Rapport LABO5 HPC

Bastien Pillonel

Date : 15.05.2024

Config :

- Distrib PopOS 64bits (Ubuntu 22.04)
- AMD Ryzen 7 5800X 8-Core Processor
- AMD K19 (Zen3) architecture
- gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0

Introduction

Pour ce labo il est demandé de faire le profiling d'un code open source C au choix. J'ai choisit d'analyser le code d'une application qui check la syntaxe textuelle d'un fichier src en C.

Lien du code source C-Code-Syntax-Checke

Modification du code de base

Le code de base présentais pas mal d'erreur et n'étais pas fonctionnel. J'ai donc pris soin de corriger les erreurs d'indexage et autre. La version en ligne sur github n'est donc pas fonctionnelle.

Le programme se compile simplement à l'aide du makefile et de la commande `make`. Ensuite on lance le programme comme suit `./Checker <nom_du_fichier_a_analyser>`.

Profiling

Perf tool

Pour profiler le code j'ai décidé d'utiliser deux fichiers C. Un avec des erreur input_error.txt et un sans erreur input_clean.c.

Les deux premiers events que je souhaite mesurer sont le nombre d'instruction retirées ainsi que le nombre de cycle cpu effectué. Pour se faire je lance la commande :

```
perf stat -e cpu-cycles,instructions ./Checker <input_file>
```

Résultat sur input_clean.c:

```
Performance counter stats for './Checker ./src/input_clean.c':

        4,464,980      cpu-cycles
        3,859,545      instructions                     #
            0.86  insn per cycle
```

```
5
6       0.003168161 seconds time elapsed
7
8       0.000000000 seconds user
9       0.003145000 seconds sys
```

Résultat sur input_error.txt:

```
1  Performance counter stats for './Checker ./src/input_error.txt':
2
3          3,821,949      cpu-cycles
4          3,061,866      instructions                     #
               0.80  insn per cycle
5
6       0.013873980 seconds time elapsed
7
8       0.003810000 seconds user
9       0.000000000 seconds sys
```

On obtient ensuite les ratios suivants:

Input_clean ratio = 0.86 [instr/cycle] Input_error ratio = 0.80 [instr/cycle]

Ensuite je mesure le ratio des caches miss sur chaque fichier:

Résultat sur input_clean.c:

```
1  Performance counter stats for './Checker ./src/input_clean.c':
2
3            967,607      L1-dcache-loads
4             20,274      L1-dcache-load-misses            #
                  2.10% of all L1-dcache accesses
5
6       0.002718538 seconds time elapsed
7
8       0.000000000 seconds user
9       0.002805000 seconds sys
```

Résultat sur input_error.txt:

```
1  Performance counter stats for './Checker ./src/input_error.txt':
2
3            763,950      L1-dcache-loads
4             20,290      L1-dcache-load-misses            #
                  2.66% of all L1-dcache accesses
5
6       0.001629154 seconds time elapsed
7
8       0.001754000 seconds user
```

```
       0.000000000 seconds sys
```

Input_clean cache misses = 2.10% Input_error cache misses = 2.66%

Enfin je regarde le nombre de branch misses :

Résultat sur input_clean.c:

```
 Performance counter stats for './Checker ./src/input_clean.c':

            39,560      branch-misses                     #
                5.75% of all branches
           687,736      branch-instructions

       0.001892185 seconds time elapsed

       0.000000000 seconds user
       0.001987000 seconds sys
```

Résultat sur input_error.txt:

```
 Performance counter stats for './Checker ./src/input_error.txt':

            40,046      branch-misses                     #
                8.52% of all branches
           469,801      branch-instructions

       0.002201889 seconds time elapsed

       0.002299000 seconds user
       0.000000000 seconds sys
```

Input_clean branch misses = 5.75% Input_error branch misses = 8.52%

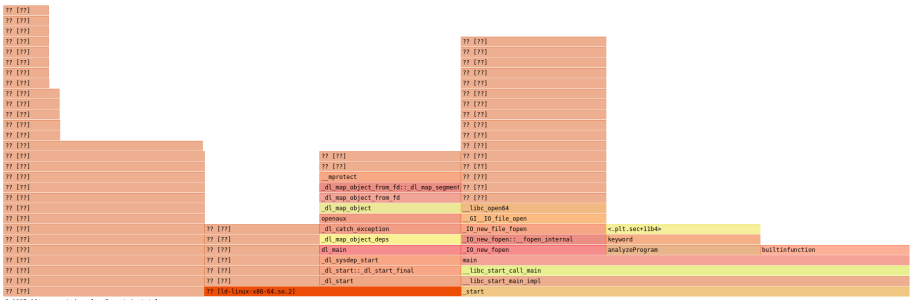
Hotspot tool

Lancement de la mesure sur hotspot à l'aide de la commande suivante:

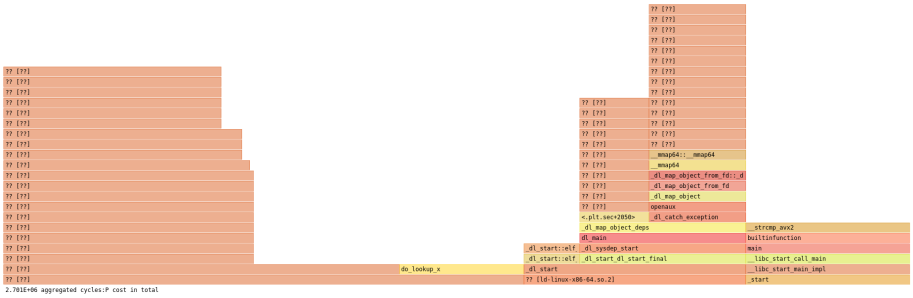
```
perf record -o <data_file_location> --call-graph dwarf --aio
    --sample-cpu <path_to_checker_exec> <file_to_analyze>
```

Je lui demande tout d'abord un aperçu des cycles cpu.

Input_clean.c:



Input_error.txt:



L'analyse des points chauds ne m'indique pas réelement de problème dans mon programme. Malheureusement je n'ai pas réussi à afficher tous les appels à mes fonctions pour une raison que j'ignore. Néanmoins il semble que l'appel aux fonctions de la shared library ait un overhead assez conséquent. Je checkerai donc les appels aux fonctions de la librairie standard.

Valgrind

Pour cette partie j'ai décidé d'utiliser l'outil callgrind qui permet de mieux représenter le nombre d'appel aux fonctions. Comme mon programme utilise beaucoup de boucle pour parcourir le fichier source de code C, il est utile de voir quelle sont les fonctions qui sont le plus souvent appelés afin de voir si une optimisation est possible.

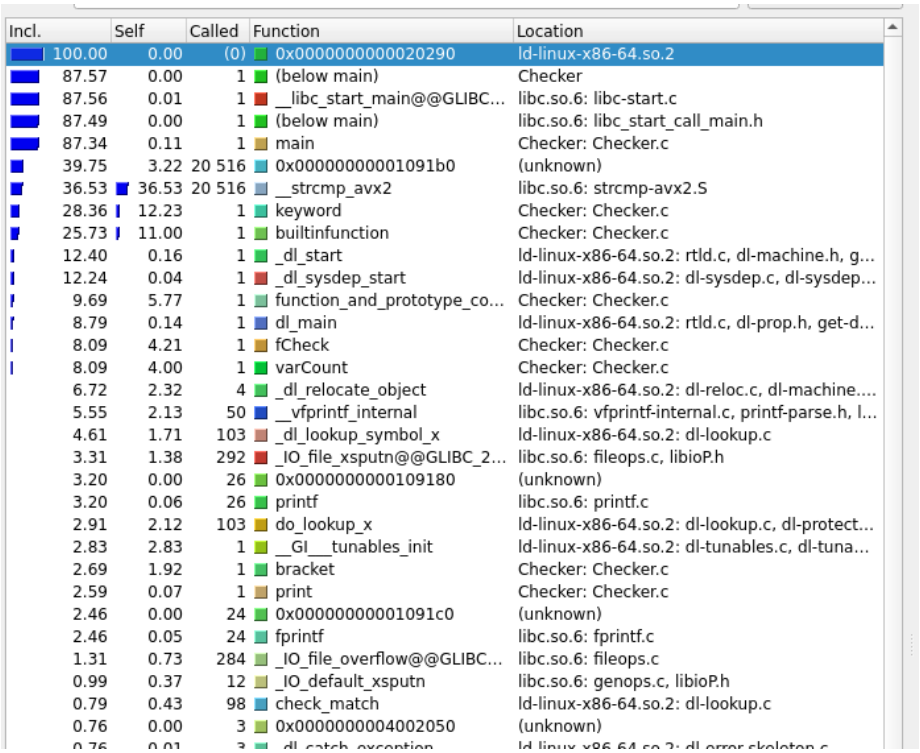
Commande utilisé :

```
1 valgrind --tool=callgrind ./Checker <fichier_a_analyzer>
```

