Capítulo 10. Modos de Direccionamiento y Formatos de Instrucciones

Un operando en una instrucción contiene el valor real del operando (inmediato) o una referencia a la dirección donde se encuentra el operando. Una amplia variedad de modos de direccionamiento se utilizan en los conjuntos de instrucciones. Estos incluyen: directo, indirecto, registro, registro indirecto y varias formas de desplazamiento.

El formato de instrucción define la estructura de los campos de la instrucción. El diseño del formato de instrucción es una tarea compleja que debe incluir consideraciones como la longitud, variable o fija, el número de bits para opcode y cada operando y cómo se determina el modo de direccionamiento.

10.1 Modos de direccionamiento

Los campos de direcciones en un formato de instrucción típico son relativamente pequeños. Nos gustaría poder referenciar un rango amplio de localidades en memoria principal o, para ciertos sistemas, memoria virtual. Para lograr este objetivo, una variedad de técnicas de direccionamiento han sido empleadas. Pueden involucrar algún tipo de compensación o intercambio entre el rango de instrucciones direccionables (flexibilidad) y el número de referencias a memoria en cada instrucción (complejidad del cálculo). En esta sección se examinan las técnicas de direccionamiento más comunes.

En la siguiente tabla se muestran los cálculos de memoria realizados para cada modo de direccionamiento con la siguiente notación:

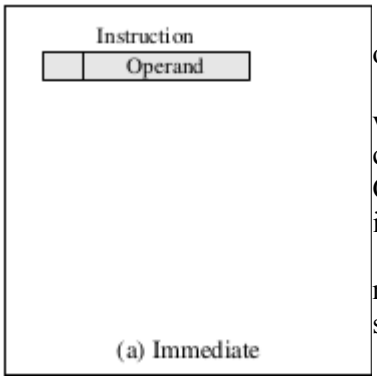
A = contenido de un campo de dirección en la instrucción
R = contenido de un campo de direccción en la instrucción que se refiere a un registro
EA = dirección efectiva de la localidad que contiene a un operando referenciado
(X) = contenido de la localidad de memoria X o el registro X

Mode	Algorithm	Principal Advantage	Principal Disadvantage
Immediate	Operand = A	No memory reference	Limited operand magnitude
Direct	EA = A	Simple	Limited address space
Indirect	EA = (A)	Large address space	Multiple memory references
Register	EA = R	No memory reference	Limited address space
Register indirect	EA = (R)	Large address space	Extra memory reference
Displacement	EA = A + (R)	Flexibility	Complexity
Stack	EA = top of stack	No memory reference	Limited applicability

Virtualmente todas las arquitecturas de computadora soportan más de uno de los modos de direccionamiento. Entonces surge la pregunta de cómo determina el procesador cuál de ellos se utilizará para una instrucción en particular. Varios enfoques se han utilizado. En ocasiones, diferentes opcodes utilizan diferentes modos de direccionamiento. También, uno o más bits en el formato de instrucción pueden utilizarse como un campo de *modo*. El valor del campo de modo determina el tipo de direccionamiento a ser utilizado.

En un sistema sin memoria virtual, el término *dirección efectiva* será la dirección de una localidad de memoria o un registro. En un sistema con memoria virtual, la dirección efectiva es una dirección virtual o un registro. El mapeo real entre direcciones virtuales y físicas es una función de la unidad de manejo de memoria (MMU, *memory management unit*) y es invisible para el programador.

Direccionamiento Inmediato



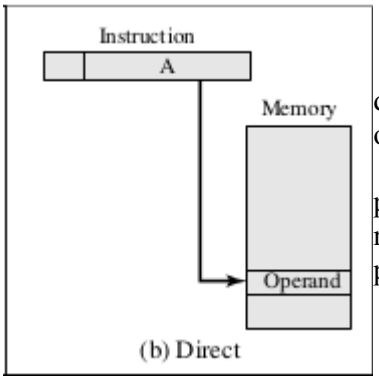
La forma más simple de direccionamiento es cuando el valor del operando está presente en la instrucción: Operando = A.

Este modo puede ser usado para definir y utilizar constantes o establecer valores iniciales para las variables. Típicamente, el número será almacenado en complemento a dos; el bit de más a la izquierda se utiliza como bit de signo. Cuando el operando se guarda en un registro, el signo de bit se extiende hacia la izquierda hasta llegar al tamaño completo de la palabra.

La ventaja del direccionamiento inmediato es que no se requiere ninguna referencia a memoria fuera de la obtención de la instrucción, con lo cual se ahorra un ciclo de memoria/cache durante el ciclo de instrucción.

La desventaja es que el tamaño del número representado está restringido por el tamaño del campo de dirección, que, en muchos conjuntos de instrucciones, es pequeño comparado con la longitud de palabra.

