Université de Versailles Saint-Quentin en Yvlines

RAPPORT DU PROJET OBHPC

BEN NACER

MERIEM

N° étudiant 22207572

Le but de ce TP est de mesurer les performances de l'exécution d'un programme en utilisant différents compilateurs : **GCC**, **CLANG**, **ICC**, **ICX**. Pour se faire on passe par les étapes suivantes :

1- Avant d'entamer le TP sur un cartable numérique, nous nous sommes assuré que le PC est connecté au secteur. Bien évidement, nous avons effectué le TP sur **Mint**. Nous allons aussi le partager sur **GitHub** en tenant compte des commandes **Git**.

2- On va par la suite s'assurer que notre **CPU** tourne à une fréquence stable, on regarde combien de cœurs contient notre **CPU** avec la commande : **lscpu** on a comme résultat :

```
Cœur(s) par socket :                     4
```

Ensuite avec la commande **sudo cpupowers –c 2 frequency –set –g performance** on fixe le cœur numéro 2 sur une fréquence :

```
uvsql@uvsql-Latitude-5590:~/TD2_OBHPC$ sudo cpupower -c 2 frequency-set -g performance
[sudo] Mot de passe de uvsql :
Setting cpu: 2
uvsql@uvsql-Latitude-5590:~/TD2_OBHPC$
```

On va maintenant extraire les informations nécessaires ainsi que la consultation des fichiers sur le **CPU** :

```
uvsql@uvsql-Latitude-5590:~/TD2_OBHPC$ lscpu
Architecture :                           x86_64
  Mode(s) opératoire(s) des processeurs :  32-bit, 64-bit
  Address sizes:                         39 bits physical, 48 bits virtual
  Boutisme :                             Little Endian
Processeur(s) :                          8
  Liste de processeur(s) en ligne :      0-7
Identifiant constructeur :               GenuineIntel
  Nom de modèle :                        Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
    Famille de processeur :              6
    Modèle :                             142
    Thread(s) par cœur :                 2
    Cœur(s) par socket :                 4
    Socket(s) :                          1
    Révision :                           10
    Vitesse maximale du processeur en MHz : 3400,0000
    Vitesse minimale du processeur en MHz : 400,0000
    BogoMIPS :                           3600.00
    Drapaux :                            fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
                                          clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdt
                                         scp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology n
```

```
                                        nt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefet
                                        ch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow v
                                        nmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi
                                        2 erms invpcid mpx rdseed adx smap clflushopt intel_pt xsaveopt xsavec
                                        xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hw
                                        p_epp md_clear flush_l1d arch_capabilities
irtualization features:
 Virtualisation :                       VT-x
aches (sum of all):
 L1d:                                   128 KiB (4 instances)
 L1i:                                   128 KiB (4 instances)
 L2:                                    1 MiB (4 instances)
 L3:                                    6 MiB (1 instance)
UMA:
 Nœud(s) NUMA :                         1
 Nœud NUMA 0 de processeur(s) :         0-7
ulnerabilities:
 Itlb multihit:                         KVM: Mitigation: VMX disabled
 L1tf:                                  Mitigation; PTE Inversion; VMX conditional cache flushes, SMT vulnerabl
                                        e
 Mds:                                   Mitigation; Clear CPU buffers; SMT vulnerable
 Meltdown:                              Mitigation; PTI
 Mmio stale data:                       Mitigation; Clear CPU buffers; SMT vulnerable
 Spec store bypass:                     Mitigation; Speculative Store Bypass disabled via prctl and seccomp
 Spectre v1:                            Mitigation; usercopy/swapgs barriers and __user pointer sanitization
 Spectre v2:                            Mitigation; Retpolines, IBPB conditional, IBRS_FW, STIBP conditional, R
                                        SB filling
 Srbds:                                 Mitigation; Microcode
 Tsx async abort:                       Not affected
```

On va les enregistrer dans un fichier **cpuinfo.dat** à partir de la commande : **lscpu > cpuinfo.dat.**

Donc, notre programme va être exécuté principalement sur le cœur 2, on fixe sur le compilateur GCC avec la commande **Make CC=gcc,** avec les 3 flags 01 (ou 02 ou 03).

