Al usar perf, utilizamos un evento llamado *"fp_arith_inst_retired.256b_packed_double"* y cuya descripción nos dice *"Number of SSE/AVX computational 256-bit packed double precision floating-point instructions retired. Each count represents 4 computations. Applies to SSE* and AVX* packed double precision floating-point instructions: ADD SUB MUL DIV MIN MAX SQRT DPP FM(N)ADD/SUB. DPP and FM(N)ADD/SUB instructions count twice as they perform multiple calculations per element"*

Es decir, cada medida del evento *fp arith inst retired.256b packed double* indica que se han realizado cuatro instrucciones de punto flotante (pueden ser sumas, restas, multiplicaciones, etc.).

En la siguiente imagen se han medido las ejecuciones de dos programas: *mat_mul.py* y *mat_mul_zeros.py*.

```
jlpadillas01@PH315-51:~/TFG$ make perf
sudo sysctl -w kernel.nmi_watchdog=0
kernel.nmi_watchdog = 0
sudo /usr/bin/perf stat --event instructions,cycles,fp_arith_inst_retired.128b_packed_double,fp_arith_inst_retired.128b_packed_single,fp_arit
ngle,fp_arith_inst_retired.scalar_double,fp_arith_inst_retired.scalar_single,fp_assist.any --cpu=0 taskset -c 0 python3 src/mat_mul.py

 Performance counter stats for 'CPU(s) 0':

    60.174.510.301      instructions              #    3,48  insn per cycle           (55,37%)
    17.307.943.403      cycles                                                        (66,53%)
                 6      fp_arith_inst_retired.128b_packed_double                             (66,63%)
                 0      fp_arith_inst_retired.128b_packed_single                             (66,72%)
    62.560.912.768      fp_arith_inst_retired.256b_packed_double                             (66,80%)
                 0      fp_arith_inst_retired.256b_packed_single                             (66,80%)
         5.203.334      fp_arith_inst_retired.scalar_double                        (44,44%)
         2.659.916      fp_arith_inst_retired.scalar_single                        (44,35%)
                 0      fp_assist.any                                              (44,27%)

       4,373287950 seconds time elapsed

sudo /usr/bin/perf stat --event instructions,cycles,fp_arith_inst_retired.128b_packed_double,fp_arith_inst_retired.128b_packed_single,fp_arit
ngle,fp_arith_inst_retired.scalar_double,fp_arith_inst_retired.scalar_single,fp_assist.any --cpu=0 taskset -c 0 python3 src/mat_mul_zeros.py

 Performance counter stats for 'CPU(s) 0':

    60.289.906.579      instructions              #    3,46  insn per cycle           (55,52%)
    17.416.990.640      cycles                                                        (66,64%)
                 3      fp_arith_inst_retired.128b_packed_double                             (66,64%)
                 0      fp_arith_inst_retired.128b_packed_single                             (66,64%)
    62.620.400.189      fp_arith_inst_retired.256b_packed_double                             (66,64%)
                 0      fp_arith_inst_retired.256b_packed_single                             (66,64%)
         4.756.251      fp_arith_inst_retired.scalar_double                        (44,48%)
         2.541.507      fp_arith_inst_retired.scalar_single                        (44,48%)
                 0      fp_assist.any                                              (44,48%)

       4,388223459 seconds time elapsed

sudo sysctl -w kernel.nmi_watchdog=1
kernel.nmi_watchdog = 1
jlpadillas01@PH315-51:~/TFG$
```

En ambos programas se hace uso de la librería *Numpy* para *Python*; en concreto, se usa la función *empty()* que permite generar matrices con contenido aleatorio, y, la otra función *matmul()* que permite multiplicar matrices.

Así, el programa *mat_mul.py* genera dos matrices cuadradas de orden 5.000 y las multiplica consiguiendo un número de instrucciones (estimados por *perf*) de unos 60,17E9; y, unos 62,56E9 eventos (medidos) de *fp_arith_inst_retired.256b_packed_double.*

Por tanto, tenemos que

$$\textbf{\textit{numOperaciones}}(\textbf{\textit{perf}}) = 62{,}56\text{E}9 \cdot 4 = \mathbf{250{,}24E9}$$

el número de operaciones medidos durante la ejecución del programa es de unos 250,24E9.

Veamos un ejemplo de una multiplicación de dos matrices de orden 3:

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} \cdot \begin{bmatrix} 8 & 7 & 6 \\ 5 & 4 & 3 \\ 2 & 1 & 0 \end{bmatrix} = \begin{bmatrix} (0\cdot 8+1\cdot 5+2\cdot 2) & (0\cdot 7+1\cdot 4+2\cdot 1) & (0\cdot 6+1\cdot 3+2\cdot 0) \\ (3\cdot 8+4\cdot 5+5\cdot 2) & (3\cdot 7+4\cdot 4+5\cdot 1) & (3\cdot 6+4\cdot 3+5\cdot 0) \\ (6\cdot 8+7\cdot 5+8\cdot 2) & (6\cdot 7+7\cdot 4+8\cdot 1) & (6\cdot 6+7\cdot 3+8\cdot 0) \end{bmatrix} = \begin{bmatrix} 9 & 6 & 3 \\ 54 & 1 & 2 \\ 99 & 78 & 57 \end{bmatrix}$$

Por cada elemento, podemos ver que se hacen 3 multilpicaciones y 2 sumas. Teniendo unos 3^2 elementos, tenemos $3^2(3+2)=45$ operaciones.

Si lo extrapolamos para una matriz de tamaño N, tenemos que el número de operaciones es igual a

$$numOperaciones = N^2(2N-1)$$

Aplicamos dicha fórmula para nuestra matriz de orden 5.000:

$$\boldsymbol{numOperaciones}(\boldsymbol{N=5.000}) = 5.000 \cdot 5.000(9.999) = 249.975.000.000 = \mathbf{249,98E9}$$

Por tanto, con todo ello conseguimos que lo medido con perf se asemeja a lo que esperamos obtener

$$numOperaciones(perf) = 250,24E9 \approx numOperaciones(N=5.000) = 249,98E9$$