Microcontrolador
Entrega 2

Indice

1 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 3.

2 Temas a evaluar

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

3 Entrega 2

3.1 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
```

3.2 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 esto.

3.3 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.

3.4 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
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
```

3.5 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.

3.6 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.

4 Casos de prueba

4.1 Modificaciones a los casos de prueba de la entrega 1

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

4.2 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

4.3 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

4.4 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

4.5 “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.