Capítulo 1

Introducción a los computadores

1.1. Introducción

1.1.1. Unidades funcionales básicas

Los elementos principales de un computador son: la unidad de procesamiento central (CPU) o procesador, la memoria, el subsistema de entrada/salida y algunos medios de interconexión de todos estos componentes. La CPU está formada por una unidad de control y un camino de datos, que por su parte consta de registros internos e interconexiones.

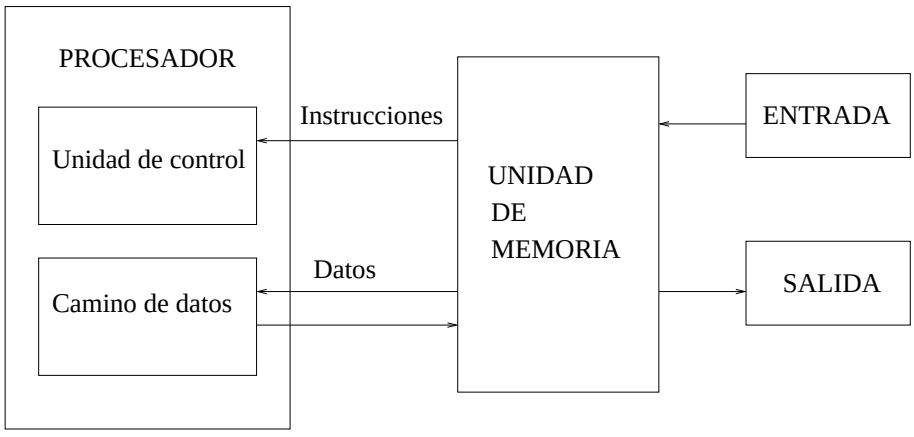


Figura 1.1: El computador: unidades funcionales básicas

- La unidad central de procesamiento (CPU, *Central Processing Unit*): controla el funcionamiento del computador y lleva a cabo sus funciones de procesamien-

to de datos. Frecuentemente se llama simplemente procesador. Sus principales componentes estructurales son:

- Unidad de control: controla el funcionamiento de la CPU y, por tanto, del computador.
- Camino de datos:
  - Unidad aritmético-lógica (ALU, Arithmetic Logic Unit): Lleva a cabo las funciones de procesamiento de datos del computador.
  - Registros: proporcionan almacenamiento interno a la CPU.
  - Interconexiones CPU: son mecanismos que proporcionan comunicación entre la unidad de control, la ALU y los registros.

- La memoria: almacena datos e instrucciones.
- E/S: transfiere datos entre el computador y el entorno externo.
- Sistema de interconexión: es un mecanismo que proporciona la comunicación entre la CPU, la memoria principal y la E/S.

1.1.2. Organización estructural de un computador

Desde el punto de vista estructural, el computador se considera dividido en varios niveles organizados jerárquicamente. La Figura 1.2 muestra la organización del computador según estos niveles. Pueden apreciarse 8 subniveles divididos en dos grandes zonas: una zona inferior donde los niveles son de naturaleza hardware y una zona superior de naturaleza software.

En el proceso de diseño de un computador se empieza por los niveles inferiores moviéndonos hacia arriba, ya que un nivel determinado utiliza los elementos diseñados en el nivel inmediatamente inferior. A la hora de realizar el análisis funcional de la máquina se sigue la dirección contraria. Se comienza analizando los elementos más complejos para ir subdividiendo sucesivamente el computador en sus elementos más básicos.

A continuación se describen los 8 subniveles empezando por abajo y en el sentido de ir aumentando paulatinamente la complejidad.

El primer nivel está constituido por los componentes. Las leyes aplicadas son las de la electrónica física. Los componentes son semiconductores de tipo n y p, metales, etc. Los sistemas construidos son diodos, transistores, resistencias, condensadores, etc.

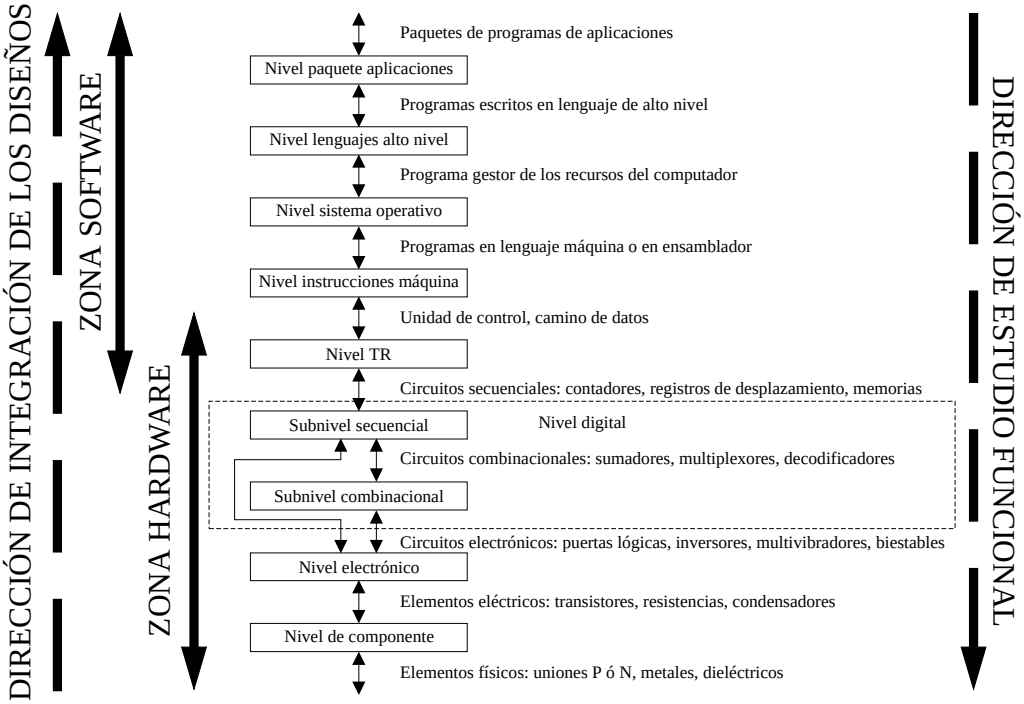


Figura 1.2: Organización estructural de un computador

El segundo nivel es el de circuito electrónico. Se construyen sistemas como las puertas lógicas, los biestables, los osciladores, etc., a partir de los componentes.

El tercer nivel es el de circuito digital. Incluyendo tanto a los dispositivos combinacionales como a los secuenciales.