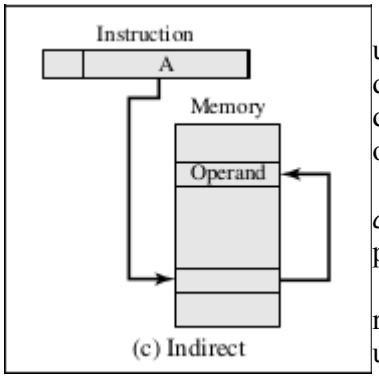
Direccionamiento Directo



Una forma muy simple de direccionamiento es el direccionamiento directo, en el cual el campo de dirección contiene la dirección efectiva del operando: EA = A.

Ésta técnica fue común en las primeras generaciones de computadoras pero no es muy común en arquitecturas contemporáneas. Sólo requiere de una referencia a memoria sin ningún cálculo especial. La limitación obvia es que provee únicamente un espacio de direcciones limitado.

Direccionamiento Indirecto



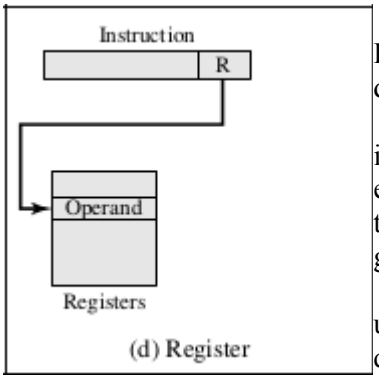
Con el direccionamiento directo, la longitud del campo de direcciones es usualmente menor al del tamaño de la palabra y, por tanto, limita el rango de direcciones. Una solución es hacer que el campo de direcciones se refiera a la dirección de una palabra en memoria, que a su vez contiene la dirección del operando. A ésto se le conoce como direccionamiento indirecto: EA = (A).

Como se definio antes, los paréntesis deben ser interpretados como *el contenido de*. La ventaja obvia de este enfoque es que para una longitud de palabra N, se tiene un espacio de direcciones disponible de 2^N.

La desventaja es que la ejecución de la instrucción requiere de dos referencias a memoria para obtener el operando: una para obtener su dirección y una segunda para obtener su valor.

Aunque el número de localidades que pueden ser referenciadas es ahora igual a 2^N, el número de direcciones efectivas diferentes que pueden ser referenciadas una a la vez es 2^K, donde K es la longitud del campo de direcciones. Una variante raramente utilizada del direccionamiento indirecto es hacer multinivel o cascada: EA = (. . . (A) . . .). En este caso, un bit de la dirección es una bandera de indirección (I). Si el bit I es 0, entonces la palabra contiene la dirección efectiva. Si el bit I es 1, entonces se invoca otro nivel de indirección.

Direccionamiento de Registro



El direccionamiento de registro es similar al direccionamiento directo. La única diferencia es que el campo de dirección se refiere a un registro en lugar de a una localidad de memoria principal: EA = R.

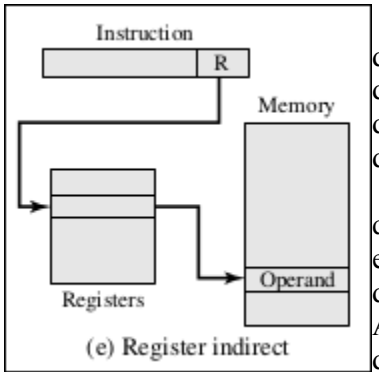
Para clarifica, si el contenido del campo de direcciones en una instrucción es 5, entonces R5 es la dirección deseada y el valor del operando se encuentra en R5. Típicamente, un campo de dirección que referencía registros tendrá de entre 3 y 5 bits, de manera que entre 8 y 32 registros de propósito general pueden ser referenciados.

Las ventajas del direccionamiento de registros son que sólo se necesita un pequeño campo de dirección y que no se requiere hacer referencias a memoria que consumen mucho tiempo. Como se ha discutido antes, el tiempo de acceso para un registro interno del procesador es mucho menor que aquel para una dirección de memoria principal.

La desventaja de esta propuesta es que el espacio de direcciones es súmamente límitado (8 a 32 registros únicamente). Si el direccionamiento de registros se utiliza frecuentemente en un conjunto de instrucciones, quiere decir que los registros del procesador serán muy usados. Debido al número severamente limitado de registros (comparado con localidades de memoria principal), su uso de esta forma sólo tiene sentido si se hace de manera muy eficiente. Si cada operando se lleva a un registro desde la memoria principal, se opera sobre él una vez, y luego se regresa a memoria principal, entonces se ha agregado un paso intermedio innecesario. Si, en lugar, el operando en un registro se mantiene en uso durante múltiples operaciones, se logran ahorros reales.

Es decisioń del programador o del complilador decidir qué valores deberán permanecer en registros y cuáles deberán ser almacenados en memoria principal. La mayoría de los procesadores modernos emplean múltiples registros de propósito general, lo cual pone la carga de una ejecución eficiente en el programador de lenguaje ensamblador.

