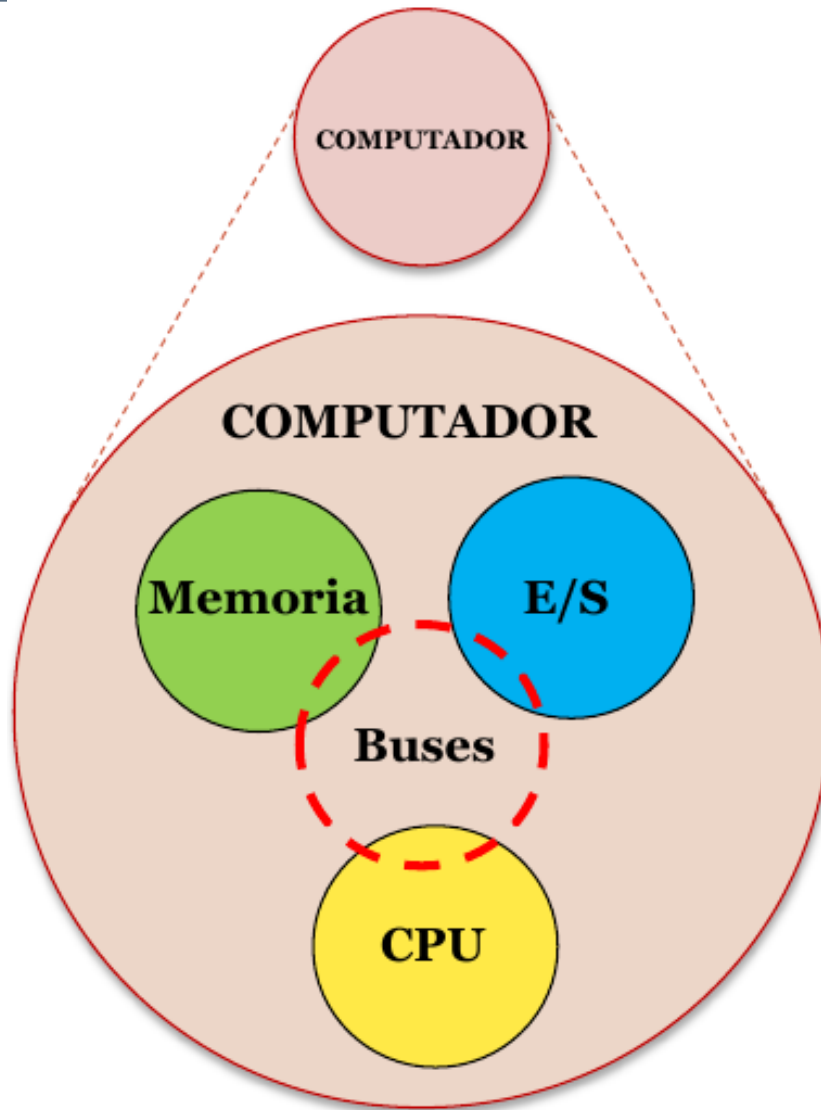


ARQUITECTURA DE COMPUTADORES

PROCESADOR Y SET INSTRUCCIONES

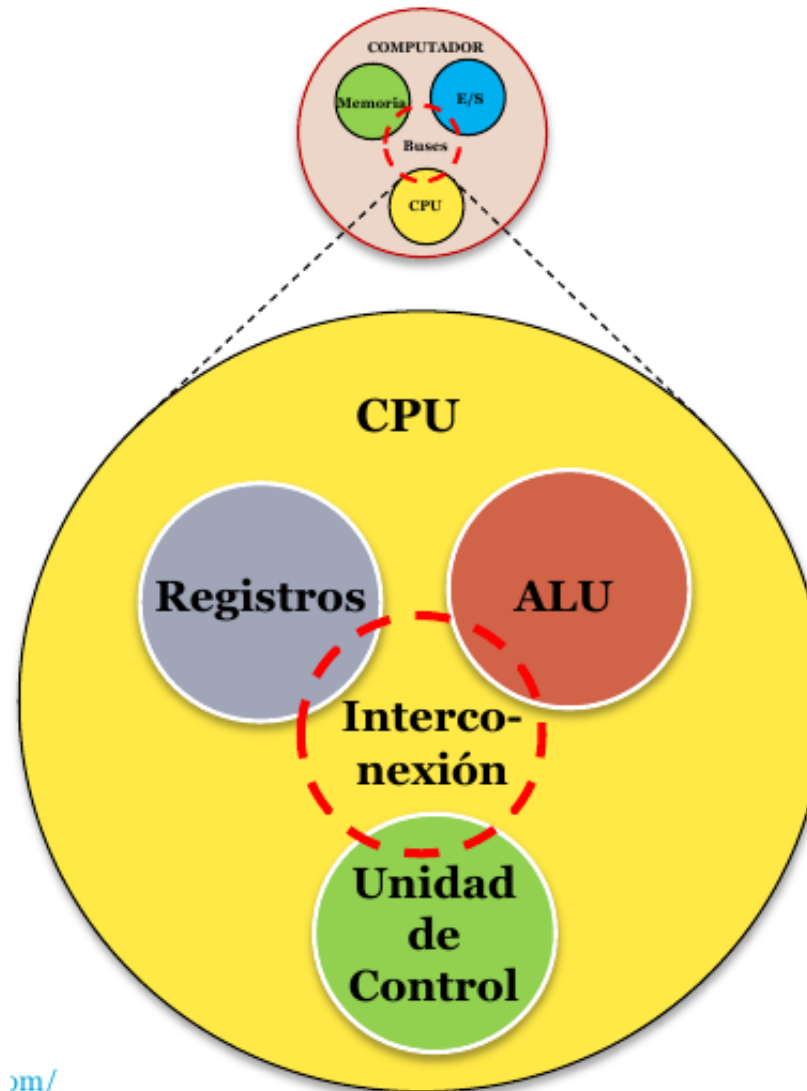
Ing Marlon Moreno Rincon

PROCESADOR.



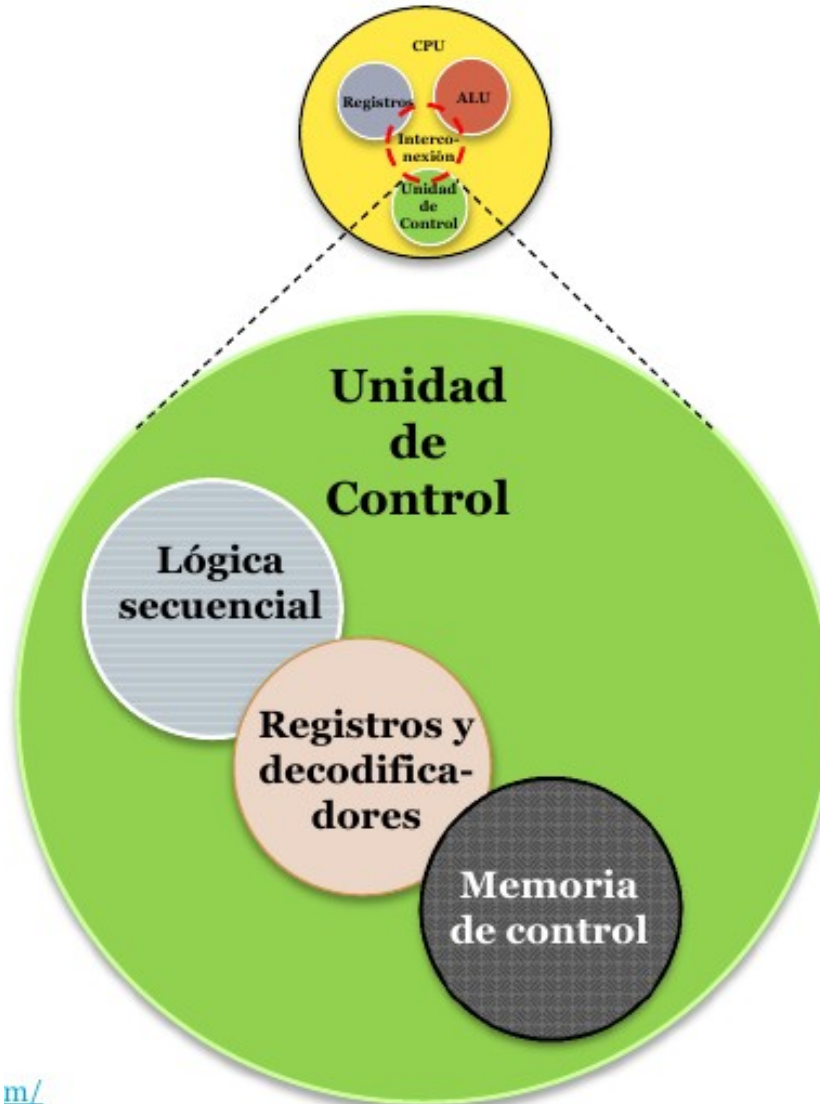
Entidad que interactuá con su exterior a través de periféricos, por los cuales recibe y procesa datos, para convertirlos en información conveniente y útil, que posteriormente se envía a otros periféricos que permiten la interpretación por el usuario.

PROCESADOR.



[om/](#)

PROCESADOR.



