Introducción a la Computación

Capítulo 10
Repertorio de instrucciones: Características y Funciones

¿Que es un set de instrucciones?

- La colección completa de instrucciones que interpreta una CPU
- Código máquina
- Binario
- Representado generalmente por código ensamblador

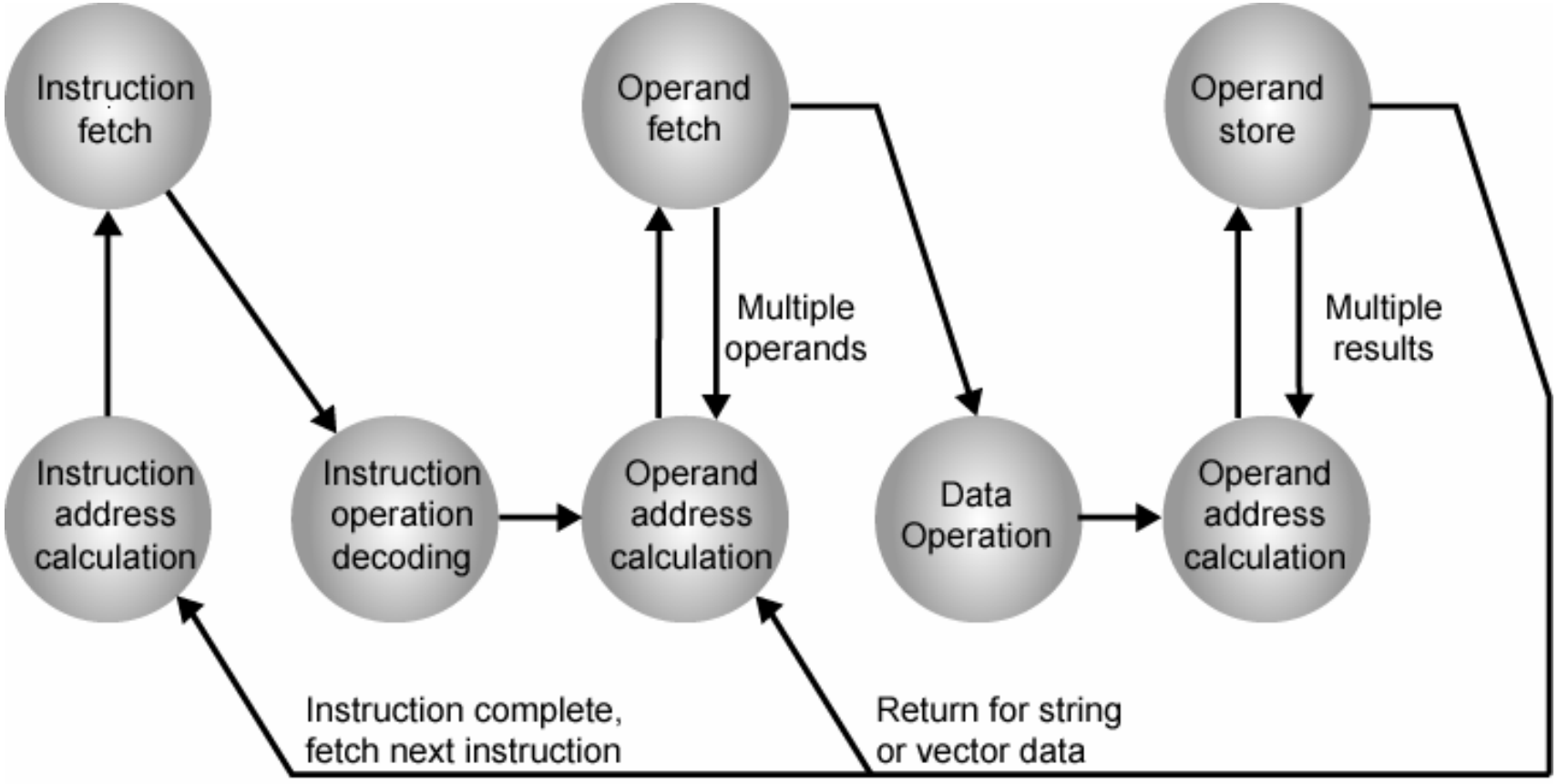
Elementos de una Instrucción

- Código de Operación (Op code)
- Referencia al Operando Fuente
- Referencia al Destino
- Referencia a la próxima Instrucción

¿Dónde se encuentran los operandos?

