PROGRAMA EJEMPLO PARA DESCRIPCIÓN DEL ENSAMBLADOR DLX
Archivo .s

```
                                ; Este programa multiplica por el número pi todos
                                ; los números de una lista almacenada en memoria
                                ; y deja los resultados almacenados en las mismas
                                ; posiciones de memoria.

        .data 0                 ; Datos a partir de la dirección de memoria 0x0 (0)

        .global a               ; Para que "a" la puedan utilizar las instrucciones
a:      .double 3.14159265358979 ; Dato de tipo doble

        .global x
x:      .double 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16

        .double 17,18,19,20,21,22,23

        .global xtop
xtop:   .double 24

        .text 256               ; El código a partir de la dirección 0x100 (256)

start:  ld f2,a                 ; Carga doble
        add r1,r0,xtop          ; Suma de registro y dato inmediato

loop:   ld f0,0(r1)             ; A partir de aquí empieza un bucle
        multd f4,f0,f2          ; Multiplicación doble
        sd 0(r1),f4             ; Almacenamiento doble

        ld f6,-8(r1)
        multd f8,f6,f2
        sd -8(r1),f8

        ld f10,-16(r1)
        multd f12,f10,f2
        sd -16(r1),f12

        ld f14,-24(r1)
        multd f16,f14,f2
        sd -24(r1),f16

        sub r1,r1,#32           ; Resta de dato inmediato
        bnez r1,loop            ; Si r1 no es 0 se salta a loop
        nop
        trap #0                 ; Excepción para pasar el control al sistema
```

CONJUNTO DE REGISTROS DEL PROCESADOR DLX

Registros visibles a los programas

- **GPRs** (R0..R31): Registros (32) de 32 bits de propósito general. R0=0 siempre.
- **FPRs** (F0..F31): Registros de 32 bits para datos en coma flotante de simple precisión. Se pueden utilizar por parejas (par-impar) para almacenar datos de coma flotante en doble precisión: D0 – D30 (16 registros de 64 bits) con D0 = (F1-F0) ....
- **FPSR**: Registro de estado para las operaciones en coma flotante. Se utiliza en las comparaciones y en las excepciones de coma flotante. Todos los movimientos a y desde el registro **FPSR** se realizan a través de los registros **GPRs**.
- **PC**: Contador de Programa que contiene la dirección de la siguiente instrucción a ser captada (se incremente de 4 en 4 ya que tiene 32 bits = 4 bytes).

Registros internos

- **IMAR**: Registro inicializado con el contenido del contador de programa de la instrucción en la etapa IF.
- **IR**: Registro en la etapa IF que se carga con la dirección de la siguiente instrucción a captar.
- **A, B**: Registros que se cargan en la etapa ID con los contenidos de los registros que se operan. Estos registros constituyen las entradas a la ALU. También existen dos registros **AHI** y **BHI** que contienen los 32 bits de la parte superior de los datos en coma flotante de doble precisión.
- **BTA**: En este registro se escribe la dirección de salto en las instrucciones branch y jump. Esta dirección se calcula en la etapa ID, y si se produce el salto, el contenido de **BTA** se carga en **PC**.
- **ALU**: Registro de 32 bits donde se cargan los resultados de la ALU. Existe un registro **ALUHI** que contiene los 32 bits más altos de un datos de doble precisión.
- **DMAR**: Registro donde se carga la dirección de memoria para un acceso a la misma para una carga o almacenamiento en la etapa MEM.
- **SDR**: Registro que contiene el dato que se va a escribir en memoria en un almacenamiento (store). Existe un registro **SDRHI** para los 32 bits superiores de los datos de doble precisión.
- **LDR**: Registro donde se almacena el dato leído de memoria en una carga (load). Existe un registro **LDRHI** para los 32 bits superiores de los datos de doble precisión.

SIMULACIÓN DEL PROGRAMA EJEMPLO

A continuación se describe un posible proceso de simulación. Usualmente, en el proceso de simulación y análisis de resultados se pasará por las siguientes etapas:

(1) Cargar el fichero donde se encuentra el programa a simular

Se selecciona ***File*** en el menú principal y después, en la ventana que se abre se selecciona ***Load Code or Data***. A continuación se escribe el nombre del fichero o bien se selecciona con el cursor entre los ficheros .s que están en el directorio de trabajo. Después se selecciona ***Select*** y después ***Load***.

Tanto si el fichero se carga sin problemas como si hay algún error aparece una ventana indicándolo. Para visualizar el código cargado se puede desplegar la ventana ***Code*** donde aparecen tres columnas (Figura 1). En la de la izquierda está la dirección de memoria donde se carga la instrucción que aparece en la columna de la derecha (en ensamblador). En el centro, aparecen (en hexadecimal) los contenidos de los 32 bits (4 bytes) que ocupa la instrucción.

