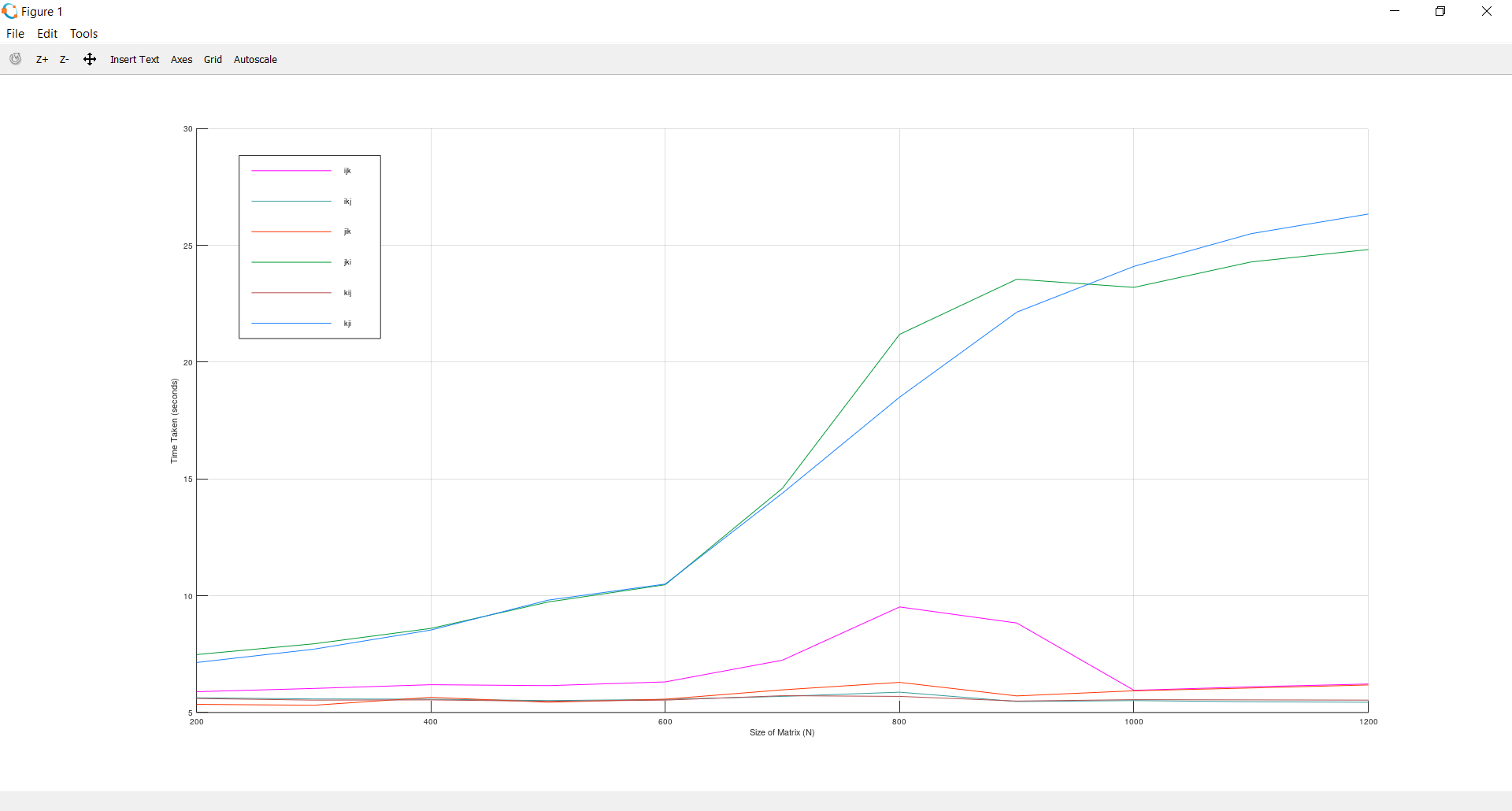
## Compile matrix multiplication and profile the code by changing the size.



On plotting the graph, we can verify that the loop in in the order *jki , kji* are more optimal than other.