Ensuite pour la visualisation :

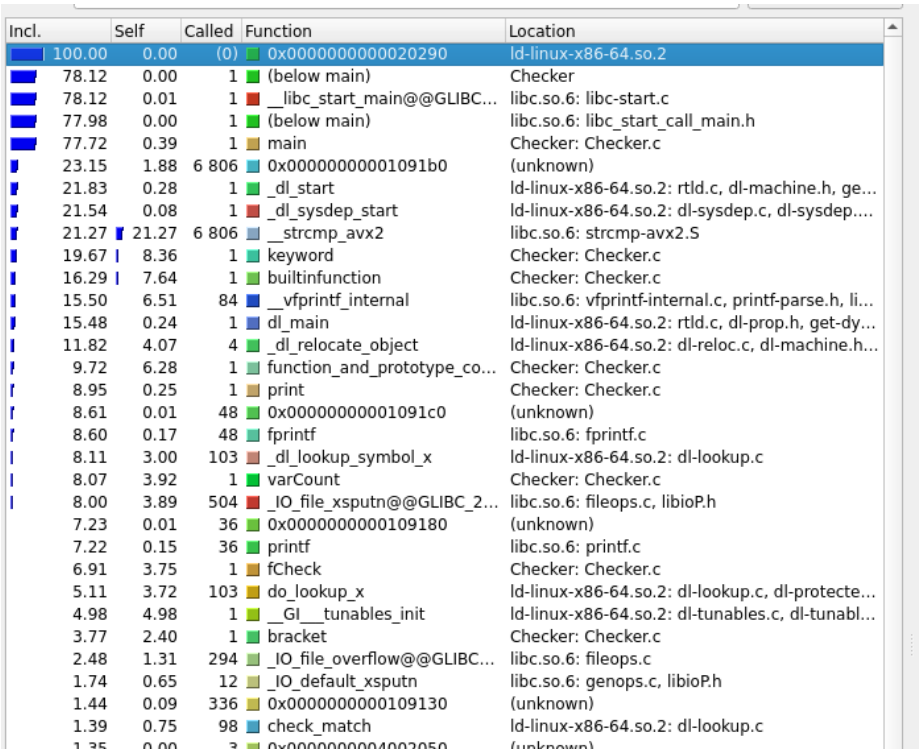
```
1 kcachegrind callgrind.out.<PID_correspondant>
```

Résultat input_clean.c:



Incl.	Self	Called	Function	Location
100.00	0.00	(0)	0x0000000000020290	ld-linux-x86-64.so.2
87.57	0.00	1	(below main)	Checker
87.56	0.01	1	__libc_start_main@@GLIBC...	libc.so.6: libc-start.c
87.49	0.00	1	(below main)	libc.so.6: libc_start_call_main.h
87.34	0.11	1	main	Checker: Checker.c
39.75	3.22	20 516	0x00000000001091b0	(unknown)
36.53	36.53	20 516	__strcmp_avx2	libc.so.6: strcmp-avx2.S
28.36	12.23	1	keyword	Checker: Checker.c
25.73	11.00	1	builtinfunction	Checker: Checker.c
12.40	0.16	1	_dl_start	ld-linux-x86-64.so.2: rtld.c, dl-machine.h, g...
12.24	0.04	1	_dl_sysdep_start	ld-linux-x86-64.so.2: dl-sysdep.c, dl-sysdep...
9.69	5.77	1	function_and_prototype_co...	Checker: Checker.c
8.79	0.14	1	dl_main	ld-linux-x86-64.so.2: rtld.c, dl-prop.h, get-d...
8.09	4.21	1	fCheck	Checker: Checker.c
8.09	4.00	1	varCount	Checker: Checker.c
6.72	2.32	4	_dl_relocate_object	ld-linux-x86-64.so.2: dl-reloc.c, dl-machine....
5.55	2.13	50	__vfprintf_internal	libc.so.6: vfprintf-internal.c, printf-parse.h, l...
4.61	1.71	103	_dl_lookup_symbol_x	ld-linux-x86-64.so.2: dl-lookup.c
3.31	1.38	292	_IO_file_xsputn@@GLIBC_2...	libc.so.6: fileops.c, libioP.h
3.20	0.00	26	0x0000000000109180	(unknown)
3.20	0.06	26	printf	libc.so.6: printf.c
2.91	2.12	103	do_lookup_x	ld-linux-x86-64.so.2: dl-lookup.c, dl-protect...
2.83	2.83	1	__GI___tunables_init	ld-linux-x86-64.so.2: dl-tunables.c, dl-tuna...
2.69	1.92	1	bracket	Checker: Checker.c
2.59	0.07	1	print	Checker: Checker.c
2.46	0.00	24	0x00000000001091c0	(unknown)
2.46	0.05	24	fprintf	libc.so.6: fprintf.c
1.31	0.73	284	_IO_file_overflow@@GLIBC...	libc.so.6: fileops.c
0.99	0.37	12	_IO_default_xsputn	libc.so.6: genops.c, libioP.h
0.79	0.43	98	check_match	ld-linux-x86-64.so.2: dl-lookup.c
0.76	0.00	3	0x0000000004002050	(unknown)

