МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №6
по дисциплине «Организация ЭВМ и систем»
Тема: Организация связи Ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы.

Студент гр. 0383 _____ Пенкин М.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

На языке С программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих заданный закон распределения. Необходимые датчики псевдослучайных чисел находятся в каталоге RAND_GEN (при его отсутствии получить у преподавателя).

Следует привести числа к целому виду с учетом диапазона изменения.

Далее должны вызываться 1 или 2 ассемблерных процедуры для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Ассемблерные процедуры должны вызываться как независимо скомпилированные модули. Передача параметров в процедуру должна выполняться через кадр стека.

Результирующий массив частотного распределения чисел по интервалам, сформированный на ассемблерном уровне, возвращается в программу, реализованную на ЯВУ, и затем сохраняется в файле и выводится на экран средствами ЯВУ.

Исходные данные:

1. Длина массива псевдослучайных целых чисел - NumRanDat (<= 16K).

2. Диапазон изменения массива псевдослучайных целых чисел [Xmin, Xmax] (м.б. биполярный, например, [-100, 100]).

3. Массив псевдослучайных целых чисел {Xi}.

4. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (<=24).

5. Массив левых границ интервалов разбиения LGrInt.

В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину, левые границы могут задаваться в произвольном порядке и иметь произвольные значения.

Если Xmin < LGrInt(1), то часть данных не будет участвовать в формировании распределения. Каждый интервал, кроме последнего, следует интерпретировать как [LGrInt(i), LGrInt(i+1)). Если у последнего интервала правая граница меньше Xmax, то часть данных не будет участвовать в формировании распределения.

Результаты:

Текстовая таблица, строка которой содержит:

- номер интервала,
- левую границу интервала,
- количество псевдослучайных чисел, попавших в интервал.

Количество строк должно быть равно числу интервалов разбиения. Таблица должна выводиться на экран и сохраняться в файле.

Задание:

Равномерное распределение случайных чисел, две ассемблерные процедуры, Nint >= D_x, Lg1 > X_{min}, ПГпосл $\leq$ X_{max}.

Выполнение работы.

В ходе работы была реализована программа из 3-х модулей, 1 на C++ (ЯВУ) и 2 других на ассемблере.

На ЯВУ написан main.cpp, который собирает от пользователя входную информацию и перенаправляет ее в ассемблерные модули. Также здесь осуществляется вывод данных в консоль и файл.

На ассемблере написано 2 модуля. Первый реализует распределение чисел по единичным отрезкам. Это сделано с помощью команды loop. Циклически записывается в новый массив количество повторений каждого числа.

Второй модуль формирует распределение тех же чисел, но уже по заданным интервалам. Это происходит благодаря нескольким циклам, в которых левые границы переводятся в неотрицательные числа и сопоставляются числам с таким же индексом из массива, полученного в первом модуле.

Связь между модулями осуществлена с помощью спецификатора extern, который позволяет выполнять раздельную компиляцию модулей.

Таблица 1. Проверка работы программы с отладочным выводом сгенерированных чисел.

Исходные данные	Результат	Примечание
NumDatRan=10 xmin=0 xmax=10 NInt=10 LGrInt={1 2 3 4 5 6 7 8 9 10}	№ Граница Количество чисел 1 1 2 2 2 1 3 3 0 4 4 1 5 5 1 6 6 1 7 7 0 8 8 0 9 9 2 10 10 0	Верно
NumDatRan= 24 xmin= -12 xmax= 12 NInt= 24 LGrInt={-11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12}	№ Граница Количество чисел 1 -11 1 2 -10 2 3 -9 0 4 -8 1 5 -7 3 6 -6 2 7 -5 0 8 -4 2 9 -3 0 10 -2 0 11 -1 0 12 0 0	Верно

	13 1 2 14 2 1 15 3 3 16 4 1 17 5 1 18 6 0 19 7 1 20 8 0 21 9 0 22 10 0 23 11 1 24 12 2	
NumDatRan= 5 xmin=-10 xmax=-5 NInt=5 LGrInt={-9 -8 -7 -6 -5}	№ Граница Количество чисел 1 -9 0 2 -8 0 3 -7 0 4 -6 1 5 -5 1	Верно

Тексты исходных файлов программ см. в приложении А.

Выводы.

В ходе выполнения данной лабораторной работы была изучена организация связи ассемблера с ЯВУ. Была реализована программа частотного распределения случайных чисел по заданным интервалам на языке C++ с использованием ассемблерных модулей.

