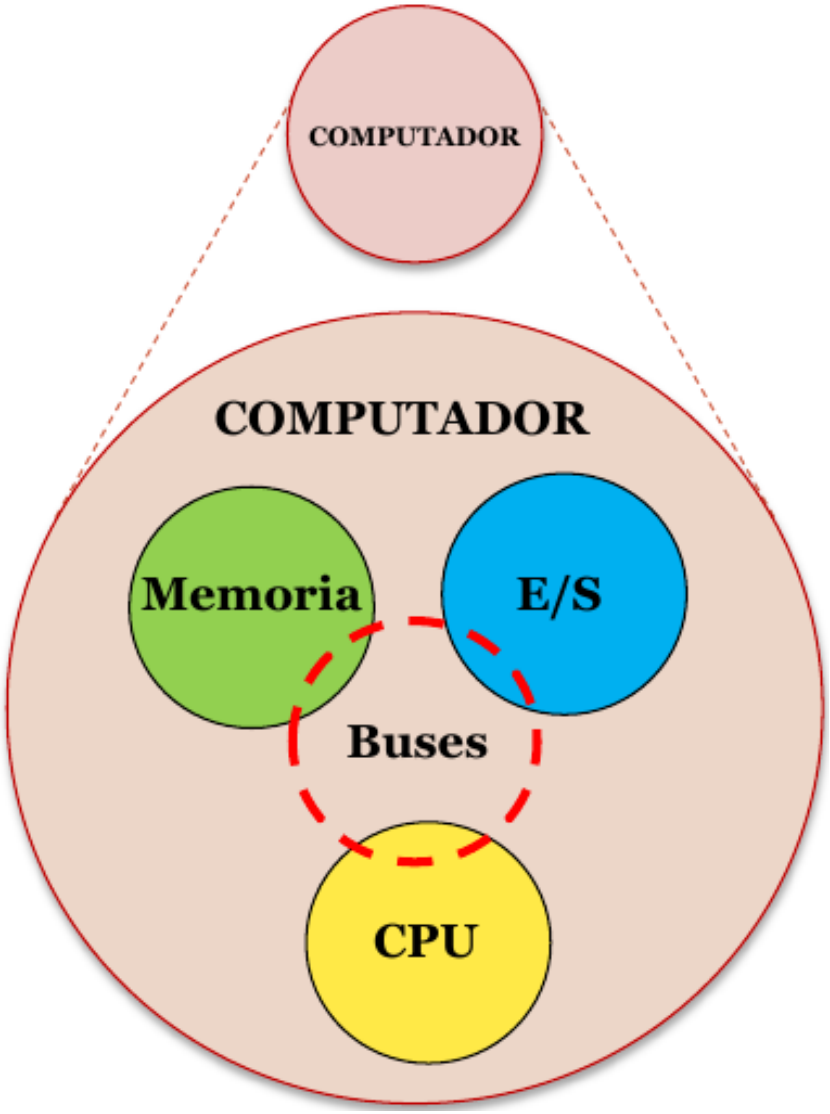
ARQUITECTURA DE COMPUTADORES

PROCESADOR Y SET INSTRUCCIONES

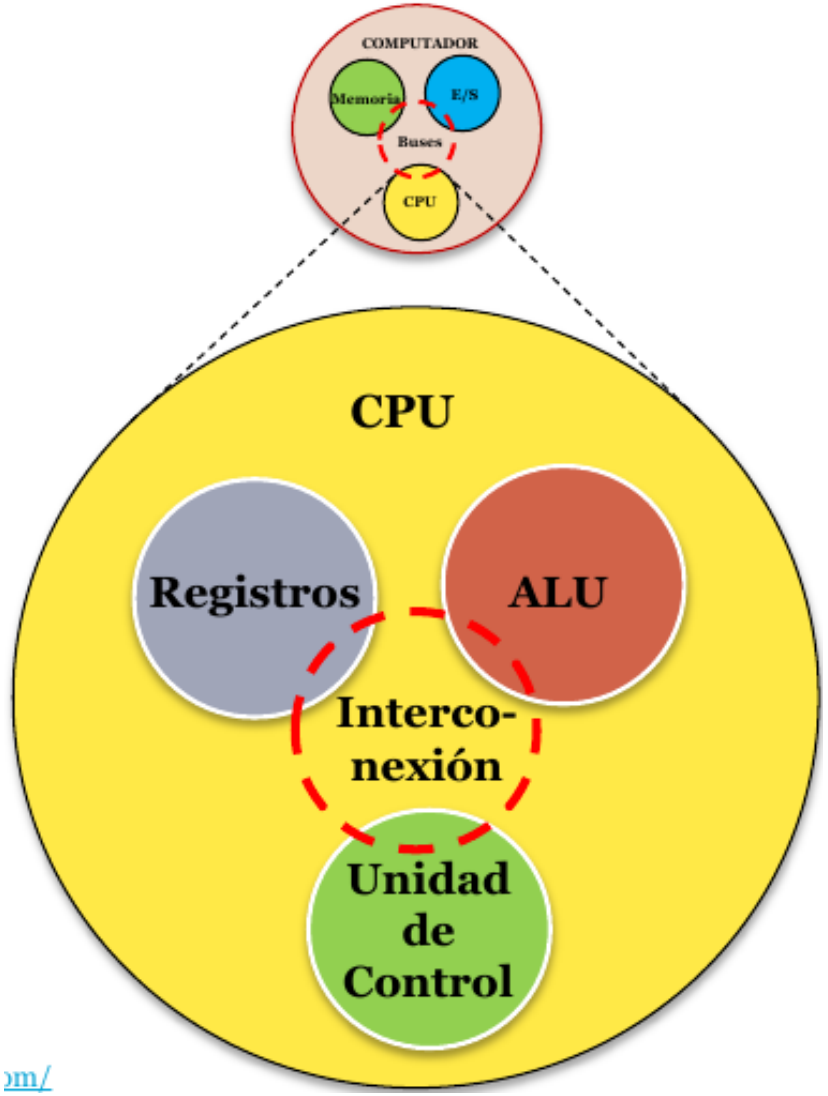
Ing Marlon Moreno Rincon

PROCESADOR

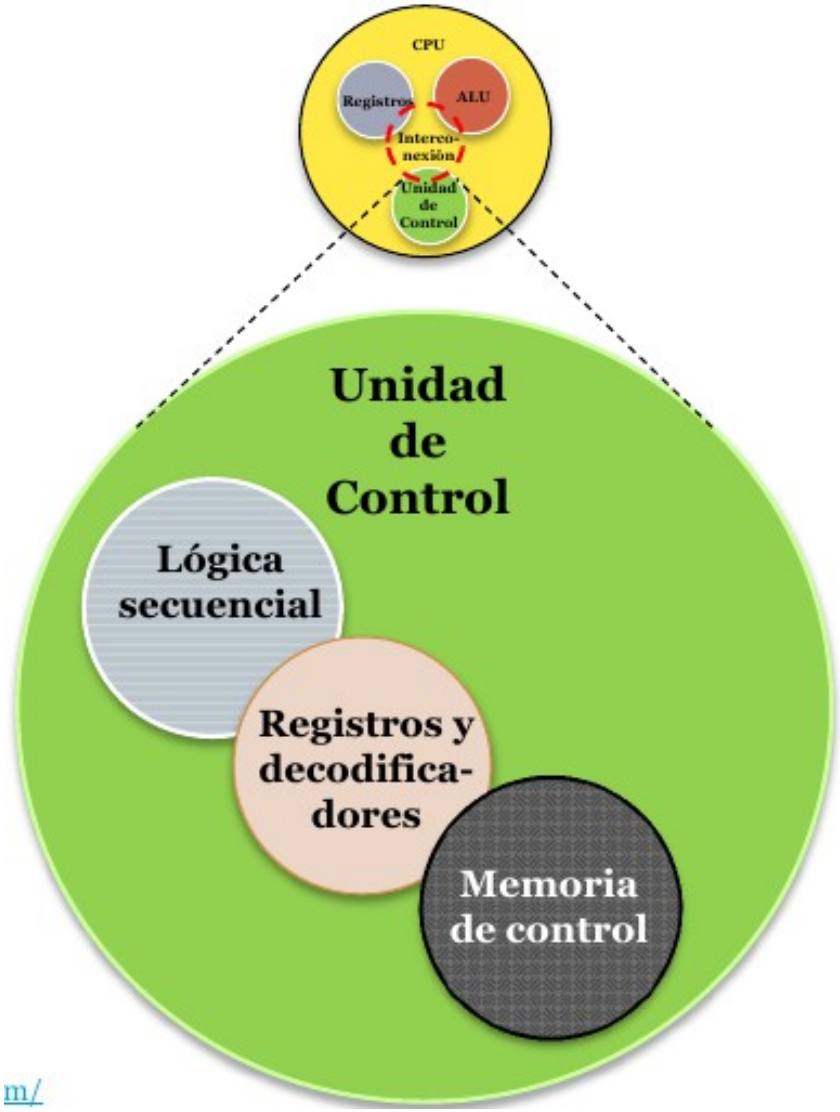


Entidad que interactuá con su exterior a través de periféricos, por los cuales recibe y procesa datos, para convertirlos en información conveniente y útil, que posteriormente se envía a otros periféricos que permite la interpretación la interpretación por el usuario.

