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

ОТЧЕТ

по практической работе №4
по дисциплине «Операционные системы»
Тема: Обработка стандартных прерываний

Студентка гр. 7382 _____ Чемова К.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2019

Цель работы.

В архитектуре компьютера существуют стандартные прерывания, за которыми закреплены определённые вектора прерываний. Вектор прерываний хранит адрес подпрограммы обработчика прерываний. При возникновении прерывания, аппаратура компьютера передаёт управление по соответствующему адресу вектора прерывания. Обработчик прерываний получает управление и выполняет соответствующие действия.

В лабораторной работе № 4 предлагается построить обработчик прерываний сигналов таймера. Эти сигналы генерируются аппаратурой через определённые интервалы времени и, при возникновении такого сигнала, возникает прерывание с определённым значением вектора. Таким образом, управление будет передано функции, чья точка входа записана в соответствующий вектор прерывания.

Необходимые сведения для составления программы.

Резидентные обработчики прерываний – это программные модули, которые вызываются при возникновении прерываний определенного типа (сигнал таймера, нажатие клавиши и т.д.), которым соответствуют определенные вектора прерывания. Когда вызывается прерывание, процессор переключается на выполнение кода обработчика, а затем возвращается на выполнение прерванной программы. Адрес возврата в прерванную программу (CS:IP) запоминается в стеке вместе с регистром флагов. Затем в CS:IP загружается адрес точки входа программы обработки прерывания и начинает выполняться его код. Обработчик прерывания должен заканчиваться инструкцией IRET (возврат из прерывания).

Вектор прерывания имеет длину 4 байта. В первом хранится значение IP, во втором – CS. Младшие 1024 байта памяти содержат 256 векторов. Вектор для прерывания 0 начинается с ячейки 0000:0000, для прерывания 1 – с ячейки 0000:0004 и т.д.

Обработчик прерывания - это отдельная процедура, имеющая следующую структуру:

```
ROUT PROC FAR
PUSH AX ; сохранение изменяемых регистров
.....................................
<действия по обработке прерывания>

POP AX ; восстановление регистров
MOV AL, 20H
OUT 20H,AL
IRET
ROUT ENDP
```

Две последние строки необходимы для разрешения обработки прерываний с более низкими уровнями, чем только что обработанное. Для установки написанного прерывания в поле векторов прерываний используется функция 25H прерывания 21H, которая устанавливает вектор прерывания на указанный адрес.

```
PUSH DS
MOV DX, OFFSET ROUT   ; смещение для процедуры в DX
MOV AX, SEG ROUT            ; сегмент процедуры
MOV DS, AX                  ; помещаем в DS
MOV AH, 25H                 ; функция установки вектора
MOV AL, 1CH                 ; номер вектора
INT 21H                     ; меняем прерывание
POP DS
```

Программа, выгружающая обработчик прерываний должна восстанавливать оригинальные векторы прерываний. Функция 35 прерывания 21H позволяет восстановить значение вектора прерывания, помещая значение сегмента в ES, а смещение в BX. Программа должна содержать следующие инструкции:

```
; -- хранится в обработчике прерываний
KEEP_CS DW 0          ; для хранения сегмента
KEEP_IP DW 0          ; и смещения прерывания
; -- в программе при загрузке обработчика прерывания
MOV AH, 35H           ; функция получения вектора
MOV AL, 1CH           ; номер вектора ШТЕ 21Р
MOV KEEP_IP, BX ; запоминание смещения
MOV KEEP_CS, ES ; и сегмента
```

```
; -- в программе при выгрузке обработчика прерываний CLI
PUSH DS
MOV DX, KEEP_IP
MOV AX, KEEP_CS
MOV DS, AX
MOV AH, 25H
MOV AL, 1CH
INT 21H    ; восстанавливаем вектор
POP DS
STI
```

Для того, чтобы оставить процедуру прерывания резидентной в памяти, следует воспользоваться функцией DOS 31h прерывания 21h. Эта функция оставляет память, размер которой указывается в качестве параметра, занятой, а остальную память освобождает и осуществляет выход в DOS.

Функция 31h int 21h использует следующие параметры:

AH – номер функции 31h;

AL – код завершения программы;

DX – размер памяти в параграфах, требуемый резидентной программе.

