

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Департамент программной инженерии

**ПРОГРАММА, ВЫЧИСЛЯЮЩАЯ МАКСИМАЛЬНОЕ ПРОСТОЕ ЧИСЛО В
ДИАПАЗОНЕ ОТ 1 ДО БЕЗЗНАКОВОГО МАШИННОГО СЛОВА**

Пояснительная записка

Листов 11

Исполнитель:

студентка группы БПИ196

_____ / А. А. Баранова /

« ____ » _____ 2020 г.

Руководитель:

профессор департамента программной инженерии

факультета компьютерных наук НИУ ВШЭ

_____ / А. И. Легалов /

« ____ » _____ 2020 г.

Москва 2020

СОДЕРЖАНИЕ

1.	ПОСТАНОВКА ЗАДАЧИ	2
2.	ОПИСАНИЕ ПРИМЕНЯЕМЫХ РАСЧЁТНЫХ МЕТОДОВ.....	3
2.1.	Метод проверки данного числа на простоту	3
2.2.	Метод определения границы поиска	3
2.3.	Метод нахождения максимального простого числа.....	3
3.	ОПИСАНИЕ ХРАНЕНИЯ ПРОМЕЖУТОЧНЫХ И ИСХОДНЫХ ДАННЫХ	5
3.1.	Описание исходных данных	5
3.2.	Описание промежуточных данных	5
4.	ТЕСТОВЫЕ ПРИМЕРЫ	6
5.	СПИСОК ЛИТЕРАТУРЫ	7
6.	ТЕКСТ ПРОГРАММЫ	8
6.1.	MaxPrime.asm	8
6.2.	MaxPrimeProcedures.inc	9

1. ПОСТАНОВКА ЗАДАЧИ

1.1. Задание:

Разработать программу, вычисляющую максимальное простое число в диапазоне от 1 до беззнакового машинного слова.

1.2. Разрабатываемая программа должна:

- a** реализовывать алгоритм нахождения максимального простого числа в диапазоне от 1 до беззнакового машинного слова;
- b** выводить в консоль информацию о полученных результатах;
- c** при выводе рассчитанных данных подробно указывать их истинное назначение.

2. ОПИСАНИЕ ПРИМЕНЯЕМЫХ РАСЧЁТНЫХ МЕТОДОВ

2.1. Метод проверки данного числа на простоту

Для определения простоты текущего числа `num`, проверяется его делимость на числа от 2 до $(num - 1)$. Проверка организована как подпрограмма, имитирующая работу с параметрами и возвращающая результат проверки в регистре `eax` (1, если число простое, 0 в противном случае).

Число для проверки хранится в регистре `bx` (используется для проверки делимости) и `sx` (используется для итерации по возможным делителям от $(num - 1)$ до 2, с каждой итерацией уменьшается на 1).

Определение делимости числа `num` на данный делитель в `sx` определяется по значению остатка от деления в регистре `dx` (команда `'div sx'` делит значение в регистре `ax` (куда предварительно помещается значение из `bx`) на значение в регистре `sx` и записывает остаток в регистр `dx`). Если после деления значение в `dx` равно 0, в `eax` записывается 0 и управление передается обратно вызывающему коду. В противном случае значение в `sx` уменьшается на единицу и осуществляется переход к следующей итерации. Когда значение в `sx` достигает 1, в `eax` записывается 1 и управление передается обратно вызывающему коду.

2.2. Метод определения границы поиска

Для определения верхней границы поиска (беззнакового машинного слова) используется переполнение. Число размером в машинное слово при увеличении на единицу дает ноль, поэтому для определения значения беззнакового машинного слова в `esx` записывается число, равное нулю, затем из `sx` вычитается единица. В результате – в регистре `sx` хранится беззнаковое машинное слово – с этого числа начинается поиск максимального простого числа.

2.3. Метод нахождения максимального простого числа

Итерируясь по значению в `sx`, начиная с беззнакового машинного слова, программа проверяет числа на простоту. Если подпрограмма, определяющая простоту числа (см. п. 2.1) вернула 0, значение в `sx` уменьшается на единицу и программа переходит к следующей итерации. Если число оказалось простым (подпрограмма, определяющая простоту числа (см. п. 2.1.) вернула 1), поиск прекращается.

Таким образом, первое (ввиду того, что итерация происходит начиная с верхней границы диапазона, оно будет наибольшим) найденное простое число будет являться результатом работы программы.