- Registros de CPU
- Memoria
- Registros de un dispositivo de E/S

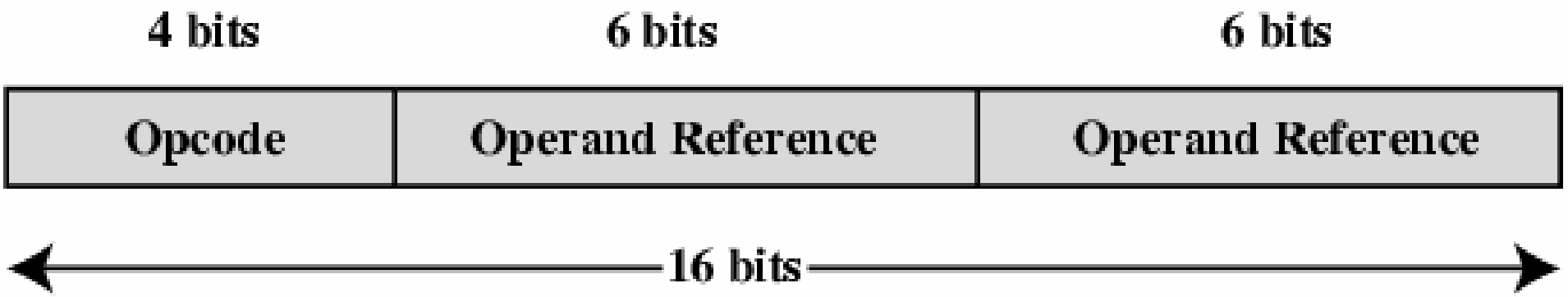
Estados del ciclo de instrucción



Representación de la Instrucción

- En código máquina cada instrucción tiene su patrón de bits que le es único
- Los programadores usan una representación simbólica
  - —ADD, SUB, LOAD
- Los operandos pueden representarse así:
  - —ADD A,B

Formato simple de instrucción



Tipos de Instrucciones

- Procesamiento
- Transferencia
- Almacenamiento
- Control

Cantidad de Referencias (a)

- 3 direcciones
  - Operando 1, Operando 2, Resultado
  - a = b + c;
  - No muy usado
  - Muy larga

Cantidad de Referencias (b)

- 2 direcciones
  - Una de ellas funciona como operando y resultado
  - a = a + b
  - Mas corta
  - Requiere trabajo extra
    - Almacenamiento temporal

Cantidad de Referencias (c)

- 1 dirección
  - Segunda dirección es implícita
  - Se utiliza un registro (acumulador)
  - Muy usado en máquinas simples

Cantidad de Referencias (d)

- 0 direcciones
  - Todas las direcciones están implícitas
  - Se usa la Pila
  - push a
  - push b
  - add
  - pop c
  - c = a + b

Cuantas Direcciones

- Muchas
  - Instrucciones mas complejas (mas potentes?)
  - Mas registros
    - Las operaciones entre registros son mas rápidas
  - Pocas instrucciones por programa
- Pocas
  - Instrucciones mas sencillas (menos potentes?)
  - Mas instrucciones por programa
  - Mas rápida búsqueda y ejecución de las instrucciones

Cuestiones de diseño (1)

- Cantidad de operaciones
  - Cuantas?
  - Que harán?
  - Cuan complejas serán?
- Tipos de Datos
- Formato
  - Longitud del OPCOD
  - Cantidad de Referencias

Cuestiones de diseño (2)

- Registros
  - —Cantidad de registros disponibles
  - —¿Que Operaciones estarán relacionadas con cada registro?
- Modos de Direccionamiento (cantidad de direcciones)
- RISC vs. CISC

Tipos de Operandos

- Direcciones
- Números (Enteros, Flotante)
- Caracteres (ASCII etc.)
- Lógicos (Bits o flags)
- Por otro lado, ¿hay diferencias entre números y caracteres?