Пример обращения к функции:

```
MOV DX, OFFSET LAST_BYTE ; размер в байтах от начала сегмента
MOV CL,4   ; перевод в параграфы
SHR DX,CL
INC DX     ; размер в параграфах
MOV AH,31h
INT 21h
```

Постановка задачи.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет следующие функции:

Проверяет, установлено ли пользовательское прерывание с вектором 1Ch.

Устанавливает резидентную функцию для обработки прерывания и настраивает вектор прерываний, если прерывание не установлено, и осуществляется выход по функции 4Ch прерывания int 21h.

Если прерывание установлено, то выводится соответствующее сообщение и осуществляется выход по функции 4Ch прерывания int 21h.

Выгрузка прерывания о соответствующему значению параметра в командной строке /un. Выгрузка прерывания состоит в восстановлении стандартного вектора прерываний и освобождении памяти, занимаемой резидентом. Затем осуществляется выход по функции 4Ch прерывания int 21h.

Шаг 2. Далее необходимо запустить отлаженную программу и убедиться, что резидентный обработчик прерывания 1Ch установлен. Работа прерывания должна отображаться на экране, а также необходимо проверить размещение прерывания в памяти. Для этого нужно запустить программу ЛР 3, которая отображает карту памяти в виде с писка блоков MCB.

Шаг 3. Затем необходимо запустить отлаженную программу еще раз и убедиться, что программа определяет установленный обработчик прерываний.

Шаг 4. Далее нужно запустить отлаженную программу с ключом выгрузки и убедиться, что резидентный обработчик прерывания выгружен, то есть сообщения на экран не выводятся, а память, занятая резидентом освобождена. Для этого также следует запустить программу ЛР 3.

Процедуры используемые в программе.

Описание процедур, используемых в работе:

- PRINT – выводит сообщение на экран;
- SET_CURS – установка курсора в заданную позицию, где DH, DL – строка, колонка соответственно;
- GET_CURS – определяет текущую позицию курсора. После выполнения, возвращает DH – текущая строка и DL – текущая колонка курсора;
- ROUT – обработчик прерываний сигналов таймера. Выводит на экран счётчик – суммарное число прерываний 1Ch. При значении check = 1

восстанавливает стандартный вектор прерывания и выгружает из памяти пользовательское прерывание;

- CHECK_INT – проверка, установлено ли пользовательское прерывание с вектором 1Ch. Есть ли в хвосте «/un», если да, то присваивает значению check единицу;
- MY_INT – установка написанного прерывания в поле векторов прерываний.

Структуры данных.

Таблица 2 – Структуры данных используемые в программе

Название поля данных	Тип	Назначение
interrupt_already_loaded	db	Вывод информации о том, что пользовательское прерывание уже было установлено
interrupt_was_unloaded	db	Вывод информации о том, что пользовательское прерывание выгружено
interrupt_was_loaded	db	Вывод информации о том, что пользовательское прерывание установлено
interrupt_already_loaded	db	Вывод информации о том, что пользовательское прерывание уже было установлено
signature	db	Сигнатура пользовательского прерывания
keep_psp	dw	Переменная для сохранения сегментного адреса PSP
keep_ss	dw	Переменная для сохранения сегментного адреса стека
keep_ax	dw	Переменная для сохранения в AX
keep_sp	dw	Переменная для сохранения указателя стека
keep_cs	dw	Переменная для сохранения в CS
keep_ip	dw	Переменная для сохранения в IP
check	dw	Флаг для определения необходимости выгружать прерывание
timer	db	Число вызовов прерывания

Ход работы.

Был написан программный модуль .EXE. исходный код представлен в приложении А.

На рис.1 представлен результат работы программы предыдущей лабораторной работы.