Ensuite, la commande : **task –c 2 ./ dgemm 100 100**, en choisissant r = 100 et n=100 par défaut. On aura les informations de performance suivantes qu'on extrait dans un fichier .dat (gcc1, gcc2, gcc3 ou clang1, clang2, clang3) avec la commande :

sed -n '1p;'"$i"'p' ./gcc/gcc3/gcc3.txt > ./gcc/gcc3/gcc3_$i.txt

title;	KiB;	MiB;	GiB;	n;	r;	min;	max;	mean;	stddev (%);	MiB/s
IJK;	234.375;	0.229;	0.000;	100;	100;	931071.660;	941454.130;	934916.615;	2151.213 (0.230 %);	81.605
IKJ;	234.375;	0.229;	0.000;	100;	100;	600445.900;	620665.840;	606278.039;	4140.148 (0.683 %);	125.840
IEX;	234.375;	0.229;	0.000;	100;	100;	652865.310;	672071.680;	654721.549;	4391.121 (0.671 %);	116.529
UNROLL4;	234.375;	0.229;	0.000;	100;	100;	477655.880;	486399.990;	478899.113;	1694.476 (0.354 %);	159.311
UNROLL8;	234.375;	0.229;	0.000;	100;	100;	502936.610;	505441.940;	503882.046;	739.442 (0.147 %);	151.412
CBLAS;	234.375;	0.229;	0.000;	100;	100;	226579.730;	227816.800;	227115.961;	303.429 (0.134 %);	335.925

Maintenant, on doit faire la répartition de ces informations dans d'autres fichiers afin de pouvoir les comparer en se basant sur les graphes grâce à la commande :

tail -n $l ./gcc/gcc3/gcc3.txt | cut -d";" -f1,11 > ./gcc/gcc3/gcc3.dat

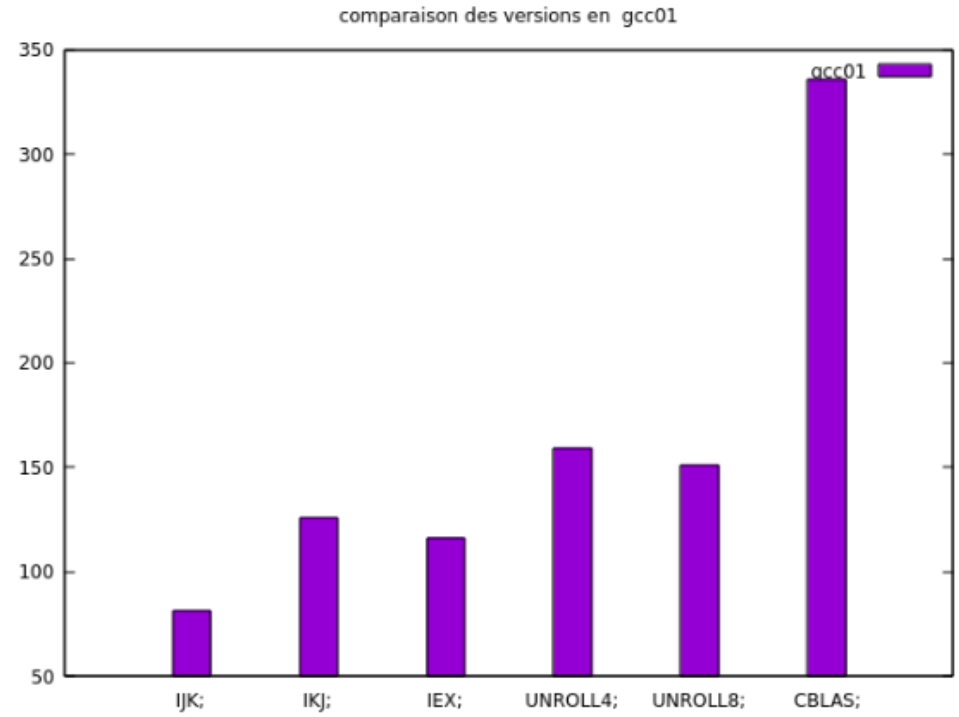
On refait les étapes en fixant le flag 02 et le flag 03.

