Arquitectura de Computadores I: Principios del ISA

Luis Alberto Chavarría-Zamora

ITCR

lachavarria@tec.ac.cr

15 de agosto de 2023

Contenido

Lección Anterior

¿Qué vimos?

1. ¿Cuales son las diferentes categorías de ISA?
2. ¿Cómo se categorizan los ISAs según el almacenamiento interno en un procesador?
3. ¿Qué tipos de endianness existen y si los hay cómo se diferencian?
4. ¿Por qué es importante el alineamiento?
5. ¿Qué tipos de direccionamiento hay y cómo se relacionan?

Operandos

Operandos: Tipo y tamaño de operandos

El tipo de operandos normalmente se encuentra, en la mayoría de los casos, en el código de operación (opcode). Alternativamente los datos son anotados mediante etiquetas (*tags*) que son interpretadas por el HW (mnemónico).

Operandos

Operandos: Tipos comúnes

- Byte (8 bits).
- *Half-Word* (16 bits).
- *Word* (32 bits) / punto flotante de precisión simple (SP-FP).
- *Double Word* (64 bits) / punto flotante de precisión doble.

Enteros: Típicamente en representación complemento a 2.
Flotantes: IEEE 754*.
Caracteres: ASCII.
Decimal: BCD.

Operandos

Operandos: BCD

0 = 0000
1 = 0001
2 = 0010
3 = 0011
4 = 0100
5 = 0101
6 = 0110
7 = 0111
8 = 1000
9 = 1001

¿Cómo se representan decimales?

Operandos

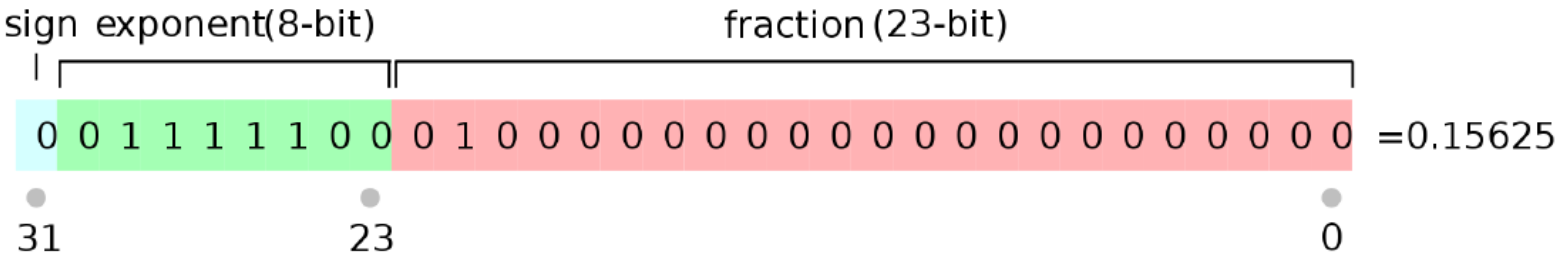
Operandos: BCD

0 = 0000
1 = 0001
2 = 0010
3 = 0011
4 = 0100
5 = 0101
6 = 0110
7 = 0111
8 = 1000
9 = 1001

¿Cómo se representan decimales?
Ejemplo: Representar $0{,}10_{10}$, en base 2: $0{,}0\overline{0011}$

IEEE 754

Representación Interna



El exponente es desplazado mediante un sesgo para poder representar exponentes negativos.

IEEE 754

Conversión de Decimal a Punto Flotante IEEE 754

1. Convertir valor decimal a binario (solo número ignorar el signo).
2. Colocar el número de la forma: numero$\times 2^0$.
3. Denotar el número de la forma 1, $a_1 a_2 \ldots a_i \times 2^n$ (se corre la coma n espacios).
4. Determinar el signo: 0 si N es mayor que 0, 1 si N es menor que 0.
5. Determinar el exponente como $E = n + 127$, luego pasar a binario.
6. Determinar la mantisa F como $F = a_1 a_2 \ldots a_i$.
7. Escribir el número según IEEE, completando con ceros a la derecha los 23 bits de la mantisa.

IEEE 754

Ejemplo: Convertir el número 22,625 a flotante

1. Paso 1: Convertir 22_{10} a binario: 10110_2. Convertir la parte decimal $0{,}625_{10}$ a binario: $0{,}625_{10} = 101_2$.
2. Paso 2: Por lo tanto $22{,}625_{10} = 10110{,}101_2$ en notación científica sería $22{,}625_{10} = 10110{,}101_2 \times 2^0$.
3. Paso 3: Se normaliza a $1{,}0110101_2 \times 2^4$. Por lo tanto $n = 4$.
4. Paso 4: Se obtiene el signo (0).
5. Paso 5: Se obtiene el exponente $E = 127 + n \rightarrow E = 127 + 4 = 131_{10} = 10000011_2$.
6. Paso 6: Se obtiene la mantisa $F = 01101010000000000000000_2$.
7. Paso 7: Se coloca en notación IEEE Signo + Exponente + Mantisa $= 01000001101101010000000000000000_2$.