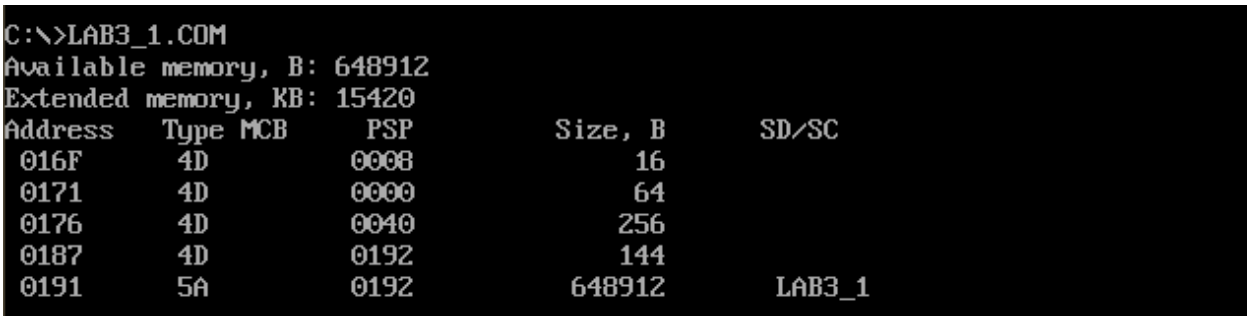

Рисунок 1 – Память до загрузки резидента

Была запущена программа, результат работы которой представлен на рис.2.

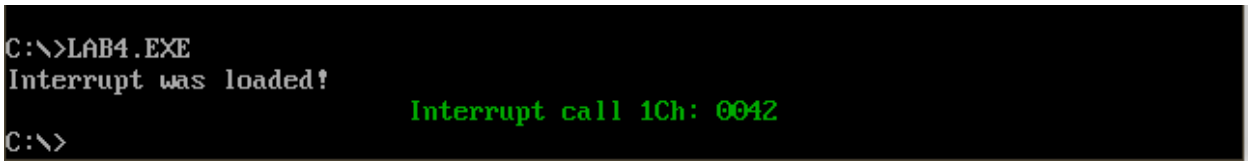

Рисунок 2 – Загрузка резидента

Повторный запуск программы показан а рис.3.

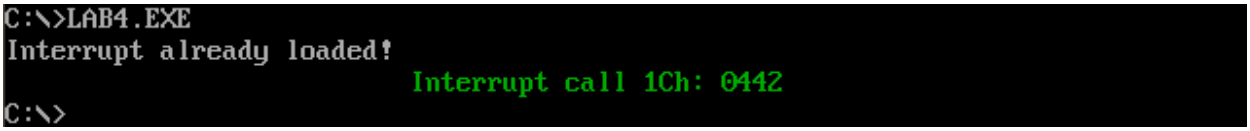

Рисунок 3 – Повторная загрузка резидента

Состояние памяти при загрузке в неё резидента показано на рис.4.

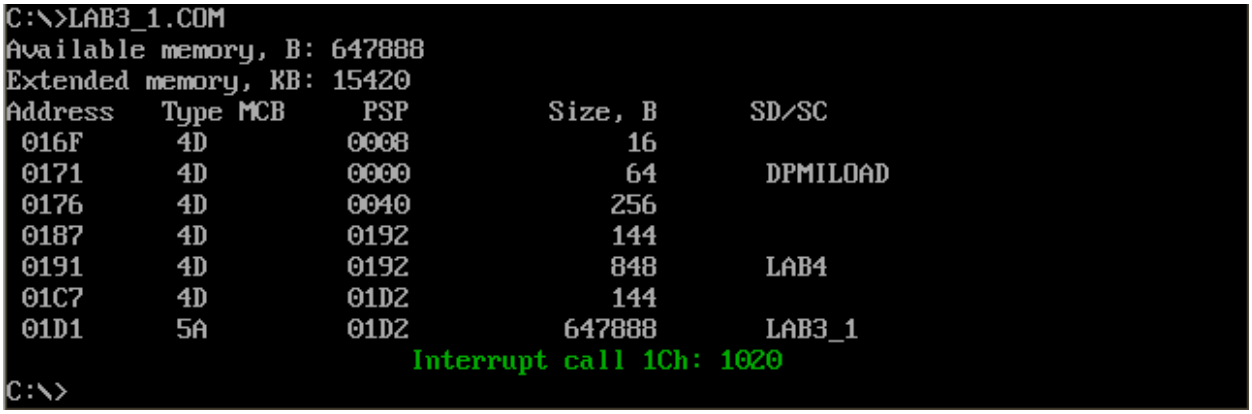

Рисунок 4 – Состояние памяти при загрузке резидента

Запуск отложенной программы с ключом </un>, происходит выгрузка резидента. Состояние памяти после выгрузки резидента показано на рис.5.