El siguiente nivel es el de TR o Transferencia entre Registros. En este caso se estudia el computador centrándose en el flujo de información que se envía de un registro a otro, pasando, en su caso, por el correspondiente circuito combinacional que lo encamina o lo transforma. Los elementos constructivos de este nivel son los buses, los registros, los bloques combinacionales, las memorias, etc. Y en el se estudia el camino de datos y el control. El funcionamiento de la unidad de control puede definirse mediante hardware (control cableado), o mediante software (control microprogramado). Por eso en la figura aparece una superposición entre las dos grandes zonas de software y hardware. Este nivel actúa además como frontera entre lo que es circuitería electrónica y lo que es programación.

El siguiente nivel es el de instrucciones máquina. En este nivel se construyen programas en lenguaje máquina, que es el lenguaje capaz de entender la CPU. También se incluye en este nivel la programación en ensamblador. Estos dos últimos niveles son los cubiertos por las asignaturas Estructura de Computadores I y II.

El sexto nivel es el del sistema operativo. Los sistemas operativos son un conjunto de programas que ayudan al usuario en la explotación del computador, siendo por tanto, una capa software con la que se rodea el hardware para facilitar su utilización.

El séptimo nivel es el de los programas en lenguaje de alto nivel (C, FORTRAN, Pascal, etc.). Estos programas deberán pasar por un proceso intermedio de traducción, denominado compilación, antes de ejecutarse. El objetivo de la compilación es traducir el lenguaje de alto nivel a un lenguaje de bajo nivel que la máquina pueda entender. El resultado de dicha traducción recibe el nombre de código objeto.

Finalmente, en el último nivel se encuentran las aplicaciones. Este nivel está formado por paquetes de programas de aplicación que sirven para resolver problemas en campos específicos de la ciencia o gestión.

1.1.3. Evolución del software

La forma más sencilla de hablar a una máquina electrónica es utilizando dos señales eléctricas: encendido y apagado. Por lo tanto el alfabeto de la máquina tiene sólo dos letras, 0 y 1, o número binarios. Los primeros programadores se comunicaban con las máquinas utilizando única y exclusivamente 0's y 1's (**lenguaje máquina**). Pronto se inventaron notaciones simbólicas y programas para traducir dichas notaciones simbólicas a 0's y 1's. El primero de estos programas fue llamado **ensamblador**. Este programa traduce la versión simbólica de una instrucción a su notación binaria. Por ejemplo:

$$\text{add A,B} \rightarrow 1000110010100000$$

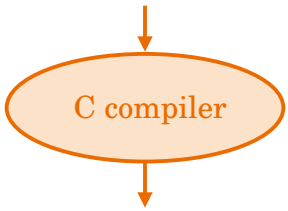
El siguiente paso fue la construcción de los **compiladores**, es decir, programas que permitiesen la traducción de un lenguaje de alto nivel más natural y fácil de leer a lenguaje ensamblador. La Figura 1.3 muestra la relación entre estos programas y lenguajes.

A medida que la programación se desarrollaba los programadores se dieron cuenta de que muchos trozos de código se reutilizaban en gran parte de sus programas. Se crearon entonces las **librerías** para contener aquellas subrutinas potencialmente aplicables a muchas circunstancias. Una de las primeras librerías fue para gestionar la E/S de datos, e incluía por ejemplo rutinas para controlar impresoras.

Pronto saltó a la vista que un conjunto de programas podía ejecutarse más eficientemente si había un programa independiente que supervisara la ejecución de esos programas de forma que se evitasen esperas improductivas. Estos programas supervisores, que incluyeron las librerías de E/S fueron las base de los llamados **sistemas operativos**. Los sistemas operativos son programas que gestionan los

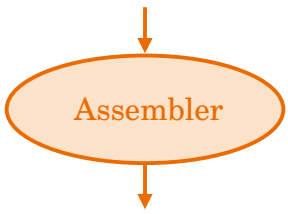
High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```



Assembly language program (for MIPS)

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```



Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Figura 1.3: Programa C compilado a lenguaje ensamblador y después ensamblado a lenguaje máquina binario

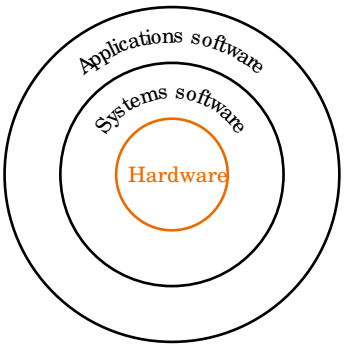


Figura 1.4: Visión simplificada de la circuitería y la programación como niveles jerárquicos

recursos de un computador.

Los programas software pasaron entonces a catalogarse según su uso:

- programas del sistema: programas orientados a los programadores, tales como sistemas operativos, compiladores y ensambladores.
- programas de aplicaciones o simplemente aplicaciones: programas orientados al usuario, tales como hojas de cálculo, editores, etc.

La Figura 1.4 muestra el clásico diseño con los niveles jerárquicos de la circuitería y la programación.

1.1.4. Evolución del hardware

La evolución de los computadores se ha caracterizado por un incremento de la velocidad del procesador, una disminución del tamaño de los componentes, un aumento del tamaño de la memoria y un aumento de la capacidad de E/S y de la velocidad.

La Figura 1.5 muestra el incremento de la capacidad de las DRAM desde 1977. Prácticamente se cuadriplica su capacidad cada 4 años.

Otro factor responsable del gran aumento de la velocidad del procesador es la disminución del tamaño de los componentes del microprocesador; esto reduce la distancia entre componentes y, por tanto, aumenta la velocidad. La Tabla 1.1 muestra las tecnologías que se han ido usando a lo largo del tiempo, con una estimación relativa del rendimiento por unidad de coste para cada una de ellas.