IEEE 754

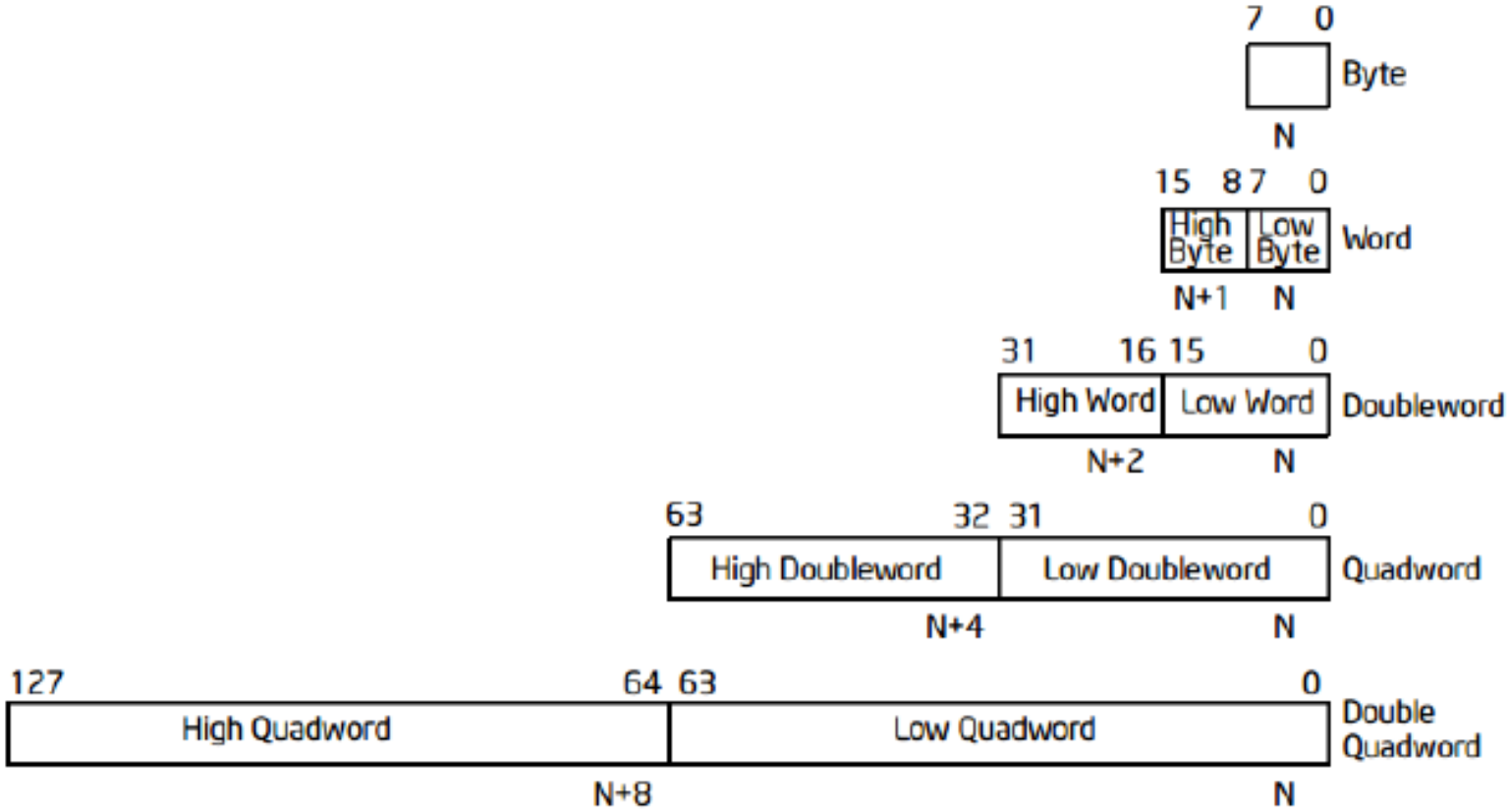
Práctica

1. Convertir $-83{,}7$ a formato IEEE precisión simple.
2. Convertir $0{,}5_{10}$ a IEEE754 precisión simple.
3. Convertir el número $C19E0000_16$ en formato IEEE754 en su equivalente decimal.

