Microcontrolador

Indice

1 El dominio

Un microcontrolador es una computadora hecha en uno o varios chips. Suelen utilizarse en la industria para controlar máquinas, herramientas, robots, teléfonos celulares, etc.

Un fabricante de microcontroladores le solicita a Ud. que haga un simulador de uno de sus modelos de microcontroladores, el cual consta de:

- Una gran cantidad de posiciones que conforman la memoria de datos. Consideramos que la memoria comienza a partir de la posición 1.
- Dos acumuladores que contienen valores enteros, cada uno identificados como A y B.
- Un program counter (PC) que comienza con el valor cero y se incrementa cada vez que el microcontrolador ejecuta una instrucción.
- Una etiqueta con el último mensaje de error producido

El fabricante nos pasó la lista de instrucciones mínimas que debe soportar:

Mnemotécnico	**Descripción**
NOP	*No operation*, el programa sigue en la próxima instrucción.
ADD	Suma los valores de los dos acumuladores, el resultado queda en el acumulador A, el acumulador B debe quedar en 0
DIV	Divide el valor del acumulador A por el valor del acumulador B, el resultado queda en el acumulador A, el acumulador B debe quedar en 0
SWAP	Intercambia los valores de los acumuladores (el del A va al B y viceversa)
LOD addr	Carga el acumulador A con el contenido de la memoria de datos en la posición addr
STR addr val	Guarda el valor *val* en la posición addr de la memoria de datos
LODV val	Carga en el acumulador A el valor *val*

2 Temas a evaluar

- Modelado de información
- Composición
- Aplicación parcial
- Ecuaciones por guardas

3 Entrega 1

3.1 Punto 1: Modelar micro

1. Modelar el tipo de dato microprocesador. Justificar el criterio utilizado.
    a. Modelar un procesador XT 8088, cuyos acumuladores están en cero, el program counter en cero, sin etiquetas de error y la memoria vacía. Nombrarlo xt8088.

3.2 Punto 2

1. Desarrollar la instrucción NOP, utilizando la abstracción que crea conveniente.
2. Desde la consola, modele un programa que haga avanzar tres posiciones el program counter.

```
NOP
NOP
NOP
```

¿Qué concepto interviene para lograr este punto?

3.3 Punto 3

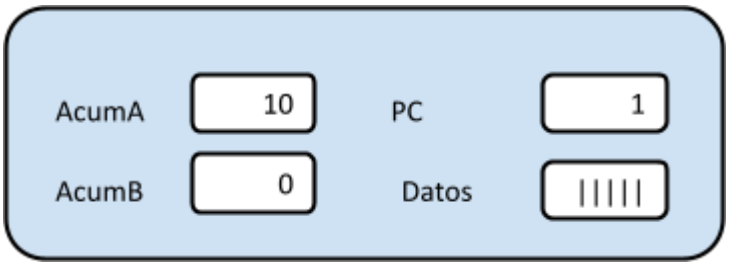
1. Modelar las instrucciones LODV, SWAP y ADD.
2. Implementar el siguiente programa, que permite sumar 10 + 22

```
LODV 10   // Cargo el valor 10 en el acumulador A
SWAP      // Cargo el valor 10 en el acumulador B (paso de A a B)
LODV 22   // Cargo el valor 22 en el acumulador A
ADD       // Realizo la suma y el resultado queda en el acumulador A
```

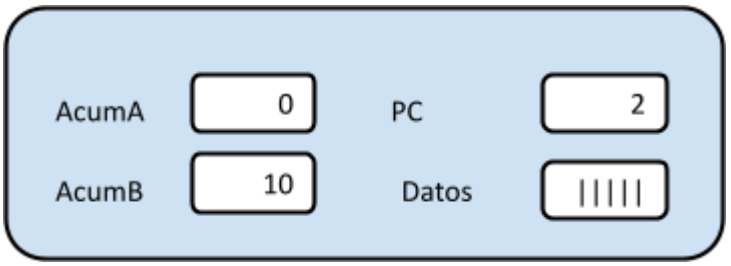
Debe procurar no repetir el código para aumentar el program counter.