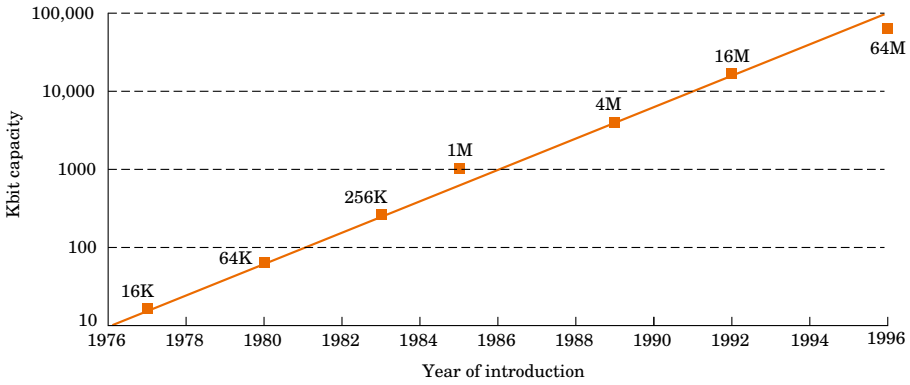


Figura 1.5: Crecimiento de la capacidad de los chips de DRAM a lo largo del tiempo

Año	Tecnología	Relación rendimiento/coste
1951	Válvula de vacío	1
1965	Transistor	35
1975	Circuito integrado	900
1995	VLSI	24000000

Cuadro 1.1: Rendimiento relativo por unidad de coste de las tecnologías utilizadas en los ordenadores a lo largo del tiempo

La Figura 1.6 muestra la evolución en el número de transistores que se puede colocar en un chip desde la década de los 70. La rapidez del progreso tecnológico puede modelarse de acuerdo con una observación llamada **ley de Moore**: el número de transistores en un chip se duplica cada 18 meses, lo que equivale a un incremento del 60 % en el número de transistores al año. Esta ley ha estado inicialmente asociada a los chips de memoria, pero es igualmente aplicable a los chips de CPU.

Sin embargo, la verdadera ganancia en velocidad en los últimos años se debe a la organización del procesador, incluyendo un fuerte uso del encauzamiento y de las técnicas de ejecución paralela, y del uso de técnicas de ejecución especulativa. Todas estas técnicas tienen como objetivo mantener al procesador ocupado el mayor tiempo posible, sacándole por tanto su máximo rendimiento.

Otro asunto crítico en el diseño de computadores, es hacer un balance de las prestaciones de los distintos elementos, de forma que esta ganancia en prestaciones en un área no perjudique a otras áreas. En particular, la velocidad del procesador ha aumentado más rápidamente que el tiempo de acceso a memoria (ver Figura 1.7). Se han usado distintas técnicas para compensar este desacople, incluyendo memorias cache, caminos de datos más anchos de la memoria del procesador, y más circuitos

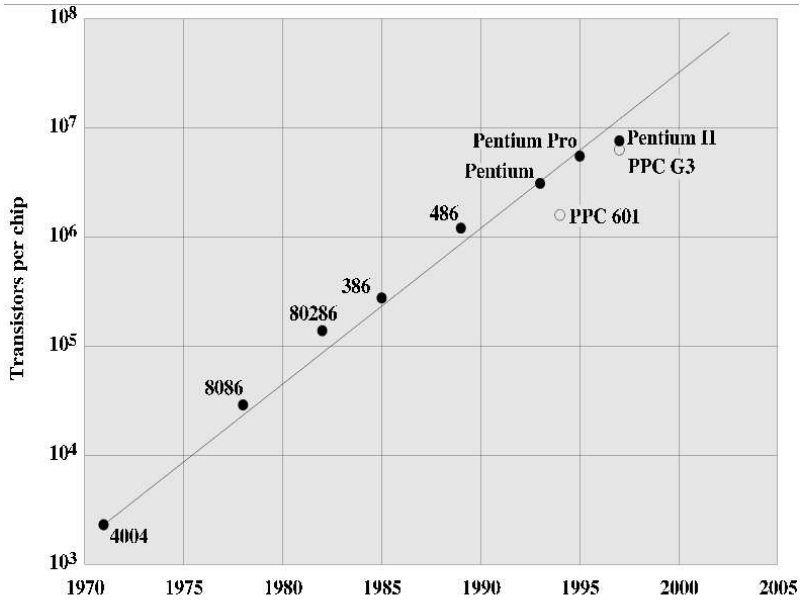


Figura 1.6: Incremento en el número de transistores que se pueden colocar en un chip

de memoria inteligentes.

1.2. Perspectiva histórica

En esta sección se revisan brevemente los sucesos históricos clave en la evolución de las computadoras.

1.2.1. La generación cero: Computadoras mecánicas (1642-1945)

La primera persona que construyó una máquina calculadora funcional fue el científico francés Blaise Pascal (1623-1642), en cuyo honor se nombró el lenguaje de programación Pascal. Su máquina calculadora sólo podía sumar y restar y funcionaba de forma totalmente mecánica.

No sucedió nada interesante en la historia de los computadores hasta que 150 años después Charles Babbage (1792-1871) construyó la llamada **máquina analítica**. La gran novedad de esta máquina era su estructura. Tenía 4 componentes: el

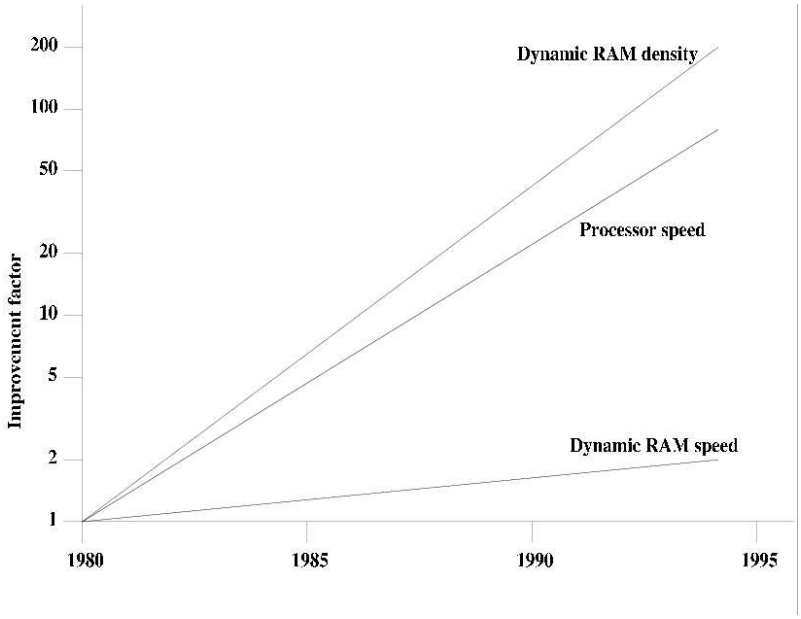


Figura 1.7: Evolución de las características de DRAM y del procesador

almacén (memoria), el molino (unidad de cómputo), la sección de entrada (lector de tarjetas perforadas) y la sección de salida (salidas perforadas e impresas). Era totalmente mecánica, no obstante su estructura es básicamente la misma que la de las computadoras digitales modernas.

