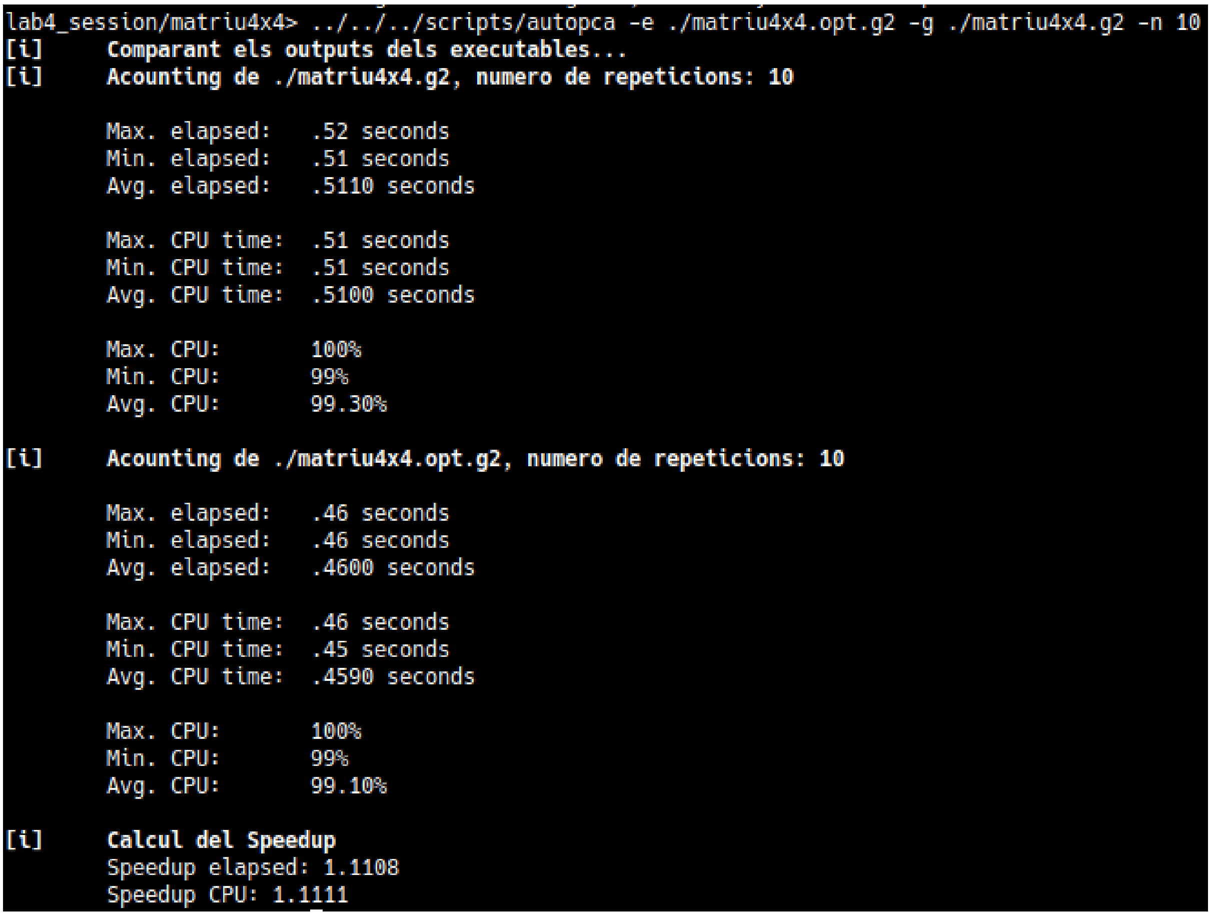
LAB4. Flow Control Optimizations

Loop unrolling and Inlining

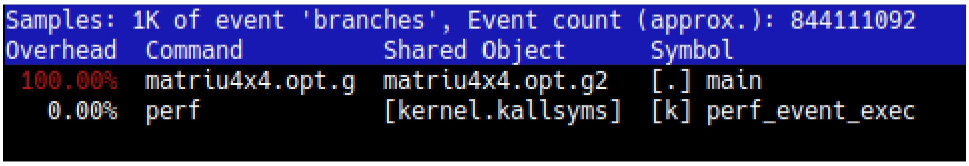
a)

i)



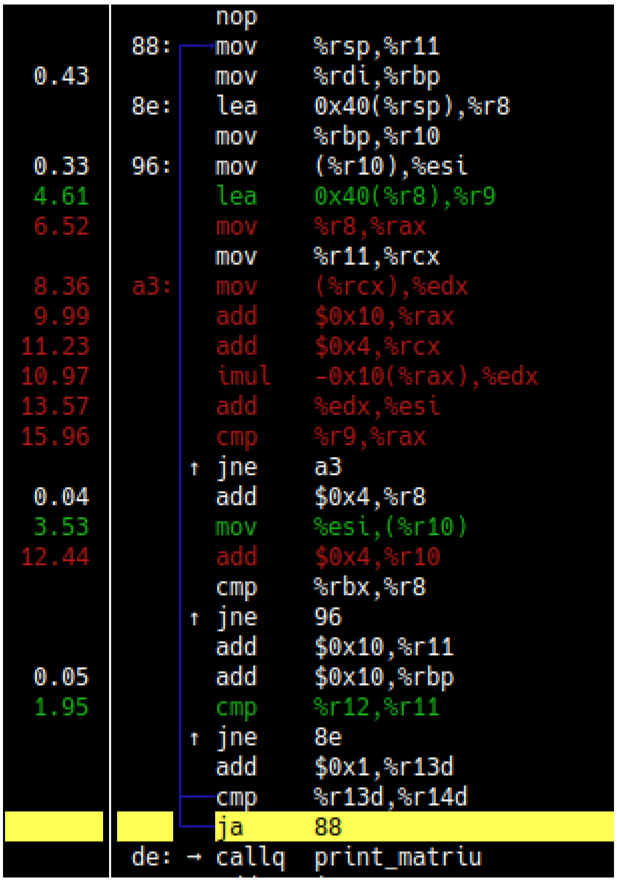
```
lab4_session/matriu4x4> ../../../scripts/autopca -e ./matriu4x4.opt.g2 -g ./matriu4x4.g2 -n 10
[i]     Comparant els outputs dels executables...
[i]     Acounting de ./matriu4x4.g2, numero de repeticions: 10

        Max. elapsed:   .52 seconds
        Min. elapsed:   .51 seconds
        Avg. elapsed:   .5110 seconds

        Max. CPU time:  .51 seconds
        Min. CPU time:  .51 seconds
        Avg. CPU time:  .5100 seconds

        Max. CPU:       100%
        Min. CPU:       99%
        Avg. CPU:       99.30%

[i]     Acounting de ./matriu4x4.opt.g2, numero de repeticions: 10

        Max. elapsed:   .46 seconds
        Min. elapsed:   .46 seconds
        Avg. elapsed:   .4600 seconds

        Max. CPU time:  .46 seconds
        Min. CPU time:  .45 seconds
        Avg. CPU time:  .4590 seconds

        Max. CPU:       100%
        Min. CPU:       99%
        Avg. CPU:       99.10%

[i]     Calcul del Speedup
        Speedup elapsed: 1.1108
        Speedup CPU: 1.1111
```

ii)



```
Samples: 1K of event 'branches', Event count (approx.): 844111092
Overhead  Command           Shared Object       Symbol
 100.00%  matriu4x4.opt.g  matriu4x4.opt.g2   [.] main
   0.00%  perf              [kernel.kallsyms]  [k] perf_event_exec
```

844M de branches



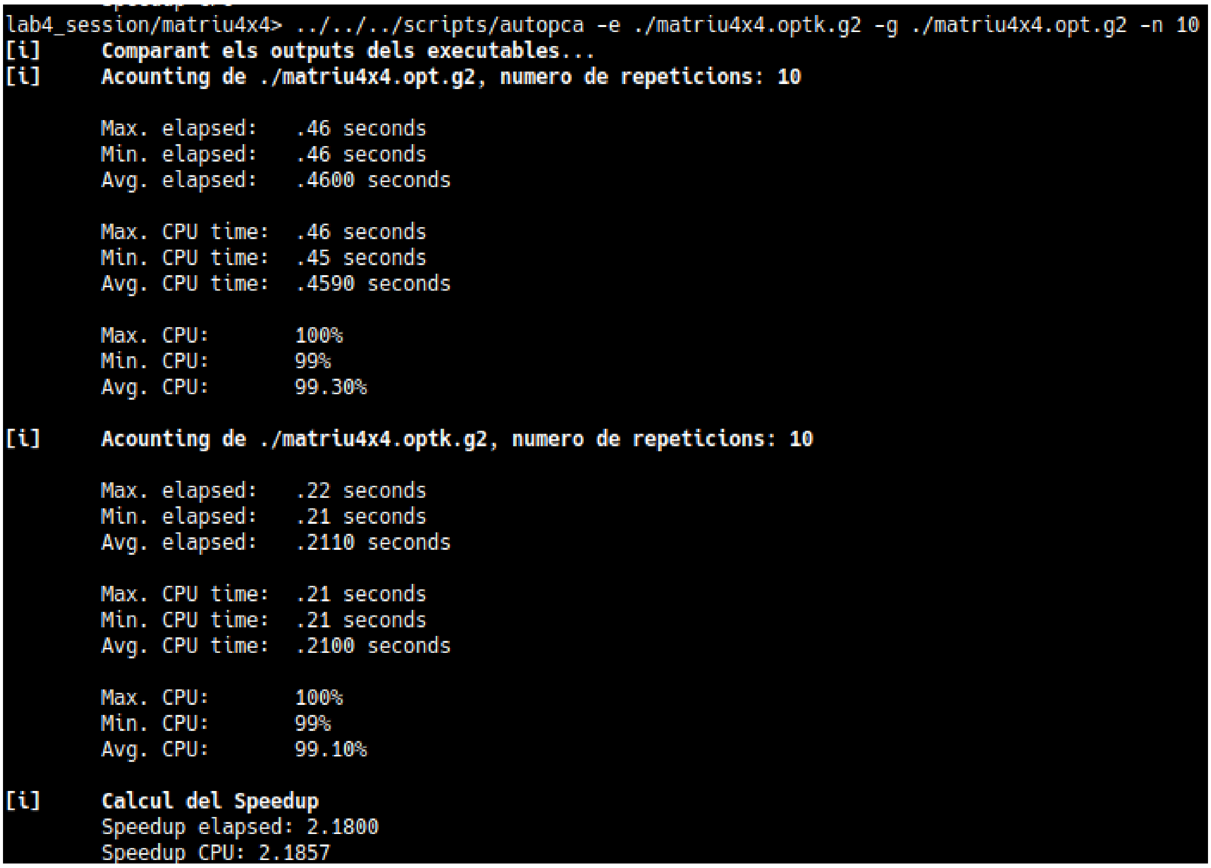
```
               nop
        88:    mov    %rsp,%r11
  0.43         mov    %rdi,%rbp
        8e:    lea    0x40(%rsp),%r8
               mov    %rbp,%r10
  0.33  96:    mov    (%r10),%esi
  4.61         lea    0x40(%r8),%r9
  6.52         mov    %r8,%rax
               mov    %r11,%rcx
  8.36  a3:    mov    (%rcx),%edx
  9.99         add    $0x10,%rax
 11.23         add    $0x4,%rcx
 10.97         imul   -0x10(%rax),%edx
 13.57         add    %edx,%esi
 15.96         cmp    %r9,%rax
             ↑ jne    a3
  0.04         add    $0x4,%r8
  3.53         mov    %esi,(%r10)
 12.44         add    $0x4,%r10
               cmp    %rbx,%r8
             ↑ jne    96
               add    $0x10,%r11
  0.05         add    $0x10,%rbp
  1.95         cmp    %r12,%r11
             ↑ jne    8e
               add    $0x1,%r13d
               cmp    %r13d,%r14d
               ja     88
        de:  → callq  print_matriu
```

iii)

```
multiplica: //(6*n_iter)*(7*4)*(7*4)*(8*4)  =  150528*n_iter
for_ITER:              // 6 ins * niter
mov     -0x8(%rsp),%r11
mov     %r15,%r12
mov     %rdi,%rbp
for_I:                 // 7 ins * 4
lea     -0x10(%r12),%r10
mov     %rsi,%rbx
for_J:                 // 7 ins * 4
mov     (%r10),%r9d
mov     %rbx,%r8
mov     %rbp,%rax
for_K:                 // 8 ins * 4
mov     (%rax),%edx
add     $0x4,%rax
add     $0x10,%r8
imul    -0x10(%r8),%edx
add     %edx,%r9d
cmp     %rax,%r11
mov     %r9d,(%r10)
jne     for_K
add     $0x4,%r10
add     $0x4,%rbx
cmp     %r10,%r12
jne     for_J
add     $0x10,%rbp
add     $0x10,%r12
add     $0x10,%r11
cmp     %r13,%rbp
jne     for_I
add     $0x1,%r14d
cmp     %ecx,%r14d
jne     for_ITER
```

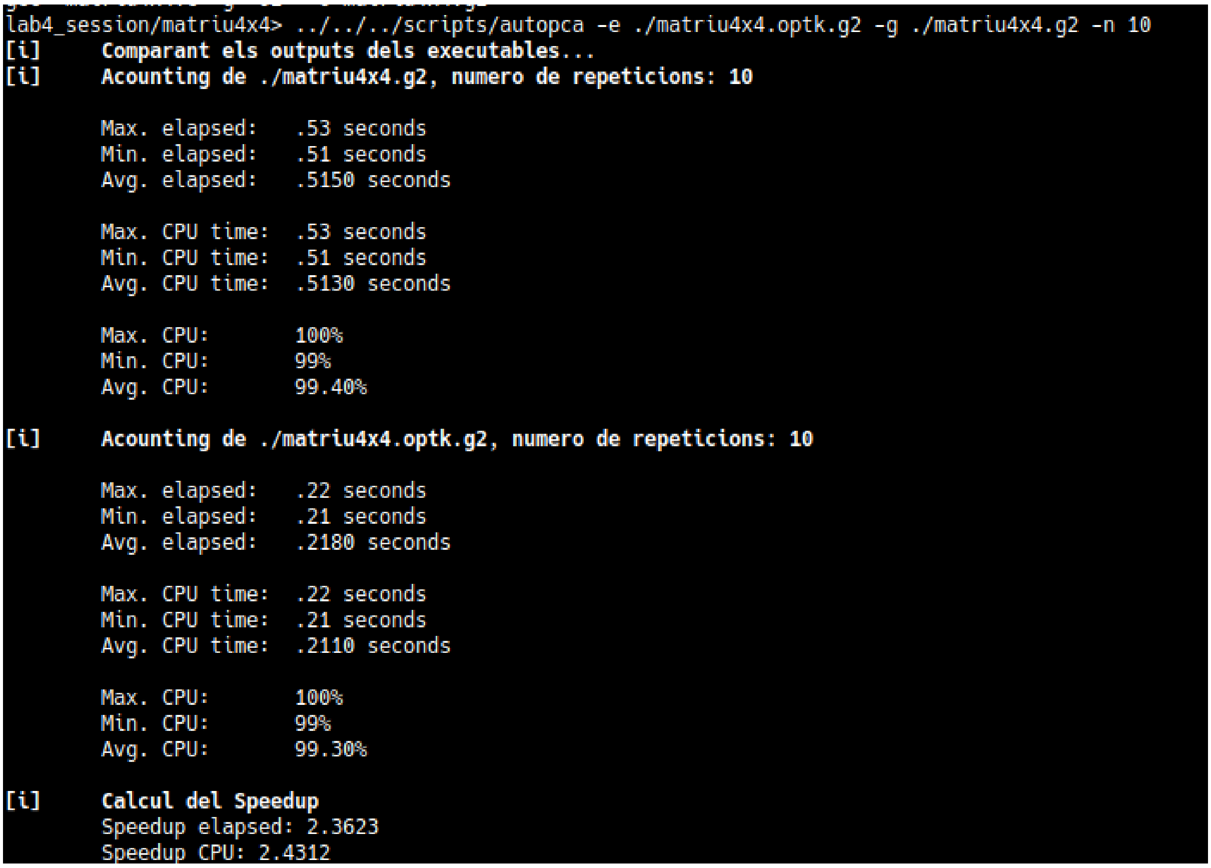
b)
i)

