Informe 3

Taller V: Electrónica digital y microcontroladores

Profesor: Belarmino Segura Giraldo

Calculadora Digital

Presentado por:

Nicolás Cortés Parra

Jacobo Gutierrez Zuluaga

Sofía de los Ángeles Hoyos Restrepo

Universidad Nacional de Colombia sede Manizales

16 de agosto de 2023

Introducción: Utilizando Python se busca desarrollar un programa de calculadora digital que muestre una interfaz donde solicite al usuario dos (2) números: número A y número B que pueden estar en sistema numérico decimal, binario, octal o hexadecimal. Después de convertirlos a sistema binario se puede hacer una de las siguientes operaciones:

- Suma
- Resta
- Multiplicación
- División
- Potencia

Al obtener el resultado de la operación se muestra el valor en sistema binario además de otro sistema numérico de selección del usuario.

Librería para la interfaz: Tkinter

Tkinter es la librería estándar de Python para realizar interfaces gráficas de usuario (GUI: por sus siglas en inglés Graphical User Interface).

Código:

Iniciamos por importar las librerías y módulos:

- Tkinter: Interfaz. Se importa bajo el sobrenombre de tk.

  Se importa la herramienta de messagebox que permitirá que una ventana se muestre sobre la interfaz bajo ciertas condiciones.

- Módulos: Son archivos *.py* que se crearon independientemente para ahorrar líneas de código y tener un código más organizado. Contienen las funciones de las operaciones a realizar sobre los números binarios.

Módulo_suma:

Línea 3: Se crea la función **"Suma_binario"** que recibe los números A y B, que ya han sido convertidos a sistema binario.

Línea 5: La variable "*max_length*" compara la cantidad de caracteres de los números A y B y guarda el valor máximo.

Línea 7 y 8 : De acuerdo al anterior valor se rellenan los números con ceros hasta alcanzar dicho máximo.

Con esto se tiene el primer paso de la suma binaria:

1. Completar con ceros para igualar el número de bits de los números A y B.

Línea 13: Se crea un ciclo *for* que recorre el *string* de cada número mediante la función *range()* que va del último caracter *(max_length -1)*, a la posición *-1*, con pasos de *-1*. Es decir, se recorre de derecha a izquierda.

Línea 14: Se van sumando cada uno de los dígitos de los números teniendo en cuenta el acarreo, que para cuando *i=0*, *acarreo=0*. El resultado se guarda en la variable "*bit_sum*".

Línea 15 y 16: Con el método *.insert(),* se pone el resultado del módulo *(%)* de dos (2) de la variable "*bit_sum*" en la posición 0 de la lista que se guarda con el nombre de *"result"*. Luego, el acarreo toma el valor de la división entera por dos (2) del valor de "bit_sum", es decir, de la suma de los dos dígitos de cada número.

```
13      for i in range(max_length - 1, -1, -1):
14          bit_sum = int(bin1[i]) + int(bin2[i]) + acarreo
15          result.insert(0, str(bit_sum % 2))
16          acarreo = bit_sum // 2
17
18      if acarreo:
19          result.insert(0, str(acarreo))
20
21      resultado = ''.join(result)
```

Imagen 1. Módulo suma.

Si el acarreo es distinto a cero se insertará en la posición 0 de la lista "*result*". Luego, con la función *.join()* se unen los dígitos en un único *string*.

La función "*Suma_binario*", devuelve "resultado": la suma en binario de los números A y B.

Modulo_resta:

Línea 3: Se crea la función "*Resta_binario*" que recibe los números A y B, que ya han sido convertidos a sistema binario: "*bin1*" y "*bin2*".

Línea 5: De manera análoga a la suma, también se compara la longitud de los números para saber cuál número es mayor; el valor se guarda en la variable "*extensión*".

Línea 6 y 7: De acuerdo al anterior valor se rellenan los números con ceros hasta alcanzar dicho máximo.

Con esto se completa el primer paso de la resta binaria:

1. Igualar el número de bits de los números A y B (minuendo y sustraendo) con ceros.

Línea 13: Se crea un ciclo *if* para identificar cuál de los números es mayor.