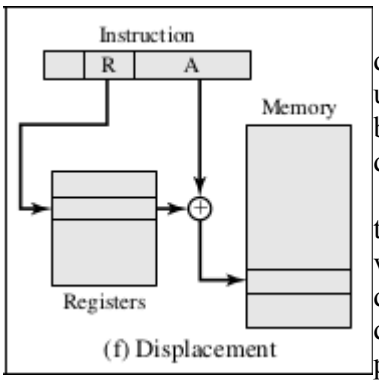
Direccionamiento Indirecto de Registro



Al igual que el direccionamiento de registro es análogo al direccionamiento directo, el direccionamiento indirecto de registro es análogo al direccionamiento indirecto. En ambos casos, la única diferencia es si el campo de dirección se refiere a una localidad de memoria o a un registro. Así, para el direccionamiento indirecto de registro: EA = (R).

Las ventajas y limitaciones de este enfoque son básicamente las mismas que para el direccionamiento indirecto. En ambos casos, la limitación del espacio de direcciones se supera al hacer que el campo se refiera a una localidad de memoria de tamaño de la palabra que contiene una dirección. Adicionalmente, el direccionamiento indirecto de registro utiliza una referencia de memoria menos que el direccionamiento indirecto.

Direccionamiento con Desplazamiento



Un modo muy poderoso que combina las capacidades del direccionamiento directo y direccionamiento indirecto de registro. Se conoce por una variedad de nombres dependiento del contexto de su uso, pero el mecanismo básico es el mismo. Nos referiremos a él como direccionamiento con desplazamiento: EA = A + (R)

El direccionamiento con desplazamiento requiere que una instrucción tenga dos campos de dirección, donde al menos uno de ellos es explícito. El valor contenido en uno de los campos de direcciones (valor = A) se utiliza directamente. El otro campo de direcciones, o una referencia implícita dependiendo del opcode, se refiere a un registro cuyo contenido se suma a A para producir la dirección efectiva.

A continuación se describen 3 de los usos más comunes del direccionamiento con desplazamiento.

Direccionamiento relativo

Para el direccionamiento relativo, el registro referenciado implícitamente es el contador de programa (PC). Es decir, la dirección de la siguiente instrucción se agrega al campo de dirección para producir la dirección efectiva del operando. Típicamente, el campo de dirección se trata como un número en complemento a dos para esta operación. Así, la dirección efectiva es un desplazamiento relativo a la dirección de la instrucción.

El direccionamiento relativo explota el concepto de localidad discutido anteriormente. Si muchas referencias a memoria están relativamente cerca de la instrucción en ejecución, entonces el uso de direccionamiento relativo ahorra bits en la instrucción.

Direccionamiento con registro base

El registro referenciado contiene una dirección de memoria principal, y el campo de dirección contiene un desplazamiento (usualmente un entero sin signo). La referencia al registro puede ser explícita o implícita. El direccionamiento con registro base también explota el principlio de localidad. En algunas implementaciones, un único registro se utiliza y es referenciado implícitamente. En otras, el programador puede elegir el registro que almacene la dirección base y la instrucción deberá referenciarlo explicitamente. En este último caso, si la longitud del campo de direcciones es K y el número de posibles registros es N, entonces una instrucción puede rreferenciar una de N areas de 2^K palabras.

Indexamiento

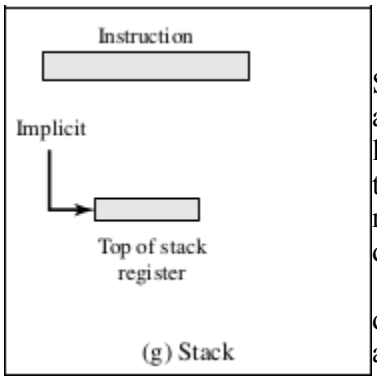
El campo de dirección referencía una dirección de memoria principal y el registro referenciado contiene un desplazamiento positivo desde dicha dirección. Note que este tipo de uso es el opuesto de la interpretación utilizada con registro base. Por supuesto, es cuestión de más que interpretación. Como el campo de dirección se considera una localidad de memoria, generalmente contiene más bits que un campo de dirección en una instrucción equivalente de registro base.

Un uso importante del indexamiento es proveer un mecanismo eficiente para realizar operaciones iterativas. Considere, por ejemplo, una lista de números almacenada en localidades de memoria que comienzan en A. Suponga que nos gustaría sumar 1 a cada elemento de la lista. Necesitamos obtener cada valor, sumarle 1, y volver a almacenarlo. La secuencia de direcciones efectivas que necesitamos es A, A + 1, A + 2, . . . , hasta la última localidad de la lista. Con indexamiento, esto se hace fácilmente. El valor A se almacena en el campo de dirección de la instrucción, y el registro elegido, llamado *registro índice* se inicializa en 0. Después de cada operación el registro índice se incrementa en 1.