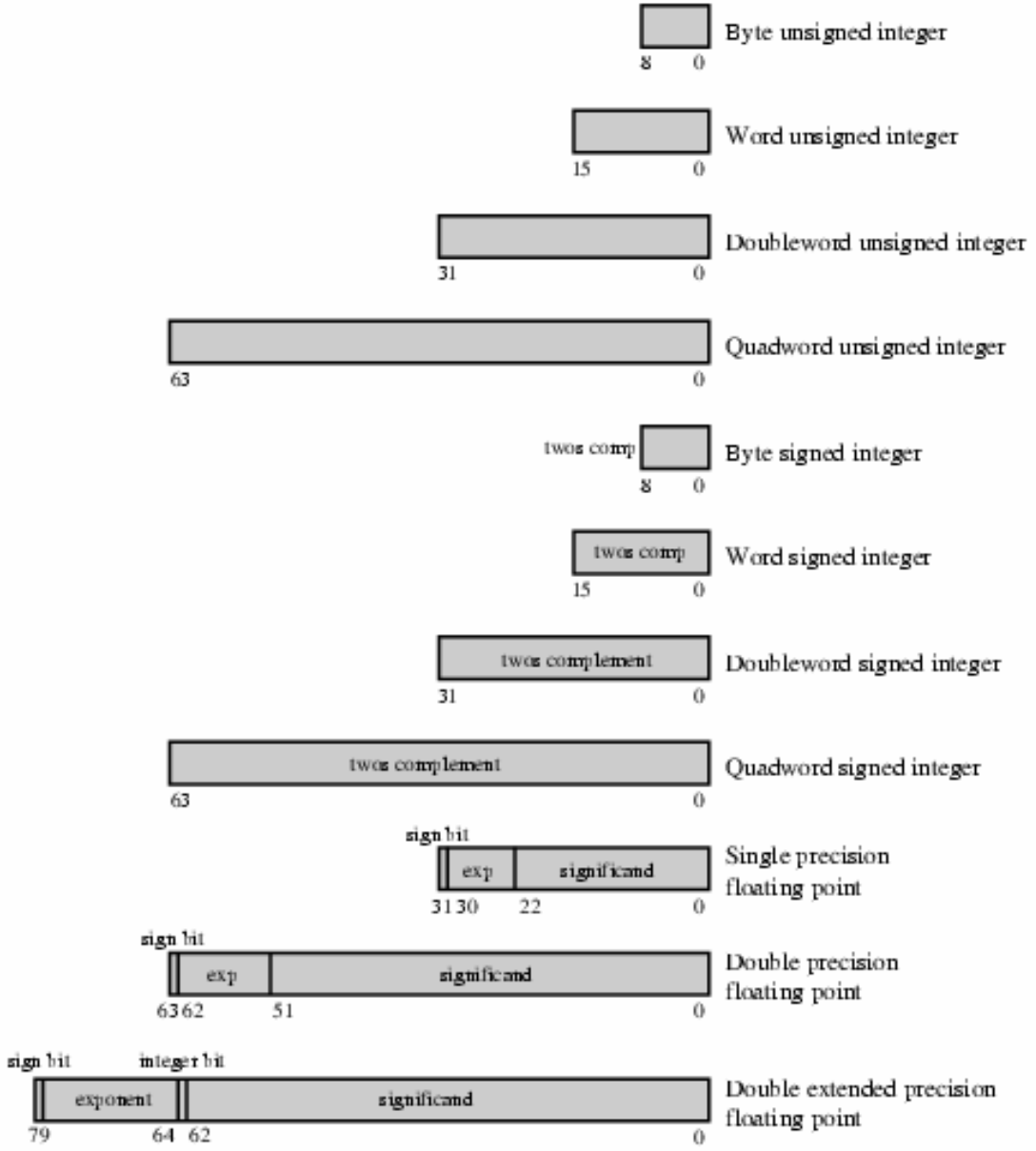
Tipos de datos Pentium

- 8 bit Byte
- 16 bit palabra
- 32 bit palabra doble
- 64 bit palabra cuádruple
- Direccionamiento en unidades de 8 bit
- Un palabra doble de 32 bit es accedida en direcciones divisibles por 4

Tipos específicos de datos

- General – Contenido binario arbitrario
- Entero – Valor binario simple
- Ordinal – Entero sin signo
- BCD – Un dígito por byte
- BCD empaquetado - 2 dígitos BCD por byte
- Puntero Near – Desplazamiento de 32 bit offset dentro del segmento
- Campo bit
- Cadena de bytes
- Punto flotante

Formatos de datos numéricos en Pentium



Tipos de datos PowerPC

- Longitudes de datos de 8 (byte), 16 (media palabra), 32 (palabra) and 64 (doble palabra)
- Algunas instrucciones necesitan los operandos alineados dentro de la palabra
- Es flexible y puede usar big- o little-endian
- El procesador de punto flotante reconoce:
  - Byte sin signo, media palabra sin signo y con signo, palabra sin signo y con signo, doble palabra sin signo, cadena de bytes (<128 bytes)
- Punto flotante
  - Cumple con IEEE 754
  - Precisión simple o doble

Tipos de Operaciones

- Transferencia de Datos
