**Paradigmas de Programación**

TP Funcional 2018: Microprocesador – Primera entrega

Profesor: Federico Scarpa

Curso: K2003

Alumnos: Agustin Priano, Ezequiel Laime, Marcos Leonel Bazán, Marcos Inca, Rodrigo Diaz

TP Funcional 2018 – Microprocesador – Primera entrega Paradigmas de programación

# INTRODUCCIÓN:

En el siguiente informe desarrollaremos las consignas brindadas por el docente a partir de la unificación y aplicación de criterios vistos en clase basados en el paradigma funcional (aplicación parcial, orden superior, pattern matching, guardas, composición de funciones, etc.) y en el lenguaje de programación *Haskell*.

Las distintas pruebas fueron implementadas en ghci y para la demostración de funcionamiento se emplearán capturas de pantalla para cada punto dado. Al final del informe se realizaron ciertos casos de prueba que también estarán acompañados de capturas de pantalla de la consola para verificar el correcto funcionamiento (las funciones utilizadas en estas pruebas no son especificadas ahí ya que son las mismas que modelamos a lo largo del desarrollo del trabajo práctico).

TP Funcional 2018 – Microprocesador – Primera entrega Paradigmas de programación

# DESARROLLO:

1. **Entrega 1**

## Punto 1: Modelar micro

* + 1. data Microprocesador = Microprocesador { memoriaDeDatos :: [Int], acumuladores :: (Int, Int), programCounter :: Int, etiquetaError :: [Char]

}deriving Show

Criterio utilizado:

Se modeló el microprocesador según las características especificadas:

* + - * Posiciones de memoria: lista de Int en la que cada posicion de la lista representa una posición de la memoria
      * Acumuladores (A y B): Dupla en la que la posición uno representa el acumulador A y la posición dos el acumulador B.
      * Program Counter (PC): Int que guarda la cantidad de operaciones que realiza el micro
      * Etiquetas de error: lista de char que en Haskell representa un string y que permitirá informar un mensaje de error.
    1. xt8088 = Microprocesador [ ] (0,0) 0 "”

xt8088 = Microprocesador [] (0,0) 0 ""

fp20 = Microprocesador [] (7,24) 0 ""

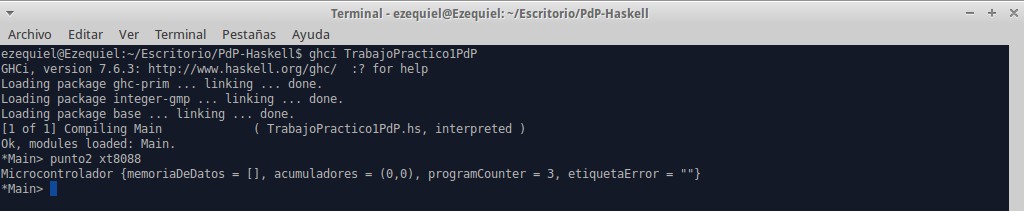
## Punto 2

* + 1. nop microprocesador=microprocesador {

programCounter = ( programCounter microprocesador ) + 1

}

* + 1. Desde la consola, modele un programa que haga avanzar tres posiciones el program counter.

punto2 xt8088 = (nop.nop.nop) xt8088 Se aplica el concepto de composición de funciones.

TP Funcional 2018 – Microprocesador – Primera entrega Paradigmas de programación

## Punto 3

* + 1. lodv val microprocesador = microprocesador { programCounter = ( programCounter microprocesador ) + 1, acumuladores=(val, (snd.acumuladores) microprocesador)

}

swap microprocesador = microprocesador {

programCounter = ( programCounter microprocesador ) + 1, acumuladores = ((snd.acumuladores) microprocesador, (fst.acumuladores)

microprocesador )

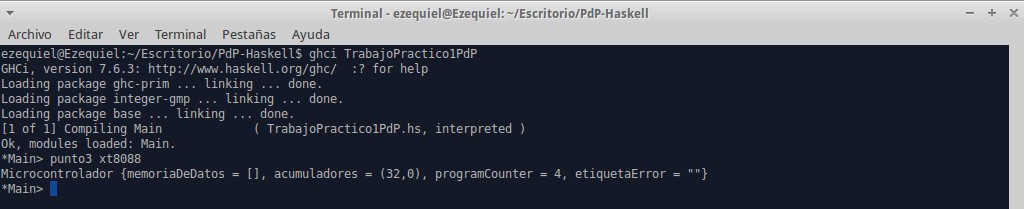
}

add microprocesador=microprocesador {

programCounter = ( programCounter microprocesador ) + 1,

acumuladores = ( (fst.acumuladores) microprocesador + (snd.acumuladores) microprocesador, (snd.acumuladores) microprocesador )

}

* + 1. Implementar el siguiente programa, que permite sumar 10 + 22 punto3 xt8088 = (add.(lodv 22).swap.(lodv 10)) xt8088

## Punto 4

TP Funcional 2018 – Microprocesador – Primera entrega Paradigmas de programación

* + 1. Modelar la instrucción DIV, STR y LOD.

str addr val (Microprocesador memoriaDeDatos acumuladores programCounter etiquetaError)

|esMayorALongitud (addr-1) memoriaDeDatos=Microprocesador memoriaDeDatos acumuladores (programCounter+1) "Tamanio mayor longitud de cadena"

|otherwise=Microprocesador (insertarElementoEnPosicion val memoriaDeDatos addr) acumuladores (programCounter+1) etiquetaError

insertElementoEnPosicion x [] 1 = x:[] insertElementoEnPosicion x ys 1 = x:(tail ys)

insertElementoEnPosicion x (y:ys) n = y:insertElementoEnPosicion x ys (n-1)

esMayorALongitud posicion lista = (posicion > (length lista))

lod addr (Microprocesador [] (a,b) programCounter etiquetaError) = Microprocesador [] (0,b) (programCounter+1) etiquetaError

lod addr (Microprocesador memoriaDeDatos acumuladores programCounter etiquetaError)

|(esMayorALongitud addr memoriaDeDatos)=Microprocesador memoriaDeDatos acumuladores (programCounter+1) "Tamanio mayor longitud de cadena"

|otherwise=Microprocesador memoriaDeDatos (memoriaDeDatos!!(addr-1), (snd acumuladores)) (programCounter+1) etiquetaError

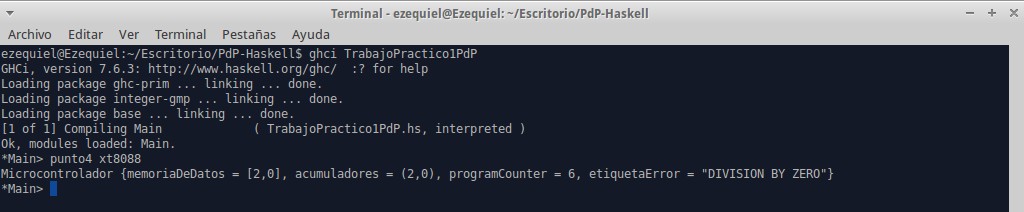
divide (Microprocesador memoriaDeDatos (a,0) programCounter etiquetaError) = Microprocesador memoriaDeDatos (a,0) (programCounter+1) "Division by zero"

divide microprocesador = microprocesador {

acumuladores = ( (fst.acumuladores) microprocesador `div` ( snd.acumuladores ) microprocesador, 0 ),

programCounter = (programCounter microprocesador ) + 1

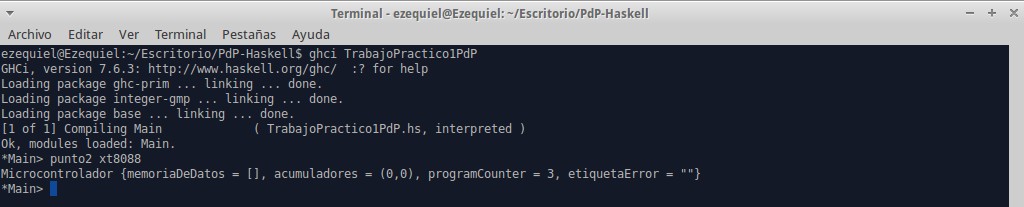
}

* + 1. Desde la consola, modele un programa que intente dividir 2 por 0. punto4 xt8088 = (division.(lod 1).swap.(lod 2).(str 2 0).(str 1 2)) xt8088

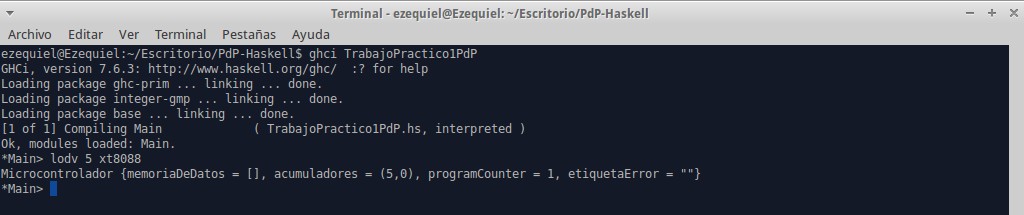
TP Funcional 2018 – Microprocesador – Primera entrega Paradigmas de programación

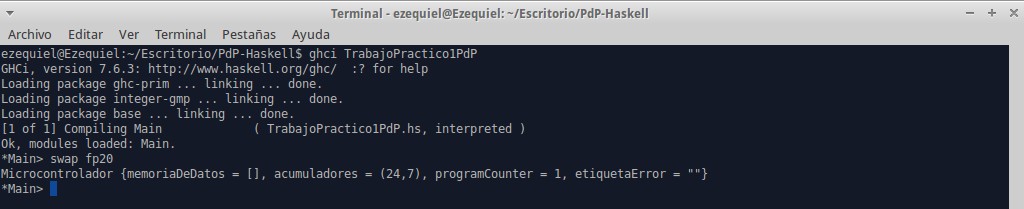
# Casos de prueba

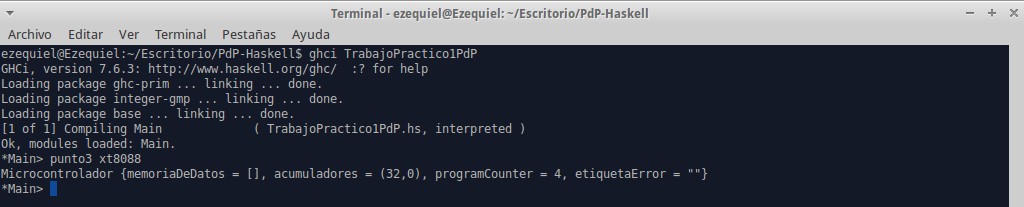
## Punto 2



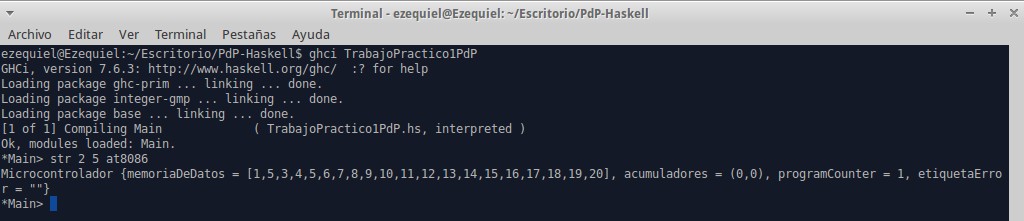
## Punto 3





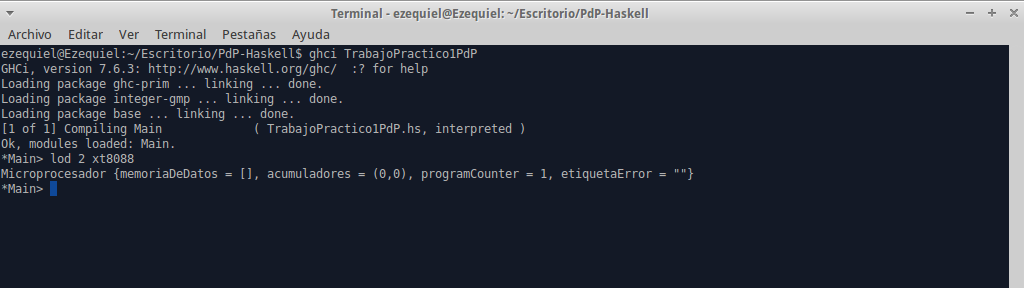


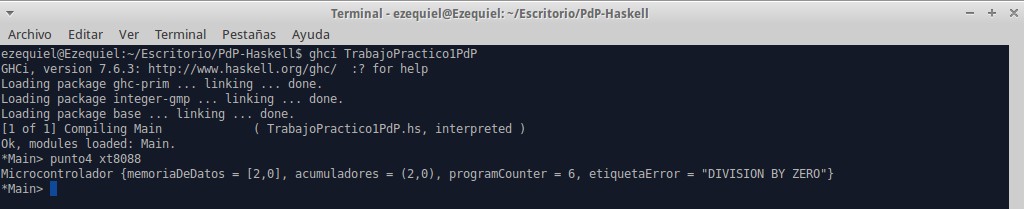
## Punto 4



TP Funcional 2018 – Microprocesador – Primera entrega Paradigmas de programación









