

Laboratorio 1: Arquitectura y Organización de Computadores

Profesores: Mauricio Solar y Viktor Tapia
Ayudantes de cátedra: Javier Rojas y Mauricio Cortés
Ayudante de Tareas: Joaquín Montes y Benjamín López

18 de Marzo 2023

1 Reglas Generales

Para la siguiente tarea se debe realizar un código programado en **Python**. Se exigirá que los archivos se presenten de la forma más limpia y legible posible. Deberá incluir un archivo **README** con las instrucciones de uso y ejecución de su programa junto a cualquier indicación que sea necesaria.

2 Tarea

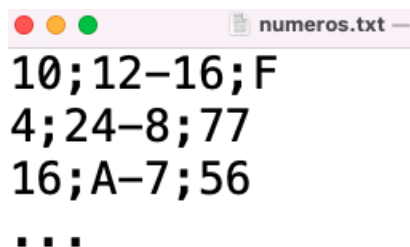
El problema y los requerimientos de la solución se presentan a continuación.

2.1 Formato Archivo

El archivo de texto **numeros.txt** contiene información respecto a números en distintas bases numéricas siguiendo el siguiente formato: **base1;numero1-base2;numero2** en donde:

- **base1**: Es la base numérica, entre 2 y 16, del numero1.
- **numero1**: Es un número **posiblemente** representado en base1.
- **base2**: Es la base numérica, entre 2 y 16, del numero2.
- **numero2**: Es un número **posiblemente** representado en base2.

Los números en el archivo **no tienen representación negativa** y además, **pueden no ser representados en la base numérica que se indica**. Considere el siguiente ejemplo:



```
10;12-16;F
4;24-8;77
16;A-7;56
...
```

Ejemplo 1: Archivo numeros.txt. Los 3 puntos significan que hay más registros.

2.2 Requerimientos

Se solicita desarrollar un programa que permita:

- Ingresar un número por pantalla. El valor indicará el **tamaño del registro**. El rango permitido es entre 1 y 32.
- Por cada número ingresado se debe recorrer el archivo numeros.txt y calcular los siguientes valores:
 - Total números en el archivo. (A)
 - Total números con error en la representación numérica. (B) `is_valid()`
 - Total números que no pueden ser representados con el valor ingresado. (C) `fits_reg()`
 - Total sumas realizadas en complemento dos que provocan overflow en registros con el tamaño ingresado por el usuario. (D) `has_overflow()`
- Por cada número ingresado se debe guardar el resultado obtenido en un archivo de texto de nombre **resultados.txt** con el formato `A;B;C;D`.
- Si el tamaño del registro es mayor a la representación de los números del archivo se debe realizar una extensión de signo. 5 [base 10] es 101 [base 2] => Registro de 5 bits => 101 tiene 3 bits => Se toma 101 como número ya representado en C2, por lo tanto su extensión de signo a 5 bits es: 11101
- Ante cualquier error encontrado en los datos no se realiza el intento de sumar en C2. 101 [base 2] representado en C2 es -3 [base 10]
- Cuando el usuario ingresa 0, se debe recorrer el archivo resultados.txt y validar si la cantidad de errores (representación, tamaño y overflow) es mayor que la cantidad de números en el archivo. Si esto es así, el programa finaliza, en caso contrario, sigue solicitando valores.

2.3 Cálculos

(*)

Conversion con 4 registros:

4 = 0100 NOT4 = 1011 NOT4 + 1 = 1100
1 = 0001 NOT1 = 1110 NOT1 + 1 = 1111

Operación: $-4 - 1 = -5$

$-4 - 1 = (-4) + (-1)$
 $= (\text{NOT}4 + 1) + (\text{NOT}1 + 1)$
 $= (1100) + (1111)$
 $= 11011$

El último dígito no va porque tenemos 4 registros.

1011 => C2 => -5 (Está correcto, overflow descartado)

Considerando los datos presentes en el archivo numeros.txt del **ejemplo 1:**

- 12 en base 10 es 1100 en binario. $\rightarrow C2 \Rightarrow -4$
- F en base 16 es 1111 en binario. $\rightarrow C2 \Rightarrow -1$
- ✗ 24 en base 4 no es representable ya que el 4 no es parte de los dígitos permitidos.
- ! 77 en base 8 es 63 en base 10 y por ende 111111 en binario.
- A en base 16 es 1010 en binario.
- ! 56 en base 7 es 41 en base 10 y por ende 101001 en binario.

Con las conversiones anteriores analicemos los resultados a obtener según lo que ingrese el usuario:

- **Acción 1:** El usuario ingresa 4 por pantalla. (4 registros disponibles)
 - ✓ Total números en el archivo = 6. ✓
 - ✗ Total números con error en la representación numérica = 1. 24 en base 4.
 - ! Total números que no pueden ser representados con el valor ingresado = 2. Con 4 bits de tamaño no se pueden representar el 77 en base 8 y el 56 en base 7.
 - ✦ Total sumas realizadas en complemento dos que provocan overflow en registros con el tamaño ingresado por el usuario = 0. La primera fila es la única que puede sumar, en este caso no se produce overflow (validar). (*)
- Resultado a escribir en el archivo **resultados.txt**: 6;1;2;0.

- **Acción 2:** El usuario ingresa 7 por pantalla. (7 registros disponibles)
 - Total números en el archivo = 6.
 - Total números con error en la representación numérica = 1. 24 en base 4.
 - Total números que no pueden ser representados con el valor ingresado = 0. Con 7 bits de tamaño se pueden representar todos.
 - Total sumas realizadas en complemento dos que provocan overflow en registros con el tamaño ingresado por el usuario = 0. La fila 1 y fila 3 pueden sumar en C2, pero no producen overflow (validar).
- Resultado a escribir en el archivo **resultados.txt**: 6;1;0;0.
- **Acción 3:** El usuario ingresa 0 por pantalla.
 - Total números en el archivo = 6.
 - Total errores hasta este punto = $1+2+1 = 4$. Esta sumando los errores que hubieron entre Accion 1 y Accion 2. (Hay que tener registro de los errores totales)
 - 6 mayor que 4 por lo que el programa no termina.
- **Acción 4:** El usuario ingresa 3 por pantalla.
 - Total números en el archivo = 6.
 - Total números con error en la representación numérica = 1. 24 en base 4.
 - Total números que no pueden ser representados con el valor ingresado = 5. Ninguno de los números puede ser representado con tamaño 3.
 - Total sumas realizadas en complemento dos que provocan overflow en registros con el tamaño ingresado por el usuario = 0. No se realizan sumas.
- Resultado a escribir en el archivo **resultados.txt**: 6;1;5;0.
- **Acción 5:** El usuario ingresa 0 por pantalla.
 - Total números en el archivo = 6.
 - Total errores hasta este punto = $1+2+1+5 +1 = 10$.
 - 6 menor que 9 por lo que el programa termina.

2.4 Consideraciones

- Se entregarán algunos archivos de prueba para validar su desarrollo, pero a la hora de revisar se cambiará el total de archivos y el contenido de estos, por lo que su programa debería seguir funcionando de la misma manera.
- La cantidad de filas en los archivos no se conoce.
- **Importante:** No se pueden utilizar funciones que ya existen en Python relacionadas con conversiones numéricas.

3 Presentación Aleatoria (Anti chat GPT)

Para cada tarea, se seleccionarán grupos al azar para presentar su tarea frente a ayudantes y eventualmente profesores, recibiendo una ponderación del 70% y 30% entre tarea y presentación respectivamente. Si su grupo presentó en una tarea, no volverá a salir nuevamente.

4 README

Debe contener como mínimo:

- Nombre, Rol y Paralelo de los integrantes.
- Especificación de los algoritmos y desarrollo realizado.
- Supuestos utilizados.

5 Consideraciones Generales

- Se deberá trabajar de a pares. Se deberá entregar en Aula a mas tardar el día 01 de Abril de 2023 a las 23:59 horas. Se descontarán 5 puntos por cada hora o fracción de atraso. Las copias serán evaluadas con nota 0 en el promedio de las tareas.
- La tarea debe ser hecha en Python. Se asume que usted sabe programar en este lenguaje, ha tenido vivencias con él, o que aprende con rapidez.
- Pueden crear todas las funciones auxiliares que deseen, siempre y cuando estén debidamente comentadas.
- La entrega considera un único archivo de nombre `convierte.py` junto con el README. Los archivos deberán ser comprimidos y enviados juntos en un archivo `.tar.gz` en el formato **LAB1_ROL1_ROL2**.
- Las preguntas deben ser hechas por Aula. De esta forma los demás grupos pueden beneficiarse en base a la pregunta, **se responderán consultas hasta 48 hrs. antes de la fecha y hora de entrega.**
- Si no se entrega README, o si su programa no funciona, la nota es 0 hasta la corrección.
- Se descontarán 50 puntos por:
 - No respetar el formato de entrega.