Après avoir exécuté en GCC, on passe au compilateur CLANG, **make clean**, ensuite **make CC=clang,** exactement les mêmes étapes précédantes avec les 3 flags 01, 02 et 03**.**

Toutes ces commandes vont être mis dan un seul script : **script.sh.**

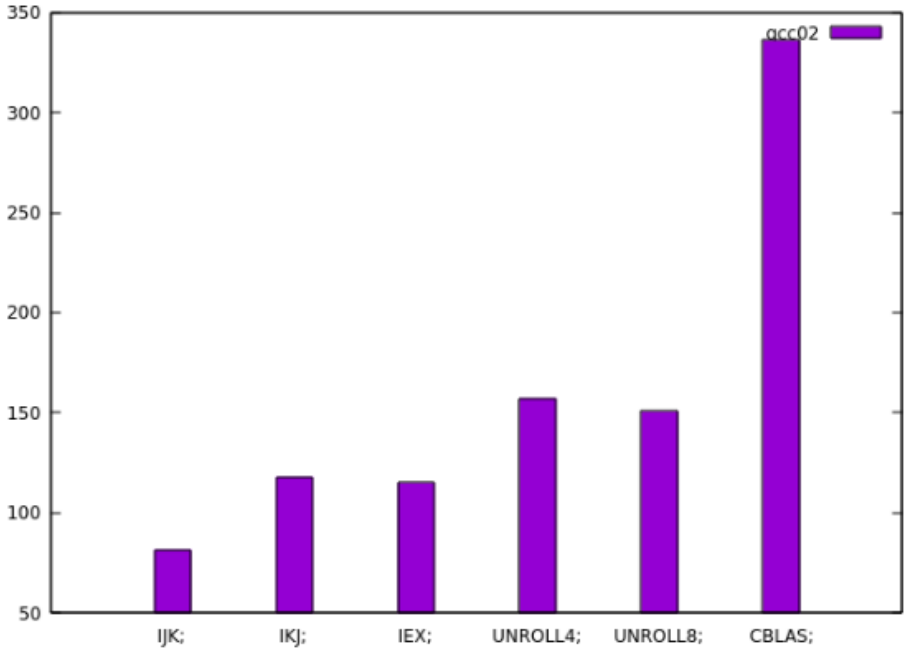
3- La génération des graphes (en histogrammes) avec Gnuplot dans un fichier **gnuplot.gp** pour la génération des histogrammes :

   a- Comparaison des versions de chaque compilateur : dans notre cas ça va être les 2 compilateurs GCC et CLANG, l'ICC et ICX ne marchait pas donc j'ai opté pour le gcc et le clang.
   GCC:
   F01 :