- Aritméticas
- Lógicas
- Conversión
- E/S
- Control del Sistema
- Control de Flujo

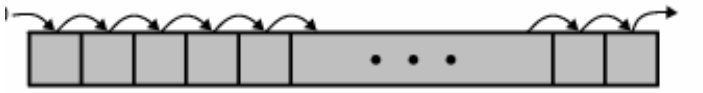
Transferencia de datos

- Especifica
  - Fuente (origen)
  - Destino
  - Cantidad de datos
- Puede haber diferentes instrucciones para distintos movimientos
  - Ej. IBM S/390
- O una instrucción y diferentes direcciones
  - Ej. VAX

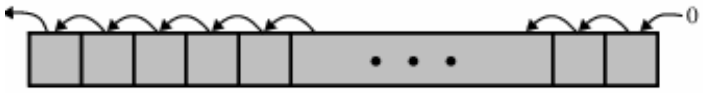
Aritmética

- Suma, Resta, Multiplicación, División
- Enteros con signo
- ¿Punto flotante?
- Puede incluir
  - Incremento (a++)
  - Decremento (a--)
  - Negación (-a)

Operaciones de rotación y desplazamiento

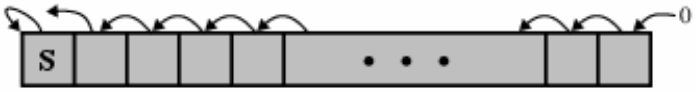


(a) Logical right shift

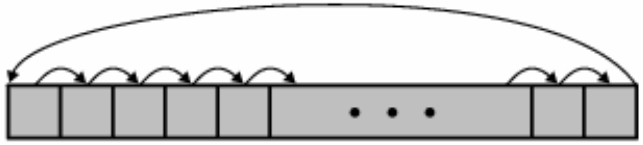


(b) Logical left shift

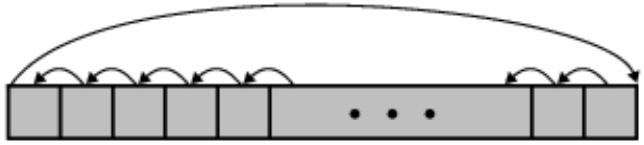
(c) Arithmetic right shift



(d) Arithmetic left shift



(e) Right rotate



(f) Left rotate

Lógicas

- Operaciones con Bits
- AND, OR, NOT

Conversión

- Ej. Binario a Decimal

Entrada/salida

- Pueden ser realizadas por instrucciones específicas
- O utilizar las de trasferencia (E/S mapeada en memoria)
- Puede ser realizada por un controlador separado (DMA)

