Nivel ISA – Parte 3

Arquitectura y Organización de Computadoras II

Ticiano J. Torres Peralta

REV2021

Las 7 Dimensiones del ISA

Que tiene que considerar un diseñador de ISA?

- La clase del ISA
- El modelo de memoria
- Los modos de direccionamiento
- **Tipos y tamaños de datos (y de operandos)**
- **Tipos de operaciones**
- **Tipos de control de flujo**
- Formato de instrucción
- *Registros Disponibles* (Arquitecturas GPR)

Tipos de Datos

El propósito de una computadora es procesar datos en una variedad de areas: financiera, científica, multimedia, etc. Estos datos tienen que ser representados de una forma precisa, y se lo hace al nivel ISA.

Vale una aclaración muy importante: hay formas infinitas de tratar todos tipos de datos a nivel software, incluyendo tipos de datos no soportados por el hardware. Datos tratados de esta forma ofrece mucha flexibilidad a costo de rendimiento.

Aquí vamos a tratar tipos de datos explicitados en el ISA, los cuales tendrán una representación y soporte del hardware.

Tipos de Datos

Hay dos categorías superiores de tipos de datos:

- Datos Numéricos
  - Enteros (integers)
    - 8, 16 (short), 32 (int), y 64 (long) bits.
    - Signados (Complemento-a-dos) o No-signados
  - Fraccionados (floating-point)
    - 32 (float), 64 (double), 128 (quadruple) bits.
    - Signados (IEEE 754)
  - Representación Decimal (BCD)
    - Cada digito decimal es codificado en 4bits (Packed) o 8 bits.
    - Cuando se necesita que el valor fraccionado y la aritmética entre ellos sea exacta.

Tipos de Datos

Hay dos categorías superiores de tipos de datos:

- Datos No-numéricos
  - Caracteres
    - 7 bits para la decodificación ASCII (char)
    - 16 bits para codificación Unicode.(wide-char, multi-byte-char)
  - Cadenas de caracteres (string)
    - Contienen un símbolo especial para indicar el final de la cadena.
    - La instrucción puede contener un campo (operando para indicar la longitud de la cadena.
  - Booleano
    - 8 , 32 o 64 bits.
    - En algunos sistemas FALSE es representado por el 0, mientras que TRUE por cualquier otro valor.
    - Hay una representación especial del booleano cuando se encuentra en un arreglo, conocido como el bit map.
  - Puntero
    - Usado para representar una dirección de memoria. 32 o 64 bits dependiendo del ancho del bus de direccionamiento.

Tipos de Operaciones

Una de las cosas mas importante en la definición de un ISA es cuales operaciones van a estar disponibles. Los tipos de operaciones mas comunes en las ISAs son las siguientes.

Operator type	Examples
Arithmetic and logical	Integer arithmetic and logical operations: add, subtract, and, or, multiply, divide
Data transfer	Loads-stores (move instructions on computers with memory addressing)
Control	Branch, jump, procedure call and return, traps
System	Operating system call, virtual memory management instructions
Floating point	Floating-point operations: add, multiply, divide, compare
Decimal	Decimal add, decimal multiply, decimal-to-character conversions
String	String move, string compare, string search
Graphics	Pixel and vertex operations, compression/decompression operations

Tipos de Operaciones

La tabla anterior es de ninguna forma una lista exhaustiva de los tipos de operaciones que pueden estar disponibles en un ISA. Mientras vaya avanzando las necesidades del mercado del software, es posible que en algún momento sea necesario incluir nuevos tipos de operaciones, por ejemplo, operaciones para de inteligencia artificial.

También es importante aclarar que, por razones de compatibilidad en reversa, uno incluye por defecto todo tipo de instrucciones presentes en diseños.

Tipos de Operaciones

Instrucciones de movimiento (o transferencia) de datos:

Cuando uno se refiere a estas, la palabra movimiento tienen un significado diferente a lo que estamos acostumbrados coloquialmente.

Aquí, cuando hablamos de movimiento de datos, en verdad estamos hablando de una duplicación del dato, creando un nuevo objeto que sera colocado en algún lugar. Por ejemplo, cuando decimos que movemos un dato de la dirección 0x10FFf30A en memoria principal a un registro, estamos haciendo una copia idéntica del dato en memoria y la información original en memoria permanece.

Las instrucciones de movimiento tienen que indicar la cantidad de datos a ser movidos. Esto puede ser explicitado en la instrucción, o puede ser un valor implícito y siempre el mismo.

Tipos de Operaciones

Hay esencialmente cuatro escenarios para mover datos:

- Memoria a Registro (LOAD)
- Registro a Memoria (STORE)
- Registro a Registro (MOVE)
- Inmediato (un caso especial de registro a registro)

Tipos de Operaciones

Operaciones Diádicas:

Estas involucran una combinación entre dos operando para producir un resultado.

