

Taller de Interrupciones

Organización del Computador 1

Primer Cuatrimestre 2019 - Turno Tarde

El presente taller consiste en extender una micro-arquitectura diseñada sobre el simulador *Logisim* con el fin de soportar interrupciones. Se buscará implementar un controlador de interrupciones y codificar un programa que haga uso de este módulo.

El simulador se puede bajar desde la página <http://www.cburch.com/logisim/> o de los repositorios de Ubuntu. Requiere Java 1.5 o superior. Para ejecutarlo, teclear en una consola `java -jar logisim.jar`.

Procesador OrgaSmallI

Se adjunta como parte de este taller las hojas de detalles del procesador *OrgaSmallI*.

Ejercicios

(1) **Introducción** - Recorrer la máquina y la hoja de datos y responder:

- a) ¿Qué componentes están involucrados en cada paso del ciclo Fetch - Decode - Execute de la máquina *OrgaSmallI*?
- b) ¿Dónde agregaría (en que momento) las interrupciones dentro del ciclo de instrucción (Fetch - Decode - Execute)?
- c) ¿Dónde se almacenan las microinstrucciones?
- d) ¿Qué es lo que define la posición de una instrucción en dicha memoria?
- e) ¿Cómo se decodifica un “IF” en un microprograma?

(2) **Interrupt Controller** - Desarrollar el componente teniendo en cuenta lo siguiente:

- El flag *I* debe encenderse al recibir la señal *STI*
- El flag *I* debe apagarse al recibir la señal *CLI*
- Al recibir la señal *INT* de manera asincrónica se debe almacenar la interrupción
- Al recibir la señal *INTA* se debe dejar de almacenar la interrupción
- $INTR = 1 \leftrightarrow I = 1 \wedge$ hay interrupción

¿Qué sucede si el cable con una X del componente de saltos en la unidad de control vale siempre 0? ¿Y si vale siempre 1?

Checkpoint 1

(3) **Microinstrucciones** - Agregar en `microCode.ops` las microinstrucciones necesarias para que:

- a) En caso de haber una interrupción para atender, el Fetch haga lo siguiente:
 - guarde el PC en la dirección 0xFF
 - inhabilite las interrupciones
 - informe que ha sido recibida la interrupción
 - salte a la rutina de atención de interrupción
- b) Las instrucciones STI y CLI manejen *I* como corresponde
- c) La instrucción IRET habilite las interrupciones y salte a la dirección almacenada en 0xFF

Checkpoint 2

(4) **Ensamblar y correr** - Escribir en un archivo, compilar y cargar el siguiente programa:

```
SET R1, 0x03
SET R2, 0x00
SET R3, rai
STR [0x00], R3
STI
```

```
loop:
CMP R1, R2
JZ fin
JMP loop
```

```
fin:
CLI
```

```
halt:
JMP halt
```

```
rai:
DEC R1
IRET
```

- a) Antes de correr el programa, identificar el comportamiento esperado.
- b) ¿Cuántas interrupciones son necesarias como mínimo para llegar a **halt**?
- c) ¿Qué sucede si se reciben varias interrupciones antes del **CMP**?
- d) ¿Hay algún problema si se recibe una interrupción entre el **CMP** y el **JZ**?

(5) **Programar**

Una empresa ferroviaria adquirió un procesador OrgaSmallII para controlar sus trenes de forma segura.

Con el fin de disminuir la velocidad en curvas y así evitar accidentes, se dispone de un sensor que solicita una interrupción al detectar que se acerca una curva.

Además de dicho sensor, se dispone de un dispositivo que lee la **dirección 0xF0** periódicamente y, dependiendo del valor obtenido, realiza distintas acciones:

0 Frena el tren.

1 Velocidad para curva.

2 Velocidad máxima.

Al terminar la curva el sensor manda otra señal para indicar que puede regresar a la velocidad máxima.

Escribir un programa que cambie la velocidad como corresponda.