Source File: `lab29.asm`
Input: Standard Input
Output: Standard Output
Value: 3

A ***prime number*** is an integer greater than 1 whose only positive divisors are 1 and the integer itself. The Greek mathematician Eratosthenes developed an algorithm, known as the ***Sieve of Eratosthenes***, for finding all prime numbers less than or equal to a given number n—that is, all primes in the range 2 through n. Consider the list of numbers from 2 through n. Two is the first prime number, but the multiples of 2 $(4, 6, 8, \ldots)$ are not, and so they are crossed out in the list. The first number after 2 that was not crossed out is 3, the next prime. We then cross out all higher multiples of 3 $(6, 9, 12, \ldots)$ from the list. The next number not crossed out is 5, the next prime, and so we cross out all higher multiples of 5 $(10, 15, 20, \ldots)$. We repeat this procedure until we reach the first number in the list that has not been crossed out and whose square is greater than n. All the numbers that remain in the list are the primes from 2 through n.

Write a program that uses this sieve method to find all the prime numbers from 2 through n. You are to construct three functions for this assignment. The first will fill an array of bytes, the second will be used for crossing out multiples, and the third will display the primes that remain. A description of the functions as well as client code for testing your implementation is shown in Figure 1, and a sample execution sequence is shown in Figure 2. To use the `Makefile` as distributed in class, add a target of `lab29` to `targets2AsmFiles`.

```
1  [list -]
2  %INCLUDE "Along32.inc"
3  %INCLUDE "Macros_Along.inc"
4  [list +]
5
6  ;-----------------------------------------------------------------------
7  extern  FillArray
8  ; HLL prototype: void FillArray(byte *array, int n);
9  ; Sets array[0] = array[1] = 0 and array[i] = 1, i = 2, 3, ..., (n-1)
10 ; Receives: ESI = starting offset of array
11 ;           ECX = # of elements in array
12 ; Returns:  nothing
13 ;-----------------------------------------------------------------------
14
15 ;-----------------------------------------------------------------------
16 extern  EliminateMultiples
17 ; HLL prototype: void EliminateNultiples(byte *array, int n, int k);
18 ; Implements the following loop:
19 ;      for (i = k + k; i < n; i += k)
20 ;        array[i] = 0
21 ; For example, if k = 2, sets array[4], array[6], array[8], ... to 0
22 ; Receives: ESI = starting offset of array
23 ;           ECX = # of elements in array
24 ;           EDX = value of k
25 ; Returns:  nothing
26 ;-----------------------------------------------------------------------
27
```

Figure 1. `/usr/local/3304/src/lab29main.asm` (Part 1 of 3)

```
28  ;-----------------------------------------------------------------------
29  extern  DisplayArray
30  ; HLL prototype: void DisplayArray(byte *array, int n);
31  ; Implements the following loop:
32  ;      for (i = 2; i < n; ++i)
33  ;        if array[i] == 1 then print i
34  ; Receives: ESI = starting offset of array
35  ;           ECX = # of elements in array
36  ; Returns:  nothing
37  ;-----------------------------------------------------------------------
38
39  SECTION .data
40  array   times   1024 db '#'
41  k       dd      2
42
43  SECTION .bss
44  n       resd    1
45
46  SECTION .text
47          global  _start
48  _start:
49          call    ReadDec                 ; read an unsigned integer
50          mov     [n],eax                 ; move the integer to n
51          inc     dword [n]               ; increment n by 1
52
53          mov     esi,array
54          mov     ecx,[n]
55          call    FillArray
56  .L0:
57          mov     eax,[k]                 ; while (k * k <= n) do
58          mul     dword [k]
59          cmp     eax,[n]
60          ja      .L2
61
62          mov     esi,array
63          add     esi,[k]
64          cmp     byte [esi],0            ;   if (array[k] != 0) then
65          je      .L1
66          mov     esi,array               ;     call EliminateMultiples
67          mov     ecx,[n]
68          mov     edx,[k]
69          call    EliminateMultiples      ;   end if
70  .L1:
71          inc     dword [k]               ;   increment k
72
73          jmp     .L0                     ; end while
74  .L2:
```

Figure 1. `/usr/local/3304/src/lab29main.asm` (Part 2 of 3)

```
75          mov     esi,array
76          mov     ecx,[n]
77          call    DisplayArray
78
79          Exit    {0}
```

Figure 1. `/usr/local/3304/src/lab29main.asm` (Part 3 of 3)

```
1  newuser@csunix ~/3304/29> cp /usr/local/3304/data/29/* .
2  newuser@csunix ~/3304/29> cp /usr/local/3304/src/Makefile .
3  newuser@csunix ~/3304/29> cp /usr/local/3304/src/lab29main.asm .
4  newuser@csunix ~/3304/29> touch lab29.asm
5  newuser@csunix ~/3304/29> make lab29
6  nasm -f elf32  -l lab29main.lst -o lab29main.o lab29main.asm -I/usr/local/3304/include/ -I.
7  nasm -f elf32  -l lab29.lst -o lab29.o lab29.asm -I/usr/local/3304/include/ -I.
8  ld -m elf_i386 --dynamic-linker /lib/ld-linux.so.2 -o lab29 lab29main.o lab29.o \
9  /usr/local/3304/src/Along32.o -lc
10 newuser@csunix ~/3304/29> ../irvine_test.sh lab29 01.dat
11 2
12 3
13 5
14 7
15 11
16 13
17 17
18 19
19 23
20 29
21 31
22 newuser@csunix ~/3304/29> ../irvine_test.sh lab29 01.dat > my.out
23 newuser@csunix ~/3304/29> diff 01.out my.out
24 newuser@csunix ~/3304/29> ../irvine_test.sh lab29 02.dat > my.out
25 newuser@csunix ~/3304/29> diff 02.out my.out
26 newuser@csunix ~/3304/29> ../irvine_test.sh lab29 03.dat > my.out
27 newuser@csunix ~/3304/29> diff 03.out my.out
28 newuser@csunix ~/3304/29> ../irvine_test.sh lab29 04.dat > my.out
29 newuser@csunix ~/3304/29> diff 04.out my.out
30 newuser@csunix ~/3304/29>
```

Figure 2. Commands to Assemble, Link, & Run Lab 29