En caso de que el primer número sea más pequeño, pasará a ser el sustraendo y el segundo número se convierte en el minuendo. Además, como en este caso el resultado va a ser negativo (número A menor al número B) se agrega el menos *(-)* que precede al resultado numérico.

```
13      if bin1 < bin2:  # cambio de ubicacion si el binario 1 es menor que el binario 2
14          condicion=1
15          bin1, bin2 = bin2, bin1
16          resultado = ["-",""]
17      if condicion==0:
18          resultado = [""]
```

Imagen 2. Módulo resta. Líneas 13 a 18.

Línea 21: Se crea un ciclo *for* que recorre cada dígito del número de derecha a izquierda, utilizando la función *range()*.

```
21      for i in range(extension - 1, -1, -1):
22          bit1 = int(bin1[i])
23          bit2 = int(bin2[i])
24
25          bit1 -= presta
26
27          if bit1 < bit2:
28              bit1 += 2
29              presta = 1
30          else:
31              presta = 0
32
33          resultado.insert(1, str(bit1 - bit2))
```

Imagen 3. Módulo resta. Ciclo for.

Línea 22 a 23: Las variables "*bit1*" y "*bit2*" almacenan un dígito de cada número; cuando *i=0*, se asignan los primeros dígitos de derecha a izquierda.

Línea 25: Después de sumar el *"bit1"* + *"presta*", que se inicializa en cero, se actualiza la variable *"bit1"*.

Línea 26: Se crea un ciclo *if* que compara los números *"bit1"* y *"bit2"*. Para cuando el número A sea menor se actualiza el valor de este al sumarle dos (2). La variable *"presta"* ahora toma el valor de uno (1). En caso contrario, las variables de los números quedan igual y la variable *"presta"* toma el valor de cero (0).

A través del método *.insert,* se agrega la resta del *"bit1"* y *"bit2"* en la posición uno (1). Cuando la resta sea negativa, en la posición cero (0) quedará el símbolo *"-"* que indica que el valor de la resta obtenido es menor a cero.

Línea 37. La función "Resta_binario" devuelve un único *string*, esto es posible a través del método *.join*: une todos los elementos de un iterable (cadena, lista, tupla) con un *string* específico.

Módulo Multiplicación:

La multiplicación tiene como base la suma. Por ello se inicia importando el "*Modulo_suma*" explicado anteriormente.

Línea 3: Se importa "*Modulo_Suma"*.

Línea 4: Se crea la función *"Multi_binario"* que recibe los números A y B en binario.

Línea 7 y 8: Se inicializan las variables *"resultado"= '0'* tipo *str* y, *x= 0* tipo *int*.

Línea 9: Se crea un ciclo *for* que recorre el número A en binario de derecha a izquierda utilizando la función *reversed()*. Esta función le da la vuelta al *string "bin1"*. Esto se hace para poder multiplicar de derecha a izquierda, como se hace en el método normal de multiplicación de decimales.

Dentro del *for* se crea un ciclo *if* que pregunta si el dígito de la derecha del número ingresado y convertido en binario es uno *(1)* o cero *(0)*. Para cuando es uno (1) se suma el *string* del número binario 2 con el *string '0'* por el número de desplazamientos realizados; cuando *bit = 0*, *x=0*. De esta manera se concatenan los *strings* y se guarda el resultado en la variable *x_des* que después se suma con la variable *"resultado"* utilizando la función de suma explicada anteriormente en el *"Modulo_suma"*.

```
9       for bit in reversed(bin1):
10          if bit == '1':
11              x_des = bin2 + '0' * x # Se crea un el binario desplazado hacia la izquierda.
12              resultado = Modulo_suma.Suma_binario(resultado, x_des) # Suma del resultado m
13          x += 1
14      return resultado
```

Imagen 4. Módulo multiplicación.

Al terminar el ciclo *if*, se actualiza la variable *x* sumándole un uno (1).

Luego, se repite el ciclo *for* hasta recorrer todos los caracteres del número binario.

Línea 14: La función *"Multi_binario"* devuelve la multiplicación en binario en la variable *"resultado"*.

Módulo Division:

Línea 2 y 3: Se importan los módulos de suma y resta.

Línea 5: Se crea una función llamada *"división"* que recibe el número A y B en binario. Se inicializan las variables *"result"*, *"temp"*, *"r"*. Luego se inicia un ciclo for que recorre la longitud del primer número. Se empieza un ciclo *if* donde se evalúa la condición de si el divisor es mayor a la variable *"temp"* y de acuerdo a ello actualiza *"result"* y *"temp"*.