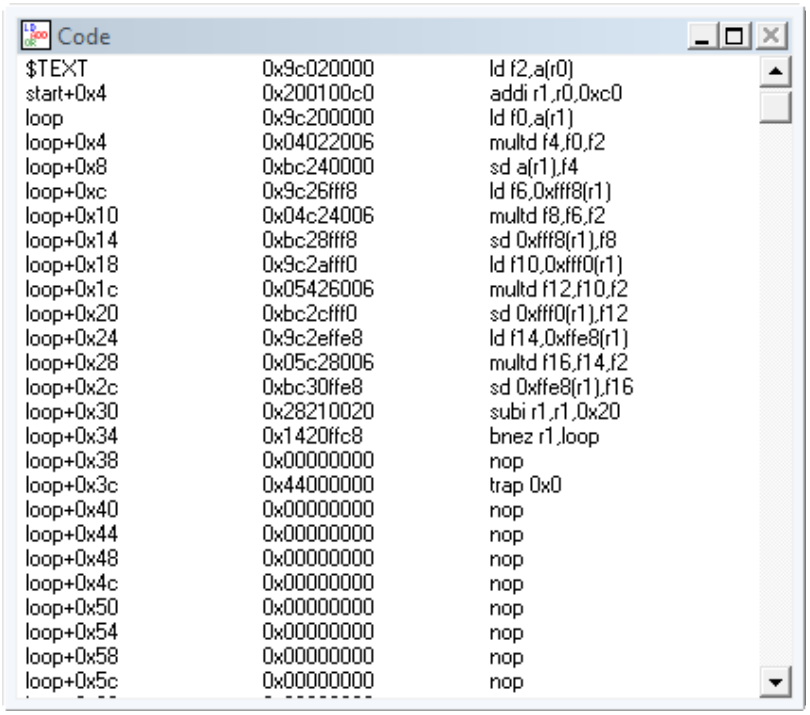

Figura 1. Ventana de código desplegada con el Programa Ejemplo cargado y una instrucción seleccionada para empezar la simulación a partir de ella.

(2) Fijar la configuración que se quiere simular para el procesador

A continuación se puede comprobar si la configuración del procesador es la adecuada para la simulación que se quiere hacer o no. Para ello se selecciona la opción ***Configuration***, y aparecerán una serie de alternativas:

- Mediante ***Floating Point Stages*** se puede cambiar el número de sumadores, multiplicadores y divisores de coma flotante que hay en el cauce y también se pueden modificar sus retardos. El número de unidades de cada tipo puede ser 8 como máximo y 1 como mínimo, y los retardos pueden ir desde 1 a 50. Los valores que aparecen por defecto corresponden a un sumador con dos ciclos de retardo, un multiplicador con cinco ciclos de retardo, y un divisor con diecinueve ciclos de retardo. Para la simulación que vamos a realizar dejaremos los valores por defecto.

- Con ***Memory Size*** se puede cambiar la cantidad de memoria (entre 512 Bytes y 16 MBytes). Dejaremos también la que hay por defecto (32 KBytes).

- También hay tres opciones que se pueden activar o desactivar. Si ***Symbolic Addresses*** está activa en la ventana de código, las direcciones de memoria que aparecen en la columna de la izquierda se expresan con sus nombre simbólicos (interesa de cara a relacionar el cargado con el listado del fichero .s correspondiente), como en la Figura 1. Si no está activa las direcciones se expresan en hexadecimal. Mediante ***Absolute Cycle Count*** se puede seleccionar que, en la ventana ***Clock Cycle Count*** aparezca el número de ciclos desde que empezó la simulación (si la opción está activa) o el número de ciclos relativo al ciclo que se está simulando. Con ***Enable Forwarding*** activa se considera en la simulación que el procesador tiene caminos de bypass que evitan ciclos de espera en el cauce cuando hay ciertas dependencias entre instrucciones que están en el cauce.

- La opción ***Store*** permite guardar la configuración que se ha seleccionado en un fichero. Esa configuración se puede cargar luego mediante la opción ***Load*** y seleccionando el fichero en cuestión para que se cargue.

(3) Asegurarse que está seleccionada la primera instrucción que se quiere simular

Antes de empezar a simular hay que asegurarse que está seleccionada la instrucción con la que se quiere empezar la ejecución del programa. Esta instrucción se puede fijar seleccionándola sobre la ventana de código. En la Figura 1 está seleccionada la primera instrucción del programa, cuya dirección es precisamente **text**.