Veamos los estados intermedios y final que tiene el microcontrolador para hacer la suma:

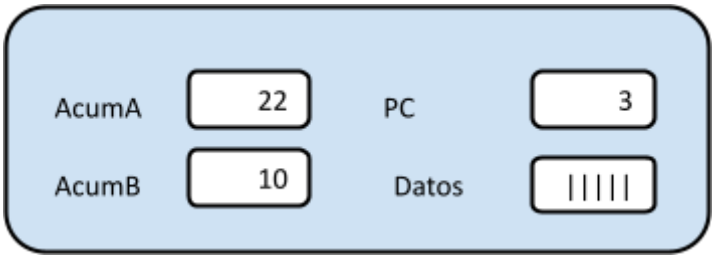
```
LODV 10
```



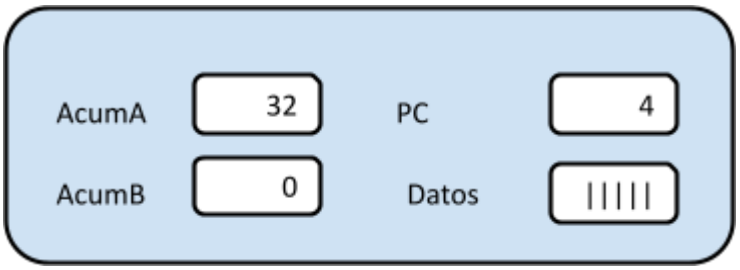
SWAP



LODV 22



ADD



Es importante encontrar una abstracción para el programa como también testear el estado final del microprocesador luego de ejecutar las instrucciones.

3.4 Punto 4

1. Modelar la instrucción DIV[1], STR y LOD.
2. Desde la consola, modele un programa que intente dividir 2 por 0.

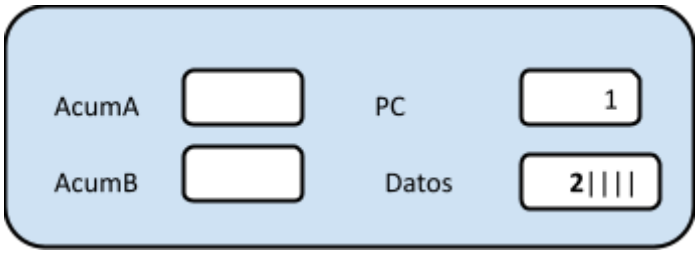
[1] Dado que ya existe una función predefinida en Haskell llamada div (división entera), recomendamos nombrarla “divide”

```
STR  1 2  // Guardo en la posición 1 de memoria el valor 2
STR  2 0  // Guardo en la posición 2 de memoria el valor 0
LOD  2    // Cargo en el acumulador A el valor 0 (pos.2)
SWAP      // Guardo el valor 0 en el acumulador B
LOD  1    // Cargo en el acumulador A el valor 2 (pos.1)
DIV       // Intento hacer la división
```

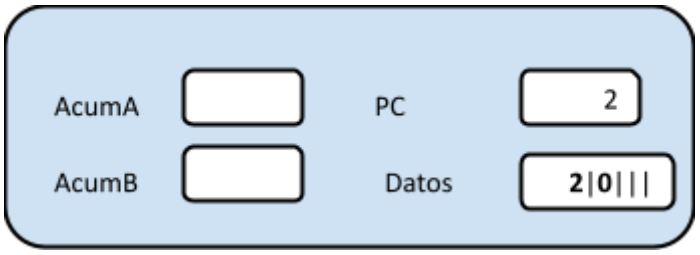
El microprocesador debe tener en la etiqueta de error el mensaje “DIVISION BY ZERO” y el Program Counter debe quedar en 6 (el índice de la instrucción donde ocurrió el error).

La secuencia de estados intermedios se describe a continuación:

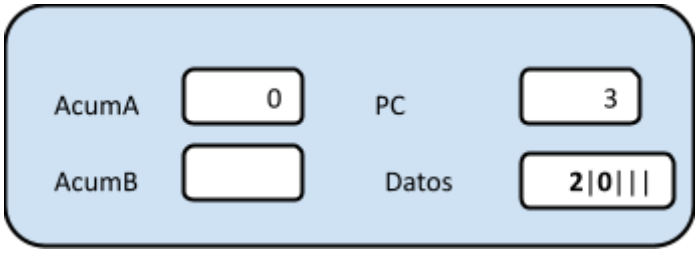
`STR 1 2`



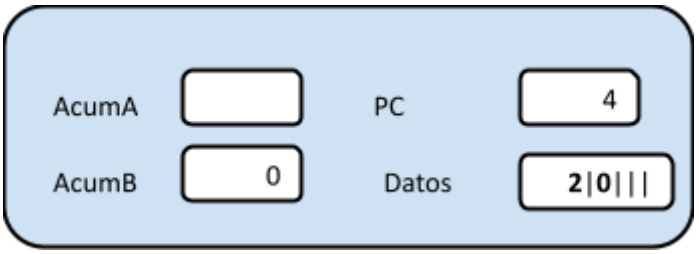
`STR 2 0`



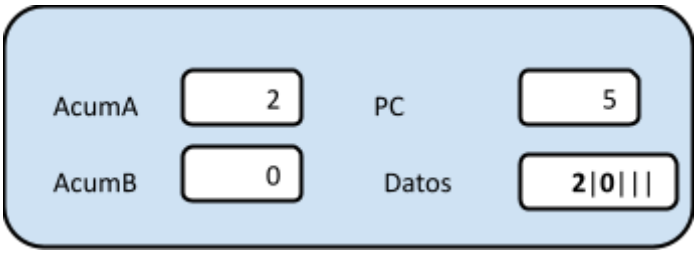
`LOD 2`



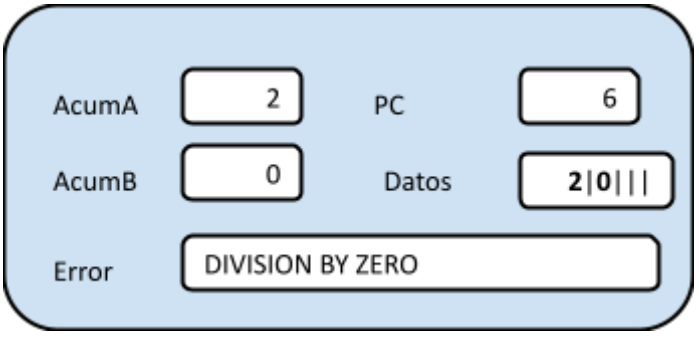
`SWAP`




```
LOD 1
```




```
DIV
```



4 Casos de prueba

4.1 Punto 2

- Luego de avanzar el procesador xt8088 tres veces, se espera que el program counter quede en 3. Los acumuladores deben quedar en cero, con la memoria vacía y sin etiqueta de errores.

4.2 Punto 3

- LODV 5 tiene
    - como precondiciones: el acumulador A y B están en cero
    - como post-condiciones: el acumulador A tiene valor 5 y el B cero.
- Dado un procesador fp20 que tiene acumulador A con 7 y acumulador B con 24, al ejecutar SWAP el acumulador A debe quedar con 24 y el B con 7.
- Luego de ejecutar el programa que suma 10 + 22, el acumulador A debe quedar en 32 y el B en 0.

4.3 Punto 4

- Dado el procesador at8086 que tiene los acumuladores en cero, el program counter en 0, sin mensaje de error y una memoria con los siguientes datos: [1..20], le ejecutamos la instrucción STR 2 5. Entonces el procesador at8086 debe quedar con un 5 en la posición 2: [1, 5, 3, 4, 5,...]
- LOD 2 de un procesador xt8088 con la memoria vacía (1024 posiciones con valores cero[2]) debe dejar con cero el acumulador A (cero = ausencia de información)
- Ejecutar por consola la división 2 por 0 para el procesador xt8088 según el programa escrito arriba, esperamos el mensaje de error “DIVISION BY ZERO”, y un 6 en el program counter.
- Ejecutar la división de 12 por 4 para el procesador xt8088 (cambiando los valores del programa anterior), que debe dar 3 y no tirar ningún mensaje de error.

[2] Nota: puede ser útil la función replicate

5 Cambios al dominio

Al fabricante de los microcontroladores le gustó nuestra propuesta, así que nos solicitó algunos cambios.

Esto involucra una serie de requerimientos que explicaremos en el punto 7.3.

6 Temas a evaluar

- Orden superior
- Modelado de información
- Refactor | Modificaciones a un código existente
- Recursividad

7 Entrega 2

7.1 Punto 1: Carga de un programa

Queremos tener la posibilidad de “cargar” un programa en el microcontrolador, que debe estar separado del área de memoria que se reserva para datos de usuario.

> **Nota:** Si aparece un error “no instance for Show” al modificar la abstracción que modela el Microprocesador, te recomendamos que leas este artículo: http://wiki.uqbar.org/wiki/articles/no-hay-instancias-para-el-show.html

Represente estos dos programas de la entrega anterior:

- la suma de 10 y 22

```
LODV 10   // Cargo el valor 10 en el acumulador A
SWAP      // Cargo el valor 10 en el acumulador B (paso de A a B)
LODV 22   // Cargo el valor 22 en el acumulador A
ADD       // Realizo la suma y el resultado queda en el acumulador A
```

- y la división de 2 por 0

```
STR  1 2  // Guardo en la posición 1 de memoria el valor 2
STR  2 0  // Guardo en la posición 2 de memoria el valor 0
LOD  2    // Cargo en el acumulador A el valor 0 (pos.2)
SWAP      // Guardo el valor 0 en el acumulador B
LOD  1    // Cargo en el acumulador A el valor 2 (pos.1)
DIV       // Intento hacer la división
```

7.2 Punto 2: Ejecución de un programa

Ejecutar un programa que esté cargado en el microcontrolador. Esto consiste en

- avanzar el program counter
- ejecutar la instrucción
    - cuando una instrucción salga con error el programa no debe ejecutar ninguna otra instrucción posterior (ni avanzar el program counter de aquí en más)

En la entrega anterior es posible que en cada instrucción hayas avanzado el program counter, es una buena oportunidad para refactorizar[3] esto.