Résultat input_error.txt:



Incl.	Self	Called	Function	Location
100.00	0.00	(0)	0x0000000000020290	ld-linux-x86-64.so.2
78.12	0.00	1	(below main)	Checker
78.12	0.01	1	__libc_start_main@@GLIBC...	libc.so.6: libc-start.c
77.98	0.00	1	(below main)	libc.so.6: libc_start_call_main.h
77.72	0.39	1	main	Checker: Checker.c
23.15	1.88	6 806	0x00000000001091b0	(unknown)
21.83	0.28	1	_dl_start	ld-linux-x86-64.so.2: rtld.c, dl-machine.h, ge...
21.54	0.08	1	_dl_sysdep_start	ld-linux-x86-64.so.2: dl-sysdep.c, dl-sysdep....
21.27	21.27	6 806	__strcmp_avx2	libc.so.6: strcmp-avx2.S
19.67	8.36	1	keyword	Checker: Checker.c
16.29	7.64	1	builtinfunction	Checker: Checker.c
15.50	6.51	84	__vfprintf_internal	libc.so.6: vfprintf-internal.c, printf-parse.h, li...
15.48	0.24	1	dl_main	ld-linux-x86-64.so.2: rtld.c, dl-prop.h, get-dy...
11.82	4.07	4	_dl_relocate_object	ld-linux-x86-64.so.2: dl-reloc.c, dl-machine.h...
9.72	6.28	1	function_and_prototype_co...	Checker: Checker.c
8.95	0.25	1	print	Checker: Checker.c
8.61	0.01	48	0x00000000001091c0	(unknown)
8.60	0.17	48	fprintf	libc.so.6: fprintf.c
8.11	3.00	103	_dl_lookup_symbol_x	ld-linux-x86-64.so.2: dl-lookup.c
8.07	3.92	1	varCount	Checker: Checker.c
8.00	3.89	504	_IO_file_xsputn@@GLIBC_2...	libc.so.6: fileops.c, libioP.h
7.23	0.01	36	0x0000000000109180	(unknown)
7.22	0.15	36	printf	libc.so.6: printf.c
6.91	3.75	1	fCheck	Checker: Checker.c
5.11	3.72	103	do_lookup_x	ld-linux-x86-64.so.2: dl-lookup.c, dl-protecte...
4.98	4.98	1	__GI___tunables_init	ld-linux-x86-64.so.2: dl-tunables.c, dl-tunabl...
3.77	2.40	1	bracket	Checker: Checker.c
2.48	1.31	294	_IO_file_overflow@@GLIBC...	libc.so.6: fileops.c
1.74	0.65	12	_IO_default_xsputn	libc.so.6: genops.c, libioP.h
1.44	0.09	336	0x0000000000109130	(unknown)
1.39	0.75	98	check_match	ld-linux-x86-64.so.2: dl-lookup.c

On remarque ici qu'un grand nombre d'appel à strcmp est effectué. Ce qui fait sens puisque notre programme va analyser la syntax du code. Mais ce qui peut potentiellement nous faire perdre du temps si certains appelle sont superflues.

