Unidad 5

Instrucciones: concepto y formatos. Lenguaje de máquina y assembly. Conjunto de instrucciones: operaciones, formatos y modos de direccionamiento.

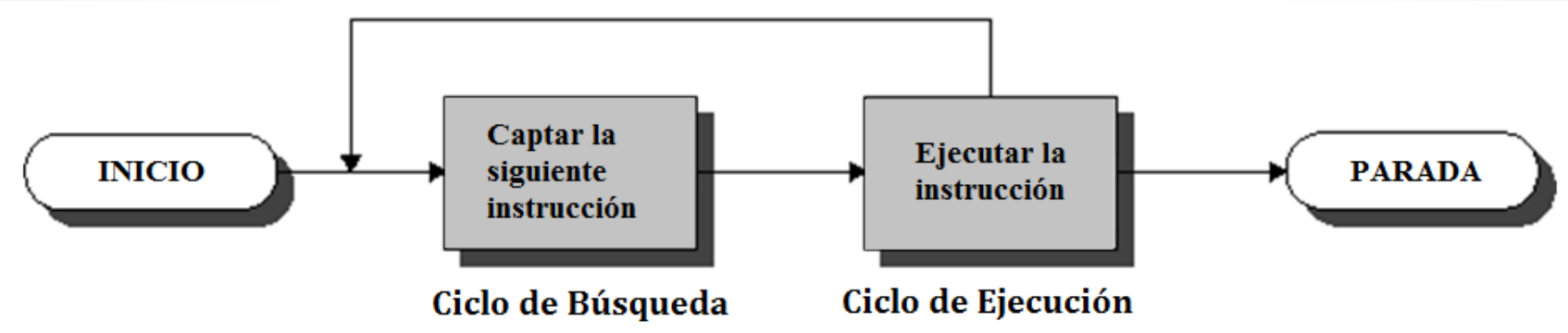
Introducción

Objetivos:

- Entender qué hace una instrucción
- Examinar y reconocer los distintos tipos de operandos.
- Reconocer los distintos tipos de operaciones que pueden especificarse mediante instrucciones máquina.

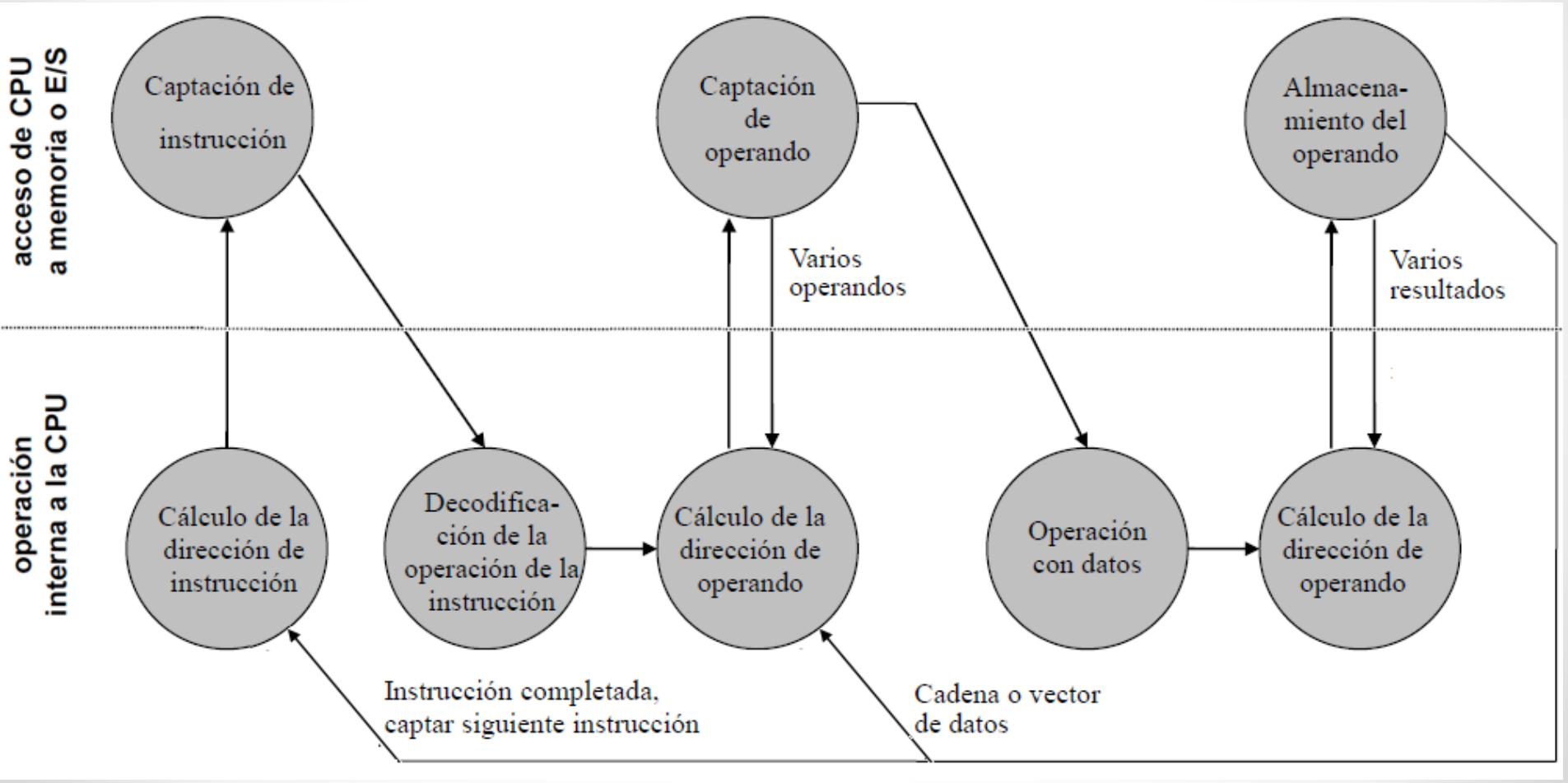
Repertorio de instrucciones

- El funcionamiento de la CPU esta determinado por las instrucciones que ejecuta.
- El conjunto de instrucciones distintas que la CPU puede ejecutar se denomina **repertorio de instrucciones**.
- Cada instrucción tiene:
  - Un *ciclo de búsqueda* (igual para todas)
  - Un *ciclo de ejecución* (distinto para cada una)



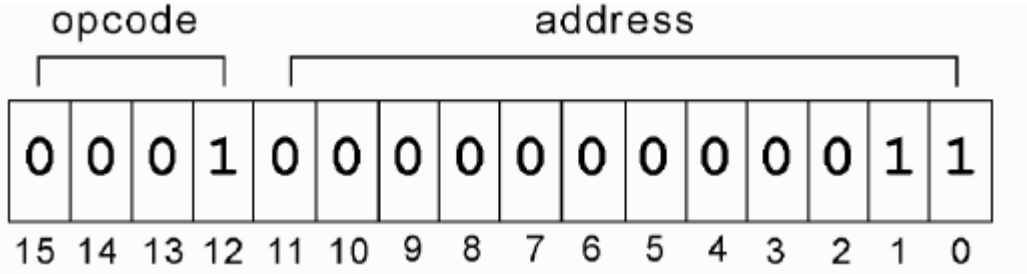
Repertorio de instrucciones

- O en una forma más detallada:



Repertorio de instrucciones

- Cada instrucción *de máquina* es una cadena de bits.
- Está dividida en campos, en los que la información está codificada. Por ejemplo:



- El código de operación ("*opcode*") dice ***qué*** se hace *("Cargar el registro B").*
- La dirección ("*address*") indica ***donde está*** el operando *("el contenido de la dirección 3_{10}").*

Repertorio de instrucciones

- Este esquema se denomina **formato de instrucción.** Cada repertorio emplea en general más de un formato.
- Durante su ejecución la instrucción se escribe en el Registro de Instrucción (IR).
- La Unidad de Control extrae y decodifica los datos de los distintos campos para realizar la operación.
- Cada instrucción **debe contener la información que necesita la CPU para su ejecución**. Un formato *general* de una instrucción es:

Código de Operación	Dirección próxima instrucción	Referencia a Operando fuente 1	Referencia a Operando fuente 2	Referencia a Operando resultado

Representación de instrucciones

- Sin embargo, es difícil para el programador tratar con las representaciones binarias de las instrucciones de máquina (Una simplificación es usar códigos hexadecimales)
- Sin embargo, es más manejable utilizar una representación simbólica de la instrucción.
- Los **códigos de operación** se representan por medio de abreviaturas, llamadas *mnemónicos* que indican la operación. Por ejemplo:
  - ADD - adición (suma)
  - SUB - sustracción (resta)
  - MOV - movimiento de un dato
  - AND, OR, XOR - operaciones lógicas

Representación de instrucciones

- Los operandos también se pueden representar de manera simbólica. Por ejemplo:

MOV reg1 , [dir1]

- La instrucción copia (mueve) el valor contenido en la posición de memoria llamada **dir1**, a un registro de la CPU denominado **reg1**.