PROCESADOR


COMPUTADOR
Memoria
E/S
Buses
CPU
CPU
Registros
ALU
Interco-
nexión
Unidad
de
Control

PROCESADOR.



m/

PROCESADOR – SET DE INSTRUCCIONES

Un conjuto de datos insertados y codificados en una estructura especifica que el procesador interpreta y ejecuta.

En cada procesador los tipos de instrucciones permitidos estan definidos y predeterminados en el conjuto de instrucciones (ISA, Instruction set arquitecture).

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 máquina es reducido y las instrucciones son simples y generalmente se ejecutan en un ciclo.

La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del processor.

Instrucciones de tamaño fijo y presentadas en un reducido número de formatos.

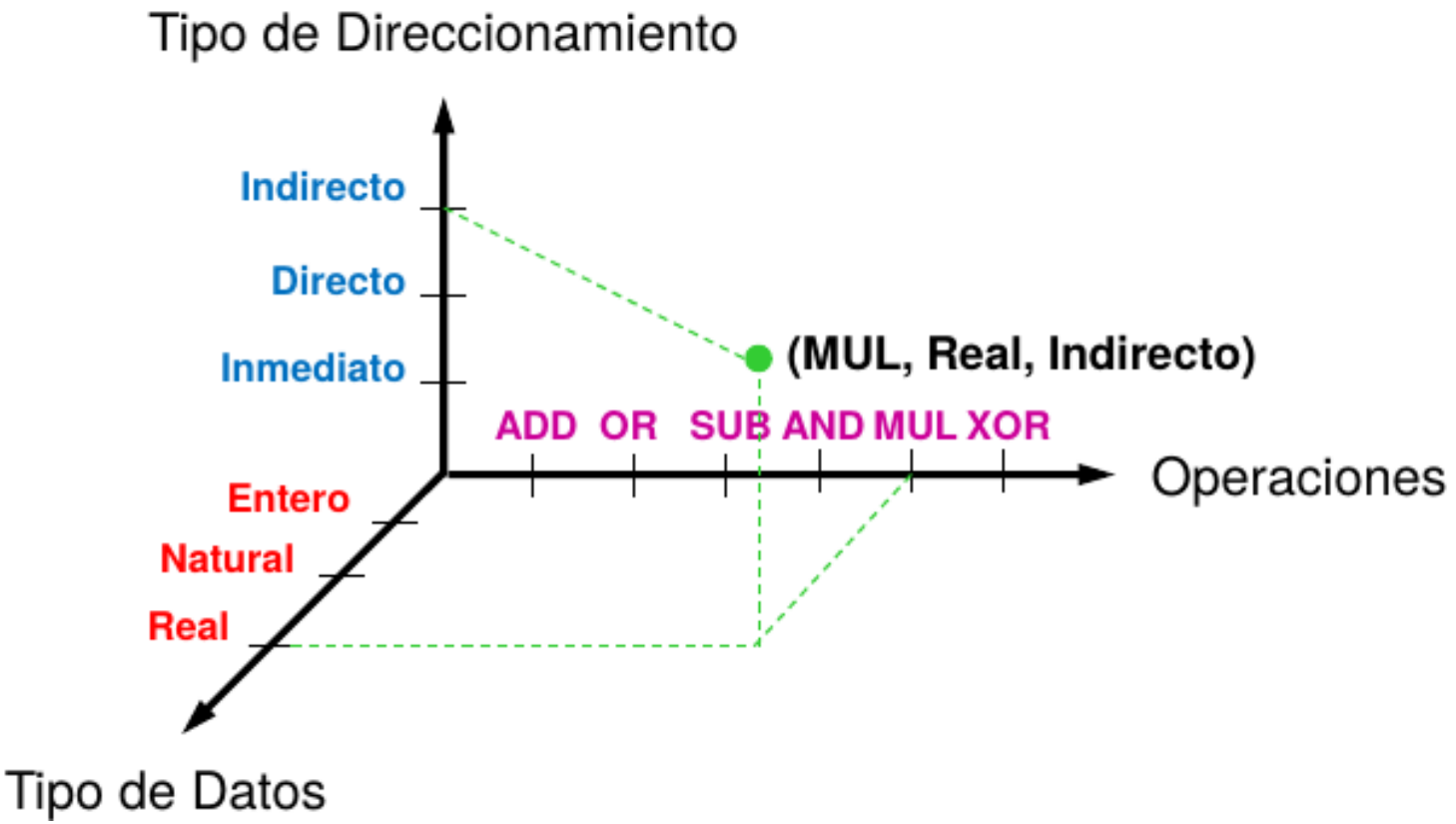
Sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos.