Optimisation de strcmp

Pour optimiser le nombre d'appelle à strcmp je décide d'observer chaque appel:

Dans la fonction keyword:

```
1 // Check if the current word is a C keyword
2 for (k = 0; k < NUMBER_OF_KEYWORDS; k++)
3 {
4         if (strcmp(currentWord, sync[k]) == 0)
5         {
6                 // Print the keyword and its line number
7                 printf("Line %d: %s\n", arr[i].lineno,
                      currentWord);
8         }
9 }
```

Ici par exemple comme les keywords sont unique, si l'on trouve un match dans la liste nous n'avons plus besoin de terminer la recherche on peut donc placer

un break après le print.

Dans la fonction builtinfunction:

```
for (l = 0; l < 30; l++)
{
	if (strcmp(currentWord, sync[l]) == 0)
	{
		total += 1; // Increment total if the word is a
		    built in function.
	}
}
```

Ici on ne peut pas vraiment optimiser le nombre de fonction builtin à optimiser car on est obligé de parcourir toute la liste afin de voir si un matching est présent

Pareil pour la fonction function_and_prototype. Néanmoins la recherche du main se fait dans une autre boucle qui reparcours tous le fichier. Cela ralentis extrêmement car on double le percours du fichier pour rien. On peut directement intégrer le check dans la boucle de parcours principale.

La dernière boucle s'occupe de compter le nombre de prototype qui (dans un programme ou chaque fonction est définie et déclarée) doit être égale au nombre de fonction moins 1 (pour le main). Encore une fois cette boucle n'est pas nécessaire et le check peut se faire directement dans la boucle principale.

Dans la fonction de comptage des variables on check si le mot est une variable avant de faire le check de si la ligne est une déclaration/définition de fonction.

```
// Check if the current word matches any of the synchronization
    words
for (size_t l = 0; l < 6; l++)
{
	if (strcmp(currnetWord, sync[l]) == 0)
	{
		pos = j;

		// Check if the line does not end with ')' to
		    ensure it is a function declaration
		if ((str[strlength - 3] != ')') && (str[strlength
		    - 2] != ')'))
		{
			// Loop through the line to count commas
			    and semicolons, indicating variable
			    declarations
			for (size_t x = 0; x < strlength; x++)
			{
			if (str[x] == ',' || str[x] == ';')
			{
```

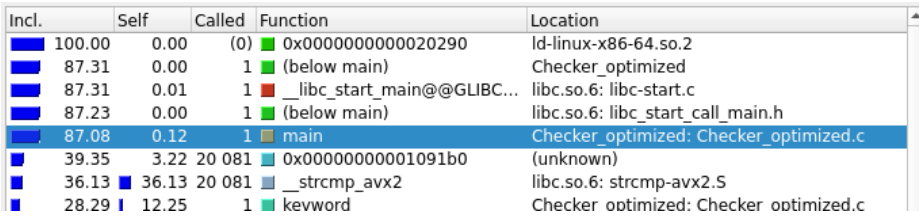
```
16                                  total[l]++;
17                          }
18                          }
19                  }
20          }
21  }
```

Si l'on inverse les deux vérification on évitera certaines comparaisons inutiles.

Test de l'optimisation

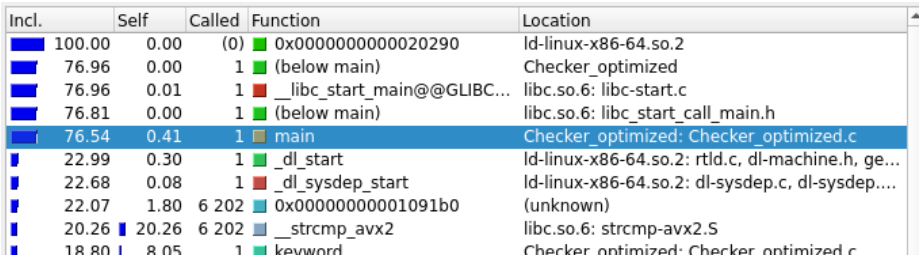
Le fichier optimisé s'appelle Checker_optimized.c.