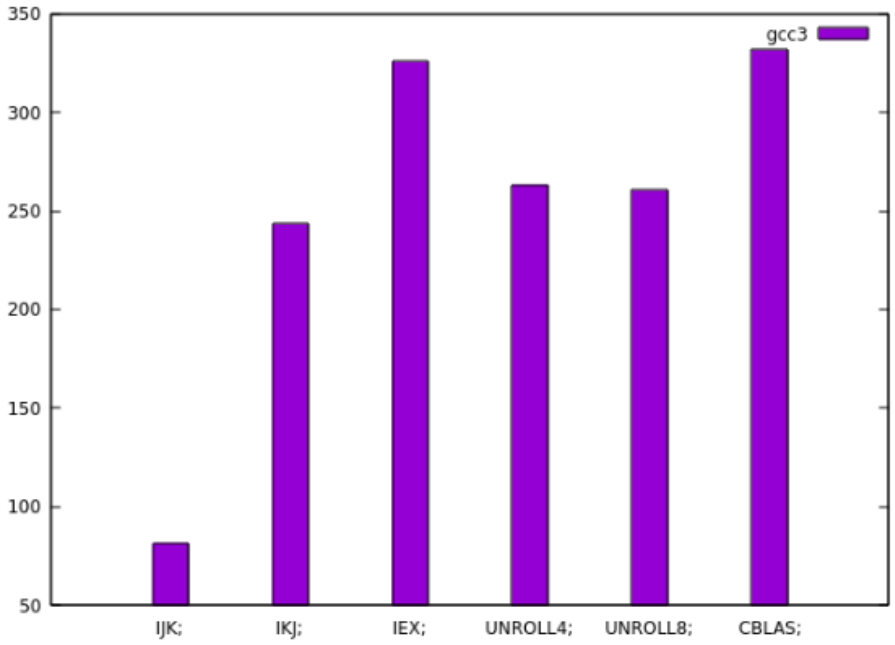
Comparaison des versions en GCC avec Flag 01

   F02 :



Comparaison des versions en GCC avec Flag 02

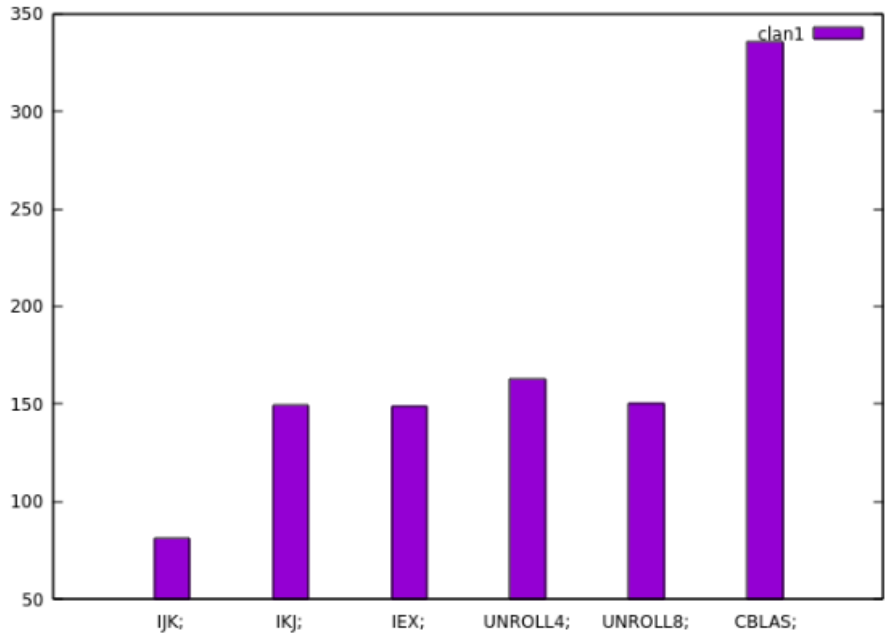
F03 :