- Esto da lugar al ***Lenguaje Ensamblador*** o ***Assembly:***
  - Textual (usa mnemónicos, no códigos numéricos)
  - Permite usar *etiquetas* y *directivas*

Lenguaje Ensamblador

Por ejemplo (1):

Inicio: MOV R0, 0x0010
MOV R1, 0x0001
ADD R0, R1
JMP **Inicio**

- Aquí, **Inicio** es una *etiqueta*

Por ejemplo (2):

V1: DW 0x001F
V2: DW 0x0FF0
ADD R0, [V1]
ADD R1, [V2]

- DW ("*Define Word*") es una *directiva* al ensamblador

Programa Ensamblador

- El ***programa*** llamado Ensamblador toma el código fuente y lo traduce a un código máquina
- Para esto, realiza los siguientes pasos:
  - Resuelve las directivas dirigidas al Ensamblador.
  - Calcula el tamaño de cada una de las instrucciones, y en base a ello resuelve los valores de las etiquetas.
  - Traduce todas las instrucciones a ceros y unos de acuerdo al formato de la instrucción.

Alto nivel / bajo nivel

- Un lenguaje de alto nivel expresa las instrucciones de forma algebraica concisa utilizando variables: Por ej.:

$$X = X + Y$$

- Un lenguaje a nivel máquina expresa las operaciones de manera elemental, implicando operaciones de transferencia de datos a o desde registros.
- Para el ejemplo anterior, pueden ser necesarias tres instrucciones máquina:

  1) Cargar un registro con el contenido de la posición de memoria X

  2) Sumar al registro el contenido de la posición de memoria Y y almacenar el resultado en otro registro.

  3) Memorizar el contenido de este registro en la posición de memoria X.

Alto nivel / bajo nivel

- En general, una instrucción de alto nivel puede requerir varias instrucciones de máquina.
- El lenguaje de alto nivel expresa operaciones en forma concisa usando variables.
- El lenguaje de máquina expresa las operaciones en forma básica involucrando movimiento de datos y uso de registros
- Cualquier programa escrito en lenguaje de alto nivel se debe convertir a un lenguaje de máquina para ser ejecutado.
- El conjunto de instrucciones de máquina debe ser capaz de expresar cualquiera de las instrucciones de un lenguaje de alto nivel.

Elementos de una Instrucción

- **Código de operación** (CodOp)
  Especifica la operación a realizar (suma, E/S, etc.)
- **Dirección de la siguiente instrucción**
  Indica a la CPU de donde captar la siguiente instrucción tras completarse la ejecución de la instrucción actual.
- **Referencia a operandos fuente**
  La operación puede implicar uno o más operandos fuente, que son entradas para la instrucción.
- **Referencia al operando resultado:**
  La operación puede producir un resultado, salida de la instrucción

Los operandos pueden ser un **registro**, una **posición de memoria**, o un **dispositivo de E/S**. Las referencias a cualquiera de ellos se denomina *dirección.*

Número de direcciones de una instrucción (1)

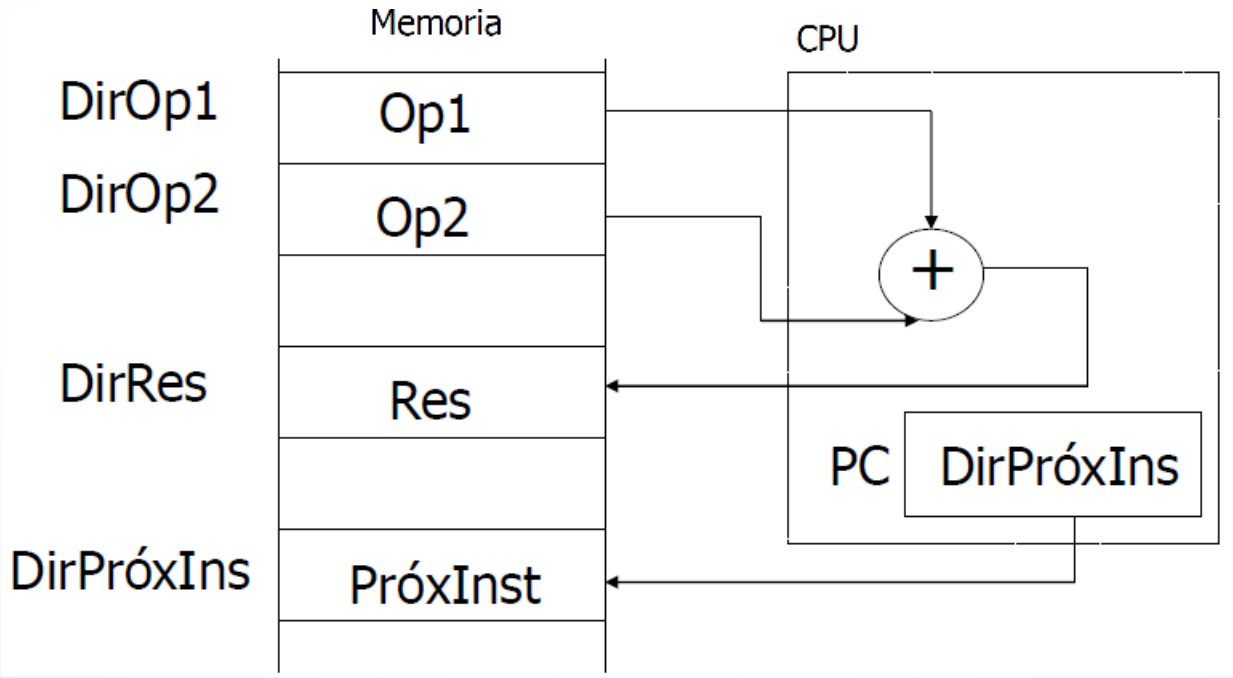
Código de Operación	Dirección próxima instrucción	Referencia a Operando fuente 1	Referencia a Operando fuente 2	Referencia a Operando resultado

- En el esquema anterior, cada campo necesita un número determinado de bits, cuya suma dará la longitud total de la instrucción
- Esto significa que la instrucción será más “larga” y ocupará más espacio en memoria, cuanto más campos explícitos tenga.
- Una primera opción es eliminar el campo de “dirección de la próxima instrucción”, haciéndola implícita mediante el uso del registro Contador de Programa (PC).
- Para los restantes, resultan las siguientes opciones, tomando como ejemplo una operación de suma (CodOp = ADD):