Para el caso en que el divisor sea menor a *"temp"* (inicia en cero), se utiliza la resta binaria creada en el módulo aparte para restar *"temp"* y el divisor. Para cuando el resultado es cero (0), se toma el siguiente dígito del dividendo (*num1[i]*) y se almacena en la variable *"temp"*.

Cuando el resultado de la resta binaria es diferente a cero, el resultado *"r"* de la resta se mantiene pero quitando los primeros dígitos que tengan los caracteres cero ('0'), esto se realiza con la función *.lstrip('0'):* del string de la resta binaria se quitan los ceros '0' de izquierda a derecha.

```
10      for i in range(len(num1)):
11          if int(num2) > int(temp):
12              result += '0'
13              temp += num1[i]
14          else:
15              r = res.Resta_binario(temp, num2)
16              if r == 0:
17                  temp = num1[i]
18                  result += '1'
19              else:
20                  r = str(r).lstrip('0')
21                  result += '1'
22                  temp = r + num1[i]
```

Imagen 5. Módulo división. Ciclo for

Antes de finalizar el for se utiliza el módulo de suma binario entre el "result" y el string '1', siempre y cuando "temp" sea diferente a cero.

Línea 29: La función "division" devuelve el valor numérico en binario en la variable "result".

```
24      if temp !=int(0):
25          result = result + '0'
26
27      result=suma.Suma_binario(result,"1")
28
29      return result
30
```

Imagen 6. Módulo división.

Módulo Potenciación: Cuando se estudia la potenciación de decimales, se entiende que no es más que una multiplicación. La base se multiplica por sí misma cuantas veces indique el

exponente. Por ello, el módulo de potenciación se sirve del anterior módulo de multiplicación para hacer las operaciones de potenciación de números binarios.

Se comienza por importar el “Módulo multiplicación” bajo el nombre de “mult”

Línea 3: Se crea la función llamada “Pot_binario” que recibe los números A y B ya convertidos en formato binario.

Línea 5: Se crea un ciclo while que se ejecuta siempre y cuando el exponente sea diferente a cero. Esta línea permite que se cumpla la condición de que todo número elevado a cero dé uno (1).

```
3   def Pot_binario(base, exponente):
4       result = '1'
5       while exponente != '0':
6           if int(exponente[-1]) % 2 == 1:
7               result = mult.Multi_binario(result, base)
8           base = mult.Multi_binario(base, base)
9           exponente = Division_binaria_por_2(exponente)
10      return result
11
```

Imagen 7. Ciclo while en módulo potenciación.

Línea 7: Se actualiza la variable “result” con la multiplicación de la base por sí misma.

Línea 8 y 9: Dentro del ciclo while se ejecutan las funciones de división y multiplicación de los módulos creados.

Línea 10: El retorno de la función “Pot_binario ” se guarda en la variable result.

Volviendo al archivo principal de la calculadora:

Input 1:

Línea 22: Se crea la función “calcular()”

Línea 27 a línea 160: Se encuentran todas las condiciones para la entrada del número A.

Con el método.get() se lee el número A y el sistema de elección del menú presentado en la interfaz.

Línea 32 a 46: Se crea un ciclo if para evaluar el SND y el valor numérico ingresado.

Si el sistema seleccionado “sist_origen1” es “Binario”, hace que el contador sea igual a cero.

En un ciclo for que recorre el número A, se verifica si todos los dígitos son cero o uno, es decir, si pertenecen al sistema binario, de lo contrario, el contador se hace uno (1) y con la

herramienta de la librería tkinter “messagebox.showerror” (línea 43), se muestra una ventana pequeña que le indica al usuario que ha ingresado un número que no corresponde al sistema numérico seleccionado.

La siguiente línea (44) se encarga de borrar el número incorrecto ingresado y la interfaz queda lista para que el usuario ingrese otro número que sí corresponda con el sistema numérico o viceversa.

```
32      if sist_origen1 == 'Binario':           #Pregunta si el sistema del número A es binario
33          cont=0
34          for i in range(len(n_bin1)):
35              if n_bin1[i]=="0":
36                  cont+=0
37              elif n_bin1[i]=="1":
38                  cont+=0
39              else:
40                  cont+=1
41
42                  if cont==1:
43                      messagebox.showerror(message="La base del numero A ingresado es incorrecta
44                      nbin1_entry.delete(0,"end")
45
46          numero1 = str(n_bin1)           #Si el número es binario, lo muestra en binario tipo st
47
```

Imagen 8. Conversiones del número A a sistema binario.

