Application Demonstration

Maple, C and Assembly Language – Performance Comparison

Milorad Pop-Tošić, Igor Skender

Department of Computer Engineering

School of Electrical Engineering, University of Belgrade, Serbia

poptosic@gmail.com, igor.skender@gmail.com

Abstract

We show how to utilize Maple external calling mechanism to speed up function execution by coding them in C and assembly language. Techniques were demonstrated on Jones' algorithm for finding the n-th prime number.

Introduction

In this article, it will be shown how to utilize Maple external calling mechanism in order to solve real problems faster, by calling external functions written in C and assembly language. Maple `define_external` function was used to call routines written in C and assembly language MASM, from DLL libraries [3,4]. These techniques were demonstrated on the example of Jones' algorithm, an algorithm for finding the n-th prime number [1, 2]. Furthermore, we made measurements and comparison of execution time for different implementations of the algorithm: assembly language, C language, and Maple procedures.

Jones' algorithm

Jones' algorithm for finding the n-th prime number states [1, 2]:

Let $P(n)$ be the n-th prime number. Formula $P(n)$, which generates the n-th prime number for given n, is given as:

$$P_n = \sum_{i=0}^{n^2} \left(1 \dot{-} \left(\left(\sum_{j=0}^{i} r\left(\left(j \dot{-} 1\right)!^2, j\right)\right) \dot{-} n\right)\right)$$

where $r(a, b)$ is a function that returns the remainder of division of number a by number b, where $r(a, 0) \equiv a$ and $\dot{-}$ is a function defined as:
$a \dot{-} b = a - b$ ***if*** $a \geq b$ ***else*** $a \dot{-} b = 0$

Function P returns the array of prime numbers $P(1)=2$, $P(2)=3$, $P(3)=5$,...
There is a function in Maple, `ithprime(n)`, which returns values of P from a database. Our goal is to illustrate what can be achieved by connecting Maple to C and assembly language, on the example of Jones' algorithm for finding the function P.

The code for the procedure in Maple for finding P function using Jones' algorithm is given below:

```
> restart:
> M := proc (x::integer, y::integer)
>   if x<y then 0;
>   else x-y;
>   fi;
> end:
> JonesM := proc (n::integer)
>   local m,s,i,p,j,f,k;
>   m := n*n;
>   s := 1;
>   for i from 1 to m do
>           p:=0;
>           for j from 1 to i do
>                     f := 1;
>                     for k from 1 to (j-1) do
>                       f := f*k*k;
>                       f := f mod j;
>                     od;
>                     p := p+f;
>           od;
>           s := s+ M(1, M(p,n));
>   od;
> RETURN (s);
> end:
```

The code in C for finding P function using Jones' algorithm is given below:

Jones.c

```
#define M(x,y) ((x)>(y) ? ((x)-(y)):(0))

int i,j,k,n,m,p,f,s;

int Jones(int n) {
          for ( m=n*n, s=i=1; i<=m; i++) {
                              for ( p=0, j=1; j<=i; j++) {
                                 for ( f=k=1; k<j; k++) { f=f*k*k; f%=j; }
```

```
                                        p+=f;
                                    }
                                    s+=M(1, M(p,n));
                }
                return s;
}
```

In order to call this code from within Maple it should, first, be compiled into a DLL (Dynamic Linking Library). A compiler from Microsoft Visual Studio .NET was chosen at this place. We compile the code by issuing the following command:

```
> cl.exe -LD -Gy -Gz JonesC.c -link /export:Jones
```

It is essential to include keyword /export: followed by the name of the function to be exported and used from within Maple. Note that any other C compiler can be used here as long as it produces a DLL with stdcall calling convention, and exports symbol Jones.

From this point onwards, compiled DLL library JonesC.dll, is connected to Maple by using `define_external` function, as follows:

```
> JonesC:=define_external(
>    'Jones',
>    'n'::integer[4],
>    'RETURN'::integer[4],
>    'LIB'="./JonesC.dll"
>    ):
```

It should be noted that Maple can also call functions from UNIX .so libraries, which have similar function as DLL libraries in Windows.

From this point onwards, the call `JonesC()` from within Maple appears to be exactly the same as a call to a built-in Maple function, although the function Jones from the DLL library gets called.

