFFh *; Инициализируем минимальный максимум (очень большое значение)*

273 xor dx, dx *; Текущий максимум строки*

274 xor ax, ax *; Сброс регистра ax*

275 xor bx, bx *; Смещение строки*

276 xor cx, cx *; Сброс счетчика строк*

277

278 mov cx, rows *; cx = количество строк*

279 xor si, si *; si = индекс текущей строки*

280

281 rowLoop:

282 push cx *; Сохраняем счетчик строк*

283 mov bx, si *; bx = номер строки*

284 imul bx, cols *; bx = смещение начала строки (учитываем количество столбцов)*

285 shl bx, 1 *; Умножаем на 2, так как элементы занимают по 2 байта*

286 mov cx, cols *; cx = количество столбцов*

287 mov dx, 8000h *; Устанавливаем максимум строки в минимальное значение*

288

289 colLoop:

290 mov ax, matrix[bx] *; Загружаем текущий элемент*

291 cmp ax, dx *; Сравниваем с текущим максимумом строки*

292 jle nextCol *; Если максимум не обновляется, перейти*

293 mov dx, ax *; Обновляем максимум строки*

294

295 nextCol:

296 add bx, 2 *; Переходим к следующему элементу строки*

297 loop colLoop *; Повторяем для всех элементов строки*

298

299 cmp dx, di *; Сравниваем максимум строки с минимальным максимумом*

300 jge skipUpdate *; Если не меньше, пропускаем обновление*

301 mov di, dx *; Обновляем минимальный максимум*

302 mov [index], si *; Сохраняем индекс текущей строки*

303

304 skipUpdate:

305 inc si *; Переход к следующей строке (индекс строки)*

306 pop cx *; Восстанавливаем счетчик строк*

307 loop rowLoop *; Переход к следующей строке*

308

309 mov [minMaxElement], di *; Сохраняем минимальный максимум*

310

311 pop di

312 pop si

313 pop dx

314 pop cx

315 pop bx

316 pop ax

317 endm

318

319 mCheckMatrixRange macro matrix, rows, cols, n, k, result

320 local rowLoop, colLoop, checkFail, nextElement, endCheck

321

322 push ax

323 push bx

324 push cx

325 push dx

326 push si

327 push di

328

329 xor si, si *; si = 0, для работы с индексами матрицы*

330 xor di, di *; di = 0, начальный результат (будет 1, если условие выполнено)*

331 mov di, 1 *; Предполагаем, что матрица соответствует условию*

332

333 mov cx, rows *; cx = количество строк*

334

335 rowLoop:

336 push cx *; Сохраняем счетчик строк*

337 mov bx, si *; bx = номер строки*

338 imul bx, cols *; bx = смещение начала строки (учет количества столбцов)*

339 shl bx, 1 *; Умножаем на 2, так как элементы занимают 2 байта*

340 mov cx, cols *; cx = количество столбцов*

341

342 colLoop:

343 mov ax, matrix[bx] *; Загружаем текущий элемент*

344 cmp ax, n *; Сравниваем с n*

345 jle checkFail *; Если элемент <= n, условие не выполнено*

346 cmp ax, k *; Сравниваем с k*

347 jge checkFail *; Если элемент >= k, условие не выполнено*

348

349 nextElement:

350 add bx, 2 *; Переход к следующему элементу строки*

351 loop colLoop *; Повторяем для всех элементов строки*

352 pop cx *; Восстанавливаем счетчик строк*

353 inc si *; Переход к следующей строке*

354 loop rowLoop *; Переход к следующей строке*

355 jmp endCheck *; Завершаем проверку*

356

357 checkFail:

358 mov di, 0 *; Условие не выполнено*

359 jmp endCheck *; Прерываем проверку*

360

361 endCheck:

362 mov [result], di *; Сохраняем результат: 1 = соответствует, 0 = не соответствует*

363

364 pop di

365 pop si

366 pop dx

367 pop cx

368 pop bx

369 pop ax

370 endm

371

372 mRowNegativeSumsCopy macro matrix, rows, cols, result

373 local copyMatrix, processRow, processCol, endMacro

374

375 push ax

376 push bx

377 push cx

378 push dx

379 push si

380 push di

381

382 *; Копируем исходную матрицу в result*

383 xor si, si *; si = индекс исходной матрицы*

384 xor di, di *; di = индекс для копии*

385 mov cx, rows *; cx = количество строк*

386 imul cx, cols *; cx = общее количество элементов*

387 shl cx, 1 *; Умножаем на 2 (размер word)*

388

389 copyMatrix:

390 mov ax, matrix[si] *; Загружаем элемент исходной матрицы*

391 mov result[di], ax *; Копируем в result*

392 add si, 2 *; Переход к следующему элементу*

393 add di, 2 *; Переход к следующей ячейке в result*

394 loop copyMatrix *; Повторяем для всех элементов*

395

396 *; Обрабатываем строки для вычисления сумм отрицательных*

397 xor si, si *; Индекс начала текущей строки в result*

398 xor di, si *; Индекс записи результата (совпадает с si)*

399 mov cx, rows *; cx = количество строк*

400

401 processRow:

402 push cx *; Сохраняем счётчик строк*

403 xor dx, dx *; dx = сумма отрицательных элементов*

404 mov bx, si *; bx = начальный адрес текущей строки*

405 mov cx, cols *; cx = количество столбцов в строке*

406

407 processCol:

408 mov ax, result[bx] *; Загружаем элемент текущей строки*

409 cmp ax, 0 *; Проверяем, является ли элемент отрицательным*

410 jge skipAdd *; Если >= 0, пропускаем*

411 add dx, ax *; Добавляем к сумме отрицательных*

412

413 skipAdd:

414 add bx, 2 *; Переход к следующему элементу в строке*

415 loop processCol *; Повторяем для всех столбцов*

416

417 cmp dx, 0

418 jz endRow *; Если сумма отрицательных не ноль, продолжаем обработку*

419 *; Записываем сумму отрицательных элементов в начало строки*

420 mov result[si], dx *; Записываем сумму в начало текущей строки*

421 endRow:

422 *; Переход к следующей строке*

423 add si, cols *; Смещение для следующей строки*

424 add si, cols *; Умножаем на 2 (размер word)*

425

426 pop cx *; Восстанавливаем счётчик строк*

427 loop processRow *; Обрабатываем следующую строку*

428

429 endMacro:

430 pop di

431 pop si

432 pop dx

433 pop cx

434 pop bx

435 pop ax

436 endm

437 .data

438 buffer db 20 dup(0)

439 endl db 13, 10, '$'

440 tab db 09, '$'

441 space db ' $'

442

443 mes\_rows db 'Enter the rows of matrix: $'

444 mes\_cols db 'Enter the cols of matrix: $'

445 mes\_inpElements db 'Enter matrix elements element by element: ', 13, 10, '$'

446

447 mes\_menu db 'To control the menu, press the ', 'corresponding key on the keyboard', 13, 10

448 db '1. Enter matrix from keyboard', 13, 10

449 db '2. Display matrix', 13, 10

450 db '3. View transposed matrix', 13, 10

451 db '4. Find Min of Max elements in rows', 13, 10

452 db '5. Check matrix consists only of elems of larger n and smaller k', 13, 10

453 db '6. Find the sum of the negative elements of line and place it on first element', 13, 10

454 db '0. Exit the program', 13, 10, '$'

455

456 mes\_choice db '>> $'

457

458 mes\_matrix db '>>Matrix<<', 13, 10, '$'

459 mes\_trans\_matrix db '>>Tronspose Matrix<<', 13, 10, '$'

460 mes\_edited\_matrix db '>>Edited matrix<<', 13, 10, '$'

461

462 mes\_minMaxIndex db 'Min of Max element in rows: $'

463 mes\_minMaxElement db 'Min of Max elements: $'

464

465 mes\_n db 'Enter n: $'

466 mes\_k db 'Enter k: $'

467 mes\_flag\_false db 'Matrix consists not only of elements of larger entered values n and smaller k$'

468 mes\_flag\_true db 'Matrix consists only of elements of larger entered values n and smaller k$'

469

470

471

472 rows dw 1

473 cols dw 1

474 matrix dw 100 dup(0)

475 trans\_matrix dw 100 dup(0)

476

477

478 index dw ?

479 minMaxElement dw ?

480

481 n dw ?

482 k dw ?

483 flag dw ?

