Organización de computadores

21 de diciembre de 2021

Universidad de Castilla-La Mancha

Legenda
Herramienta
Ejercicio

Estos apuntes pertenecen a la asignatura *Organización de Computadores* del grado en ingeniería informática impartido en la Escuela Superior de Informática de la Universidad de Castilla-La Mancha. Elaborados por Félix Jesús Villanueva Molina.

Índice

Introducción

En los últimos años, hemos asistido a un evento muy interesante en la arquitectura de computadores (ver figura 1), a pesar de que la tecnología y la capacidad de integración de mas puertas lógicas sigue la ley de Moore, las prestaciones que deberían ir de la mano han tenido que adoptar otro tipo de optimizaciones para poder seguir incrementando el rendimiento.

Puedes consultar las nuevas arquitecturas/procesadores en `https://en.wikichip.org`

Este es el objetivo de este primer tema.

Para poder estudiar qué ha pasado, necesitamos entender cómo medir el rendimiento y qué debemos buscar cuando planteamos una mejora de una arquitectura. Este estudio del rendimiento se hace a todos los niveles de abstracción, por ejemplo, desde el proceso químico seguido para construir los transistores con los cuales construimos los registros utilizados en los puertos de entrada/salida (Figura 2).

Prestaciones y rendimiento

Cualquier proceso de ingeniería, el diseño e implementación de hardware o software, evaluación de soluciones/configuraciones hard-

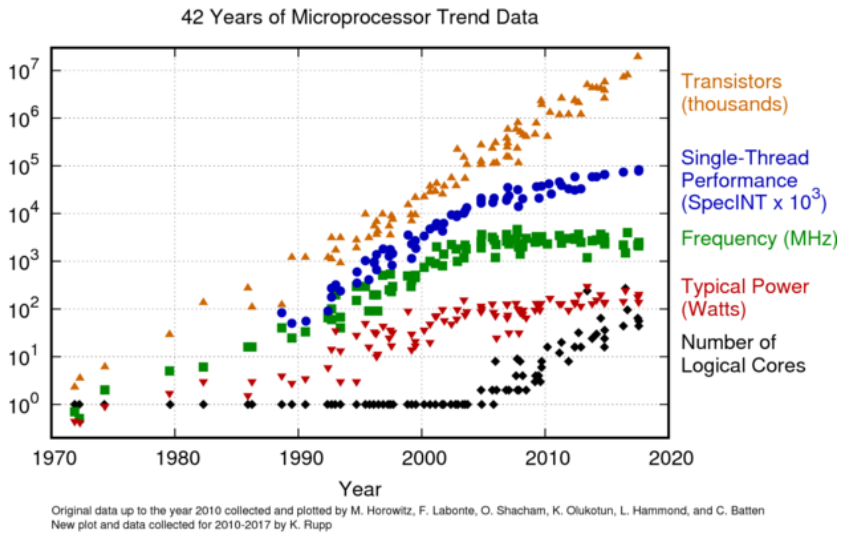

Figura 1: Tendencia en procesadores. Figura elaborada por Karl Rupp y extraida de su blog `https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/`

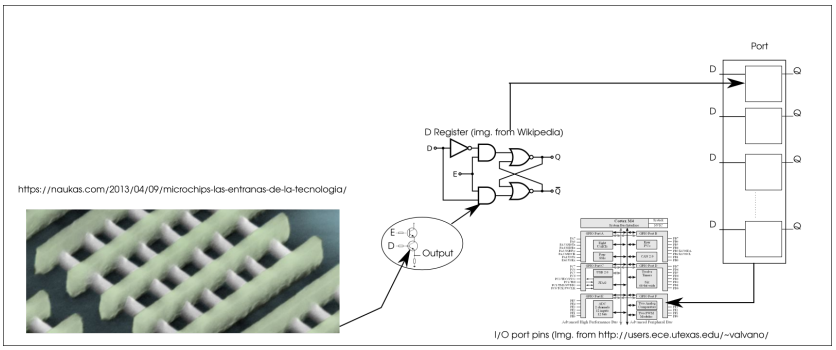

Figura 2: Niveles de abstracción en el estudio de microprocesadores

ware/software, etc. debe ir dirigido por un proceso contínuo de evaluación empírica de sus características.