Decimal $\rightarrow$ IEEE 754 y IEEE 754 $\rightarrow$ Decimal

Operandos

Operandos: Ejemplo x86



Lección Anterior

Operandos: Ejemplo ARM

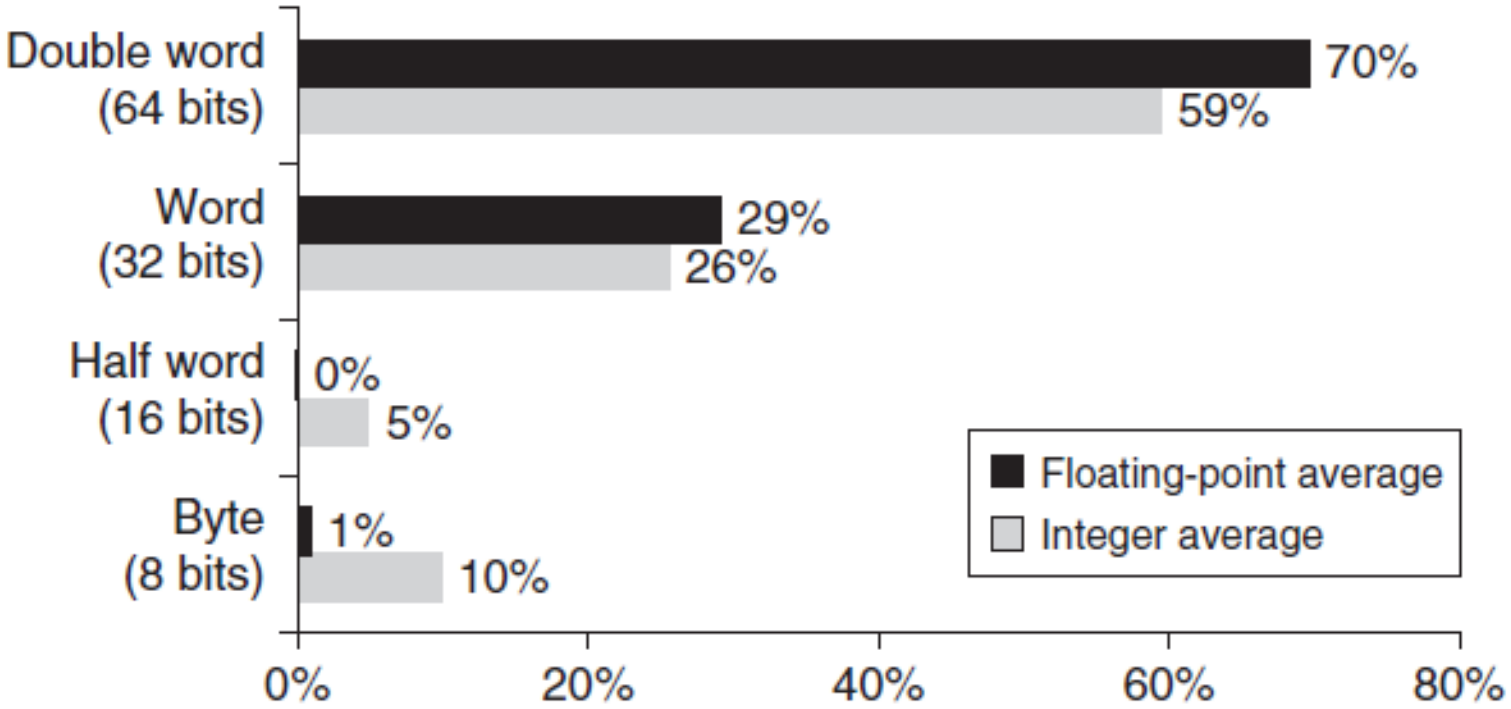
Type	A32	A64	Description
int/long	32-bit	32-bit	integer
short	16-bit	16-bit	integer
char	8-bit	8-bit	byte
long long	64-bit	64-bit	integer
float	32-bit	32-bit	single-precision IEEE floating-point
double	64-bit	64-bit	double-precision IEEE floating-point
bool	8-bit	8-bit	Boolean
wchar_t[a]	16-bit unsigned	16-bit unsigned	short (compiler dependent)
	32-bit unsigned	32-bit unsigned	int (compiler dependent)
void* pointer	32-bit	64-bit	addresses to data or code
enumerated types	32-bit	32-bit[b]	signed or unsigned integer
bit fields	not larger than their natural container size		
ABI defined extension types			
__int128/__uint128	128-bit	128-bit	signed/unsigned quadword
__f16	16-bit	16-bit	half precision

a. Environment-dependent. In GNU-based systems (such as Linux) this type is always 32-bit.

b. If the set of values in an enumerated type cannot be represented using either int or unsigned int as a container type, and the language permits extended enumeration sets, then a long long or unsigned long long container may be used.

Operandos

Benchmark: Distribución de accesos



Operandos

Tipos de operaciones

Existen varios tipos de operaciones soportadas por las arquitecturas. Las más comúnes son:

- Aritméticas y lógicas: Operaciones aritméticas y lógicas de enteros: sumar, restar y, o, multiplicar, dividir.
- Transferencia de datos: *Load-Store* (instrucciones de movimiento en computadoras con direccionamiento de memoria)
- Control de flujo: *Branch, jump, procedure call and return, traps*.
- Sistema: *Operating system call, virtual memory management instructions*.
- Punto flotante: Operaciones de punto flotante: sumar, multiplicar, dividir, comparar.
- Decimal: Decimal add, decimal multiply, decimal-to-character conversions.
- String:String move, string compare, string search.
- Gráficos: Pixel and vertex operations, compression/decompression operations.

Algunas arquitecturas presentan operaciones avanzadas para uso de strings o imágenes (tratamientos sobre píxeles, etc).

Control de Flujo

Instrucciones

Estas instrucciones son diferentes de otro tipo de instrucciones. Estas puede cambiar el flujo lineal de ejecución de un programa.