Flat profile:

Each sample counts as 0.01 seconds.

no time accumulated

% cumulative self self total

time seconds seconds calls Ts/call Ts/call name

0.00 0.00 0.00 866 0.00 0.00 access\_counter

0.00 0.00 0.00 433 0.00 0.00 add\_sample

0.00 0.00 0.00 433 0.00 0.00 clear

0.00 0.00 0.00 433 0.00 0.00 get\_counter

0.00 0.00 0.00 433 0.00 0.00 has\_converged

0.00 0.00 0.00 433 0.00 0.00 reset

0.00 0.00 0.00 433 0.00 0.00 start\_counter

0.00 0.00 0.00 117 0.00 0.00 ijk

0.00 0.00 0.00 116 0.00 0.00 jki

0.00 0.00 0.00 74 0.00 0.00 kji

0.00 0.00 0.00 54 0.00 0.00 checkresult

0.00 0.00 0.00 54 0.00 0.00 fcyc

0.00 0.00 0.00 54 0.00 0.00 fcyc\_full

0.00 0.00 0.00 54 0.00 0.00 init\_sampler

0.00 0.00 0.00 54 0.00 0.00 run

0.00 0.00 0.00 50 0.00 0.00 kij

0.00 0.00 0.00 42 0.00 0.00 jik

0.00 0.00 0.00 34 0.00 0.00 ikj

0.00 0.00 0.00 1 0.00 0.00 init

% the percentage of the total running time of the

time program used by this function.

cumulative a running sum of the number of seconds accounted

seconds for by this function and those listed above it.

self the number of seconds accounted for by this

seconds function alone. This is the major sort for this

listing.

calls the number of times this function was invoked, if

this function is profiled, else blank.

self the average number of milliseconds spent in this

ms/call function per call, if this function is profiled,

else blank.

total the average number of milliseconds spent in this

ms/call function and its descendents per call, if this

function is profiled, else blank.

name the name of the function. This is the minor sort

for this listing. The index shows the location of

the function in the gprof listing. If the index is

in parenthesis it shows where it would appear in

the gprof listing if it were to be printed.

Copyright (C) 2012-2018 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,

are permitted in any medium without royalty provided the copyright

notice and this notice are preserved.

Call graph (explanation follows)

granularity: each sample hit covers 2 byte(s) no time propagated

index % time self children called name

0.00 0.00 433/866 start\_counter [7]

0.00 0.00 433/866 get\_counter [4]

[1] 0.0 0.00 0.00 866 access\_counter [1]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[2] 0.0 0.00 0.00 433 add\_sample [2]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[3] 0.0 0.00 0.00 433 clear [3]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[4] 0.0 0.00 0.00 433 get\_counter [4]

0.00 0.00 433/866 access\_counter [1]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[5] 0.0 0.00 0.00 433 has\_converged [5]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[6] 0.0 0.00 0.00 433 reset [6]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[7] 0.0 0.00 0.00 433 start\_counter [7]

0.00 0.00 433/866 access\_counter [1]

-----------------------------------------------

0.00 0.00 117/117 fcyc\_full [13]

[8] 0.0 0.00 0.00 117 ijk [8]

-----------------------------------------------

0.00 0.00 116/116 fcyc\_full [13]

[9] 0.0 0.00 0.00 116 jki [9]

-----------------------------------------------

0.00 0.00 74/74 fcyc\_full [13]

[10] 0.0 0.00 0.00 74 kji [10]

-----------------------------------------------

0.00 0.00 54/54 run [15]

[11] 0.0 0.00 0.00 54 checkresult [11]

-----------------------------------------------

0.00 0.00 54/54 run [15]

[12] 0.0 0.00 0.00 54 fcyc [12]

0.00 0.00 54/54 fcyc\_full [13]

-----------------------------------------------

0.00 0.00 54/54 fcyc [12]

[13] 0.0 0.00 0.00 54 fcyc\_full [13]

0.00 0.00 433/433 clear [3]

0.00 0.00 433/433 reset [6]

0.00 0.00 433/433 start\_counter [7]

0.00 0.00 433/433 get\_counter [4]

0.00 0.00 433/433 add\_sample [2]

0.00 0.00 433/433 has\_converged [5]

0.00 0.00 117/117 ijk [8]

0.00 0.00 116/116 jki [9]

0.00 0.00 74/74 kji [10]

0.00 0.00 54/54 init\_sampler [14]

0.00 0.00 50/50 kij [16]