The procedure in assembly language for finding *P* function using Jones' algorithm is given below [7]:

<u>**JonesASM.asm**</u>

```
.586
.model flat, stdcall

.code
LibMain proc h:DWORD, r:DWORD, u:DWORD
        mov eax, 1
        ret
LibMain Endp
                             ; s=ax   j=bx   k=cx  p=si  i=di
Jones proc n:DWORD
    LOCAL m:DWORD
    LOCAL s:DWORD
    push ebx
    push ecx
    push edx
    mov eax, n
```

```
    mul eax
    mov m, eax
    mov eax, 1
    mov s,  eax
    mov edi, eax
loop1:
    cmp edi, m
    jg  l0
    mov ebx, 1
    xor esi, esi
loop2:
    cmp ebx, edi
    jg  l1
    mov eax, 1
    mov ecx, eax
loop3:
    cmp ecx, ebx
    jge l2
    mul ecx
    mul ecx
    div ebx
    mov eax, edx
    inc ecx
    jmp loop3
l2:
      add esi, eax
      inc ebx
      jmp loop2
l1:
  cmp esi, n
  jg  skip
  inc s
skip:
    inc edi
    jmp loop1
l0:
    pop edx
    pop ecx
    pop ebx

    mov eax, s
ret

Jones endp

End LibMain
```

Apart form this file, one more is needed. It lists functions, which are to be exported from the DLL, which is, in this case, the function Jones.

JonesASM.def

```
LIBRARY JonesASM
EXPORTS Jones
```

Compiling and linking the DLL library, using MASM is done by issuing:

```
  > \masm32\bin\ml /c /coff JonesASM.asm
  > \masm32\bin\Link /SUBSYSTEM:WINDOWS /DLL /DEF:JonesASM.def JonesASM.obj
```

Generated library JonesASM.dll is connected to Maple, as follows:

```
> JonesASM:=define_external(
    'Jones',
    'n'::integer[4],
    'RETURN'::integer[4],
    'LIB'="./JonesASM.dll"
    ):
```

The three shown implementations of function Jones are called from within Maple by issuing the following commands, respectively:

```
> n := 30;
```

$$n := 30 \tag{3.1}$$

```
> JonesM(n);          # Call to Maple procedure
> JonesC(n);          # Call to function in C DLL
> JonesASM(n);        # Call to function in ASM DLL
```

$$113$$
$$113$$
$$113 \tag{3.2}$$

Conclusion

Measurements and comparison of execution time for Jones' algorithm were made for all three presented implementations. To accomplish that, we used Maple `time()` function which returns total processor time used for executing expression. We used this function to calculate execution time for the three solutions, for $n \in \{1..50\}$, by issuing the following commands:

```
> time(JonesM(n));    # Maple procedure execution
```