[m/](#)

PROCESADOR – SET DE INSTRUCCIONES

- ♦ *Un conjunto de datos insertados y codificados en una estructura específica que el procesador interpreta y ejecuta.*
- ♦ *En cada procesador los tipos de instrucciones permitidos están definidos y predeterminados en el conjunto de instrucciones (ISA, Instruction set architecture).*

CISC (Complex Instruction Set Computer)

- ♦ *Disponen de más de 80 instrucciones máquina en su repertorio, algunas de las cuales son muy sofisticadas y potentes.*
- ♦ *Requieren múltiples ciclos para su ejecución y buscan ortogonalidad.*
- ♦ *Una ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúan como macros.*
- ♦ *En esta arquitectura se dificulta el paralelismo entre instrucciones, por las diferencias de ciclos de instrucciones entre esta.*

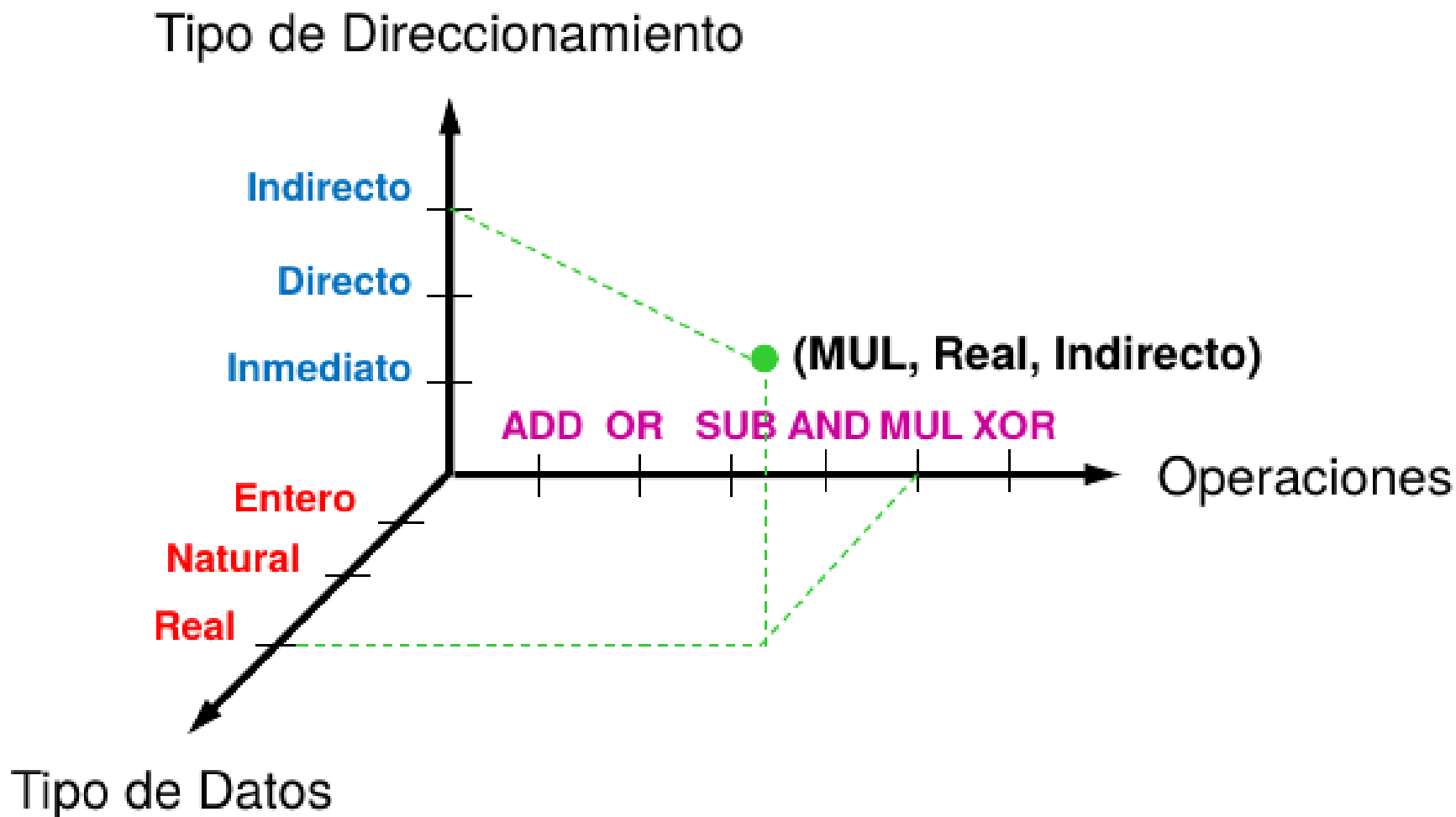
RISC (Reduced Instruction Set Computer)

- ♦ *El repertorio de instrucciones es reducido, instrucciones simples y se ejecutan en un solo ciclo maquina*
- ♦ *La sencillez y rapidez de las instrucciones permiten optimizar el hardware y software del procesador.*
- ♦ *Instrucciones de tamaño fijo, presentadas en un reducido número de formatos.*
- ♦ *Sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos.*
- ♦ *Gran numero de registros de propósito general.*

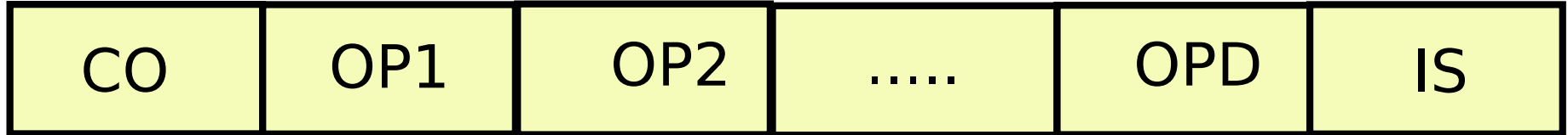
SISC (Simple Instruction Set Computing)

- ♦ *Set de instrucciones destinado a aplicaciones muy concretas.*
- ♦ *El juego de instrucciones, es reducido, específico.*
- ♦ *Las instrucciones se adaptan a las necesidades de la aplicación prevista.*

ORTOGONALIDAD.



FORMATO DEL SET DE INSTRUCCIONES.



CO = CÓDIGO DE OPERACIÓN.

OP1 = OPERANDO UNO.

OP2 = OPERANDO DOS.

OPD = OPERANDO DE DESTINO.

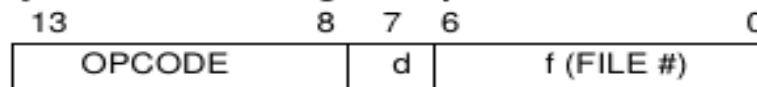
IS = INSTRUCCIONES SIGUIENTE.

FORMATO DEL SET DE INSTRUCCIONES.

- ♦ *El formato determina la cantidad de bits para la codificación de las instrucción y los campos para los términos citados.*
- ♦ *El numero de instrucciones y tipo de codificación determina la longitud, bits usados en el espació de código operación.*
- ♦ *Entre mas corta la codificación de la instrucción se puede buscar mas rápido en memoria de programa (FETCH), su decodificación se realiza mas rápido (DECODE) y requieren menor espacio de memoria.*

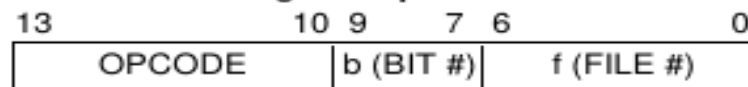
FORMATO DEL SET DE INSTRUCCIONES.

Byte-oriented file register operations



d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

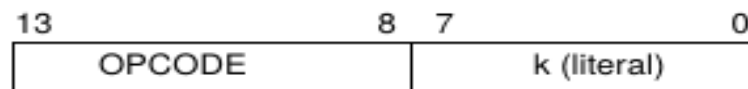
Bit-oriented file register operations



b = 3-bit bit address
f = 7-bit file register address

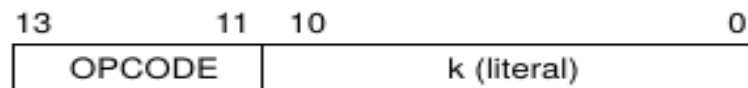
Literal and control operations

General



k = 8-bit immediate value

CALL and GOTO instructions only

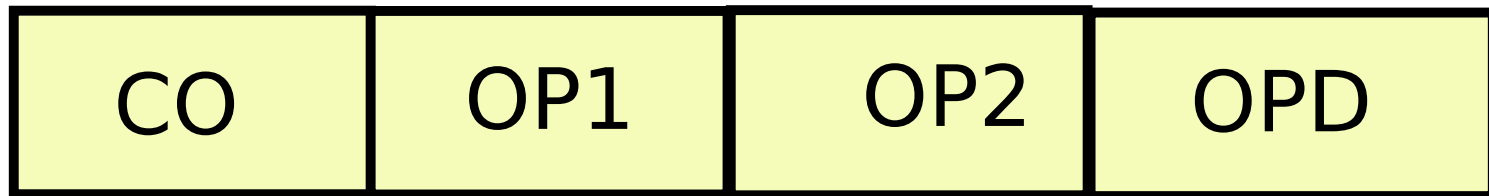


k = 11-bit immediate value

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

- *3 Operandos explícitos*

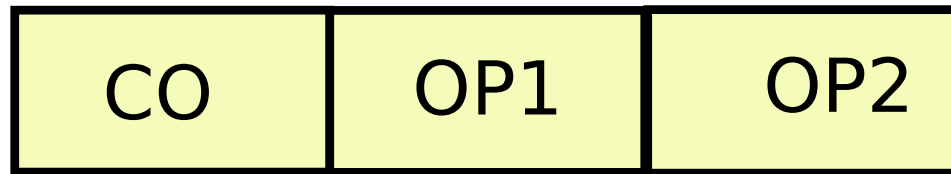


- ***Ejemplo: ADD B,C,A $B \leftarrow A + C$***
- ***MÁXIMA FLEXIBILIDAD.***
- ***OCUPA MUCHA MEMORIA SI LOS OPERANDOS NO ESTÁN EN REGISTROS.***

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

- *2 Operandos explícitos*

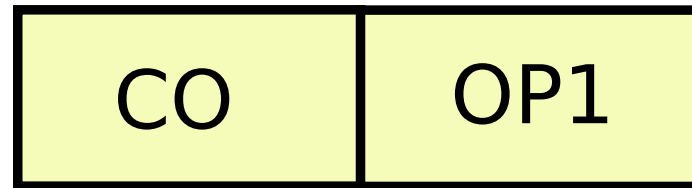


- **Ejemplo: *ADD B,C* $B \leftarrow B + C$**
- *Reduce el tamaño de las instrucciones.*
- *Se pierde un operando.*

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

- *1 Operando explícito.*

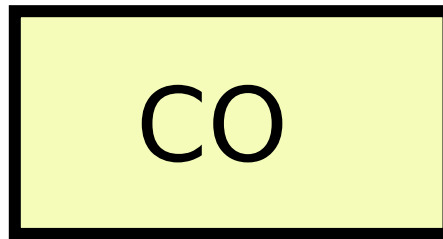


- Ejemplo: ADD B Acumulador \leftarrow Acumulador + B
- Supone que una de las fuentes y el destino son un mismo registro predefinido (Acumulador).
- Se pierden un operando fuente.

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

- 0 Operandos Explícitos.



- Ejemplo: $\text{ADD Cima de pila} = \text{Cima de pila} + \text{Cima de pila} - 1$
- Computadores que trabajan sobre pila.

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

$$E = (A-B) * (C+D)$$

3 Operandos		2 Operandos		1 Operandos		0 Operandos	
Add	C, D, C	Add	C, D	Load	C	Push	(Load D)
Sub	A, B, A	Sub	A, B	Add	D	Push	(Load C)
Mul	A, C, E	Mul	A, C	Store	C	Add	
		Mov	E, A	Load	A	Push	(Load B)
				Sub	B	Push	(Load A)
				Store	A	Sub	
				Mul	C	Mul	
				Store	E	Pop	(Store E)

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

3 Operandos		2 Operandos		1 Operando		0 Operandos	
Add	C, D, C	Add	C, D	Load	C	Push	(Load D)
Sub	A, B, A	Sub	A, B	Add	D	Push	(Load C)
Mul	A, C, E	Mul	A, C	Store	C	Add	
		Mov	E, A	Load	A	Push	(Load B)
				Sub	B	Push	(Load A)
				Store	A	Sub	
				Mul	C	Mul	
				Store	E	Pop	(Store E)

El registro destino se especifica al final
 $C \leftarrow C + D$

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

$$E = (A-B)*(C+D)$$

3 Operandos

Add C, D, C
Sub A, B, A
Mul A, C, E

2 Operandos

Add C, B
Sub A, B
Mul A, C
Mov E, A

$A \leftarrow A - B$

1 Operandos

C
Add D
Store C
Load A
Sub B
Store A
Mul C
Store E

0 Operandos

Push (Load D)
Push (Load C)
Add
Push (Load B)
Push (Load A)
Sub
Mul
Pop (Store E)

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

$$E = (A-B) * (C+D)$$

3 Operandos		2 Operandos		1 Operandos		0 Operandos	
Add	C, D, C	Add	E ← A x C		C	Push	(Load D)
Sub	A, B, A	Sub			D	Push	(Load C)
Mul	A, C, E	Mul	A, C	Store	C	Add	
		Mov	E, A	Load	A	Push	(Load B)
				Sub	B	Push	(Load A)
				Store	A	Sub	
				Mul	C	Mul	
				Store	E	Pop	(Store E)

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

3 Operandos			2 Operandos		1 Operando		0 Operandos	
Add	C, D, C		Add	C, D	Load	C	Push	(Load D)
Sub	A, B, A		Sub	A, B	Add	D	Push	(Load C)
Mul	A, C, E		Mul	A, C	Store	C	Add	
			Mov	E, A	Load	A	Push	(Load B)
					Sub	B	Push	(Load A)
					Store	A	Sub	
					Mul	C	Mul	
					Store	E	Pop	(Store E)

$E = (A - B) * C$

El destino se especifica al inicio
 $C \leftarrow C + D$

CLASIFICACIÓN DEL SET DE INSTRUCCIONES. NUMERO DE OPERANDOS EXPLICITOS.

$$E = (A-B)*(C+D)$$

3 Operandos			2 Operandos		1 Operandos		0 Operandos	
Add	C, D, C		Add	C, D	A ← A - B		Push	(Load D)
Sub	A, B, A		Sub	A, B			Push	(Load C)
Mul	A, C, E		Mul	A, C	Add	D	Add	
			Mov	E, A	Store	C	Push	(Load B)
					Load	A	Push	(Load A)
					Sub	B	Sub	
					Store	A	Mul	
					Mul	C	Pop	(Store E)
					Store	E		

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

$$E = (A-B)*(C+D)$$

3 Operandos		2 Operandos		1 Operandos	0 Operandos
Add	C, D, C	Add	C, D	A ← A x C	Push (Load D)
Sub	A, B, A	Sub	A, B		Push (Load C)
Mul	A, C, E	Mul	A, C	Store C	Add
		Mov	E, A	Load A	Push (Load B)
				Sub B	Push (Load A)
				Store A	Sub
				Mul C	Mul
				Store E	Pop (Store E)

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

$$E = (A-B)*(C+D)$$

3 Operandos		2 Operandos		1 Operandos		0 Operandos	
Add	C, D, C	Add	C, D	Load	C	Push	(Load D)
Sub	A, B, A	Sub	A, B	E ← A		Push	(Load C)
Mul	A, C, E	Mul	A, C			Pop	(Store E)
		Mov	E, A	Load	A	Push	(Load B)
				Sub	B	Push	(Load A)
				Store	A	Sub	
				Mul	C	Mul	
				Store	E	Pop	(Store E)

CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

$$E = (A-B)*(C+D)$$

3 Operandos		2 Operandos		1 Operandos		0 Operandos	
Add	C, D, C	Add	C, D	Load	C	Push	(Load D)
Sub	A, B, A	Sub	A, B	Add	D	Push	(Load C)
Mul	A, C, E	Mul	A, C	Store	C	Add	
		Mov	E, A	Load	A	Push	(Load B)
				Sub	B	Push	(Load A)
				Store	A	Sub	
				Mul	C	Mul	
				Store	E	Pop	(Store E)

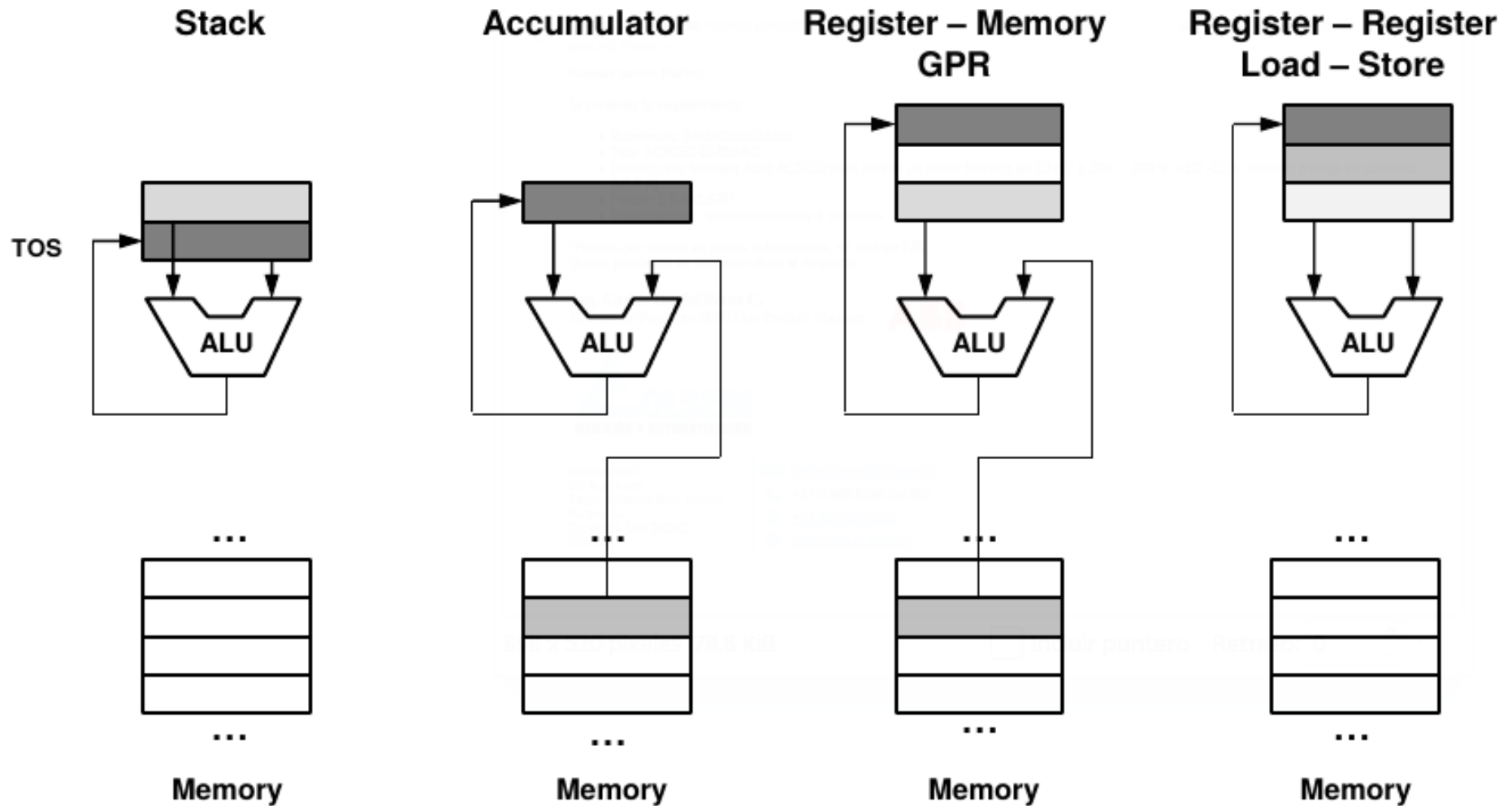
CLASIFICACIÓN DEL SET DE INSTRUCCIONES.

NUMERO DE OPERANDOS EXPLÍCITOS

$$C = (A + B) * C + A * (C - B)$$

3 OPERANDOS	2 OPERANDOS	1 OPERANDO	0 OPERANDO

CLASIFICACIÓN DEL SET DE INSTRUCCIONES: FORMA DE ALMACENAR LOS OPERANDO EN LA CPU



CLASIFICACIÓN DEL SET DE INSTRUCCIONES: FORMA DE ALMACENAR LOS OPERANDO EN LA CPU

- Arquitectura de pila.
- Arquitectura de acumulador.
- Arquitectura en base de registros.
 - Registro – Memoria.
 - Registro - Registro.

CLASIFICACIÓN DEL SET DE INSTRUCCIONES: FORMA DE ALMACENAR LOS OPERANDO EN LA CPU

Operandos en Memoria	Arquitectura
3 – 0	Registro – Registro (Load - Store) Utilizan 3 operandos con 0 en memoria. Formato de longitud fija y codificación simple. Ejm: SPARC, MIPS, Power PC
2 – 1	Registro – Memoria Utilizan 2 operandos con 1 ubicado en memoria. Ejm: Intel 80x86, Motorola 68000
3 – 3	Memoria – Memoria Utilizan 3 operandos ubicados en memoria. Ejm: VAX

CLASIFICACIÓN DEL SET DE INSTRUCCIONES: LONGITUD FIJA Y VARIABLE.

Código de operación de longitud fija.

- *Con un código de operación de n bits se tienen 2^n operaciones posibles. Cada una de las operaciones se identifica con un número único entre 0 y 2^n .*
- *Utilizando los bits de los operando se puede ampliar el número de operaciones posibles.*

CLASIFICACIÓN DEL SET DE INSTRUCCIONES: LONGITUD FIJA Y VARIABLE.

0000	R	OP	15 instrucciones de 2 operandos (CO de 4 bits)
0001	R	OP	
.	.	.	
.	.	.	
1110	R	.	15 instrucciones de 1 operando (CO de 8 bits)
1111 0000	.	.	
1111 0001	.	.	
.	.	.	
1111 1110	.	OP	2 ¹⁶ = 65.536 instrucciones de 0 operandos (CO de 24 bits)
1111 1111	0000 0000 0000 0000	.	
1111 1111	0000 0000 0000 0001	.	
.	.	.	
1111 1111	1111 1111 1111 1111	.	

CLASIFICACIÓN DEL SET DE INSTRUCCIONES: LONGITUD FIJA Y VARIABLE.

Código de operación de longitud variable.

- *El numero de bits del código de operación no esta predefinido. Se debe utilizar un algoritmo de optimización.*
 - ➔ *Frecuencia de aparición en el programa – Optimización de memoria.*
 - ➔ *Frecuencia de ejecución en el programa – Optimización trafico CPU-memoria.*

CLASIFICACIÓN DEL SET DE INSTRUCCIONES: LONGITUD FIJA Y VARIABLE.

Para optimizar el CO se puede utilizar la codificación de Huffman, que genera un código de longitud variable con la propiedad de no superposición de los CO resultantes. Garantiza que el CO de una determinada instrucción no coincide con la subcadena inicial de bits del CO de otra instrucción. La decodificación de un código de Huffman deberá realizarse de forma serie de izquierda a derecha.

Codificación Huffman es un algoritmo usado para compresión de datos.

CODIFICACIÓN HUFFMAN.

TIPO DE INSTRUCCIONES	FRECUENCIA DE EJECUCIÓN
ADD	0.53
SUB	0.25
MUL	0.12
DIV	0.03
STA	0.03
LDA	0.02
JMP	0.02

CODIFICACIÓN HUFFMAN.

1) Se escriben las instrucciones en una columna y a su derecha su frecuencia de ejecución. Cada elemento de la columna será un nodo terminal del árbol de decodificación.

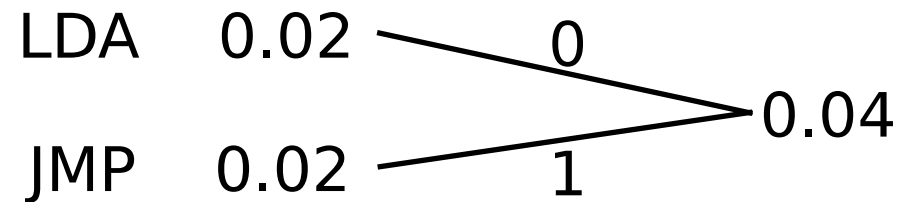
ADD 0.53

SUB 0.25

MUL 0.12

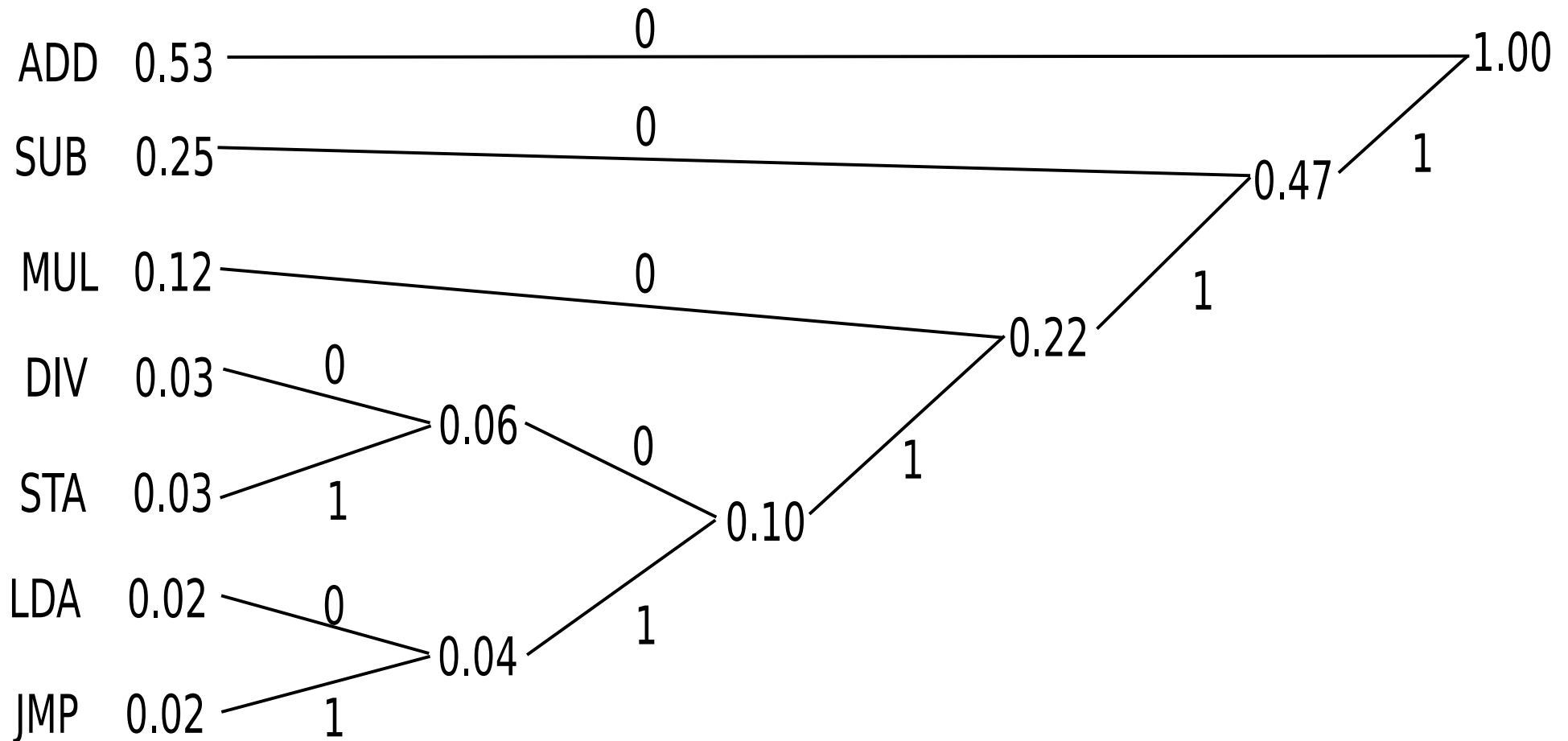
2) Se modifica la columna actual uniendo las dos frecuencias menores de dicha columna con arcos, obteniéndose un nuevo nodo cuyo valor será la suma de los nodos de procedencia.

CODIFICACIÓN HUFFMAN.



- 3) Se repite el paso 2 hasta llegar a la raíz del árbol que tendrá valor 1.
- 4) Se procede a asignar un 1 al arco inferior y 0 al arco superior hasta llegar a los nodos terminales
- 5) Para obtener el código de cada instrucción se recorre el árbol de la raíz a la instrucción concatenando cada uno de los valores de los arcos encontrados en el camino.

CODIFICACIÓN HUFFMAN.



CODIFICACIÓN HUFFMAN.

TIPO DE INSTRUCCIONES	FRECUENCIA DE EJECUCIÓN	CÓDIGO DE HUFFMAN
ADD	0.53	0
SUB	0.25	10
MUL	0.12	110
DIV	0.03	11100
STA	0.03	11101
LDA	0.02	11110
JMP	0.02	11111

$$I_m = \sum_i f_i * l_i = 0.53 * 1 + 0.25 * 2 + 0.12 * 3 + 0.03 * 5 + 0.03 * 5 + 0.02 * 5 + 0.02 * 5$$
$$1.89 \text{ bits} < 3 \text{ bits}$$

CODIFICACIÓN HUFFMAN.

TIPO DE INSTRUCCIONES	FRECUENCIA DE EJECUCIÓN	CÓDIGO DE HUFFMAN
ADD	0.32	
SUB	0.25	
MUL	0.2	
DIV	0.08	
STA	0.06	
LDA	0.02	
JMP	0.02	
SHR	0.02	
SHL	0.015	
COMP	0.015	