484

485 copy\_matrix dw 100 dup(0)

486

487

488 .code

489 start:

490 mov ax, @data

491 mov ds, ax

492

493 mClear 0000b *; Макрос очистки экрана и установки вида окна*

494 mWriteStr mes\_menu *; Макрос вывода строки на экран*

495 mWriteStr endl

496

497 menu: *; Вывод на экран меню, а также осуществление выбора следующего пункта программы*

498 mov ah, 01h

499 int 16h *; Ожидание нажатия символа и получение его значения в al*

500

501 cmp al, "0"

502 je exit

503 cmp al, "1"

504 je consoleInput

505 cmp al, "3"

506 je transposeMatrix

507 cmp al, "4"

508 je task1

509 cmp al, "5"

510 je task2

511 cmp al, "6"

512 je task3

513 writeMatrix: *; Вывод элементов матрицы на экран*

514 mClear 0000b *; Макрос очистки экрана и установки вида окна*

515 mWriteStr mes\_menu *; Макрос вывода строки на экран*

516 mWriteStr endl

517 mov ah, 02h

518 mov dx, 0900h

519 mov bh, 0

520 int 10h

521 mWriteStr mes\_matrix

522 mWriteMatrix matrix, rows, cols

523

524 mov ah, 07h *; Задержка экрана*

525 int 21h

526 jmp menu

527

528 consoleInput: *; Ввод элементов матрицы из консоли*

529 mClear 0000b *; Макрос очистки экрана и установки вида окна*

530 mWriteStr mes\_rows *; Макрос вывода строки на экран*

531 mReadAX buffer 2 *; Макрос ввода значения регистра AX с клавиатуры*

532 mov rows, ax

533

534 xor ax,ax

535

536 mWriteStr mes\_cols *; Макрос вывода строки на экран*

537 mReadAX buffer 2

538 mov cols, ax

539

540 mWriteStr endl *; Макрос вывода строки на экран*

541 mWriteStr mes\_inpElements *; Макрос вывода строки на экран*

542 mReadMatrix matrix, rows, cols

543 jmp writeMatrix

544

545 transposeMatrix: *; Получение и вывод транспонированной матрицы*

546 mTransposeMatrix matrix, rows, cols, trans\_matrix

547 mWriteStr endl

548 mWriteStr mes\_trans\_matrix

549 mWriteMatrix trans\_matrix, cols, rows

550

551 mov ah, 07h *; Задержка экрана*

552 int 21h

553 jmp menu

554 task1: *; Перемещение нулей в начало строки*

555 mFindMinMaxInRows matrix, rows, cols, index, minMaxElement

556

557 mWriteStr endl

558 mWriteStr mes\_minMaxIndex

559 mov ax, index

560 mWriteAX

561

562 xor ax, ax

563

564 mWriteStr endl

565 mWriteStr mes\_minMaxElement

566 mov ax, minMaxElement

567 mWriteAX

568 mWriteStr endl

569

570

571 mov ah, 07h *; Задержка экрана*

572 int 21h

573 jmp menu

574 task2: *; Проверка строк на монотонность*

575 mWriteStr mes\_n

576 mReadAX buffer 5

577 mov n, ax

578 xor ax, ax

579 mWriteStr mes\_k

580 mReadAX buffer 5

581 mov k, ax

582

583 mCheckMatrixRange matrix, rows, cols, n, k, flag

584

585 cmp flag, 0

586 jz task2False

587 mWriteStr mes\_flag\_true

588 mov ah, 07h *; Задержка экрана*

589 int 21h

590 jmp menu

591

592 task2False: mWriteStr mes\_flag\_false

593 mov ah, 07h *; Задержка экрана*

594 int 21h

595 jmp menu

596

597 task3: *; Получение среднего арифметического элементов выше диагоналей*

598 mWriteStr mes\_edited\_matrix

599

600 mRowNegativeSumsCopy matrix, rows, cols, copy\_matrix

601

602 mWriteMatrix copy\_matrix, rows, cols

603

604 mov ah, 07h *; Задержка экрана*

605 int 21h

606 jmp menu

607 *; Завершение программы*

608 exit:

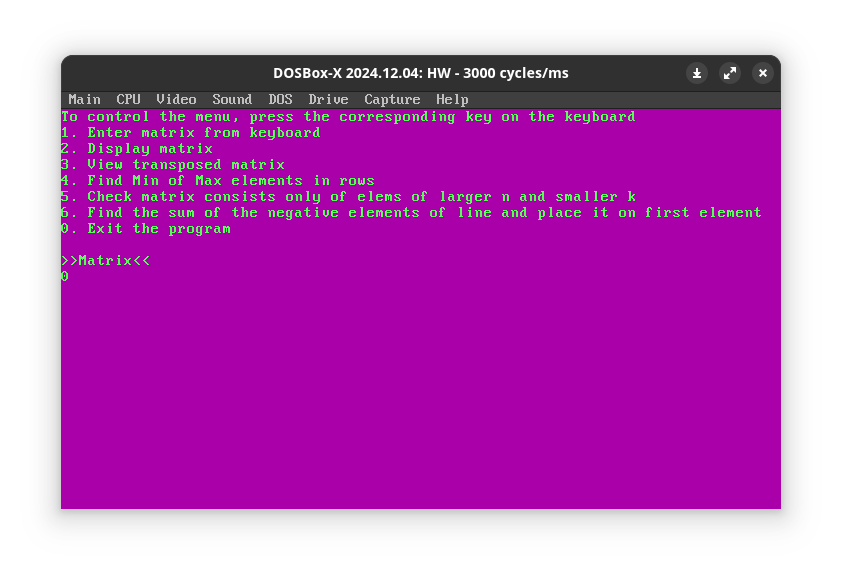
609 mov ax, 4c00h

610 int 21h

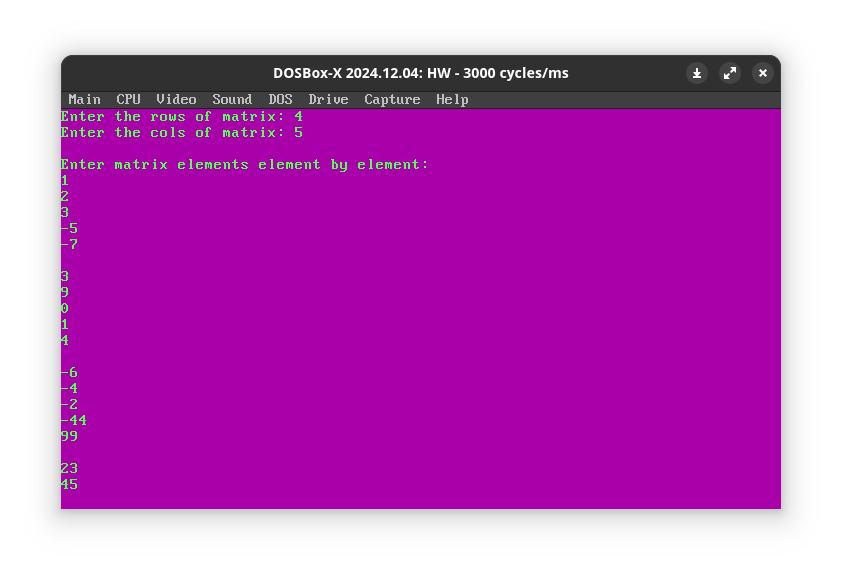
611 end Start

**Результат выполнения программы:**

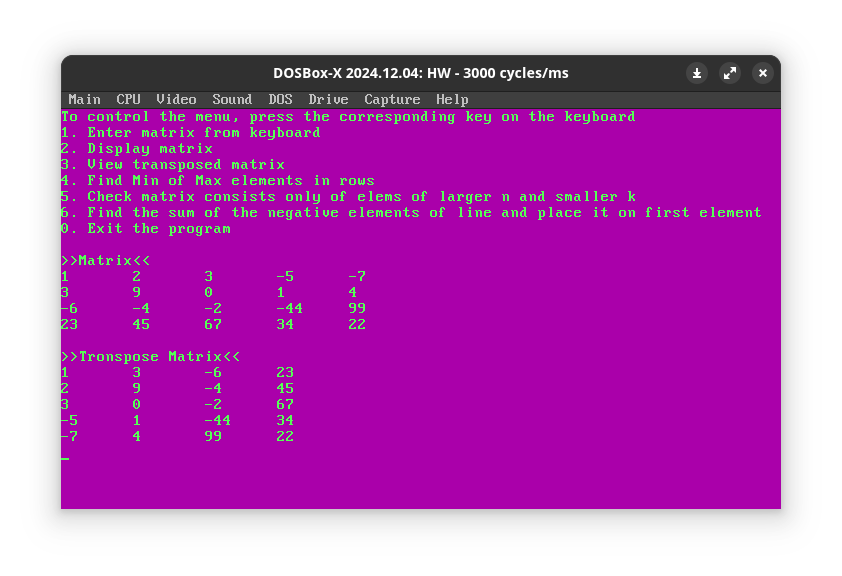
*Меню:*

**

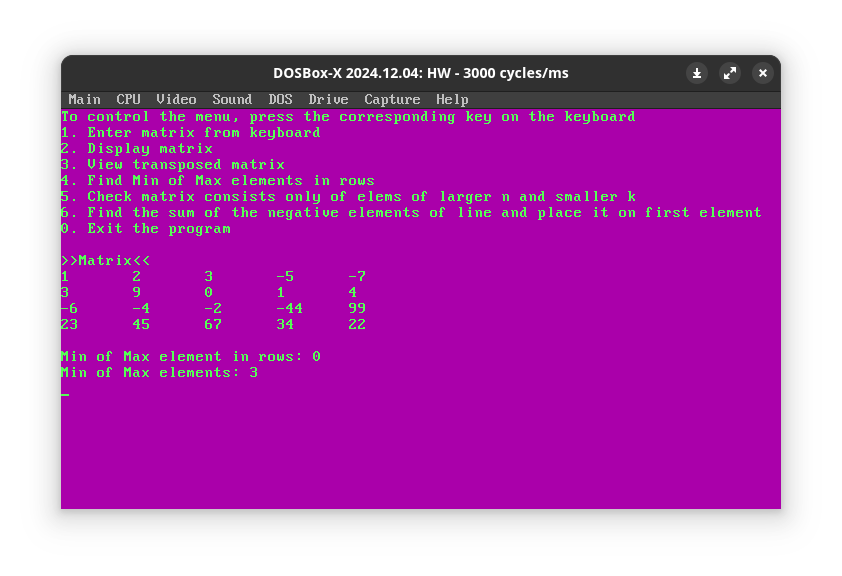
*Ввод значений матрицы:*

**

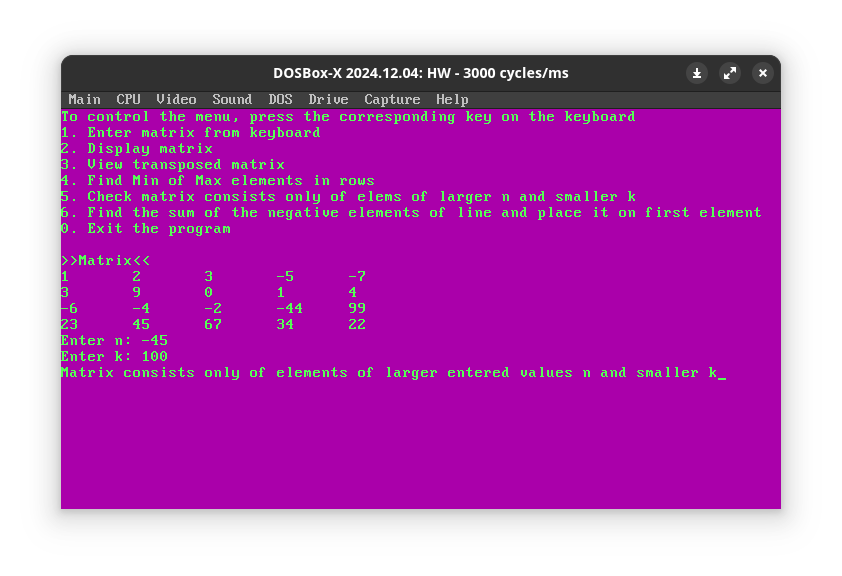
*Транспонирование:*

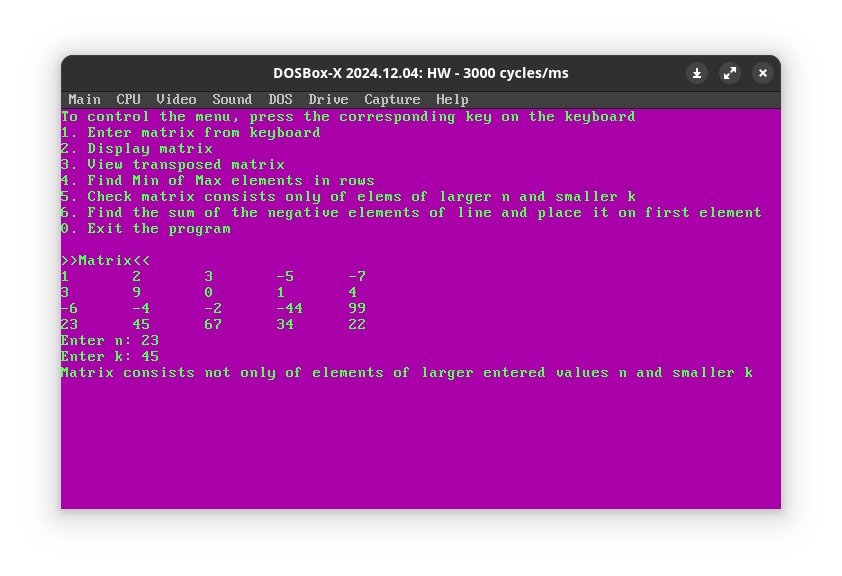
**

*a) Поиск минимального из максимальных элементов строк матрицы:*

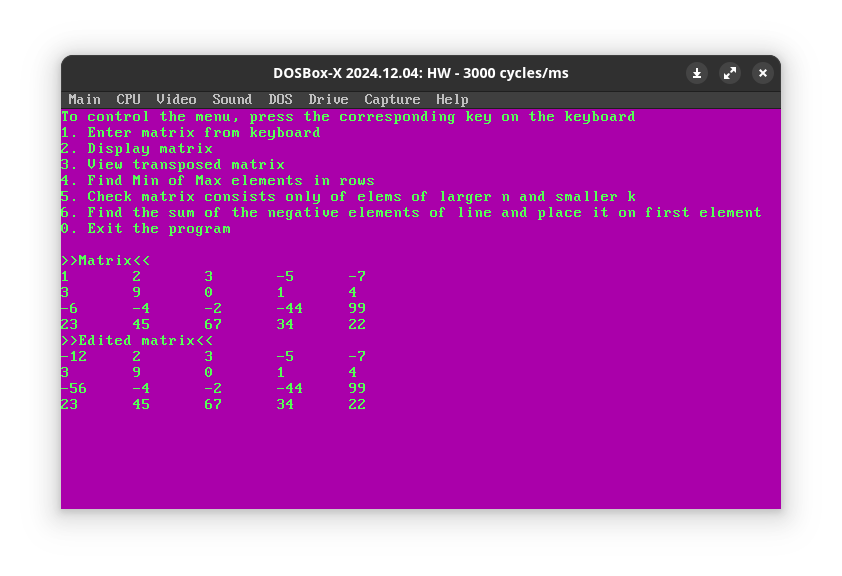
**

*б) Проверка что матрица состоит только из элементов между n и k*