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 =CODIGO DE OPERACIÓN.

OP1 = OPERANDO UNO.

OP2 = OPERANDO DOS.

OPD = OPERANDO DE DESTINO.

IS = INSTRUCCIONES SIGUIENTE.

FORMATO DEL SET DE INSTRUCCIONES.

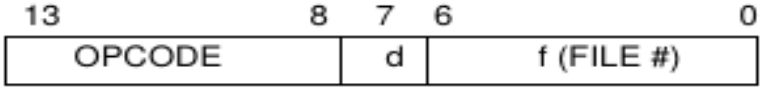
El formato determinar la cantidad de bits para la codificación de las instrucción y los campos para los terminos citados.

El numero de instrucciones y tipo de codificación determina la longuitud bits usados en el espación de codigo operación.

Entre mas corta la codificación de la instrucción se puede buscar mas rapido en memoria de programa (FETCH), su decodificación se realiza mas rapido (DECODE) y requieren menor espacio de memoria.

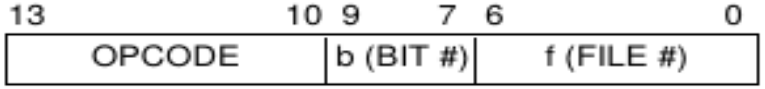
FORMATO DEL SET DE INSTRUCCIONES.

Byte-oriented file register operations



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

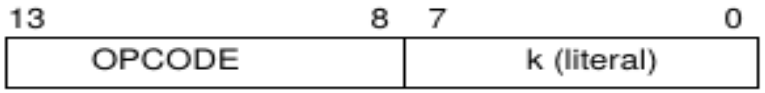
Bit-oriented file register operations



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

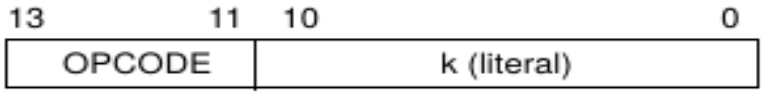
Literal and control operations

General



k = 8-bit immediate value

CALL and GOTO instructions only



k = 11-bit immediate value

CLASIFICACIÓN DEL SET DE INSTRUCCIONES. NUMERO DE OPERANDOS EXPLICITOS.

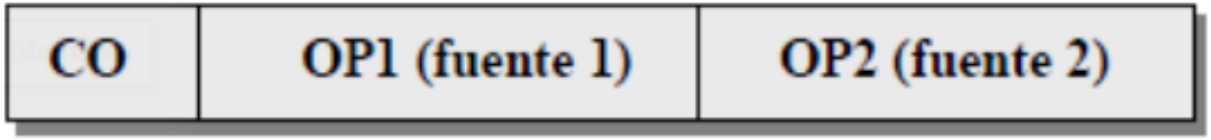
- 3 Operandos Explícitos.