0.00 0.00 42/42 jik [17]

0.00 0.00 34/34 ikj [18]

-----------------------------------------------

0.00 0.00 54/54 fcyc\_full [13]

[14] 0.0 0.00 0.00 54 init\_sampler [14]

-----------------------------------------------

0.00 0.00 54/54 main [29]

[15] 0.0 0.00 0.00 54 run [15]

0.00 0.00 54/54 fcyc [12]

0.00 0.00 54/54 checkresult [11]

-----------------------------------------------

0.00 0.00 50/50 fcyc\_full [13]

[16] 0.0 0.00 0.00 50 kij [16]

-----------------------------------------------

0.00 0.00 42/42 fcyc\_full [13]

[17] 0.0 0.00 0.00 42 jik [17]

-----------------------------------------------

0.00 0.00 34/34 fcyc\_full [13]

[18] 0.0 0.00 0.00 34 ikj [18]

-----------------------------------------------

0.00 0.00 1/1 main [29]

[19] 0.0 0.00 0.00 1 init [19]

-----------------------------------------------

This table describes the call tree of the program, and was sorted by

the total amount of time spent in each function and its children.

Each entry in this table consists of several lines. The line with the

index number at the left hand margin lists the current function.

The lines above it list the functions that called this function,

and the lines below it list the functions this one called.

This line lists:

index A unique number given to each element of the table.

Index numbers are sorted numerically.

The index number is printed next to every function name so

it is easier to look up where the function is in the table.

% time This is the percentage of the `total' time that was spent

in this function and its children. Note that due to

different viewpoints, functions excluded by options, etc,

these numbers will NOT add up to 100%.

self This is the total amount of time spent in this function.

children This is the total amount of time propagated into this

function by its children.

called This is the number of times the function was called.

If the function called itself recursively, the number

only includes non-recursive calls, and is followed by

a `+' and the number of recursive calls.

name The name of the current function. The index number is

printed after it. If the function is a member of a

cycle, the cycle number is printed between the

function's name and the index number.

For the function's parents, the fields have the following meanings:

self This is the amount of time that was propagated directly

from the function into this parent.

children This is the amount of time that was propagated from

the function's children into this parent.

called This is the number of times this parent called the

function `/' the total number of times the function

was called. Recursive calls to the function are not

included in the number after the `/'.

name This is the name of the parent. The parent's index

number is printed after it. If the parent is a

member of a cycle, the cycle number is printed between

the name and the index number.

If the parents of the function cannot be determined, the word

`<spontaneous>' is printed in the `name' field, and all the other

fields are blank.

For the function's children, the fields have the following meanings:

self This is the amount of time that was propagated directly

from the child into the function.

children This is the amount of time that was propagated from the

child's children to the function.

called This is the number of times the function called

this child `/' the total number of times the child

was called. Recursive calls by the child are not

listed in the number after the `/'.

name This is the name of the child. The child's index

number is printed after it. If the child is a

member of a cycle, the cycle number is printed

between the name and the index number.

If there are any cycles (circles) in the call graph, there is an

entry for the cycle-as-a-whole. This entry shows who called the

cycle (as parents) and the members of the cycle (as children.)

The `+' recursive calls entry shows the number of function calls that

were internal to the cycle, and the calls entry for each member shows,

for that member, how many times it was called from other members of

the cycle.

Copyright (C) 2012-2018 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,

are permitted in any medium without royalty provided the copyright

notice and this notice are preserved.

Index by function name

[1] access\_counter [5] has\_converged [16] kij

[2] add\_sample [8] ijk [10] kji

[11] checkresult [18] ikj [6] reset

[3] clear [19] init [15] run

[12] fcyc [14] init\_sampler [7] start\_counter

[13] fcyc\_full [17] jik

[4] get\_counter [9] jki

0.00 0.00 0.00 74 0.00 0.00 kji

0.00 0.00 0.00 54 0.00 0.00 checkresult

0.00 0.00 0.00 54 0.00 0.00 fcyc

0.00 0.00 0.00 54 0.00 0.00 fcyc\_full

0.00 0.00 0.00 54 0.00 0.00 init\_sampler

0.00 0.00 0.00 54 0.00 0.00 run

0.00 0.00 0.00 50 0.00 0.00 kij

0.00 0.00 0.00 42 0.00 0.00 jik

0.00 0.00 0.00 34 0.00 0.00 ikj

0.00 0.00 0.00 1 0.00 0.00 init

% the percentage of the total running time of the

time program used by this function.