El siguiente avance importante ocurrió a finales de la década de los treinta, cuando un estudiante de ingeniería alemán llamado Konrad Zuse construyó una serie de máquinas calculadoras automáticas empleando relevadores electromagnéticos.

1.2.2. La primera generación: Válvulas de vacío (1945-1955)

El estímulo para la construcción de computadoras electrónicas fue la segunda guerra mundial. Se necesitaban máquinas capaces de descifrar los mensajes en clave de los enemigos, así como el cálculo de tablas de alcance para disparar la artillería. John Mauchley y su estudiante de postrado J. Eckert (Univ. de Pennsylvania) aprovecharon la ocasión y pidieron una subvención al gobierno para construir una computadora electrónica. La propuesta fue aceptada en 1943 e iniciaron la construcción del ENIAC (*Electronic Numerical Integrator and Computer*), primer computador electrónico de propósito general

La máquina quedó finalizada en 1946, demasiado tarde para la guerra. En tamaño, el ENIAC era dos ordenes de magnitud más grande que las máquinas actuales.

Era una máquina decimal, es decir, los números estaban representados en decimal, y la aritmética utilizada era decimal. La forma de programarla (con un gran número de interruptores) era lenta y tediosa.

El proceso de programación podría ser más fácil si el programa se representase en una forma adecuada para ser guardado en la memoria, junto con los datos (codificación binaria). Esta idea, conocida como *concepto de programa almacenado* se atribuye a J. von Newmann, uno de los asesores del proyecto ENIAC. Todos los computadores actuales se construyen en base a esta idea:

- Las instrucciones se representan como números
- Los programas pueden almacenarse en memoria para ser leídos o escritos igual que números

En 1946, von Newmann y sus colegas empezaron a construir en la Univ. de Princeton un computador con el concepto de programa almacenado (no completado hasta 1952). Este computador tenía la estructura básica de todas las computadoras actuales de propósito general: poseía una unidad de memoria, una unidad aritmético-lógica, una unidad de control y una unidad de entrada/salida.

Aunque en realidad, la primera computadora construida con el concepto de programa almacenado fue la EDSAC (*Electronic Delay Storage Automatic Calculator*), construida por Maurice Wilkes en 1949 en la Univ. de Cambridge.

En 1947 Eckert y Mauchley formaron una compañía para manufacturar computadores comerciales. El UNIVAC I comercializado por dicha compañía fue el primer ordenador comercial de éxito (se construyeron 48 unidades!!).

Por esas fechas IBM ya se encontraba en el mercado de los computadores, pero se dedicaba básicamente al negocio de las tarjetas perforadas. No empezó a fabricar ordenadores hasta la década de los 50.

1.2.3. La segunda generación: Transistores (1955-1965)

El transistor fue inventado en los laboratorios Bell en 1948 por John Bardeen, Walter Brattain y William Shockley, quienes fueron galardonados con el Nobel de física por su trabajo en 1956.

En un plazo de 10 años el transistor revolucionó el mundo de los computadores. La primera computadora transistorizada se construyó en el MIT y se llamó TX-0 (computadora Transistorizada eXperimental 0). No fue una máquina de mucho éxito y tuvieron que pasar varios años hasta que en 1960 DEC (Digital Equipment

Corporation) presenta la PDP-1. DEC vendió docenas de estas máquinas, y con ella comenzó la industria de las minicomputadoras. En esa época varias compañías se subieron al carro aportando diferentes soluciones. Entre ellas IBM.

En 1964 una nueva compañía llamada CDC (*Control Data Corporation*) presenta su máquina CDC6600. Esta máquina era tremendamente rápida y puede ser considerada como el primer supercomputador. El secreto de su velocidad era que dentro de la CPU había una máquina con un alto grado de paralelismo: contaba con varias unidades funcionales para sumar, multiplicar y dividir, y todas podían operar en paralelo. Muchas de las ideas fundamentales en las que se basan las computadoras modernas se remontan directamente a la 6600. El diseñador de la 6600 fue Seymour Cray, el cual dedicó toda su vida a construir supercomputadoras entre las que cabe destacar el CRAY-1 (1976). Esta máquina fue simultáneamente la más rápida del mundo, la más cara y el computador con la mejor relación coste/rendimiento para programas científicos.

1.2.4. La tercera generación: circuitos integrados (1965-1980)

La invención del circuito integrado de silicio por Robert Noyce en 1958 hizo posible colocar docenas de transistores en un sólo chip. Este empaquetamiento permitió construir computadoras más pequeñas, más rápidas y menos costosas.

Dentro de esta era destaca la familia de ordenadores IBM Sistema/360 y el DEC PDP-8. El Sistema/360 fue la primera familia de computadores de la historia. La familia abarcaba un amplio rango de prestaciones y precios. Los distintos modelos eran compatibles, un programa escrito para un modelo podía ser ejecutado en cualquier otro modelo de la familia siendo la única diferencia el tiempo de ejecución. La idea de las familias de máquinas de inmediato fue bien recibida, y en unos cuantos años casi todos los fabricantes de computadoras tenían una familia de máquinas similares.

En el mismo año que IBM lanza su familia DEC presenta el PDP-8. El PDP-8 fue el primer miniordenador comercial. Era lo bastante pequeño para ser colocado en lo alto de una mesa de laboratorio o embutido en otro equipo. Fue un éxito inmediato.

1.2.5. La cuarta generación: integración a muy alta escala (1980-?)

Más allá de la tercera generación hay menos acuerdo general en la definición de las generaciones. Una posible clasificación es tomar como cuarta y última generación

la época a partir de los años 80. En los años 80 la VLSI (integración a muy alta escala, *Very Large Scale Integration*) había hecho posible colocar primero decenas, luego miles y finalmente millones de transistores en un único chip. Este avance dio pie a computadoras más pequeñas, más rápidas y más baratas. En los años 80 los precios habían bajado tanto que una persona podía tener su propia computadora. Se había iniciado la era de la computadora personal.

En estas dos últimas décadas diferentes compañías han ocupado el primer puesto en ventas y popularidad. En los años 80 la compañía líder de mercado era IBM. IBM se metió en el mercado de los computadores personales utilizando componentes fabricados por otras compañías, en lugar de hacer una máquina partiendo de cero. En 1981 introdujo en el mercado la IBM Personal Computer, construida con un procesador de la casa Intel, en concreto, el 8088. Se convirtió de inmediato en la máquina más vendida de la historia. La versión inicial de la IBM PC venía equipada con el sistema operativo MS-DOS provisto por la entonces pequeña compañía Microsoft Corporation.