CO	OP1 (fuente 1)	OP2 (fuente 2)	OP3 (destino)

  - Ejemplo: ADD B,C,A $A \leftarrow B + C$
  - Máxima flexibilidad
  - Ocupa mucha memoria si los operandos no están en registros

CLASIFICACIÓN DEL SET DE INSTRUCCIONES. NUMERO DE OPERANDOS EXPLICITOS.

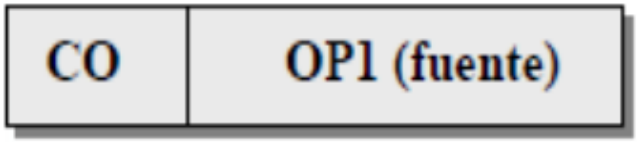
- 2 Operandos Explícitos.



  - Ejemplo: ADD B,C $B \leftarrow B + C$
  - Reduce el tamaño de la instrucción
  - Se pierde uno de los operandos

CLASIFICACIÓN DEL SET DE INSTRUCCIONES. NUMERO DE OPERANDOS EXPLICITOS.

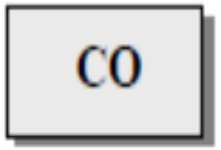
- 1 Operando Explícito.



  - Ejemplo: ADD B Acumulador ← <Acumulador> + B
  - Supone que tanto fuente como destino son un mismo registro predeterminado (Acumulador)
  - Se pierde un operando fuente

CLASIFICACIÓN DEL SET DE INSTRUCCIONES. NUMERO DE OPERANDOS EXPLICITOS.

- 0 Operandos Explícitos.



  - Ejemplo: ADD Cima de la Pila ← <Cima de la Pila> + <Cima de la Pila - 1>
  - Computadores que trabajan sobre una pila

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	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 EXPLICITOS.

El registro destino se especifica al final
C ← C + D

3 Operandos		2 Op		andos		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 EXPLICITOS.

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

3 Operandos		2 Operandos		1 Operandos		0 Operandos	
Add	C, D, C	Add		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)

A ← A - B

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	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)

$E \leftarrow A \times C$

CLASIFICACIÓN DEL SET DE INSTRUCCIONES. NUMERO DE OPERANDOS EXPLICITOS.

E = (A-B)*

El destino se especifica al inicio
C ← 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 EXPLICITOS.

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

3 Operandos		2 Operandos		1 Operandos		0 Operandos	
Add	C, D, C	Add	C, D			ush	(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)

A ← A - B

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 x C		ush	(Load D)
Sub	A, B, A	Sub	A, B			ush	(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 EXPLICITOS.

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			Push	(Load C)
Mul	A, C, E	Mul	A, 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

CLASIFICACIÓN DEL SET DE INSTRUCCIONES. NUMERO DE OPERANDOS EXPLICITOS.

