Arquitectura de Computadoras 2019

Práctico N° 2: Test del procesador de un ciclo

Antes de comenzar, se debe bajar de las carpetas públicas los siguientes archivos:

- Set de archivos <SingleCycleProcessorPatterson-Modules> (.sv) con la descripción de los módulo del procesador conforme a los esquemas de las Fig. 1 y Fig. 2.
- Set de archivos <armv8_createAssembly> que contiene el template de un archivo ASM y su makefile para generar binarios a partir de código assembler ARMv8.

A continuación crear un proyecto cuya *Top-level Entity* sea <processor_arm> y agregar los archivos .sv del paquete <SingleCycleProcessorPatterson-Modules> y todos los archivos de descripción desarrollados en el la guia 1. Finalmente verificar las conexiones resultantes según los siguientes diagramas:

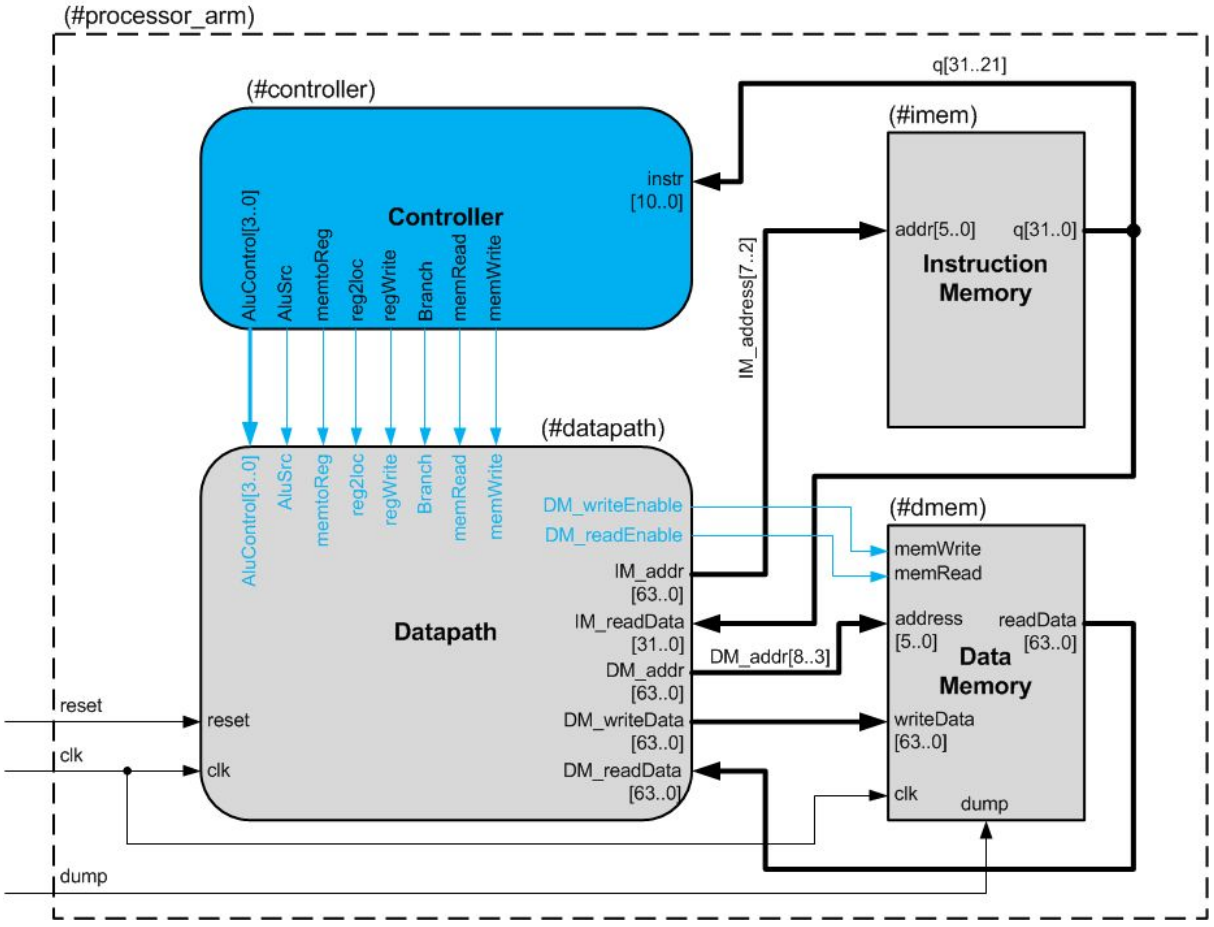

Figura 1: Top Level ARM processor

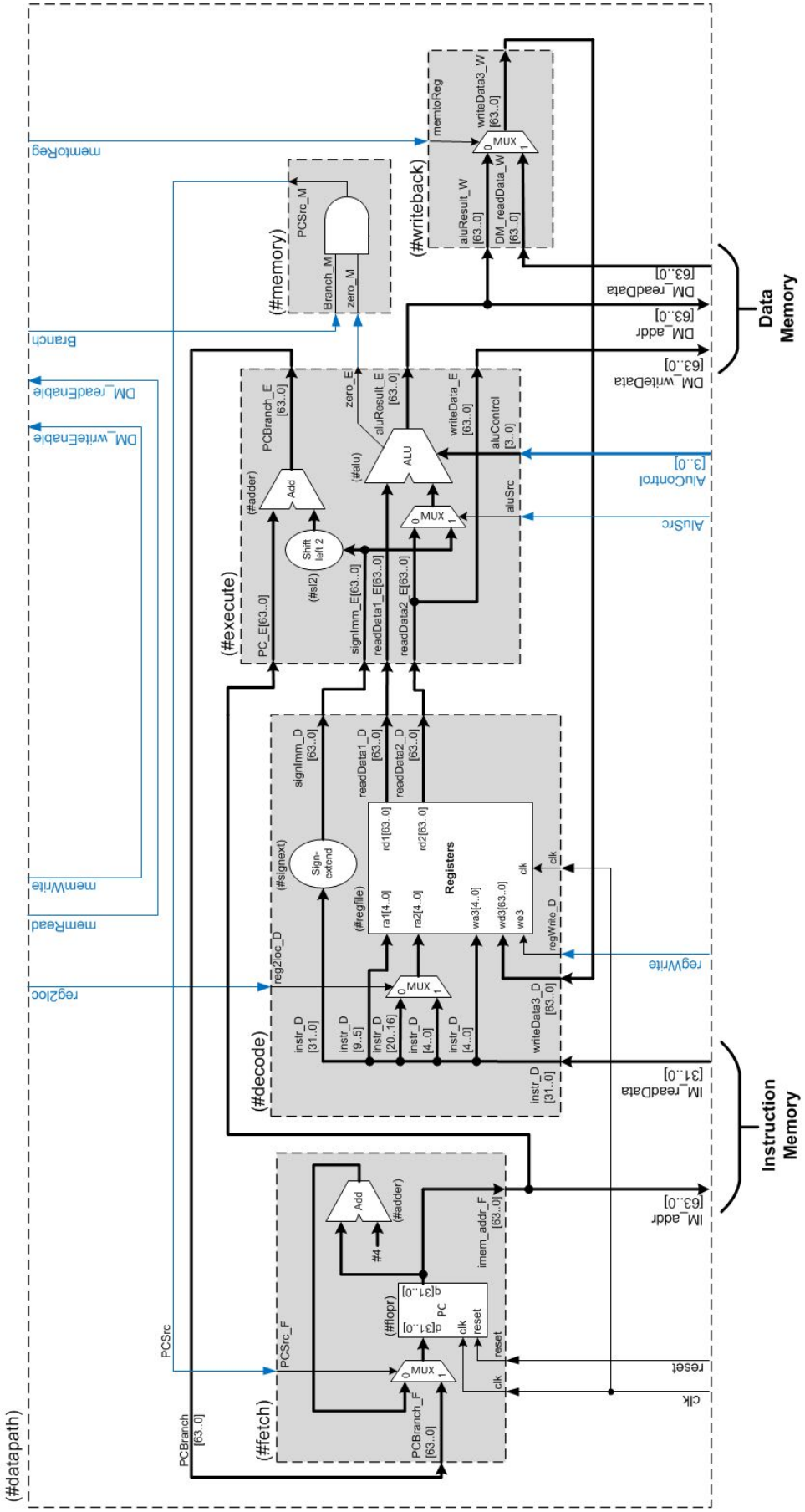

Figura 2: Datapath ARM processor

Ejercicio 1:
Asociar el test bench suministrado <processor_tb>, compilar el proyecto (solo análisis y síntesis) y correr la simulación RTL.
La memoria de programa (imem) diseñada en la guía 1 contiene precargado un programa que toma los valores de los registros X0 a X6 (previamente inicializados con los valores 0 a 6 respectivamente) y los guarda a partir de la posición “0” de memoria. Luego, resta el contenido de X14 a sí mismo, compara el resultado con “0” y salta a un lazo que escribe 0x01 en X15 y guarda este resultado en la posición “7” de la memoria.
Para asegurar el correcto funcionamiento del procesador analizar que el archivo “mem.dump” que se genera en la carpeta ...\simulation\modelsim tenga el siguiente contenido:

```
Memoria RAM de Arm:
    Address Data
0 0x0000000000000000
1 0x0000000000000001
2 0x0000000000000002