Comparaison des versions en GCC avec Flag 03

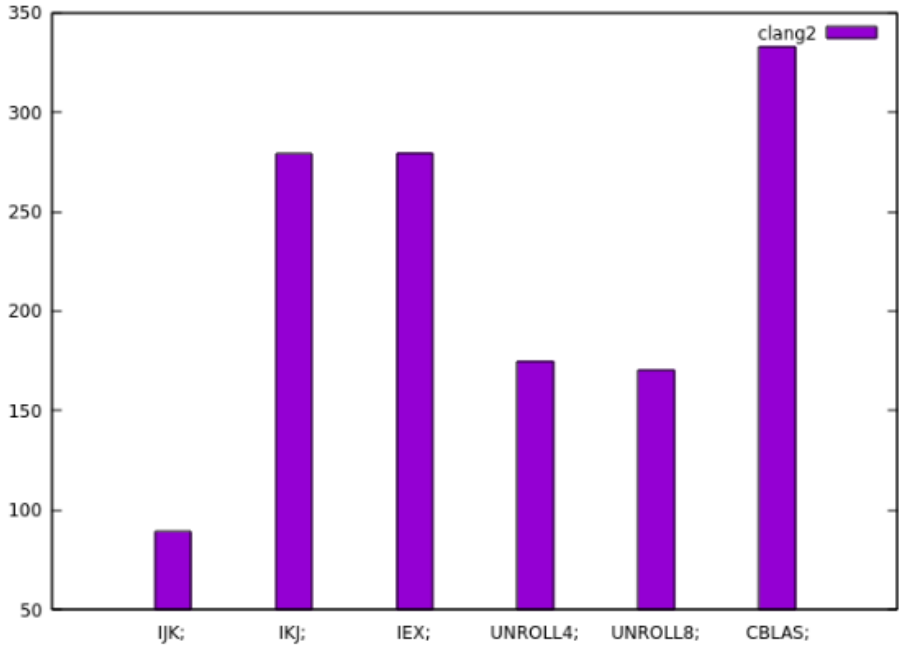
CLANG :

F01 :



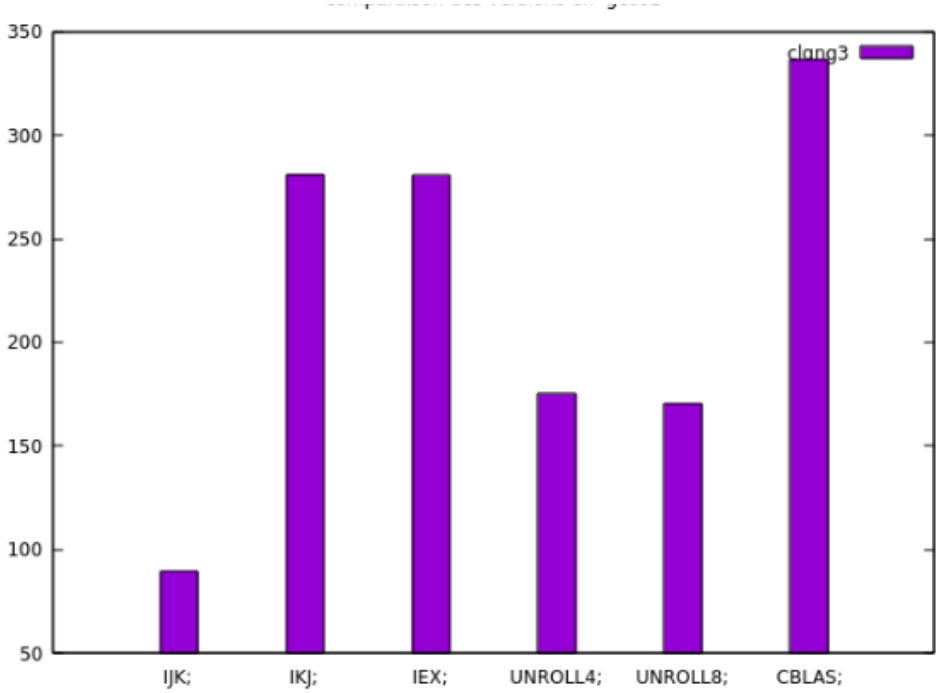
Comparaison des versions en CLANG avec Flag 01

F02 :