Instrucciones de tres direcciones

Add DirRes, DirOp1, DirOp2

Add	DirRes	DirOp1	DirOp2

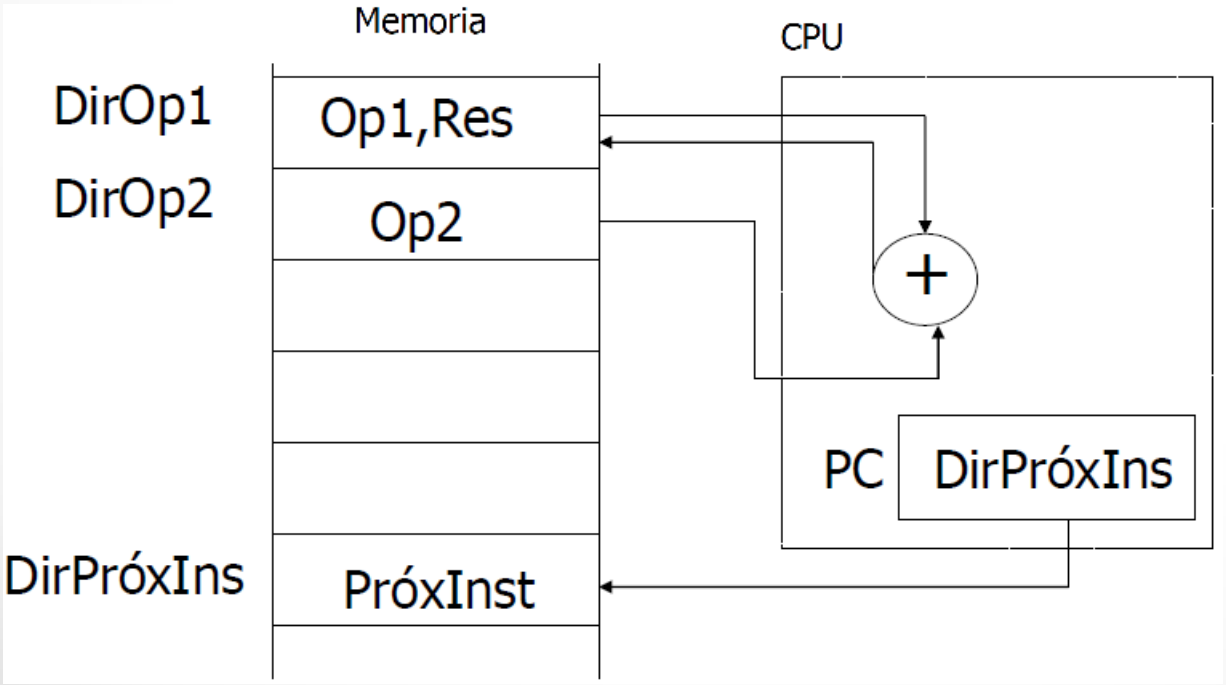


Instrucciones de dos direcciones

Add, DirOp1, DirOp2

Add	DirOp1	DirOp2

- Una de las direcciones debe hacer el servicio doble de uno de los operandos y del resultado.
- Requiere almacenamiento temporario para algún operando
- Reduce la longitud de la instrucción

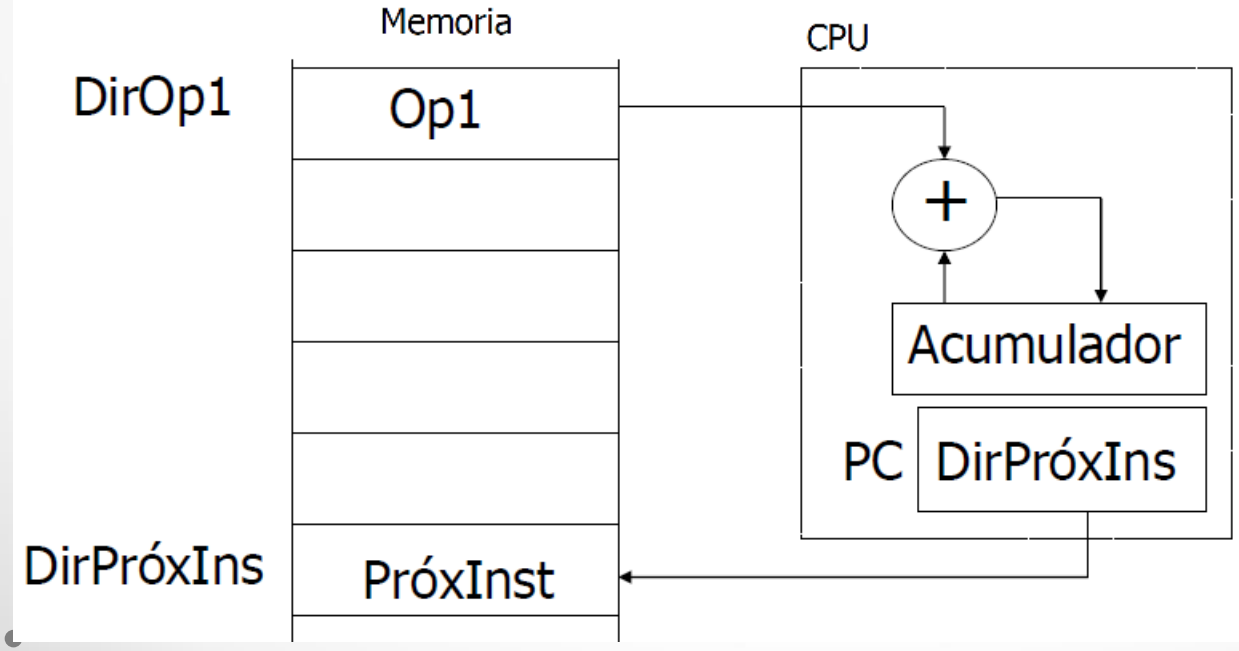


Instrucciones de una dirección

Add DirOp1

Add	DirOp1

- Una segunda dirección debe estar implícita.
- Usualmente es el registro acumulador (AC), que funciona como fuente de un operando y destino.



Instrucciones de 0 (cero) direcciones

- Todas las direcciones son implícitas
- Usan una pila, una estructura en memoria del tipo last-in-first-out (el ultimo en entrar es primero en salir).
- Ej.: c = a + b

```
push a
push b
add
pop c
```

¿Cúantas direcciones por instrucción? (1)

- Es una decisión **básica de diseño.**
- Menos direcciones
  - CPU menos compleja
  - Instrucciones más cortas y simples
  - Más instrucciones por programa
  - Rápidas instrucciones de captación/ejecución de instrucciones
- Más direcciones
  - Instrucciones más complejas y largas
  - Permite disponer de un mayor numero de registros de uso general
    Las operaciones realizadas Inter-registros son mas rápidas
  - Menos instrucciones por programa
- Por razones de flexibilidad y facilidad para utilizar varios registros, las maquinas contemporáneas emplean una combinación de dos y de tres direcciones.

¿Cúantas direcciones por instrucción? (2)

- Otros aspectos a considerar
  - Si una dirección hace referencia a una **posición de memoria** o a **un registro.**

    Si hay menos registros, entonces se necesitan menos bits para referenciarlos
  - Una máquina puede permitir diversos modos de direccionamiento

    Dicha especificación consume uno o más bits.

  - **La mayoría de los diseños de CPU hacen uso de varios formatos de instrucciones**

Tipos de instrucciones

Las instrucciones de máquina se clasifican en:

- **Procesamiento de Datos**: instrucciones aritméticas y lógicas.
- **Almacenamiento de datos**: instrucciones de transferencias de información dentro del CPU y la memoria.
- **Entrada/salida**: instrucciones de transferencia entre la CPU/memoria y dispositivos de comunicación con el exterior.
- **Control**: Instrucciones de comprobación y de bifurcación.

Tipos de operandos

- Las instrucciones máquina operan con datos. Las categorías más importantes son:
- **Direcciones**

  En ocasiones debe realizarse algún cálculo sobre la referencia de un operando. Se consideran como números enteros sin signo.
- **Números**

  Enteros, punto fijo/flotante
- **Caracteres**

  ASCII, BCD, etc.
- **Datos lógicos**

  Bits o flags: una palabra se considera una unidad de *n* elementos o datos de 1 bit, donde cada elemento vale **0** o **1**. Permiten almacenar una matriz de datos binarios o booleanos, con lo que la memoria se utiliza más eficientemente, o bien permiten la manipulación de bits individuales de un dato.

Tipos de operaciones

En todas las máquinas se encuentran tipos generales de operaciones:

1) Transferencia de datos
2) Aritméticas
3) Lógicas
4) De conversión
5) De E/S
6) De control del sistema
7) De control de flujo.

Tipos de operaciones: Transferencia de datos

- Es el tipo más básico de instrucción; son las operaciones son las más sencillas que puede realizar la CPU.
- Debe especificar:
  - Posiciones del operando fuente
  - Posiciones del operando destino
    - Podrían ser de memoria, un registro o la cabecera de una pila.
  - Longitud de los datos a transferir
  - El modo de direccionamiento para cada operando.
- Si el origen y el destino son registros, la CPU hace la operación interna.
- Si uno o ambos operandos están en memoria → La CPU debe realizar alguna o todas las siguientes tareas:
  1. Calcular la dirección de memoria basándose en el modo de direccionamiento utilizado.
  2. Si la dirección hace referencia a memoria virtual: traducir de dirección virtual a real.
  3. Determinar si el elemento direccionado esta en cache.
  4. Si no, cursar la orden al modulo de memoria.

Tipos de operaciones: Aritméticas

- Suma, Resta, Multiplicación, División
- Existen siempre para enteros con signo (coma fija)
- A menudo se proporcionan para números en punto flotante y para decimales empaquetados
- Pueden incluir varias instrucciones de un solo operando
  - Increment (a++): incrementa en 1 el operando
  - Decrement (a--): decrementa en 1 el operando
  - Negate (-a): cambia el signo del operando
  - Absolute: obtiene el valor absoluto del operando
- Su ejecución puede implicar operaciones de transferencia de datos para ubicar los operandos como entradas a la ALU y para almacenar la salida de la ALU.
- Además la ALU debe realizar la operación deseada.

