Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores (II/2017)

Enunciado Tarea 01

Tipos de datos y señales, 60 % de la entrega 01

1. Motivación

Como ya han visto a la hora de trabajar en un computador existen muchos tipos de datos diversos, la idea de esta tarea es que se logren familiarizar con estos y logren entender como se representan.

2. Descripción

La siguiente tarea tendrá 3 partes, dos de carácter obligatorio y uno de bonus. En la primera parte trabajarán con pares de números, los que deben ser capaces de convertir de un tipo de dato a otro. La segunda parte es un bonus; el objetivo es trabajar con pares de números de tipo float y transformarlo a diferentes estándares de punto flotante. Finalmente, la tercera parte consiste en desplegar diferentes señales de status según el resultado de una operación aritmética básica.

2.1. Parte 1: Tipos de datos 40 %

En esta parte deberán crear una función **en python** capáz de transformar números en cualquier base a base 10 y restringiéndolo a un tipo de dato en especifico. Para esto su función deberá recibir 3 argumentos: El numero a transformar, la base en la que se encuentra el numero y el tipo de dato a cual lo queremos transformar.

Los tipos de datos permitidos son:

TIPO DE DATOS	SE ESCRIBE	MEMORIA REQUERIDA*	RANGO ORIENTATIVO*
Entero	int	2 bytes	- 32768 a 32767
Entero largo	long	4 bytes	- 2147483648 a 2147483647
Decimal simple	float	4 bytes	$-3{,}4\cdot10^{38}$ a $3{,}4\cdot10^{38}$
Decimal doble	double	8 bytes	$-1{,}79\cdot10^{308}$ a $1{,}79\cdot10^{308}$
Carácter	char	1 bytes	0 a 255

Cabe destacar que los diferentes tipos de datos manejan diferentes cantidades de bits y por lo tanto su programa debe ser capaz de extraer la cantidad de bits máxima por tipo si es que el numero entregado tiene una representación binaria mayor a lo esperado para ese tipo. Por ejemplo:

```
Transformar el numero 342 en base 10 a char:

      El 342 en base 10 se escribe en binario como: 0000000101000100
      Un char soporta hasta 8 bits: 00000001[01000100]
      Por lo tanto el numero resultante es 01000100 = 68 en base 10
```

Para esta parte, el input vendrá en el siguiente formato:

```
Formato:
    numero-base-tipo_de_dato
Ejemplo:
    342-10-char
    12312312313-4-int
```

Observaciones

- Los datos en concreto a analizar son int_16 como int, int_32 como long, float_32 estándar IEEE 754 para la primera parte, char_8
- Notar que los números en float pierden presición por lo cual existen numeros no representables en este formato como 16777303 cuyo valor más cercano es 16777304
- Por si al investigar sobre datos se dan cuenta que en algunas partes sale int como 2 o 4 bytes es porque en lenguajes como **C** el dato de tipo int depende de la arquitectura del computador.
- Las bases de los números no superará al 10, es decir, trabajaremos con números de bases entre 2 y 10.

2.2. BONUS: Manejo de float en diferentes estandares 10 %

Para tener Bonus, su programa además de implementar el estandar pedido, debe poder variar la cantidad de bits de exponente y de mantisa, el input de esta sección va a ser de la forma "número-exponente-mantisa" y el output esperado es el numero representado como un float con esa configuración. Obs: numero puede ser cualquiera de los datos que deben ser soportados en esta tarea. Por ejemplo: 253-9-2 significa que el numero 253 debe intentar ser representado con un float que tengo 9 bits de exponentes y 2 bits de mantisa. Luego indicar que numero en base 10 representa ese float.

2.3. Parte 2: Identificación de señales 40 %

En esta parte de la tarea deberán simular una serie de operaciones entre dos registros de 16 bits y desplegar en pantalla las seales que están activa y desactivadas. Las señales son:

- Señal P: Se activa cuando el ultimo bit del resultado es 1.
- Señal N: Se activa cada vez que se realiza una resta de la forma A - B y A $<$ B.
- Señal Z: Se activa cuando el resultado de la operación es 0.
- Señal C: Se activa cuando sumas y el resultado es menor a los sumandos.

Las operaciones soportadas son:

- Suma: Representada por el símbolo "+" y requiere de dos nḿeros para sumar ambos. Ejemplo: 2 + 7.
- Resta: Representada por el símbolo "-" y requiere de dos nḿeros para restar ambos. Ejemplo: 2 + 7.
- AND: Representada por el símbolo "&&" y requiere de dos nḿeros para aplicar la siguiente tabla de verdad entre sus bit. Ejemplo: 2 && 7.

AND	1	0
1	1	0
0	0	0

- Or: Representada por el simbolo "||" y requiere de dos nḿeros para aplicar la siguiente tabla de verdad entre cada bit de ambos. Ejemplo: 2 || 7.

OR	1	0
1	1	1
0	1	0

- Xor: Representada por el simbolo "∧" y requiere de dos nḿeros para sumar ambos. Ejemplo: 2 + 7.

XOR	1	0
1	0	1
0	1	0

- Negación: Consiste en negar cada bit del nḿero, es decir, los 1 se transforman en 0 y los 0 en 1. Representada por el símbolo "¬" y requiere de un nḿero para aplicar la negació. Ejemplo: ¬5.
- Rotate Left: Consiste en desplazar a la izquierda en una posición cada bit del numero. En caso de bit mas significativo (el que está mas a la izquierda), este bit pasa a ser el bit de la posición 0 del numero. Representada por el símbolo "<<" y requiere de un número para aplicar rotate. Ejemplo: << 0.
- Rotate right: Consiste en desplazar a la derecha en una posición cada bit del numero. En caso de bit mas significativo (el que está mas a la izquierda), este bit pasa a ser el bit de la posición 0 del numero. Representada por el símbolo ">>" y requiere de un nḿero para aplicar rotate. Ejemplo: >> 0.
- Shift Left: Consiste en desplazar a la izquerda en una posición cada bit del numero. A diferencia del rotate, el ultimo bit se pierde mientras que el primero pasa a ser 0 automáticamente. Representada por el símbolo "<" y requiere de un nḿero para aplicar rotate. Ejemplo: < 0.
- Shift Right: Consiste en desplazar a la derecha en una posición cada bit del numero. A diferencia del rotate, el ultimo bit se pierde mientras que el primero pasa a ser 0 automáticamente. Representada por el símbolo ">" y requiere de un nḿero para aplicar rotate. Ejemplo: > 0.

El input será numero operación numero con espacio entre cada numero y la operacion. En caso de que sea una operación de un solo símbolo, será símbolo numero con espacio entre el símbolo y el núnero El resultado debe estar con el siguiente formato: P-C-Z-N

2.4. Informe 20 %

Además del programa en Python ustedes deben entregar un informe que contenga un análizis sobre que representan las distintas señales usadas en la parte 2 y en caso de haber hecho el bonus agregar un apartado describiendo que algoritmo utilizaron. Además deberán responder las siguientes preguntas o solicitudes.

- Para cada tipo de dato: De un ejemplo de aplicación en la cual sea mejor ese tipo por sobre los otros. Justifique su respuesta
- Qué pasa si priorizamos exponente sobre mantisa y viceversa?

3. Metodología

Recibirá un archivo con con diferentes input para cada parte y su programa deberá generar otro archivo con los resultados de cada uno en este nuevo archivo. En caso de no hacer el bonus, cuando empiece esa seccion DEBE dejar un NULL en ese espacio. Ejemplo de archivo:

```
test.txt                          Resultados.txt
 Parte 1 - 2                       Parte 1:
 342-10-char                       X
 12312312313-4-int                 y
 Parte 2 - 2                       Parte 2:
 4 + 7                             1-0-0-1
 -2 || 2                           1-1-0-0
 Bonus - 2                         Bonus:
 231-5-9                           NULL
 764-8-2                           NULL
```

El numero al lado de cada sección representa la cantidad de lineas de esta

4. Restricciones

- Cualquier librería extra que se use debe ser previamente consultada con los ayudantes.
- No se pueden usar las funciones bin() ni int() de python.

5. Contacto

- Francesca Lucchini: flucchini@uc.cl
- Felipe Pezoa: fipezoa@uc.cl
- Hernán Valdivieso: hfvaldivieso@uc.cl
- Luis Leiva: lileiva@uc.cl

6. Integridad académica

Los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile deben mantener un comportamiento acorde a la Declaración de Principios de la Universidad. En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Docencia de la Escuela de Ingeniería.

Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno, sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno copia un trabajo, obtendrá nota final 1,1 en el curso y se solicitará a la Dirección de Docencia de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona.

Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente.

Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile. Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.