**

**

*в) Нахождение суммы отрицательных элементов строки и установка её на место первого:*

**

**Вывод:** в ходе выполнения работы были сформированы практические навыки разработки программного кода на языке Ассемблера; изучены приёмы работы с массивами средствами языка Ассемблера.

**ЛИТЕРАТУРА**

**Основная литература**

1. Калашников О.А. Ассемблер- это просто. Учимся программировать [Текст] / О.А. Калашников.- СПб. БХВ-Петербург, 2012.- 336 с.
2. Кирнос В. Н. Введение в вычислительную технику: основы организации ЭВМ и программирование на Ассемблере[Электронный ресурс]: учеб.пособие / В. Н. Кирнос. — Томск: Эль Контент, 2011. -172с. URL://biblioclub.ru/index.php?page=book\_red&id=208652

**Дополнительная литература**

1. Юров В. И. ASSEMBLER[Текст]. Учебник для вузов /В. И. Юров. 2-е изд.– Спб.:Питер 2010. – 637с.: ил.
2. Юров В. И. ASSEMBLER[Текст]. Практикум. / В. И. Юров. 2-е изд.– Спб.:Питер 2007. – 399 с.
3. Зубков С.В. ASSEMBLER для DOS, WINDOWS, UNIX [Текст] / С.В. Зубков-3-е изд., М.:ДМК Пресс; 2004. – 608 с.: ил.

**Электронные ресурсы:**

1. Научная электронная библиотека http://eLIBRARY.ru.
2. Электронно-библиотечная система http://e.lanbook.com.
3. Электронно-библиотечная система «Университетская библиотека онлайн» http://biblioclub.ru.