Приложение А

Тексты исходных файлов программ

Название файла: main.cpp

```
DATA SEGMENT
        KEEP_CS DW 0 ; для хранения сегмента
        KEEP_IP DW 0 ; и смещения вектора прерывания

        HELLO   DB 'Hello World!',10,13,'$'
      MESEND   DB 'End!',10,13,'$'

DATA ENDS


AStack    SEGMENT  STACK
          DW 12 DUP(?)    ; ♦⁝♦♦♦♦♦ 12  ♦ ♦♦♦♦♦♦
AStack    ENDS



CODE      SEGMENT
          ASSUME CS:Code, DS:DATA, SS:AStack

SUBR_INT PROC FAR
        jmp start_proc
        KEEP_SS DW 0
      KEEP_SP DW 0
      KEEP_AX DW 0

      ;MESEND   DB 'End!',10,13,'$'
        BStack DW 12 DUP(?)
start_proc:

        MOV KEEP_SP, SP
        MOV KEEP_AX, AX
        MOV AX, SS
        MOV KEEP_SS, AX

        MOV AX, KEEP_AX

        MOV SP, OFFSET start_proc


        MOV AX, seg BStack
        MOV SS, AX

        PUSH AX ; сохранение изменяемых регистров
        PUSH DX;
```

```
            MOV   DX, OFFSET HELLO
            MOV   AH,9
          metka:
            int   21h  ; Вызов функции DOS по прерыванию
          loop metka ; Вывод сообщения заданное число раз


          ;mov al, 0
          ;mov ah, 86h
          ;mov cx, 0098h
          ;mov dx, 9680h
          int 15h; Фиксированная задержка



                  MOV     DX, OFFSET MESEND ; Вывод сообщения о
завершении обработчика
            MOV   AH,9
          int   21h


            POP DX;
            POP AX ; восстановление регистров

            MOV  KEEP_AX, AX
            MOV SP, KEEP_SP
            MOV AX, KEEP_SS
            MOV SS, AX
            MOV AX, KEEP_AX

            MOV AL, 20H
            OUT 20H,AL
            IRET
     SUBR_INT ENDP


     Main      PROC  FAR

              push  DS        ;\  Сохранение адреса начала PSP в
стеке
            sub   AX,AX    ; > для последующего восстановления по
                push   AX          ;/   команде ret, завершающей
процедуру.
            mov   AX,DATA             ; Загрузка сегментного
            mov   DS,AX               ; регистра данных.

            MOV AH, 35H ; функция получения вектора
            MOV AL, 60H ; номер вектора
                 INT  21H  ; возвращает текущее значение вектора
прерывания
            MOV KEEP_IP, BX ; запоминание смещения
            MOV KEEP_CS, ES ; и сегмента вектора прерывания

            PUSH DS
            MOV DX, OFFSET SUBR_INT ; смещение для процедуры в DX
```

```
            MOV AX, SEG SUBR_INT ; сегмент процедуры
            MOV DS, AX ; помещаем в DS
            MOV AH, 25H ; функция установки вектора
            MOV AL, 60H ; номер вектора
            INT 21H ; меняем прерывание
            POP DS

          mov cx, 10
          ;mov dx, 3
            int 60H; вызов измененного прерывания

            CLI
            PUSH DS
            MOV DX, KEEP_IP
            MOV AX, KEEP_CS
            MOV DS, AX
            MOV AH, 25H
            MOV AL, 60H
            INT 21H ; восстанавливаем старый вектор прерывания
            POP DS
            STI

            RET
      Main      ENDP
      CODE      ENDS
                END Main
```

Название файла: first.asm

```
      .586p
      .MODEL FLAT, C
      .CODE
      PUBLIC C first
      first  PROC  C  array:  dword,  arraysize:  dword,  res:  dword,
xmin: dword

      push esi
      push edi

      mov edi, array
      mov ecx, arraysize
      mov esi, res

      for_numbers:
      mov eax, [edi]
      sub eax, xmin
      mov ebx, [esi + 4*eax]
      inc ebx
      mov [esi + 4*eax], ebx
      add edi, 4
      loop for_numbers

      pop edi
      pop esi
```

```
ret
first ENDP
END
```

Название файла: second.asm

```
.586p
.MODEL FLAT, C
.CODE
PUBLIC C second
second PROC C array: dword, array_size: dword, xmin: dword,
borders: dword, intN: dword, result: dword

push esi
push edi
push ebp

mov edi, array
mov esi, borders
mov ecx, intN


for_borders:
mov eax, [esi]
sub eax, xmin
mov [esi], eax
add esi, 4
loop for_borders


mov esi, borders
mov ecx, intN
mov ebx, 0
mov eax, [esi]

for_loop:
push ecx
mov ecx, eax
push esi
mov esi, result

    for_array:
      cmp ecx, 0
      je end_for
         mov eax, [edi]
         add [esi + 4*ebx], eax
         add edi, 4
         loop for_array

end_for:
    pop esi
    inc ebx
mov eax, [esi]
add esi, 4
```

```
sub eax, [esi]
neg eax
pop ecx
loop for_loop

mov esi, result
mov ecx, intN
mov eax, 0

fin_for:
add eax, [esi]
add esi, 4
loop fin_for

mov esi, result
sub eax, array_size
neg eax

add [esi + 4*ebx], eax

pop ebp
pop edi
pop esi

ret
second ENDP
END
```