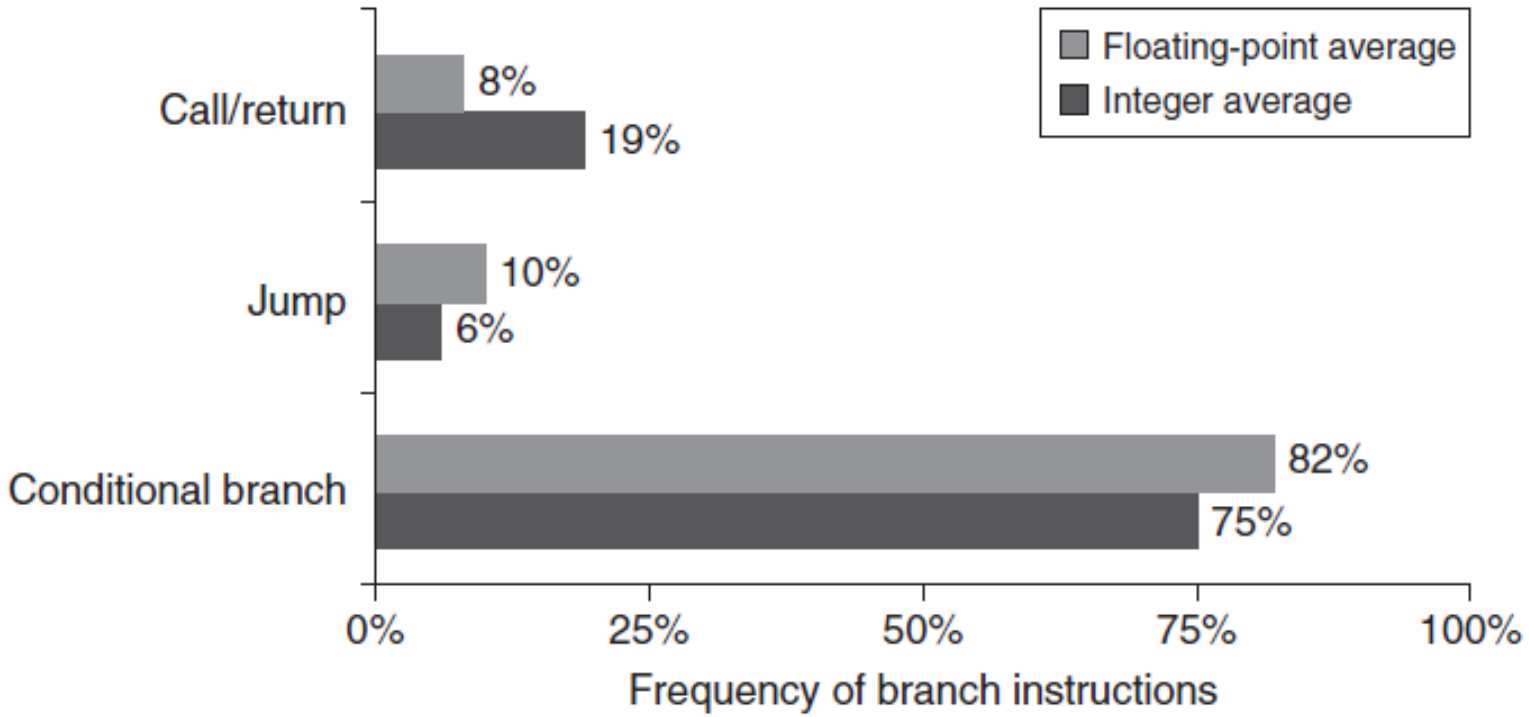
No hay un nombre consistente, en 1950 se llamaba *transfers*, en 1960 *branch*, luego se llaman *jumps*.

Cuatro tipos principales:

- *Conditional branches*.
- *Jumps*.
- *Procedure calls*.
- *Procedure returns*.

Control de Flujo

Instrucciones



Control de Flujo

Ejemplos: Jump x86

```
1  TOP:
2      SUB AX, BX           ;AX = AX - BX
3      JC  EXIT_:           ;IF AX < BX then jump to EXIT_
4      JMP TOP     ;else, repeat until AX < BX
5  EXIT_:
6      MOV AX, 4C00H
7      INT 21H
```

Control de Flujo

Ejemplos: CMP x86

```
1       MOV CX, 5
2   TOP_2:
3       CMP CX, 0       ; CX = 0 ?
4       JE  EXIT_:              ; IF CX = 0, then EXIT_
5       SUB AX, BX      ; AX = AX - BX
6       DEC CX
7       JMP TOP_2       ;else, repeat until CX = 0
8   EXIT_:
9       MOV AX, 4C00H
10      INT 21H
```

Control de Flujo

Ejemplos: IF-THEN-ELSE x86

- **IF-THEN-ELSE**

```
    CMP AL, BL          ; AL < BL?
    JA  ELSE_           ; no, go to "else"
    MOV DL, BL          ; then, result = BL
    JMP DISPLAY
ELSE_:                  ; else, AL >=BL
    MOV DL, AL          ; result = AL
DISPLAY:                ; display result
    MOV AH, 2
    INT 21H
```