cumulative a running sum of the number of seconds accounted

seconds for by this function and those listed above it.

self the number of seconds accounted for by this

seconds function alone. This is the major sort for this

listing.

calls the number of times this function was invoked, if

this function is profiled, else blank.

self the average number of milliseconds spent in this

ms/call function per call, if this function is profiled,

else blank.

total the average number of milliseconds spent in this

ms/call function and its descendents per call, if this

function is profiled, else blank.

name the name of the function. This is the minor sort

for this listing. The index shows the location of

the function in the gprof listing. If the index is

in parenthesis it shows where it would appear in

the gprof listing if it were to be printed.

Copyright (C) 2012-2018 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,

are permitted in any medium without royalty provided the copyright

notice and this notice are preserved.

Call graph (explanation follows)

granularity: each sample hit covers 2 byte(s) no time propagated

index % time self children called name

0.00 0.00 433/866 start\_counter [7]

0.00 0.00 433/866 get\_counter [4]

[1] 0.0 0.00 0.00 866 access\_counter [1]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[2] 0.0 0.00 0.00 433 add\_sample [2]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[3] 0.0 0.00 0.00 433 clear [3]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[4] 0.0 0.00 0.00 433 get\_counter [4]

0.00 0.00 433/866 access\_counter [1]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[5] 0.0 0.00 0.00 433 has\_converged [5]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[6] 0.0 0.00 0.00 433 reset [6]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[7] 0.0 0.00 0.00 433 start\_counter [7]

0.00 0.00 433/866 access\_counter [1]

-----------------------------------------------

0.00 0.00 117/117 fcyc\_full [13]

[8] 0.0 0.00 0.00 117 ijk [8]

-----------------------------------------------

0.00 0.00 116/116 fcyc\_full [13]

[9] 0.0 0.00 0.00 116 jki [9]

-----------------------------------------------

0.00 0.00 74/74 fcyc\_full [13]

[10] 0.0 0.00 0.00 74 kji [10]

-----------------------------------------------

0.00 0.00 54/54 run [15]

[11] 0.0 0.00 0.00 54 checkresult [11]

-----------------------------------------------

0.00 0.00 54/54 run [15]

[12] 0.0 0.00 0.00 54 fcyc [12]

0.00 0.00 54/54 fcyc\_full [13]

-----------------------------------------------

0.00 0.00 54/54 fcyc [12]

[13] 0.0 0.00 0.00 54 fcyc\_full [13]

0.00 0.00 433/433 clear [3]

0.00 0.00 433/433 reset [6]

0.00 0.00 433/433 start\_counter [7]

0.00 0.00 433/433 get\_counter [4]

0.00 0.00 433/433 add\_sample [2]

0.00 0.00 433/433 has\_converged [5]

0.00 0.00 117/117 ijk [8]

0.00 0.00 116/116 jki [9]

0.00 0.00 74/74 kji [10]

0.00 0.00 54/54 init\_sampler [14]

0.00 0.00 50/50 kij [16]

0.00 0.00 42/42 jik [17]

0.00 0.00 34/34 ikj [18]

-----------------------------------------------

0.00 0.00 54/54 fcyc\_full [13]

[14] 0.0 0.00 0.00 54 init\_sampler [14]

-----------------------------------------------

0.00 0.00 54/54 main [29]

[15] 0.0 0.00 0.00 54 run [15]

0.00 0.00 54/54 fcyc [12]

0.00 0.00 54/54 checkresult [11]

-----------------------------------------------

0.00 0.00 50/50 fcyc\_full [13]

[16] 0.0 0.00 0.00 50 kij [16]

-----------------------------------------------

0.00 0.00 42/42 fcyc\_full [13]

[17] 0.0 0.00 0.00 42 jik [17]

-----------------------------------------------

0.00 0.00 34/34 fcyc\_full [13]

[18] 0.0 0.00 0.00 34 ikj [18]

-----------------------------------------------

0.00 0.00 1/1 main [29]

[19] 0.0 0.00 0.00 1 init [19]

-----------------------------------------------

This table describes the call tree of the program, and was sorted by

the total amount of time spent in each function and its children.