Entre ellas están:

- Suma, resta, multiplicación, división.
- AND, OR son las mas comunes pero hay 16 teóricas en total.
- LOAD, STORE (instrucciones de movimientos).

Aparte de el concepto lógico que llevan el AND y el OR, estas dos operaciones tienen unos usos de suma importancia:

- Masking (AND): se usa cuando querés extraer una porción de bits de una palabra.
- Packing (OR): se usa cuando querés empaquetar una porción de bits a una palabra.

Tipos de Operaciones

Operaciones Monádicas:

Estas involucran un operando para producir un resultado.

Entre ellas están:

- Desplazamientos lógicos y aritméticos (shift), desplazamientos circulares (rotation).
- JUMP, BR, GOTO.
- INC, DEC, NEG, CLR.

Aveces hay una especifica operación diádica que tiene tanto uso que se crea una equivalente monádica. Por ejemplo, INC que incrementa el valor del operando por 1 es la equivalente que hacer un ADD entre un operando y otro de valor 1.

Tipos de Operaciones

Instrucciones de entrada/salida:

Hay tres esquemas de instrucciones para tratar entradas y salidas desde dispositivos del sistema.

- E/S programada con espera activa: el procesador contiene instrucciones que inician una comunicación con un dispositivo y quedan en un bucle esperando la respuesta. Usada en sistemas que tienen que reaccionar inmediatamente a un cambio externo (real-time emebeded systems).
- E/S controlada por interrupciones: las instrucciones inician el dispositivo de E/S y le dicen que genere una interrupción cuando esté listo. Aunque es considerado un mejor esquema que el anterior, si tiene sus problemas, dispositivos que generan muchas interrupciones puede ser costoso para el procesador.
- E/S DMA (direct memory access – acceso directo a memoria): este esquema es parecido al de E/S programada con espera activa, pero pateamos la responsabilidad a otro lado. En vez que el procesador se haga cargo del dispositivo, hay un controlador llamado DMA que se hace cargo. El controlador tienen 4 registros.
  - Un registro contiene la dirección de memoria a ser leída o escrita.
  - Un registro contiene el valor de la cantidad de bytes (o words) que tienen que ser transferidos.
  - Un registro especifica el numero o la dirección del dispositivo.
  - Un registro especifica si los datos serán leídos del o escritos al dispositivo.

Tipos de Operaciones

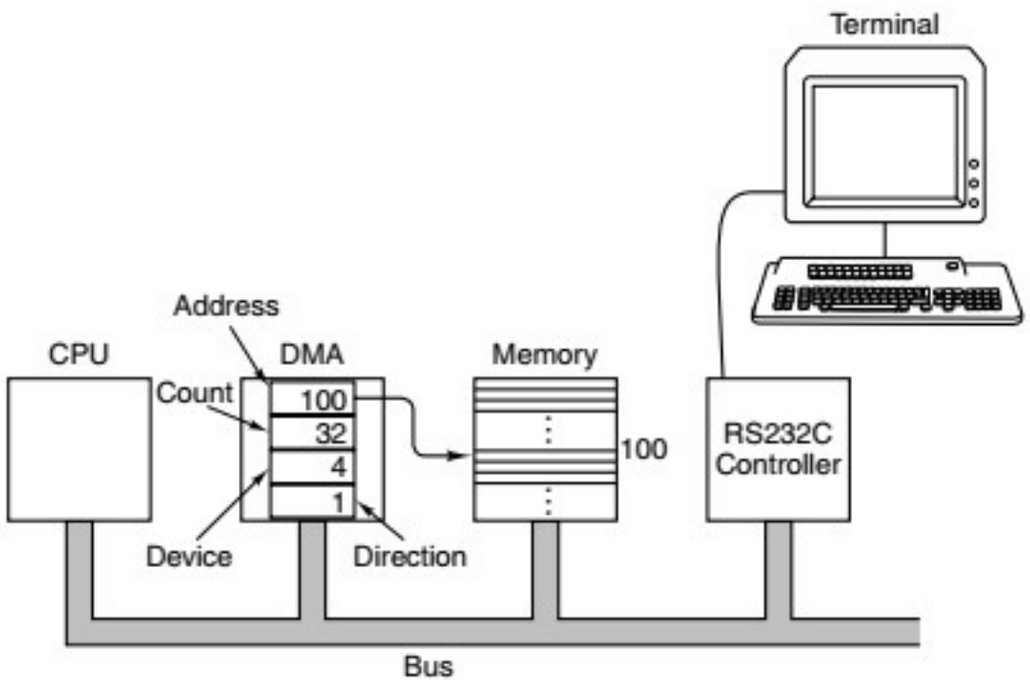


Figure 5-32. A system with a DMA controller.

En este ejemplo, el procesador le indica al controlador DMA que tienen que leer dirección 100 de memoria y transferir 32 Bytes al dispositivo 4, la terminal.
El controlador toma control del bus, lee el primer byte de memoria y lo transfiere a la terminal. Al terminar esa operación, el controlador incrementa la dirección por uno (a 101) y disminuye la cantidad de bytes por uno (a 31) y repite el proceso a la velocidad del dispositivo.
Recién al terminar la transferencia completa, el controlador DMA afirma una interrupción al CPU, reduciendo tremendamente la cantidad de interrupciones en comparación si el CPU tuviera que procesar esa operación (32 interrupciones vs 1).
DMA no viene sin sus consecuencias. Es posible que DMA requiera muchos ciclos del bus para poder transferir todo los datos. Esto puede impedir al CPU a usar el bus (DMA tienen mas prioridad que el CPU) obligándolo igual a esperar. Cuando pasa esto, se llama cycle stealing (robo de ciclo).

Control de Flujo

La mayoría de la instrucciones no alteran el flujo del programa, después que una instrucción es ejecutada, la próxima en memoria es buscada, ejecutada y el contador de programa es aumentado por el tamaño de la instrucción. En otras palabras, la secuencia de ejecución es en el orden en que aparecen en memoria.

Si un programa contiene ramificaciones, el flujo del programa puede ser modificado de forma dinámica (run time – tiempo de ejecución) y la secuencia de ejecución y el orden en que aparecen en memoria ya no es el mismo.

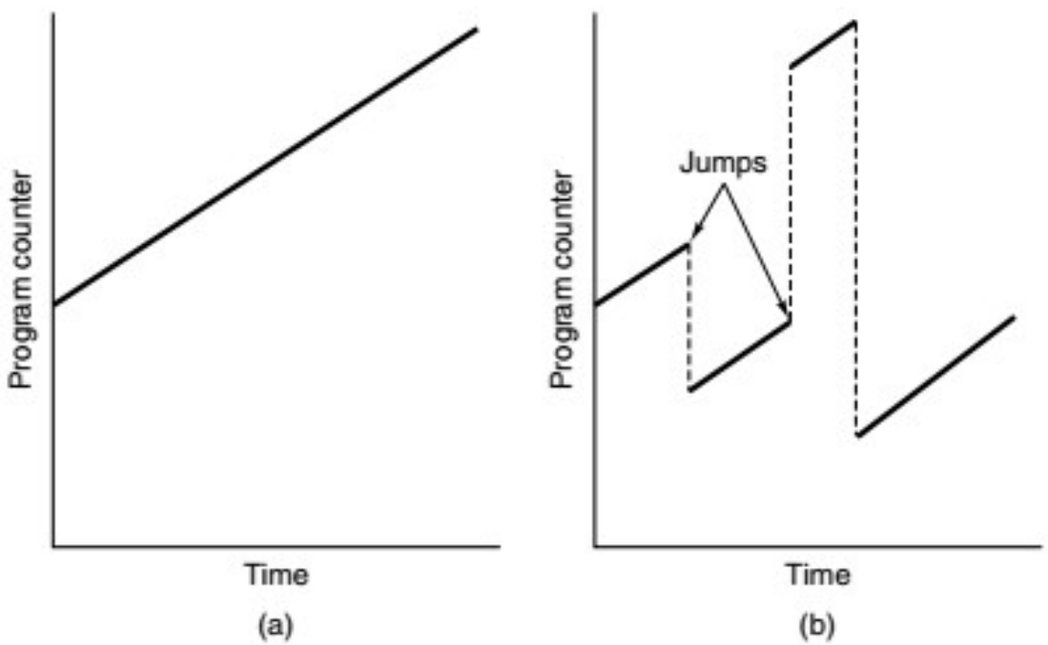


Figure 5-36. Program counter as a function of time (smoothed). (a) Without branches. (b) With branches.

Control de Flujo

Hay varios conceptos que pueden modificar el control del flujo del programa:

- Ramificaciones condicionales (branches)
- Ramificaciones incondicionales (jumps)
- Llamados a procedimientos (calls)
- Retorno de procedimientos (returns)
- Trampas (traps)
- Interrupciones (interrupts)

Control de Flujo

Opciones de ramificación:

Name	Examples	How condition is tested	Advantages	Disadvantages
Condition code (CC)	80x86, ARM, PowerPC, SPARC, SuperH	Tests special bits set by ALU operations, possibly under program control.	Sometimes condition is set for free.	CC is extra state. Condition codes constrain the ordering of instructions since they pass information from one instruction to a branch.
Condition register	Alpha, MIPS	Tests arbitrary register with the result of a comparison.	Simple.	Uses up a register.
Compare and branch	PA-RISC, VAX	Compare is part of the branch. Often compare is limited to subset.	One instruction rather than two for a branch.	May be too much work per instruction for pipelined execution.

Una propiedad notable de los tests es que suelen ser simples comparaciones con el valor cero.

Control de Flujo

Aveces hay disponible un registro especial que contienen una serie de banderas o códigos de condición. El mismo se lo conoce con varios nombres: Flag Register, Program Status Word, Status Register, y Condition Code Register.

Este registro contienen una serie de bits que indican una condición, la cual una instrucción puede testar. Estos bits, por lo general, son generados automaticamente por el procesador si se cumple la condición en cuestión.

N — Set when the result was Negative.

Z — Set when the result was Zero.

V — Set when the result caused an oVerflow.

C — Set when the result caused a Carry out of the leftmost bit.

A — Set when there was a carry out of bit 3 (Auxiliary carry—see below).

P — Set when the result had even Parity.

Control de Flujo

En el caso de la arquitectura MIPS, la condición es resuelta en la instrucción de ramificación

MIPS Branch Instructions

Branch instructions: conditional transfer of control

- **Compare** on:
  - **equality or inequality of two registers**

    Opcode rs, rt, target

    rs, rt: the registers to be compared

    target: the branch target
  - **>, <, ≥, ≤ of a register & 0**

    Opcode rs, target

    rs: the register to be compared with an implicit 0

    target: the branch target
- **Branch** to a target that is a signed displacement (expressed in number of *instructions*) from the instruction *following* the branch

Some examples:

```
beq $t0, $t1, Target    # branch to Target if $t0 == $t1
bgez $t0, Target        # branch to Target if $t0 ≥ 0
```

Control de Flujo

Modos de direccionamiento:

Por lo general, la dirección del destino de la instrucción siempre tienen que ser explicitada. Una excepción grande es la instrucción de retorno.

La forma mas simple de especificar el destino hacer un desplazamiento relativo al PC, osea, la instrucción actual. Este modo de direccionamiento se llama relativo al PC (PC relative).

Tiene varias ventajas:

- El destino por lo general es cercano al lugar donde se produce la ramificación.
- Requiere menos bits para representar el destino del salto.
- Permite al código ejecutarse independientemente de donde se lo carga a memoria. Esto se llama independencia posicional (positional independance).

Control de Flujo

Modos de direccionamiento:

Para implementar retornos y saltos indirectos donde el destino no es conocido en tiempo de compilación, otra estrategia es necesaria.

En estos casos, se tiene que designar la dirección de forma dinámica ya que puede cambiar entre diferentes tiempos de ejecución. Aquí registros o pilas pueden ser usadas para guardar esos valores y se pueden utilizar cualquiera de los modos de direccionamiento estudiado.

Control de Flujo

Opciones de invocación de procedimientos:

Las llamadas y retornos de procedimientos involucran una transferencia de control y posiblemente la necesidad de guardar estados. Como mínimo, la dirección de retorno tienen que ser guardada.

Hay dos convenciones para guardar valores de registros:

- Dentro del procedimiento que hace el llamado (caller saving).
- Dentro del procedimiento llamado (callee saving).

Control de Flujo

Vale aclarar una diferencia entre una llamada a un procedimiento en general y una llamada a una corutina (un caso especial de procedimiento). La diferencia es en el método en que se transfiere el control entre el llamador y el llamado.

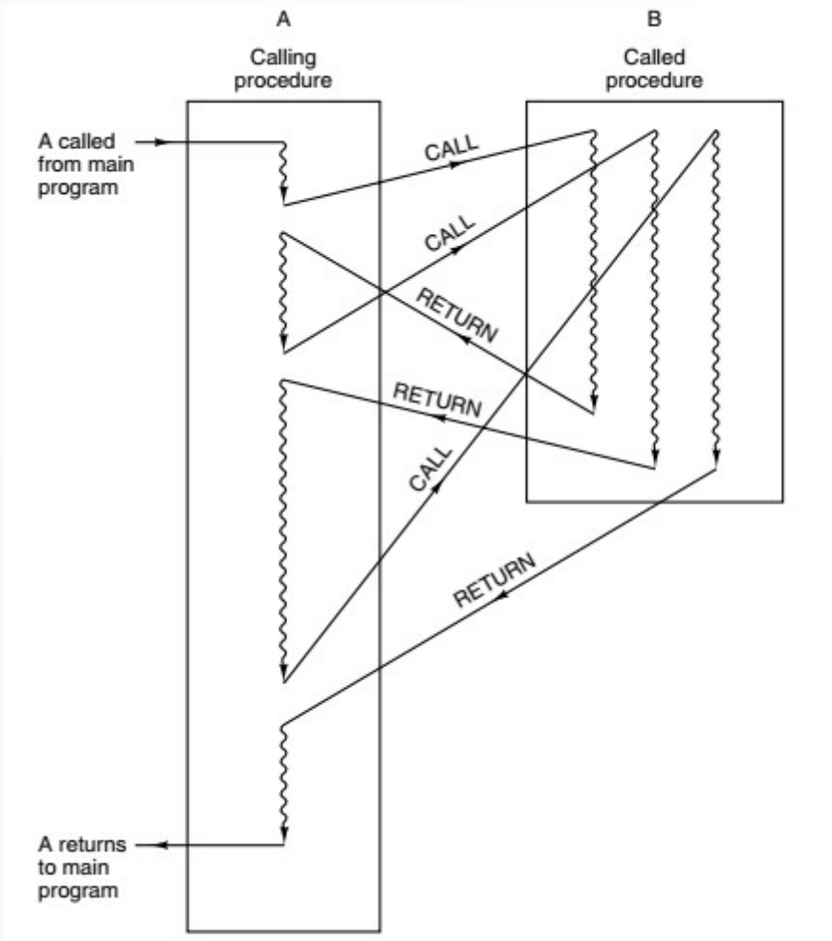
Con una instrucción que llama a un procedimiento, la dirección de retorno es guardada y el llamado usa la instrucción de retorno para transferir de vuelta el control.

En una corutina, cada procedimiento llama al otro y continua una instrucción adelante desde la ultima llamada. Usa instrucciones diferentes a las típicas CALL y RETURN.

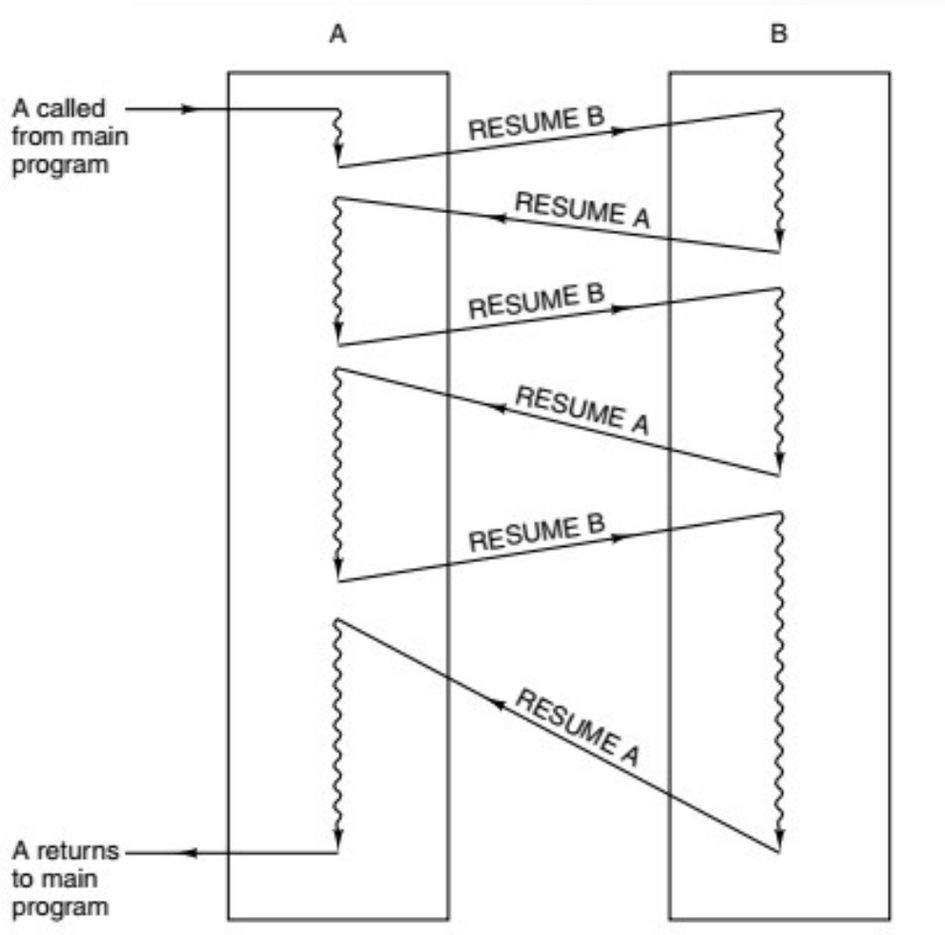
La corutina son la manera de simular procesamiento en paralelo en un solo procesador (o core).