Timing de inlining - unrolling amb només inlining



```
lab4_session/matriu4x4> ../../../scripts/autopca -e ./matriu4x4.optk.g2 -g ./matriu4x4.opt.g2 -n 10
[i]     Comparant els outputs dels executables...
[i]     Acounting de ./matriu4x4.opt.g2, numero de repeticions: 10

        Max. elapsed:   .46 seconds
        Min. elapsed:   .46 seconds
        Avg. elapsed:   .4600 seconds

        Max. CPU time:  .46 seconds
        Min. CPU time:  .45 seconds
        Avg. CPU time:  .4590 seconds

        Max. CPU:       100%
        Min. CPU:       99%
        Avg. CPU:       99.30%

[i]     Acounting de ./matriu4x4.optk.g2, numero de repeticions: 10

        Max. elapsed:   .22 seconds
        Min. elapsed:   .21 seconds
        Avg. elapsed:   .2110 seconds

        Max. CPU time:  .21 seconds
        Min. CPU time:  .21 seconds
        Avg. CPU time:  .2100 seconds

        Max. CPU:       100%
        Min. CPU:       99%
        Avg. CPU:       99.10%

[i]     Calcul del Speedup
        Speedup elapsed: 2.1800
        Speedup CPU: 2.1857
```

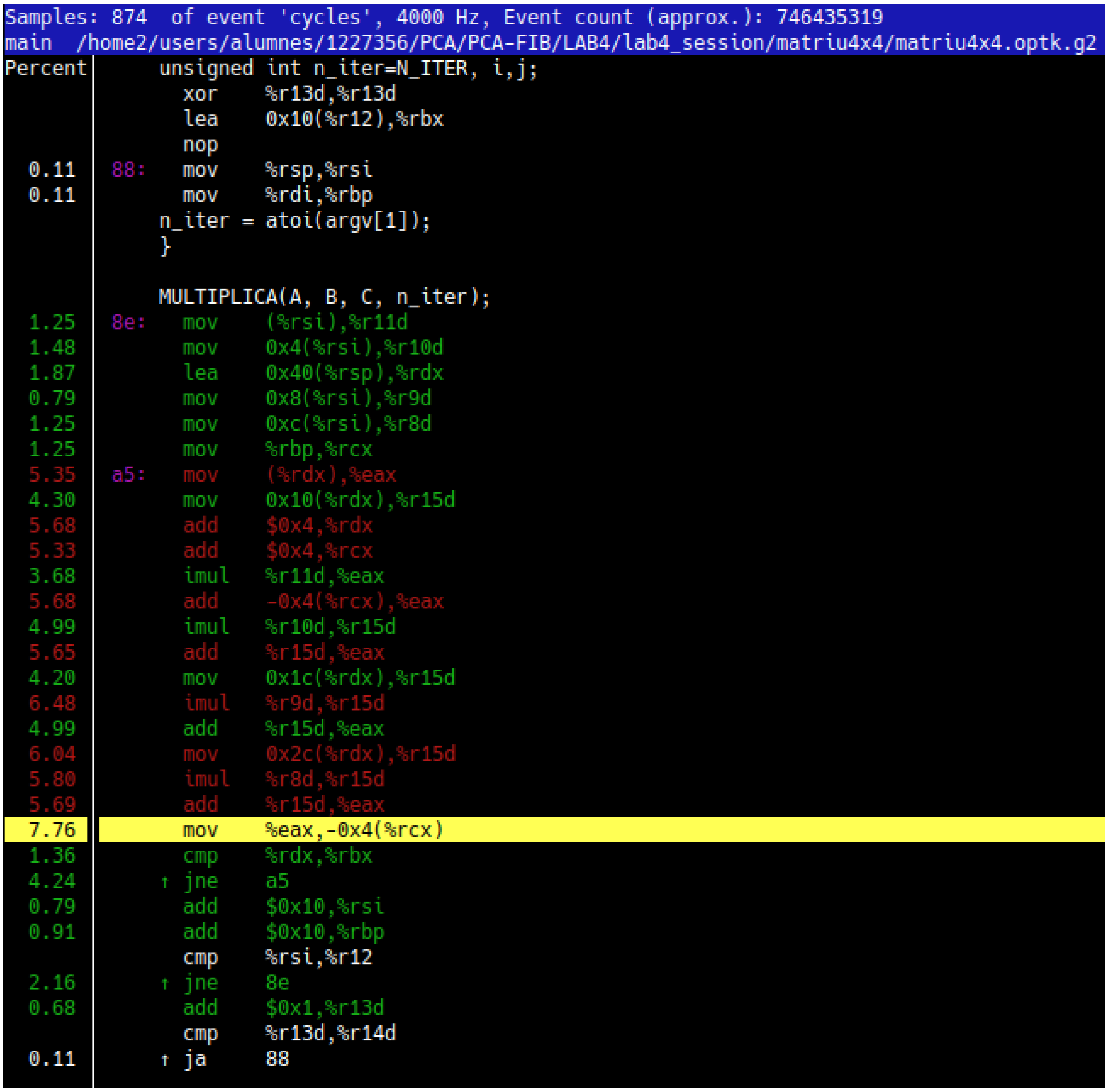
Timing amb la versió original.



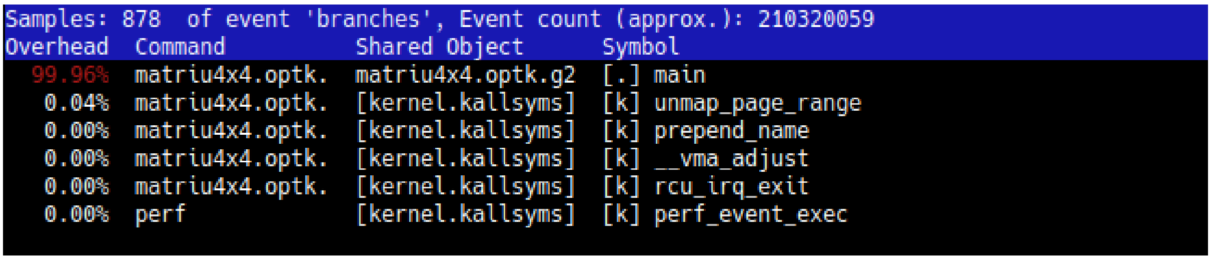
```
lab4_session/matriu4x4> ../../../scripts/autopca -e ./matriu4x4.optk.g2 -g ./matriu4x4.g2 -n 10
[i]     Comparant els outputs dels executables...
[i]     Acounting de ./matriu4x4.g2, numero de repeticions: 10

        Max. elapsed:   .53 seconds
        Min. elapsed:   .51 seconds
        Avg. elapsed:   .5150 seconds

        Max. CPU time:  .53 seconds
        Min. CPU time:  .51 seconds
        Avg. CPU time:  .5130 seconds

        Max. CPU:       100%
        Min. CPU:       99%
        Avg. CPU:       99.40%

[i]     Acounting de ./matriu4x4.optk.g2, numero de repeticions: 10

        Max. elapsed:   .22 seconds
        Min. elapsed:   .21 seconds
        Avg. elapsed:   .2180 seconds

        Max. CPU time:  .22 seconds
        Min. CPU time:  .21 seconds
        Avg. CPU time:  .2110 seconds

        Max. CPU:       100%
        Min. CPU:       99%
        Avg. CPU:       99.30%

[i]     Calcul del Speedup
        Speedup elapsed: 2.3623
        Speedup CPU: 2.4312
```

um

ii)



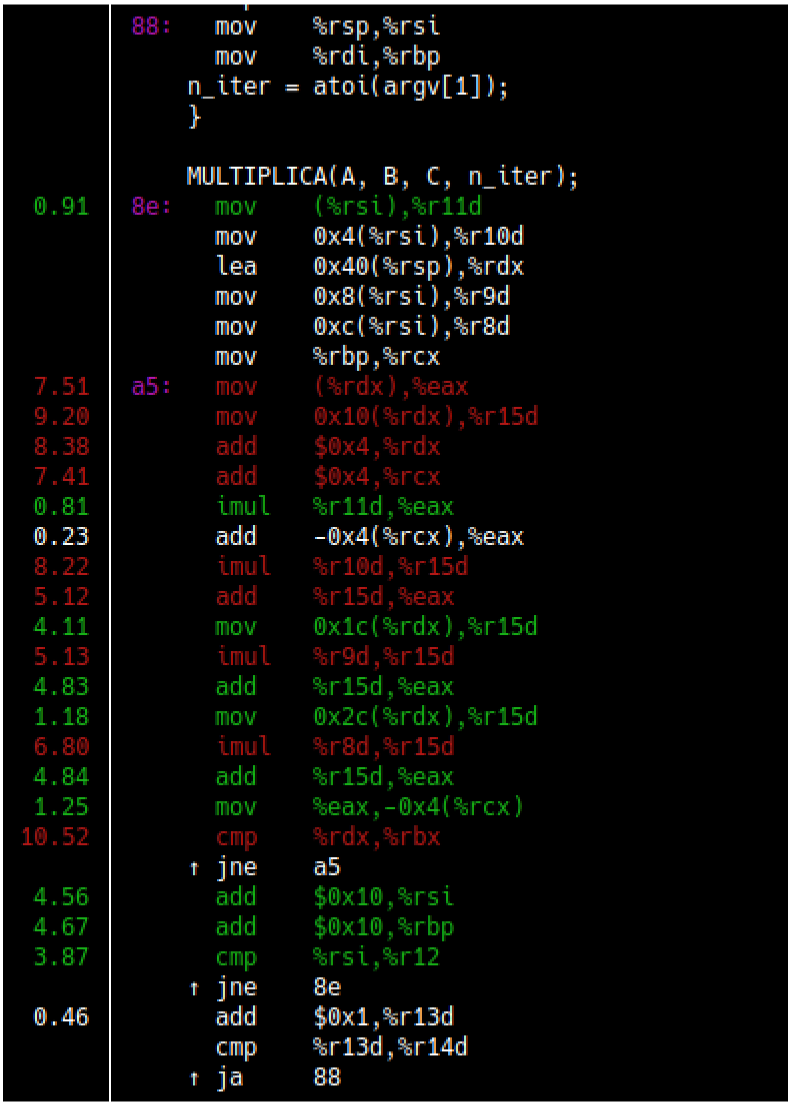
```
Samples: 874  of event 'cycles', 4000 Hz, Event count (approx.): 746435319
main  /home2/users/alumnes/1227356/PCA/PCA-FIB/LAB4/lab4_session/matriu4x4/matriu4x4.optk.g2
Percent│      unsigned int n_iter=N_ITER, i,j;
       │        xor    %r13d,%r13d
       │        lea    0x10(%r12),%rbx
       │        nop
  0.11 │ 88:    mov    %rsp,%rsi
  0.11 │        mov    %rdi,%rbp
       │      n_iter = atoi(argv[1]);
       │      }
       │
       │      MULTIPLICA(A, B, C, n_iter);
  1.25 │ 8e:    mov    (%rsi),%r11d
  1.48 │        mov    0x4(%rsi),%r10d
  1.87 │        lea    0x40(%rsp),%rdx
  0.79 │        mov    0x8(%rsi),%r9d
  1.25 │        mov    0xc(%rsi),%r8d
  1.25 │        mov    %rbp,%rcx
  5.35 │ a5:    mov    (%rdx),%eax
  4.30 │        mov    0x10(%rdx),%r15d
  5.68 │        add    $0x4,%rdx
  5.33 │        add    $0x4,%rcx
  3.68 │        imul   %r11d,%eax
  5.68 │        add    -0x4(%rcx),%eax
  4.99 │        imul   %r10d,%r15d
  5.65 │        add    %r15d,%eax
  4.20 │        mov    0x1c(%rdx),%r15d
  6.48 │        imul   %r9d,%r15d
  4.99 │        add    %r15d,%eax
  6.04 │        mov    0x2c(%rdx),%r15d
  5.80 │        imul   %r8d,%r15d
  5.69 │        add    %r15d,%eax
  7.76 │        mov    %eax,-0x4(%rcx)
  1.36 │        cmp    %rdx,%rbx
  4.24 │      ↑ jne    a5
  0.79 │        add    $0x10,%rsi
  0.91 │        add    $0x10,%rbp
       │        cmp    %rsi,%r12
  2.16 │      ↑ jne    8e
  0.68 │        add    $0x1,%r13d
       │        cmp    %r13d,%r14d
  0.11 │      ↑ ja     88
```



```
Samples: 878  of event 'branches', Event count (approx.): 210320059
Overhead  Command          Shared Object      Symbol
  99.96%  matriu4x4.optk.  matriu4x4.optk.g2  [.] main
   0.04%  matriu4x4.optk.  [kernel.kallsyms]  [k] unmap_page_range
   0.00%  matriu4x4.optk.  [kernel.kallsyms]  [k] prepend_name
   0.00%  matriu4x4.optk.  [kernel.kallsyms]  [k] __vma_adjust
   0.00%  matriu4x4.optk.  [kernel.kallsyms]  [k] rcu_irq_exit
   0.00%  perf             [kernel.kallsyms]  [k] perf_event_exec
```

210M branches → s'han reduit considerablement.

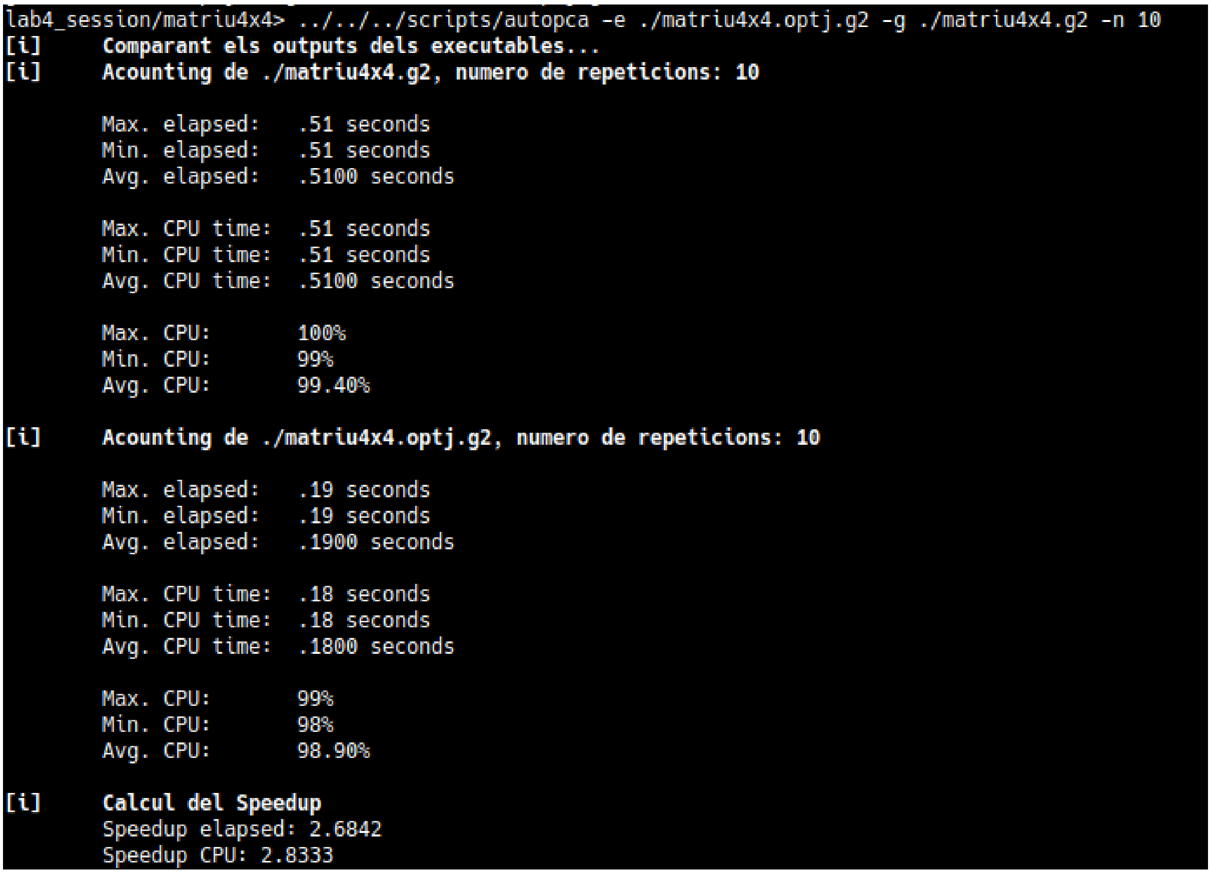
iii)