Tipos de operaciones: Lógicas

a) **Operaciones para manipular bits individuales dentro de una palabra o de otra unidad direccionable.**

- AND, OR, XOR, NOT
- Pueden aplicarse desde bit a bit hasta n bits
- Si dos registros contienen los siguiente:
- (R1)= 10100101 ; Contenido de la posición R1
- (R2)= 00001111 ; Contenido de la posición R2
- (R1) AND (R2)=00000101
- Esta operación puede utilizarse como mascara, para poner a 0 ciertos bits y seleccionar otros.

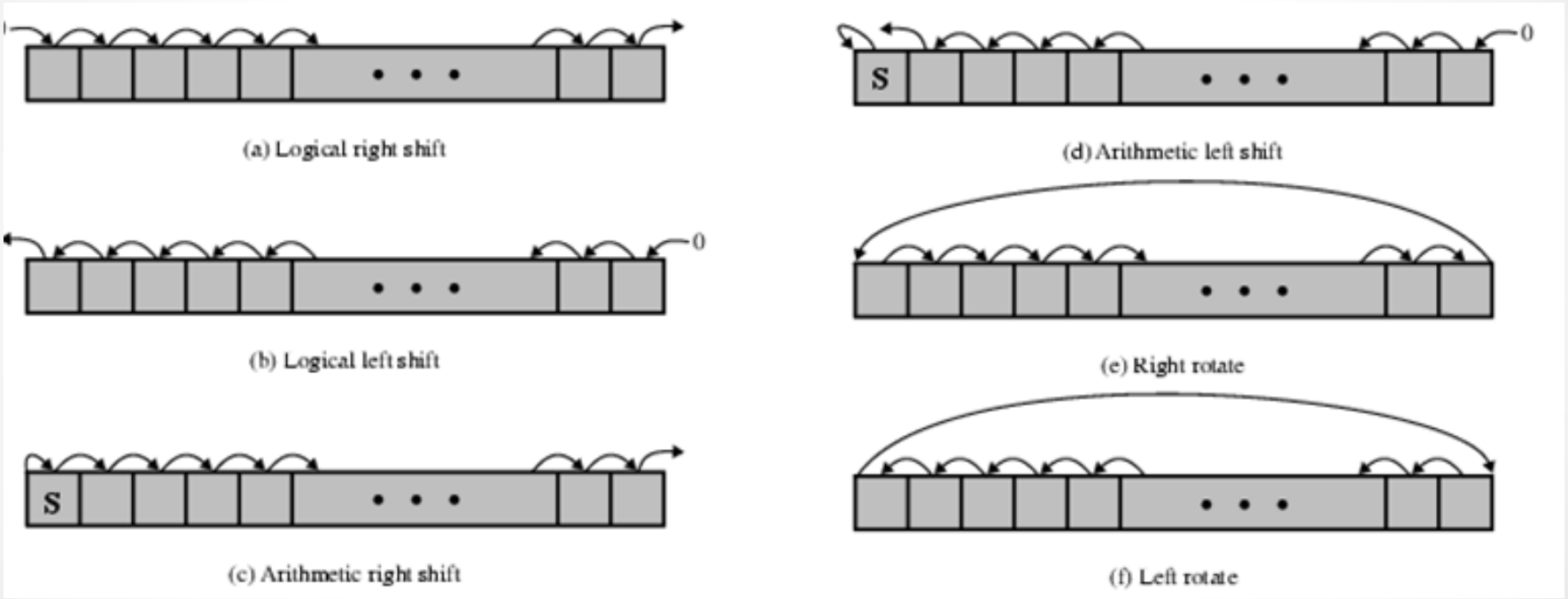
Tipos de operaciones: Lógicas

b) **Funciones de desplazamiento y rotación**.

- Desplazamiento lógico: se desplazan a la derecha o a la izquierda los bits de la palabra
- Desplazamiento aritmético: trata el dato como entero con signo, y no desplaza el bit de signo
- Rotación o desplazamiento cíclico.
- Preserva todos los bits con que se esta operando.

  Ej.: Ir volcando sucesivamente cada bit en la posición mas a la izquierda, donde puede ser identificado comprobando el bit de signo del dato.
- Las operaciones lógicas implican actividad de la ALU.

Operaciones de desplazamiento y de rotación



Tipos de operación: Conversión

- Cambian el formato u operan sobre el formato de los datos. Ej. Conversión de decimal a binario.
- Convierte un código de 8 bits a otro
  - TR R1, R2, L
  - Translate
  - R2: Contiene la dirección de comienzo de una tabla de códigos de 8 bits.
  - Se traducen los L bytes que comienzan en la dirección especificada por R1.
  - Se sustituye cada byte por el contenido del elemento de la tabla indexada por dicho byte.
  - Por ej.: Para convertir de EBCDIC a ASCII

Tipos de operaciones: Entrada/Salida

- Las instrucciones admiten aproximaciones muy diversas.
- Incluyen E/S programadas aisladas.
- Incluyen E/S programadas asignadas en memoria.
- Instrucciones de movimiento de datos (mapeo de memoria)
- Pueden ser hechas mediante un controlador separado (DMA).
- Muchas implementaciones ofrecen solo unas pocas instrucciones de E/S con acciones especificas indicadas mediante parámetros, códigos o palabras de órdenes.

Tipos de operaciones: Control del sistema

- Son instrucciones privilegiadas.
- La CPU necesita estar en un estado específico
- Están reservadas para ser usadas por el sistema operativo
- Ej: Leer o alterar un registro de control
- Ej.: para leer o modificar una clave de protección de memoria

Tipos de operaciones: Control de flujo o transferencia de control

- Existe una serie de instrucciones cuya misión es cambiar la secuencia de ejecución.
- La operación que realiza la CPU es actualizar el PC para que contenga la dirección de alguna instrucción que esta en memoria
- Varias razones para utilizar estas instrucciones:
  - Si se va a procesar una tabla o una lista de elementos → lo normal es utilizar un bucle de programa.
  - Los programas incluyen tomas de decisión
  - Para programas muy largos, se parte la tarea en trozos más pequeños.

Tipos de operaciones: Control de flujo o transferencia de control

a) Bifurcación (Salto). Ejemplo:

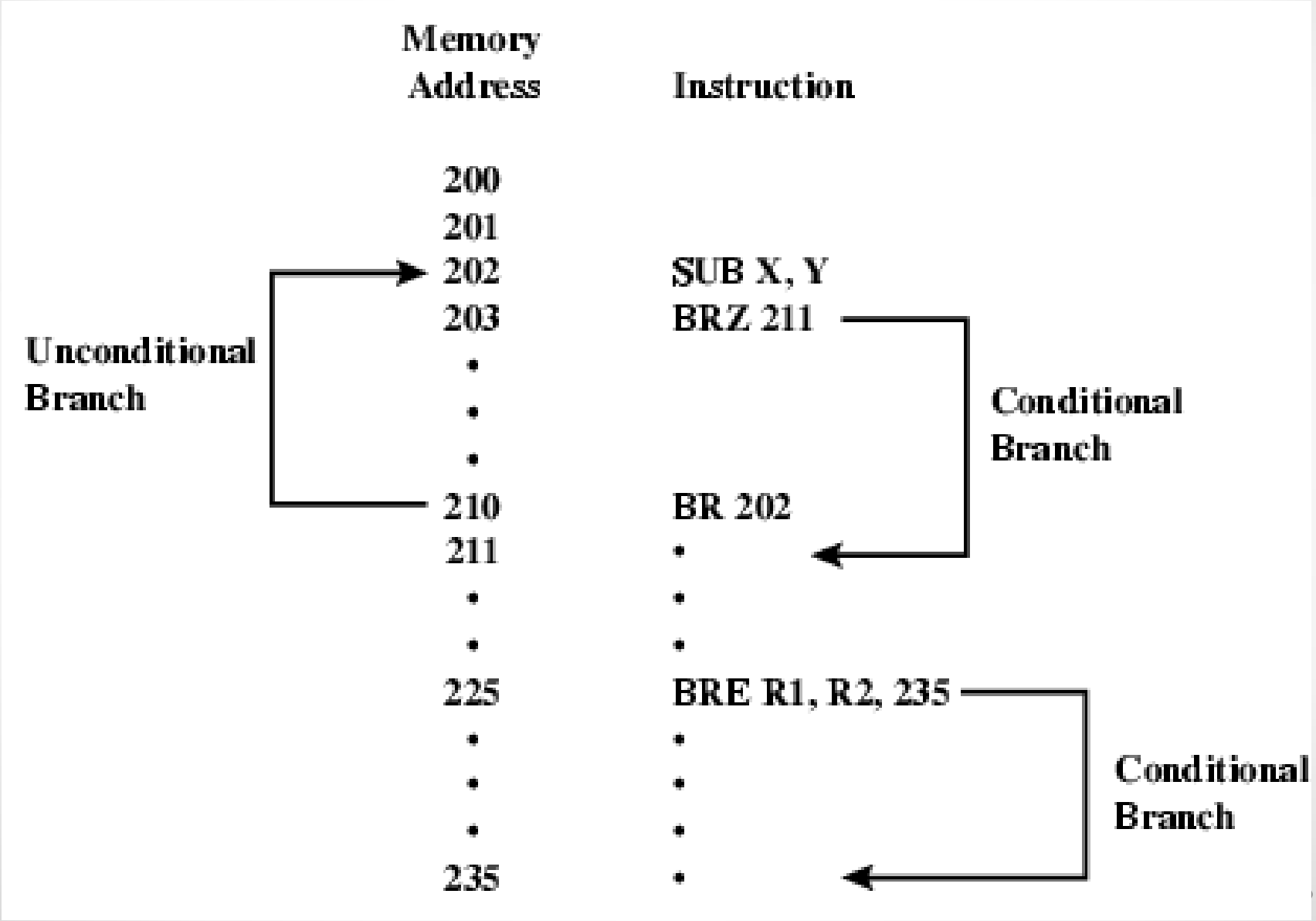
- Se ejecuta el salto (se actualiza el PC con la dirección especificada en el operando) si se cumple la condición.
- Si no se cumple se ejecuta la instrucción siguiente de la secuencia (se incrementa el PC de forma habitual).
  - BRZ X (saltar a X si el **resultado** es cero)
  - BRP X (saltar a la posición X, si el **resultado** es positivo)
  - BRN X (saltar a X si el **resultado** es negativo)
  - BR0 X (saltar a la posición X, si el **resultado** es desbordamiento)