(4) Realizar la simulación

Para elegir el tipo de simulación que se quiere realizar se utiliza la opción ***Execute***. Se puede realizar una ejecución completa del programa (seleccionando ***Run***); una simulación hasta que una instrucción que se indica se esté ejecutando en una etapa que también se puede indicar (seleccionando ***Run to***); la simulación de un ciclo de reloj a partir del estado presente (seleccionando ***Single Cycle***); o el número de ciclos que se desee a partir del ciclo actual (seleccionando ***Multiple Cycles***). Mientras se está realizando una simulación ciclo a ciclo (o de varios ciclos) se puede visualizar la evolución de las instrucciones en el cauce abriendo la ventana ***Clock Cycle Diagram***, que permite visualizar un diagrama espacio-temporal en el que se aparece la situación de las instrucciones en las distintas etapas del cauce en varios ciclos anteriores al presente. En esta ventana también aparecen flechas indicando las dependencias que existen entre las instrucciones. Las **flechas verdes** corresponden a dependencias que no dan lugar a ciclos perdidos en el cauce (stall) debido al uso de caminos de bypass o la interposición de instrucciones entre las instrucciones dependientes, etc. Si se visualiza esta ventana después de realizar la simulación completa del programa se visualizan sólo la información del cauce correspondiente a los últimos 40 ciclos del programa. En la Figura 2 se muestra el aspecto de la ventana de diagrama de ciclos de reloj en un momento de la simulación del programa, indicándose las dependencias entre las instrucciones. Mediante la ventana ***Pipeline*** se puede ver la ocupación de las distintas etapas del cauce en el ciclo que se esté simulando (siempre que la ventana se abra con un tamaño suficientemente grande).

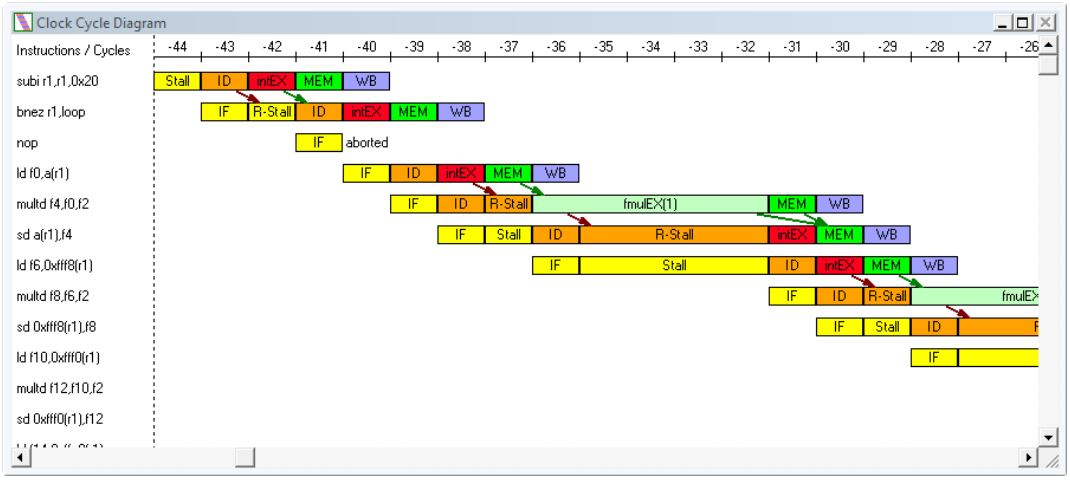

Figura 5. Diagrama de ciclos.

(5) Analizar los resultados y actuar en consecuencia (cambiar el programa y/o la configuración con la que se ha realizado la simulación y volver a simular, etc.)

Cuando se termina la simulación se pueden analizar las prestaciones y la forma en que el programa ha utilizado los recursos del procesador a partir de la ventana ***Statistics***. En esa ventana, la primera información que aparece corresponde al número de ciclos que ha tardado en ejecutarse el programa en el procesador, el número de instrucciones que se han decodificado (han pasado por la etapa ID), y el número de instrucciones que quedan en el cauce en el momento que se produce el trap hace que se ceda el control al sistema y termina la simulación. A continuación se da información de la configuración del procesador con la que se ha hecho la simulación.

Después se informa de las detenciones (stalls) que se han producido en el cauce durante la simulación. Se indican cuantos ciclos se han perdido y los tipos de dependencias que han dado lugar a esas detenciones. De esta manera se pueden analizar las características de paralelismo a nivel de instrucciones que presenta el código y se pueden extraer conclusiones respecto a posibles mejoras. También se informa del número de instrucciones de salto condicional que se han ejecutado indicando cuantas veces se ha producido salto.

Además, se indican el número de instrucciones de acceso a memoria, y de coma flotante. Por último se indica el número de traps que se han producido.

Por otra parte, los valores de las variables en memoria y los registros también se pueden visualizar para comprobar que el programa se ha ejecutado (o se está ejecutando) correctamente. Los valores de los registros se pueden ver abriendo la ventana ***Register***, y los contenidos de las variables en memoria seleccionando la opción ***Memory***.

La opción ***Memory*** permite, a su vez: (1) visualizar zonas de memoria (mediante ***Display***) indicando si se quiere que se visualicen bytes, palabras, etc. y si se quiere que se muestren en hexadecimal, decimal,...; (2) cambiar los valores de una posición de memoria (mediante ***Change***) indicando la dirección correspondiente; y (3) consultar la lista de etiquetas del programa y los valores que tienen asociados, cambiarlas, etc. (mediante ***Symbols***).

Una vez terminada una simulación se puede iniciar otra con el mismo fichero si se selecciona la opción ***Reset DLX*** en ***File***, o bien realizar otra con otro programa nuevo si se selecciona ***Reset All***.

Para salir del simulador se puede seleccionar ***Quit WINDLX***, también en ***File***.