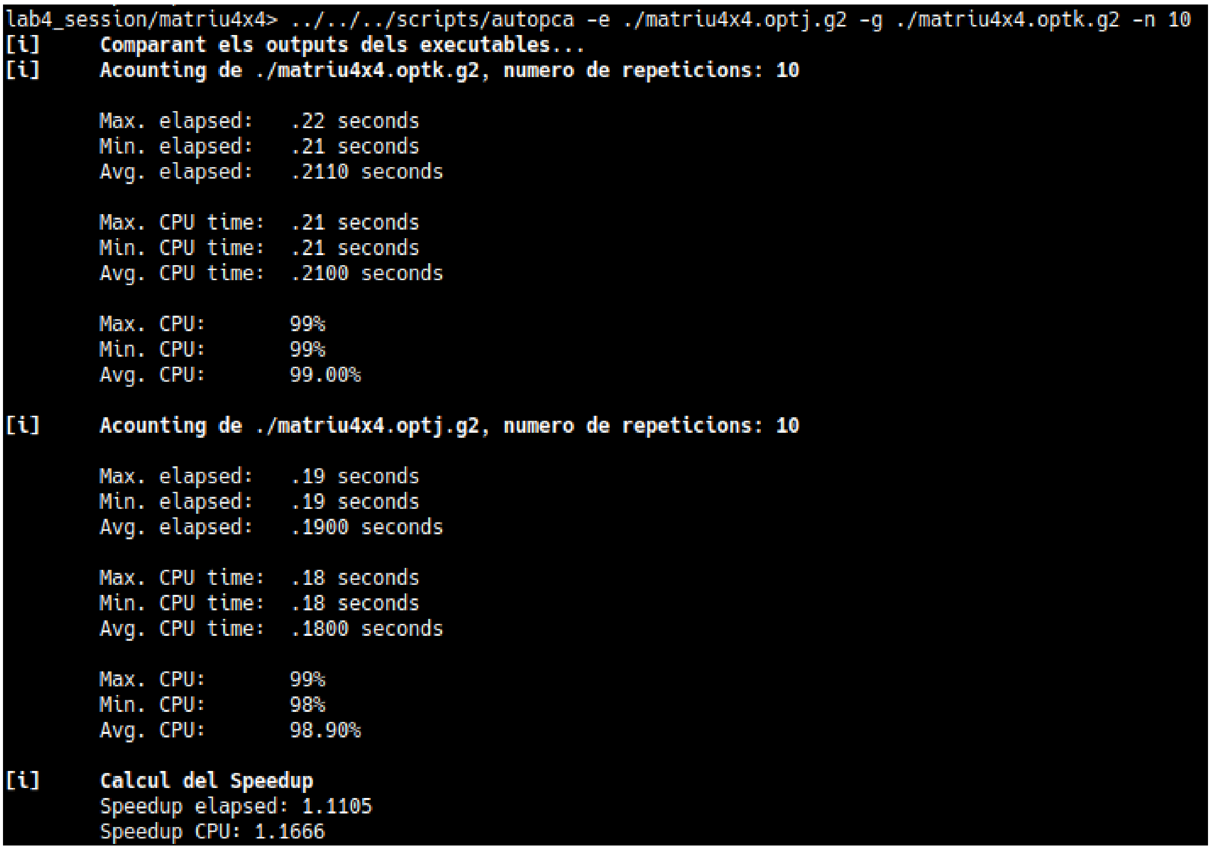
```
       │ 88:   mov    %rsp,%rsi
       │       mov    %rdi,%rbp
       │     n_iter = atoi(argv[1]);
       │     }
       │
       │     MULTIPLICA(A, B, C, n_iter);
  0.91 │ 8e:   mov    (%rsi),%r11d
       │       mov    0x4(%rsi),%r10d
       │       lea    0x40(%rsp),%rdx
       │       mov    0x8(%rsi),%r9d
       │       mov    0xc(%rsi),%r8d
       │       mov    %rbp,%rcx
  7.51 │ a5:   mov    (%rdx),%eax
  9.20 │       mov    0x10(%rdx),%r15d
  8.38 │       add    $0x4,%rdx
  7.41 │       add    $0x4,%rcx
  0.81 │       imul   %r11d,%eax
  0.23 │       add    -0x4(%rcx),%eax
  8.22 │       imul   %r10d,%r15d
  5.12 │       add    %r15d,%eax
  4.11 │       mov    0x1c(%rdx),%r15d
  5.13 │       imul   %r9d,%r15d
  4.83 │       add    %r15d,%eax
  1.18 │       mov    0x2c(%rdx),%r15d
  6.80 │       imul   %r8d,%r15d
  4.84 │       add    %r15d,%eax
  1.25 │       mov    %eax,-0x4(%rcx)
 10.52 │       cmp    %rdx,%rbx
       │     ↑ jne    a5
  4.56 │       add    $0x10,%rsi
  4.67 │       add    $0x10,%rbp
  3.87 │       cmp    %rsi,%r12
       │     ↑ jne    8e
  0.46 │       add    $0x1,%r13d
       │       cmp    %r13d,%r14d
       │     ↑ ja     88
```

17*4 *10*4*5*N_iter = **13600 * N_iter**

c)
i)



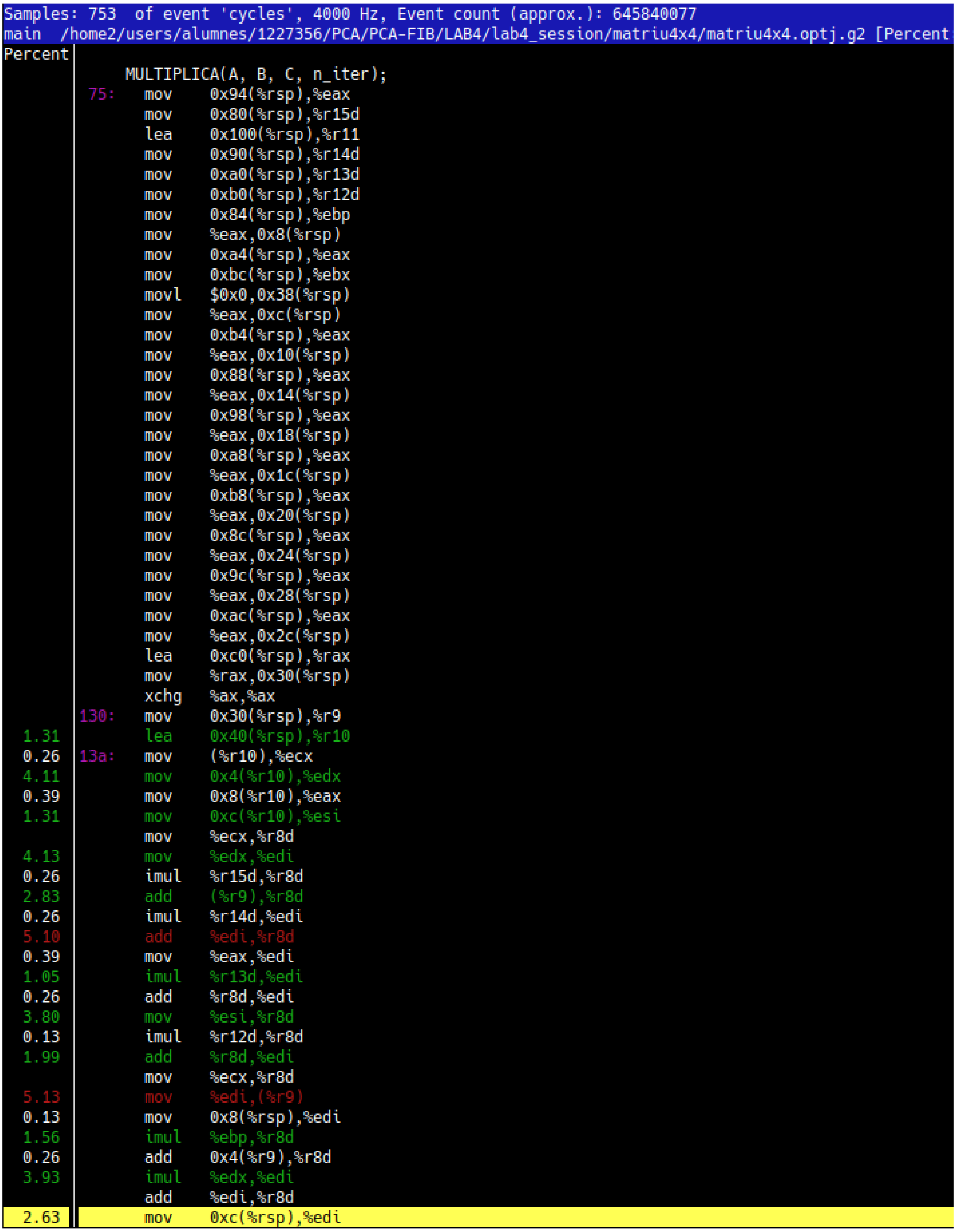
```
lab4_session/matriu4x4> ../../../scripts/autopca -e ./matriu4x4.optj.g2 -g ./matriu4x4.g2 -n 10
[i]     Comparant els outputs dels executables...
[i]     Acounting de ./matriu4x4.g2, numero de repeticions: 10

        Max. elapsed:   .51 seconds
        Min. elapsed:   .51 seconds
        Avg. elapsed:   .5100 seconds

        Max. CPU time:  .51 seconds
        Min. CPU time:  .51 seconds
        Avg. CPU time:  .5100 seconds

        Max. CPU:       100%
        Min. CPU:       99%
        Avg. CPU:       99.40%

[i]     Acounting de ./matriu4x4.optj.g2, numero de repeticions: 10

        Max. elapsed:   .19 seconds
        Min. elapsed:   .19 seconds
        Avg. elapsed:   .1900 seconds

        Max. CPU time:  .18 seconds
        Min. CPU time:  .18 seconds
        Avg. CPU time:  .1800 seconds

        Max. CPU:       99%
        Min. CPU:       98%
        Avg. CPU:       98.90%

[i]     Calcul del Speedup
        Speedup elapsed: 2.6842
        Speedup CPU: 2.8333
```



```
lab4_session/matriu4x4> ../../../scripts/autopca -e ./matriu4x4.optj.g2 -g ./matriu4x4.optk.g2 -n 10
[i]     Comparant els outputs dels executables...
[i]     Acounting de ./matriu4x4.optk.g2, numero de repeticions: 10

        Max. elapsed:   .22 seconds
        Min. elapsed:   .21 seconds
        Avg. elapsed:   .2110 seconds

        Max. CPU time:  .21 seconds
        Min. CPU time:  .21 seconds
        Avg. CPU time:  .2100 seconds

        Max. CPU:       99%
        Min. CPU:       99%
        Avg. CPU:       99.00%

[i]     Acounting de ./matriu4x4.optj.g2, numero de repeticions: 10

        Max. elapsed:   .19 seconds
        Min. elapsed:   .19 seconds
        Avg. elapsed:   .1900 seconds

        Max. CPU time:  .18 seconds
        Min. CPU time:  .18 seconds
        Avg. CPU time:  .1800 seconds

        Max. CPU:       99%
        Min. CPU:       98%
        Avg. CPU:       98.90%

[i]     Calcul del Speedup
        Speedup elapsed: 1.1105
        Speedup CPU: 1.1666
```

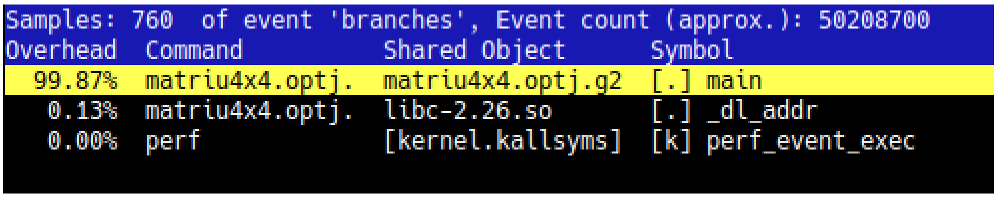
ii)

Cicles:



```
Samples: 753  of event 'cycles', 4000 Hz, Event count (approx.): 645840077
main  /home2/users/alumnes/1227356/PCA/PCA-FIB/LAB4/lab4_session/matriu4x4/matriu4x4.optj.g2 [Percent
Percent│
       │    MULTIPLICA(A, B, C, n_iter);
       │ 75:  mov    0x94(%rsp),%eax
       │      mov    0x80(%rsp),%r15d
       │      lea    0x100(%rsp),%r11
       │      mov    0x90(%rsp),%r14d
       │      mov    0xa0(%rsp),%r13d
       │      mov    0xb0(%rsp),%r12d
       │      mov    0x84(%rsp),%ebp
       │      mov    %eax,0x8(%rsp)
       │      mov    0xa4(%rsp),%eax
       │      mov    0xbc(%rsp),%ebx
       │      movl   $0x0,0x38(%rsp)
       │      mov    %eax,0xc(%rsp)
       │      mov    0xb4(%rsp),%eax
       │      mov    %eax,0x10(%rsp)
       │      mov    0x88(%rsp),%eax
       │      mov    %eax,0x14(%rsp)
       │      mov    0x98(%rsp),%eax
       │      mov    %eax,0x18(%rsp)
       │      mov    0xa8(%rsp),%eax
       │      mov    %eax,0x1c(%rsp)
       │      mov    0xb8(%rsp),%eax
       │      mov    %eax,0x20(%rsp)
       │      mov    0x8c(%rsp),%eax
       │      mov    %eax,0x24(%rsp)
       │      mov    0x9c(%rsp),%eax
       │      mov    %eax,0x28(%rsp)
       │      mov    0xac(%rsp),%eax
       │      mov    %eax,0x2c(%rsp)
       │      lea    0xc0(%rsp),%rax
       │      mov    %rax,0x30(%rsp)
       │      xchg   %ax,%ax
       │130:  mov    0x30(%rsp),%r9
  1.31 │      lea    0x40(%rsp),%r10
  0.26 │13a:  mov    (%r10),%ecx
  4.11 │      mov    0x4(%r10),%edx
  0.39 │      mov    0x8(%r10),%eax
  1.31 │      mov    0xc(%r10),%esi
       │      mov    %ecx,%r8d
  4.13 │      mov    %edx,%edi
  0.26 │      imul   %r15d,%r8d
  2.83 │      add    (%r9),%r8d
  0.26 │      imul   %r14d,%edi
  5.10 │      add    %edi,%r8d
  0.39 │      mov    %eax,%edi
  1.05 │      imul   %r13d,%edi
  0.26 │      add    %r8d,%edi
  3.80 │      mov    %esi,%r8d
  0.13 │      imul   %r12d,%r8d
  1.99 │      add    %r8d,%edi
       │      mov    %ecx,%r8d
  5.13 │      mov    %edi,(%r9)
  0.13 │      mov    0x8(%rsp),%edi
  1.56 │      imul   %ebp,%r8d
  0.26 │      add    0x4(%r9),%r8d
  3.93 │      imul   %edx,%edi
       │      add    %edi,%r8d
  2.63 │      mov    0xc(%rsp),%edi
```