  Resultado: referencia a la ultima operación ejecutada que afecte al código de condición.

  - Otro ejemplo con una instrucción de 3 direcciones:

    BRE R1, R2, X Saltar a X si el contenido de R1 = R2

Instrucciones de bifurcación



Tipos de operaciones: Control de flujo o transferencia de control

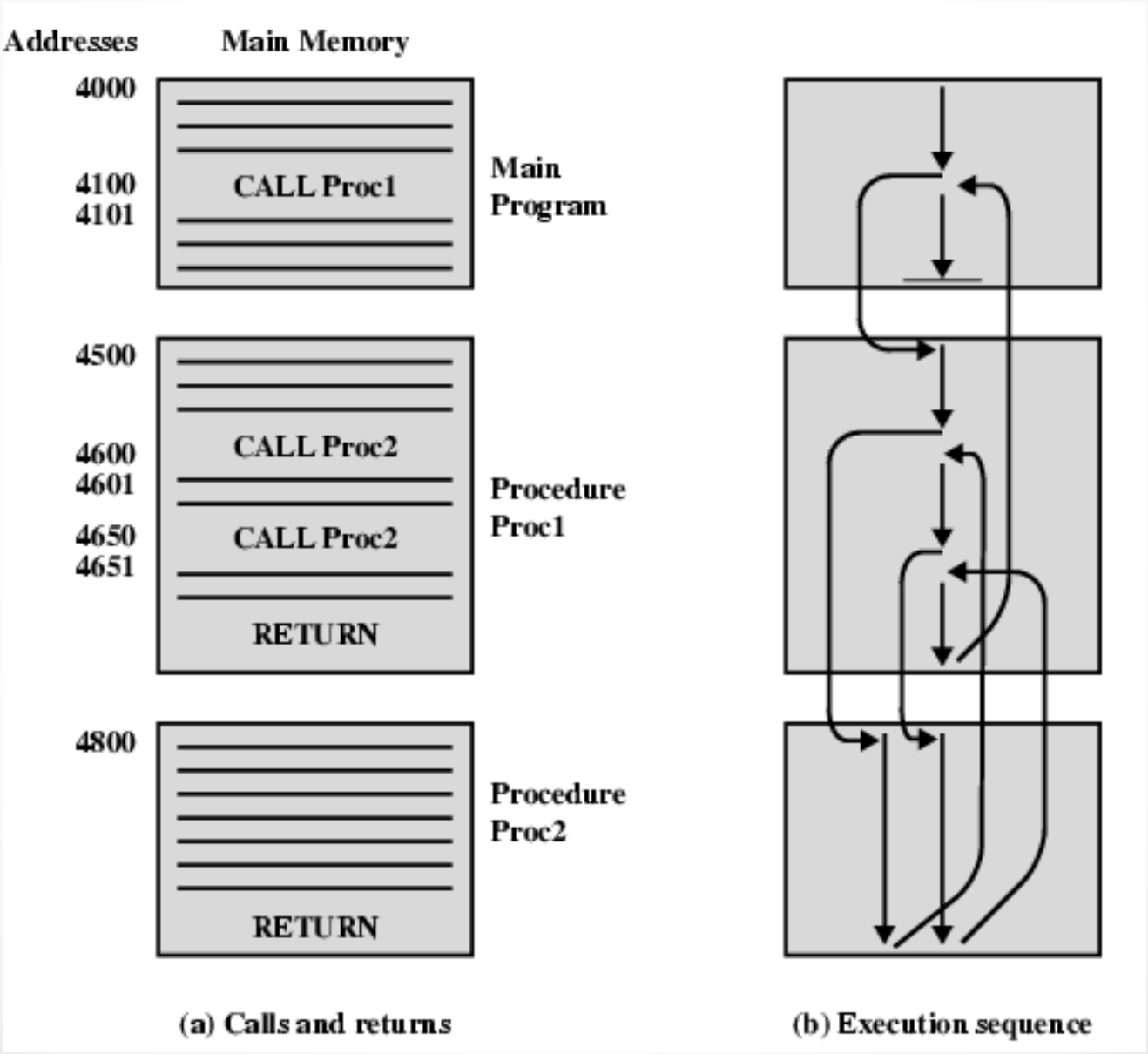
b) Salto implícito ("SKIP"). Incluye una dirección de manera implícita. Implica que se va a saltar una instrucción

- La dirección implícita es igual a la dirección de la siguiente instrucción mas la longitud de una instrucción.
- Ej.: Instrucción "incrementar y saltar si es cero" (ISZ)

c) Llamada a procedimiento

- Procedimiento : programa autoconsistente que se incorpora en uno mas grande.
- Razones de su uso: economía y modularidad.
- Dos instrucciones básicas: "Call", "Return".

Procedimientos anidados



Tipos de operaciones: Control de flujo o transferencia de control

- Una llamada a procedimiento puede hacerse desde distintos puntos → la CPU debe guardar la dirección de retorno en algún sitio.
- Existen 3 lugares habituales:
  - Un registro
  - Al principio de procedimiento
  - En la cabecera de una pila: es la opción mas potente y más general

Direccionamiento

- Como se vio, una instrucción utiliza un campo de bits para expresar el código de operación: lo que hace. Pero también necesita una importante cantidad de bits para especificar de dónde provienen los datos: **direccionarlos**.
- Es importante reducir el tamaño de esa referencia.
- Hay dos métodos generales para lograrlo:

  1) Si un operando va a usarse varias veces puede colocarse en un registro.
  - el acceso es más rápido.
  - se necesitan menos bits para referenciar un registro que para una posición de memoria.

  2) Referenciar uno o más operandos en forma implícita.

Dependiendo del caso, el programador podrá hacer uso de estas posibilidades, utilizando los **modos de direccionamiento** que le ofrezca el repertorio de instrucciones.

Modos de direccionamiento

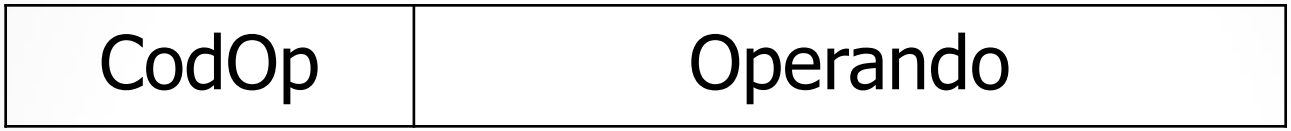
- Los modos de direccionamiento son distintas formas en que una misma operación puede ejecutarse para reducir el número de bits de la instrucción o hacer un manejo más eficiente de los datos.
- Por lo tanto, una misma operación puede dar lugar a varias instrucciones, con un modo de direccionamiento diferente:
    1. Inmediato
    2. Directo
    3. Indirecto
    4. Registro
    5. Indirecto con registro
    6. Con desplazamiento (Indexado)
    7. Pila

Modos de direccionamiento

- La Unidad de Control determina el modo de direccionamiento que utiliza cada instrucción mediante:
  - CodOps diferentes, o bien,
  - Uno o más bits en un campo de modo, que indica que tipo de direccionamiento se emplea
- Notación:
  - A = Contenido de un campo de dirección en la instrucción.
  - R = Contenido de un campo de dirección en la instrucción que referencia un registro.
  - EA = Dirección real (efectiva) de la posición de memoria que contiene el operando que se referencia.
  - (X) = Contenido de la posición X.