Each entry in this table consists of several lines. The line with the

index number at the left hand margin lists the current function.

The lines above it list the functions that called this function,

and the lines below it list the functions this one called.

This line lists:

index A unique number given to each element of the table.

Index numbers are sorted numerically.

The index number is printed next to every function name so

it is easier to look up where the function is in the table.

% time This is the percentage of the `total' time that was spent

in this function and its children. Note that due to

different viewpoints, functions excluded by options, etc,

these numbers will NOT add up to 100%.

self This is the total amount of time spent in this function.

children This is the total amount of time propagated into this

function by its children.

called This is the number of times the function was called.

If the function called itself recursively, the number

only includes non-recursive calls, and is followed by

a `+' and the number of recursive calls.

name The name of the current function. The index number is

printed after it. If the function is a member of a

cycle, the cycle number is printed between the

function's name and the index number.

For the function's parents, the fields have the following meanings:

self This is the amount of time that was propagated directly

from the function into this parent.

children This is the amount of time that was propagated from

the function's children into this parent.

called This is the number of times this parent called the

function `/' the total number of times the function

was called. Recursive calls to the function are not

included in the number after the `/'.

name This is the name of the parent. The parent's index

number is printed after it. If the parent is a

member of a cycle, the cycle number is printed between

the name and the index number.

If the parents of the function cannot be determined, the word

`<spontaneous>' is printed in the `name' field, and all the other

fields are blank.

For the function's children, the fields have the following meanings:

self This is the amount of time that was propagated directly

from the child into the function.

children This is the amount of time that was propagated from the

child's children to the function.

called This is the number of times the function called

this child `/' the total number of times the child

was called. Recursive calls by the child are not

listed in the number after the `/'.

name This is the name of the child. The child's index

number is printed after it. If the child is a

member of a cycle, the cycle number is printed

between the name and the index number.

If there are any cycles (circles) in the call graph, there is an

entry for the cycle-as-a-whole. This entry shows who called the

cycle (as parents) and the members of the cycle (as children.)

The `+' recursive calls entry shows the number of function calls that

were internal to the cycle, and the calls entry for each member shows,

for that member, how many times it was called from other members of

the cycle.

Copyright (C) 2012-2018 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,

are permitted in any medium without royalty provided the copyright

notice and this notice are preserved.

Index by function name

[1] access\_counter [5] has\_converged [16] kij

[2] add\_sample [8] ijk [10] kji

[11] checkresult [18] ikj [6] reset

[3] clear [19] init [15] run

[12] fcyc [14] init\_sampler [7] start\_counter

[13] fcyc\_full [17] jik

[4] get\_counter [9] jki

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[3] 0.0 0.00 0.00 433 clear [3]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[4] 0.0 0.00 0.00 433 get\_counter [4]

0.00 0.00 433/866 access\_counter [1]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[5] 0.0 0.00 0.00 433 has\_converged [5]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[6] 0.0 0.00 0.00 433 reset [6]

-----------------------------------------------

0.00 0.00 433/433 fcyc\_full [13]

[7] 0.0 0.00 0.00 433 start\_counter [7]

0.00 0.00 433/866 access\_counter [1]

-----------------------------------------------

0.00 0.00 117/117 fcyc\_full [13]

[8] 0.0 0.00 0.00 117 ijk [8]

-----------------------------------------------

0.00 0.00 116/116 fcyc\_full [13]

[9] 0.0 0.00 0.00 116 jki [9]

-----------------------------------------------

0.00 0.00 74/74 fcyc\_full [13]

[10] 0.0 0.00 0.00 74 kji [10]

-----------------------------------------------

0.00 0.00 54/54 run [15]

[11] 0.0 0.00 0.00 54 checkresult [11]

-----------------------------------------------

0.00 0.00 54/54 run [15]

[12] 0.0 0.00 0.00 54 fcyc [12]

0.00 0.00 54/54 fcyc\_full [13]

-----------------------------------------------

0.00 0.00 54/54 fcyc [12]

[13] 0.0 0.00 0.00 54 fcyc\_full [13]

0.00 0.00 433/433 clear [3]

0.00 0.00 433/433 reset [6]

0.00 0.00 433/433 start\_counter [7]