Para mediados de los 80 una nueva idea llamada RISC comenzó a dominar, reemplazando arquitecturas complejas (CISC) por otras más sencillas pero más rápidas.

En los 90 comenzaron a aparecer las CPU superescalares. Estas máquinas podían ejecutar varias instrucciones al mismo tiempo, a menudo en un orden diferente al que tenían en el programa.

Año tras año el precio de los computadores continúa cayendo, mientras que las prestaciones y la capacidad de estos sistemas sigue creciendo. El aumento en velocidad y prestaciones se consigue gracias a mejoras en el diseño de la arquitectura de la máquina, más que a un cambio de la tecnología existente. Sin una tecnología revolucionaria en el horizonte, no está claro cuando va a aparecer una quinta generación.

En la Tabla 1.2 mostramos un resumen de las diferentes generaciones de ordenadores. La clasificación realizada se basa en la tecnología utilizada en cada generación.

1.3. Rendimiento

A la hora de comprar un equipo informático el usuario se interesa fundamentalmente por dos parámetros: su coste y su rendimiento. En esta sección nos vamos a centrar en el segundo de los aspectos, el rendimiento.

Generación	Fechas	Tecnologías	Nuevo producto principal
1	1945-55	Válvulas de vacío	Ordenador electrónico comercial
2	1955-65	Transistores	Ordenadores más baratos
3	1965-80	Circuitos integrados	Minicomputadores
4	1980-?	Circuitos LSI/VLSI	Ordenadores personales y multiprocesadores

Cuadro 1.2: Generaciones de ordenadores

1.3.1. Medidas del rendimiento

Cada componente de un equipo informático refleja sus prestaciones atendiendo a unos parámetros característicos:

- Memorias: los dos parámetros fundamentales que influyen en sus prestaciones son su tamaño (en número de palabras) y su latencia o tiempo de respuesta (normalmente dado en ns). El tamaño aumenta la funcionalidad del sistema, mientras que su latencia está más ligada a la rapidez de ejecución de los programas. Desafortunadamente estos dos parámetros son inversamente proporcionales.

- Buses: se caracterizan por el número de líneas de comunicación y por la frecuencia de transmisión de los datos (normalmente dado en MHz). En este caso ambos parámetros se funden en una única métrica, el ancho de banda, que indica la velocidad de comunicación por el bus (normalmente expresado en Megabytes por segundo). Por ejemplo, el bus local PCI del Pentium tiene una anchura de 64 líneas y una frecuencia de 66 MHz, lo que proporciona un ancho de banda de 528 Mbytes/s.

- El procesador: su rendimiento vendrá caracterizado por el tiempo de CPU, es decir, el tiempo que la CPU dedica a la ejecución de una tarea concreta y no incluye el tiempo perdido en actividades de E/S o en la ejecución de otros programas (si es que el computador es de tiempo compartido).

Todo el sistema en conjunto puede ser evaluado a través del tiempo que tarda en ejecutar una tarea, o a través del número de tareas que es capaz de ejecutar en un determinado tiempo. Si se estuviera ejecutando un programa en dos estaciones de trabajo diferentes, se podría decir que la más rápida es la que acaba el trabajo primero. En cambio, si se estuviese dirigiendo un centro de computación compuesto

por varias máquinas utilizadas por diferentes usuarios en tiempo compartido, se diría que se obtiene un mayor rendimiento cuanto mayor sea el número de trabajos que se completen a lo largo del día. Distinguimos por tanto dos medidas diferentes del rendimiento:

- Tiempo de respuesta: Tiempo transcurrido entre el comienzo y el final de un evento, denominado también tiempo de ejecución o latencia.
- Productividad: cantidad total de trabajo realizada en un determinado tiempo, denominado también *throughput* o ancho de banda.

Los usuarios individuales están más interesados en reducir el tiempo de respuesta, mientras que los directivos de un centro de computación están habitualmente interesados en incrementar la productividad.

Evidentemente en el tiempo de respuesta estará influyendo la velocidad de su CPU, de su memoria y de sus buses, y todo ello en diferente medida dependiendo de las características del programa que se esté ejecutando. Además la velocidad de la CPU también dependerá del tipo de aplicación, ya que su diseño puede estar muy optimizado para determinado tipo de código y menos optimizado para otros. Por ello medir el rendimiento de una máquina de forma genérica no es tarea fácil, de hecho, para diferentes tipos de aplicaciones, diferentes aspectos de un computador podrían ser los más significativos.

Nosotros en este curso nos vamos a centrar en el estudio del procesador, por ello a la hora de medir el rendimiento nos ocuparemos principalmente del tiempo de ejecución. Para maximizar el rendimiento lo que se desea es minimizar el tiempo de ejecución:

$$Rendimiento = \frac{1}{Tiempo\ de\ ejecucion}$$

El tiempo de ejecución de un programa se mide es segundos por programa. Pero el tiempo puede ser definido de diferentes maneras dependiendo de lo que se cuente. La definición más sencilla es el tiempo de reloj (*wall clock time*), tiempo de respuesta (*response time*) o tiempo transcurrido (*elapsed time*): tiempo total que tarda una tarea en completarse, incluye accesos a disco, accesos a memoria, actividades de E/S y la carga adicional introducida por el sistema operativo. Por otro lado el tiempo de ejecución de CPU o simplemente el tiempo de CPU es el tiempo que la CPU dedica a ejecutar una tarea y no incluye la E/S (la CPU trabaja en otro programa mientras espera la E/S) o el tiempo dedicado en la ejecución de otros programas. Además, el tiempo de CPU puede ser dividido en tiempo de CPU del usuario (tiempo consumido por el programa) y tiempo de CPU del sistema (tiempo de CPU consumido por el sistema operativo).

Casi todos los ordenadores tienen un reloj que funciona a una frecuencia concreta y que determina el momento en el que tienen lugar los sucesos en la circuitería. El inverso de la frecuencia de reloj es el periodo de la señal de reloj. La longitud del periodo de reloj recibe el nombre de ciclo de reloj o simplemente ciclo. El tiempo de ejecución de un programa vendrá dado por:

$$T_{CPU} = Ciclos\ de\ reloj \times Tiempo\ del\ ciclo = \frac{Ciclos\ de\ reloj}{Frecuencia\ de\ reloj}$$

Para aumentar el rendimiento basta con aumentar las frecuencia de reloj o reducir el número de ciclos requeridos para ejecutar un programa. Desafortunadamente ambas magnitudes están relacionadas de forma que muchas de las técnicas que reducen el número de ciclos reducen también la frecuencia del reloj y viceversa.