Direccionamiento inmediato (1)

- Modo de direccionamiento inmediato:



- Este modo puede utilizarse para definir y utilizar constantes, o para fijar valores iniciales de variables.
- Normalmente el número se almacena en complemento a dos; el bit más a la izquierda del campo operando se utiliza como bit de signo.

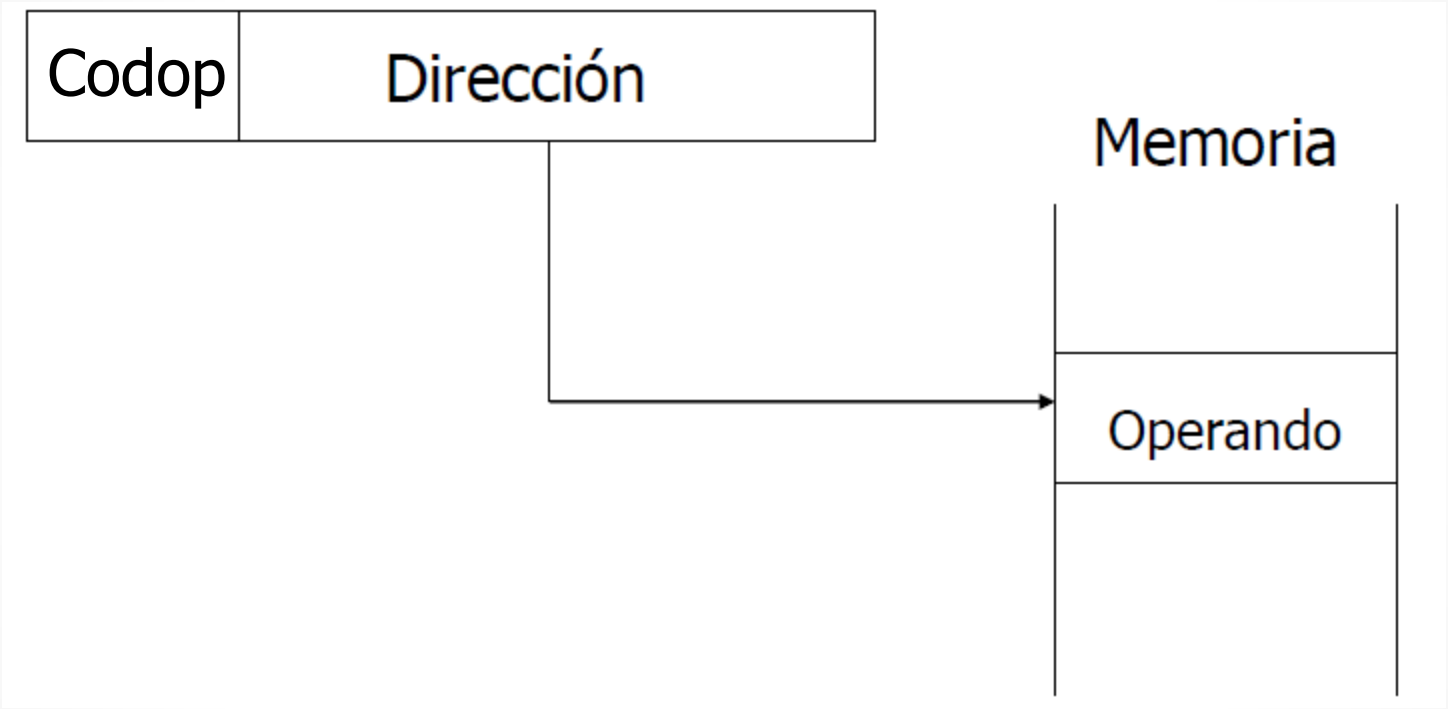
Direccionamiento inmediato (2)

- El operando está presente en la propia instrucción:
  - Operando = A
  - E.j. ADD 5
  - Suma 5 al contenido del acumulador
- Rápido / muy sencillo
- Ventaja: Una vez captada la instrucción, no se requiere una referencia a memoria para obtener el operando; se ahorra un ciclo de memoria en el ciclo de instrucción.
- Desventaja: Rango limitado. Tamaño del número limitado a la longitud del campo de direcciones, que es pequeño comparado con la longitud de la palabra.

Direccionamiento directo (1)

- El campo de direcciones contiene la dirección efectiva del operando
- EA = A
- P.ej., ADD A
  - Busca en memoria la dirección A para el operando
- Sólo requiere una referencia a memoria
- No necesita ningún cálculo especial
- Espacio de direcciones restringido

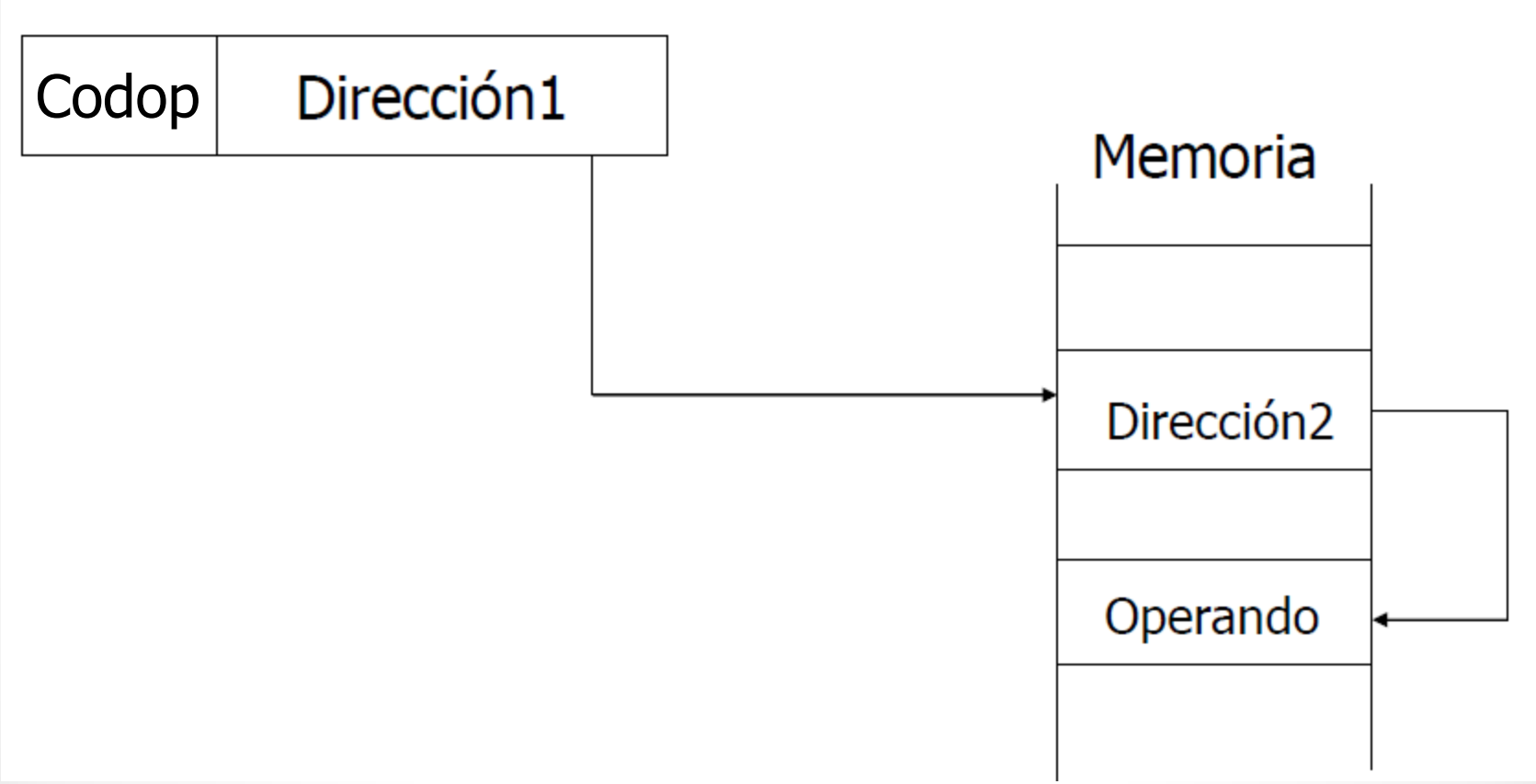
Direccionamiento directo (2)



Direccionamiento indirecto (1)

- Problema del modo anterior: La longitud del campo de direcciones es normalmente la longitud de la palabra
  - limita el rango de direcciones.
- El campo de direcciones referencia la dirección de una palabra de memoria
- La palabra de memoria contiene la dirección completa del operando
- EA = (A)
  - Mira en A, encuentra el contenido de A (A) y busca allí el operando
- P. ej., ADD (A)
  - Agrega el contenido de la celda apuntada por el contenido de A al acumulador

Direccionamiento indirecto (2)



Direccionamiento indirecto (3)

- Espacio de direccionamiento grande.
- Si longitud de la palabra = N bits , se dispone de un espacio de 2^N direcciones.
- Desventajas:
  - La ejecución de la instrucción requiere dos referencias a memoria para captar el operando: una para captar su dirección y otra para obtener su valor.
  - Lento.

