**ELEMENTOS DE UNA INSTRUCCIÓN DE MAQUINA**

Una instrucción de máquina es un código binario compuesto por una cantidad determinada de bits. Los bits son usados por la Unidad de Control (CU) para continuar y completar el proceso de ejecución de la instrucción. Para ello, el código debe contener suficiente “información” para que la Unidad de Control pueda resolverla. Las instrucciones tienen distinto grado de complejidad. Van desde las que no hacen nada, hasta las que operan sobre múltiples datos.

En general las instrucciones más complejas requieren de múltiples pasos o acciones. Las más sencillas se resuelven en pocos pasos. Un ejemplo de instrucción bastante “compleja” es la que busca 2 operandos y produce un resultado. Para esta instrucción la CPU necesita definir:

* Tipo de operación
* Lugar del 1er operando (donde está)
* Lugar del 2do operando (donde está)
* Lugar del resultado (donde guardarlo)
* Donde está la próxima instrucción

Dado que hay diferentes datos que proveer, la instrucción está organizada en conjuntos de bits, comúnmente denominados campos, que proveerán esos datos. La cantidad de bits de un campo dependen de la cantidad de información que debe proveer. Por ejemplo, si un procesador dispone de 256 tipos de instrucciones, y un campo se usa para especificar el tipo de instrucción (campo de Código de Operación COP), entonces ese campo requiere de al menos 8 bits.

Los campos que pueden estar incluidos en una instrucción son:

1. Identificación de la instrucción (Código de operación): Especifica la operación a realizar (ej. suma).
2. Referencias a los operandos: Establece la información relativa a él o los operandos fuentes (si es que se requieren). La información relativa puede ser de distinto tipo (se verá con los modos de direccionamiento). La operación puede involucrar uno o más operando fuente (o, de entrada). Por ejemplo, en una suma se requiere especificar 2 operandos
3. Referencia del resultado: Establece dónde almacenar el resultado. La operación puede involucrar uno o más resultados (de salida). Por ejemplo, en una suma se produce 1 resultado solamente.
4. Referencia de la siguiente instrucción: Provee a la CPU con información para determinar donde buscar la siguiente instrucción después de la ejecución de la instrucción anterior. En la mayoría de los casos, la próxima instrucción se ubica a continuación de la instrucción actual, es decir, en la siguiente posición de memoria. Esta situación se conoce como acceso secuencial a instrucciones consecutivas de memoria.

Respecto de los puntos 2 y 3, los operandos (fuente) y el resultado (destino) pueden residir en tres lugares: Memoria, Registro de la CPU y Dispositivo de E/S.

**LENGUAJE DE MAQUINA O ABSOLUTO**

El lenguaje absoluto o de máquina es el binario. Las instrucciones están codificadas mediante un conjunto de bits almacenados en palabras de memoria (1 palabra o más). El conjunto de bits que forma una instrucción se puede considerar organizado en campos, en correspondencia con la información que proveen. La representación de las instrucciones en memoria es puramente binaria. Se pueden emplear otras representaciones numéricas para facilitar su escritura (BCH, BCD, etc.).

**LENGUAJE SIMBÓLICO (ASSEMBLY)**

Dado que es prácticamente imposible para el programador, tratar con las representaciones binarias de las instrucciones de máquina, se usan otras representaciones más “amigables”. Estas representaciones tienen distintos grados o niveles de abstracción. En el nivel siguiente al nivel de lenguaje de máquina está la Representación simbólica (Lenguaje Assembly). En la representación simbólica, los campos que forman la instrucción se pueden representar con textos o números.

Por ejemplo, el campo binario de la instrucción que contiene la información del tipo de instrucción (campo de código de operación COP) puede ser reemplazado, en el lenguaje simbólico (Assembly) por mnemónicos más sencillos de memorizar:

* ADD (significa código de operación de) adición (suma)
* SUB sustracción (resta)
* MOV movimientodedatos
* AND,OR,XOR operaciones lógica

Los operandos también se pueden representar de manera simbólica. Por ejemplo, una instrucción que mueve un dato desde una posición de memoria memoY a un registro reg1 se puede escribir así:

**MOV reg1 , memoY**

En el ejemplo anterior se ha referenciado la posición de memoria mediante un texto simbólico memoY, y el registro mediante el texto reg1.

**Instrucciones de lenguajes de alto nivel (HLL) y de máquina**

Lenguaje de alto nivel (HLL): En lenguajes de alto nivel las sentencias se escriben de la siguiente manera:

**X := X +Y**

Esta sentencia suma 2 variables X e Y, y el resultado se guarda en una de ellas X. El lenguaje no referencia el lugar físico de las variables, solo su identificación. Las sentencias en un lenguaje de alto nivel, expresan acciones en forma “concisa”. Las instrucciones en el lenguaje de máquina expresan acciones en forma “básica”. Por lo tanto, una sentencia de alto nivel requerirá de varias instrucciones de máquina para su resolución.

Dado que la máquina solo “entiende” (es capaz de resolver) las instrucciones de máquina, los programas escritos en lenguajes de alto nivel se deben convertir (compilar) al lenguaje de máquina para ser ejecutados. El conjunto de instrucciones de máquina debe ser capaz de expresar (resolver) cualquiera de las instrucciones de un lenguaje de alto nivel.

Gap semántico: Hay un factor de relación entre la sentencia de alto nivel y la cantidad de instrucciones de máquina que la resuelve. Este factor se calcula como la cantidad de instrucciones de máquina que se requieren para resolver una sentencia de un HLL. Esta relación se conoce como “Gap semántico”. Este Gap depende del lenguaje de alto nivel, y del conjunto de instrucciones de máquina (y también del compilador).

**FORMATO DE INSTRUCCIÓN**

Para entender la estructura de una instrucción, es necesario analizar el tipo de información que referencia cada campo de la misma.

Referencias a operandos: Los operandos pueden estar en:

* Un registro, en cuyo caso la referencia en la instrucción será un número de registro
* Una posición de memoria, en cuyo caso la referencia será una dirección de memoria

Referencias a resultados: Son iguales a las referencias a operandos, es decir un registro o una posición de memoria.

Referencias a la próxima instrucción a ejecutar: Una posición de memoria, en cuyo caso la referencia será una dirección de memoria.

La estrategia de organización de los campos de una instrucción y la información contenida en ellos se conoce como formato de instrucción.

**TAXONOMÍA BASADA EN FORMATO DE INSTRUCCIÓN**

Una forma de clasificación de las máquinas está basada en el formato de instrucción. La clasificación identifica la cantidad de campos que contienen direcciones de memoria. Suponiendo el caso de una instrucción de suma de 2 operandos en memoria y resultado almacenado en memoria, se necesitan:

* 2 campos de direcciones para hacer referencia a los operandos.
* 1 campo de dirección para hacer referencia a donde almacenar el resultado.
* 1 campo de dirección de referencia de la ubicación de la próxima instrucción.

**FORMATO DE INSTRUCCIÓN – MÁQUINA DE 4 DIRECCIONES**

Una máquina cuya instrucción requiere 2 operandos y produce 1 resultado (por ejemplo, una suma) dispone de 4 campos de dirección:

Add DirRes DirOp1 DirOp2 DirPróxIns

En lenguaje Assembly, la instrucción debería tener la siguiente forma:

Add DirRes, DirOp1, DirOp2, DirPróxIns

La operación de suma se ejecutaría de la siguiente manera:

Memoria

DirOp1 Op1

DirOp2 Op2

DirRes Res

DirPróxIns PróxInst

Características de la máquina de 4 direcciones:

Todas las direcciones son explícitamente expresadas: operandos, resultado y próxima instrucción. Cada campo de dirección tiene que tener suficientes bits para “acomodar” o referenciar una dirección completa. Si, por ejemplo, la dirección requiere de 24 bits, la instrucción tiene: 4 x 24 = 96 bits de referencias.

**FORMATO DE INSTRUCCIÓN – MÁQUINA DE 3 DIRECCIONES**

La máquina de 4 direcciones requiere de muchísimos bits para especificarla. Se puede reducir su tamaño si se elimina el campo que referencia la dirección de la próxima instrucción, y se lo reemplaza por un registro en la CPU (llamado Contador de Programa, PC) que cumpla esa función.

Add DirRes DirOp1 DirOp2

En lenguaje Assembly, la instrucción debería tener la siguiente forma:

Add DirRes, DirOp1, DirOp2

La operación de suma se ejecutaría de la siguiente manera:

Memoria

DirOp1 Op1

DirOp2 Op2

DirRes Res

CPU + PC DirPróxIns

Características de la máquina de 3 direcciones:

Las direcciones de operandos, resultado y próxima instrucción son explícitamente referenciadas. Se elimina la referencia a la dirección de la próxima instrucción a ejecutar. Se agrega un registro específico para contener la dirección de la próxima instrucción a ejecutar. Este registro se conoce como Contador de Programa (o PC: Program Counter). Cada campo de dirección tiene que tener suficientes bits para “acomodar” o referenciar una dirección completa. Si, por ejemplo, la dirección requiere de 24 bits, la instrucción tiene: 3 x 24 = 72 bits de referencias.

**FORMATO DE INSTRUCCIÓN – MÁQUINA DE 2 DIRECCIONES**

Se puede reducir aún más el tamaño de la instrucción si se elimina otro campo que referencia una dirección. En función del campo que se elimina, se tienen 2 opciones:

Opción 1: eliminar la referencia al resultado.

Add DirOp1 DirOp2

En assembly:

Add DirOp1, DirOp2

Opción 2: eliminar la referencia a un operando.

Add DirRes DirOp2

En assembly:

Add DirRes, DirOp2

Características de la máquina de 2 direcciones:

Las direcciones de los 2 operandos (opción 1), o la de 1 operando y la de resultado (opción 2) son explícitamente referenciadas. Cada campo de dirección tiene que tener suficientes bits para “acomodar” o referenciar una dirección completa. En la opción 1, el resultado “pisa” (sobrescribe) alguno de los operandos (en memoria o en registro). En la opción 2, el primer operando está en algún lugar predeterminado al que se debe mover previamente. Si, por ejemplo, la dirección requiere de 24 bits, la instrucción tiene: 2 x 24 = 48 bits de referencias.

La operación de suma se ejecutaría de la siguiente manera:

Memoria

DirOp1 Op1, Res

DirOp2 Op2

CPU + PC DirPróxIns

DirPróxIns PróxInst

**FORMATO DE INSTRUCCIÓN – MÁQUINA DE 1 DIRECCIÓN**

Se puede simplificar aún más el tamaño de la instrucción si se elimina otro campo más, que referencia una dirección. Es decir, la instrucción dispone de 1 solo campo de dirección, que se usa para referenciar un operando o un resultado.

Add DirOp1/DirRes

En lenguaje Assembly:

Add DirOp1

La operación de suma se ejecutaría de la siguiente manera:

Memoria

DirOp1

CPU

Op1 + Acumulador

PC DirPróxIns

DirPróxIns PróxInst

Características de la máquina de 1 dirección:

Una sola dirección es explícitamente referenciada. El campo de dirección tiene que tener suficientes bits para “acomodar” o referenciar una dirección completa. Dado que la instrucción especifica una sola dirección donde reside el segundo operando, el primer operando está implícitamente almacenado en algún lugar, normalmente en un registro de la CPU, denominado Acumulador. El lugar a donde se va a guardar el resultado está implícito. Normalmente es en el mismo registro que contenía el primer operando, es decir, en el Acumulador. La CPU requiere de “registros especiales” (Acumulador) y nuevas instrucciones para cargar y descargar los “registros especiales” (Acumulador). El lugar donde reside un operando está predefinido (implícito). El lugar donde se guarda el resultado está predefinido (implícito). Si, por ejemplo, la dirección requiere de 24 bits, la instrucción tiene: 1 x 24 = 24 bits de referencias.

**COMPARACIÓN ENTRE MÁQUINAS DE 3, 2 Y 1 DIRECCIÓN**

Ejemplo: Analizar la cantidad de instrucciones y accesos a memoria para el siguiente ejercicio: a = (b + c) \* d - e

Para máquinas de 3 direcciones:

3 accesos a instrucciones

9 accesos a datos

Total: 12 accesos

Para máquinas de 2 direcciones:

4 accesos a instrucciones

11 accesos a datos

Total: 15 accesos

Para máquinas de 1 dirección:

5 accesos a instrucciones

5 accesos a datos

Total: 10 accesos

**MODOS DE DIRECCIONAMIENTO**

Tal como se mencionó anteriormente, en una instrucción hay un campo reservado para expresar el código de operación (COP) que identifica qué debe hacer la instrucción. El resto de los campos hacen referencia a los lugares donde residen los operandos y a dónde se va a mandar (guardar) el resultado.

Hay 3 métodos para reducir el tamaño de los campos que referencian operandos o resultados:

Especificando registros: La cantidad de registros es pequeña, con lo que la cantidad de bits para especificarlo es también muy baja. Normalmente, si un dato va a usarse repetidas veces, es conveniente cargarlo en un registro. Usar un registro para una variable tiene una ventaja adicional: el acceso al registro es mucho más rápido que a la memoria.