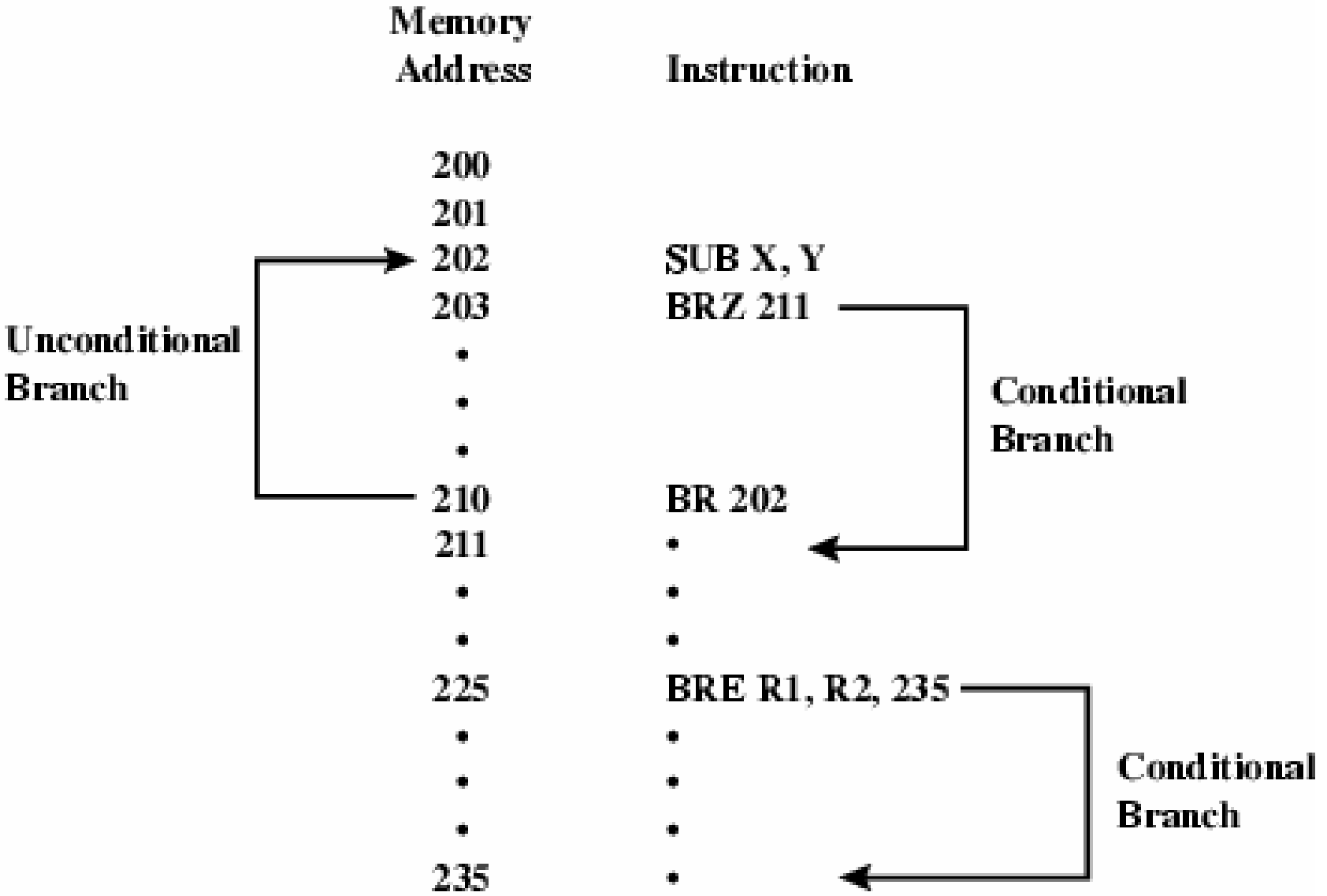
Control del sistema

- Instrucciones privilegiadas
- La CPU necesita estar en un estado específico
  - Anillo 0 o 80386+
  - Modo Kernel
- Para uso del Sistema Operativo exclusivamente