0.00 0.00 433/433 get\_counter [4]

0.00 0.00 433/433 add\_sample [2]

0.00 0.00 433/433 has\_converged [5]

0.00 0.00 117/117 ijk [8]

0.00 0.00 116/116 jki [9]

0.00 0.00 74/74 kji [10]

0.00 0.00 54/54 init\_sampler [14]

0.00 0.00 50/50 kij [16]

0.00 0.00 42/42 jik [17]

0.00 0.00 34/34 ikj [18]

-----------------------------------------------

0.00 0.00 54/54 fcyc\_full [13]

[14] 0.0 0.00 0.00 54 init\_sampler [14]

-----------------------------------------------

0.00 0.00 54/54 main [29]

[15] 0.0 0.00 0.00 54 run [15]

0.00 0.00 54/54 fcyc [12]

0.00 0.00 54/54 checkresult [11]

-----------------------------------------------

0.00 0.00 50/50 fcyc\_full [13]

[16] 0.0 0.00 0.00 50 kij [16]

-----------------------------------------------

0.00 0.00 42/42 fcyc\_full [13]

[17] 0.0 0.00 0.00 42 jik [17]

-----------------------------------------------

0.00 0.00 34/34 fcyc\_full [13]

[18] 0.0 0.00 0.00 34 ikj [18]

-----------------------------------------------

0.00 0.00 1/1 main [29]

[19] 0.0 0.00 0.00 1 init [19]

-----------------------------------------------

This table describes the call tree of the program, and was sorted by

the total amount of time spent in each function and its children.

Each entry in this table consists of several lines. The line with the

index number at the left hand margin lists the current function.

The lines above it list the functions that called this function,

and the lines below it list the functions this one called.

This line lists:

index A unique number given to each element of the table.

Index numbers are sorted numerically.

The index number is printed next to every function name so

it is easier to look up where the function is in the table.

% time This is the percentage of the `total' time that was spent

in this function and its children. Note that due to

different viewpoints, functions excluded by options, etc,

these numbers will NOT add up to 100%.

self This is the total amount of time spent in this function.

children This is the total amount of time propagated into this

function by its children.

called This is the number of times the function was called.

If the function called itself recursively, the number

only includes non-recursive calls, and is followed by

a `+' and the number of recursive calls.

name The name of the current function. The index number is

printed after it. If the function is a member of a

cycle, the cycle number is printed between the

function's name and the index number.

For the function's parents, the fields have the following meanings:

self This is the amount of time that was propagated directly

from the function into this parent.

children This is the amount of time that was propagated from

the function's children into this parent.

called This is the number of times this parent called the

function `/' the total number of times the function

was called. Recursive calls to the function are not

included in the number after the `/'.

name This is the name of the parent. The parent's index

number is printed after it. If the parent is a

member of a cycle, the cycle number is printed between

the name and the index number.

If the parents of the function cannot be determined, the word

`<spontaneous>' is printed in the `name' field, and all the other

fields are blank.

For the function's children, the fields have the following meanings:

self This is the amount of time that was propagated directly

from the child into the function.

children This is the amount of time that was propagated from the

child's children to the function.

called This is the number of times the function called

this child `/' the total number of times the child

was called. Recursive calls by the child are not

listed in the number after the `/'.

name This is the name of the child. The child's index

number is printed after it. If the child is a

member of a cycle, the cycle number is printed

between the name and the index number.

If there are any cycles (circles) in the call graph, there is an

entry for the cycle-as-a-whole. This entry shows who called the

cycle (as parents) and the members of the cycle (as children.)

The `+' recursive calls entry shows the number of function calls that

were internal to the cycle, and the calls entry for each member shows,

for that member, how many times it was called from other members of

the cycle.

Copyright (C) 2012-2018 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,

are permitted in any medium without royalty provided the copyright

notice and this notice are preserved.

