En informática, la compilación en tiempo de ejecución (también conocida por sus siglas inglesas, JIT, *just-in-time*), también conocida como traducción dinámica, es una técnica para mejorar el rendimiento de sistemas de programación que compilan a *bytecode*, consistente en traducir el *bytecode* a código máquina nativo en tiempo de ejecución. La compilación en tiempo de ejecución se construye a partir de dos ideas anteriores relacionadas con los entornos de ejecución: la compilación a *bytecode* y la compilación dinámica.

En un sistema que use compilación a *bytecode* como por ejemplo Smalltalk, Perl, GNU CLISP o las primeras versiones de Java, el código fuente es traducido a un código intermedio llamado *bytecode*. El *bytecode* no es el código máquina de ninguna computadora en particular, y puede por tanto ser portable entre diferentes arquitecturas. El *bytecode* es entonces interpretado, o ejecutado por una máquina virtual.

Un entorno con compilación dinámica es aquél en el que el compilador puede ser usado durante la ejecución. Por ejemplo, la mayoría de los sistemas Commons Lisp tienen una función `compile` que permite compilar nuevas funciones creadas durante la ejecución del programa. Aunque ventajoso en la depuración interactiva, la compilación dinámica es menos útil en un sistema en explotación desatendido. Ese método es más común en emuladoresmodernos y frecuentemente comerciales que requieren mucha velocidad, como el Qemu y el VirtualPC (PC) o el Executor (Macintosh 68k).

En un entorno de compilación en tiempo de ejecución, la compilación a *bytecode* es el primer paso, reduciendo el código fuente a una representación intermedia portable y optimizable. El *bytecode* se despliega en el sistema de destino. Cuando dicho código se ejecuta, el compilador en tiempo de ejecución lo traduce a código máquina nativo. Esto puede realizarse a nivel de fichero (programa) o de funciones, compilándose en este último caso el código correspondiente a una función justo cuando va a ejecutarse (de aquí el nombre de *just-in-time*, «justo a tiempo»).

El objetivo es combinar muchas de las ventajas de la compilación a código nativo y a *bytecode*: la mayoría del «trabajo pesado» de procesar el código fuente original y realizar optimizaciones básicas se realiza en el momento de compilar a *bytecode*, mucho antes del

despliegue: así, la compilación a código máquina del programa resulta mucho más rápida que partiendo del código fuente. El *bytecode* desplegado es portable, a diferencia del código máquina para cualquier arquitectura concreta. Los compiladores dinámicos son más fáciles de escribir, pues el compilador a *bytecode* ya realiza buena parte del trabajo.

Uso de compilación JIT en emuladores

Al momento de emular la CPU, el programa traduce el código máquina del software emulado en código nativo de la máquina emuladora y lo escribe en una caché con permisos de ejecución y escritura (generalmente en UNIX o en sistemas compatibles con POSIX se asigna este permiso con la función mprotect()). La traducción se detiene cuando se encuentra cualquier instrucción que provoque un cambio en el contador de programa, como una interrupción, una instrucción de salto o una llamada a subrutina y es interpretado como un retorno de rutina al compilador o a otras tareas del programa. Luego el emulador ejecuta el código contenido en la caché.

La ventaja de este método es que si el tamaño de la caché es grande, el emulador no necesita recompilar el código, lo que aumenta mucho la velocidad. Sin embargo, surgen problemas cuando el programa emulado escribe su código, y es necesario volver a recompilar dicho código, a menos que el código emulado tenga permisos de solo lectura, por ejemplo en sistemas UNIX con memoria protegida.