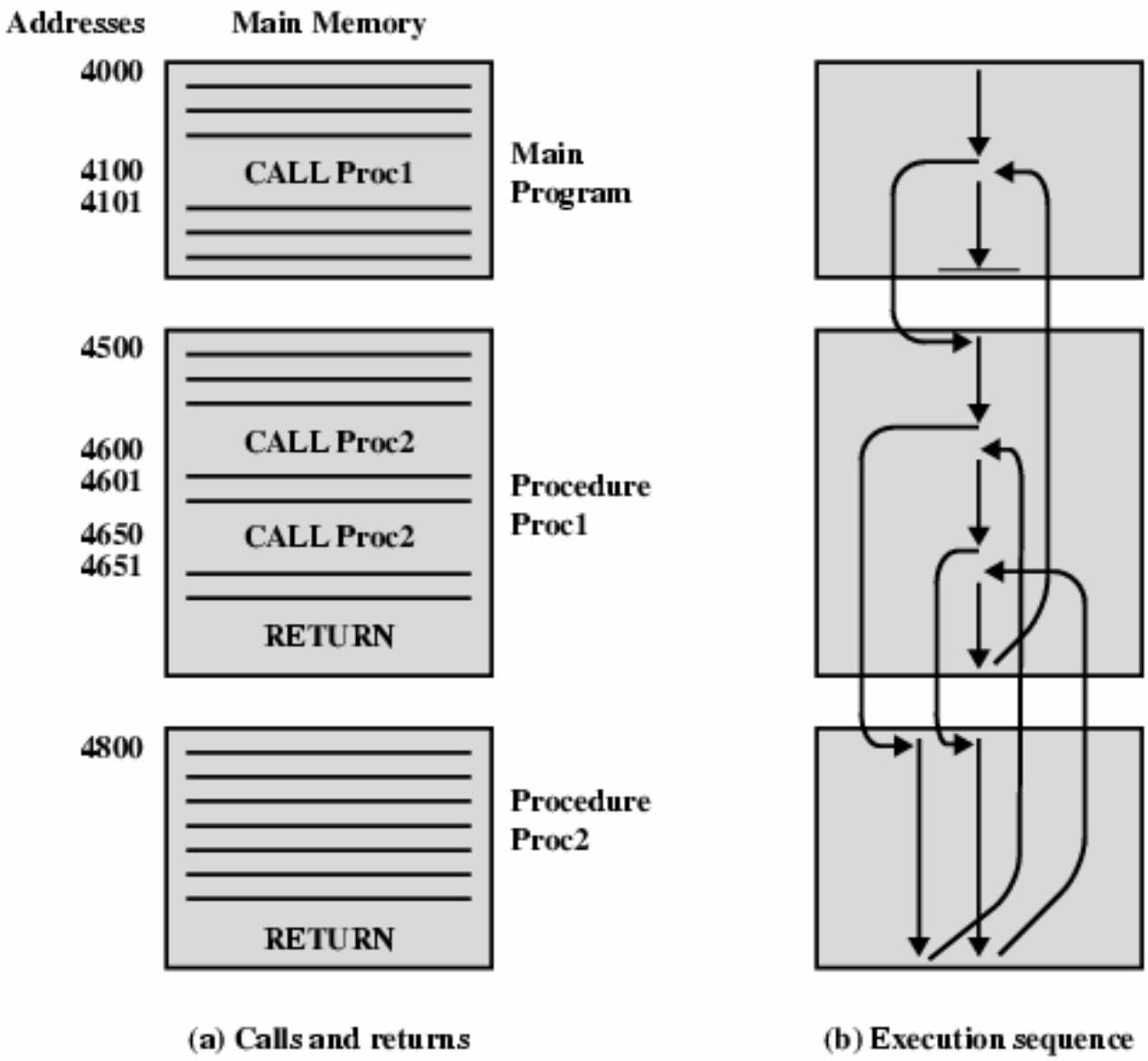
Control de flujo

- Bifurcación
  - Ej. Ir a x si el resultado es cero
- Salto
  - Ej. incrementar y saltar si cero
  - ISZ Registro1
  - Branch xxxx
  - ADD A
- Llamada a subrutina
  - Interrupciones por software

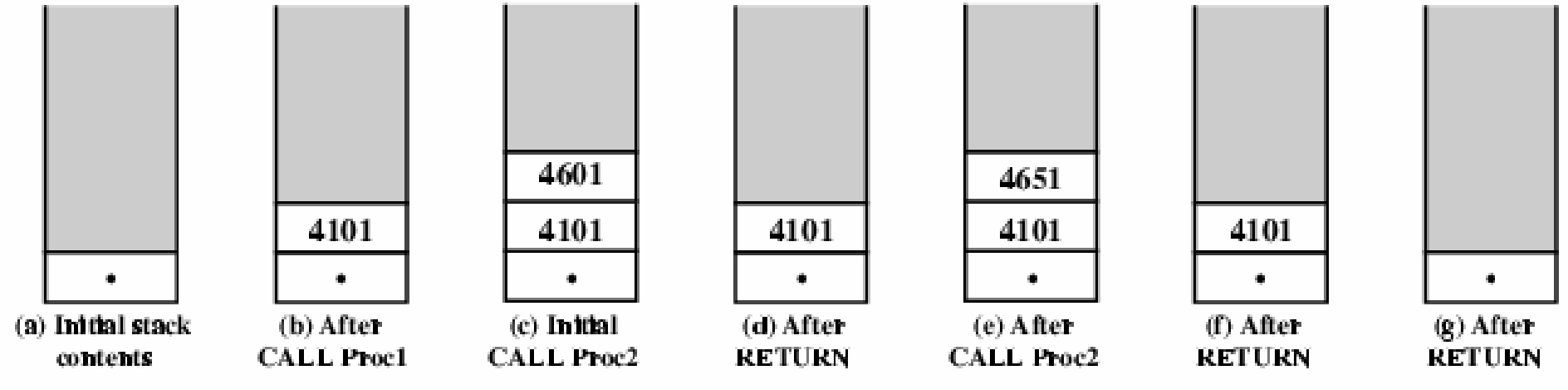
Instrucción de Bifurcación



Subrutinas anidadas



Uso de la Pila



Ordenamiento de Bytes

- En que orden se leen números que ocupan mas de 1 byte
- 12345678 se almacena en 4 celdas de 8bit

Ejemplo del ordenamiento de Bytes

Direcc.	Valor (1)	Valor(2)
184	12	78
185	34	56
186	56	34
186	78	12

- ¿Como se lee de arriba abajo o de abajo hacia arriba?