Comparaison des versions en CLANG avec Flag 02

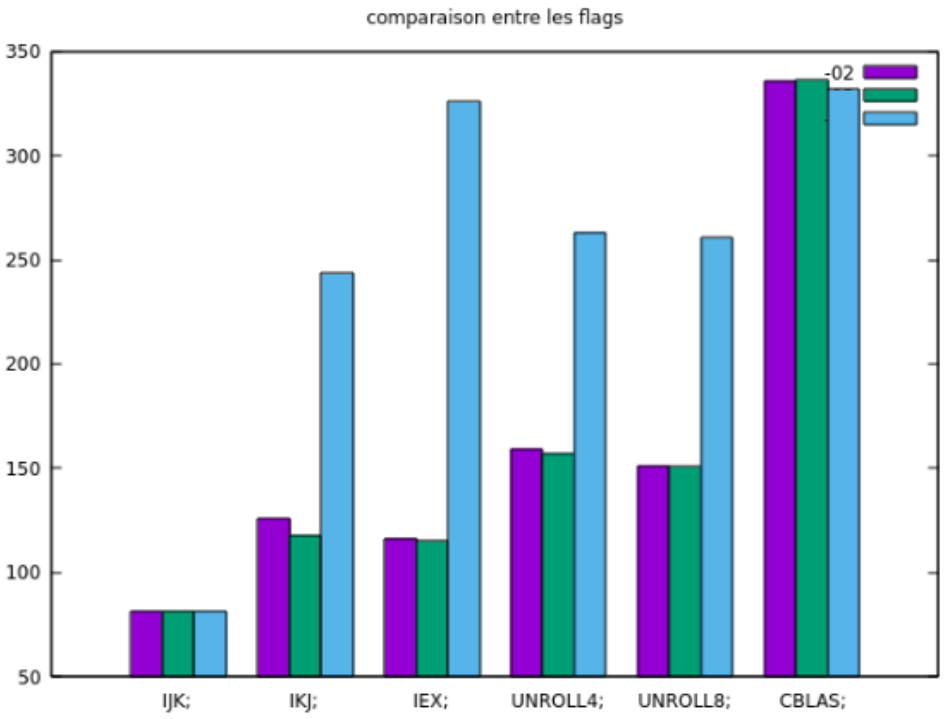
F03 :



Comparaison des versions en CLANG avec Flag 03

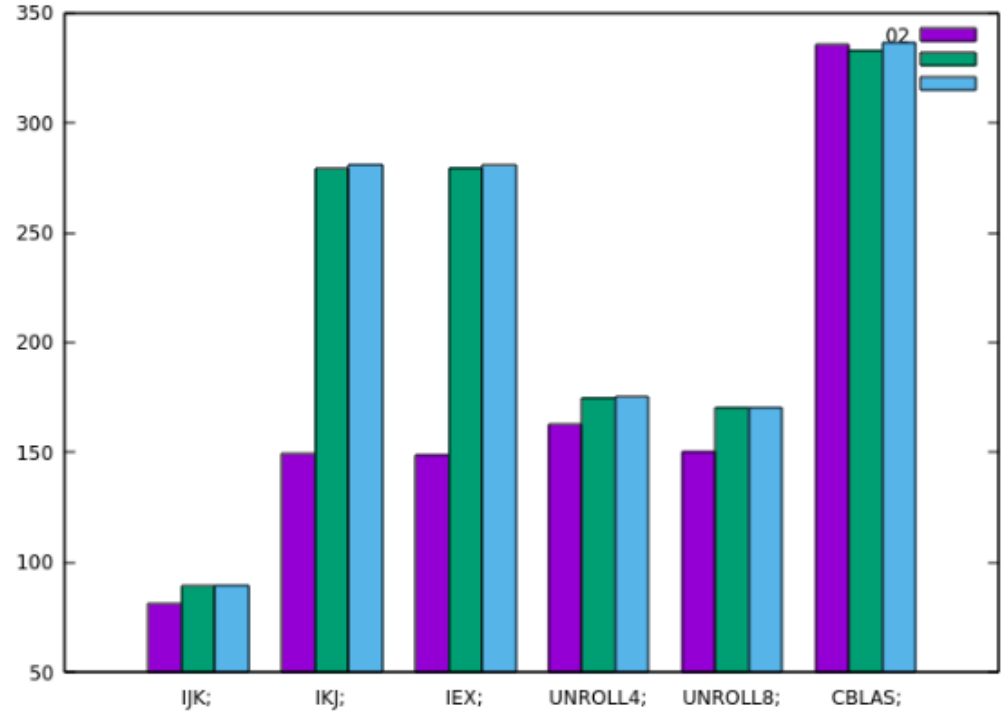
On remarque quelque soit le flag **CBLAS** est meilleur.

