Laboratorio 1 - Mejora de procesador con pipeline

Dado el procesador que venimos desarrollando en clase, se piden distintas mejoras y modificaciones para lograr un procesador con más instrucciones y que sea más eficiente. Las consignas propuestas y resueltas por el grupo son las siguientes:

Ejercicio 1

Con el objetivo de agregar funcionalidad al procesador, se pide agregar la instrucción `MOVZ` que permite mover un inmediato de 16 bits a un registro colocando los otros 48 bits en 0.

Ejecución

Lo primero realizado fue adaptar el procesador utilizando los módulos que proveía la cátedra, realizando algunas modificaciones para lograr obtener el funcionamiento y los resultados de memoria esperados.
Las modificaciones fueron varias. La que realizamos en principio fueron las propuestas por la cátedra en las "modificaciones recomendadas", luego vimos necesario agregar un bit al bus de direccionamiento en imem para ahora poder almacenar hasta 128 instrucciones en la memoria ROM. Además, utilizamos para generar el código inicial, el script también proveido por la cátedra, al cual se le pasó el archivo main.s que previamente editamos para que contenga los NOP(ADD XZR, XZR, XZR) necesarios para que el procesador funcione correctamente y evite hazards tanto de datos como de control.

Una vez el procesador quedó en funcionamiento, restaba agregar la nueva instrucción `MOVZ`. Para esto, por un lado se agregó un registro de 2 bits binarios para almacenar el LSL que se le debe aplicar al inmediato de 16 bits. Se debe almacenar dado que si no vamos a utilizar el LSL de la instrucción n+1 para el inmediato de la instrucción n.
Por otro lado, se configuraron las señales en el maindec para que en la instrucción `MOVZ`, se use el signext para extender el inmediato de 16 bits a 64 bits, y luego enviar el resultado a la ALU.
El LSL se conecta al módulo de execute, puntualmente a la ALU, donde se realiza el calculo del MOVZ. Para esto, se agregó un nuevo case en el módulo de la ALU, donde el alucontrol es 4'b0100. En este caso, se realiza un shift left lógico del inmediato de 16 bits, tantas veces como el LSL indique.
Finalmente, el resultado de la ALU continua su camino por el procesador normalmente, para poder escribirlo en el registro correspondiente. Para esto, no fue necesario agregar nada nuevo, simplemente setear las señales correspondientes en el maindec.

Las señales en maindec fueron configuradas:

- `Reg2Loc` = 0
- `ALUSrc` = 1
- `MemtoReg` = 0
- `RegWrite` = 1
- `MemRead` = 0
- `MemWrite` = 0
- `Branch` = 0
- `ALUOp` = 2'b11

Luego en el módulo de aludec, a partir del `ALUOp` se setean las señales correspondientes para que la ALU realice el shift left lógico. Puntualmente:

- `ALUControl` = 4'b0100

Finalmente, en el módulo de la ALU, se agrega un nuevo case para el `ALUControl` 4'b0100, donde se realiza el shift left lógico del inmediato. Tal que:

- `4'b0100: result = b << (LSL * 16);`

Resultados

Para probar el funcionamiento de la instrucción `MOVZ`, se agregó al archivo main.s las siguientes instrucciones para probar cada uno de los distintos lsl posibles del inmediato:

```
MOVZ X26, 0xDEAD, LSL 0
MOVZ X27, 0x7ECA, LSL 16
MOVZ X28, 0xC0CA, LSL 32
MOVZ X29, 0xFEED, LSL 48
```

Resultando en el siguiente programa:

```
    STUR X1, [X0, #0]
    STUR X2, [X0, #8]
    STUR X3, [X16, #0]
    ADD X3, X4, X5
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    STUR X3, [X0, #24]
    SUB X3, X4, X5
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    STUR X3, [X0, #32]
    SUB X4, XZR, X10
```

```
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
STUR X4, [X0, #40]
ADD X4, X3, X4
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
STUR X4, [X0, #48]
SUB X5, X1, X3
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
STUR X5, [X0, #56]
AND X5, X10, XZR
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
STUR X5, [X0, #64]
AND X5, X10, X3
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
STUR X5, [X0, #72]
AND X20, X20, X20
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
STUR X20, [X0, #80]
ORR X6, X11, XZR
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
STUR X6, [X0, #88]
ORR X6, X11, X3
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
STUR X6, [X0, #96]
LDUR X12, [X0, #0]
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
ADD X7, X12, XZR
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
STUR X7, [X0, #104]
STUR X12, [X0, #112]
ADD XZR, X13, X14
ADD XZR, XZR, XZR //NOP
ADD XZR, XZR, XZR //NOP
```

```
    STUR XZR, [X0, #120]
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    CBZ X0, L1
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    STUR X21, [X0, #128]
L1:
    STUR X21, [X0, #136]
    ADD X2, XZR, X1
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
L2:
    SUB X2, X2, X1
    ADD X24, XZR, X1
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    STUR X24, [X0, #144]
    ADD X0, X0, X8
    CBZ X2, L2
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    STUR X30, [X0, #144]
    ADD X30, X30, X30
    SUB X21, XZR, X21
    ADD XZR, XZR, XZR //NOP
    ADD X30, X30, X20
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    LDUR X25, [X30, #-8]
    ADD X30, X30, X30
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    ADD X30, X30, X16
    ADD XZR, XZR, XZR //NOP
    ADD XZR, XZR, XZR //NOP
    STUR X25, [X30, #-8]
    MOVZ X26, 0xDEAD, lsl #0
    MOVZ X27, 0x7ECA, lsl #16
```

```
    MOVZ X28, 0xC0CA, lsl #32
    MOVZ X29, 0xFEED, lsl #48
    STUR X26, [x30, #0]  // MEM 22: 0xDEAD
    STUR X27, [x30, #8]  // MEM 23: 0x7ECA0000
    STUR X28, [x30, #16] // MEM 24: 0xC0CA00000000
    STUR X29, [x30, #24] // MEM 25: 0xFEED000000000000
finloop: CBZ XZR, finloop
```

El resultado de la ejecución de estas instrucciones produce el siguiente resultado en memoria:

```
000000000000DEAD
000000007ECA0000
0000C0CA00000000
FEED000000000000
```

El resultado de agregar la instrucción `MOVZ` fue el esperado. El procesador continua funcionando correctamente y los cambios realizados en cuanto a las señales y el registro de LSL pueden verse en la siguiente imagen: