

Clase 09 - Representaciones numéricas - Números Racionales

Profesor: **IIC2343 - Arquitectura de Computadores**
- Felipe Valenzuela González
Correo:
frvalenzuela@alumni.uc.cl

Resumen de la clase pasada

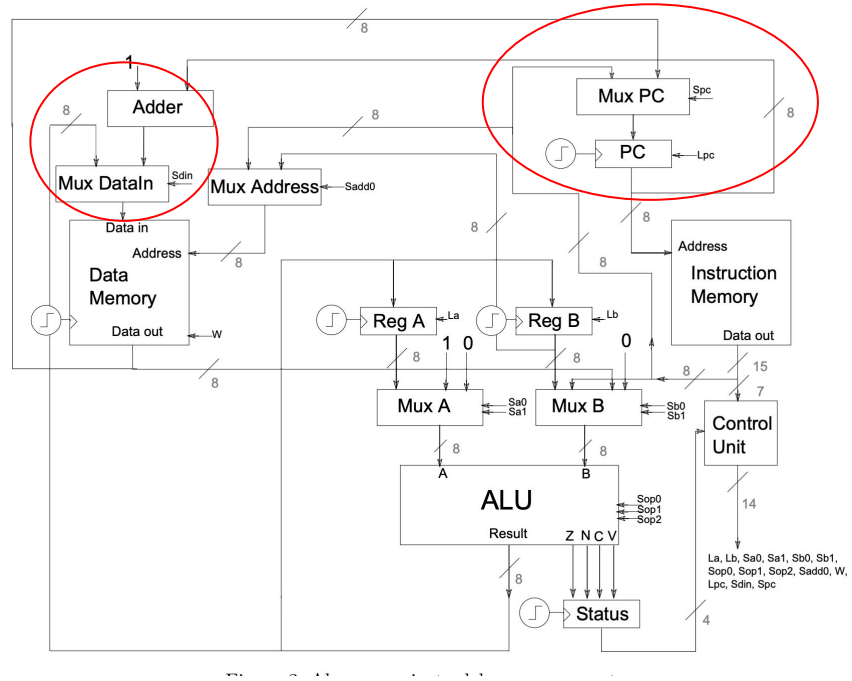
Subrutina: Requisitos

- Parámetro de Entrada
- Valor de retorno
- Llamada a la subrutina (salto de ida y de vuelta)

```
def encontrar_maximo(arreglo):  
    largo_maximo = len(arreglo)  
  
    maximo = arreglo[0]  
    i = 0  
  
    while i < largo_maximo:  
        if arreglo[i] > maximo:  
            maximo = arreglo[i]  
        i += 1  
  
    return maximo
```

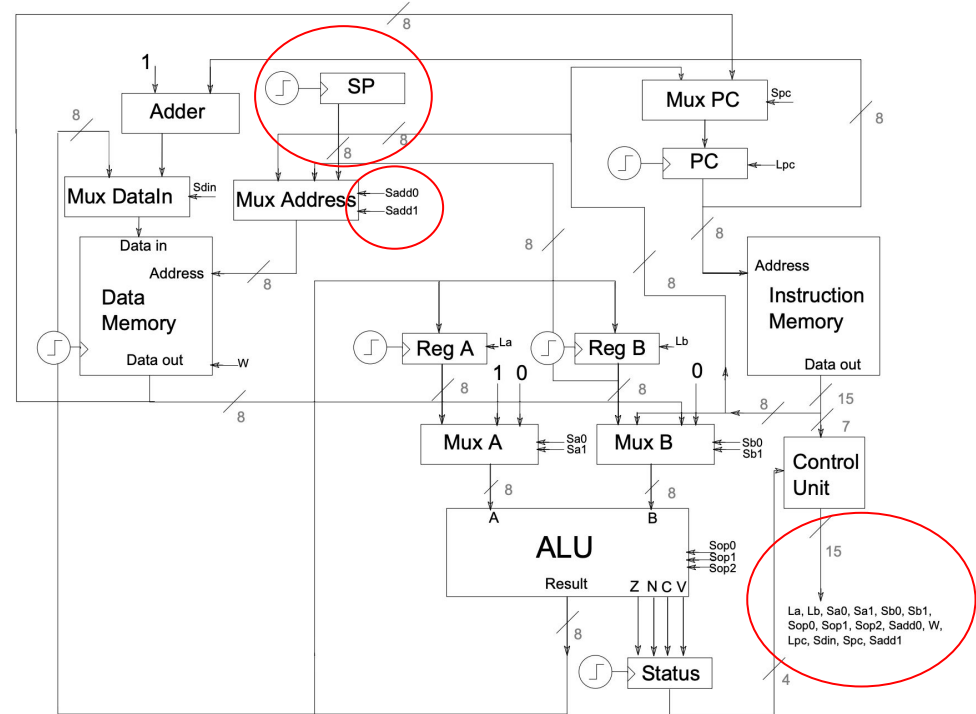
Subrutina: Implementación

- Agregamos un **Mux DataIn** a la entrada de la Memoria
- Agregamos un **Mux PC** a la entrada del Program Counter
- ¿Cómo nos aseguramos que no choquen las variables?



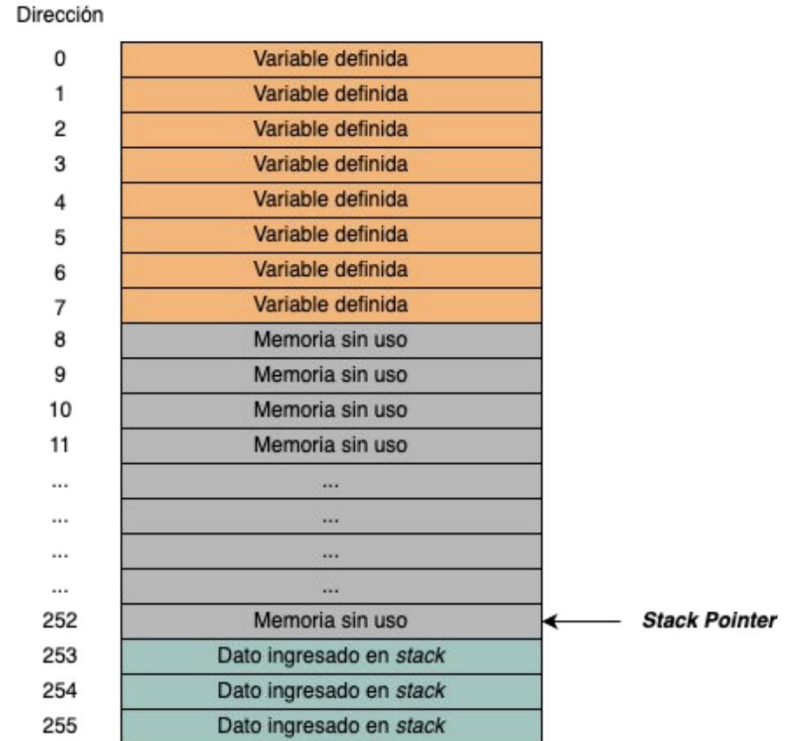
Subrutina: Implementación - SP

- Agregamos un nuevo registro **SP** a la entrada del Mux Address
- Su valor parte en 255 al ser de 8 bits (el valor más alto en base 10)



Subrutina: Memoria de Stack

- Utilizaremos el modelo de pila o en inglés Stack
- Si el *stack* choca con las variables definidas se produce un **stackoverflow**



Subrutina: Assembly - Resumen

Instrucción	Operandos	Operación	Condiciones	Ejemplo de uso
CALL	Dir	$\text{Mem}[\text{SP}] = \text{PC} + 1$, $\text{SP} - -$, $\text{PC} = \text{Dir}$		CALL func
RET		$\text{SP}++$ $\text{PC} = \text{Mem}[\text{SP}]$		- -
PUSH	A	$\text{Mem}[\text{SP}] = \text{A}$, $\text{SP} - -$		-
PUSH	B	$\text{Mem}[\text{SP}] = \text{B}$, $\text{SP} - -$		-
POP	A	$\text{SP}++$ $\text{A} = \text{Mem}[\text{SP}]$		- -
POP	B	$\text{SP}++$ $\text{B} = \text{Mem}[\text{SP}]$		- -

¿Dudas?

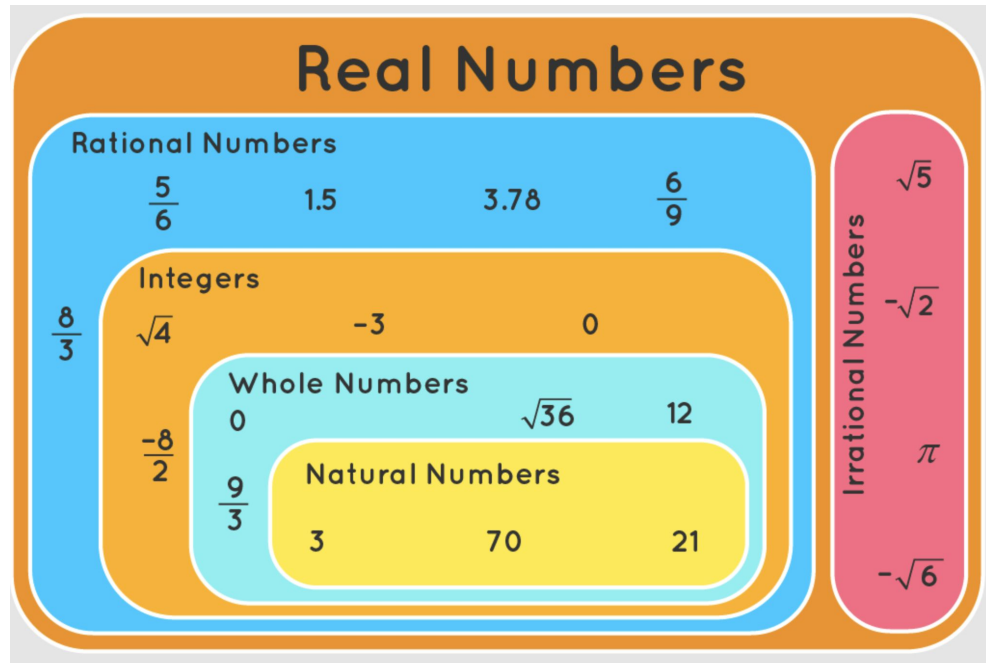
Introducción del curso:

- Un computador lo definimos como una **máquina programable que ejecuta programas.** ✓
- Para programar necesitamos: ?
 - Datos: números (enteros, **reales**) , texto, imágenes, etc ? ?
 - Operaciones: suma, resta, multiplicación, división, etc ✓
 - Variables: simples, arreglos ✓
 - Control de flujo: comparaciones, manejo de ciclos ✓
- Para poder tener números distinto a enteros necesitamos **¡punto flotante!**

¿Dudas?

Reales: Imposible

- Un computador tiene memoria y espacio **finito**
- Por lo mismo es **el imposible representar números irracionales**
- Sin embargo, los números **racionales** son posibles



Racionales: Sistema posicional decimal

- Un número racional puede ser representado por la **expansión con exponentes negativos**

$$987,65 = 9 \times 10^2 + 8 \times 10^1 + 7 \times 10^0 + 6 \times 10^{-1} + 5 \times 10^{-2}$$

Racionales: Sistema posicional binaria

- Un número racional puede ser representado por la **expansión con exponentes negativos**

$$101,01 = 1x2^2 + 0x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2} = 5,25$$

Racionales: División binaria

- Podemos obtener el valor en binario de un número racional en base decimal **obteniendo el valor de la división binaria**
- **Ejemplo (pizarra)**

$$0,75 = 3/4 = 11b \ / \ 110b$$

Racionales: División binaria

- Podemos obtener el valor en binario de un número racional en base decimal **obteniendo el valor de la división binaria**
- **Ejemplo (pizarra)**

$$0,1 = 1/10 = 1b / 1010b$$

¿Dudas?

Racionales: Periodicidad

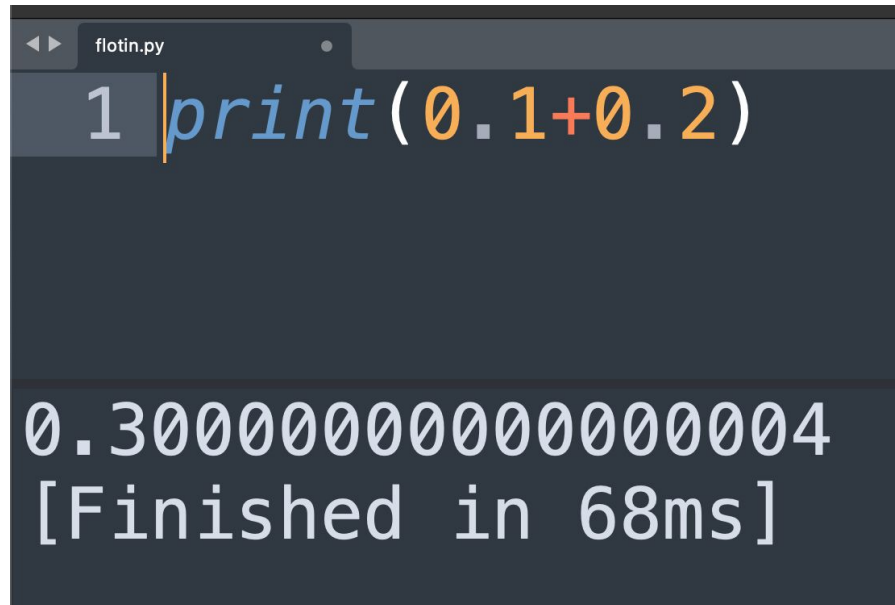
- Números racionales **periódicos** pueden ser finitos en **otra base**
- La base escogida puede afectar en la **finitud** del número racional que se busca representar

$$(0,3333...)_{10} = (1)_{10}/(3)_{10}$$

$$(1)_3/(3)_3 = (0,1)_3$$

Racionales: Error de redondeo de punto flotante

- Un computador no puede representar el valor exacto de 0.1
- Al operar con el decimal 0.1, no toda operación dará el valor esperado
- Esto es causado por la limitante física para representar números de periodicidad infinita en base binaria