Referenciando en forma implícita operandos y/o resultado: Ejemplo: reg2 = reg2 + fuente1

Usando técnicas de referenciamiento: Estas técnicas se conocen como modos de direccionamiento.

Los Modos de direccionamiento (Mdd) son las distintas formas o estrategias que tiene un procesador para referenciar operandos, resultado y/o próxima instrucción a ejecutar. Los Mdd tienen como objetivos:

* Disminuir la cantidad de bits en la instrucción.
* Posibilitar manejar referencias obtenidas durante la ejecución del programa (calculadas).
* Permitir un manejo más eficiente de datos (arreglos).

Existe una amplia variedad de modos de direccionamiento (Mdd). A continuación, se analizan algunos de los más importantes: Inmediato, Directo, Por registro, Indirecto por memoria, Indirecto por registro, Por desplazamiento y Del stack.

**MODO DE DIRECCIONAMIENTO INMEDIATO**

En el modo de direccionamiento inmediato, el operando está contenido dentro de la instrucción misma. El formato de la instrucción incluye el código de operación (COP) y el operando inmediato. El operando se obtiene al leer la instrucción, por lo que no se requiere una referencia adicional a la memoria para buscar el dato. Este modo es útil para la definición de constantes e inicialización de variables cuando el dato es conocido al momento de la compilación. Sin embargo, una desventaja es que el tamaño del operando influye directamente en el tamaño de la instrucción.

**MODO DE DIRECCIONAMIENTO DIRECTO O ABSOLUTO**

En el modo de direccionamiento directo o absoluto, la instrucción contiene la dirección completa de memoria donde reside el operando. El formato de la instrucción incluye el código de operación (COP) y la dirección completa. La dirección efectiva (EA) es el campo que contiene la dirección donde se encuentra el operando. Aunque este modo es simple, tiene un espacio limitado de direcciones debido a la cantidad de bits disponibles para el campo de dirección. Es útil para el acceso a variables globales, cuya dirección es conocida en el momento de la compilación. La desventaja es que para espacios de direcciones grandes, se requieren muchos bits para especificar la dirección completa, lo que aumenta el tamaño de la instrucción.

**MODO DE DIRECCIONAMIENTO A REGISTRO**

En el modo de direccionamiento a registro, la instrucción contiene el número del registro donde reside el operando. El formato de la instrucción incluye el código de operación (COP) y el número de registro. En este modo, se especifica un número de registro en lugar de una posición de memoria. Conceptualmente, es similar al direccionamiento directo, pero con la diferencia de que la referencia es a un registro. Este modo es útil para el acceso a variables locales o globales. La ventaja es que la referencia a un registro requiere menos bits que la especificación de una dirección y no requiere accesos adicionales a la memoria de datos. La desventaja es que los registros son limitados y representan un recurso valioso.

**MODOS DE DIRECCIONAMIENTO: INDIRECTO VÍA MEMORIA**

En el modo de direccionamiento indirecto vía memoria, la instrucción contiene la dirección de memoria donde reside la dirección del operando. El formato de la instrucción incluye el código de operación (COP) y una dirección inicial que apunta a otra dirección (Dirección1 y Dirección2). La Dirección1 puede ser más corta que la Dirección2, lo que permite usar una dirección de menos bits para apuntar a una dirección de más bits. La Dirección2 se puede obtener durante la ejecución del programa. Este modo es útil para la administración de estructuras de datos. La ventaja es un espacio de direccionamiento mayor, mientras que la desventaja son los múltiples accesos a memoria necesarios.

**MODOS DE DIRECCIONAMIENTO: INDIRECTO VÍA REGISTRO**

En el modo de direccionamiento indirecto vía registro, la instrucción contiene el número del registro donde reside la dirección del operando. El formato de la instrucción incluye el código de operación (COP) y el número del registro. En este modo, se especifica el registro que contiene la dirección efectiva (EA) del operando. Este modo es útil para la administración de estructuras de datos. La ventaja es que la instrucción es más corta porque se requieren menos bits para especificar el registro en comparación con el indirecto por memoria. Además, el espacio de direccionamiento es grande y hay menos accesos a memoria porque la EA está en un registro interno. La desventaja es que hay pocos registros disponibles.