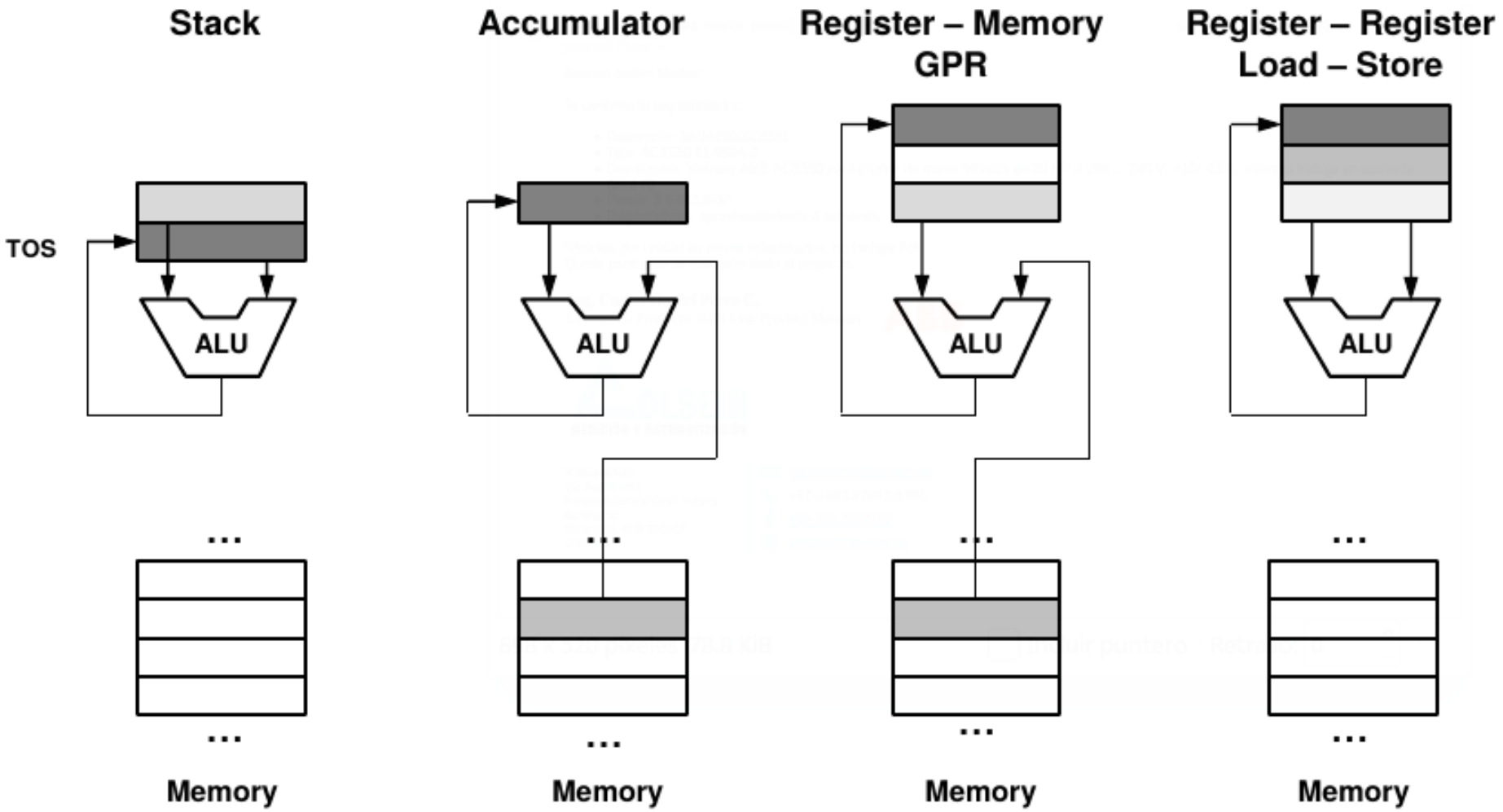
E = (A-B)*(C+D)

3 Operandos		2 Operandos		1 Operandos		0 Operandos	
Add	C, D, C	Add	C, D	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: 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



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: LONGUITUD FIJA Y VARIABLE.

Código de operación de longuid fija.

- *Con un codigo de operación de n bits se tienen 2^n operaciones posibles. Cada una de las operaciones se indentifica con un numero unico entre 0 y 2^n.*
- *Utilizando los bits de los operando se puede ampliar el numero de operaciones posibles.*

CLASIFICACIÓN DEL SET DE INSTRUCCIONES: LONGUITUD FIJA Y VARIABLE.

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

CLASIFICACIÓN DEL SET DE INSTRUCCIONES: LONGUITUD FIJA Y VARIABLE.

Código de operación de longuid variable.

- ***El numero de bits del codigo 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: LONGUITUD 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

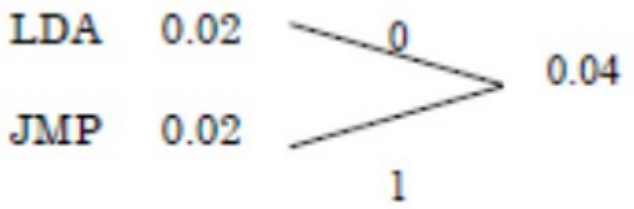
CODIFICACIÓN HUFFMAN.