Branches:



```
Samples: 760  of event 'branches', Event count (approx.): 50208700
Overhead  Command           Shared Object        Symbol
  99.87%  matriu4x4.optj.   matriu4x4.optj.g2    [.] main
   0.13%  matriu4x4.optj.   libc-2.26.so         [.] _dl_addr
   0.00%  perf              [kernel.kallsyms]    [k] perf_event_exec
```

50M de branches

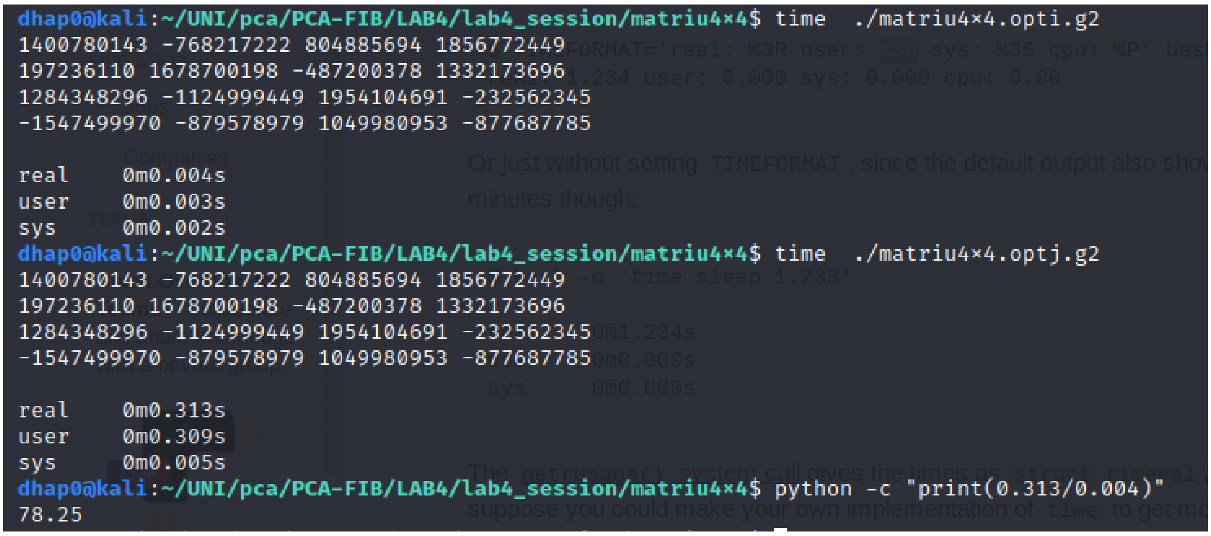
iii)

33 (tots els moves de l'inici) + 57*4 * 6 * N_iter = 33 + 1368 * N_iter

d)

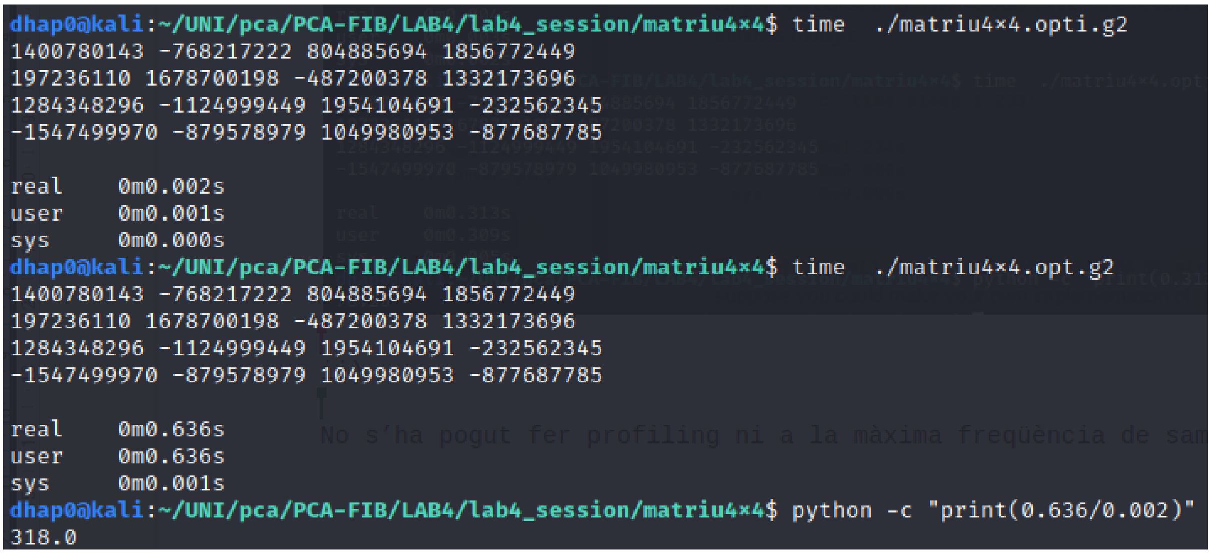
i) No s’ha pogut fer el timing amb el GNU time ja que aquest només permet precisió fins a les centèsimes de segon. Hem usat el time de bash que te precisió fins als mil·lisegons per a poder calcular el speedup.

Speedup respecte la versió anterior:



```
dhap0@kali:~/UNI/pca/PCA-FIB/LAB4/lab4_session/matriu4x4$ time  ./matriu4x4.opti.g2
1400780143 -768217222 804885694 1856772449
197236110 1678700198 -487200378 1332173696
1284348296 -1124999449 1954104691 -232562345
-1547499970 -879578979 1049980953 -877687785

real	0m0.004s
user	0m0.003s
sys	0m0.002s
dhap0@kali:~/UNI/pca/PCA-FIB/LAB4/lab4_session/matriu4x4$ time  ./matriu4x4.optj.g2
1400780143 -768217222 804885694 1856772449
197236110 1678700198 -487200378 1332173696
1284348296 -1124999449 1954104691 -232562345
-1547499970 -879578979 1049980953 -877687785

real	0m0.313s
user	0m0.309s
sys	0m0.005s
dhap0@kali:~/UNI/pca/PCA-FIB/LAB4/lab4_session/matriu4x4$ python -c "print(0.313/0.004)"
78.25
```

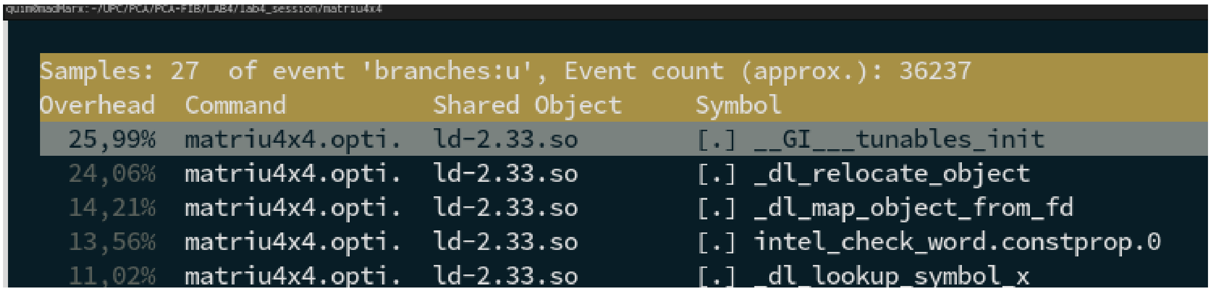
Speedup respecte la versió original:



```
dhap0@kali:~/UNI/pca/PCA-FIB/LAB4/lab4_session/matriu4x4$ time  ./matriu4x4.opti.g2
1400780143 -768217222 804885694 1856772449
197236110 1678700198 -487200378 1332173696
1284348296 -1124999449 1954104691 -232562345
-1547499970 -879578979 1049980953 -877687785

real	0m0.002s
user	0m0.001s
sys	0m0.000s
dhap0@kali:~/UNI/pca/PCA-FIB/LAB4/lab4_session/matriu4x4$ time  ./matriu4x4.opt.g2
1400780143 -768217222 804885694 1856772449
197236110 1678700198 -487200378 1332173696
1284348296 -1124999449 1954104691 -232562345
-1547499970 -879578979 1049980953 -877687785

real	0m0.636s
user	0m0.636s
sys	0m0.001s
dhap0@kali:~/UNI/pca/PCA-FIB/LAB4/lab4_session/matriu4x4$ python -c "print(0.636/0.002)"
318.0
```

ii)

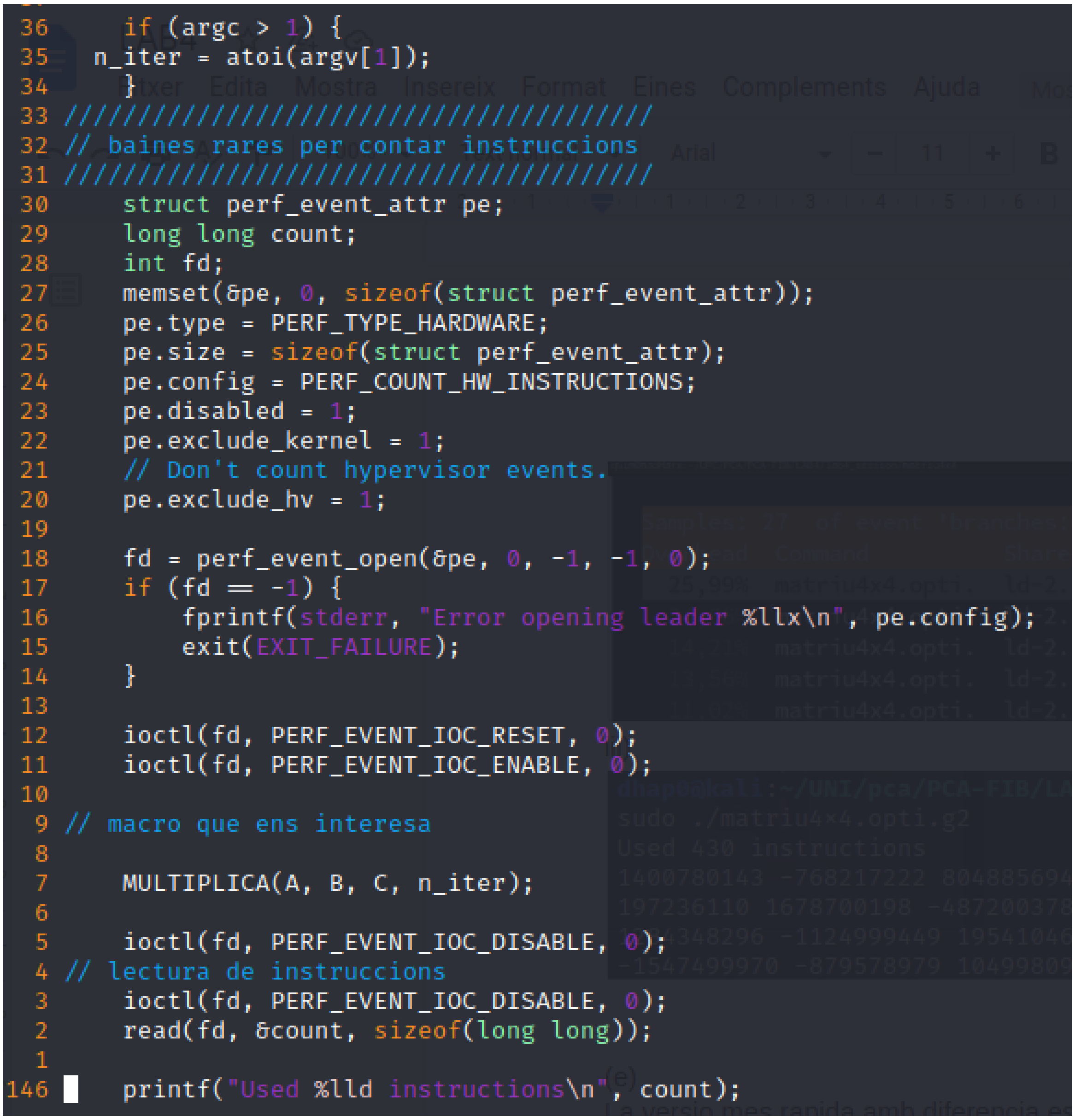
Hem reduït considerablement el nombre de branches de 50 milions a 36.237.



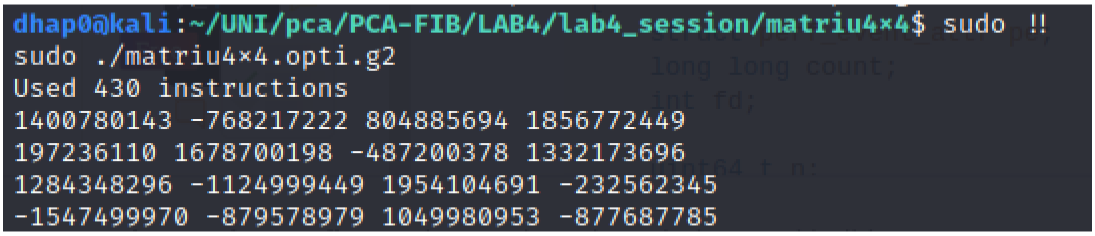
```
quim@madMarx:~/UPC/PCA/PCA-FIB/LAB4/lab4_session/matriu4x4

Samples: 27  of event 'branches:u', Event count (approx.): 36237
Overhead  Command          Shared Object      Symbol
  25,99%  matriu4x4.opti.  ld-2.33.so         [.] __GI___tunables_init
  24,06%  matriu4x4.opti.  ld-2.33.so         [.] _dl_relocate_object
  14,21%  matriu4x4.opti.  ld-2.33.so         [.] _dl_map_object_from_fd
  13,56%  matriu4x4.opti.  ld-2.33.so         [.] intel_check_word.constprop.0
  11,02%  matriu4x4.opti.  ld-2.33.so         [.] _dl_lookup_symbol_x
```

iii)

gracies a:
https://stackoverflow.com/questions/13313510/quick-way-to-count-number-of-instructions-executed-in-a-c-program



```c
    if (argc > 1) {
  n_iter = atoi(argv[1]);
    }
////////////////////////////////////////
// baines rares per contar instruccions
////////////////////////////////////////
    struct perf_event_attr pe;
    long long count;
    int fd;
    memset(&pe, 0, sizeof(struct perf_event_attr));
    pe.type = PERF_TYPE_HARDWARE;
    pe.size = sizeof(struct perf_event_attr);
    pe.config = PERF_COUNT_HW_INSTRUCTIONS;
    pe.disabled = 1;
    pe.exclude_kernel = 1;
    // Don't count hypervisor events.
    pe.exclude_hv = 1;

    fd = perf_event_open(&pe, 0, -1, -1, 0);
    if (fd == -1) {
        fprintf(stderr, "Error opening leader %llx\n", pe.config);
        exit(EXIT_FAILURE);
    }

    ioctl(fd, PERF_EVENT_IOC_RESET, 0);
    ioctl(fd, PERF_EVENT_IOC_ENABLE, 0);

// macro que ens interesa

    MULTIPLICA(A, B, C, n_iter);

    ioctl(fd, PERF_EVENT_IOC_DISABLE, 0);
// lectura de instruccions
    ioctl(fd, PERF_EVENT_IOC_DISABLE, 0);
    read(fd, &count, sizeof(long long));

    printf("Used %lld instructions\n", count);
```