**MODOS DE DIRECCIONAMIENTO: DIRECCIONAMIENTOS CALCULADOS**

Los modos de direccionamiento calculados son aquellos en los que se realiza una operación aritmética o lógica para obtener la dirección efectiva (EA) del operando. La instrucción contiene dos o más referencias numéricas que se combinan para obtener la EA, generalmente mediante una suma aritmética. La suma es aritmética y, dado que se trata de direcciones, el valor resultante es un número sin signo. Estos modos permiten obtener la EA a

**DIRECCIONAMIENTO CALCULADO: RELATIVO AL PC**

En el modo de direccionamiento relativo al PC, la instrucción contiene un campo numérico que es un desplazamiento (Δ) que se suma al valor actual del contador de programa (PC) para obtener la dirección efectiva (EA). El registro referenciado es el PC, y el desplazamiento Δ se trata como un número en complemento a 2, permitiendo números con signo (positivos o negativos). Si el número es positivo, la referencia es hacia adelante; si es negativo, hacia atrás. El resultado de la operación de suma puede utilizarse de dos maneras: obtener la EA del operando sin alterar el PC o obtener la dirección de la próxima instrucción a ejecutar, causando un salto en el proceso de ejecución normal del programa.

**DIRECCIONAMIENTO CALCULADO: BASE**

En el modo de direccionamiento base, la instrucción contiene dos campos: una referencia a un registro base y un campo numérico. El registro base contiene una dirección de memoria, y el campo numérico es un desplazamiento (offset) que se suma al contenido del registro base para obtener la dirección efectiva (EA).

**DIRECCIONAMIENTO CALCULADO: INDEXADO**

En el modo de direccionamiento indexado, la instrucción contiene dos campos: una dirección base y una referencia a un registro índice. A la dirección base se le suma el contenido del registro índice para obtener la dirección efectiva (EA). Este modo es similar al modo base, pero los papeles de base y desplazamiento están intercambiados. La indexación proporciona un mecanismo eficiente para operaciones iterativas, y en algunas máquinas, el registro índice se incrementa o decrementa como parte de la instrucción.

**MODOS DE DIRECCIONAMIENTO: PILA O STACK**

El modo de direccionamiento tipo Pila no requiere referencia a la dirección del operando. El operando está en el tope de una estructura tipo LIFO (Last In, First Out), conocida como pila o stack. La pila es un arreglo lineal de posiciones de memoria accesibles solo desde el tope. Asociado con la pila, hay un registro apuntador (o registro puntero de pila), cuyo contenido es la dirección del tope de la pila.

**DISEÑO DEL CONJUNTO DE INSTRUCCIONES**

El conjunto de instrucciones es el mecanismo mediante el cual el programador controla las acciones de la CPU. Los aspectos importantes en el diseño del repertorio de instrucciones incluyen:

1. Formato de instrucciones: Tamaño en bits que ocupa la instrucción, número de campos que forman la instrucción, cantidad de referencias explícitas a operandos y resultados, y tamaño en bits de cada campo.
2. Tipos de operaciones: Cuántas operaciones (instrucciones) tiene el procesador y qué tipo y variedad de instrucciones provee el repertorio.
3. Modos de direccionamiento: Cantidad y tipo de modos de direccionamiento.
4. Registros: Cantidad y tipo de registros, y registros referenciados por las instrucciones.
5. Tipos de datos: Numéricos (representaciones con y sin signo), punto fijo, punto flotante, representaciones alfanuméricas.

**DISEÑO DEL CONJUNTO DE INSTRUCCIONES: TIPOS COMUNES**

Los tipos de instrucciones más comunes incluyen:

* Transferencia de datos: Ejemplo en Assembly: Mov (load/store).
* Aritméticas: Ejemplo en Assembly: Add, Sub, Inc, Dec, Mul.
* Lógicas: Ejemplo en Assembly: And, Or, Xor, Not.
* Conversión: Ejemplo en Assembly: DAA.
* Entrada/salida: Ejemplo en Assembly: In, Out.
* Transferencia de control (salto, bifurcación): Ejemplo en Assembly: JMP, CALL, RET.
* Control del sistema: Ejemplo en Assembly: HLT.

**DISEÑO DEL CONJUNTO DE INSTRUCCIONES: TIPOS DE DATOS**

* Los tipos de datos más comúnmente usados son:
* Entero sin signo
* Entero con signo
* Punto fijo
* Punto flotante
* Caracteres: ASCII, BCD
* Variables lógicas