Для реализации циклов в программе используется инструкция 'loopw', защищающая от закливания. После каждой итерации значение в `sx` уменьшается на единицу, программа переходит к следующей итерации, если значение в `sx` не равно нулю.

3. ОПИСАНИЕ ХРАНЕНИЯ ПРОМЕЖУТОЧНЫХ И ИСХОДНЫХ ДАННЫХ

3.1. Описание исходных данных

Все исходные данные – строки с информацией, выводимой на различных этапах работы программы. Хранятся в переменных, описанных в секции '.data' программы:

- 1) strRange – информация о диапазоне поиска;
- 2) strStart – вспомогательная строка для запуска программы;
- 3) strResult – строка для вывода результата;
- 4) strResInfo – дополнительная информация о значении найденного числа;
- 5) strEnd – вспомогательная строка для выхода.

3.2. Описание промежуточных данных

Промежуточные данные хранятся в регистрах:

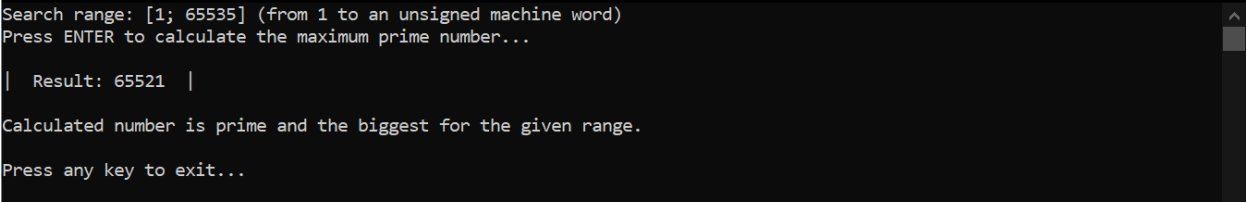
- 1) eax – используется для возврата значений из функции проверки числа на простоту (см. п. 2.1.);
- 2) ax – используется в арифметических операциях при проверки делимости (как делимое);
- 3) bx – используется для хранения текущего числа (при поиске максимального простого числа) при переходе к проверке на простоту (значение в cx будет потеряно при итерации по потенциальным делителям);
- 4) cx – используется для итерации по числам (при поиске максимального простого числа) и потенциальным делителям числа (при проверке на простоту);
- 5) dx – используется в арифметических операциях при проверки делимости (как остаток от деления).

4. ТЕСТОВЫЕ ПРИМЕРЫ

4.1. Описание области допустимых значений входных параметров:

Программа не предполагает входных параметров.

4.2. Примеры работы программы:



```
Search range: [1; 65535] (from 1 to an unsigned machine word)
Press ENTER to calculate the maximum prime number...

| Result: 65521 |

Calculated number is prime and the biggest for the given range.
Press any key to exit...
```

Программа выводит информацию о цели своей работы, а также диапазон поиска.

Программа выводит результат работы, а также информацию о значении полученного числа.

Результат работы программы (число 65521) является корректным: оно простое и следующее за ним простое число 65537 не входит в указанный диапазон.

5. СПИСОК ЛИТЕРАТУРЫ

- 1) flat assembler 1.73 Programmer's Manual: [Электронный ресурс]: Режим доступа: URL: <https://flatassembler.net/docs.php?article=manual>, свободный. (дата обращения: 30.10.2020).
- 2) Программирование на языке ассемблера: [Электронный ресурс]: Режим доступа: URL: <http://natalia.appmat.ru/c&c++/assembler.html>, свободный. (дата обращения: 30.10.2020).
- 3) Программирование на языке ассемблера. Микропроект. Требования к оформлению. 2020-2021 уч.г. [Электронный ресурс]: Режим доступа: URL: <http://softcraft.ru/edu/comparch/tasks/mp01/>, свободный. (дата обращения: 30.10.2020).
- 4) Word (computer architecture) [Электронный ресурс]: Режим доступа: URL: https://en.wikipedia.org/wiki/Prime_number свободный. (дата обращения: 30.10.2020).
- 5) Prime number [Электронный ресурс]: Режим доступа: URL: [https://en.wikipedia.org/wiki/Word_\(computer_architecture\)](https://en.wikipedia.org/wiki/Word_(computer_architecture)) свободный. (дата обращения: 30.10.2020).
- 6) Типы данных в ассемблере [Электронный ресурс]: Режим доступа: URL: <https://prog-cpp.ru/asm-datatypes/> свободный. (дата обращения: 30.10.2020).
- 7) Byte++ – FASM. add, sub, mul, div, neg. Арифметика ассемблер. #3 [Электронный ресурс]: Режим доступа: URL: <https://www.youtube.com/watch?v=gzDiLwIWcCY> свободный. (дата обращения: 30.10.2020).
- 8) Список простых чисел от 1 до 100 000 [Электронный ресурс]: Режим доступа: URL: <http://denisx.ru/tech/prime-number/prime-numbers-list/> свободный. (дата обращения: 30.10.2020).