Index by function name

[1] access\_counter [5] has\_converged [16] kij

[2] add\_sample [8] ijk [10] kji

[11] checkresult [18] ikj [6] reset

[3] clear [19] init [15] run

[12] fcyc [14] init\_sampler [7] start\_counter

[13] fcyc\_full [17] jik

[4] get\_counter [9] jki

0.00 0.00 54/54 fcyc [12]

0.00 0.00 54/54 checkresult [11]

-----------------------------------------------

0.00 0.00 50/50 fcyc\_full [13]

[16] 0.0 0.00 0.00 50 kij [16]

-----------------------------------------------

0.00 0.00 42/42 fcyc\_full [13]

[17] 0.0 0.00 0.00 42 jik [17]

-----------------------------------------------

0.00 0.00 34/34 fcyc\_full [13]

[18] 0.0 0.00 0.00 34 ikj [18]

-----------------------------------------------

0.00 0.00 1/1 main [29]

[19] 0.0 0.00 0.00 1 init [19]

-----------------------------------------------

were internal to the cycle, and the calls entry for each member shows,

for that member, how many times it was called from other members of

the cycle.

Copyright (C) 2012-2018 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,

are permitted in any medium without royalty provided the copyright

notice and this notice are preserved.

1. **Describe the 2019 state of the art Intel processor:**

# INTEL® CORE™ X-SERIES PROCESSOR FAMILY

Intel released the new X-Series Processor in its 10-Gen Core processors family.

This year Intel has released 4 Processor in X-Series as below:

* [Intel® Core™ i9-10980XE](https://www.intel.in/content/www/in/en/products/processors/core/x-series/i9-10980xe.html)
* [Intel® Core™ i9-10940X](https://www.intel.in/content/www/in/en/products/processors/core/x-series/i9-10980xe.html)
* [Intel® Core™ i9-10920X](https://www.intel.in/content/www/in/en/products/processors/core/x-series/i9-10980xe.html)
* [Intel® Core™ i9-10900X](https://www.intel.in/content/www/in/en/products/processors/core/x-series/i9-10980xe.html)

**Major Features:**

* Contains upto 18 cores (36 Threads).
* Upto 4.60 GHz Turbo Frequency.
* 24.75 Mb Intel Smart Cache.

1. **Using gdb, disassemble the object code.**

Source Code:

*#include* <stdio.h>

int square(int a){

*return* a\*a;

}

int main(){

printf("Hello World!\n");

printf("%d\n", square(9));

*return* 0;

}

GDB Output:

ashu@Ashutosh-MSI:/mnt/e/IIIT Sri City/Semester-3/COS$ gcc -g main.c

ashu@Ashutosh-MSI:/mnt/e/IIIT Sri City/Semester-3/COS$ gdb a.out

GNU gdb (Ubuntu 8.1-0ubuntu3.1) 8.1.0.20180409-git

Copyright (C) 2018 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

There is NO WARRANTY, to the extent permitted by law. Type "show copying"

and "show warranty" for details.

This GDB was configured as "x86\_64-linux-gnu".

Type "show configuration" for configuration details.

For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>.

Find the GDB manual and other documentation resources online at:

<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".

Type "apropos word" to search for commands related to "word"...

Reading symbols from a.out...done.

(gdb) disassemble square

Dump of assembler code for function square:

0x000000000000068a <+0>: push %rbp

0x000000000000068b <+1>: mov %rsp, %rbp

0x000000000000068e <+4>: mov %edi, -0x4(%rbp)

0x0000000000000691 <+7>: mov -0x4(%rbp), %eax

0x0000000000000694 <+10>: imul -0x4(%rbp), %eax

0x0000000000000698 <+14>: pop %rbp

0x0000000000000699 <+15>: retq

End of assembler dump.

(gdb) disassemble main

(gdb) disassemble main

Dump of assembler code for function main:

0x000000000000069a <+0>: push %rbp

0x000000000000069b <+1>: mov %rsp, %rbp

0x000000000000069e <+4>: lea 0xaf(%rip), %rdi # 0x754

0x00000000000006a5 <+11>: callq 0x550 <puts@plt>

0x00000000000006aa <+16>: mov $0x9, %edi

0x00000000000006af <+21>: callq 0x68a <square>

0x00000000000006b4 <+26>: mov %eax, %esi

0x00000000000006b6 <+28>: lea 0xa4(%rip), %rdi # 0x761

0x00000000000006bd <+35>: mov $0x0, %eax

0x00000000000006c2 <+40>: callq 0x560 <printf@plt>

0x00000000000006c7 <+45>: mov $0x0, %eax

0x00000000000006cc <+50>: pop %rbp

0x00000000000006cd <+51>: retq

End of assembler dump.

(gdb)