Evidentemente el número de ciclos ejecutados por un programa va a depender del número de instrucciones que ejecuta dicho programa:

$$Ciclos\ de\ reloj\ de\ CPU = NI \times CPI$$

donde NI es el número de instrucciones máquina en que se transforma el programa, y CPI (*Clock Cycles per Instruction*) es el número medio de ciclos de reloj que se necesita para ejecutar cada instrucción máquina. Como instrucciones diferentes podrían necesitar diferentes cantidades de ciclos dependiendo de lo que hacen, el CPI es una media de todas las instrucciones ejecutadas en el programa.

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i I_i)}{recuento\ de\ instrucciones} = \frac{Ciclos\ de\ reloj\ de\ la\ CPU}{recuento\ de\ instrucciones}$$

Por tanto, el tiempo de ejecución de CPU vendrá dado por:

$$T_{CPU} = NI \times CPI \times T = \frac{NI \times CPI}{F}$$

NI depende fundamentalmente del compilador. Si éste está optimizado para un procesador, será capaz de traducir un lenguaje de alto nivel en un programa ejecutable con el menor número de instrucciones máquina posible. El segundo factor, CPI, está más relacionado con el conjunto de instrucciones del procesador. Depende del diseño del repertorio de instrucciones. En un procesador RISC (*Reduced Instruction Set Computer*) el CPI suele ser menor que en uno CISC (*Complex Instruction Set Computer*), en contrapartida, el número de instrucciones necesarias para ejecutar un determinado programa suele ser mayor en un procesador RISC que en un CISC.

El tercer factor, la frecuencia de reloj del procesador, está más relacionada con la tecnología hardware y con las técnicas de integración de circuitos.

Podemos entonces aumentar el rendimiento aumentando la frecuencia de reloj, o disminuyendo el número de ciclos de reloj requeridos por instrucción, o disminuyendo el número necesario de instrucciones para ejecutar un programa. Desafortunadamente las tres medidas están relacionadas de forma que disminuir una de ellas puede hacer que aumente otra. Se ha de tener siempre en cuenta que la única medida completa y fiable del rendimiento es el tiempo. Cambiar el repertorio de instrucciones para disminuir el cómputo total de las mismas podría llevar a una organización con un ciclo de reloj mayor que contrarreste la mejora. Igualmente, ya que el CPI depende de la mezcla de instrucciones, el código que ejecute el menor número de instrucciones podría no ser el más rápido.

Medidas populares del rendimiento

En la búsqueda de una medida estándar del rendimiento de computadores se han adoptado métricas alternativas al tiempo de CPU. Las más conocidas son los MIPS y los MFLOPS.

- **MIPS (millones de instrucciones por segundo):** Para un programa dado:

$$MIPS = \frac{NI}{T_{CPU} \times 10^6}$$

  Como es inversamente proporcional al tiempo de ejecución, máquinas más rápidas tendrán un mayor MIPS. Su principal inconveniente es que son dependientes del repertorio de instrucciones, a mayor complejidad del repertorio de instrucciones menor será su número, sin que esto quiera decir que el programa se está ejecutando más lentamente. Por tanto no sirve para comparar computadores con diferente número de instrucciones. Además no es una medida única del rendimiento, tendrá diferentes valores para diferentes programas, y finalmente y fundamentalmente, puede variar inversamente al rendimiento.

- **MFLOPS (millones de operaciones en punto flotante por segundo)**: Esta métrica está basada en operaciones en lugar de instrucciones con el objeto de conseguir una independencia de la representación interna del programa fuente y, por tanto, del repertorio de instrucciones del procesador. Su fórmula es:

$$MFLOPS = \frac{Numero\ de\ operaciones\ en\ punto\ flotante\ en\ un\ programa}{T_{CPU} \times 10^6}$$

Una operación en punto flotante es una operación de suma, resta, multiplicación o división aplicada a un número en representación simple o doble precisión en punto flotante (float, double, real, ...).

El inconveniente de los MFLOPS es que no mide el rendimiento del sistema en general, si no que se centra en su unidad de cálculo en punto flotante. Además, los MFLOPS contabilizan por igual una suma que una división, cuando en realidad esta última tarda mucho más en ejecutarse. Sigue siendo dependiente del programa, distintos programas darán distinto número de MFLOPS sobre la misma máquina.

A veces los fabricantes nos ofrecen como medida de rendimiento el *rendimiento pico*, que viene a ser el rendimiento máximo alcanzable por la máquina, es decir, el rendimiento que una máquina tiene garantizado no sobrepasar. Así por ejemplo, los MFLOPS pico de una máquina se corresponderían con la máxima tasa de MFLOPS posible para cualquier segmento de código. Este rendimiento máximo suele estar muy lejos del rendimiento real observado, y no es muy útil como medida del rendimiento.

1.3.2. Programas de evaluación (benchmarks)

Puesto que cada usuario tiene unas necesidades diferentes, es decir, va a ejecutar sobre la máquina diferentes tipos de aplicaciones con diferentes requerimientos, necesitamos obtener una métrica fiable del rendimiento de la máquina. Para ello surgen los conjuntos de programas de evaluación o *benchmarks*. Estos conjuntos estará formado por programas de muy diversas características, tratando de abarcar el amplio espectro de funciones para las que normalmente son usados lo computadores. De esta manera, cada usuario se interesará por los parámetros de rendimiento que el computador consigue en aquellas aplicaciones que mejor representen a las tareas que va a ejecutar él.

Hoy en día, se está ampliamente de acuerdo en que el mejor tipo de programas para usar como pruebas son las aplicaciones reales. Estas podrían ser aplicaciones que el usuario emplea regularmente, o simplemente aplicaciones típicas. Por ejemplo, en un entorno en el que los usuarios son principalmente ingenieros se podría utilizar como conjunto de programas de prueba varias aplicaciones científicas.