Control de Flujo

Ejemplos: Case x86

- **CASE Structure**

```
        MOV AH, 1
        INT 21H
        CMP AL, 'A'
        JE CASE_A
        CMP AL, 'B'
        JE CASE_B
        CMP AL, 'C'
        JE CASE_C
        CMP AL, 'D'
        JE CASE_D

CASE_A:
             ; code for case A goes here
        JMP NEXT_
CASE_B:
             ; code for case B goes here
        JMP NEXT_
CASE_C:
             ; code for case C goes here
        JMP NEXT_
CASE_D:
             ; code for case D goes here

NEXT_:
             ; code continues here
```

Direccionamiento en Control de flujo

Dirección relativa

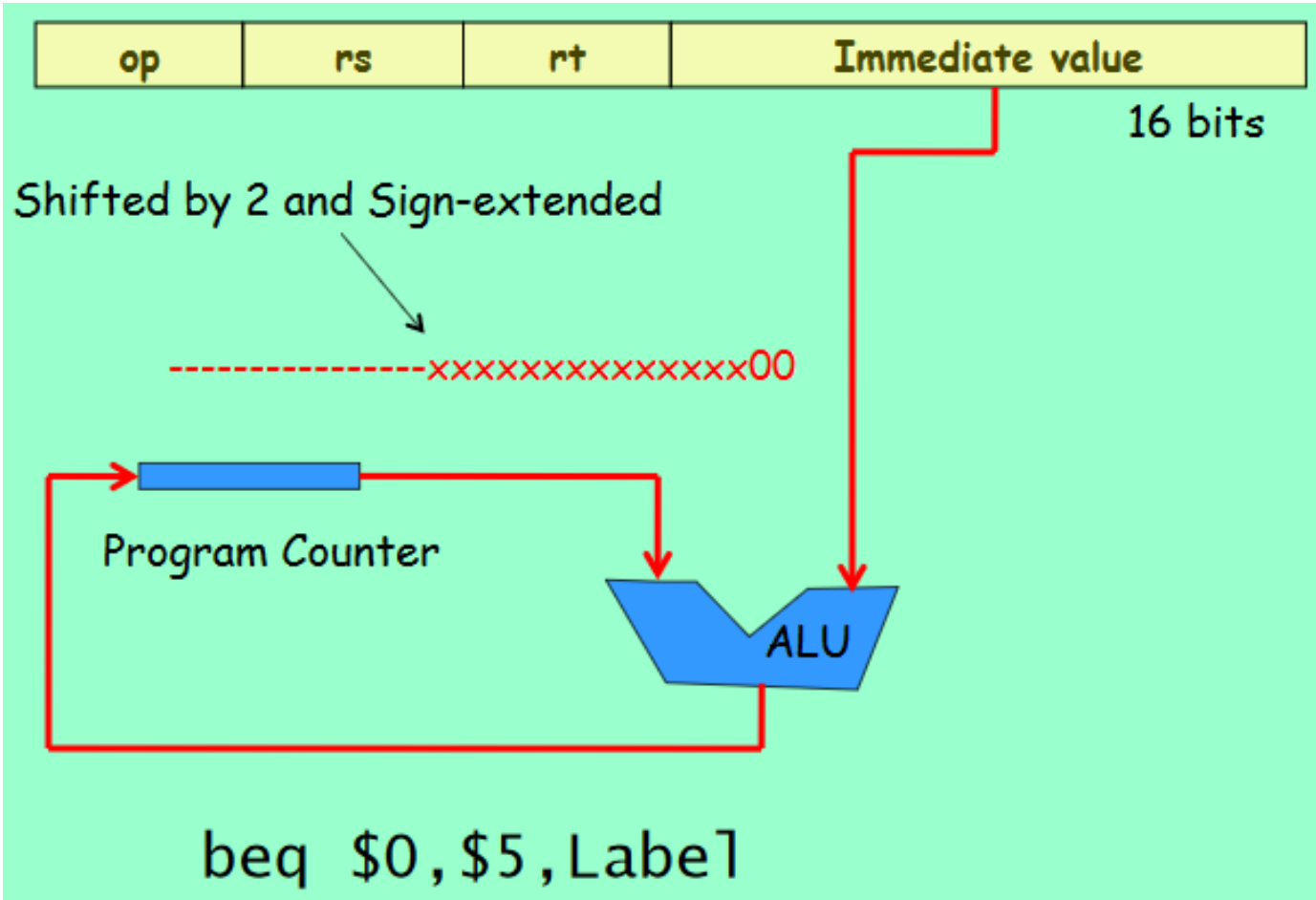
La dirección de destino de una instrucción de control de flujo siempre debe ser especificada: explícita o implícitamente.

Forma más común: **Dirección relativa al PC**.

- Requiere menos bits para especificar dirección.
- Dirección generalmente se encuentra cerca a la instrucción que está siendo ejecutada.
- Ocurre *position independence*, lo que le permite al código correrlo independientemente de donde sea cargado.

Direccionamiento en Control de flujo

Dirección relativa



Direccionamiento en Control de flujo

Dirección relativa

En algunos casos no es posible utilizar direcciones relativas al PC: La dirección de 'salto' **NO es conocida** en tiempo de compilación.

- Saltos indirectos.
- *Returns*.

Se debe especificar dinámicamente la dirección, **¿cómo?**

Direccionamiento en Control de flujo

Dirección relativa

En algunos casos no es posible utilizar direcciones relativas al PC: La dirección de 'salto' **NO es conocida** en tiempo de compilación.