La última línea (46) de esta parte pasa el número a tipo string. No se necesita de ninguna conversión extra, pues las operaciones a realizar se harán en el formato binario.

Línea 48 a 77: Dentro del mismo ciclo *if* anterior se pregunta si el sistema seleccionado en la interfaz es “Decimal”.

Para cuando la condición es *True* se asigna el valor de cero (0) al contador “cont” y se procede con el ciclo for que recorre la extensión del número, chequeando dígito por dígito, para asegurar que el número sí pertenece al sistema decimal. Lo anterior se hace a través de una secuencia de “elif” desde el número cero (0) al nueve (9), que asigna el valor cero (0) al “cont” cuando la condición se cumple.

En caso contrario, el contador se hace uno (1) y esto lleva a que la línea 74 utilizando el “message.boxshowerror”, muestre una ventana que le avisa al usuario que el SND y el número A no son correspondientes. Luego se limpia la casilla donde se ingresó el número incorrecto: (`nbin1_entry.delete(0,"end"))`

```
48  elif sist_origen1 == 'Decimal':                #Si el sistema del número A es decimal
49      cont=0
50      for i in range(len(n_bin1)):
51          if n_bin1[i]=="0":
52              cont+=0
53          elif n_bin1[i]=="1":
54              cont+=0
55          elif n_bin1[i]=="2":
56              cont+=0
57          elif n_bin1[i]=="3":
58              cont+=0
59          elif n_bin1[i]=="4":
60              cont+=0
61          elif n_bin1[i]=="5":
62              cont+=0
63          elif n_bin1[i]=="6":
64              cont+=0
65          elif n_bin1[i]=="7":
66              cont+=0
67          elif n_bin1[i]=="8":
68              cont+=0
69          elif n_bin1[i]=="9":
70              cont+=0
71          else:
72              cont+=1
73              if cont==1:
74                  messagebox.showerror(message="La base del numero A ingresado es incorrecta
75                  nbin1_entry.delete(0,"end")
```

Imagen 9. Condiciones para verificar que el número ingresado corresponde a decimal.

Línea 77: Es la instrucción que sigue dentro del ciclo for. Se convierte el número decimal a binario:

1. Pasa el número de formato string a int: `int(n_bin1)`.
2. Se convierte a binario con bin(): `(bin(int(n_bin1)`.
3. El binario obtenido se pasa a formato string: `str(bin(int(n_bin1)`.
4. Se eliminan los primeros dos caracteres del string obtenido: `str(bin(int(n_bin1)[2:]`. Esto se hace para evitar los prefijos por defecto "0b" de python para binarios. Más adelante se verá que se procede igual para los prefijos del sistema octal y hexadecimal.

```
77      numero1 = str(bin(int(n_bin1)))[2:] #Convierte el número A de decimal a binario
78
```

Imagen 10. Conversión de decimal a binario.

Línea 79 a 104: Se procede de igual manera a lo anterior pero para el sistema octal. Así que dentro del ciclo if, la secuencia de "elif" va desde el número cero (0) al siete (7). También se agrega un mensaje de error si el usuario ingresa incorrectamente el SND y el número. Se

convierte el número de octal a binario utilizando las funciones int() y bin(). El binario convertido se pasa a string y se elimina el prefijo “0o” (`[2:]`).

Línea 106 a 159: De la misma forma, se pregunta si el SND ingresado por el usuario es “Hexadecimal”. Con un ciclo for se recorre la extensión del número verificando que solo estén presentes los caracteres que pertenecen a dicho sistema, es decir desde el cero (0) al nueve (9) y desde la “A” hasta la “F”. Cabe destacar que se añadieron instrucciones para reconocer el sistema hexadecimal sin importar que el usuario ingrese letras en mayúscula o minúscula.

Línea 159: Se convierte el número hexadecimal a int, luego a binario, luego a string y por último se quita el prefijo “x0”.