El primer programa de evaluación estándar fue creado por SPEC (*Standard Performance Evaluation Corporation*), una organización fundada en 1988 por los principales vendedores de computadores. Podéis encontrar su página oficial en `http://www.specbench.org`. El conjunto de programas SPEC está formado por aplicaciones reales. Estas aplicaciones están clasificadas en varios grupos cuyos objetivos son medir el rendimiento de la máquina desde diferentes puntos de vista. Dentro de

Name	Brief Description
164.gzip	Data compression utility
175.vpr	FPGA circuit placement and routing
176.gcc	C compiler
181.mcf	Minimum cost network flow solver
186.crafty	Chess program
197.parser	Natural language processing
252.eon	Ray tracing
253.perlbmk	Perl
254.gap	Computational group theory
255.vortex	Object-oriented database
256.bzip2	Data compression utility
300.twolf	Place and route simulator

Cuadro 1.3: Los programas de prueba CINT2000

los conjuntos de programas SPEC los más populares son los benchmarks CPU, los cuales tienen como objetivo medir el rendimiento del procesador teniendo en cuenta su memoria y su compilador. La versión más moderna de estos benchmarks es el conjunto CPU2000. Este conjunto se compone internamente de 2 colecciones de programas:

- CINT2000: compuesto de 12 programas de aritmética entera los cuales representan aplicaciones comerciales (ver Tabla 1.3).
- CFP2000, compuesto de 14 programas de aritmética en punto flotante, representando aplicaciones científicas (ver Tabla 1.4).

Las colecciones CINT2000 y CFP2000 se pueden utilizar para medir las siguientes métricas:

- SPECint_base2000: media geométrica de las tasas SPEC para todos los programas de enteros compilando dichos programas con opciones de compilación conservativas.
- SPECfp_base2000: media geométrica de las tasas SPEC para todos los programas CFP2000 compilando dichos programas con opciones de compilación conservativas.
- SPECint2000: media geométrica de las tasas SPEC para todos los programas de enteros cuando dichos programas fueron compilados con opciones de compilación agresivas.

Name	Brief Description
168.wupwise	Quantum chromodynamics
171.swim	Shallow water modeling
172.mgrid	Multi-grid solver in 3D potential field
173.applu	Parabolic/elliptic partial differential equations
177.mesa	3D graphics library
178.galgel	Fluid dynamics: analysis of oscillatory instability
179.art	Neural network simulation: adaptive resonance theory
183.equake	Finite element simulation: earthquake modeling
187.facerec	Computer vision: recognizes faces
188.ammp	Computational chemistry
189.lucas	Number theory: primality testing
191.fma3d	Finite-element crash simulation
200.sixtrack	Particle accelerator model
301.apsi	Solves problems regarding temperature, wind, distribution of pollutants

Cuadro 1.4: Los programas de prueba CFP2000

- SPECfp2000: media geométrica de las tasas SPEC para todos los programas CFP2000 cuando dichos programas fueron compilados con opciones de compilación agresivas.

La tasa SPEC para cada programa se calcula dividiendo el tiempo de ejecución del programa sobre una máquina de referencia, entre el tiempo de ejecución sobre la máquina que estamos evaluando. De esta forma la tasa SPEC es inversamente proporcional al tiempo de ejecución, o lo que es lo mismo, mayor tasa SPEC implica mayor rendimiento. La máquina de referencia se va actualizando a medida que la tecnología avanza al igual que se van actualizando los programas de prueba. Para los programas CPU2000 la máquina de referencia es una SUN Ultra5_10 a 300 MHZ.

Se toman medidas con opciones de compilación conservativas y agresivas para aislar el efecto del compilador de las medidas de rendimiento. Un buen compilador puede hacer que un programa se ejecute más rápidamente. Las máquinas con mejores compiladores pueden así parecer mejores cuando en realidad el mérito no está en la configuración hardware sino en el compilador.

La Figura 1.8 muestra las tasas SPEC para el procesador Pentium III a diferentes

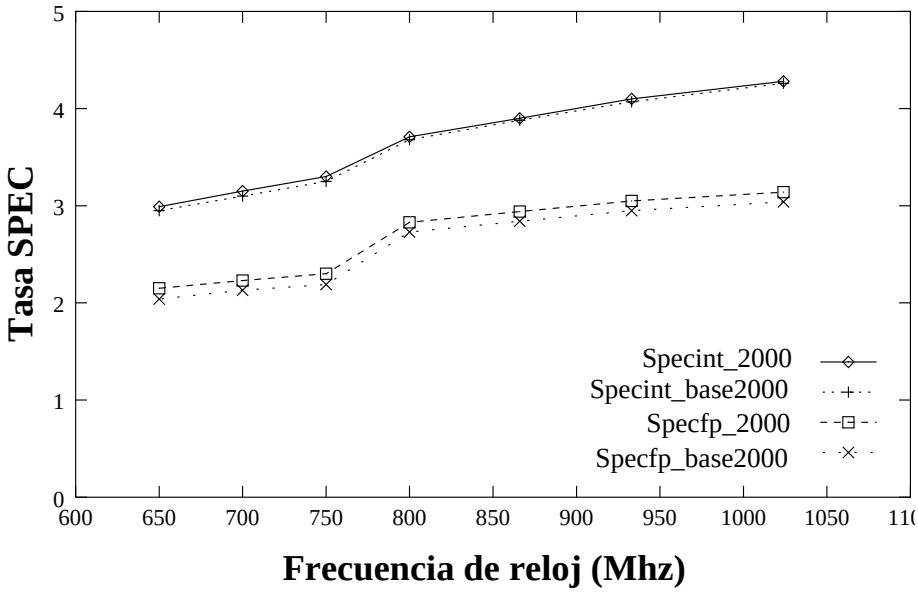


Figura 1.8: Tasas de los SPEC CPU2000 para los procesadores Pentium III

velocidades de reloj. Vemos como el rendimiento del procesador va aumentando a medida que se aumenta la frecuencia de reloj.

Los incrementos en el rendimiento de la CPU para una determinada arquitectura pueden deberse a tres razones: Incrementos en la frecuencia de reloj, mejoras en la organización del procesador (disminución del CPI), mejoras en el compilador. En las Figuras 1.9, 1.10 se muestran las tasa SPEC para los procesadores Pentium y Pentium Pro a diferentes velocidades. Las medidas en este caso han sido tomadas con la versión anterior de los programas SPEC, es decir, con los SPEC CPU95. Se puede observar la mejora del rendimiento del Pentium Pro respecto al Pentium para una misma frecuencia de reloj debido a una mejor organización del procesador.