Résultat du nombre d'appel à strcmp input_clean.c:



Incl.	Self	Called	Function	Location
100.00	0.00	(0)	0x0000000000020290	ld-linux-x86-64.so.2
87.31	0.00	1	(below main)	Checker_optimized
87.31	0.01	1	__libc_start_main@@GLIBC...	libc.so.6: libc-start.c
87.23	0.00	1	(below main)	libc.so.6: libc_start_call_main.h
87.08	0.12	1	main	Checker_optimized: Checker_optimized.c
39.35	3.22	20 081	0x00000000001091b0	(unknown)
36.13	36.13	20 081	__strcmp_avx2	libc.so.6: strcmp-avx2.S
28.29	12.25	1	keyword	Checker_optimized: Checker_optimized.c

Résultat du nombre d'appel à strcmp input_error.txt:



Incl.	Self	Called	Function	Location
100.00	0.00	(0)	0x0000000000020290	ld-linux-x86-64.so.2
76.96	0.00	1	(below main)	Checker_optimized
76.96	0.01	1	__libc_start_main@@GLIBC...	libc.so.6: libc-start.c
76.81	0.00	1	(below main)	libc.so.6: libc_start_call_main.h
76.54	0.41	1	main	Checker_optimized: Checker_optimized.c
22.99	0.30	1	_dl_start	ld-linux-x86-64.so.2: rtld.c, dl-machine.h, ge...
22.68	0.08	1	_dl_sysdep_start	ld-linux-x86-64.so.2: dl-sysdep.c, dl-sysdep....
22.07	1.80	6 202	0x00000000001091b0	(unknown)
20.26	20.26	6 202	__strcmp_avx2	libc.so.6: strcmp-avx2.S
18.80	8.05	1	keyword	Checker_optimized: Checker_optimized.c

On remarque une légère baisse d'environs 500 appels.

Ensuite je relance le script measure.sh en changeant l'application à profiler pour comparer avec la baseline du début.

```
1  # started on Sun May 19 17:54:40 2024
2
3
4   Performance counter stats for './Checker_optimized
        ./src/input_clean.c':
5
6           2,866,494      cpu-cycles
7           2,429,009      instructions                     #
                0.85  insn per cycle
8
9         0.002637315 seconds time elapsed
```

```
10
11        0.002676000 seconds user
12        0.000000000 seconds sys
13
14
15 # started on Sun May 19 17:54:40 2024
16
17
18  Performance counter stats for './Checker_optimized
      ./src/input_error.txt':
19
20          2,279,781      cpu-cycles
21          1,822,003      instructions                     #
              0.80  insn per cycle
22
23        0.001623332 seconds time elapsed
24
25        0.001738000 seconds user
26        0.000000000 seconds sys
27
28
29 # started on Sun May 19 17:54:40 2024
30
31
32  Performance counter stats for './Checker_optimized
      ./src/input_clean.c':
33
34            745,492      L1-dcache-loads
35             18,370      L1-dcache-load-misses            #
                  2.46% of all L1-dcache accesses
36
37        0.001598304 seconds time elapsed
38
39        0.001687000 seconds user
40        0.000000000 seconds sys
41
42
43 # started on Sun May 19 17:54:40 2024
44
45
46  Performance counter stats for './Checker_optimized
      ./src/input_error.txt':
47
48            622,549      L1-dcache-loads
49             18,948      L1-dcache-load-misses            #
                  3.04% of all L1-dcache accesses
```

```
50
51        0.001449956 seconds time elapsed
52
53        0.001463000 seconds user
54        0.000000000 seconds sys
55
56
57 # started on Sun May 19 17:54:40 2024
58
59
60  Performance counter stats for './Checker_optimized
      ./src/input_clean.c':
61
62             36,485      branch-misses                    #
                   7.67% of all branches
63            475,465      branch-instructions
64
65        0.001552869 seconds time elapsed
66
67        0.001639000 seconds user
68        0.000000000 seconds sys
69
70
71 # started on Sun May 19 17:54:40 2024
72
73
74  Performance counter stats for './Checker_optimized
      ./src/input_error.txt':
75
76             37,198      branch-misses                    #
                   10.15% of all branches
77            366,386      branch-instructions
78
79        0.001601821 seconds time elapsed
80
81        0.001681000 seconds user
82        0.000000000 seconds sys
```

On observe une diminution de presque moitié du nombre d'instruction ainsi que du nombre de cycle CPU et une très légère augmentation du cache miss et du branch predict ce qui est sûrement dû au fait d'avoir rajouter quelques test en plus dans la boucle principale du check des fonctions et prototype.

Globalement cela reste une bonne amélioration des performances du code.