Estrictamente hablando, un programa de lenguaje de máquina es un programa de categoría binaria. Debido a la equivalencia simple entre binario y el octal o hexadecimal, es costumbre referir a la catergoria como lenguaje de máquina. Debido a la relación uno a uno entre las instrucción simbólica y su equivalente binario, un lenguaje ensamblador es considerado un lenguaje de nivel de máquina.

Lenguaje de programación se refiere al estudio de la estructura d elos diversos lenguajes de programación de alto nivel. Este estudio se lleva a cabo independiente d ecualquier dispositivo de cómputo particula y su hardware. Puesto que estamos interesados en las relaciones entre software y hardware, el término utilizado aquí tiene connotaciones diferentes puesto que incluye lenguajes de máquina de alto nivel.

**Formatos de las instrucciones**

Cada instrucción debe contener la información que necesita la CPU para su ejecución.

Los elementos constitutivos de una instrucción máquina son:

**Código de operación:** Especifica la operación a realizar (suma, E/S, etc.), La operación se indica mediatne un código binario, denominado código de operación o, abreviadamente, codop.

**Referencia a operandos fuente:** La operación puede implicar a uno o más operandos fuente, es decir, operandos que son entradas para la instrucción.

**Referencia al operando resultado:** La operación puede producir un resultado.

**Referencia a la siguiente instrucción:** Dice a la CPU de dónde captar la siguiente instrucción tras completarse la ejecución de la instrucción actual.

La siguiente instrucción a capturar está en memoria principal o, en el caso de un sistema de memoria virtual, bien en memoria principal o en memoria secundaria (disco). En la mayoría de los casos, la siguiente instrucción a captar sigue inmediatamente a la instrucción en ejecución. En tales casos no hay referencia explícita a la siguiente instrucción. Cuando sea necesaria una referencia explícita, debe suministrarse la dirección de memoria principal o de memoria virtual.

Los operandos fuente y resultado pueden estar en alguna de las siguientes áreas:

Memoria principal o virtual: como en las referencias a instrucciones siguientes, debe indicarse la dirección de memoria principal o de memoria virutal.

Registro de la CPU: salvo raras excepciones, una CPU contine uno o más registros que pueden ser referenciados por instrucciones máquina. Si sólo existe un registro, la referencia a él puede ser implícita. Si existe más de uno, cada registro tenfrá asignado un número único, y la instrucción debe contener el número del registro deseado.

Dispositivo de E/S: La instrucción debe especificar el módulo y ldispositivo de E/S para la operación. En el caso de E/S asignadas en memoria, se dará otra dirección de memoria principal o virtual.

Los diseñadores referencian la duración de un *periodo de reloj* como el tiempo necesario para completar un *ciclo de reloj* (por ejemplo, 10 nanosegundos, o 10 ns), y la *frecuencia de reloj* (por ejemplo, 100 megahercios, o 100 MHz) como la inversa del periodo de reloj.

**Modos de direccionamiento**

Dentro del computador, cada instrucción se representa por una secuencia de bits. La instrucción está dividida en campos, correspondientes a los elementos constitutivos dela misma. En la mayoría de los repertorios de instrucciones se emplea más de un formato.

Durante su ejecución, la instrucción se escribe en un registro de insturcción IR de la CPU.

La CPU debe ser capaz de extraer los datos de los distintos campos dela instrucción para realizar la operación requerida.

Es una práctica común utilizar representaciones simbólicas de las instrucciones máquina.

Los codops se representan mediante abreviaturas, denominadas nemotécnicos, que indican la operación en cuestión. Ejemplos usuales son:

ADD Sumar

SUB Restar

MPY Multiplicar

DIV Dividir

LOAD Cargar datos de memoria

STOR Almacenar datos en memoria (memorizar)

Los operandos tambien suelen representarse simbólicamente. Por ejemplo, la instrucción

ADD R,Y Puede significar sumar el valor contenido en la posición de datos Y al contenido del registro R. En este ejemplo, Y hace referencia a la dirección de una posición de memoria, y R a un registro particular. Observe que la operación se realiza con el contenido de la posición, no con su dirección.

Es posible pues, escribir un programa en lenguaje máquina de forma simbólica. Cada codop simbólico tiene una representación binaria fija, y el programador especifica la posición de cada operando simbólico. Por ejemplo, el programador podría comenzar con una lista de definiciones:

X=513

Y=514

Es raro encontrar ya programadores en lenguaje máquina. La mayoría d elos programas actuales se escriben en un lenguaje de alto nivel o, en ausencia delm ismo, en lenguaje ensamblador, sobre el que trataremos al final de este capítulo. No obstante, el lenguaje máquina simbólico sigue siendo útil para describir las instrucciones máquina, y con ese fin lo utilizaremos.

**Instrucciones Típicas**

Considere una instrucción de alto nivel, tal y como se expresaría en un lenguaje como el

BASIC o el FORTRAN. Por ejemplo:

X=X+Y

Esta senctencia ordena al comptuador sumar los valores almacenados en X y en Y, y poner el resultado en X. Supongamos que las variables X e Y corresponden a las pociiones 513 y 514. Considerando un repertorio simple de instrucciones máquina, la operación podría llevarse a cabo con tres instrucciones:

1. Cargar un registro con el contenido de la posición de memoria 513

**2.** Sumar al registro el contenido de la posición de memoria 514

**3.** Memorizar el contenido del registro en la posición de memoria 513

Una sola instrucción en un lenguaje de alto nivel puede requerir tres instrucciones máquina.

Una sola instrucción en un lenguaje de alto nivel puede requerir tres instrucciones máquina. Este es un caso típico de relación entre un lenguaje de alto nivel y un lenguaje máquina.

una de las principales diferencias entre las arquitecturas RISC y las CISC es el conjunto de instrucciones que pueden acceder a la memoria. Una cuestión relacionada con ésta, que afecta tanto a las arquitecturas RISC como a las CISC, es la elección de los modos de direccionamiento que admite la arquitectura. Los modos de direcclonamiento de una arquitectura son el conjunto de sintaxis y métodos que usan las instrucciones para especificar una dirección de memoria, ya sea la dirección objeto de una referencia a memoria o la dirección de salto de una bifurcación.

DISEÑO DEL REPERTORIO DE INSTRUCCIONES

Uno de los aspectos más interesantes y más analizados del diseño de un computador, es el diseño del repertorio de instrucciones del lenguaje máquina. El diseño de un repertorio de instrucciones es muy complejo, ya que afecta a muchos aspectos del computador. El repertorio de instrucciones define muchas de las funciones realizadas por la CPU y tiene, por tanto.

Un efecto significativo sobre la implementación de la misma. El repertorio de instrucciones es el medio que tiene el programador para controlar la Cpu. En consecuencia, deben considerarse las necesidades del programador a la hora de diseñar el repertorio de instrucciones.

Puede sorprender saber que algunos de los aspectos más básicos relativos al diseño de repertorios O de instrucciones siguen siendo temas de controversia. Los más importantes entre dichos aspectos de diseño son:

. Repertorio de operaciones: Cuántas y qué operaciones considerar, y cuán complejas deben ser.

. Tipos de datos: Los distintos tipos de datos con los que se efectúan operaciones.

. Formatos de instrucciones: Longitud de la instrucción (en bits), número de direcciones, tamaño de los distintos campos, etc.

. Registros: Número de registros de la CPU que pueden ser referenciados por instrucciones, y su uso.

. Direccionamiento: El modo o modos de direccionamiento mediante los cuales puede especificarse la dirección de un operando.

DISEÑO DEL REPERTORIO DE INSTRUCCIONES

Uno de los aspectos más interesantes y más analizados del diseño de un computador, es el diseño del repertorio de instrucciones del lenguaje máquina. El diseño de un repertorio de instrucciones es muy complejo, ya que afecta a muchos aspectos del computador. El repertorio de instrucciones define muchas de las funciones realizadas por la CPU y tiene, por tanto.

Un efecto significativo sobre la implementación de la misma. El repertorio de instrucciones es el medio que tiene el programador para controlar la Cpu. En consecuencia, deben considerarse las necesidades del programador a la hora de diseñar el repertorio de instrucciones.

Puede sorprender saber que algunos de los aspectos más básicos relativos al diseño de repertorios O de instrucciones siguen siendo temas de controversia. Los más importantes entre dichos aspectos de diseño son:

. Repertorio de operaciones: Cuántas y qué operaciones considerar, y cuán complejas deben ser.

. Tipos de datos: Los distintos tipos de datos con los que se efectúan operaciones.

. Formatos de instrucciones: Longitud de la instrucción (en bits), número de direcciones, tamaño de los distintos campos, etc.

. Registros: Número de registros de la CPU que pueden ser referenciados por instrucciones, y su uso.

. Direccionamiento: El modo o modos de direccionamiento mediante los cuales puede especificarse la dirección de un operando.

Estos aspectos están fuertemente interrelacionados, y deben considerarse conjuntamente en el diseño de un repertorio de instrucciones.

(RISC, Reduced Instruction Set Computer). La arquitectura RISC cuestiona muchas de las decisiones tomadas sobre repertorios de instrucciones de computadores comerciales contemporáneos. Un ejemplo de máquina RISC es el PowerPC.

En los computadores son usuales tres tipos de datos numéricos:

. Enteros o en coma fija

. En coma flotante

. En decimal

1: LD rl, (r2)

2: ADD r5, r6, r7 Ciclo 1: LD rl, (r2) ADD r5,r6,r7

3: SUB r4, r1, r4 Ciclo 2: SUB r4, r1,r4 MUL r8, r9, r10

4: MUL r8, r9, r10 Ciclo 3: ST (r11); r4

5: ST (r11), r4

**Figura**. Ejemplo de paralelismo entre instrucciones.

**Descripción : ¿Qué es el paralelismo entre instrucciones?**

Los procesadores con paralelismo entre instrucciones aprovechan el hecho de que muchas de las instrucciones en un programa secuencial no dependen de las instrucciones del programa cercanas que las preceden. Por ejemplo, considere el fragmento de programa de la parte izquierda de la figura. Las instrucciones 1, 3 y 5 son dependientes entre si porque la instrucción 1 genera un valor que es usado por la instrucción 3, la cual genera un resultado que utiliza la instrucción 5. Las instrucciones 2 y 4 no usan los resultados de ninguna otra instrucción del fragmento y no generan resultados que se usen por ninguna instrucción del fragmento. Estas dependencias requieren que las instrucciones 1, 3 y 5 sean ejecutadas en orden para generar un resultado correcto, pero las instrucciones 2 y 4 pueden ejecutarse antes, después o en paralelo con cualquiera de las otras instrucciones sin cambiar los resultados del fragmento de programa.

En un procesador que ejecute las instrucciones secuencialmente, el tiempo de ejecución de este fragmento de programa seria de al menos 5 ciclos, incluso en un procesador sin segmentar con una latencia de un para cada instrucción. Por el contrario, in procesador no segmentado que sea capaz de ejecutar dos instrucciones de forma simultanea podría ejecutar este fragmento de programa en tres ciclos si cada instrucción tuviera una latencia de un ciclo, como se muestra en la parte derecha de la figura.

Los saltos limitan el paralelismo entre instrucciones porque el procesador no puede saber que instrucciones debe ejecutar después de un salto hasta que su ejecución se haya completado. Esto requiere que el procesador tenga que esperar a que el salto se complete antes de ejecutar la siguiente instrucción.

**EJEMPLO**

Considere el siguiente fragmento de programa:

ADD r1, r2, r3

LD r4 (r5)

SUB r7, r1, r9

MUL r5,r4, r4

SUB r1, r12, r10

ST (r13), r14

OR r 15, r14, r12

¿Cuanto tardaría en emitirse este fragmento de programa en un procesador que puede ejecutar dos instrucciones simultáneamente? ¿Y en un procesador que pueda ejecutar cuatro instrucciones simultáneamente? Suponga que el procesador puede ejecutar las instrucciones en cualquier orden siempre que no se violen las dependencias de datos, que todas las instrucciones tienen una latencia de un ciclo y que todas las unidades de ejecución del procesador pueden ejecutar cualquier tipo de instrucción.

**SOLUCION**

En un procesador que pueda ejecutar dos instrucciones simultáneamente, este fragmento de programa tardaría cuatro ciclos en emitirse. A continuación se muestra una posible secuencia de emisión, aunque se pueden formar mas secuencias de instrucciones que tarden lo mismo.

Ciclo 1: ADD r1,r2,r3 LD r4, (r5)

Ciclo 2: SUB r7, r1, r9 MUL r5, r4, r4

Ciclo 3 : SUB r1, r12, r10 ST (r13), r14

Ciclo 4: OR r15, r14, r12

Si el procesador pudiera ejecutar cuatro instrucciones simultáneamente, el programa se podría emitir en dos ciclos, tal y como se muestra a continuación:

Ciclo 1: ADD r1, r2, r3 LD r4, (r5) ST (r13), r14 OR r15, r14, r12

Ciclo 2: SUB r7, r1, r9 MUL r5, r4, r4 SUB r1, r12, r10

Observe que, independientemente del numero de instrucciones que el procesador pueda ejecutar simultáneamente, no es posible emitir todo el fragmento de programa en un solo ciclo, ya que existen dependencias RAW entre las instrucciones ADD r2, r3 y SUB r7, r1,

r9 y entre las instrucciones LD r4, (r5) y MUL r5, r4, r4. Obsérvese también que las instrucciones SUB r7, r1, r9 y SUB r1, r12, r10, que tienen una dependencia WAR, se emiten en el mismo ciclo de reloj.

**Buses del sistema.**

El bus se puede definir como un conjunto de líneas conductoras de hardware utilizadas para la transmisión de datos entre los componentes de un sistema informático. Un bus es en esencia una ruta compartida que conecta diferentes partes del sistema, como el microprocesador, la controladora de unidad de disco, la memoria y los puertos de entrada/salida (E/S), para permitir la transmisión de información.

En el bus se encuentran dos pistas separadas, el bus de datos y el bus de direcciones. La

CPU escribe la dirección de la posición deseada de la memoria en el bus de direcciones accediendo a la memoria, teniendo cada una de las líneas carácter binario. Es decir solo pueden representar 0 o 1 y de esta manera forman conjuntamente el número de la posición dentro de la memoria (es decir: la dirección). Cuantas más líneas hayan disponibles, mayor es la dirección máxima y mayor es la memoria a la cual puede dirigirse de esta forma. En el bus de direcciones original habían ya 20 direcciones, ya que con 20 bits se puede dirigir a una memoria de 1 Mb y esto era exactamente lo que correspondía a la CPU.

Esto que en le teoría parece tan fácil es bastante mas complicado en la práctica, ya que aparte de los bus de datos y de direcciones existen también casi dos docenas más de líneas de señal en la comunicación entre la CPU y la memoria, a las cuales también se acude. Todas las tarjetas del bus escuchan, y se tendrá que encontrar en primer lugar una tarjeta que mediante el envío de una señal adecuada indique a la CPU que es responsable de la dirección que se ha introducido. Las demás tarjetas se despreocupan del resto de la comunicación y quedan a la espera del próximo ciclo de transporte de datos que quizás les incumba a ellas.

Este mismo concepto es también la razón por la cual al utilizar tarjetas de ampliación en un PC surgen problemas una y otra vez, si hay dos tarjetas que reclaman para ellas el mismo campo de dirección o campos de dirección que se solapan entre ellos.

Los datos en si no se mandan al bus de direcciones sino al bus de datos. El bus XT tenía solo 8 bits con lo cual sólo podía transportar 1 byte a la vez. Si la CPU quería depositar el contenido de un registro de 16 bits o por valor de 16 bits, tenía que desdoblarlos en dos bytes y efectuar la transferencia de datos uno detrás de otro.

De todas maneras para los fabricantes de tarjetas de ampliación, cuyos productos deben atenderse a este protocolo, es de una importancia básica la regulación del tiempo de las señales del bus, para poder trabajar de forma inmejorable con el PC. Pero precisamente este protocolo no ha sido nunca publicado por lBM con lo que se obliga a los fabricantes a medir las señales con la ayuda de tarjetas ya existentes e imitarlas. Por lo tanto no es de extrañar que se pusieran en juego tolerancias que dejaron algunas tarjetas totalmente eliminadas.