Control de Flujo

Llamado a procedimiento



Llamado a corutina



Control de Flujo

Trampas:

Una trampa es un procedimiento iniciado de forma automática según una condición causada por un programa.

Un buen ejemplo es overflow. Cuando el resultado de una operación aritmética excede el valor mas grande que puede ser representado, activa una trampa. Esto significa que el programa transfiere el control a un procedimiento conocido, llamado manejador de trampas (trap handler). Este toma alguna acción apropiada, como imprimir una advertencia o error en pantalla. Dependiendo de la acción, después puede devolver el control al programa que causó la trampa.

La condición que activa las trampas son excepciones, son causadas por un programa y detectadas por el hardware.

Es mucho mas eficiente definir trampas que se activen a nivel hardware a que un programador tenga que hacer los tests a nivel software.

Control de Flujo

Trampas comunes:

- Overflow y underflow en punto-flotante
- Overflow en enteros
- Violación de protección
- Código operación no definido
- Overflow en la pila
- Dispositivo de E/S no existente
- División por cero
- Traer una palabra de una dirección impar

Control de Flujo

Interrupciones:

A diferencia de los otro controles de flujo, una interrupción no es causada por el programa en ejecución, sino por otra cosa (normalmente un dispositivo E/S).

Como una trampa, detienen la ejecución de un programa y transfiere el control al manejador de interrupciones (interrupt handler) que toma alguna acción apropiada. Este, cuando termina, siempre retorna el control al programa interrumpido.

Se puede hacer una comparación interesante entre interrupciones y trampas. Se considera que las trampas funcionan de forma sincrona con tu programa. En términos de reproducibilidad la trampa siempre ocurrirá en el mismo lugar del programa. En cambio, la interrupciones son consideradas asíncronas.

Control de Flujo

Interrupciones:

Como en cualquier transferencia de control entre procedimientos, los estados de los registros tienen que ser guardados.

Si una interrupción restaura completamente los estados de los registros a los de antes de la interrupción, esta interrupción es considerada transparente.

Control de Flujo

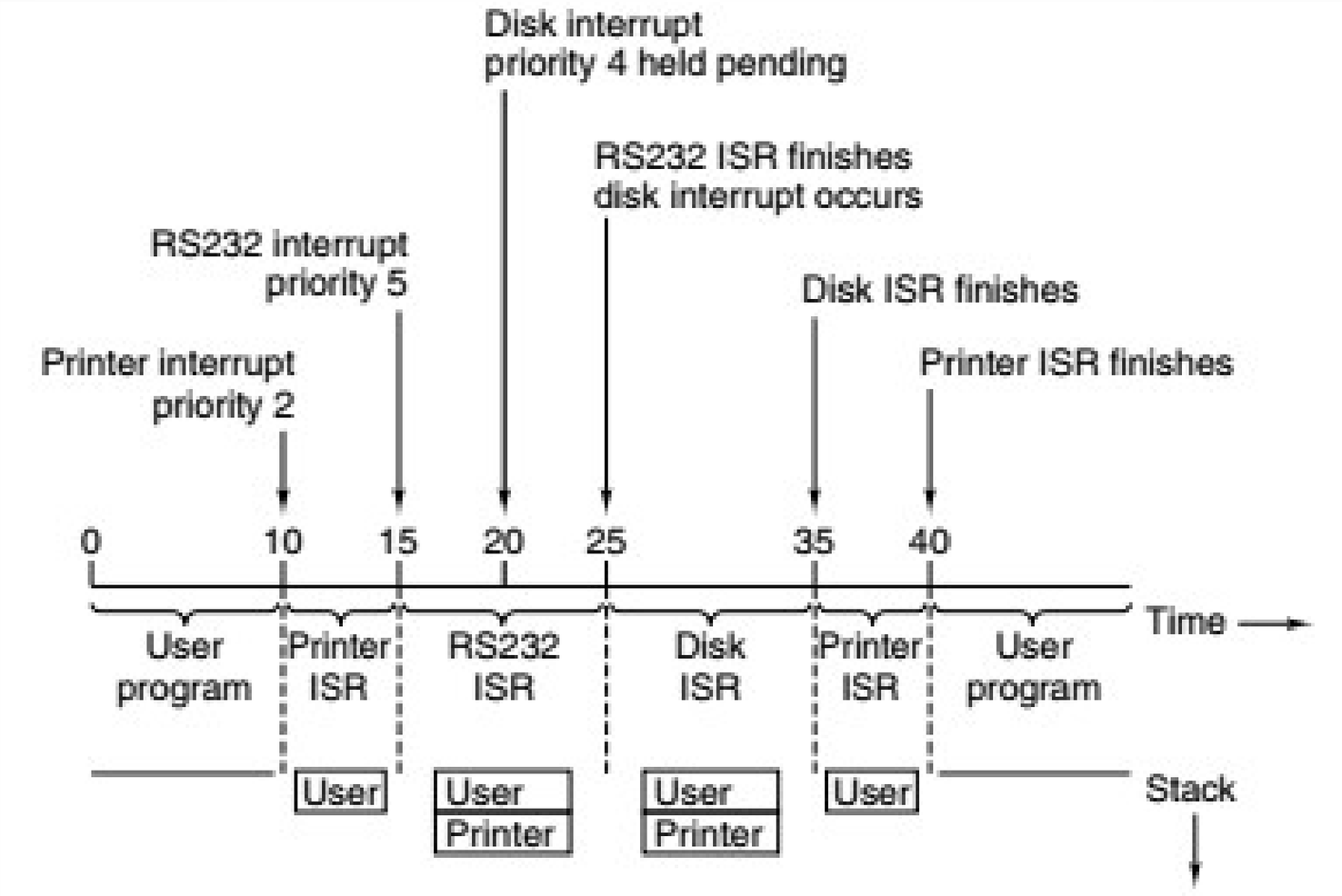
Interrupciones:

Al menos que tu computadora tenga un solo dispositivo, es muy posible que uno trate de interrumpir en el medio de otra interrupción.

Hay dos formas de tratar este problema:

- Que la primera instrucción que ejecute la interrupción deshabilite las interrupciones. Es una estrategia simple que serializa las interrupciones. Puede traer problemas a dispositivos que no pueden tolerar mucha espera.
- En un sistema con dispositivos de E/S de tiempo critico, un mejor diseño es asignarle una prioridad a cada dispositivo. Al mismo tiempo, al CPU también se debería asignar una prioridad. Aquí, si hay una interrupción activa y otro dispositivo trata de activar su interrupción, pasan dos escenarios: si el dispositivo es me menor prioridad, se ignora, mientras si el dispositivo es de mayor prioridad, se trata la interrupción de forma inmediata. Con este esquema donde una interrupción puede interrumpir otra interrupción, es de suma importancia que las interrupciones sean transparentes.

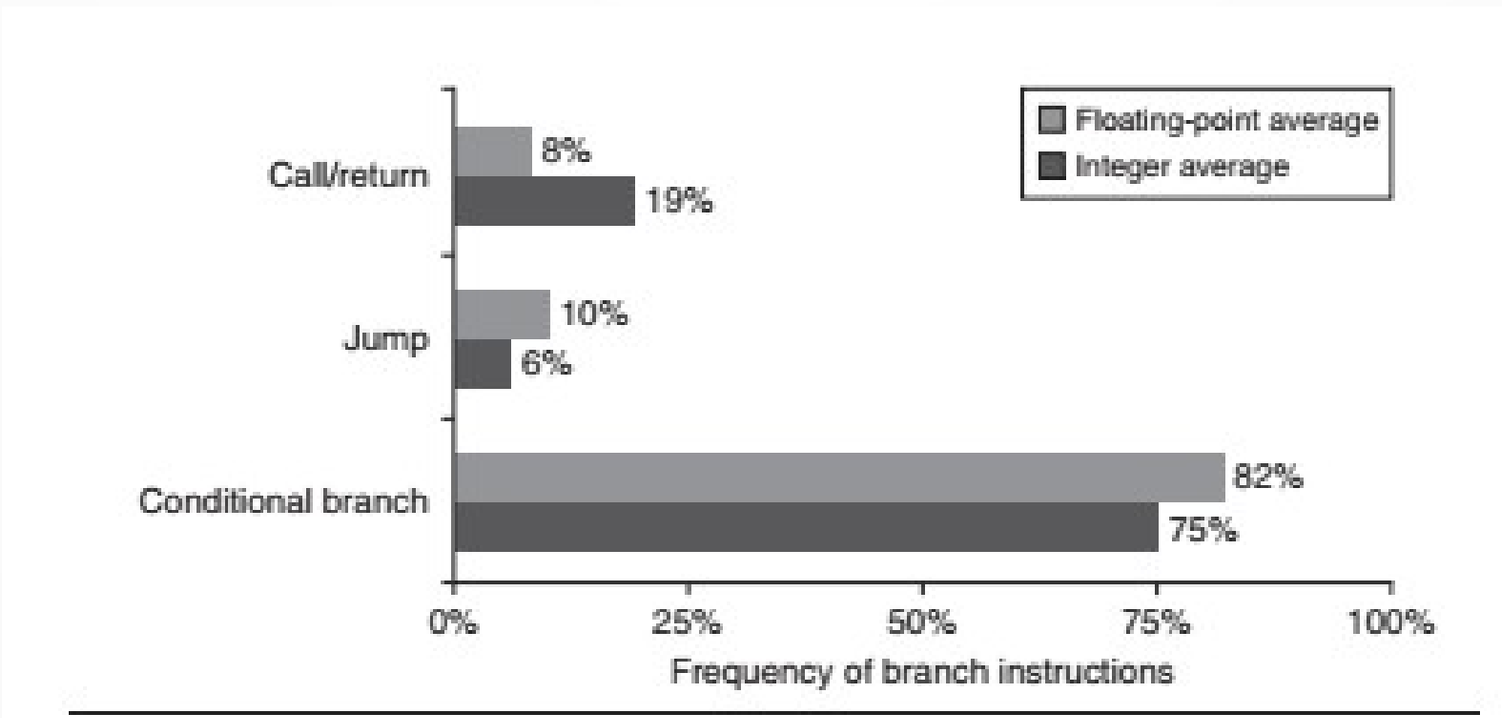
Control de Flujo



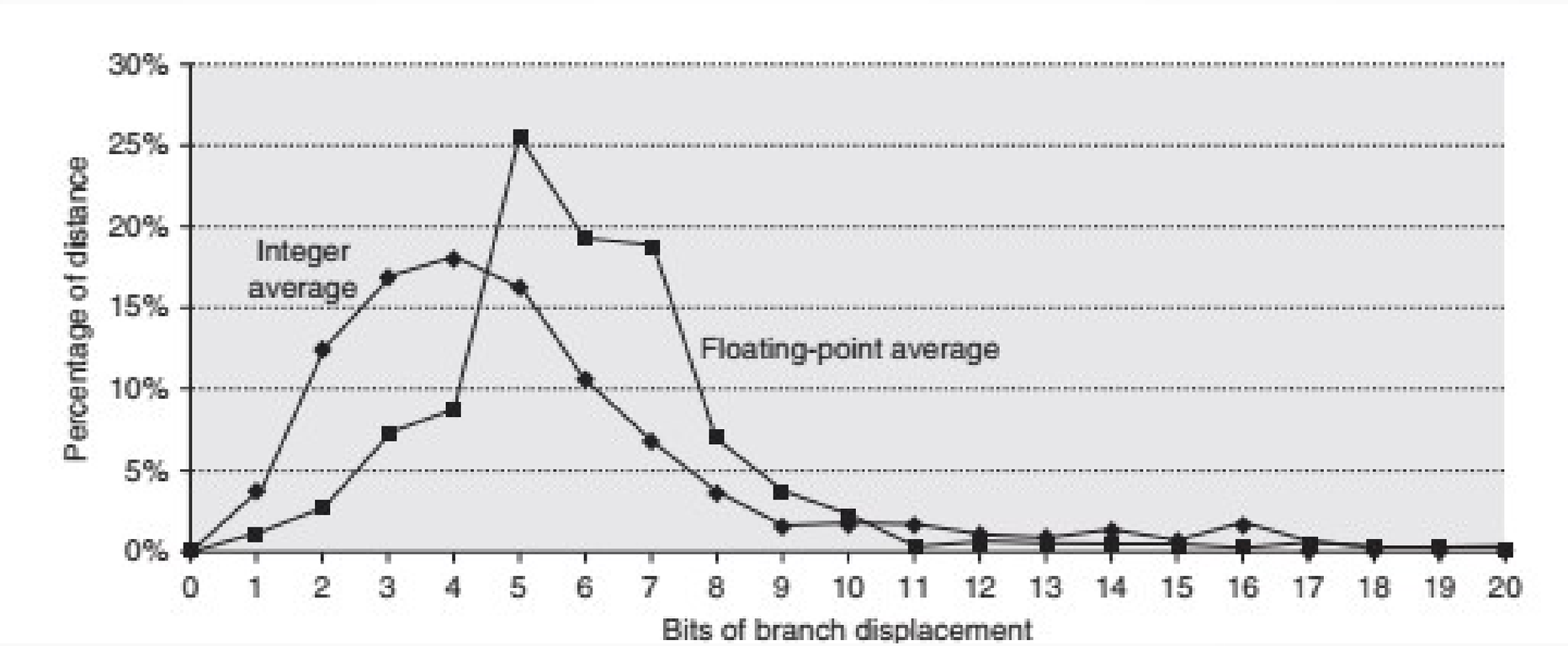
Gráficos Interesantes

Rank	80x86 instruction	Integer average (% total executed)
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
Total		**96%**

Gráficos Interesantes



Gráficos Interesantes



Gráficos Interesantes