Este proceso de medición de características (coste, ciclos por instrucción, tamaño de la memoria, frecuencia del reloj del microprocesador, etc.) es la única forma de tomar decisiones de una forma cuantitativa y de minimizar los errores a la hora de tomar estas decisiones. Estaremos evaluándo el rendimiento de un determinado hardware cuando escogemos, para el problema que queremos resolver, el que mejor resultado nos da en función de los medios/indicadores utilizados. Es necesario recordar varias puntualizaciones importantes:

- El problema determina las prestaciones mas importantes. Es decir, no es lo mismo seleccionar el hardware para un sensor alimentado por batería, donde el consumo es una variable importante, que el hardware para un ordenador destinado a los videojuegos donde ese mismo consumo, no es de las características mas determinan-

tes y otras irrelevantes en el sensor como los *frames* por segundo. De igual forma no es lo mismo un hardware destinado al procesamiento de imagen que un hardware de propósito general que puede ser destinado a ejecutar cualquier tipo de programa.

- El concepto de *mejor resultado* debe ser cuantitativo, comparable y reproducible. Es decir, debemos comparar usando las mismas medidas/indicadores obtenidos mediante un proceso de medición consistente.

- El coste y la eficiencia (que de nuevo depende del problema), como en muchos procesos de ingeniería, siempre será una de las medidas mas importantes y en función de la que se evaluará el resultado de la solución planteada.

El primer paso para medir las prestaciones de un hardware es identificar qué medidas o indicadores son las mas relevantes en el proceso en el cual estamos inmersos. En función de nuestro objetivo final, daremos prioridad a mejorar unos indicadores u otros. A menudo, estaremos obligados a buscar compromisos al no poder mejorar todos los indicadores o ser estos indicadores contrapuestos (e.j. aumentar la memoria RAM de un computador aumenta el coste de este computador).

Debemos recordar también que, en función de los indicadores elegidos, necesitamos procedimientos para calcular y/o medir esos indicadores con el mínimo error posible.

En sistemas complejos, el concepto de prestaciones y rendimiento de un equipo depende directamente de las prestaciones y rendimiento de cada una de sus partes. Si queremos mejorar el rendimiento de un sistema, debemos estudiar qué componentes mejorar y cómo incide esa mejora en un componente en el rendimiento del sistema al completo.

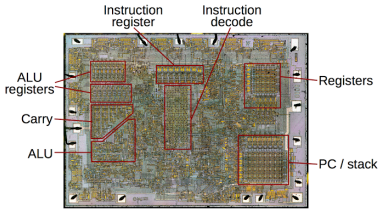

Figura 3: Procesador 8080: Las prestaciones de una arquitectura dependen de las prestaciones de sus partes. Figura extraida de www.techspot.com

Un ejemplo que ilustra este concepto es el del triatlon o *ironman* (figura 4), disciplina deportiva donde tu rendimiento global (el tiempo que tardas en hacer la prueba), depende de tu rendimiento en cada una de las partes, maratón (42,2km), natación (3,86km) y ciclismo (200km) individuales ya que el tiempo se suma. A partir de este sistema global, acortar el tiempo en la parte de natación un 5 % tiene un impacto distinto que acortar el tiempo en la parte de ciclismo en un 5 %. Si tomamos los tiempos habituales en las competiciones de este tipo, una mejora del 5 % en natación (prueba en la que los tiempos rondan la hora) implica una mejora global de unos 2-3 minutos mientras que en la parte de ciclismo (prueba en la que los tiempos están entorno a las 6-7 horas) puede suponer de 16-19 minutos. Esto implica que, sobre el tiempo total de la prueba, la mejora en natación

supondría una mejora en 0,5 % mientras que en ciclismo sería del 3,2 % (tomando como referencia unas 10 horas de duración total de las tres pruebas).

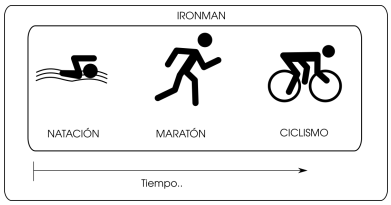

Figura 4: Al igual que en la ruta crítica en sistemas complejos, en el triatlon el tiempo total es la suma de sus partes

Indicadores de rendimiento para CPUs

En todas las disciplinas/tareas existen una serie de indicadores que me sirven para medir la eficiencia. Estos indicadores de rendimiento o *Key Performance indicators (KPI)* se aplican en todas las fases, productos, procesos de ingeniería. [1] ¿Qué indicadores podemos considerar en el rendimiento de una CPU ya construida?. Entre los indicadores globalmente aceptados y mas relevantes para medir el rendimiento de una CPU podemos encontrar:

[1] Identificar los KPIs mas relevantes para todo problema/solución es un proceso clave en Ingeniería y el primero a ser realizado

- **Frecuencia de un procesador** velocidad de operaciones por segundo (transición de los transistores que forman parte del procesador). Se mide en Hz, un hercio sería un cambio de estado por segundo. Cuantos mas rápido sea la frecuencia mas transiciones por segundo y mas información podemos procesar por segundo. En la actualidad los procesadores actuales pueden llegar hasta los 5GHz (5 GigaHercios)[2]

Prefijo	Unidad
Peta	10^{15}
Tera	10^{12}
Giga	10^{9}
Mega	10^{6}
Kilo	10^{3}

Cuadro 1: Prefijos-Unidades útiles

[2] Comparativa de procesadores www.tomshardware.com

$$Periodo = 1/5GHz = 0,000000001\, segundos = 1\, nanosegundo$$

Prefijo	Unidad
mili	10^{-3}
micro	10^{-6}
nano	10^{-9}
pico	10^{-12}

Cuadro 2: Prefijos-Unidades útiles

- **Número de Instrucciones(NI)** es el número de instrucciones que una tarea/programa ejecuta. Aunque no es una medida de rendimiento propiamente dicha, nos servirá para definir medidas de rendimiento como veremos a continuación. Ojo que el número de instrucciones puede depender de las opciones de compilación como también veremos mas adelante.

- **Ciclos por instrucción (CPI)**: Cada tipo de instrucción tarda en ejecutarse un número determinado de ciclos que van en función de su complejidad y del cometido que realize dicha instrucción. Por ejemplo, en el procesador MIPS hay cinco tipos de instrucciones como son carga (5), almacenamiento (4), tipo R (4), salto condicional(3) y salto incondicional(3).El número entre paréntesis es el número de ciclos por instrucción[3]. En general, una arquitectura tendrá un mejor rendimiento cuanto menor sea el número medio de ciclos de reloj por instrucción que utilice.

[3] Wikipedia (MIPS CPI)

- **Instrucciones por ciclo (IPC)**: es la inversa de la anterior, es decir, cuantas instrucciones ejecuta una CPU como media por cada ciclo. El objetivo de las técnicas de parelización es aumentar este número.

- **Tiempo de respuesta, tiempo de ejecución o latencia (TE)** Es el tiempo requerido para completar una tarea. Hay que tener cuidado al medir este tiempo ya que podemos medir tiempo no relacionado con la ejecución de esa tarea, por ejemplo, en los sistemas operativos convencionales, los cambios de contexto, interrupciones, entrada/salida, etc. puede interferir esta medida. **Es el indicador de rendimiento por excelencia, a menor tiempo de ejecución para una determinada tarea, mejor es el rendimiento de ese hardware en esa tarea**. Por lo tanto, rendimiento y tiempo de ejecución son inversamente proporcionales por lo que se define como rendimiento de una máquina X a la inversa de su tiempo de ejecución (TE) para una determinada tarea:

  $Rendimiento_X = \frac{1}{TE_X}$

  A menor tiempo de ejecución, mayor es el rendimiento de una máquina y viceversa.

  El tiempo de ejecución teórico viene dado por el número de instrucciones ejecutadas, por el número de ciclos usado por instrucción y por el tiempo de un ciclo de reloj (periodo):

  $$TE = NI \times CPI \times TC \tag{1}$$

  Donde NI es el número de instrucciones, CPI es el número de Ciclos por Instrucción y TC es el tiempo de ciclo. Varios matices:

  - Se puede calcular un CPI específico de un programa o tarea atendiendo al número y tipo de instrucciones que lo componen.
  - NI es el número de instrucciones ejecutadas no el número de instrucciones que contiene el programa.
  - Cuando ejecutamos una tarea en un sistema operativo convencional, generalmente esa tarea coexiste con otras tareas que le pueden *robar* tiempo.
  - Cuando necesitamos granularidad muy fina, por ejemplo en sistemas de tiempo real críticos, aspectos como interrupciones, número de fallos en los accesos a caché, etc. influyen en el tiempo de ejecución final.

- **Productividad**: número de tareas realizadas por unidad de tiempo.

- **MIPS** *million instructions per second* mide el número de instrucciones por segundo ejecutados por un computador. Hay que tener cuidado con esta medida ya que, a la hora de medirla, el compilador y sus opciones pueden influir en la medida (el número de instrucciones generadas).