1.3.3. Ley de Amdahl

A la hora de diseñar una máquina tendremos que intentar maximizar el rendimiento a la vez que se cumplen las restricciones de coste. Para ello es imprescindible conocer que atributos son importantes para la nueva máquina. Puesto que todos los componentes de un computador están interconectados, un cambio en las prestaciones de un subsistema tiene un impacto inmediato en el rendimiento del sistema en general. Ahora bien, puestos a mejorar algo, el sistema se beneficiará más si mejoramos el subsistema donde está el cuello de botella (subsistema o subsistemas que degradan el rendimiento del equipo en general, en los sistemas actuales suele ser

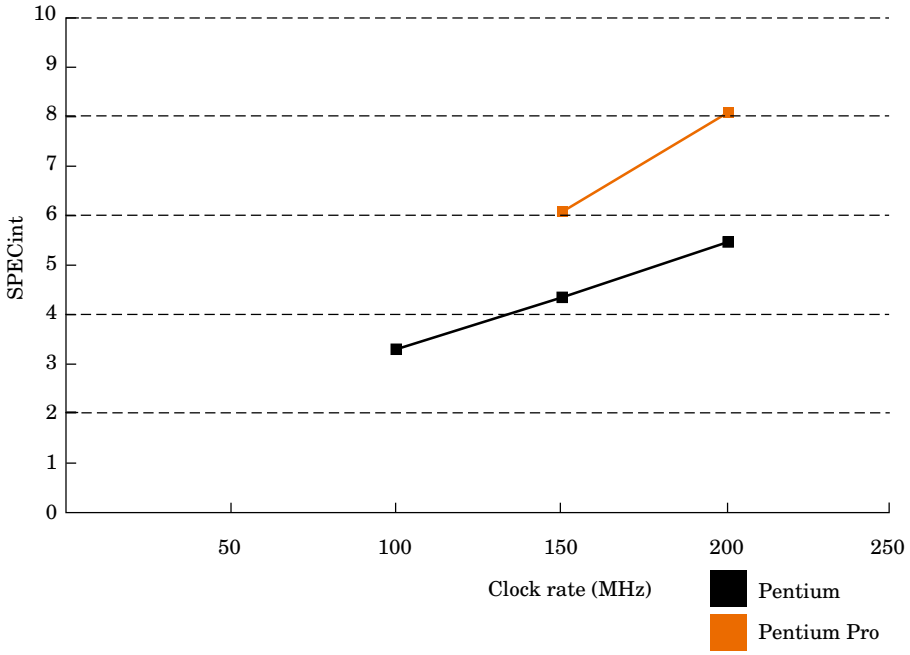


Figura 1.9: Tasas de los SPECint95 para los procesadores Pentium y Pentium Pro

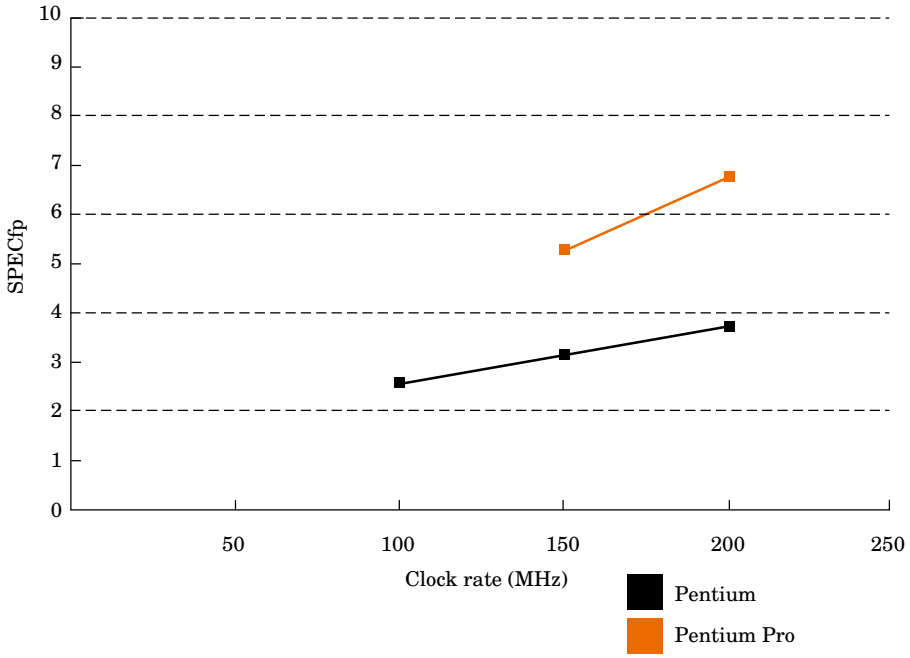


Figura 1.10: Tasas de los SPECfp95 para los procesadores Pentium y Pentium Pro

el sistema de memoria), y de forma inversa, si queremos abaratar costes , lo mejor es penalizar el subsistema con mejor rendimiento. Por otra parte, como no todos los componentes de un PC tienen la misma frecuencia de uso en la ejecución de un programa, resultará más beneficioso mejorar un subsistema cuanto más utilizado sea este.

La ley de Amdahl puede ayudarnos a determinar el aumento de rendimiento que puede tener una arquitectura mejorada. Saber los casos que son más frecuentes es crítico para mejorar el rendimiento. La **ley de Amdahl** establece que la mejora obtenida en el rendimiento de un sistema debido a la alteración de uno de sus componentes está limitada por la fracción de tiempo que se utiliza dicho componente. Se puede escribir como:

$$Aceleracion = \frac{Tiempo\ de\ ejecucion\ antes\ de\ la\ mejora}{Tiempo\ de\ ejecucion\ despues\ de\ la\ mejora} \tag{1.1}$$

El tiempo de ejecución después de la mejora vendrá dado por la expresión:

$$T_d = \frac{T_a}{Cantidad\ de\ la\ mejora} + T_{na} \tag{1.2}$$

donde T_a se corresponde con el tiempo de ejecución afectado por la mejora, y T_{na} se corresponde con el tiempo de ejecución no afectado por la mejora. Sustituyendo la Ecuación 1.2 en 1.1 obtenemos:

$$Aceleracion = \frac{1}{(1 - F_m) + \frac{F_m}{A_m}}$$

La aceleración conseguida depende de 2 factores:

- F_m: La fracción del tiempo de cálculo de la máquina original que pueda utilizarse para aprovechar la mejora, es decir, la fracción de tiempo que el sistema utiliza el subsistema que se ha alterado.
- A_m: La optimización lograda por el modo de ejecución mejorado, es decir el factor de mejora que se ha introducido en el subsistema alterado.