6. ТЕКСТ ПРОГРАММЫ**6.1. MaxPrime.asm**

```
; Baranova Anastasia BSE196
; Course: Computer System Architecture

; Microproject:
; Develop a program calculating the maximum prime number
; in the range from 1 to an unsigned machine word

format PE console
entry start

include 'win32a.inc'
include 'MaxPrimeProcedures.inc'

section '.data' data readable
    strRange      db 'Search range: [1; %d] (from 1 to an unsigned machine word)', 10, 0
    strStart       db 'Press ENTER to calculate the maximum prime number...', 0
    strResult      db 10, '| Result: %d |', 10, 0
    strResInfo     db 10, 'Calculated number is prime and the biggest for the given range.', 10, 0
    strEnd         db 10, 'Press any key to exit...', 0

;=====

section '.code' code readable executable

start:
    Start          ; Prints information about the process
    FindMaxPrime   ; Finds the maximum prime number in the range
    End            ; Prints result + info

finish:
    call [getch]
    push 0
    call [ExitProcess]

;=====

section '.idata' import data readable

    library kernel, 'kernel32.dll', \
        msvcrt, 'msvcrt.dll'
    import kernel, \
        ExitProcess, 'ExitProcess'
    import msvcrt, \
        printf, 'printf', \
        scanf, 'scanf', \
        getch, '_getch'
```

6.2. MaxPrimeProcedures.inc

```

;----Function checking if number is prime-----
; The number to test is in bx register
; Result puts in eax register (1 if number is prime, 0 otherwise)
macro IsPrime {
    sub cx, 1          ; put first possible divisor (number - 1) in cx

findDivisor:
    ; check if its the last possible divisor (no need to check 1 as a divisor)
    cmp cx, 1
    je prime          ; number is prime

    mov ax, bx        ; put the number in ax
    xor dx, dx        ; dx = 0
    div cx            ; ax / cx, (remainder of the division is in dx)
    cmp dx, 0         ; check if cx is a divisor (the remainder = 0)
    je notPrime       ; number is not prime (cx is a divisor)

    loopw findDivisor ; to the next divisor

prime:
    ; put result in eax (number is prime)
    mov eax, 1
    jmp endIsPrime
notPrime:
    ; put result in eax (number is not prime)
    mov eax, 0
    jmp endIsPrime
endIsPrime:
}
;----End of function-----

;----Function looking for the maximum prime number in the range from 1 to an unsigned machine word--
; Result is in ecx register
macro FindMaxPrime {
    ; find an unsigned machine word value
    xor ecx, ecx
    sub cx, word 1

nextNumber:
    mov bx, cx          ; store the current number in bx register
    IsPrime             ; check if the value in cx(bx) is prime

    mov cx, bx          ; restore the current number in cx register

    cmp eax, 1          ; if the number is prime go to the end
    je endFindMaxPrime

    loopw nextNumber

endFindMaxPrime:
}
;----End of function-----

;----Procedure printing information to the user-----
macro Start {
    xor ecx, ecx
    sub cx, word 1      ; find an unsigned machine word value

    push ecx            ; print information about the search range
    push strRange
    call [printf]

```

```

add esp, 8          ; move the stack pointer

push strStart       ; print start line
call [printf]
call [scanf]        ; wait for user to press enter
add esp, 4          ; move the stack pointer

```

```
endPrintInfo:
```

```
}
```

```
;----End of procedure-----
```

```
;----Procedure printing program result-----
```

```
; Result is in ecx register
```

```
macro End {
```

```
    push ecx          ; print program result
```

```
    push strResult
```

```
    call [printf]
```

```
    add esp, 8        ; move the stack pointer
```

```
    push strResInfo   ; print additional information
```

```
    call [printf]
```

```
    add esp, 4        ; move the stack pointer
```

```
    push strEnd        ; print exit prompt
```

```
    call [printf]
```

```
    add esp, 4        ; move the stack pointer
```

```
endPrintResult:
```

```
}
```

```
;----End of procedure-----
```