Direccionamiento de registro (1)

- El campo de direcciones referencia un registro, en lugar de una dirección de memoria principal.
  - EA = R
- Un campo de direcciones que referencia a registros consta de 3 o 4 bits, → pueden referenciarse un total de 8 o 16 registros de uso general
- Es necesario un campo pequeño de direcciones en la instrucción
  - Instrucciones más cortas
  - Instrucciones de captación más rápidas

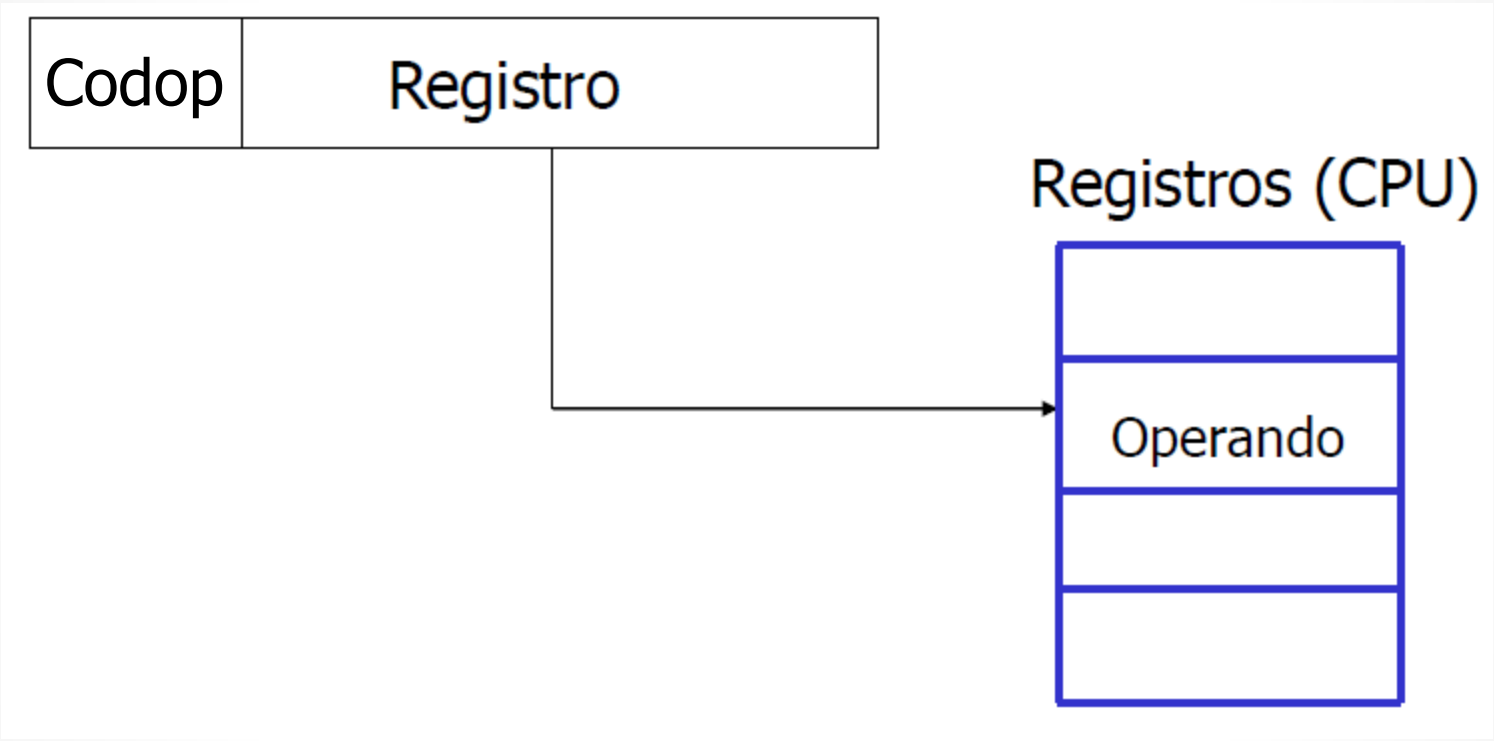
Direccionamiento de registro(2)

- No se requieren referencias a memoria
- Ejecución muy rápida
- Desventaja: Espacio de direcciones muy limitado
- Las CPU modernas emplean múltiples registros de uso general.
- Requiere una buena programación en lenguaje ensamblador (ej. cuando se desarrollan compiladores) para conseguir una ejecución eficiente.

Direccionamiento de registro(3)

- En caso de utilizar masivamente este tipo, los registros de la CPU se utilizan intensivamente.
- Debido a su numero limitado se deben utilizar eficientemente solo si tiene sentido.
- P ej.: almacenamiento de resultados intermedios de un calculo. Implementación de un algoritmo de multiplicación en complemento a dos: una cierta variable se referencia muchas veces, y resulta conveniente implementarla en registro en lugar de memoria principal.

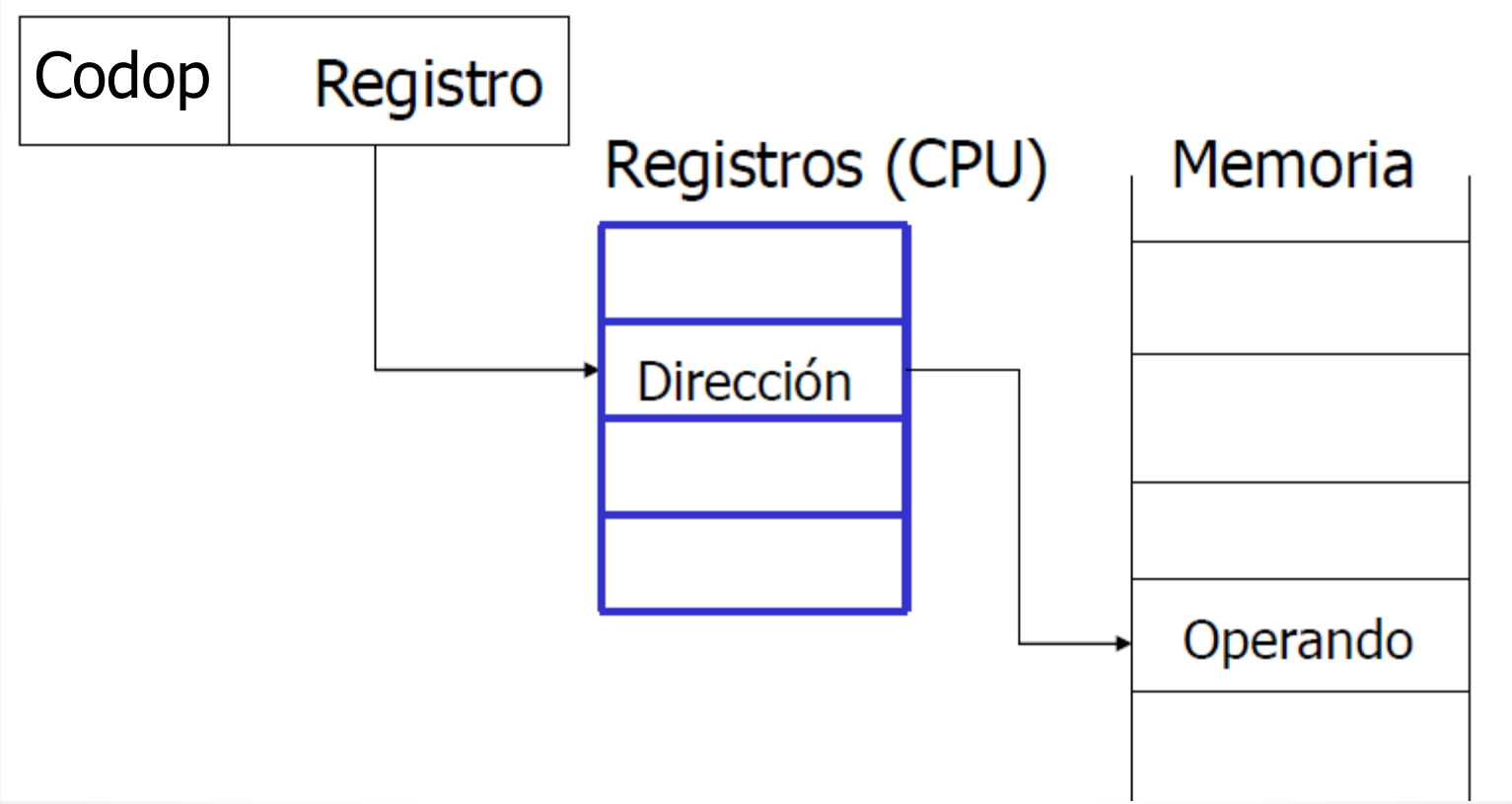
Direccionamiento de registro (4)