Nombres del ordenamiento de Bytes

- Problema llamado Endian
- El sistema que tiene el byte menos significativo en la dirección mas alta se denomina big-endian
- El sistema que tiene el byte menos significativo en la dirección mas baja se denomina little-endian

Ejemplo de estructuras de datos en C

```
struct{
   int      a;     //0x1112_1314                    word
   int      pad;   //
   double   b;     //0x2122_2324_2526_2728          doubleword
   char*    c;     //0x3132_3334                    word
   char     d[7];  //'A','B','C','D','E','F','G'   byte array
   short    e;     //0x5152                         halfword
   int      f;     //0x6161_6364                    word
} s;
```

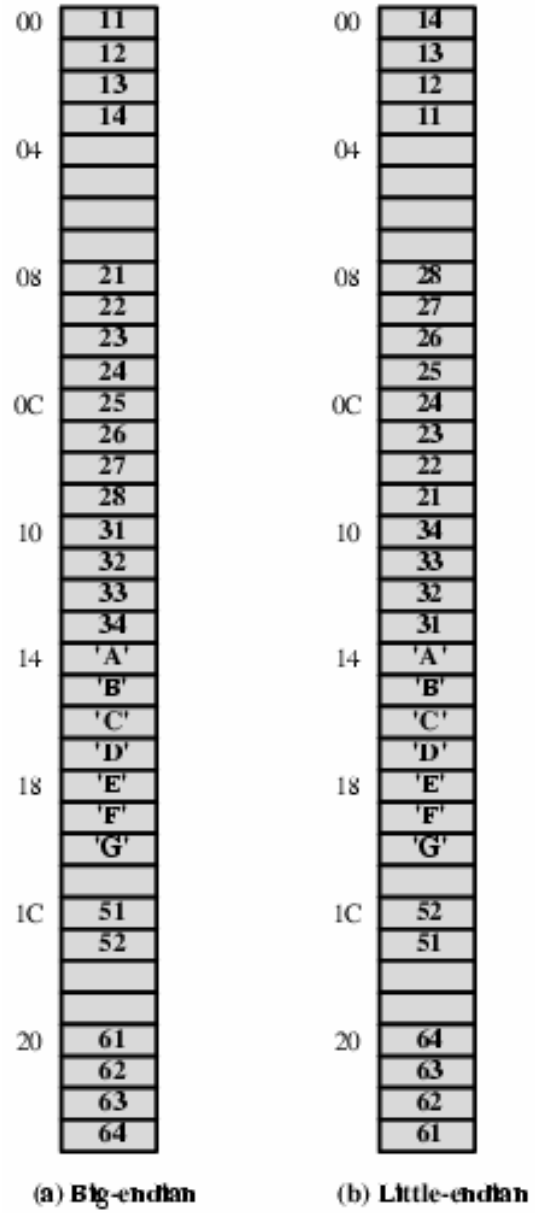
Big-endian address mapping

Byte Address								
00	**11** 00	**12** 01	**13** 02	**14** 03	04	05	06	07
08	**21** 08	**22** 09	**23** 0A	**24** 0B	**25** 0C	**26** 0D	**27** 0E	**28** 0F
10	**31** 10	**32** 11	**33** 12	**34** 13	**'A'** 14	**'B'** 15	**'C'** 16	**'D'** 17
18	**'E'** 18	**'F'** 19	**'G'** 1A	1B	**51** 1C	**52** 1D	1E	1F
20	**61** 20	**62** 21	**63** 22	**64** 23				

Little-endian address mapping

								Byte Address
				11 03	**12** 02	**13** 01	**14** 00	00
21 0F	**22** 0E	**23** 0D	**24** 0C	**25** 0B	**26** 0A	**27** 09	**28** 08	08
'D' 17	**'C'** 16	**'B'** 15	**'A'** 14	**31** 13	**32** 12	**33** 11	**34** 10	10
		51 1D	**52** 1C		**'G'** 1A	**'F'** 19	**'E'** 18	18
				61 23	**62** 22	**63** 21	**64** 20	20

Vista alternativa del mapa de memoria



¿Que es estándar?

- Pentium (80x86), VAX son little-endian
- IBM 370, Motorola 680x0 (Mac), casi todos los RISC son big-endian
- Internet es big-endian
  - Hace que escribir programas en una PC para Internet sea mas complicado
  - WinSock provee funciones de conversión **htoi** y **itoh** (Host to Internet & Internet to Host)