Maintenant on va comparer entre les versions avec les 3 flags en GCC:



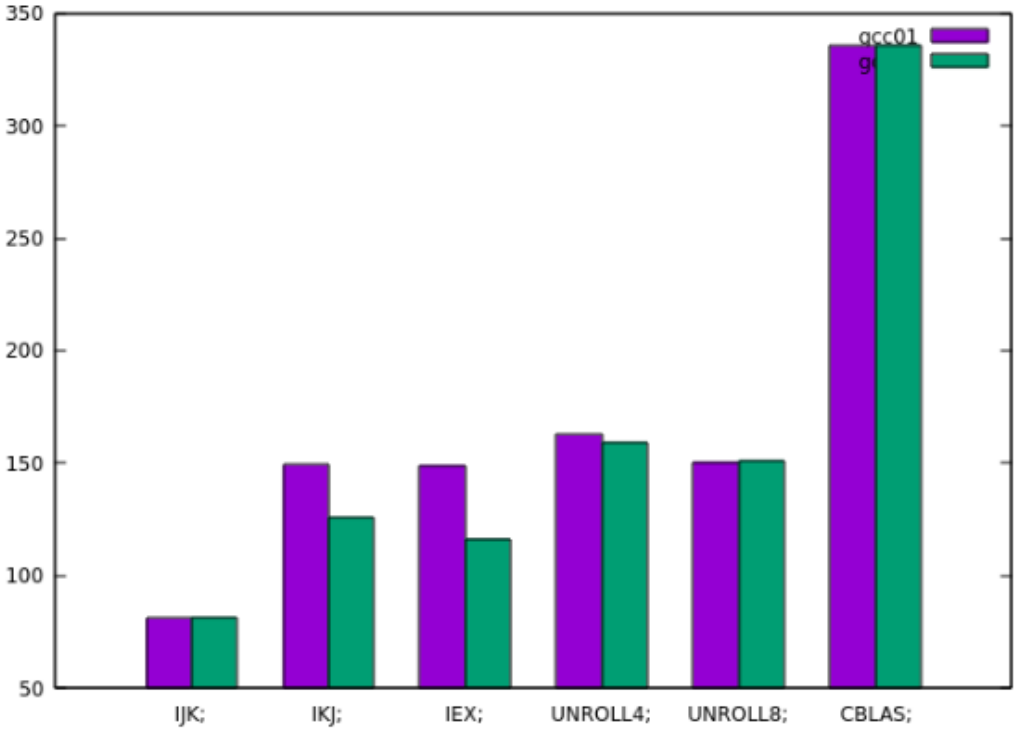
Ici le flag 03 est mieux, il donne de meilleurs résultats dans la majorité des versions.

En clang :



On voit clairement que les 2 flags 01 et 02 sont meilleurs que le flag 03.

Et enfin nous allons comparer entre clang et gcc en fixant un flag par exemple ici le flag 01 :



4- Cette fois, nous allons refaire l’exécution avec le compilateur GCC et CLANG dans les dossiers dotprod et reduc respectivement dans le **script.sh.** Vous trouverez tout les fichiers nécéssaire sur le git.