Direccionamiento indirecto con registro (1)

- Es análogo al indirecto
  - EA = (R)
- El operando está en una celda de memoria referenciada mediante el contenido del Registro R
- La limitación del espacio se supera haciendo que R referencie a una posición de palabra completa (un registro), que contenga la dirección.
- Emplea una referencia menos a memoria que el direccionamiento indirecto.

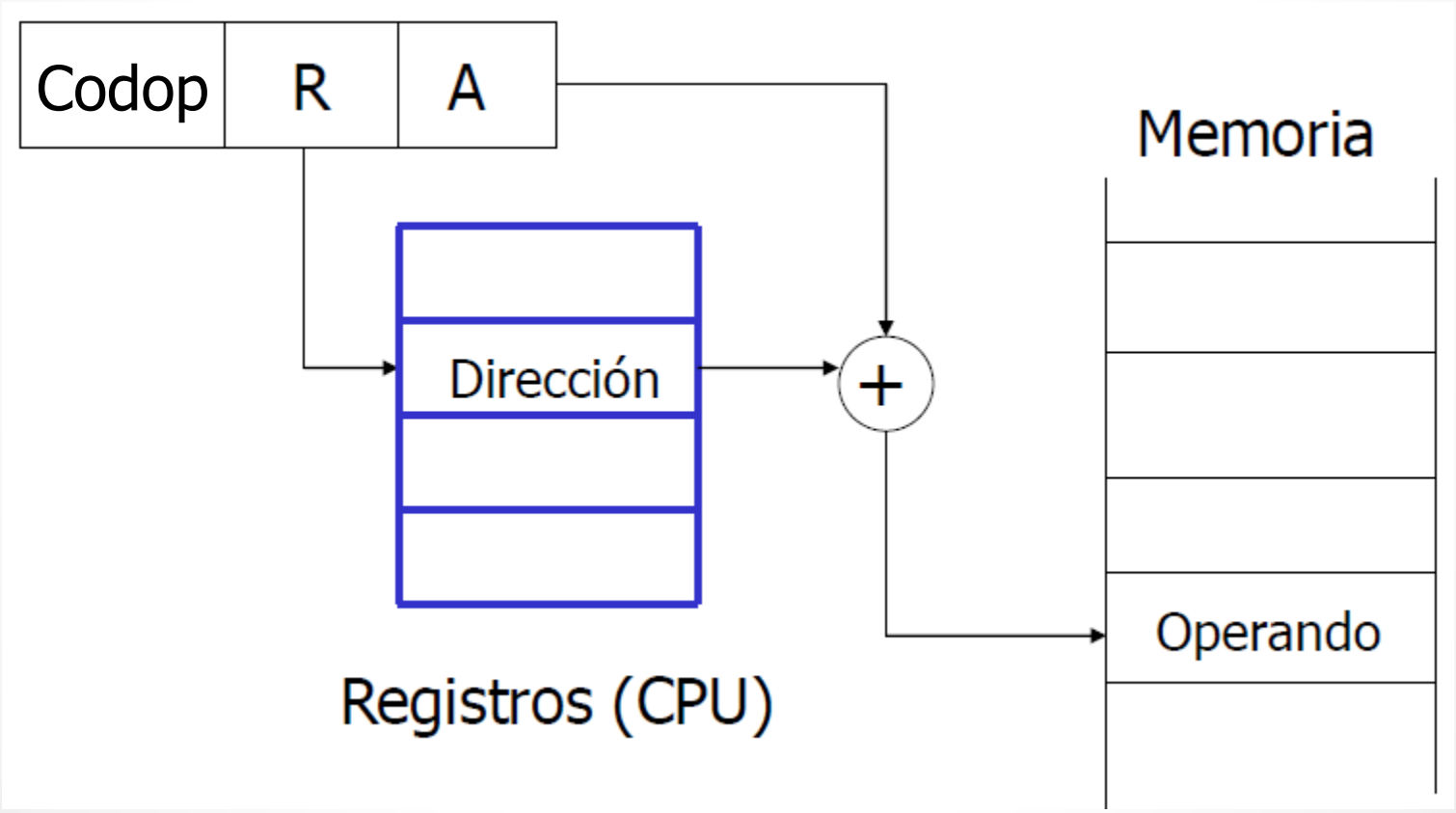
Direccionamiento indirecto con registro (2)



Direccionamiento con desplazamiento (1)

- Modo muy potente: combina direccionamiento directo e indirecto con registro.
  - EA = A + (R)
- Requiere que las instrucciones tengan dos campos de direcciones, al menos uno de ellos es explícito.
  - A = valor base, se utiliza directamente
  - R = registro que contiene el desplazamiento, cuyo contenido se suma a A para obtener la dirección efectiva.
  - o viceversa
- Tres versiones:
  - Desplazamiento relativo
  - Direccionamiento con registro base.
  - Indexado.

Direccionamiento con desplazamiento (2)



Direccionamiento relativo (1)

- Es una versión del direccionamiento con desplazamiento
  - R = Contador de programa, PC
  - La dirección de instrucción actual se suma al campo de direcciones
- EA = A + (PC): La dirección efectiva es un desplazamiento relativo a la dirección de la instrucción.
- P. ej., toma el operando de la posición A a partir de la localización corriente apuntada por el PC
- Si la mayoría de las referencias a memoria están próximas a la instrucción en ejecución, permite ahorrar bits de direcciones en la instrucción.

Direccionamiento con registro-base (2)

- EA = A + (R)
- El registro referenciado contiene una dirección de memoria
- Y el campo de dirección contiene un desplazamiento desde dicha dirección
- R contiene un apuntador a la dirección de memoria base
- R (la referencia a registro) puede ser explícita o implícita

Direccionamiento indexado (3)

- Es inverso a la interpretación del registro-base.
- El registro referenciado contiene un desplazamiento positivo
- Campo dirección es una dirección de memoria principal → contiene más bits que un campo de direcciones de una instrucción comparable que emplee el método anterior
- El método para calcular EA en ambos es igual
  - A = base
  - R = contiene un desplazamiento positivo desde esa dirección
  - EA = A + R
- Las referencias a registro pueden ser explicitas o implícitas.
- Mecanismo eficiente para ejecutar operaciones iterativas.
  - EA = A + R
  - R++

Direccionamiento indexado (4)

- Ejemplo: Dada una lista de números almacenados a partir de la posición A .
  - Se quiere sumar 1 a cada elemento de la lista.
  - Se necesita captar c/elemento, sumar 1 y memorizar el resultado.
  - La secuencia de direcciones efectivas necesarias es:
  - A, A+1, A+2, ….
- El valor A se almacena en el campo de dirección de la instrucción;
- El registro elegido (registro índice) se inicializa a 0.
- Luego de cada operación, el registro se incrementa en 1.
- Esta operación puede hacerse automáticamente, como parte del ciclo de instrucción (autoindexado).

Direccionamiento de pila

- Pila: Matriz lineal de posiciones
- Los elementos se añaden en la cabecera.
- La pila tiene asociado un puntero, cuyo valor es la dirección de la cabecera o tope de la pila.
- El puntero de pila se mantiene en un registro.
- El operando está (implícitamente) en el tope de la pila.
- Son de hecho, direcciones de acceso indirecto con registro.
- Las instrucciones máquina no necesitan incluir una referencia a memoria, operan implícitamente con la cabecera de la pila.
- Bastante comunes en microprocesadores.

Referencias

- Stallings, Williams - Organización y Arquitectura de Computadoras - 5º Ed. - Prentice Hall. Año 2000.

  → *Capítulos 9 y 10*