3 0x0000000000000003
4 0x0000000000000004
5 0x0000000000000005
6 0x0000000000000006
7 0x0000000000000001
8 0x0000000000000000
9 0x0000000000000000
          ...
63 0x0000000000000000
```

El archivo contiene las direcciones de la memoria de datos (columna izquierda) y su contenido (columna derecha). La memoria se inicializa con ceros y los datos guardados corresponden a la ejecución del programa de prueba.

Ejercicio 2:
Escribir programas en assembler LEGv8 en base a los siguientes enunciados, con el fin de corroborar el correcto funcionamiento del procesador implementado:

2-A) Con la menor cantidad de registros e instrucciones, inicializar con el valor de su índice las primeras N posiciones de memoria (comenzando en la dirección “0”).

2-B) Realizar la sumatoria de las primeras N posiciones de memoria y guardar el resultado en la posición N+1.

2-C) Realizar la multiplicación de dos registros: X16 y X17 y guardar el resultado en la posición “0” de la memoria.

A TENER EN CUENTA:

- Para este ejercicio se podrán utilizar SOLO las instrucciones LEGv8 soportadas en el procesador implementado en HDL (LDUR, STUR, CBZ, ADD, SUB, AND, ORR).

- Para obtener el código ensamblado en hexadecimal, se debe escribir el programa a implementar en el archivo “main.s” del paquete <armv8_createAssembly>, luego escribir por terminal:

  ```
  $ make
  ```

  y copiar en el módulo **imem** las instrucciones generadas en “main.list”, respetando el formato del ejercicio 4 de la guia 1.

  Nota: verificar que se tiene instalada la toolchain de aarch64, caso contrario escribir:

  ```
  $ sudo apt install gcc-aarch64-linux-gnu
  ```

- Verificar que los registros X0 a X30 estén inicializados con los valores 0 a 30 respectivamente en la descripción del módulo **regfile**.