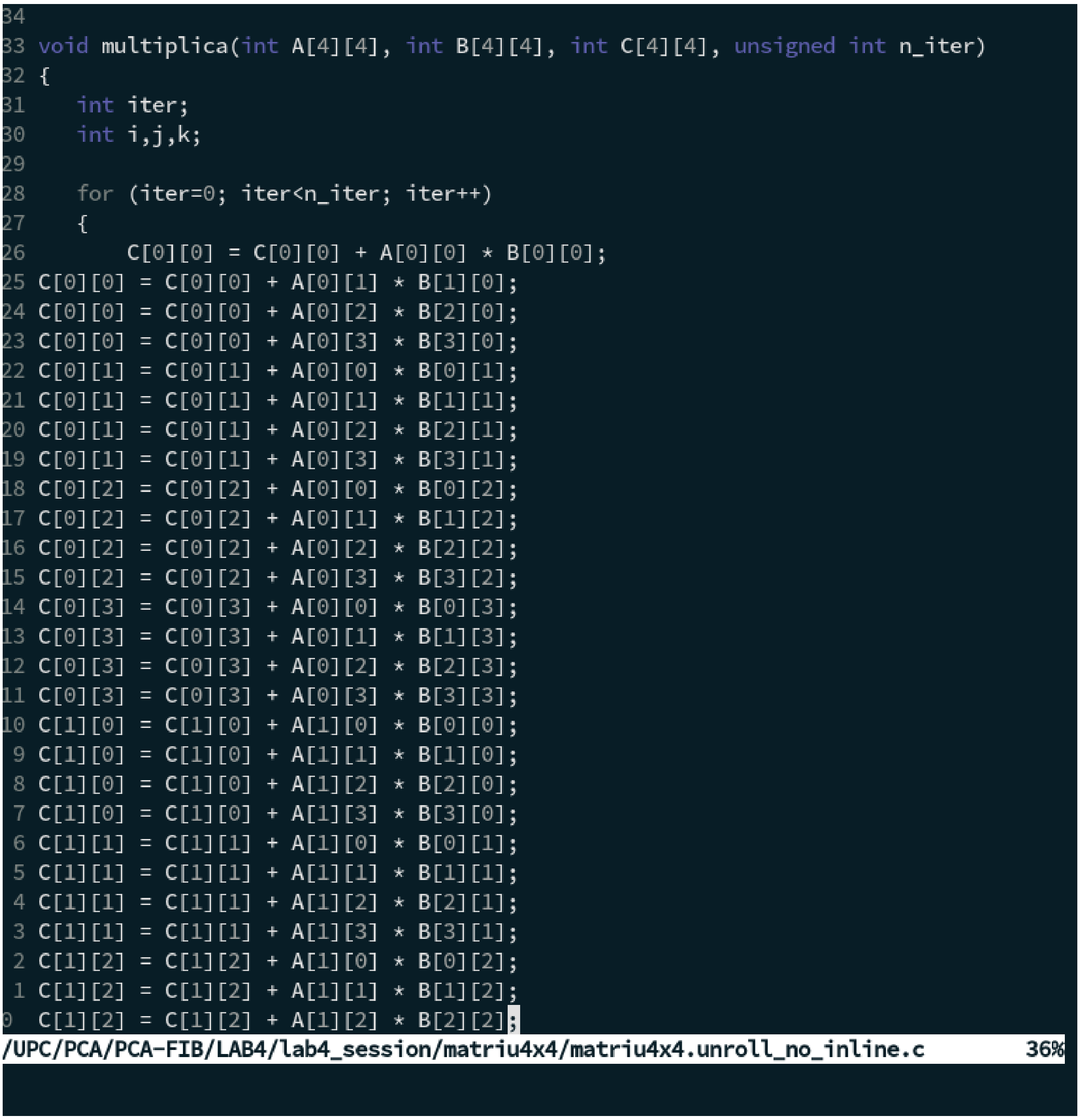
```
dhap0@kali:~/UNI/pca/PCA-FIB/LAB4/lab4_session/matriu4×4$ sudo !!
sudo ./matriu4×4.opti.g2
Used 430 instructions
1400780143 -768217222 804885694 1856772449
197236110 1678700198 -487200378 1332173696
1284348296 -1124999449 1954104691 -232562345
-1547499970 -879578979 1049980953 -877687785
```

hem passat de les (33 + 1368 * N_iter) instruccions a nomes 430

(e)

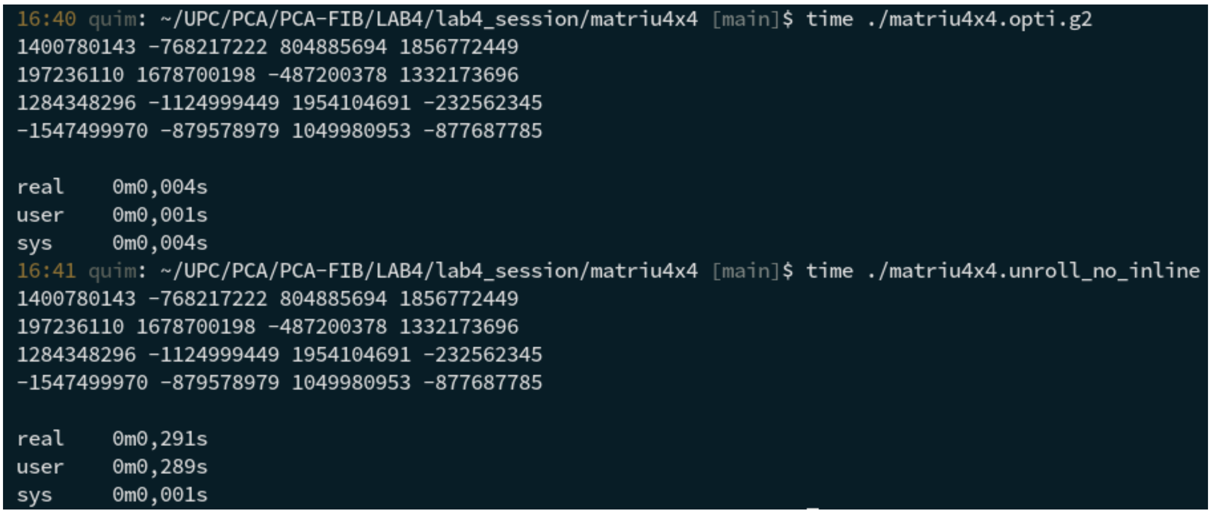
La versió més ràpida amb diferència és la del full-unroll, ja que obte un speedup de 318 respecte l’original i passa dels 844 milions de salts que tenia la primera versió del codi a nomes 36.237 salts.

(f)



```

void multiplica(int A[4][4], int B[4][4], int C[4][4], unsigned int n_iter)
{
   int iter;
   int i,j,k;

   for (iter=0; iter<n_iter; iter++)
   {
       C[0][0] = C[0][0] + A[0][0] * B[0][0];
C[0][0] = C[0][0] + A[0][1] * B[1][0];
C[0][0] = C[0][0] + A[0][2] * B[2][0];
C[0][0] = C[0][0] + A[0][3] * B[3][0];
C[0][1] = C[0][1] + A[0][0] * B[0][1];
C[0][1] = C[0][1] + A[0][1] * B[1][1];
C[0][1] = C[0][1] + A[0][2] * B[2][1];
C[0][1] = C[0][1] + A[0][3] * B[3][1];
C[0][2] = C[0][2] + A[0][0] * B[0][2];
C[0][2] = C[0][2] + A[0][1] * B[1][2];
C[0][2] = C[0][2] + A[0][2] * B[2][2];
C[0][2] = C[0][2] + A[0][3] * B[3][2];
C[0][3] = C[0][3] + A[0][0] * B[0][3];
C[0][3] = C[0][3] + A[0][1] * B[1][3];
C[0][3] = C[0][3] + A[0][2] * B[2][3];
C[0][3] = C[0][3] + A[0][3] * B[3][3];
C[1][0] = C[1][0] + A[1][0] * B[0][0];
C[1][0] = C[1][0] + A[1][1] * B[1][0];
C[1][0] = C[1][0] + A[1][2] * B[2][0];
C[1][0] = C[1][0] + A[1][3] * B[3][0];
C[1][1] = C[1][1] + A[1][0] * B[0][1];
C[1][1] = C[1][1] + A[1][1] * B[1][1];
C[1][1] = C[1][1] + A[1][2] * B[2][1];
C[1][1] = C[1][1] + A[1][3] * B[3][1];
C[1][2] = C[1][2] + A[1][0] * B[0][2];
C[1][2] = C[1][2] + A[1][1] * B[1][2];
C[1][2] = C[1][2] + A[1][2] * B[2][2];
/UPC/PCA/PCA-FIB/LAB4/lab4_session/matriu4x4/matriu4x4.unroll_no_inline.c        36%
```

Com podem veure a la captura següent hi ha un canvi considerable entre els 0,004s que triga la versió amb inlining i els 0,291 que es triguen sense inlining:



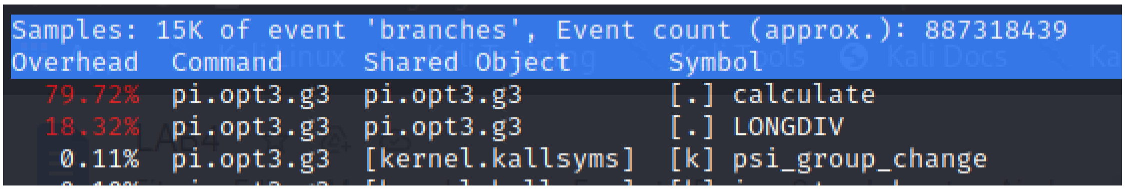
```
16:40 quim: ~/UPC/PCA/PCA-FIB/LAB4/lab4_session/matriu4x4 [main]$ time ./matriu4x4.opti.g2
1400780143 -768217222 804885694 1856772449
197236110 1678700198 -487200378 1332173696
1284348296 -1124999449 1954104691 -232562345
-1547499970 -879578979 1049980953 -877687785

real    0m0,004s
user    0m0,001s
sys     0m0,004s
16:41 quim: ~/UPC/PCA/PCA-FIB/LAB4/lab4_session/matriu4x4 [main]$ time ./matriu4x4.unroll_no_inline
1400780143 -768217222 804885694 1856772449
197236110 1678700198 -487200378 1332173696
1284348296 -1124999449 1954104691 -232562345
-1547499970 -879578979 1049980953 -877687785

real    0m0,291s
user    0m0,289s
sys     0m0,001s
```

Això és degut a que el compilador no pot preveure els paràmetres que es passen a la funció quan no es fa inlining i per tant no pot aplicar certes optimitzacions en el codi de la funció que si que podria fer en el cas de que es fes inlining.

Optimizacions de Pi.c

Unrolling

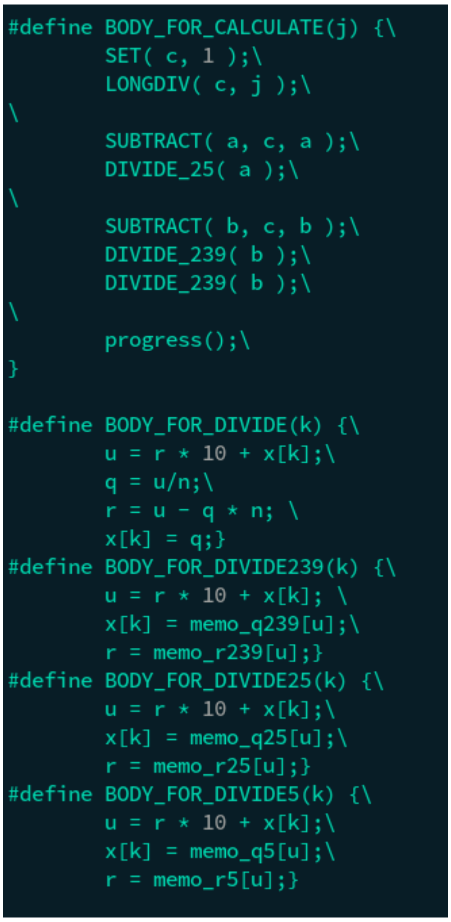
Fent profiling de la nostra millor versió de la pràctica anterior veiem que hi ha molts salts a les funcions calculate (que executa els divides) i LONGDIV.



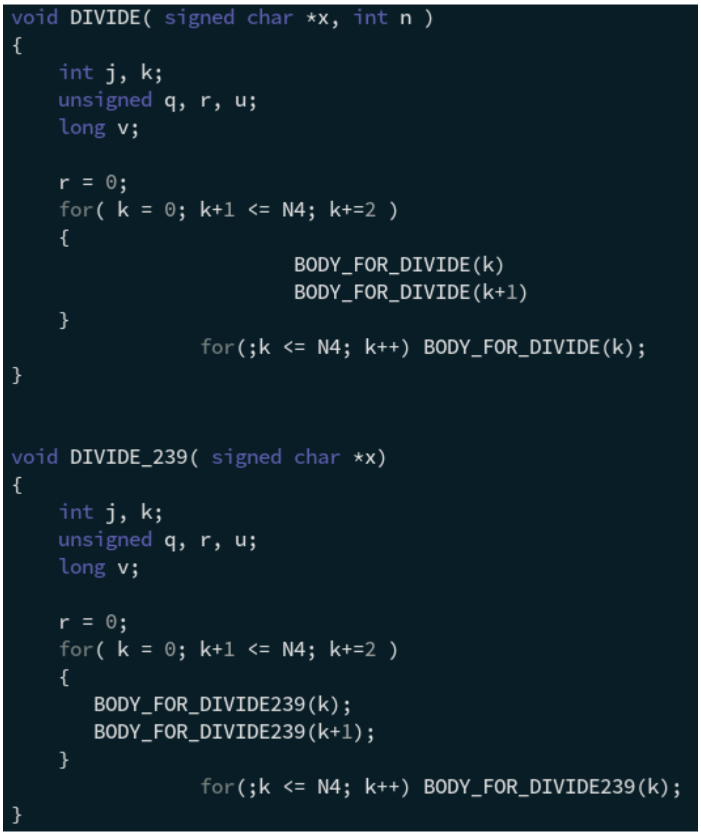
```
Samples: 15K of event 'branches', Event count (approx.): 887318439
Overhead  Command     Shared Object       Symbol
  79.72%  pi.opt3.g3  pi.opt3.g3          [.] calculate
  18.32%  pi.opt3.g3  pi.opt3.g3          [.] LONGDIV
   0.11%  pi.opt3.g3  [kernel.kallsyms]   [k] psi_group_change
```

887,3 M de branches.