Como los registros índice son comunmente utilizados para este tipo de tareas iterativas, es tipo que sea necesario incrementar o decrementar el registro índice después de cada referencia a él. Como es una operación

tan común, algunos sistemas lo hacen de manera automática como parte del mismo ciclo de instrucción. A ésto se le conoce como *autoindexado*. Si ciertos registros se dedican exclusivamente al indexado, entonces el autoindexado puede ser invocado implícitamente y automáticamente.

Direccionamiento con Stack



El último modo de direccionamiento es el llamado direccionamiento con Stack. Un estac es un arreglo lineal de localidades reservadas. Los elementos se agregan al tope del están de manera que, en cualquier momento, el bloque de localidades está parcialmente lleno. Un puntero cuyo valor es la dirección del tope está asociado con el stack. Dicho puntero se mantiene en un registro. Así, referencias a localidades del stack en memoria son de hecho direccionamientos de indirectos de registro.

El modo stack de direccionamiento se una forma implícita de direccionamiento. Las instrucciones máquina no necesitan incluir una referencia a memoria sino que implícitamente operan sobre el tope del stack.

10.2 Formatos de Instrucción

Un formato de instrucción define la estructura de los bits de una instrucción, en términos de los campos que lo constituyen. Un formato de instrucción debe incluír un opcode e, implícitamente o explícitamente, cero o más operandos. Cada operando explícito es referenciado usando uno de los modos de direccionamiento descritos en la sección previa. El formato debe indicar el modo de direccionamiento utilizado para cada operando. Para la mayoría de los conjuntos de instrucciones, más de un formato de instrucción es utilizado.

El diseño de un formato de instrucciones es un arte complejo, y una sorprendente variedad de diseños se han implementado. A continuación se discuten algunos de los aspectos principales que se deben de tomar en cuenta al realizar esta tarea.

Longitud de Instrucción

La decisión de diseño más básica es la longitud del formato de instrucción. Esta decisión afecta, y es afectada por, el tamaño de la memoria, la organización de la memoria, la estructura del bus, la complejidad del procesador y la velocidad del procesador. Esta decisión determina la riquesa y flexibilidad de la máquina para el programador.

Los intercambios más obvios son entre el deseo de tener un repertorio de instrucciones basto y poderoso y la necesidad de ahorrar espacio. Los programadores desean más opcodes, más operandos, más modos de direccionamiento y un mayor rango de direcciones. Más opcodes y más operandos hacen la vida fácil al programador porque se pueden escribir programas más cortos para lograr los objetivos deseados. Similarmente, más modos de direccionamiento dan al programador mayor flexibilidad para implementar ciertas funciones. Y, por supuesto, con el aumento de los tamaños de memoria y el uso creciente de la memoria virtual, los programadores desean ser capaces de direccionar espacios de memoria cada vez más grandes. Sin embargo, todos estos elementos (opcodes, operandos, modos de direccionamiento) requieren de bits y promueven el uso de instrucciones cada vez más grandes. Pero instrucciones grandes pueden ser desperdiciadas. Una instrucción de 64 bits ocupa el doble de espacio que una de 32 y no necesariamente es el doble de útil.

Más allá de esta consideración inicial, existen otras a tomar en cuenta. La longitud de instrucción debería ser igual a la longitud de las transferencias de memoria (en un sistema con bus, el ancho del bus) o una debería ser múltiplo de la otra. De otra manera, no se obtendrá un número entero de instrucciones por ciclo. Una consideración relacionada es la tasa de transferencia de memoria. La memoria se puede convertir en un cuello de botella si el procesador puede ejectuar las instrucciones más rápido de lo que puede obtenerlas. Una solución a este problema es utilizar memoria caché, otra es utilizar instrucciones más pequeñas. Las instrucciones de 16 bits pueden ser obtenidas lo doble de rápido que las de 32 bits pero probablemente no se puedan ejecutar el doble de rápido.

Distribución de los bits

La decisión sobre cómo distribuir los bits de una instrucción en el formato es muy difícil pues los compromisos/intercambios involucrados son bastante complejos. Para una longitud de instrucción determinada, existe un intercambio claro entre el número de códigos de operación y el poder de direccionamiento. Aumentar el número de opcodes significa más bits para dicho propósito y, a su vez, menos para el direccionamiento de operandos.

Existe un refinamiento interesante para esta cuestión, el uso de opcodes de longitud variable. En esta propuesta, existe una longitud mínima para los opcodes pero, para algunos de ellos, se pueden especificar operaciones adicionales utilizando bits adicionales de la instrucción. Esta opción se utiliza para aquellas instrucciones que requieren menos operandos o direccionamiento menos poderoso.