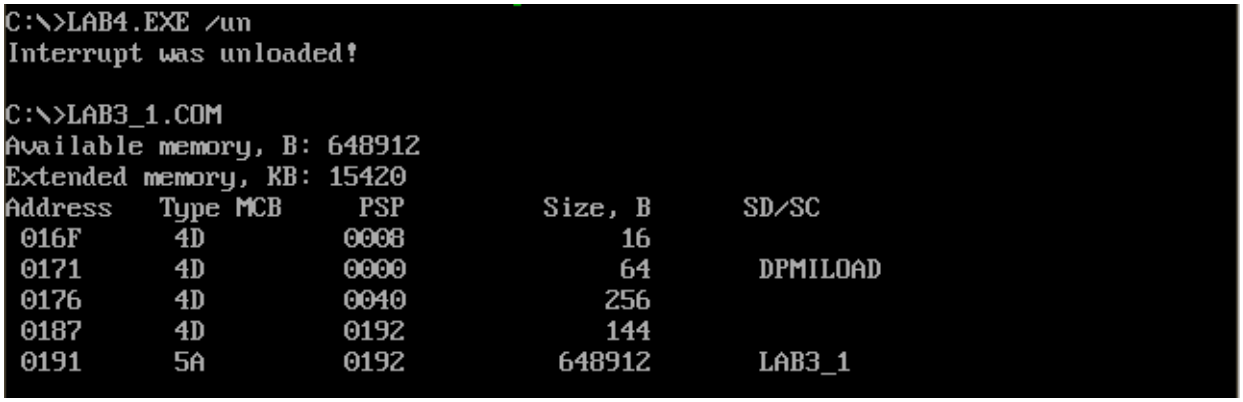

Рисунок 5 – Выгрузка резидента

Вывод.

В ходе лабораторной работы был построен обработчик прерывания от сигналов таймера. Изучены дополнительные функции работы с памятью: установка программы-резидента и его выгрузка из памяти.

Ответы на контрольные вопросы.

1) **Как реализован механизм прерывания от часов?**

Сначала сохраняется содержимое регистров, потом определяется источник прерывания, по номеру которого определяется смещение в таблице векторов прерывания, сохраняется в CS:IP, передаётся управление по адресу CS:IP и происходит выполнение обработчика, и в конце происходит возврат управления прерванной программе. Аппаратное прерывание от таймера происходит каждые 55 мс.

2) **Какого типа прерывания использовались в работе?**

Аппаратные прерывания (1Ch), прерывания функций DOS (21h), прерывания функций BIOS (10h).