7.3 Punto 3: IFNZ

Modelar la instrucción múltiple IFNZ, que ejecutará una serie de instrucciones en caso de que el acumulador A no tenga el valor 0.

Por ejemplo, si tenemos IFNZ de las siguientes instrucciones:

```
LODV 3    // Cargo el valor 3 en el acumulador A
SWAP      // Invierto los acumuladores
```

Y lo ejecutamos en el microprocesador fp20, en el acumulador A debe quedar 24, y en el B el valor 3. No nos interesa cómo queda el program counter ya que el usuario todavía no lo definió.

En caso de error en alguna de las instrucciones, el resto no debe ejecutarse.

7.4 Punto 4: Depuración de un programa

Queremos depurar un programa, que consiste en eliminar todas las instrucciones innecesarias, es decir, las que luego de ejecutarse en un micro (el microprocesador *xt8088* es utilizado como caso testigo), dejan en cero el acumulador A, el B y todas las celdas de memoria (de datos).

Por ejemplo, si queremos depurar este programa:

```
SWAP
NOP
LODV 133
LODV 0
```

[3] Refactorizar es la acción de mejorar el código. Podés ver más sobre eso acá o esperar a cursar Diseño de Sistemas

```
STR 1 3
STR 2 0
```

Debería quedarnos estas instrucciones:

```
LODV 133   // La suma de acumuladores y memoria da 133
STR 1 3    // La suma de acumuladores y memoria da 3
```

7.5 Punto 5: Memoria ordenada

Saber si la memoria de un micro está ordenada, esto ocurre cuando las celdas de memoria contienen valores menores o iguales que las subsiguientes. El procesador at8088 y el xt8088 tienen la memoria ordenada, uno que contenga en la memoria los valores [2, 5, 1, 0, ...] no.

7.6 Punto 6: Memoria infinita

Modelar

- un procesador que tenga “memoria infinita” inicializada en cero.
- ¿Qué sucede al querer cargar y ejecutar el programa que suma 10 y 22 en el procesador con memoria infinita?
- ¿Y si queremos saber si la memoria está ordenada (punto anterior)?
- Relacione lo que pasa con el concepto y justifique.

8 Casos de prueba

8.1 Modificaciones a los casos de prueba de la entrega 1

Se elimina la necesidad de testear el program counter aislado de cada instrucción.

8.2 Pruebas de los programas

- Al cargar y luego ejecutar el programa que suma 10 + 22 en el microprocesador xt8088,
  - el acumulador A debe quedar en 32
  - el acumulador B debe quedar en 0
  - el program counter debe quedar en 4
- Al cargar el programa que divide 2 por 0 en el microprocesador xt8088,
  - el acumulador A debe quedar en 2
  - el acumulador B debe quedar en 0
  - el mensaje de error debe decir “DIVISION BY ZERO”
  - el program counter debe quedar en 6

- y los primeros dos elementos de la memoria de datos deben ser 2 y 0

8.3 Pruebas sobre IFNZ

- Al ejecutar la instrucción IFNZ de las instrucciones LODV 3 y SWAP sobre el microprocesador fp20, que tiene inicialmente 7 en el acumulador A y 24 en el acumulador B
  - el acumulador A debe quedar en 24
  - el acumulador B debe quedar en 3
- Al ejecutar la instrucción IFNZ de las instrucciones LODV 3 y SWAP sobre el microprocesador xt8088
  - el acumulador A debe continuar en 0
  - el acumulador B debe continuar en 0

8.4 Depuración de un programa

- Al depurar este programa

```
SWAP
NOP
LODV 133
LODV 0
STR 1 3
STR 2 0
```

- Deben quedar dos instrucciones
  - La primera instrucción debe ser LODV 133
  - La segunda instrucción debe ser STR 1 3
  - En resumen debe haber 2 instrucciones en el nuevo programa

8.5 “Orden” de la memoria

- La memoria del microprocesador at8086 está ordenada
- La memoria del microprocesador microDesorden no lo está, porque tiene los acumuladores A y B en 0, un programa vacío, el program counter en 0, no tiene mensaje de error y la memoria de datos tiene los valores 2, 5, 1, 0, 6, y 9.