- Saltos indirectos.
- *Returns*.

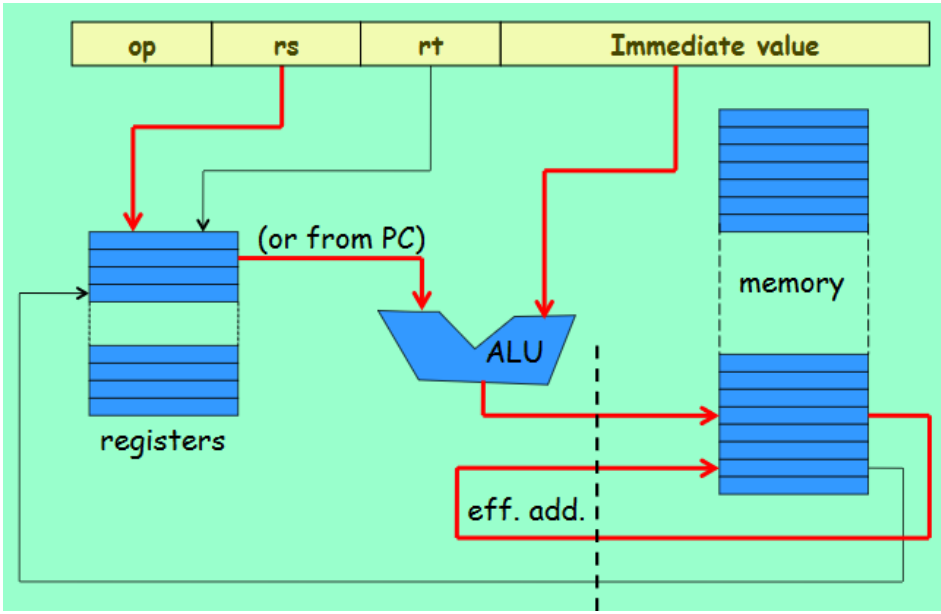
Se debe especificar dinámicamente la dirección, **¿cómo?**

Registros para almacenar la dirección, útil en estructuras *case-switch*, punteros a funciones, bibliotecas dinámicas.

Direccionamiento en Control de flujo

Dirección relativa

SPARC:	`jmpl %o7`
MIPS:	`jr $ra`
X86:(AT&T Syntax)	`jmp *%eax`
X86:(Intel Syntax)	`jmp eax`
ARM:	`mov pc, r2`
Itanium:	`br.ret.sptk.few rp`
6502:	`jmp ($0DEA)`
65C816:	`jsr ($0DEA,X)`
Z80:	`jp (hl)`
Intel 8080:	`pchl`
IBM System z	bcr cond,r1[3]