ПРИЛОЖЕНИЕ А
ИСХОДНЫЕ КОД ПРОГРАММЫ

```
MY_STACK SEGMENT STACK
      DW 64 DUP (?)
MY_STACK ENDS

ASTACK SEGMENT STACK
      DW 64 DUP (?)
ASTACK ENDS
;ДАННЫЕ
;-------------------------------------------------------------------------------
---------------
DATA SEGMENT
interrupt_already_loaded  db 'Interrupt already loaded!', 0DH, 0AH, '$'
interrupt_was_unloaded    db 'Interrupt was unloaded!', 0DH, 0AH, '$'
interrupt_was_loaded      db 'Interrupt was loaded!', 0DH, 0AH, '$'
DATA ENDS
;-------------------------------------------------------------------------------
---------------
CODE SEGMENT
      ASSUME CS:CODE, DS:DATA, ES:DATA, SS:ASTACK
;ПРОЦЕДУРЫ
;-------------------------------------------------------------------------------
---------------
PRINT PROC NEAR
;вывод строки
      mov   AH,  0009h
      int   21h
      ret
PRINT ENDP
;-------------------------------------------------------------------------------
---------------
SET_CURS PROC
;установка позиции курсора; установка на строку 25 делает курсор невидимым
      push  AX
      push  BX
      push  CX
      mov   AH,  0002h
      mov   BH,  0000h
      int   0010h       ;выполнение функции 0002h
      pop   CX          ;вход: BH = видео страница
      pop   BX          ;DH, DL = строка, колонка (считая от 0)
      pop   AX
      ret
SET_CURS ENDP
;-------------------------------------------------------------------------------
---------------
GET_CURS PROC
;функция, определяющая позицию и размер курсора
      push  AX
      push  BX
      push  CX
```

```
      mov   AH,  0003h  ;0003h читать позицию и размер курсора
      mov   BH,  0000h  ;Вход: BH = видео страница
      int   10h         ;выполнение функции 0003h
      pop   CX          ;Выход: DH, DL = текущая строка, колонка курсора
      pop   BX          ;CH, CL = текущая начальная, конечная строки курсора
      pop   AX
      ret
GET_CURS ENDP
;--------------------------------------------------------------------------
---------------
ROUT PROC FAR
;обработчик прерывания
      jmp   ROUT_BEGINNING
;ДАННЫЕ
;--------------------------------------------------------------------------
---------------
      signature  db  '0000'                          ;некоторый код, который
идентифицирует резидента
      keep_cs    dw  0                              ;для хранения сегмента
      keep_ip    dw  0                              ;и смещения прерывания
      keep_psp   dw  0                              ;PSP
      check      dw  0                               ;надо выгружать прерывание
или нет
      keep_ss    dw  0                              ;сегмента стека
      keep_ax    dw  0
      keep_sp    dw  0
      timer      db  'Interrupt call 1Ch: 0000 $'  ;счётчик
;--------------------------------------------------------------------------
---------------
ROUT_BEGINNING:
      mov   keep_ax,  AX
      mov   keep_ss,  SS
      mov   keep_sp,  SP
      mov   AX,  MY_STACK   ;устанавливаем собственный стек
      mov   SS,  AX
      mov   SP,  64h
      mov   AX,  keep_ax
      push  DX              ;сохраняем изменяемые регистры
      push  DS
      push  ES
      cmp   check,  1
      je    ROUT_REC
      call  GET_CURS        ;получаем текущее положение курсора
      push  DX              ;сохраняем положения курсора в стеке
      mov   DH,  17h        ;DH, DL - строка, колонка (считая от 0)
      mov   DL,  1Ah        ;определяем местоположение надписи
      call  SET_CURS        ;устанавливаем курсор
;**************************************************************************
***************
ROUT_COUNT:
;счётчик количества прерываний
      push  SI                ;сохраняем все изменяемые регистры
      push  CX
      push  DS
      mov   AX,  seg timer
      mov   DS,  AX
```

```
      mov   SI,  offset timer
      add   SI,  0017h         ;смещение на последнюю цифру
;*************************************************************************
***************
count:
      mov   AH,  [SI]   ;получаем цифру
      inc   AH
      mov   [SI],  AH   ;возвращаем
      cmp   AH,  3Ah    ;если не равно 9
      jne   END_COUNT   ;завершение и вывод результата
      mov   AH,  30h    ;обнуляем
      mov   [SI],  AH
        dec SI
        loop     count
;*************************************************************************
***************
END_COUNT:
;печать счётчика-строки на экран
      pop   DS
      pop   CX
      pop   SI
      push  ES
      push  BP
      mov   AX,  seg timer
      mov   ES,  AX
      mov   AX,  offset timer
      mov   BP,  AX            ;вход: ES:BP выводимая строка
      mov   AH,  00013h        ;выдаёт строку в позиции курсора
      mov   AL,  0000h         ;режим вывода
      mov   CX,  0019h         ;длина строки = 25 символов