- **MFLOPS**: En grandes computadores usados para cáculos científicos, es habitual medir su rendimiento por el número de operaciones en punto flotante por segundo. En inglés *mega floating-point operations per second*. En la actualidad ya es habitual hablar de GFLOPS (Giga-), PFLOPS(Peta-)

Hagamos algunos ejercicios:

1. Un programa ejecuta 3400 instrucciones, de las cuales 200 son saltos condicionales, 300 son saltos incondicionales, 1000 de carga y el resto son almacenamiento y de tipo R. ¿Cual es su CPI si lo vamos a ejecutar en un procesador tipo MIPS?. Tomando los números del procesador MIPS que hemos visto anteriormente cuando definíamos el CPI, calculamos el porcentaje total y multiplicamos por el CPI de cada tipo de instrucción:

$$CPI = \frac{200}{3400} \times 3 + \frac{300}{3400} \times 3 + \frac{1000}{3400} \times 5 + \frac{1900}{3400} \times 4 = 4,1468$$

   4,1468 ciclos por instrucción es el CPI de este programa. Otra opción es calcular el número de ciclos totales del programa:

$$Ciclos = 200 \times 3 + 300 \times 3 + 1000 \times 5 + 1900 \times 4 = 14100$$

$$CPI = \frac{14100}{3400} = 4,1470$$

   Los redondeos hacen que los CPIs no coincidan exactamente.

2. ¿Qué tiempo de ejecución tendría el programa anterior en un procesador de 5GHz?:

$$TE = 3400 \times 4,1468 \times 0,000000001 = 0,000014099\, segundos$$

3. Atendiendo al tiempo de ejecución anterior ¿Cuantos son los MIPS para esta máquina?. En este caso, podemos hacer una regla de tres, si en 0,000014099 ejecuta 3400 instrucciones, en un segundo ejecutará 241.149.802 instrucciones por lo que 241 MPIS.

Eficiencia energética

El consumo de energía de un computador con una arquitectura determinada nos da una medida de su eficiencia en la realización de una tarea determinada con respecto a su eficiencia energética. En los grandes computadores una medida de esta eficiencia que se está haciendo popular es el **MFLOP/watt** que no es sino el número de operaciones en punto flotante que ejecuta un computador por watio consumido.

Al ser la energía uno de los principales costes recurrentes de los centros de cálculo (entre la energía consumida por los servidores y

la refrigeración), esta métrica está tomando cada vez mas relevancia desde el punto de vista operativo en este tipo de instalaciones.

La potencia consumida por una CPU está directamente relacionada con su frecuencia de reloj y su voltaje de trabajo. A menor frecuencia, menor consumo y menores prestaciones. A mayor frecuencia mas cargas y descargas de los elementos que forman parte de la arquitectura y mas potencia consumida con las necesidades de disipación de calor asociadas.

$$Potencia\,Consumida = C \times V \times F^2$$

Donde C es la capacitancia del procesador (una constante).

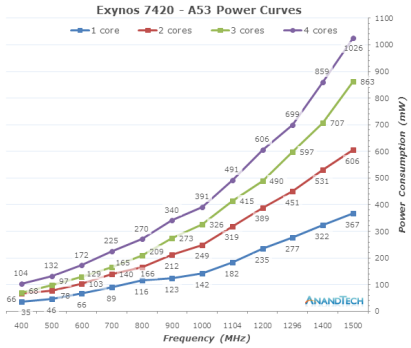

Figura 5: Consumo en función de la frecuencia (Fuente: AnandTech)

Midiendo con la herramienta perf

Vamos a ver cómo podemos extraer algunas métricas de nuestro computador en la ejecución de determinados programas mediante la herramienta *perf*[4]. Existen otras herramientas similares para cualquier sistema operativo, es importante que añadas este tipo de programas a tu *caja de herramientas* de ingeniero de computadores ya que te servirán para:

[4] Documentación de la herramienta perf

- Ver las prestaciones de tus programas y estudiar posibles puntos a mejorar. Cuenta los eventos del kernel por lo que sus posibilidades son infinitas.
- Ver problemas de rendimiento en un determinado programa/-computador.
- Evaluar/comparar computadores en base a unas cargas de trabajo predeterminadas.