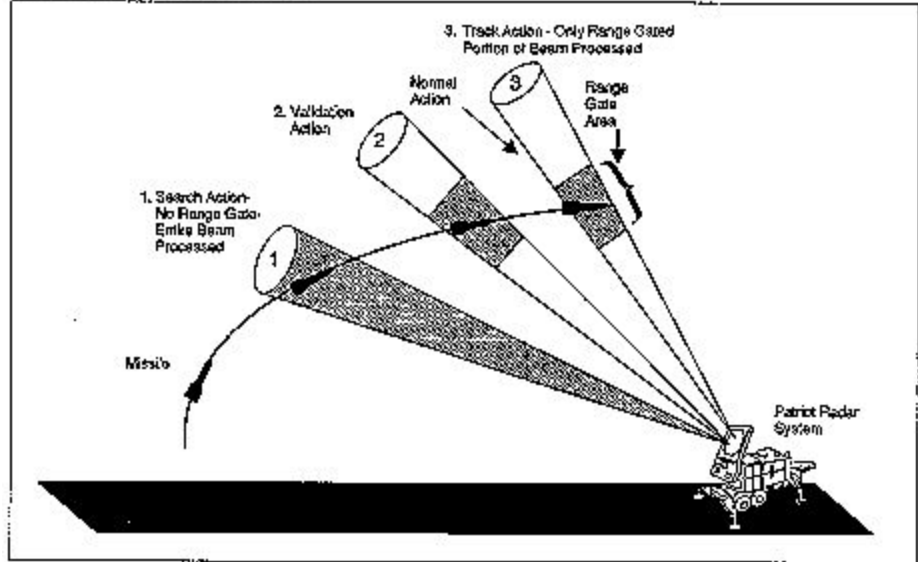
A screenshot of a terminal window titled 'flotin.py'. The first line shows the command `1 print(0.1+0.2)` with a cursor at the end. The second line shows the output `0.30000000000000004` followed by `[Finished in 68ms]` on the next line. The output demonstrates the floating-point rounding error where the sum of 0.1 and 0.2 is not exactly 0.3.

```
flotin.py
1 print(0.1+0.2)
0.30000000000000004
[Finished in 68ms]
```

Racionales: Casos reales - Misil Patriot (1991)

- Sistema que intercepta objetivos aéreos con misiles. **No interceptó un misil Scud** (Unión Soviética) por una falla en el seguimiento del objetivo
- Error fue ocasionado **por aproximación de un decimal finito** mediante un número binario infinito
- Debido al error, **28 personas murieron**

Figure 3: Correctly Calculated Range Gate



Racionales: Casos reales - Ariane 5 (1996)

- Un número de punto flotante de 64 bits **fue convertido incorrectamente** en un entero con signo de 16 bits, lo que provocó un desbordamiento y la destrucción del cohete, **costando millones de dólares**



Racionales: Casos reales - V8 (2011)

- Sigue siendo un tema en la **actualidad**
- El motor del navegador Google Chrome ha tenido vulnerabilidades relacionadas a las transformaciones de **punto flotante**

Google Chrome v8 Engine Floating Point Memory Corruption Vulnerability

Last Update Date: 28 Jan 2011 | Release Date: 2 Oct 2009 | 5088 Views

RISK: Medium Risk



A vulnerability has been identified in Google Chrome, which could be exploited by attackers to compromise a vulnerable system. This issue is caused by a memory corruption error in the v8 engine when parsing strings into floating point numbers via the "dtoa()" implementation, which could allow remote attackers to crash an affected browser or execute arbitrary code inside the sandbox.

Racionales: Punto fijo

- Se divide de forma **fija** en tres partes
 1. Signo
 2. Parte Entera (t)
 3. Parte Fractional (f)
- Simple y rápido de ocupar
- Rango **muy** pequeño

Ejemplo 8 bits:

10,111

0	010	1110
<hr/>		
signo	t	f

¿Dudas?

Racionales: Notación científica normalizada decimal

- Un número racional puede ser representado por la **una notación científica**

$$987,65 = 9,875 \times 10^{-2}$$

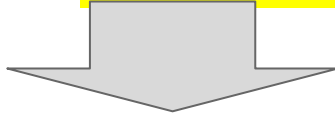


Mantiza o Significante

Racionales: Notación científica normalizada binaria

- Un número racional puede ser representado por la **una notación científica**

$$01111100b = 1.111100b \times 10^{110}$$



Mantiza o Significante

Racionales: Punto flotante

- Representación que permite “**mover**” la coma (flota)
- Se compone de **exponente** y **significante**
- **Pérdida de precisión**

Ejemplo 8 bits:

$10,111 \rightarrow 1,01 \text{ }^{001}$

0	101	0	001
<hr/>			
signo s	s	signo e	e

Racionales: Single Precision Floating Point - IEEE754

- Basada en notación científica **normalizada**
- Exponente **desfasado** en 127
- Significante **normalizado**
- Tiene un **gran** rango
- Tiene **valores reservados**

1 bit	8 bits	23 bits
signo	significante	exponente
		significante

Valores reservados:

+0=	00000000	000000000000000000000000
-0=	10000000	000000000000000000000000
+∞=	01111111	000000000000000000000000
-∞=	11111111	000000000000000000000000
NaN=	01111111	xxxxxxxxxxxxxxxxxxxxxxxxxxxx

Racionales: Single Precision Floating Point - IEEE754

- Un número de punto flotante, en estándar IEEE754, se ve así:

$$N = (-1)^{\text{signo}} \times 1.\text{significante} \times 10^{(\text{exponente} - 127)_b}$$

Ejemplo:

$$0.00101b = (1.01 * 10^{-11})b$$

$$\text{exponente} - 127 = -3 \Rightarrow \text{exponente} = 124 = (01111100)$$

$$\text{Según IEEE754: } 0.00101b = 001111100010000000000000000000$$

Racionales: Double Precision Floating Point - IEEE754

- Basada en notación científica
normalizada
- Exponente **desfasado** en 1023
- Significante **normalizado**
- Tiene un **gran** rango
- Tiene **valores reservados**

1 bit	11 bits	52 bits
signo	exponente	significante

¿Dudas?

Racionales: Alternativas...

- En caso de **no querer lidiar con floating point IEEE754** tiene distintas alternativas:
 - Decimales como números enteros
 - Punto flotante con base decimal y precisión fija
 - Punto flotante con base decimal y precisión arbitraria

```
flotin.py x
1 from decimal import Decimal
2
3 print(Decimal("0.1")+Decimal("0.2"))

0.3
[Finished in 94ms]
```

Racionales: Algoritmo de Suma

1. **Comparar los exponentes.** Hacer **shift right** a la mantisa del **número más pequeño** hasta que su exponente sea igual al del número más grande.
2. **Sumar** las mantisas.
3. **Normalizar la suma.** Esto puede ser haciendo shift right e incrementando el exponente, o shift left y decrementar el exponente. Depende del caso.
4. **Redondear el significativo** al número apropiado de bits. Revisar nuevamente si está normalizado, en caso que no lo esté, volver al paso anterior.

Ejercicio:

Sumar $A = 1,11 \times 10^{11}$ y $B = 1,11 \times 10^1$

Racionales: Algoritmo de Multiplicación

1. **Sumar los exponentes desfasados** y restar el desfase a la suma para obtener el nuevo exponente desfasado.
2. **Multiplicar** las mantisas.
3. **Normalizar el producto.** Esto puede ser haciendo shift right e incrementando el exponente, o shift left y decrementar el exponente. Depende del caso.
4. **Redondear el significativo** al número apropiado de bits. Revisar nuevamente si está normalizado, en caso que no lo esté, volver al paso anterior.
5. **Verificar el signo.** Si los signos de ambos números es el mismo, es positivo. Si los signos son distintos, el producto es negativo. Para asignar el nuevo signo debemos hacer un **XOR** de los signos originales.

Ejercicio:

Multiplicar $C = 1,11 \times 10^{11}$ y $D = 1,11 \times 10^1$

Introducción del curso: ✓

- Un computador lo definimos como una **máquina programable que ejecuta programas.** ✓
- Para programar necesitamos:
 - Datos: números (enteros, racionales) , texto, imágenes, etc ✓
 - Operaciones: suma, resta, multiplicación, división, etc ✓
 - Variables: simples, arreglos ✓
 - Control de flujo: comparaciones, manejo de ciclos ✓
- ¡COMPLETAMOS LA INTRODUCCIÓN! **Ahora comienza el curso...**

¿Dudas?

Clase 09 - Representaciones numéricas - Números Racionales

Profesor: **IIC2343 - Arquitectura de Computadores**
- Felipe Valenzuela González
Correo:
frvalenzuela@alumni.uc.cl