Los siguientes factores interrelacionados determinan el uso de los bits de direccionamiento:

- **Número de modos de direccionamiento:** algunas veces un modo de direccionamiento puede ser indicado de forma implícita. Por ejemplo, ciertos opcodes pueden siempre utilizar indexado. En otros casos, los modos de direccionamiento deben ser explícitos, y uno o más bits de modo son necesarios.
- **Número de operandos:** se ha mencionado que menos operandos provocan programas más largos. Las instrucciones típicas en máquinas contemporáneas proveen dos operandos. Cada dirección de operando en la instrucción puede requerir su propio indicador de modo, o el uso de un indicador de modo puede estar limitado a sólo uno de los campos de dirección.
- **Registro vs memoria:** una máquina debe tener registros de manera que los datos puedan ser llevados al procesador. Con un único registro visible al usuario, una dirección de operando es implícita y no consume bits de instrucción. Sin embargo, la programación con un sólo registro es incómoda y requiere de muchas instrucciones. Incluso con múltiples registros, sólo unos cuantos bits se necesitan para especificar el registro. Mientras más se pueda utilizar a los registros como referencias de operando, menos bits son necesarios. La mayoría de las máquinas contemporáneas tienen al menos 32 registros visibles al usuario.
- **Número de conjuntos de registros:** algunas arquitecturas, incluyendo la x86, tienen una colección de dos o más conjuntos especializados de registros (por ejemplo para almacenar datos o direcciones de desplazamiento). Una ventaja de ésto es que, para un número fijo de registros, una división funcional requiere menos bits para la instrucción. Por ejemplo, con dos conjuntos de 8 registros, sólo 3 bits se requieren para identificar un registro; el opcode o el registro de modo determinará qué conjunto de registros se está referenciando.
- **Rango de direcciones:** para direcciones que referencían a memoria, el rango de direcciones que pueden ser referenciadas está relacionado al número de bits de dirección. Como esto impone una limitación severa, el direccionamiento directo es raramente utilizado. Con direccionamiento con desplazamiento, el rango se abré hasta la longitud del registro de dirección. Si se requieren desplazamientos más largos se requerirá un mayor número de bits de dirección en la instrucción.
- **Granularidad de las direcciones:** para direcciones que referencian a memoria en lugar de registros, otro factor es la granularidad del direccionamiento. En un sistema con palabras de 16 o 32 bits, una dirección puede referenciar una palabra o un byte según la decisión del diseñador. El direccionamiento por byte es conveniente para la manipulación de caracteres pero requiere de más bits de dirección.

Así, el diseñador encara diversos factores a considerar y balancear. Qué tan críticas son las opciones no es muy claro. Como ejemplo, un estudio que comparó varios enfoques de formatos de instrucción, incluyendo el uso del stack, registros de propósito general, un acumulador, y únicamente memoria-a-registro, utilizando un conjunto consistente de asumpsiones, no observó una diferencia significativa en espacio de código o tiempo de ejecución.

A continuación se presentan dos ejemplos históricos para observar cómo balancean los factores mencionados anteriormente.

PDP-8

Uno de los diseños de instrucción más simples para una computadora de propósito general. Utiliza instrucciones de 12-bits y opera con palabras de 12 bits. Utiliza un único registro de propósito general, el acumulador.

A pesar de las limitaciones del diseño, el direccionamiento es bastante flexible. Cada referencia a memoria consiste de 7 bits más 2 modificadores de 1 bit. La memoria se divide en páginas de longitud fija de 128 palabras cada una. El cálculo de una dirección se basa en referencias a la página 0 o la página actual, determinado por un bit de página. El segundo bit modificador indica si se utiliza direccionamiento directo o indirecto. Estos dos modos se utilizan en combinación, de manera que una dirección indirecta es una dirección de 12 bits contenida en una palabra de la página 0 o de la página actual. En adición, 8 palabras dedicadas en la página 0 son registros de autoindexamiento.

La figura siguiente muestra el formato de instrucción de la PDP-8. Existe un opcode de 3 bits y 3 tipos de instrucciones. Para los opcodes 0-5, el formato de instrucción es de una sóla referencia amemoria con un bit de página y un bit de indirección. Únicamente existen 6 operaciones básicas. Para aumentar el grupo de operaciones, el opcode 7 define una referencia a registro o *microinstrucción***.** En este formato, los bits restantes se utilizan para codificar operaciones adicionales. En general, cada bit define una operación específica y estos bits pueden ser cominados en una única instrucción.

El opcode 6 es una operación I/O: 6 bits se utilizan para seleccionar uno de los 64 dispositivos, y 3 bits especifican un comando I/O en particular.

El formato de instrucción de la PDP-8 es muy eficiente. Soporta direccionamiento indirecto, direccionamiento por desplazamiento e indexamiento. Con el uso de la extensión de opcode, soporta un total de aproximadamente 35 instrucciones. Dadas las las limitaciones de la longitud de 12 bits, difícilmente lo podrían haber hecho mejor.

PDP-10

En constraste con el conjunto de instrucciones de la PDP-8 el PDP-10 fue diseñado para ser un sistema de tiempo compartido a gran escala, con énfasis en que fuera fácil de programar, inclúso si costo adicional de hardware era necesario.

Entre los principios de diseño empleados en la implementación de su conjunto de instrucciones fueron:

- **Ortogonalidad:** es un principio por el cual dos variables son independientes una de la otra. En el contexto de un conjunto de instrucciones, el término indica que los elementos de una instrucción son independientes del opcode. Los diseñadores usaron el término para describir el hecho de que una dirección siempre se calcula de la misma manera, independientemente del opcode.
- **Completitud:** cada tipo de datos aritmético (entero, punto fijo, punto flotante) debería tener un conjunto de operaciones completo e idéntico.
- **Direccionamiento directo:** en lugar de utilizar un direccionamiento que dificultara la tarea al programador, se utilizó direccionamiento directo.

Memory reference instructions

Opcode			D/I	Z/C	Displacement						
0		2	3	4	5						11

Input/output instructions

1	1	0	Device							Opcode	
0		2	3						8	9	11

Register reference instructions

Group 1 microinstructions

1	1	1	0	CLA	CLL	CMA	CML	RAR	RAL	BSW	IAC
0	1	2	3	4	5	6	7	8	9	10	11

Group 2 microinstructions

1	1	1	0	CLA	SMA	SZA	SNL	RSS	OSR	HLT	0
0	1	2	3	4	5	6	7	8	9	10	11

Group 3 microinstructions

1	1	1	0	CLA	MQA	0	MQL	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11

D/I = Direct/Indirect address
Z/C = Page 0 or Current page
CLA = Clear Accumulator
CLL = Clear Link
CMA = CoMplement Accumulator
CML = CoMplement Link
RAR = Rotate Accumulator Right
RAL = Rotate Accumulator Left
BSW = Byte SWap
IAC = Increment ACcumulator
SMA = Skip on Minus Accumulator
SZA = Skip on Zero Accumulator
SNL = Skip on Nonzero Link
RSS = Reverse Skip Sense
OSR = Or with Switch Register
HLT = HaLT
MQA = Multiplier Quotient into Accumulator
MQL = Multiplier Quotient Load

Figure PDP-8 Instruction Formats

Cada uno de estos principios tenía como finalidad la facilidad para programar. La PDP-10 tiene una longitud de palabra de 36 bits y una longitud de instrucción de 36 bits. El formato se muestra en la figura siguiente.

Opcode		Register		I	Index register		Memory address		
0	8	9	12		14	17	18		35

I = indirect bit

El opcode ocupa 9 bits, permitiendo 512 operacioens. De hecho, un total de 365 instrucciones diferentes fueron definidas. La mayoría de las instrucciones tienen dos direcciones, una de las cuales es uno de los 16 resitros de propósito general. Así, este operando ocupa sólo 4 bits. El otro operando comienza con un campo de dirección de memoria de 18 bits. Este puede ser usado como un operando inmediato o una dirección de memoria. Posteriormente, tanto indexado como direccionamiento indirecto se permitieron. Los mismos registros de propósito general pueden ser usados como registros base.

Una longitud de instrucción de 36 bits es un lujo: no es necesario hacer trucos para obtener maś opcodes, 9 bits son más que necesarios. Un campo de dirección de 18 bits permite que el direccionamiento directo sea adecuado, incluso el direccionamiento inmediato se vuelve atractivo.

El conjunto de instrucciones cumple con los objetivos listados anteriormente. Facilita la tarea al programador con el costo de una utilización ineficiente del espacio, pero ésta fue una decisión consciente de los diseñadores y nose puede considerar como un diseño pobre.

Instrucciones de Longitud Variable

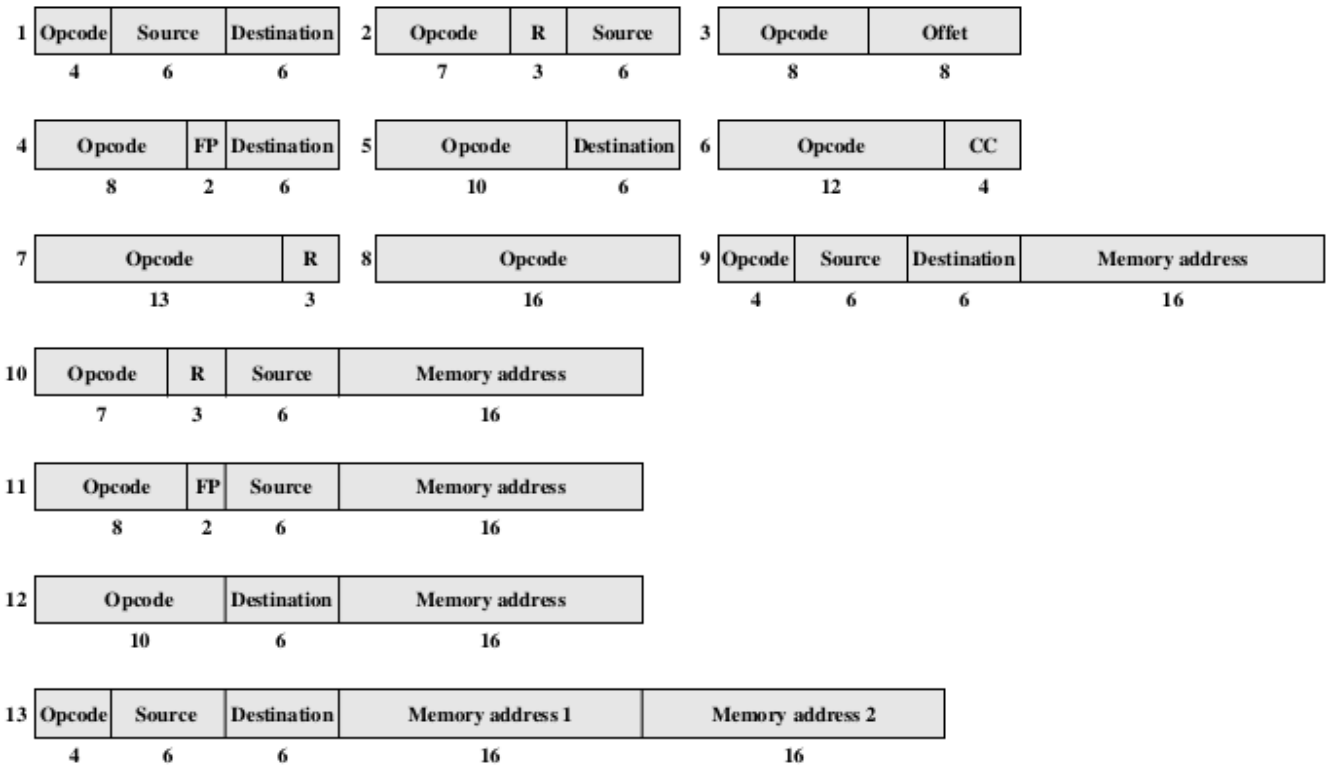
Los ejemplos presentados hasta ahora han utilizado una longitud de instrucción fija, pero el diseñador puede elegir proveer una variedad de formatos de instrucción con diferentes longitudes. Esta táctica hace fácil el proveer un gran repertorio de opcodes, con diferentes longitudes. El direccionamiento puede ser más flexible, con varias combinaciones de referencias a registros y memoria y diferentes modos de direccionamiento.

El precio principal a pagar por tener instrucciones de longitud variable es un aumento en la complejidad del procesador. La caída de los precios del hardware, el uso de la microprogramación, y el aumento general del entendimiento de los principios del diseño de procesadores han contribuido en que sea un pequeño precio a pagar. Sin embargo, las máquinas RISC y superescalares pueden explotar el uso de instrucciones de longitud fija para proveer un desempeño mejorado.

El uso de instrucciones de longitud variable no elimina el deseo de hacer que todas las longitudes de instrucción estén relacionadas con la longitud de palabra. Debido a que el procesador no conoce la longitud de la siguiente instrucción a ser obtenida, una estrategia típica es obtener un número de bytes o palabras que sea al menos igual a la longitud de la instrucción más grande. Esto implica que algunas veces se obtienen instrucciones múltiples, lo cual es una buena estrategia a seguir.

PDP-11

La PDP-11 fue diseñada para proveer un conjunto de instrucciones poderoso y flexible dentro de las limitaciones de una minicomputadora de 16 bits. Utiliza un conjunto de 8 registros de propósito general de 16 bits. Dos de estos registros tienen un significado adicional: uno se utiliza como apuntador de stack para operaciones especiales y otro se utiliza como contador de programa. La figura siguiente muestra los 13 formatos de instrucción utilizados, que incluyen instrucciones de cero, una y dos direcciones.



El opcode puede variar de 4 a 16 bits en longitud. Las referencias a registro son de 6 bits. Tres bits identifican el registro y los restantes 3 identifican el modo de direccionamiento. La PDP-11 fue proveída con un conjunto importante de modos de direccionamiento. Una ventaja de enlacar el modo de direccionamiento con el operando en lugar del opcode es que cualquier modo de direccionamiento puede ser utilizado con cualquier opcode (ortogonalidad).

Las instrucciones tienen usualmente una palabra de longitud. Para algunas instrucciones, se agregan una o dos direcciones de memoria, de manera que instrucciones de 32 y 48 bits son parte del repertorio. Ésto provee mayor flexibilidad al direccionamiento. Si bien el conjunto de instrucciones y la capacidad de direccionamiento son complejos, lo cual aumenta el costo del hardware y la complejidad de programación. La ventaja es que se pueden crear programas más eficientes y compactos.

VAX

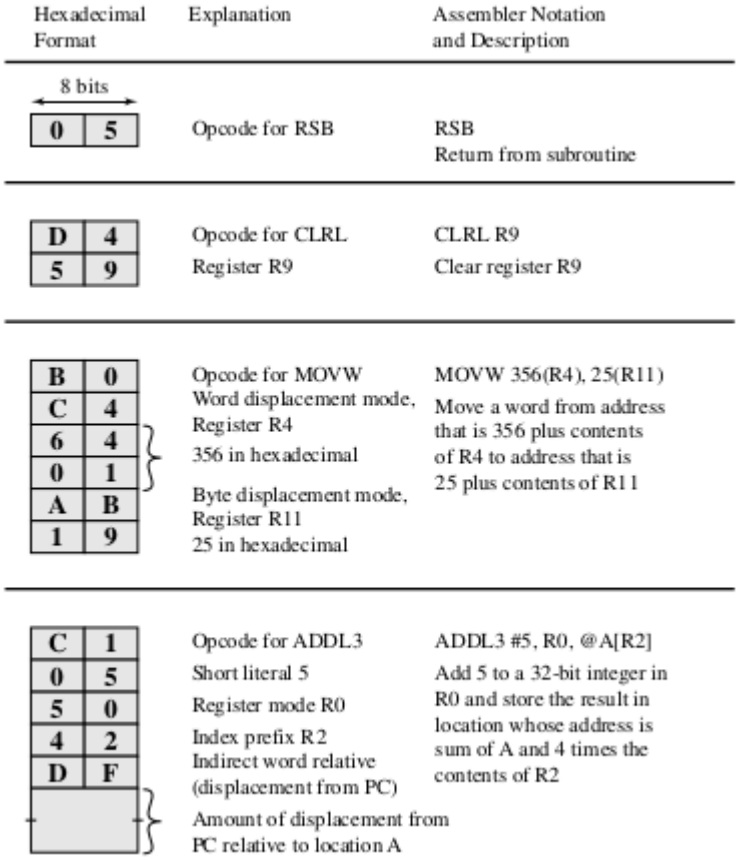
La mayoría de las arquitecturas proveen un número relativamente pequeño de formatos fijos de instrucción. Esto puede causar dos problemas para el programador. Primero, el modo de direccionamiento y el opcode no son ortogonales, por ejemplo, para cierta operación un operando siempre debe venir de un registro y el otro de memoria, o ambos de registro, etc. Segundo, sólo un número limitado de operandos puede ser incluído: típicamente dos o tres. Como algunas operaciones inherentemente requiere más operandos, varias estrategias se debn utilizar para lograr los resultados deseados usando dos o más instrucciones.

Para evitar estos problemas, dos criterios se utilizaron al diseñar el formato de instrucción de la VAX:

1. Todas las instrucciones deben tener el número “natural” de operandos.
2. Todos los operandos deben tener la misma generalidad en su espicificación.

El resultado es un formato de instrucción altamente variable. Una instrucción consiste de un código de

operación de 1 o 2 bytes seguidos de entre cero y seis especificadores de operandos, dependiendo del opcode. La longitud mínima de una instrucción es 1 byte, e instrucciones de hasta 37 bytes pueden ser construídas. En la siguiente figura se presentan algunos ejemplos:



La instrucción VAX comienza con un opcode de 1 byte. Esto permite manejar hasta 255 instrucciones. Sin embargo, existen 300 instrucciones diferentes, 8 bits no son suficientes. Los códigos hexadecimales FD y FF indican un opcode extendido, donde el opcode real se especifica en el segundo byte. El resto de la instrucción consiste de hasta 6 especificadores de operando. Un operando tiene, al menos, un formato de 1 byte en el cual los 4 de bits de más a la izquierda se utilizan como especificador de modo de direccionamiento. La única excepción a esta regla es el modo literal, el cual se señala con el patrón 00 en los 2 bits de más a la izquierda, dejando espacio para 6 bits literales. Debido a esta excepción un total de 12 diferentes modos de direccionamiento pueden ser especificados.

Un operando consiste usualmente de sólo un byte, con los 4 bits de más a la derecha especifiando uno de los 16 registros de propósito general. La longitud del especificador de operando puede extenderse de dos maneras. Primero, un valor constante de uno o más bytes puede seguir de manera inmediata al primer byte. Un ejemplo de esto es el modo de desplazamiento, en el cual un desplazamiento de 8, 16 o 32 bits es utilizado. Segundo, un modo deindexamiento puede ser usado. En este caso, el primer byte del especificador de operando consiste del código de direccionamiento 0100 (4) seguido por in identificador de registro índice de 4 bits. El resto del especificador del operando consiste del identificador de la dirección base, el cuál, a su vez, puede tener más de un byte de longitud.

El conjunto de instrucciones de la VAX provee una amplia variedad de operaciones y modos de direccionamiento. Esto da al programador una herramienta poderosa y flexible. En teoría, ésto debería conducir a complilaciones de lenguaje máquina muy eficientes y, en general, al uso eficiente de los recursos del procesador. El precio a pagar por estos beneficios es la complejidad del procesador comparado con un procesador con un formato de instrucciones más simple.

10.3 Lenguaje Ensamblador