Los subcomandos de perf como stat, trace, record, top, etc. te dan una flexibilidad enorme a la hora de hacer *profiling*, busca un buen *cheat sheet* para hacerte una idea de sus posibilidades y añadelo a tu caja de herramientas

Empezemos por un sencillo programa en C que guardaremos en un archivo ejemplo.c

```
#include <stdlib.h>
#include <time.h>
```

```
#include <stdio.h>

#define SIZE 1000
int matriz1[SIZE][SIZE];
int matriz2[SIZE][SIZE];
long int matriz3[SIZE][SIZE];

int main(){

    long int suma=0;
    int i,x,k=0;
    srand(time(NULL));
    for(i = 0; i < SIZE; i++)
        for (x = 0; x < SIZE; x++){
            int z = rand()%10;
            matriz1[i][x]=z;
            matriz2[x][i]=rand()%10;
    }

    for (i = 0; i < SIZE; i++) {
      for (x = 0; x < SIZE; x++) {
        for (k = 0; k < SIZE; k++) {
          suma = suma + matriz1[i][k]*matriz2[k][x];
        }
        matriz3[i][x] = suma;
        suma = 0;
      }
    }
}
```

Prueba a cambiar el tamaño de las matrices para variar los resultados

Como podemos observar, se trata de operaciones con matrices, una de las estructuras mas habituales en todo tipo de operaciones (grafos, imágenes, vectores, etc.). Si lo compilamos obtendremos un archivo binario, que ejecutaremos desde la herramienta perf con el comando *perf stat binario* (con privilegios de superusuario):

```
# gcc -O1 ejemplo.c -o ejemploO1
# perf stat ./ejemploO1
Performance counter stats for './ejemploO1':

          1.036,01 msec task-clock                #    0,998 CPUs utilized
                55      context-switches          #   53,089 M/sec
                 0      cpu-migrations            #    0,000 K/sec
             3.957      page-faults               # 3819,498 M/sec
     3.790.317.334      cycles                    # 3658607,465 GHz
     8.155.440.049      instructions              #    2,15  insn per cycle
     1.039.511.042      branches                  # 1003389036,680 M/sec
         1.112.540      branch-misses             #    0,11% of all branches

       1,038211903 seconds time elapsed

       1,032677000 seconds user
       0,004002000 seconds sys

# gcc -O3 ejemplo.c -o ejemploO3
# perf stat ./ejemploO3

Performance counter stats for './ejemploO3':

            593,47 msec task-clock                #    1,000 CPUs utilized
                 2      context-switches          #    3,373 M/sec
                 0      cpu-migrations            #    0,000 K/sec
             3.957      page-faults               # 6672,850 M/sec
     2.177.437.451      cycles                    # 3671901,266 GHz
     5.649.783.504      instructions              #    2,59  insn per cycle
       287.625.122      branches                  # 485033932,546 M/sec
           353.500      branch-misses             #    0,12% of all branches

       0,593740991 seconds time elapsed

       0,589804000 seconds user
       0,004012000 seconds sys
```

Estas medidas se pueden ver afectadas por muchos factores por lo que es conveniente tratar de repetirlas varias veces y obtendremos los valores medios algo mas fiables (en perf, con la opción -r n ejecuta n veces el binario y devuelve la media de las medidas).

Como puedes ver, hemos compilado el programa con el mismo compilador (gcc) pero con distintas opciones de optimización. Fíjate en el número de instrucciones, el programa es el mismo y las optimizaciones del compilador pueden hacer que un mismo programa tenga una cantidad considerable menos de instrucciones ejecutadas.

*Bancos de pruebas (*Benchmark*)*

Cuando vamos a elegir un procesador, comparar dos procesadores o ver si un procesador con una modificación es mejor que sin esa modificación, tenemos que medir y comparar en bases a las métricas anteriormente descritas. El proceso de las medidas debe ser realizado con rigurosidad para que sean validas. Como ya hemos dicho, el tiempo de ejecución es la medida de rendimiento por excelencia en cuanto a la rapidez de una arquitectura para realizar una tarea específica. Lo que ocurre es que nos surge una pregunta inmediata ¿De qué tarea mido el tiempo de ejecución?.

Idealmente, debería coger los programas que voy a ejecutar en el procesador y medir su tiempo de ejecución y/o rendimiento. Esto es complicado de realizar correctamente. Desde hace tiempo se crean bancos de pruebas, es decir conjuntos de programas que representan tareas habituales para ser utilizados en medir prestaciones y comparar.