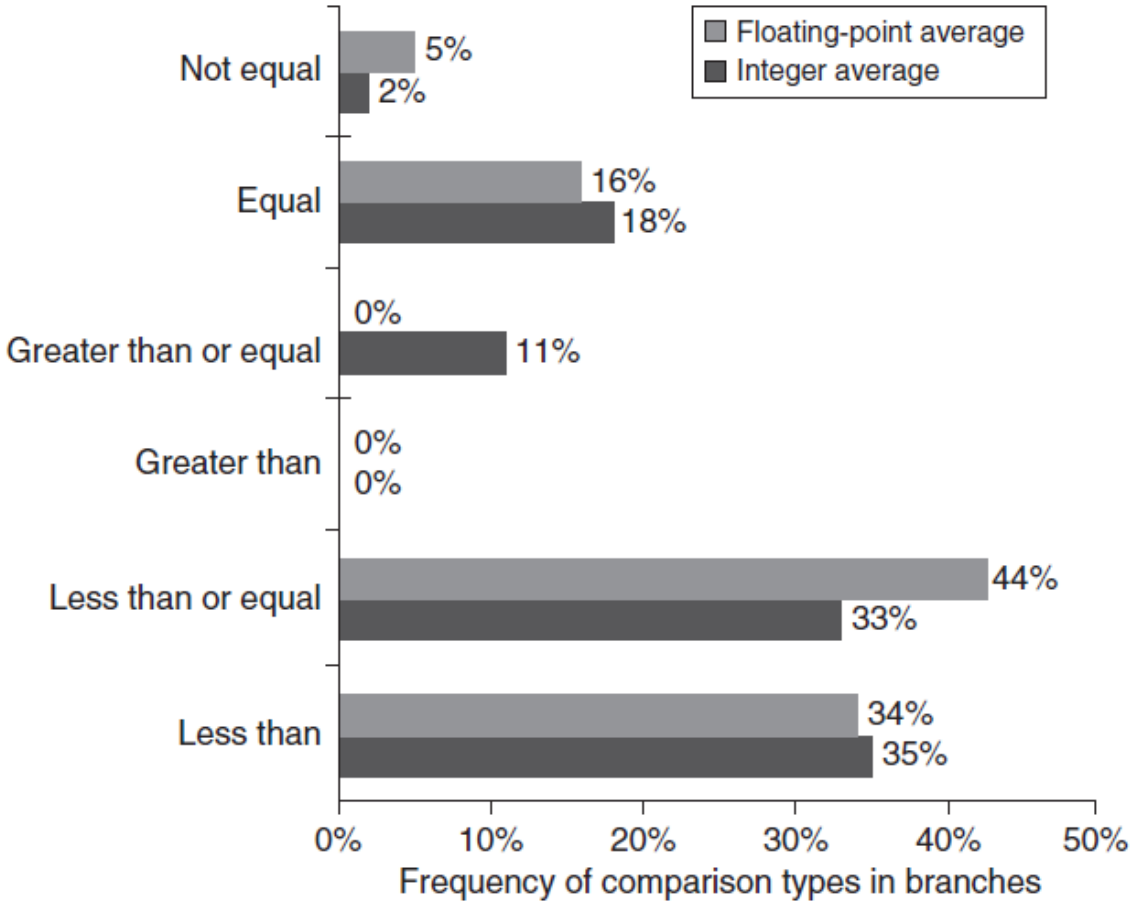
Direccionamiento en Control de flujo

Dirección relativa

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.

Direccionamiento en Control de flujo

Dirección relativa



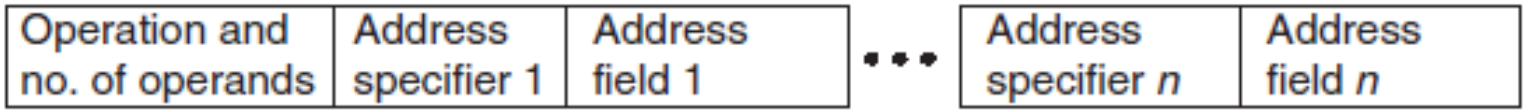
Codificación

Al tener un tamaño de palabra finito y determinado para las instrucciones, es vital maximizar su uso para poder describir cada instrucción.

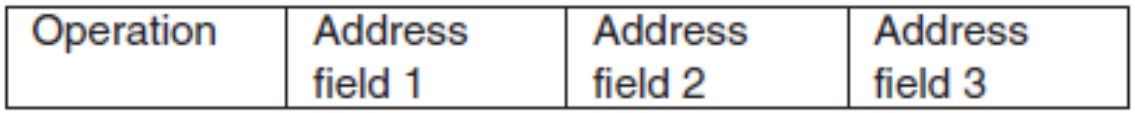
Respecto al tamaño de instrucción existen 3 formas comunes de longitud de instrucción:

- Variable $\rightarrow$ CISC.
- Fija $\rightarrow$ RISC.
- Híbrido.

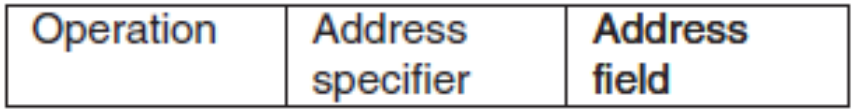
Codificación

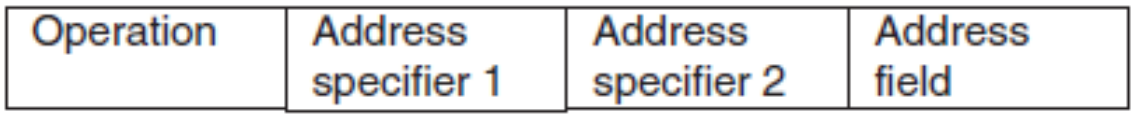


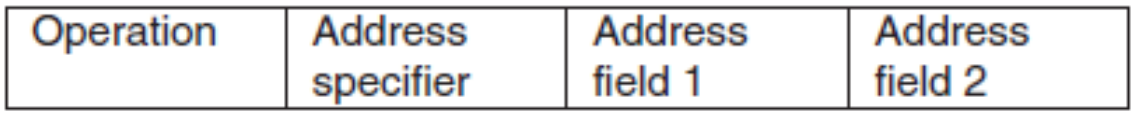
(a) Variable (e.g., Intel 80x86, VAX)



(b) Fixed (e.g., Alpha, ARM, MIPS, PowerPC, SPARC, SuperH)







(c) Hybrid (e.g., IBM 360/370, MIPS16, Thumb, TI TMS320C54x)

Codificación

Ejemplo

Un set de instrucciones de **16 bits**, está constituido por instrucciones de **0, 1 y 2 operandos**,los operandos tienen un tamaño de 6 bits, si en él ya existen 5 instrucciones de 2 operandos y 33 instrucciones de 0 operandos. ¿Cuál es el número máximo de instrucciones de 1 operando que se pueden codificar con dicho ancho de palabra y como sería la codificación del ISA?

Codificación

Ejemplo

Un set de instrucciones de **16 bits**, está constituido por instrucciones de **0, 1 y 2 operandos**,los operandos tienen un tamaño de 6 bits, si en él ya existen 5 instrucciones de 2 operandos y 33 instrucciones de 0 operandos. ¿Cuál es el número máximo de instrucciones de 1 operando que se pueden codificar con dicho ancho de palabra y como sería la codificación del ISA?

Se necesita diferenciar entre si es un operando de 0, 1 o 2 operandos. Para identificarlos se usan 2 bits.