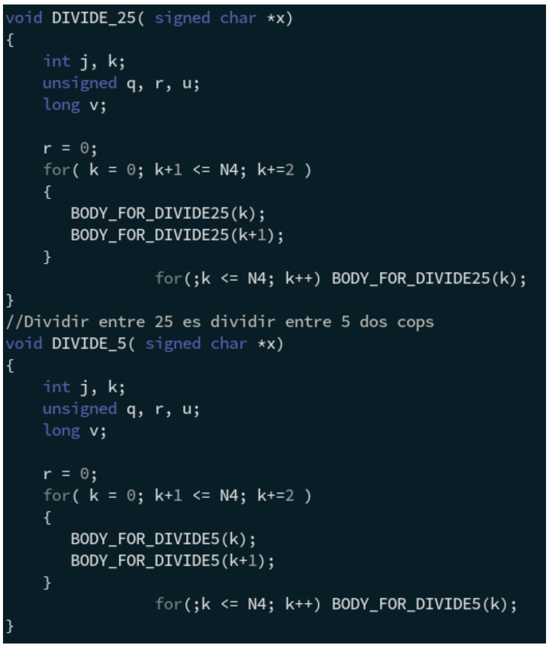
Per a començar apliquem un unroll de 2:



```
#define BODY_FOR_CALCULATE(j) {\
        SET( c, 1 );\
        LONGDIV( c, j );\
\
        SUBTRACT( a, c, a );\
        DIVIDE_25( a );\
\
        SUBTRACT( b, c, b );\
        DIVIDE_239( b );\
        DIVIDE_239( b );\
\
        progress();\
}

#define BODY_FOR_DIVIDE(k) {\
        u = r * 10 + x[k];\
        q = u/n;\
        r = u - q * n; \
        x[k] = q;}
#define BODY_FOR_DIVIDE239(k) {\
        u = r * 10 + x[k]; \
        x[k] = memo_q239[u];\
        r = memo_r239[u];}
#define BODY_FOR_DIVIDE25(k) {\
        u = r * 10 + x[k];\
        x[k] = memo_q25[u];\
        r = memo_r25[u];}
#define BODY_FOR_DIVIDE5(k) {\
        u = r * 10 + x[k];\
        x[k] = memo_q5[u];\
        r = memo_r5[u];}
```

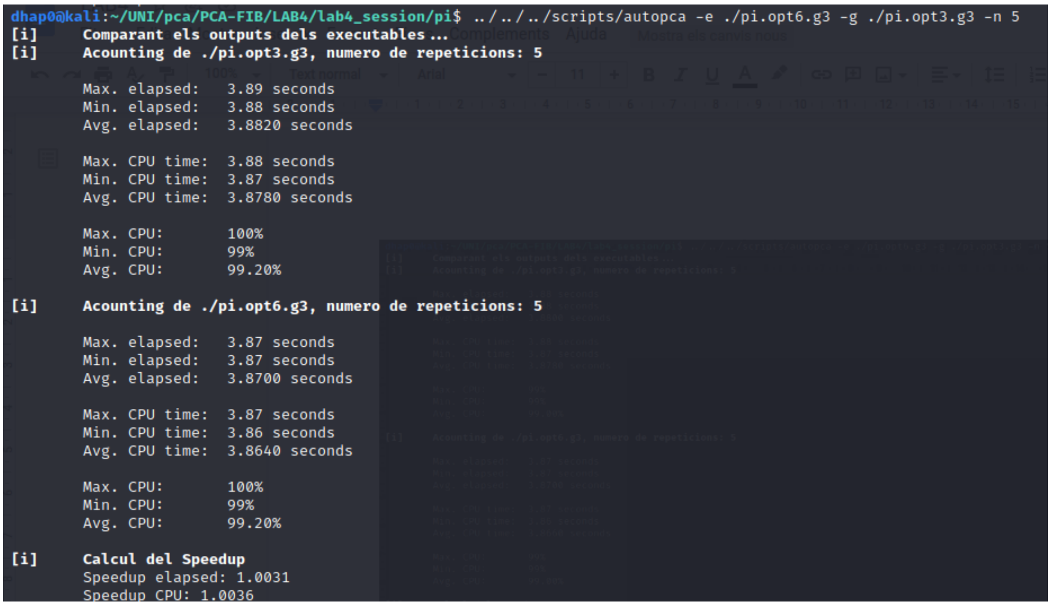


```
void DIVIDE( signed char *x, int n )
{
    int j, k;
    unsigned q, r, u;
    long v;

    r = 0;
    for( k = 0; k+1 <= N4; k+=2 )
    {
                    BODY_FOR_DIVIDE(k)
                    BODY_FOR_DIVIDE(k+1)
    }
            for(;k <= N4; k++) BODY_FOR_DIVIDE(k);
}


void DIVIDE_239( signed char *x)
{
    int j, k;
    unsigned q, r, u;
    long v;

    r = 0;
    for( k = 0; k+1 <= N4; k+=2 )
    {
       BODY_FOR_DIVIDE239(k);
       BODY_FOR_DIVIDE239(k+1);
    }
            for(;k <= N4; k++) BODY_FOR_DIVIDE239(k);
}
```



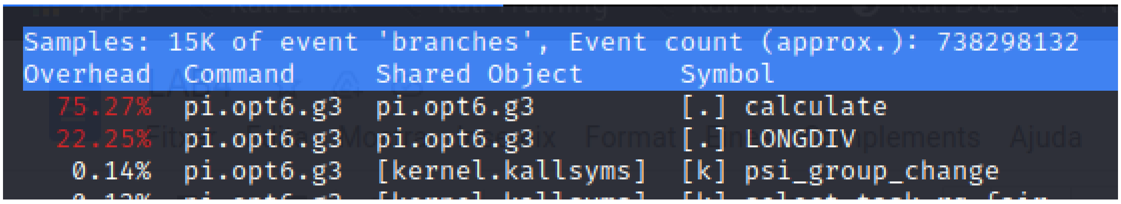
```
void DIVIDE_25( signed char *x)
{
    int j, k;
    unsigned q, r, u;
    long v;

    r = 0;
    for( k = 0; k+1 <= N4; k+=2 )
    {
       BODY_FOR_DIVIDE25(k);
       BODY_FOR_DIVIDE25(k+1);
    }
            for(;k <= N4; k++) BODY_FOR_DIVIDE25(k);
}
//Dividir entre 25 es dividir entre 5 dos cops
void DIVIDE_5( signed char *x)
{
    int j, k;
    unsigned q, r, u;
    long v;

    r = 0;
    for( k = 0; k+1 <= N4; k+=2 )
    {
       BODY_FOR_DIVIDE5(k);
       BODY_FOR_DIVIDE5(k+1);
    }
            for(;k <= N4; k++) BODY_FOR_DIVIDE5(k);
}
```

Timing d’aquesta versió anomenada **pi.opt6.c** que obté un speedup respecte el laboratori anterior de **1.0031**:



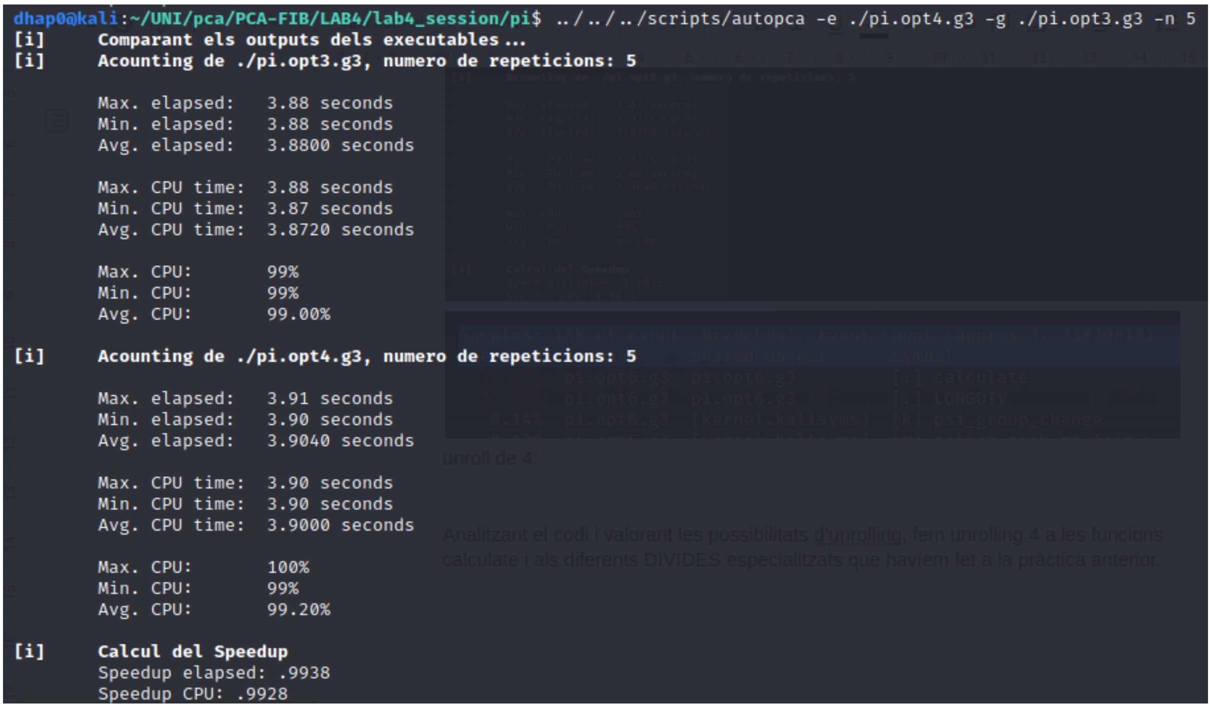
```
dhap0@kali:~/UNI/pca/PCA-FIB/LAB4/lab4_session/pi$ ../../../scripts/autopca -e ./pi.opt6.g3 -g ./pi.opt3.g3 -n 5
[i]      Comparant els outputs dels executables ...
[i]      Acounting de ./pi.opt3.g3, numero de repeticions: 5

         Max. elapsed:   3.89 seconds
         Min. elapsed:   3.88 seconds
         Avg. elapsed:   3.8820 seconds

         Max. CPU time:  3.88 seconds
         Min. CPU time:  3.87 seconds
         Avg. CPU time:  3.8780 seconds

         Max. CPU:       100%
         Min. CPU:       99%
         Avg. CPU:       99.20%

[i]      Acounting de ./pi.opt6.g3, numero de repeticions: 5

         Max. elapsed:   3.87 seconds
         Min. elapsed:   3.87 seconds
         Avg. elapsed:   3.8700 seconds

         Max. CPU time:  3.87 seconds
         Min. CPU time:  3.86 seconds
         Avg. CPU time:  3.8640 seconds

         Max. CPU:       100%
         Min. CPU:       99%
         Avg. CPU:       99.20%

[i]      Calcul del Speedup
         Speedup elapsed: 1.0031
         Speedup CPU: 1.0036
```

I al profiling veiem que hem reduït el nombre de branches:

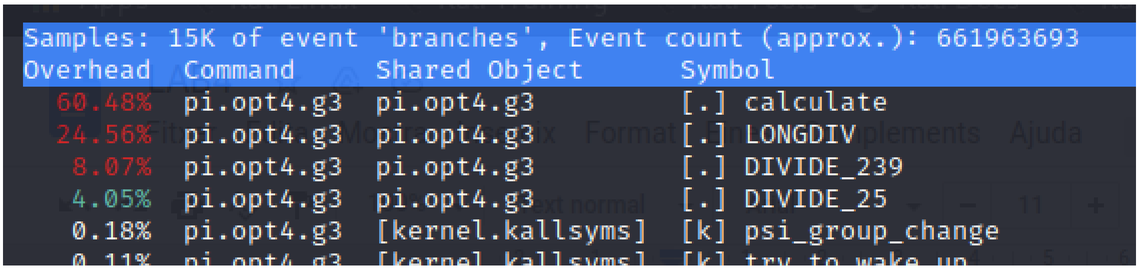


```
Samples: 15K of event 'branches', Event count (approx.): 738298132
Overhead  Command      Shared Object      Symbol
  75.27%  pi.opt6.g3   pi.opt6.g3         [.] calculate
  22.25%  pi.opt6.g3   pi.opt6.g3         [.] LONGDIV
   0.14%  pi.opt6.g3   [kernel.kallsyms]  [k] psi_group_change
```

Si apliquem un unroll de 4 perdem tot l'speedup:

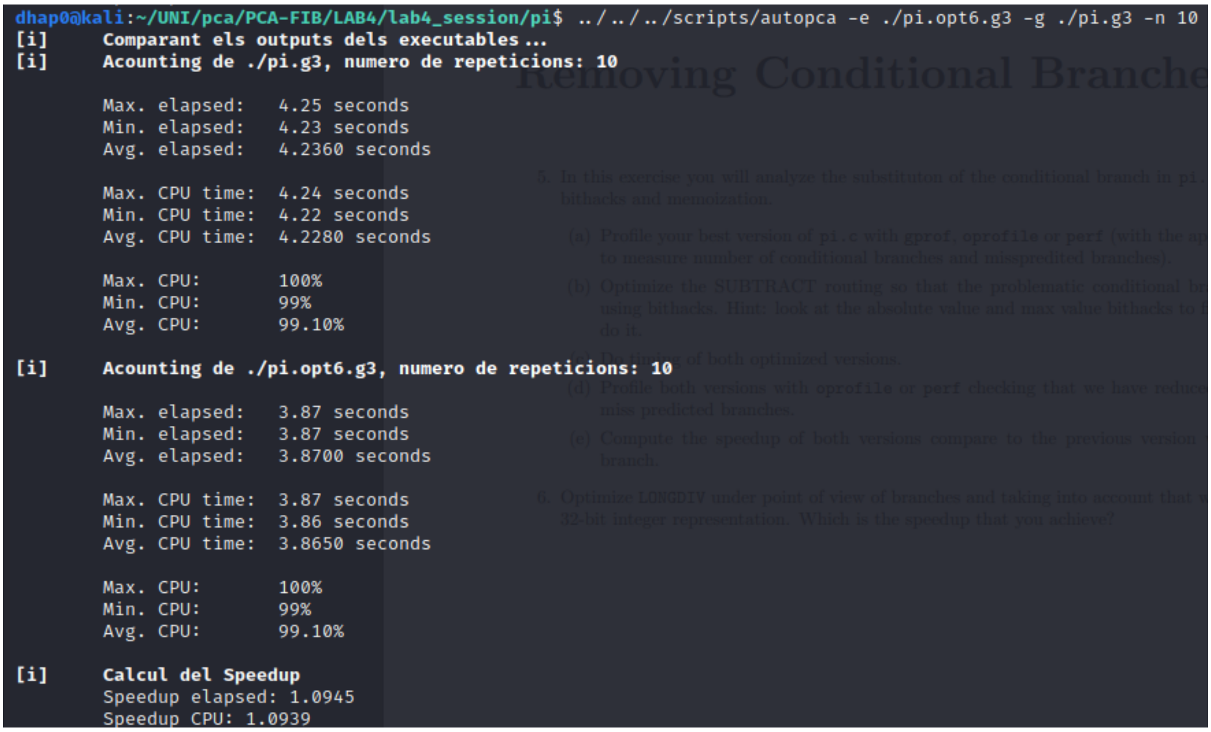


```
dhap0@kali:~/UNI/pca/PCA-FIB/LAB4/lab4_session/pi$ ../../../scripts/autopca -e ./pi.opt4.g3 -g ./pi.opt3.g3 -n 5
[i]      Comparant els outputs dels executables ...
[i]      Acounting de ./pi.opt3.g3, numero de repeticions: 5

         Max. elapsed:   3.88 seconds
         Min. elapsed:   3.88 seconds
         Avg. elapsed:   3.8800 seconds

         Max. CPU time:  3.88 seconds
         Min. CPU time:  3.87 seconds
         Avg. CPU time:  3.8720 seconds

         Max. CPU:       99%
         Min. CPU:       99%
         Avg. CPU:       99.00%

[i]      Acounting de ./pi.opt4.g3, numero de repeticions: 5

         Max. elapsed:   3.91 seconds
         Min. elapsed:   3.90 seconds
         Avg. elapsed:   3.9040 seconds

         Max. CPU time:  3.90 seconds
         Min. CPU time:  3.90 seconds
         Avg. CPU time:  3.9000 seconds

         Max. CPU:       100%
         Min. CPU:       99%
         Avg. CPU:       99.20%

[i]      Calcul del Speedup
         Speedup elapsed: .9938
         Speedup CPU: .9928
```



```
Samples: 15K of event 'branches', Event count (approx.): 661963693
Overhead  Command     Shared Object        Symbol
  60.48%  pi.opt4.g3  pi.opt4.g3           [.] calculate
  24.56%  pi.opt4.g3  pi.opt4.g3           [.] LONGDIV
   8.07%  pi.opt4.g3  pi.opt4.g3           [.] DIVIDE_239
   4.05%  pi.opt4.g3  pi.opt4.g3           [.] DIVIDE_25
   0.18%  pi.opt4.g3  [kernel.kallsyms]    [k] psi_group_change
   0.11%  pi.opt4.g3  [kernel.kallsyms]    [k] try_to_wake_up
```

