Лабораторная работа 1. Вариант 1.а вычисление чисел Фибоначчи

В процессе работы была написана следующая программа:

main.c

```c
#include<stdio.h>
#include"fibonacci.h"


int main(){
        int n = 11;
        printf("Fibonacci element number %d= %lld\n", n, fibonacci(n));
        return 0;
}
```

fibonacci.c

```c
#include <stdio.h>
#include "fibonacci.h"

long long int fibonacci(int n){
        if(n == 0){return n;}
        else if(n == 1){return n;}
        return fibonacci(n - 1) + fibonacci(n - 2);
}

```

fibonacci.h

```c
#ifndef FIBONACCI
#define FIBONACCI


long long int fibonacci(int n);


#endif

```

Трансляция

```
user@user-VirtualBox:~/Рабочий стол/OS$ gcc -S fibonacci.c -o fibonacci.s
user@user-VirtualBox:~/Рабочий стол/OS$ gcc -S main.c -o main.s
user@user-VirtualBox:~/Рабочий стол/OS$ gcc -S -O1 main.c -o main01.s
user@user-VirtualBox:~/Рабочий стол/OS$ gcc -S -O2 main.c -o main02.s
user@user-VirtualBox:~/Рабочий стол/OS$ gcc -S -O3 main.c -o main02.s
```

Ассемблер код с комментариями

```
        .file   "main.c"                                ; Имя исходного файла main.c
        .section        .rodata                         ; Начало секции только для чтения данных
        .align 4                                        ; Выравнивание данных по границе 4 байта
.LC0:
        .string "Fibonacci element number %d= %lld\n"   ; Определение строки форматирования для вывода
        .text                                           ; Начало секции кода
        .globl  main                                    ; Объявление глобальной функции main
        .type   main, @function                         ; Определение типа функции main как функции
main:
.LFB0:                                                  ; Начало базового блока функции main
        .cfi_startproc                                  ; Начало процедуры сохранения информации о кадре
        leal    4(%esp), %ecx                           ; Получение адреса параметров командной строки в регистр %ecx
        .cfi_def_cfa 1, 0                               ; Определение текущего адреса стека
        andl    $-16, %esp                              ; Выравнивание стека по границе 16 байт
        pushl   -4(%ecx)                                ; Занесение значения в стек
        pushl   %ebp                                    ; Сохранение значения регистра %ebp
        .cfi_escape 0x10,0x5,0x2,0x75,0                 ; Управление информацией о кадре
        movl    %esp, %ebp                              ; Сохранение указателя стека в регистре %ebp
        pushl   %ecx                                    ; Занесение значения %ecx в стек
        .cfi_escape 0xf,0x3,0x75,0x7c,0x6               ; Управление информацией о кадре
        subl    $20, %esp                               ; Выделение места для локальных переменных
        movl    $11, -12(%ebp)                          ; Установка значения переменной на стеке
        subl    $12, %esp                               ; Выделение места для аргументов функции
        pushl   -12(%ebp)                               ; Занесение значения переменной в стек
        call    fibonacci                               ; Вызов функции fibonacci
        addl    $16, %esp                               ; Очистка стека
        pushl   %edx                                    ; Занесение значения edx в стек
        pushl   %eax                                    ; Занесение значения eax в стек
        pushl   -12(%ebp)                               ; Занесение значения переменной в стек
        pushl   $.LC0                                   ; Занесение адреса строки форматирования в стек
        call    printf                                  ; Вызов функции printf
        addl    $16, %esp                               ; Очистка стека
        movl    $0, %eax                                ; Установка возвращаемого значения в 0
        movl    -4(%ebp), %ecx                          ; Восстановление значения регистра ecx
        .cfi_def_cfa 1, 0                               ; Определение текущего адреса стека
        leave                                           ; Завершение процедуры
        .cfi_restore 5                                  ; Восстановление значения регистра ebp
        leal    -4(%ecx), %esp                          ; Корректировка адреса стека
        .cfi_def_cfa 4, 4                               ; Определение текущего адреса стека
        ret                                             ; Возврат из функции
        .cfi_endproc                                    ; Окончание процедуры
.LFE0:                                                  ; Конец базового блока функции main
        .size   main, .-main                            ; Размер функции main
        .ident  "GCC: (Ubuntu 4.9.2-10ubuntu13) 4.9.2"  ; Информация о компиляторе
        .section        .note.GNU-stack,"",@progbits    ; Секция для стека GNU
```

Усовершенствование программы добавлением параллельного процесса

```c
#include<stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include"fibonacci.h"


#define MAXBUFSIZE 1024
const int STDIO = 1;


int n = 11;


int main(){
    char buf[MAXBUFSIZE];
    int p[2];
    pipe(p);
    ssize_t rc = fork();
    if (rc == 0) { // a child process
        long long int fibonaccii = fibonacci(n);
        ssize_t l = snprintf(buf, sizeof(buf), "Fibonacci number of %d is %lld\n", n, fibonaccii);
        close(p[0]);
        write(p[1], buf, MAXBUFSIZE);
        close(p[1]);
        _exit(0);
    }
    else if (rc > 0) {
        int crc;
        close(p[1]);
        wait(&crc);
        ssize_t l = read(p[0], buf, MAXBUFSIZE);
        write(STDIO, buf, strlen(buf));
        close(p[0]);
    }
    else {
        printf("Error creating a child process!\n");
    }
    return 0;
}
```

Makefile

```
.PHONY: clean run asm

OBJS = main.o fibonacci.o
CC = gcc
CFLAGS = -Wall -Wextra -Wpedantic -O3 -g

run: fibonacci
	./fibonacci

fibonacci: $(OBJS)
	$(CC) $(CFLAGS) -o fibonacci $(OBJS)

%.o: %.c
	$(CC) $(CFLAGS) -c -o $@ $<

clean:
	rm -f fibonacci $(OBJS)

asm:
	$(CC) -S -o fibonacci.s fibonacci.c
	$(CC) -S -o main.s main.c
```