Codificación

Ejemplo

Se necesita diferenciar entre si es un operando de 0, 1 o 2 operandos. Para identificarlos se usan 2 bits.

A	B	Y
0	0	0 op
0	1	1 op
1	0	2 op
1	1	X

Codificación

Ejemplo

Se necesita diferenciar entre si es un operando de 0, 1 o 2 operandos. Para identificarlos se usan 2 bits.

A	B	Y
0	0	0 op
0	1	1 op
1	0	2 op
1	1	X

Codificación

Ejemplo

Un set de instrucciones de **16 bits**, está constituido por instrucciones de **0, 1 y 2 operandos**,los operandos tienen un tamaño de 6 bits, si en él ya existen 5 instrucciones de 2 operandos y 33 instrucciones de 0 operandos. ¿Cuál es el número máximo de instrucciones de 1 operando que se pueden codificar con dicho ancho de palabra y como sería la codificación del ISA?

$$2 \text{ operandos} \rightarrow 12 \text{ bits} \rightarrow 5 \text{ instrucciones}$$

$$0 \text{ operandos} \rightarrow 0 \text{ bits} \rightarrow 33 \text{ instrucciones}$$

$$1 \text{ operandos} \rightarrow 6 \text{ bits} \rightarrow ?$$

Codificación

Ejemplo

$$2 \text{ operandos} \rightarrow 12 \text{ bits} \rightarrow 5 \text{ instrucciones}$$

15 14	13 ... 2	1 0
# Op	2 operandos (12 bits = 2 × 6 bits)	Id inst.

¿Se puede identificar 5 instrucciones con 2 bits?

Codificación

Ejemplo

$$2 \text{ operandos} \rightarrow 12 \text{ bits} \rightarrow 5 \text{ instrucciones}$$

15 14	13 ... 2	1 0
# Op	2 operandos (12 bits = 2 × 6 bits)	Id inst.

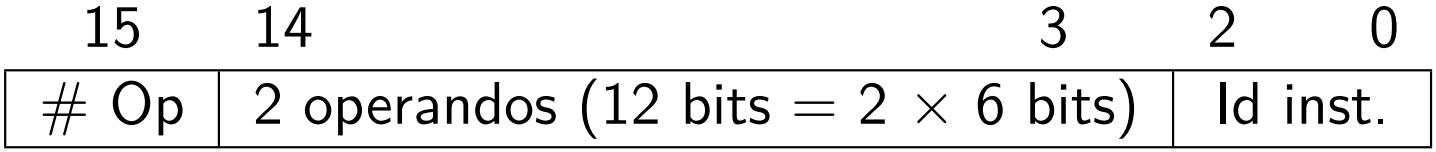
¿Se puede identificar 5 instrucciones con 2 bits?

A	B	Y
0	0	0 op
0	1	1 op
1	0	2 op
1	1	X

Codificación

Ejemplo

A	B	Y
0	0	0 op
0	1	1 op
1	0	2 op
1	1	



Ahora se pueden identificar 8 instrucciones de 2 operandos.
Hay 3 instrucciones disponibles.

Codificación

Ejemplo

$$0 \text{ operandos} \rightarrow 0 \text{ bits} \rightarrow 33 \text{ instrucciones}$$

15	14	13 ... 0
# Op		Id inst. (14 bits o 16384 instrucciones)

Quedan 16351 instrucciones disponibles.

Codificación

Ejemplo

$$0 \text{ operandos} \rightarrow 0 \text{ bits} \rightarrow 33 \text{ instrucciones}$$

15 14	13 0
# Op	Id inst. (14 bits o 16384 instrucciones)

Quedan 16351 instrucciones disponibles.

$$1 \text{ operandos} \rightarrow 6 \text{ bits} \rightarrow ?$$

15 14	13 8	7 0
# Op	1 operandos (6 bits)	Id inst. (**256 instrucciones**)

Codificación

Ejemplo

$0 \text{ operandos} \rightarrow 0 \text{ bits} \rightarrow 33 \text{ instrucciones}$

15 14	13 0
# Op	Id inst. (14 bits o 16384 instrucciones)

$1 \text{ operandos} \rightarrow 6 \text{ bits} \rightarrow \mathbf{256 \text{ instrucciones}}$

15 14	13 8	7 0
# Op	1 operandos (6 bits)	Id inst. (**256 instrucciones**)

$2 \text{ operandos} \rightarrow 12 \text{ bits} \rightarrow 5 \text{ instrucciones}$

15	14 3	2 0
# Op	2 operandos (12 bits $= 2 \times 6$ bits)	Id inst.

Referencias

J. Hennesy y D. Patterson (2012)

Computer Architecture: A Quantitative Approach. 5th Edition. Elsevier – Morgan Kaufmann.

J. González y R. García (2019)

Notas de clase de los profesores: Jeferson González y Ronald García.

ARMv8-A Architecture Reference Manual

Intel® 64 and IA-32 architectures software developer's manual combined volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4

Arquitectura de Computadores I: Principios del ISA

Luis Alberto Chavarría-Zamora

ITCR

lachavarria@tec.ac.cr

15 de agosto de 2023