

Microcontrolador

Entrega 1



Indice

[1 El dominio](#)

[2 Temas a evaluar](#)

[3 Entrega 1](#)

[3.1 Punto 1: Modelar micro](#)

[3.2 Punto 2](#)

[3.3 Punto 3](#)

[3.4 Punto 4](#)

[4 Casos de prueba](#)

[4.1 Punto 2](#)

[4.2 Punto 3](#)

[4.3 Punto 4](#)

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	<i>No operation</i> , 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 <i>val</i> en la posición addr de la memoria de datos
LODV val	Carga en el acumulador A el valor <i>val</i>

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

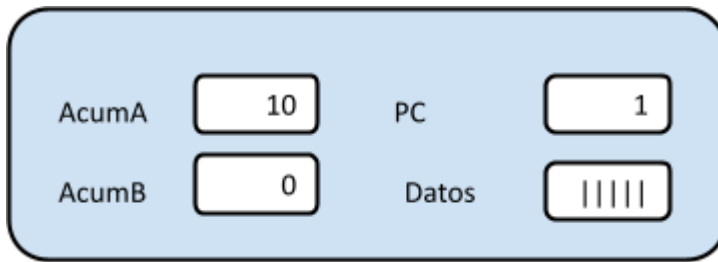
1. Modelar las instrucciones LODV, SWAP y ADD.
2. Implementar el siguiente programa, que permite sumar $10 + 22$

```
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
```

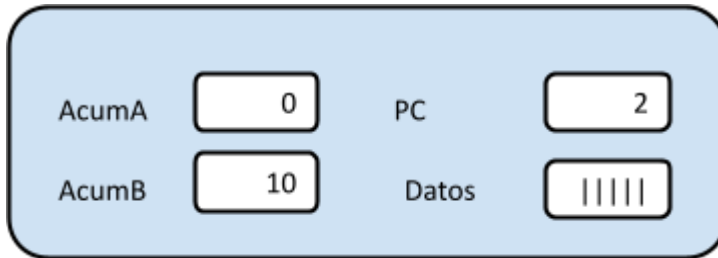
Debe procurar no repetir el código para aumentar el program counter.

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

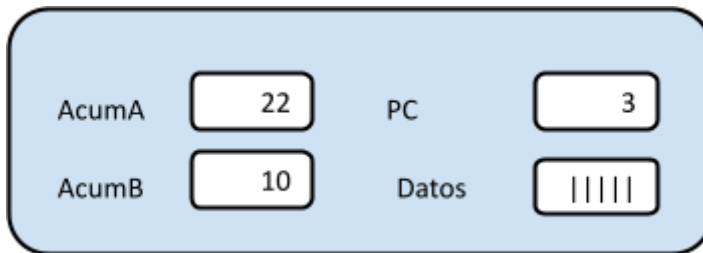
```
LODV 10
```



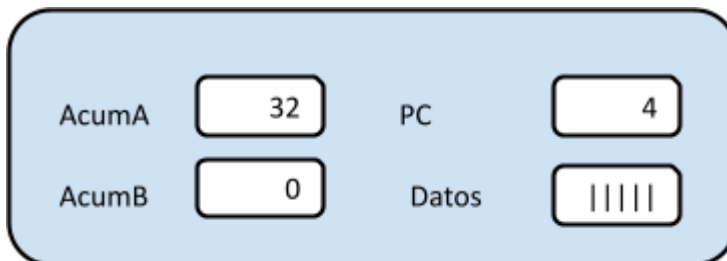
SWAP



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 DIV¹, STR y LOD.
2. Desde la consola, modele un programa que intente dividir 2 por 0.

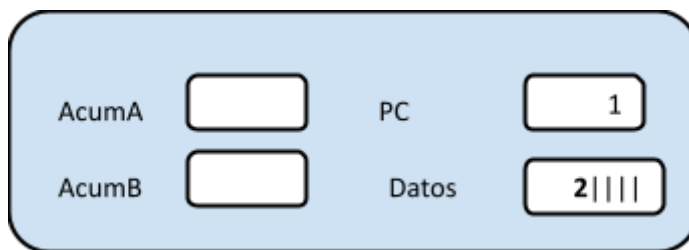
¹ Dado que ya existe una función predefinida en Haskell llamada div (división entera), recomendamos nombrarla "divide"

```
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
```

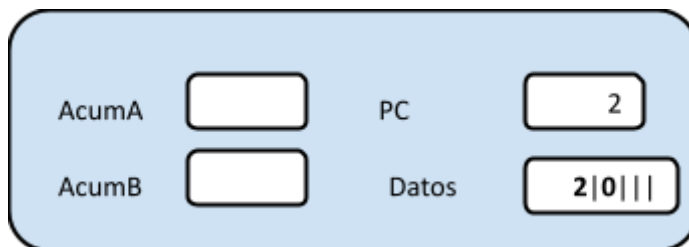
El microprocesador debe tener en la etiqueta de error el mensaje "DIVISION BY ZERO" y el Program Counter debe quedar en 6 (el índice de la instrucción donde ocurrió el error).

La secuencia de estados intermedios se describe a continuación:

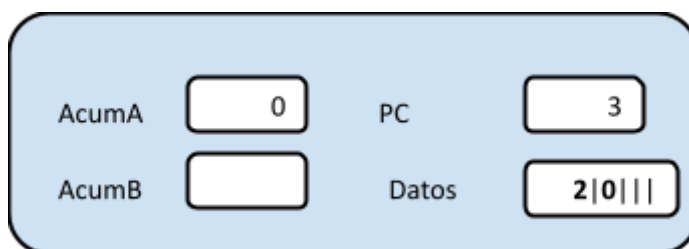
STR 1 2



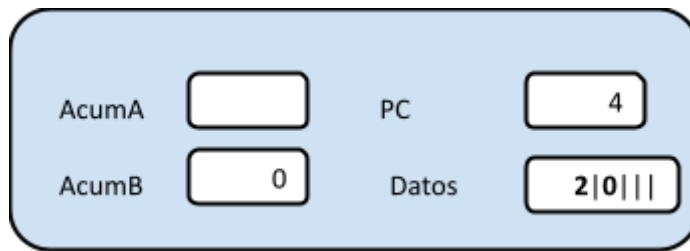
STR 2 0



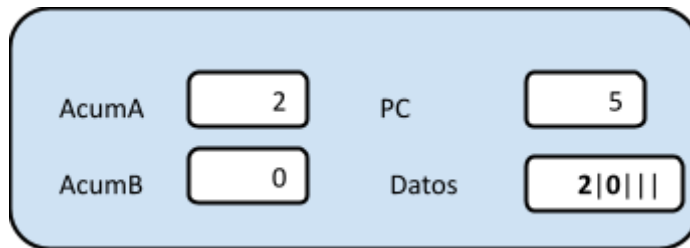
LOD 2



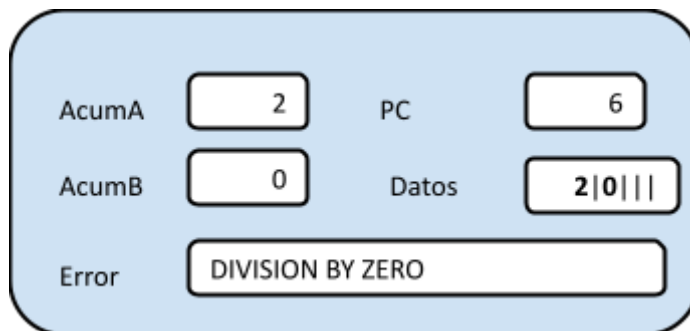
SWAP



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 cero²) 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