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.

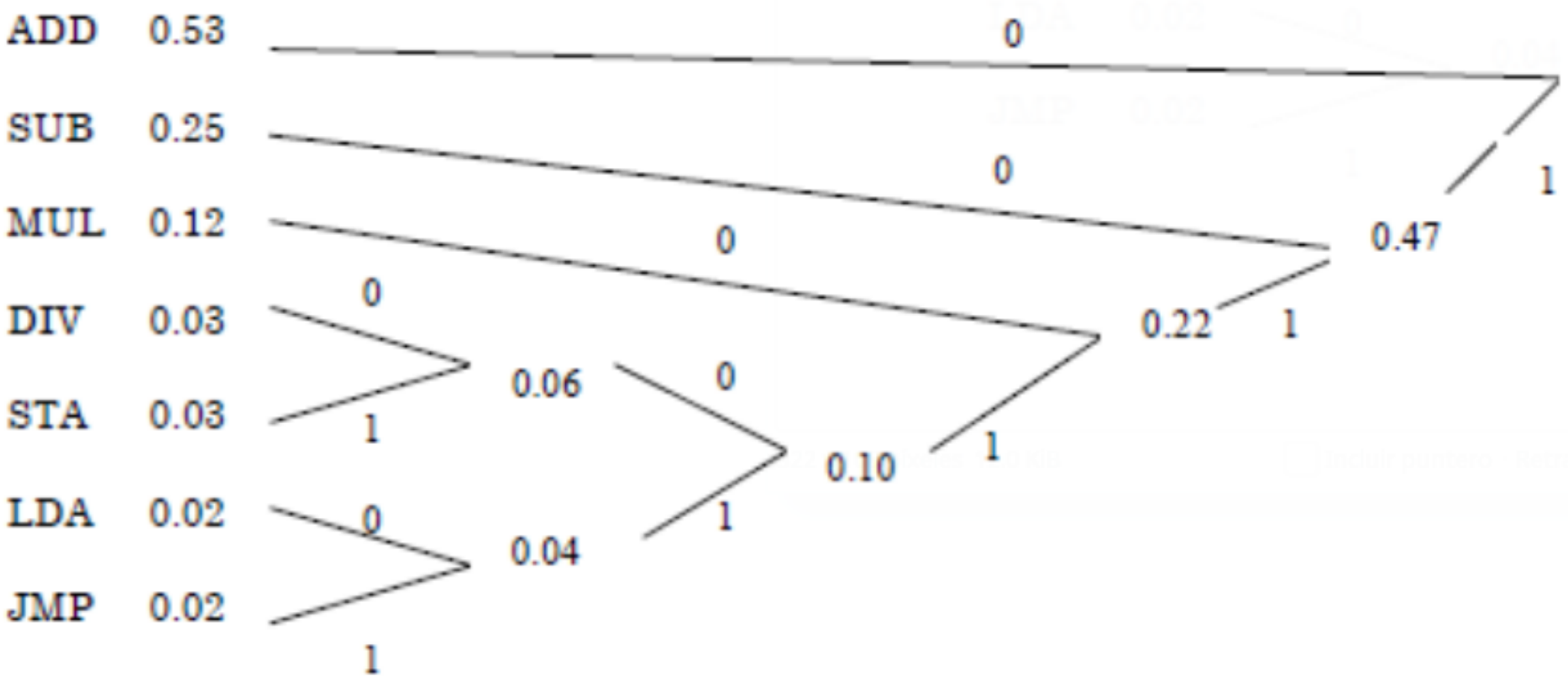


3) Se repite el paso 2 hasta llegar a la raíz del árbol que tendrá valor 1.

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

Resulta códigos de 1, 2, 3 y 5 bits con una longitud media l_n dada por la siguiente expresión:

$$l_m = \sum_i f_i \times l_i = 0.53 \times 1 + 0.25 \times 2 + 0.12 \times 3 + 0.003 \times 5 + 0.003 \times 5 + 0.02 \times 5 + 0.02 \times 5 =$$

$$1.89\_bits < 3\_bits$$