Hasta aquí el primer bloque sobre el input del número A.

Input 2:

Línea 164 a 295: El procedimiento es el mismo del Input 1, con la única diferencia que ahora el número considerado es el B: “numero 2”.

Sistema binario, ciclo for, y conversión a string: Línea 164 a 182.

Sistema decimal, ciclo for, y conversiones: Línea 184 a 213.

Sistema octal, ciclo for, y conversiones: Línea 215 a 240.

Sistema hexadecimal, ciclo for y conversiones: Línea 242 a 295.

Operaciones:

En vista de que se crearon los módulos de operaciones independientes al programa principal de la calculadora, se pueden ahorrar muchas líneas de código en este bloque que ya están incluidas y explicadas en los módulos respectivos. Como se dijo al inicio, esto permite visualizar más fácilmente el código y las operaciones por aparte.

Suma, resta, multiplicación, división y potenciación:

Una vez finalizadas las conversiones entre diferentes SND incluídas en la primera parte del código: Input uno (1) e Input dos (2). Se da paso al bloque de operaciones: (línea 300 a 330) suma, resta, multiplicación, división, y potenciación.

Línea 300: Se lee el símbolo de la operación seleccionado por el usuario en la interfaz.

Línea 302 a 311: Se crea un ciclo if que de acuerdo al símbolo escogido almacena la operación en la variable “result”. En esta, se llama cada una de las funciones de los módulos explicados al inicio de este informe.

```
300     operacion = operacion_var.get()
301
302     if operacion=="+":
303         result = (suma.Suma_binario(numero1,numero2))
304     elif operacion=="-":
305         result = (resta.Resta_binario(numero1,numero2))
306     elif operacion=="*":
307         result = (mult.Multi_binario(numero1,numero2))
308     elif operacion=="/":
309         result = (div.binary_division(numero1,numero2))
310     elif operacion=="^":
311         result = (pot.Pot_binario(numero1,numero2))
```

Imagen 11. Secuencia de ciclo if para las operaciones.

Los argumentos son “numero1” y “numero2” que ya están convertidos a binario sin importar el SND original con el que fueron ingresados a la interfaz.

Línea 313: Se lee el SND de preferencia del usuario para visualizar el resultado de la operación.

Línea 315 a 330: Se crea otro ciclo for que de acuerdo a la línea anterior (313) hace las conversiones de binario a los demás sistemas numéricos.

Las líneas 327 y 328 cuentan la extensión del valor numérico final para ser visualizado en la interfaz: “Número de dígitos”.

La línea 330 finaliza la función “calcular()”. El retorno es el valor operado en binario y convertido al SND de selección.

En términos generales sobre la función “calcular()”, puede decirse:

- ★ Lee los números A y B ingresados a la interfaz.
- ★ Convierte los números A y B a sistema binario sin importar el sistema numérico con el que hayan sido introducidos (binario, decimal, octal o hexadecimal).
- ★ Lee la operación a realizar.
- ★ Opera los números en binario.
- ★ Lee el SND de visualización del resultado.
- ★ Retorna el valor de la operación entre el número A y B en binario y en otro SND de preferencia.
- ★ Retorna el número de dígitos del resultado, tanto en binario como en el otro SND de elección.

Conversión de número A y B a binario:

Se crean las funciones 'mostrar_bin1' y 'mostrar_bin2' que van a convertir los números A y B de cualquiera de los SND a binario. Luego se muestran en la interfaz.

Línea 336: Se crea la función 'mostrar_bin1' .

Línea 337 y 338: Se lee el número y sistema origen ingresado.

Línea 340 a 351: Se crea un ciclo if que de acuerdo al sistema ingresado convierte el número a binario. En la última línea con la herramienta .set() se retorna el número convertido a binario que se mostrará en la interfaz.

Línea 357 a 374: Se hace lo anterior pero para la función 'mostrar_bin2'. La función devuelve el número B convertido a binario para mostrarlo en la interfaz.

Interfaz:

La librería Tkinter ya se importó al inicio del código. Se llama en la línea 380 para empezar a desarrollar la interfaz con dichas herramientas.

Línea 381: .title: asigna el título a la ventana emergente de la interfaz.

Línea 382: .configure(bg=background), permite poner el color al fondo de la ventana.

Línea 383: .geometry, designa las dimensiones del cuadro de la interfaz.