$$115.874 \tag{4.1}$$

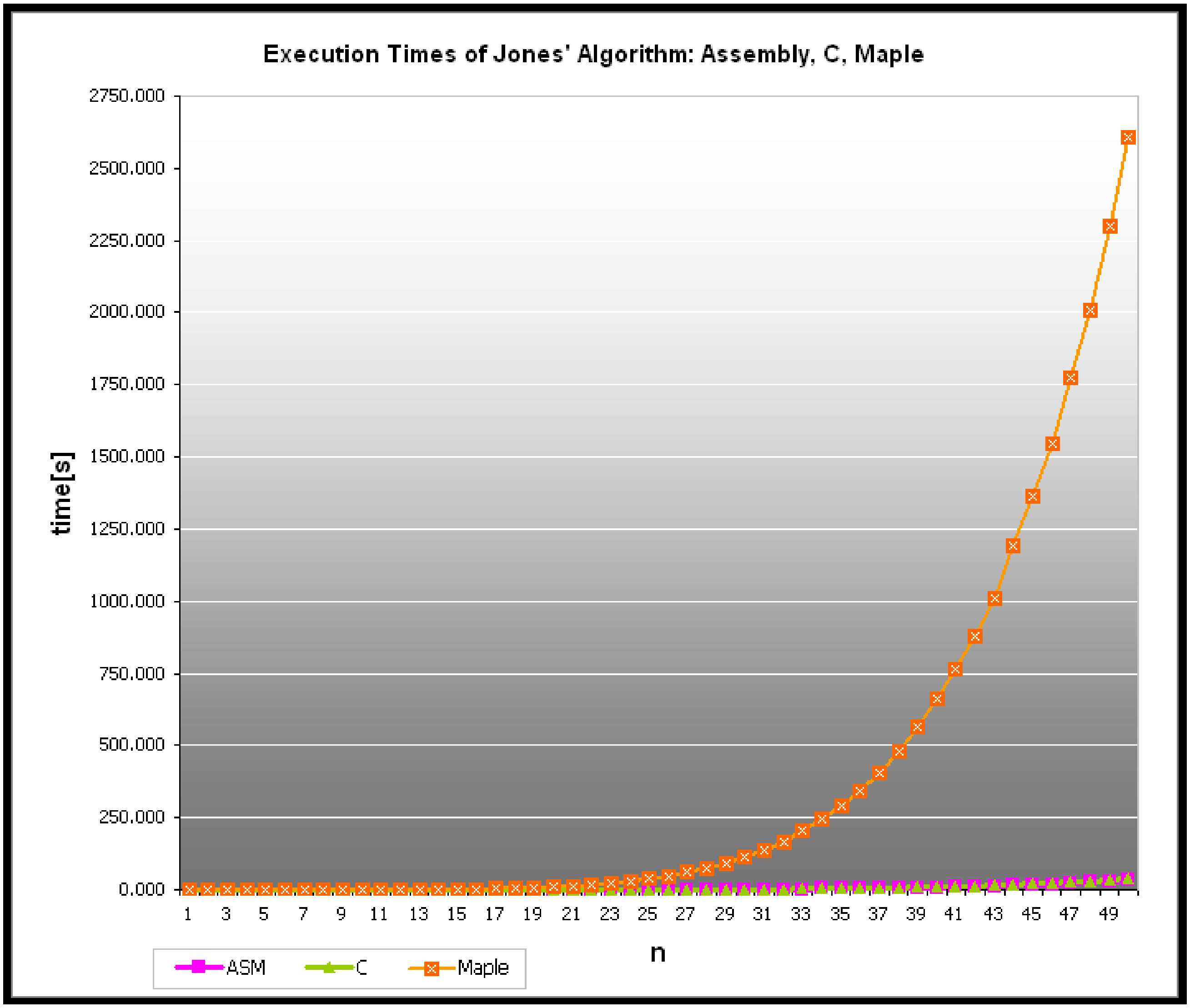
```
> time(JonesC(n));    # C function execution time
```

