//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// lab 2

// was written by : [Reham Abass 201277811] and [Byata 205797475]

#include<dos.h>

#include<stdio.h>

#include "stdlib.h"

//////////////////////////////////////////////////

void interrupt (\*old1)(void);

void interrupt (\*old8)(void);

void unfollow();

void start();

//////////////////////////////////////////////

volatile int count,timer,flag; //global variables which gonna be used in the interrupt routines

typedef struct node{ // struct for the linked list

struct node \*prev ; //pointer to previus node

struct node \*next; //pointer to the next node

void (\*function)(int); //pointer to the function

int \* varptr; //pointer to the variable

int varvalue; //value of the vaiable

char\*varname; //name o the variable

char\*funcname; //name of the function

int startTime; //start time of this node<this call of follow>

}node;

volatile node \*head; //pointer to the first node of all the linked list

volatile node\*tail; //pointer to the last node oa all the linked list

volatile node \*ptr; //pointer to run on all the linked list each time in interrupt 1

volatile node\* tmp; //pointer temp

volatile void \*f; //global variable to save the instruction in the assembly line

//////////////////////////////////////////////

void interrupt my1(){

old1();

if(flag ==1){

ptr =head;

while(ptr!=NULL){

asm{ //copy the instruction from cs:ip to global variable: f

push ax

Mov ax, [bp+18] //[bp+18] has the cs

Mov word ptr f, ax

Mov ax, [bp+20] //[bp+20] has the ip

Mov word ptr f+2, ax

pop ax

}

if(f == ptr->function)

printf("\nprocedure %s has been called",ptr->funcname);

else if(\*(ptr->varptr) != ptr->varvalue){

if(ptr->varvalue <1000) //values of fibonacci will not increse until 1000 in 5 sec's, they all smaller than 1000

printf("\nVariable %s has changed from %d to %d",ptr->varname,ptr->varvalue,\*(ptr->varptr));

ptr->varvalue= \*(ptr->varptr); //update variable

}

if (((head->startTime)+5) == timer) { // check end condition

tmp=head;

(head->next)->prev =NULL;

head=head->next; //first called to follow uses the first node (head)which gonna end first one in the list

free(tmp);

}

ptr=ptr->next; //increase the pointer so could point on the next node

}

} //flag

}

//////////////////////////////////////////////

void interrupt my8(){

count++;

timer =count%19; //timer is by seconds

old8();

}

//////////////////////////////////////////////

/////////////////////////////////////////

void unFollow(){

setvect(1,old1); //setvect to the old routine

setvect(8,old8); //setvect to the old routine

//turn off TF flag

asm{

pushf

pop ax

and ax,1111111011111111B

push ax

popf

}

//free allocations :

while(head!= NULL){

tmp =head;

head = head->next;

free(tmp);

}

}

//////////////////////////////////////////

void follow(int \*iptr,char str[],void (\*funptr)(), char fstr[]){

node \* node1= (node\*)malloc(sizeof(node));

if(node1== NULL){printf("\n can't allocate");exit(1);}

if(head==NULL){ //if it was first node

head = node1;

tail=head;

head->prev= NULL;

head->next=NULL;

}

else{ //if it is not the first node

tail->next =node1;

node1->prev =tail;

node1->next =NULL;

node1 =tail;

}

//iniate the pointers inside the node struct

node1->varptr=iptr; //save pointer of the variable

node1->function = funptr; //save pointer to the function

node1->varvalue = \*iptr; //save value of the variable

node1->varname = str; //save name of the variable

node1->funcname = fstr;//save name of the function

node1->startTime= timer;//save the start time of this follow , so it could finish following after 5 seconds

flag=1; //this flag says to the interrupt 1 that the follow is called now

}

///////////////////////////////////////////

void start(){

flag =0; //this flag says to interrupt 1 that the follow doesn't called yet

//turn on TF flag

asm{

pushf

pop ax

or ax,100000000B

push ax

popf

}

head=tail=NULL; //iniation of the pointers

old8=getvect(8); //save old 8

old1 = getvect(1); //save old 1

setvect(8,my8); //set the new routine

setvect(1,my1); //set the new routine

}

///////////////////////////////////////////

void func(){} //doesn't do anything

////////////////////////////////////////////

void main(void){

int i , f1 = 1 , f2 = 1 , fibo = 1;

start(); //to iniate the variables and to turn on the trap flag +setvects

follow(&fibo, "fibo", sleep, "sleep");

sleep(3);

follow(&f1, "f1", func, "func");

for(i=0; i < 10; i++){

sleep(1);

func();

f1 = f2;

f2 = fibo;

fibo = f1 + f2;

}

unFollow();

f1=0;

fibo = 0;

}

//////////////////////////////////////////////