A partir de la línea 385 se empiezan a crear los widgets donde se ingresarán las entradas del usuario, y se mostrarán las salidas del programa.

Línea 386 a 433: Se llevan los resultados del código a la interfaz a través de la herramienta tk.StringVar().

Para mostrar el texto se utiliza .Label y se especifican parámetros como: tipo de de fuente (letra), tamaño, estilo: cursiva, negrita, etc.

Para las casillas o cajas donde se muestra o ingresa el resultado se utiliza .Entry() y se fijan los colores del frente y fondo del recuadro, también su tamaño.

```
388   bin1_label = tk.Label(interfaz, text='Número A: ',font=("Lucida Bright","12","bold"))
389   nbin1_entry= tk.Entry(interfaz,bd=5,width=25)
390   bin2_label = tk.Label(interfaz, text='Número B: ',font=("Lucida Bright","12","bold"))
391   nbin2_entry = tk.Entry(interfaz,bd=5,width=25)
```

Imagen 12. Interfaz. Entradas y texto de salida.

Líneas 394, 396, 412, y 416: En el caso de los menú se utiliza el método .OptionMenu, que contiene los string alfabéticos que muestran los cuatro (4) SND.

```
393   sist_origen1_var = tk.StringVar(value='Decimal')
394   sist_origen1_menu = tk.OptionMenu(interfaz, sist_origen1_var, 'Binario', 'Octal', 'Decimal
395   sist_origen2_var = tk.StringVar(value='Decimal')
396   sist_origen2_menu = tk.OptionMenu(interfaz, sist_origen2_var, 'Binario', 'Octal', 'Decimal
```

Imagen 13. Interfaz. Creación de menú.

Líneas 398, 404, y 418: Creación de los botones convertir número A, convertir número B y calcular. Usando .Button se especifica el texto del botón y se le asigna el rol a través de alguna de las funciones desarrolladas en el código.

Para mostrar la conversión del número A y B a binario se usa la función 'mostrar_bin1' y 'mostrar_bin2'. En el caso de la operación y conversión final el command = ‘calcular’.

```
404   mostrar_bin2_button = tk.Button(interfaz, text='Convertir', command=mostrar_bin2)
```

```
418   calcular_button = tk.Button(interfaz, text='Calcular', command=calcular)
419   resultado_label = tk.Label(interfaz, text='El resultado es:',font=("Lucida Bright","12","b
```

Imagen 14 y 15. Interfaz. Creación de botones.

Líneas 437 a 469: Se asignan las coordenadas de posición a cada una de las variables a mostrar en la interfaz.

Con ayuda de la herramienta .grid() se fija la fila (row), columna (column), la distancia de separación en pixeles en el eje x y y: padx y pady.

```
437   bin1_label.grid(row=0, column=0, padx=5, pady=25)
438   nbin1_entry.grid(row=0, column=1, padx=5, pady=5)
439   sist_origen1_menu.grid(row=0, column=3, padx=0, pady=5)
440
441   bin2_label.grid(row=2, column=0, padx=5, pady=25)
442   nbin2_entry.grid(row=2, column=1, padx=5, pady=5)
443   sist_origen2_menu.grid(row=2, column=3, padx=0, pady=5)
```

Imagen 16. Interfaz. Coordenadas de la ubicación de los widgets.

El código finaliza con la línea 471 con la instrucción: interfaz.mainloop(), es necesaria para que aparezca en pantalla la interfaz programada.

Referencias:

- Librería Tkinter. What is tkinter used for and how to install it. Tomado de: https://www.activestate.com/resources/quick-reads/what-is-tkinter-used-for-and-how-to-install-it/

- Métodos de string de python explicados con ejemplos. Método .join(). Tomado de: https://www.freecodecamp.org/espanol/news/metodos-de-string-de-python-explicados-con-ejemplo/
- Función reversed en python. Tomado de: []https://keepcoding.io/blog/funcion-reversed-en-python/
- Función .lstrip(). Tomado de: https://www.scaler.com/topics/python-lstrip/
- Herramientas padx y pady, .grid() y .Label() . Layout management in Tkinter. Tomado de: https://zetcode.com/tkinter/layout/#:~:text=The%20padx%20and%20the%20pady,borders%20of%20the%20root%20window.