$$1.907 \tag{4.2}$$

```
> time(JonesASM(n));  # ASM function execution time
```

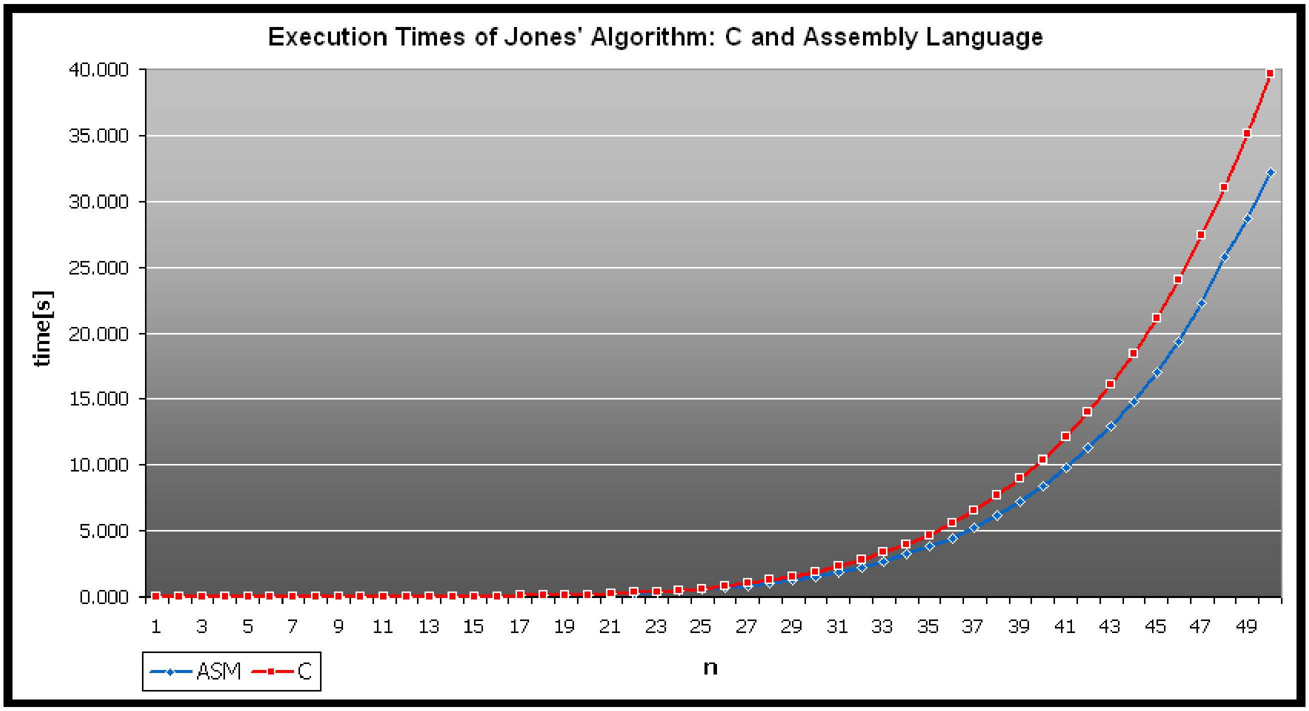
$$1.514 \tag{4.3}$$

Based on measured values, the chart that shows execution times for three presented implementations was created.



It can be observed from this chart that C and assembly language solutions have considerably better performance, compared to Maple procedures. For $n = 50$ procedure in Maple completes in about half an hour, while the same result, by applying C and assembly language solutions, is computed in the matter of seconds. The function that describes the time of execution of Maple procedures rises more sharply, so the differences are even more stressed for larger n.
From what is said can be concluded that Maple procedures are rather slow solution for problems which contain large number of iterations, primarily because Maple code is interpreted, and not compiled. In such cases, it is much more efficient to program in C, or even assembly language.

The chart that follows shows the comparison of execution time for procedures written in assembly language and C. We can observe performance advantage of assembly language over C, which becomes more stressed, as n gets larger. For instance, assembly language implementation is about 25 % faster for $n = 50$. For that reason, putting in more effort in producing assembly language code, especially for loops repeating billion times or more. For loops repeating couple of million times, there is a minor difference between assembly language and C in terms of execution time, so it is simpler to write such a function in C.



Acknowledgement

This article is based on a semester work in course Practicum of Computer Tools in Mathematics, which is taught in the 5th semester of Computer Engineering program on the Faculty of Electrical Engineering, University of Belgrade. We wish to thank assistant professor Dr Branko Malešević for acquainting us with this topic.

References

[1] James P. Jones: Formula for the nth prime number,
Canadian Mathematical Bulletin 18, (1975), pp. 433--434.

[2] James P. Jones; Daihachiro Sato; Hideo Wada; Douglas Wiens:
Diophantine Representation of the Set of Prime Numbers,
The American Mathematical Monthly Vol. 83, No. 6 (Jun., 1976), pp. 449-464

[3] Aleksandrs Mihailovs: Writing DLL in Assembly Language for External Calling in Maple - Technical Report,
Department of Mathematics, Tennessee Technological University
TR No. 2004-5, July 2004, http://www.math.tntech.edu/techreports/TR_2004_5.pdf

[4] Aleksandrs Mihailovs: Writing DLL in Assembly Language for External Calling in Maple,
http://www.maplesoft.com/applications/app_center_view.aspx?AID=1295&CID=9&SCID=63

[5] Michael Monagan: Programming in Maple: The Basics,
Institut für Wissenschaftliches Rechnen ETH-Zentrum, CH-8092 Zürich, Switzerland

[6] David A. Patterson, John L. Henessy: Computer Organization and Design: The Hardware/Software Interface,

Morgan Kaufmann; 3rd edition

[7] Branko Malešević: Examples for the special course – Algorithms in C, (according to the MSc course of Department of Algebra and Mathematical Logic, Faculty of Mathematics, Belgrade 1995).

[8] Veljko Milutinović: The Best Method for Presentation of Research Results,
Department of Computer Engineering, School of Electrical Engineering, University of Belgrade