Guía para añadir nuevas instrucciones al motor de depuración.

Este documento muestra los pasos necesarios para añadir nuevas instrucciones y modificar la estructura del microprocesador simulado. Son necesarios ciertos conocimientos de Java para ello. No se comenta la inclusión de nuevas instrucciones al compilador ya que esta operación se describe en la documentación del microcontrolador suministrada por Xilinx.

Los pasos para añadir una nueva instrucción es la siguiente:

- En las clases `PicoInstruccion` y `MotorDebugger` es conveniente añadir una constante que indique el tipo de instrucción que se va a realizar. Por ejemplo, en el caso de añadir la instrucción de multiplicación de dos registros, podría ser:

```
…
private final int RETURNI = 45;
private final int ENABLEINT = 46;
private final int DISABLEINT = 47;
private final int MULTREGREG = 48;
…
```

- Ahora hay que introducir en el método `decodificar()` de `PicoInstruccion` la información para el desensamblador del código de operación para poder decodificarla, el mnemónico de la instrucción y su duración en ciclos. Así en el caso de este ejemplo, suponiendo que la duración son 3 ciclos y que el código de operación es 0x9---, donde los 4 bits menos significativos del primer byte es el primer registro y el segundo byte es el segundo registro.

```
…
case (byte) 0x90 :    case (byte) 0x91 :
case (byte) 0x92 :    case (byte) 0x93 :
case (byte) 0x94 :    case (byte) 0x95 :
case (byte) 0x96 :    case (byte) 0x97 :
   op = MULTREGREG;
   arg1 = b1&0x07;
   arg2 = b2&0x07;
   duracion = 3;
```

```
   return "MULT\t"+REGNAME[b1&0x07] + ", (" +
REGNAME[b2&0x07] + ")";
```

…

- Tras esto, se codificará en el depurador el código que simula la instrucción. Para ello hay que editar la clase `MotorDebugger`. Siguiendo la filosofía seguida en la implementación, se recomienda hacerlo en 2 pasos, para mantener la claridad en el código. En primer lugar, en el método `ejecutar()` añadir el '`case`' que llame al método que realmente se encargará de la simulación. Siguiendo con nuestro ejemplo:

  …

  ```
  case MULTREGREG: exeMult(op, arg1, arg2);
  ```

  …

  Ahora hay que crear el método que se encarga de la ejecución, así, el método podría ser:

```
   private void exeMult(int op, int arg1, int arg2){
        registros[arg1]*=arg2;
        carry=registros[arg1]>0xFF?true:false;
        registros[arg1] = registros[arg1] &0xFF;
        zero=(registros[arg1]==0);
        pc++;
   }
```

Además, el procesador puede ser adaptado fácilmente cambiando sus características, así es simple cambiar el número de registros del microcontrolador, el número de instrucciones del programa, el nombre de los registros o la longitud máxima de la pila. Esta información está al comienzo de la clase `PicoInstruccion`.