Uno de los *Benchmarks* mas conocidos son los SPEC[5] (*Standard Performance Evaluation Corporation*), que crean conjunto de programas tipo para medir las prestaciones de arquitecturas, tareas específicas, servicios software, etc. como la CPU, almacenamiento, potencia, virtualización, plataforma java, etc.

[5] www.spec.org

Algunas empresas mandan los resultados obtenidos por sus productos para publicarlos[6].

[6] Resultados SPEC CPU2017 primer trimestre 2019

Ten cuidado a la hora de tomar como referencia los resultados publicados por los vendedores ya que los resultados de los *Benchmarks* van a depender del compilador, el sistema operativo, la carga del sistema, etc.

Comparando arquitecturas

La comparativa entre dos arquitecturas debe hacerse con sumo cuidado tanto en la métrica utilizada como en la forma en la cual la medida es tomada en cada máquina. Si queremos saber qué arquitectura es mejor ejecutando una tarea o banco de pruebas X, debemos tratar de homogeneizar la forma en la cual medimos las prestaciones de cada arquitectura en la ejecución de esa tarea X, por ejemplo usar el mismo compilador y la misma configuración de compilación, tratar de que otras tareas no influyan en la medida, etc. Si tomamos el tiempo de ejecución para dos arquitecturas A y B ejecutando una misma tarea X:

$$\frac{TE_A}{TE_B} = n \tag{2}$$

En este caso diremos que la arquitectura A es n veces mas rápida

que la arquitectura B. Obviamente si $n < 1$ la arquitectura A tiene un tiempo de ejecución mas bajo y por lo tanto es mas rápida que la arquitectura B mientras que si $n > 1$ la arquitectura A es mas lenta al tener un tiempo de ejecución mayor.

En rendimiento, tomando las inversas de los tiempos de ejecución:

$$\frac{\frac{1}{TE_A}}{\frac{1}{TE_B}} = \frac{Rendimiento_B}{Rendimiento_A} = n \tag{3}$$

En este caso B es mejor n veces que A (si $n > 1$).

A menudo se toma una arquitectura como referencia (arquitectura C) bien parametrizada y con el rendimiento basado en unos bancos de prueba estándar y a continuación se comparan ambas arquitecturas con dicha arquitectura de referencia.

Ley de Amdahl's

La ley de Amdahl nos compara dos ejecuciones de una misma tarea para comparar una mejora que hagamos en una misma arquitectura comparando el tiempo de ejecución antes y despues de la modificación/mejora.

$$Mejora = \frac{TE_{old}}{TE_{new}} \tag{4}$$

donde TE_{old} es el tiempo de ejecución original o *old* (antiguo) y TE_{new} es el tiempo de ejecución nuevo con la modificación realizada.

Existe otra forma de estimar esta mejora si sabemos a qué porcentaje de ejecución hemos afectado y cómo hemos mejorado ese porcentaje. Si denominamos G(m) a esa ganancia parcial o aceleración(*speedup*):

$$Mejora = \frac{1}{(1 - F_{mejorada}) + \frac{F_{mejorada}}{G(m)}} \tag{5}$$

donde $F_{mejorada}$ es la fracción de tiempo que se utiliza el componente al cual se aplica la mejora en la máquina original y G(m) es el ratio de mejora que hemos obtenido al modificar/cambiar el componente.

Hagamos un pequeño ejercicio:

En una arquitectura tipo RISC hemos detectado un fallo de implementación en el VHDL y lo solucionamos. Con esta modificación, la ALU reduce su tiempo de ejecución a la mitad. Según los BENCHMARK aplicados a los procesadores creados con esta arquitectura, esa ALU se utiliza un 25 % del tiempo. ¿Qué mejora hemos conseguido?:

$F_{mejorada} = 0{,}25$

$G(m) = 2$

$$Mejora = \frac{1}{(1 - F_{mejorada}) + \frac{F_{mejorada}}{G(m)}} = \frac{1}{(1 - 0{,}25) + \frac{0{,}25}{2}} = \frac{1}{0{,}875} = 1{,}14$$

La arquitectura con el fallo arreglado es un 1.14 mas rápida que la antigua arquitectura.

Enlaces de referencia

Puedes consultar las prestaciones de las nuevas CPUs (y GPUs, SSD, RAM, etc.) en varios sitios, por poner algunos:

- https://cpu.userbenchmark.com/
- https://www.cpubenchmark.net/

El blog de Daniel López Azaña es una buena introducción a conceptos básicos actuales:

- CPU física, CPU lógica, core, etc.