      mov   BH,  0000h         ;видео страница, её номер
      mov   BL,  0002h         ;установка атрибута
      int   10h
      pop   BP
      pop   ES
      pop   DX                 ;возвращение курсора
      call  SET_CURS
      jmp   ROUT_END
;*************************************************************************
***************
ROUT_REC:
;восстановление вектора прерывания
      CLI                     ;запрещение прерывания, путём сбрасывания флага
IF
      mov   DX,  keep_ip
      mov   AX,  keep_cs
      mov   DS,   AX           ;вектор прерывания: адрес программы обработки
прерывания
      mov   AH,  25h       ;устанавливает вектор прерывания
      mov   AL,  1Ch       ;номер вектора прерывания
      int   21h
      mov   ES,  keep_psp
      mov   ES,   ES:[2Ch]  ;сегментный адрес (параграф) освобождаемого блока
памяти
      mov   AH,  49h       ;освобождает распределённый блок памяти
      int   21h
```

```
      mov   ES,  keep_psp
      mov   AH,  49h
      int   21h
      STI                     ;разрешение прерывания
;*****************************************************************************
***************
ROUT_END:
      pop   ES            ;восстановление регистров
      pop   DS
      pop   DX
      mov   SS,  keep_ss
      mov   SP,  keep_sp
      mov   AX,  keep_ax
      iret

LAST_BYTE:
        ROUT ENDP
;-----------------------------------------------------------------------------
---------------
CHECK_INT PROC
;проверка, установлено ли пользовательское прерывание с вектором 1Ch
      mov   AH,  35h               ;даёт вектор прерывания
      mov   AL,  1Ch               ;номер прерывания
      int   21h                           ;выход: ES:BX = адрес обработчика
прерывания
      mov   SI,  offset signature
      sub   SI,   offset ROUT         ;смещение signature относительно начала
функции прерывания
      mov   AX,  '00'              ;сравнеение известного значения сигнатуры
      cmp   AX,  ES:[BX+SI]        ;с реальным кодом, находящимся в резиденте
      jne   NOT_LOADED                ;если значения разные, то резидент не
установлен
      cmp   AX,  ES:[BX+SI+2]
      jne   NOT_LOADED
      jmp   LOADED
;*****************************************************************************
***************
NOT_LOADED:
      call  MY_INT                 ;установка пользовательского прерывания
      mov   DX,  offset LAST_BYTE  ;размер в байтах от начала
      mov   CL,  4                 ;перевод в параграфы
      shr   DX,  CL                ;сдвиг на 4 разряда вправо
      inc   DX
      add   DX,  CODE              ;прибавляем адрес сегмента CODE
      sub   DX,  keep_psp
      xor   AL,  AL
      mov   AH,  31h               ;оставляем нужное количество памяти
      int   21h
;*****************************************************************************
***************
LOADED:
;смотрим, есть ли в хвосте /un , тогда нужно выгружать
      push  ES
      push  AX
      mov   AX, keep_psp
      mov   ES, AX
```

```
      cmp   byte ptr ES:[0082h],  '/'
      jne   NOT_UNLOAD
      cmp   byte ptr ES:[0083h],  'u'
      jne   NOT_UNLOAD
      cmp   byte ptr ES:[0084h],  'n'
      je    UNLOAD
;************************************************************************
***************
NOT_UNLOAD:
      pop   AX
      pop   ES
      mov   DX,  offset interrupt_already_loaded
      call  PRINT
      ret
;************************************************************************
***************
UNLOAD:
      pop   AX
      pop   ES
      mov   byte ptr ES:[BX+SI+10], 1          ;check = 1
      mov   DX, offset interrupt_was_unloaded
      call  PRINT
      ret
CHECK_INT ENDP
;------------------------------------------------------------------------
---------------
MY_INT PROC
;установка написанного прерывания в поле векторов прерываний
      push  DX
      push  DS
      mov   AH,  35h                        ;функция получения вектора
      mov   AL,  1Ch                        ;номер вектора
      int   21h
      mov   keep_ip,  BX                    ;запоминание смещения
      mov   keep_cs,  ES
      mov   DX,  offset ROUT                ;смещение для процедуры в DX
      mov   AX,  seg ROUT
      mov   DS,  AX
      mov   AH,  25h                        ;функция установки вектора
      mov   AL,  1Ch                        ;номер вестора
      int   21h
      pop   DS
      mov   DX,  offset interrupt_was_loaded
      call  PRINT
      pop   DX
      ret
MY_INT ENDP
;------------------------------------------------------------------------
---------------
MAIN  PROC  near
      mov   AX,  DATA
      mov   DS,  AX
      mov   keep_psp,  ES
      call  CHECK_INT
      xor   AL,  AL
      mov   AH,  4Ch
```

```
      int   21H
MAIN  ENDP
CODE ENDS
      END  MAIN
```