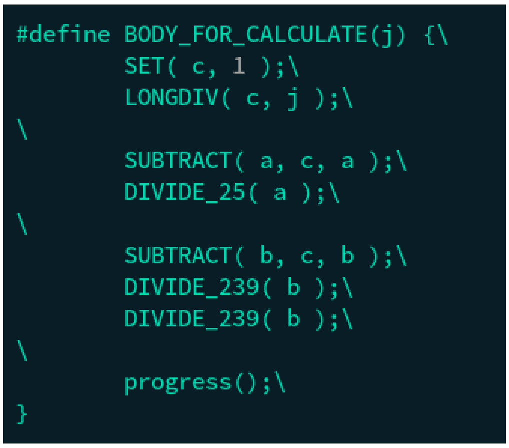
Doncs la versió definitiva d’aquesta secció de la pràctica és **pi.opt6.c** que ens dona un speedup respecte el pi.c original de **1.0945.**



```
dhap0@kali:~/UNI/pca/PCA-FIB/LAB4/lab4_session/pi$ ../../../scripts/autopca -e ./pi.opt6.g3 -g ./pi.g3 -n 10
[i]     Comparant els outputs dels executables...
[i]     Acounting de ./pi.g3, numero de repeticions: 10

        Max. elapsed:   4.25 seconds
        Min. elapsed:   4.23 seconds
        Avg. elapsed:   4.2360 seconds

        Max. CPU time:  4.24 seconds
        Min. CPU time:  4.22 seconds
        Avg. CPU time:  4.2280 seconds

        Max. CPU:       100%
        Min. CPU:       99%
        Avg. CPU:       99.10%

[i]     Acounting de ./pi.opt6.g3, numero de repeticions: 10

        Max. elapsed:   3.87 seconds
        Min. elapsed:   3.87 seconds
        Avg. elapsed:   3.8700 seconds

        Max. CPU time:  3.87 seconds
        Min. CPU time:  3.86 seconds
        Avg. CPU time:  3.8650 seconds

        Max. CPU:       100%
        Min. CPU:       99%
        Avg. CPU:       99.10%

[i]     Calcul del Speedup
        Speedup elapsed: 1.0945
        Speedup CPU: 1.0939
```

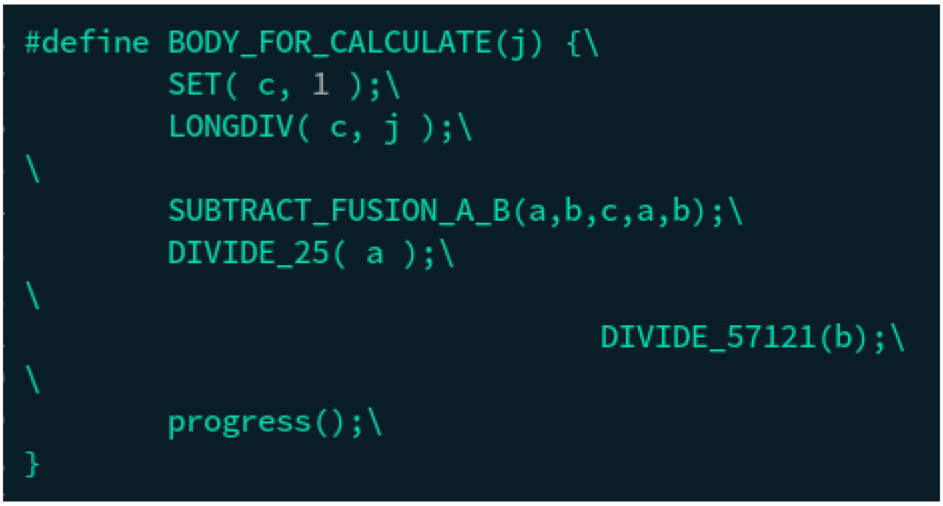
LoopFusion

Fusionem els DIVIDE_239 i els SUBSTRACT(a,c,a) i SUBSTRACT (b,c,b) de la següent fracció de codi:



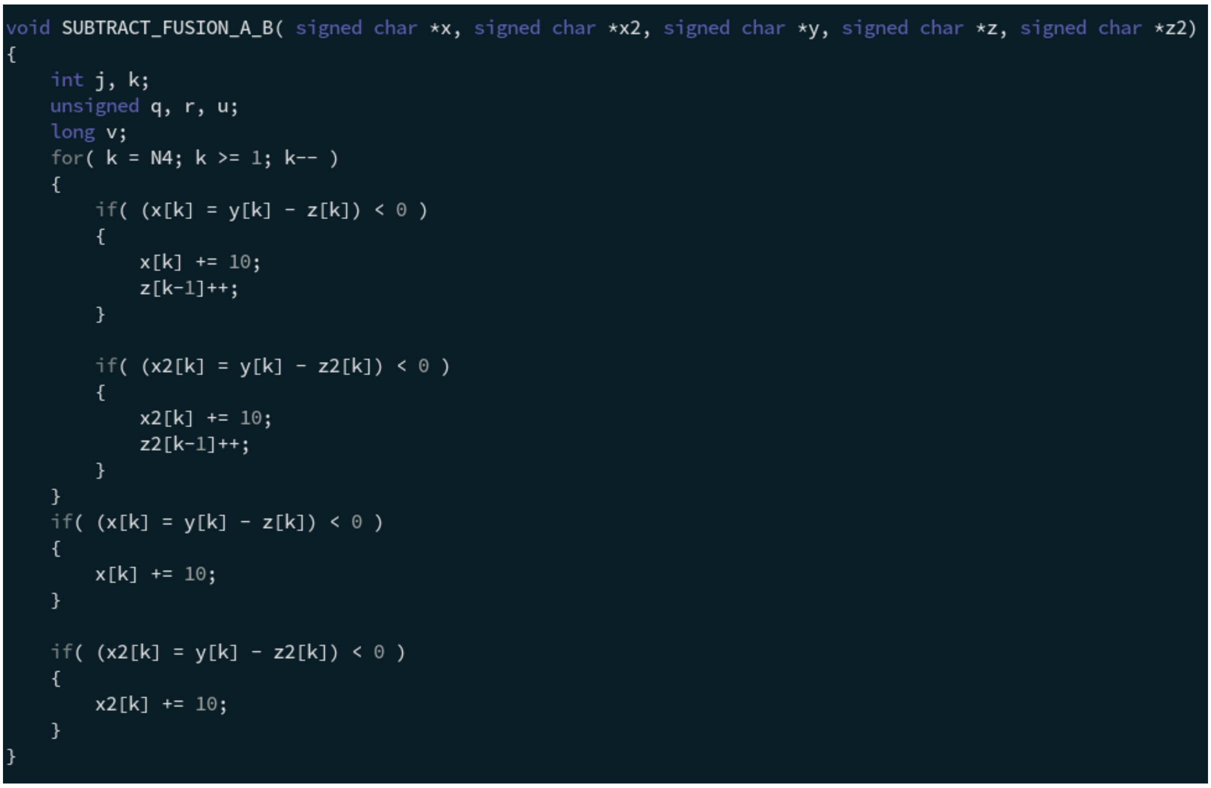
```
#define BODY_FOR_CALCULATE(j) {\
        SET( c, 1 );\
        LONGDIV( c, j );\
\
        SUBTRACT( a, c, a );\
        DIVIDE_25( a );\
\
        SUBTRACT( b, c, b );\
        DIVIDE_239( b );\
        DIVIDE_239( b );\
\
        progress();\
}
```

De tal manera que la macro BODY_FOR_CALCULATE quedarà de la següent manera:

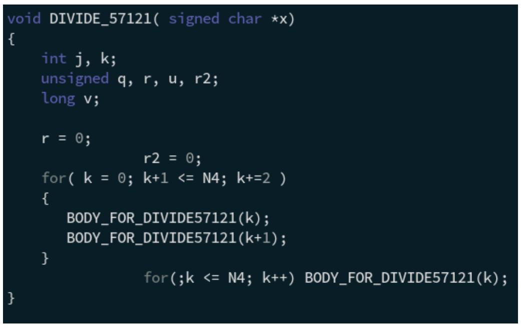


```
#define BODY_FOR_CALCULATE(j) {\
        SET( c, 1 );\
        LONGDIV( c, j );\
\
        SUBTRACT_FUSION_A_B(a,b,c,a,b);\
        DIVIDE_25( a );\
\
                                DIVIDE_57121(b);\
\
        progress();\
}
```

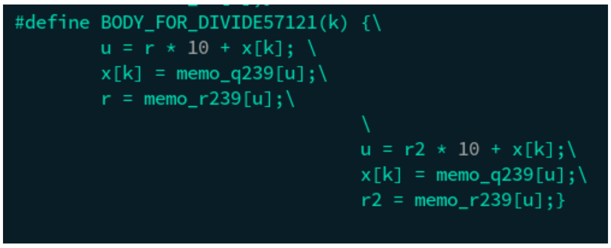
i les noves funcions:



```
void SUBTRACT_FUSION_A_B( signed char *x, signed char *x2, signed char *y, signed char *z, signed char *z2)
{
    int j, k;
    unsigned q, r, u;
    long v;
    for( k = N4; k >= 1; k-- )
    {
        if( (x[k] = y[k] - z[k]) < 0 )
        {
            x[k] += 10;
            z[k-1]++;
        }

        if( (x2[k] = y[k] - z2[k]) < 0 )
        {
            x2[k] += 10;
            z2[k-1]++;
        }
    }
    if( (x[k] = y[k] - z[k]) < 0 )
    {
        x[k] += 10;
    }

    if( (x2[k] = y[k] - z2[k]) < 0 )
    {
        x2[k] += 10;
    }
}
```

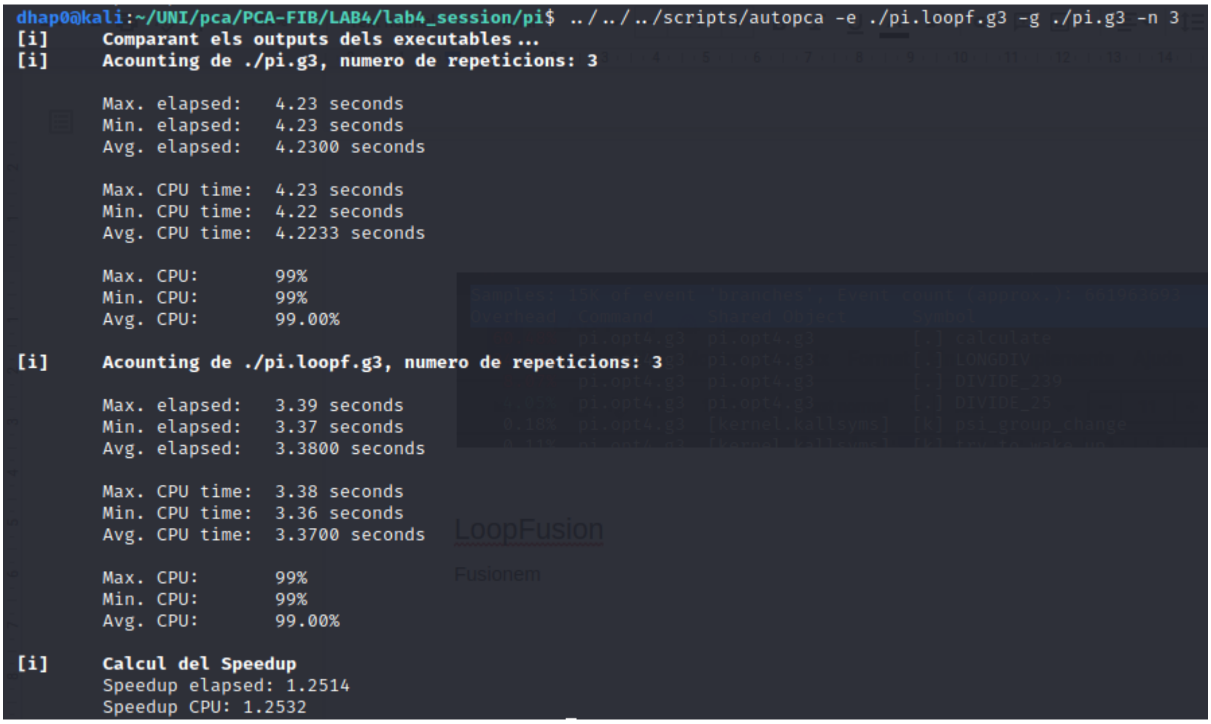


```
void DIVIDE_57121( signed char *x)
{
    int j, k;
    unsigned q, r, u, r2;
    long v;

    r = 0;
            r2 = 0;
    for( k = 0; k+1 <= N4; k+=2 )
    {
       BODY_FOR_DIVIDE57121(k);
       BODY_FOR_DIVIDE57121(k+1);
    }
            for(;k <= N4; k++) BODY_FOR_DIVIDE57121(k);
}
```



```
#define BODY_FOR_DIVIDE57121(k) {\
        u = r * 10 + x[k]; \
        x[k] = memo_q239[u];\
        r = memo_r239[u];\
                            \
                            u = r2 * 10 + x[k];\
                            x[k] = memo_q239[u];\
                            r2 = memo_r239[u];}
```

Aquesta nova versió que anomenem **pi.loopf.c** aconsegueix un speedup respecte el programa original de **1.2514.**

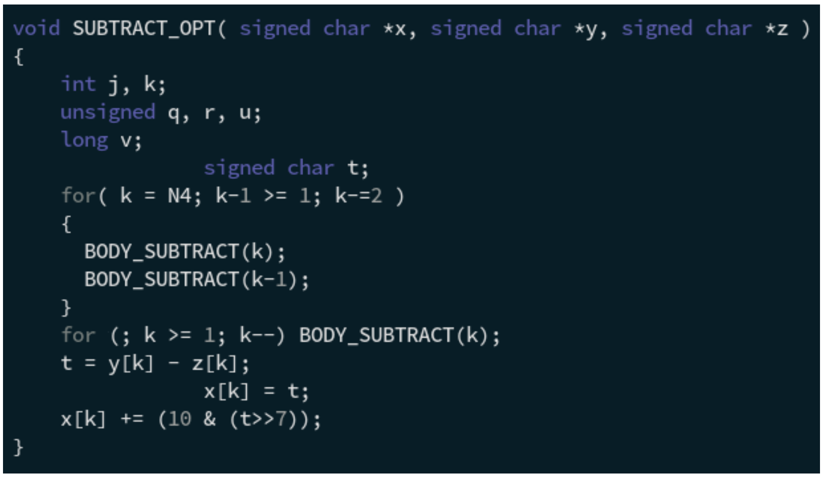


```
dhap0@kali:~/UNI/pca/PCA-FIB/LAB4/lab4_session/pi$ ../../../scripts/autopca -e ./pi.loopf.g3 -g ./pi.g3 -n 3
[i]     Comparant els outputs dels executables...
[i]     Acounting de ./pi.g3, numero de repeticions: 3

        Max. elapsed:   4.23 seconds
        Min. elapsed:   4.23 seconds
        Avg. elapsed:   4.2300 seconds

        Max. CPU time:  4.23 seconds
        Min. CPU time:  4.22 seconds
        Avg. CPU time:  4.2233 seconds

        Max. CPU:       99%
        Min. CPU:       99%
        Avg. CPU:       99.00%

[i]     Acounting de ./pi.loopf.g3, numero de repeticions: 3

        Max. elapsed:   3.39 seconds
        Min. elapsed:   3.37 seconds
        Avg. elapsed:   3.3800 seconds

        Max. CPU time:  3.38 seconds
        Min. CPU time:  3.36 seconds
        Avg. CPU time:  3.3700 seconds

        Max. CPU:       99%
        Min. CPU:       99%
        Avg. CPU:       99.00%

[i]     Calcul del Speedup
        Speedup elapsed: 1.2514
        Speedup CPU: 1.2532
```

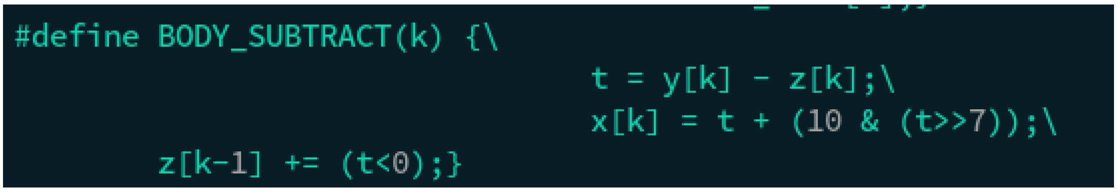
Removing Conditional Branches

En aquesta part de la pràctica es treballa en una versió que anomenarem **pi.loopf.opt2.c.** Aquest nou codi incorporarà la millora de les funcions SUBTRACT i LONGDIV per tal d'eliminar els salts condicionals per mitjà de bithacks. També s'ha afegit unroll a les funcions SUBTRACT que no s'havia incorporat anteriorment. La funció LONGDIV ha sigut canviada per la funció DIVIDE optimitzada amb unrolling.

Canvis a la funció SUBTRACT:

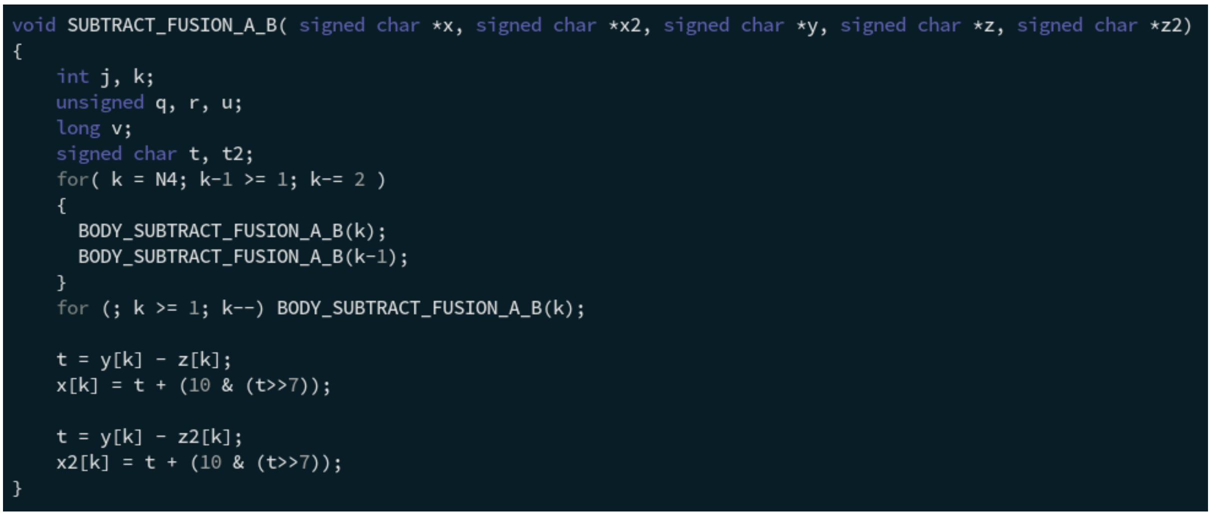


```
void SUBTRACT_OPT( signed char *x, signed char *y, signed char *z )
{
    int j, k;
    unsigned q, r, u;
    long v;
                signed char t;
    for( k = N4; k-1 >= 1; k-=2 )
    {
      BODY_SUBTRACT(k);
      BODY_SUBTRACT(k-1);
    }
    for (; k >= 1; k--) BODY_SUBTRACT(k);
    t = y[k] - z[k];
                x[k] = t;
    x[k] += (10 & (t>>7));
}
```

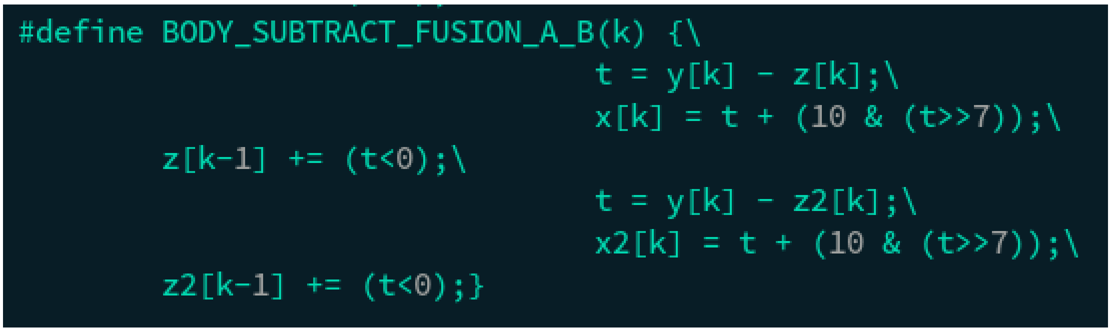


```
#define BODY_SUBTRACT(k) {\
                                t = y[k] - z[k];\
                                x[k] = t + (10 & (t>>7));\
        z[k-1] += (t<0);}
```

Canvis a la funció SUBTRACT_FUSION_A_B:

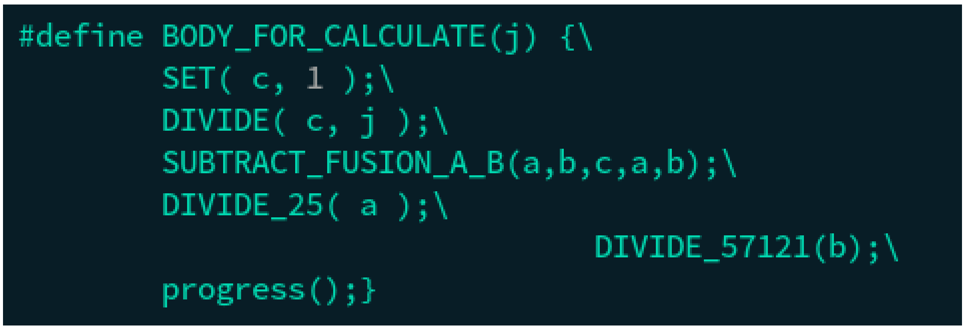


```
void SUBTRACT_FUSION_A_B( signed char *x, signed char *x2, signed char *y, signed char *z, signed char *z2)
{
    int j, k;
    unsigned q, r, u;
    long v;
    signed char t, t2;
    for( k = N4; k-1 >= 1; k-= 2 )
    {
      BODY_SUBTRACT_FUSION_A_B(k);
      BODY_SUBTRACT_FUSION_A_B(k-1);
    }
    for (; k >= 1; k--) BODY_SUBTRACT_FUSION_A_B(k);

    t = y[k] - z[k];
    x[k] = t + (10 & (t>>7));

    t = y[k] - z2[k];
    x2[k] = t + (10 & (t>>7));
}
```

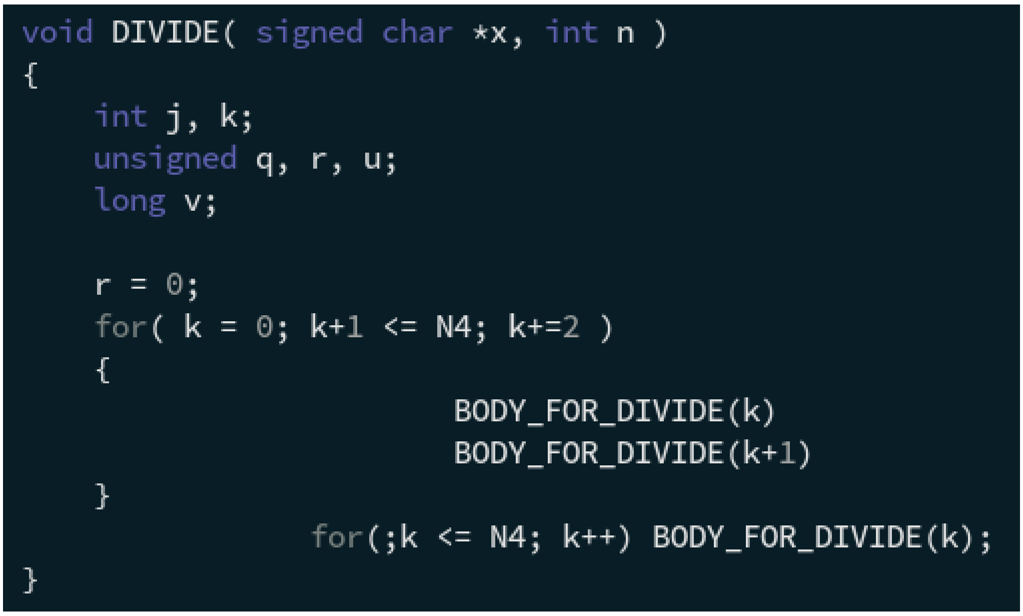


```
#define BODY_SUBTRACT_FUSION_A_B(k) {\
                                t = y[k] - z[k];\
                                x[k] = t + (10 & (t>>7));\
        z[k-1] += (t<0);\
                                t = y[k] - z2[k];\
                                x2[k] = t + (10 & (t>>7));\
        z2[k-1] += (t<0);}
```

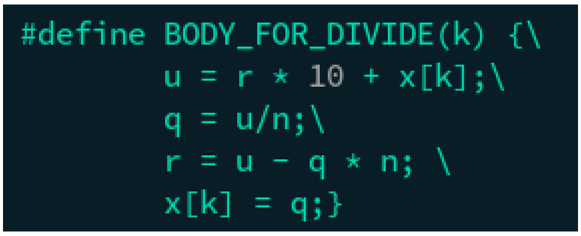
La funció LONGDIV ha sigut substituida per la funció DIVIDE i s’ha aplicat l’unroll com als altres DIVIDEs:



```
#define BODY_FOR_CALCULATE(j) {\
        SET( c, 1 );\
        DIVIDE( c, j );\
        SUBTRACT_FUSION_A_B(a,b,c,a,b);\
        DIVIDE_25( a );\
                                DIVIDE_57121(b);\
        progress();}
```

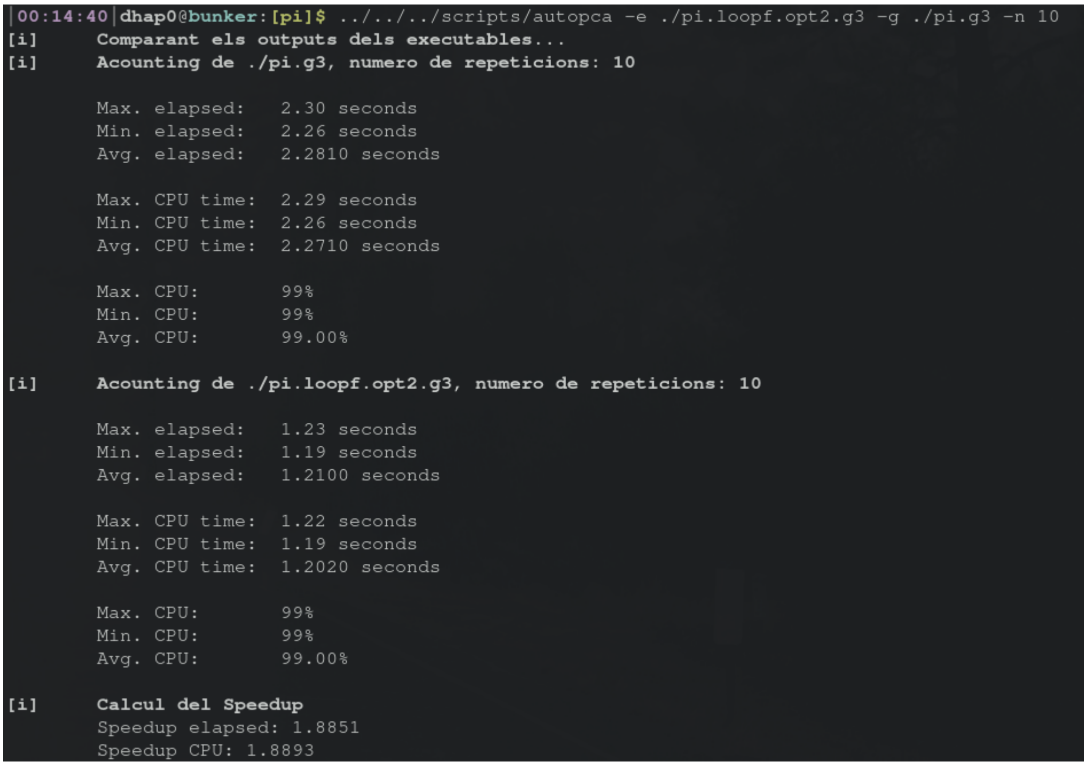


```c
void DIVIDE( signed char *x, int n )
{
    int j, k;
    unsigned q, r, u;
    long v;

    r = 0;
    for( k = 0; k+1 <= N4; k+=2 )
    {
                    BODY_FOR_DIVIDE(k)
                    BODY_FOR_DIVIDE(k+1)
    }
            for(;k <= N4; k++) BODY_FOR_DIVIDE(k);
}
```



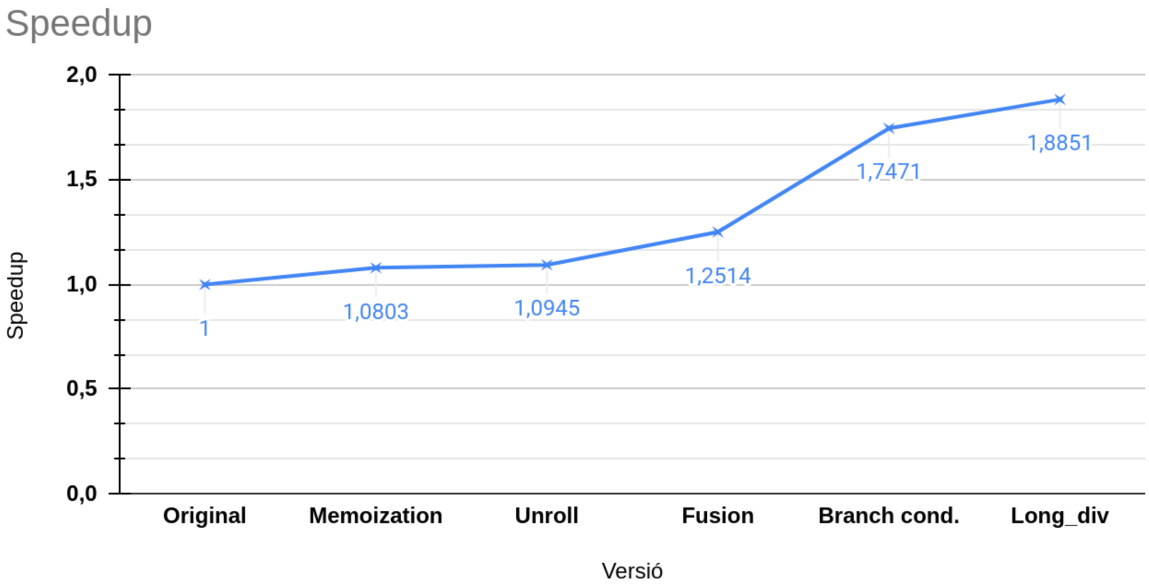
```c
#define BODY_FOR_DIVIDE(k) {\
        u = r * 10 + x[k];\
        q = u/n;\
        r = u - q * n; \
        x[k] = q;}
```

Aquesta versió del codi és la definitiva. Entregada per l'usuari **pca06**, anomenat **pi.loopf.opt2.c** amb un speedup respecte l'original de **1.8851.**



```
|00:14:40|dhap0@bunker:[pi]$ ../../../scripts/autopca -e ./pi.loopf.opt2.g3 -g ./pi.g3 -n 10
[i]     Comparant els outputs dels executables...
[i]     Acounting de ./pi.g3, numero de repeticions: 10

        Max. elapsed:   2.30 seconds
        Min. elapsed:   2.26 seconds
        Avg. elapsed:   2.2810 seconds

        Max. CPU time:  2.29 seconds
        Min. CPU time:  2.26 seconds
        Avg. CPU time:  2.2710 seconds

        Max. CPU:       99%
        Min. CPU:       99%
        Avg. CPU:       99.00%

[i]     Acounting de ./pi.loopf.opt2.g3, numero de repeticions: 10

        Max. elapsed:   1.23 seconds
        Min. elapsed:   1.19 seconds
        Avg. elapsed:   1.2100 seconds

        Max. CPU time:  1.22 seconds
        Min. CPU time:  1.19 seconds
        Avg. CPU time:  1.2020 seconds

        Max. CPU:       99%
        Min. CPU:       99%
        Avg. CPU:       99.00%

[i]     Calcul del Speedup
        Speedup elapsed: 1.8851
        Speedup CPU: 1.8893
```

La taula següent mostra els diferents speedups que s’han anat aconseguint en les diferents versions del codi pi.c.


Speedup
Speedup
Original
Memoization
Unroll
Fusion
Branch cond.
Long_div
Versió