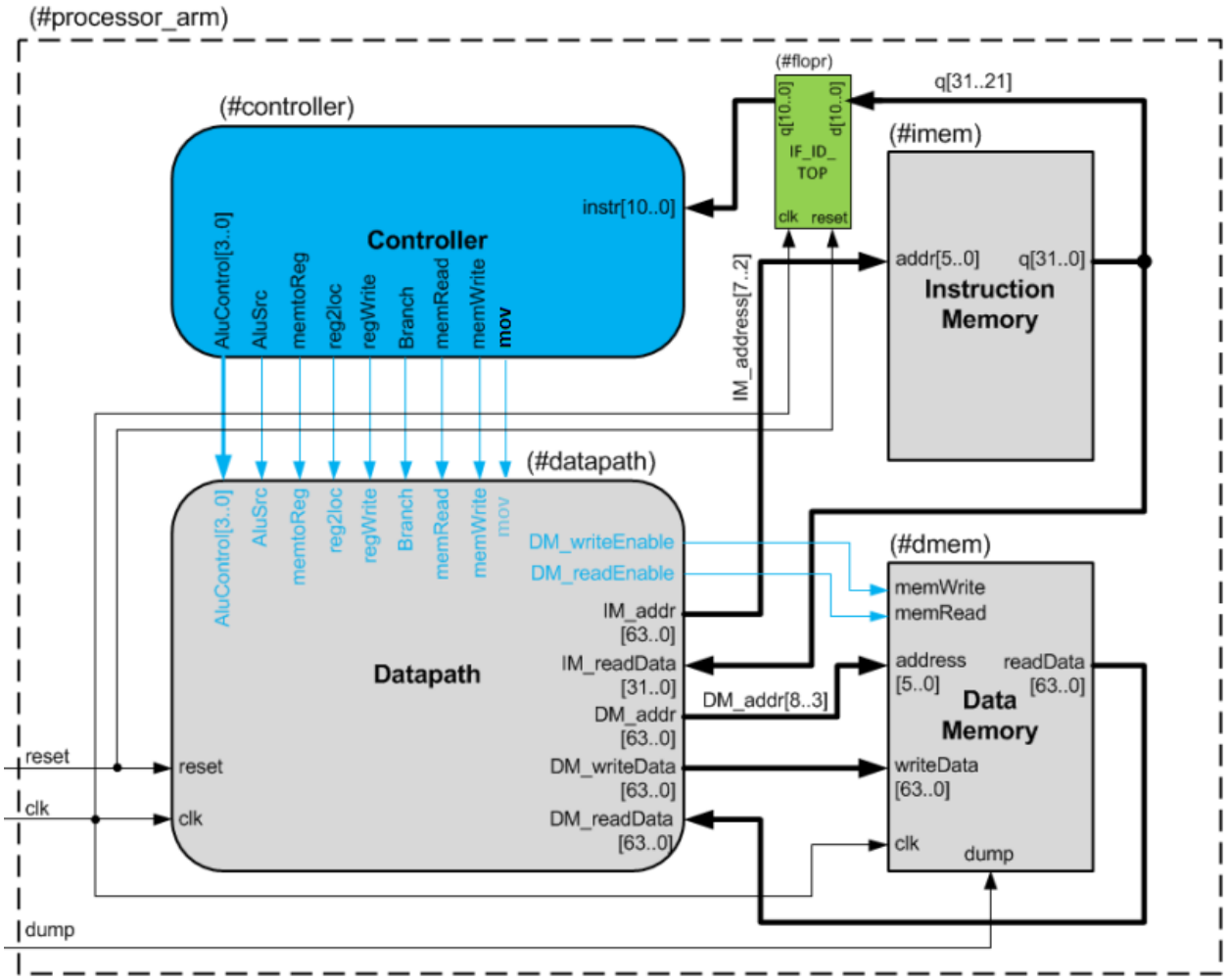
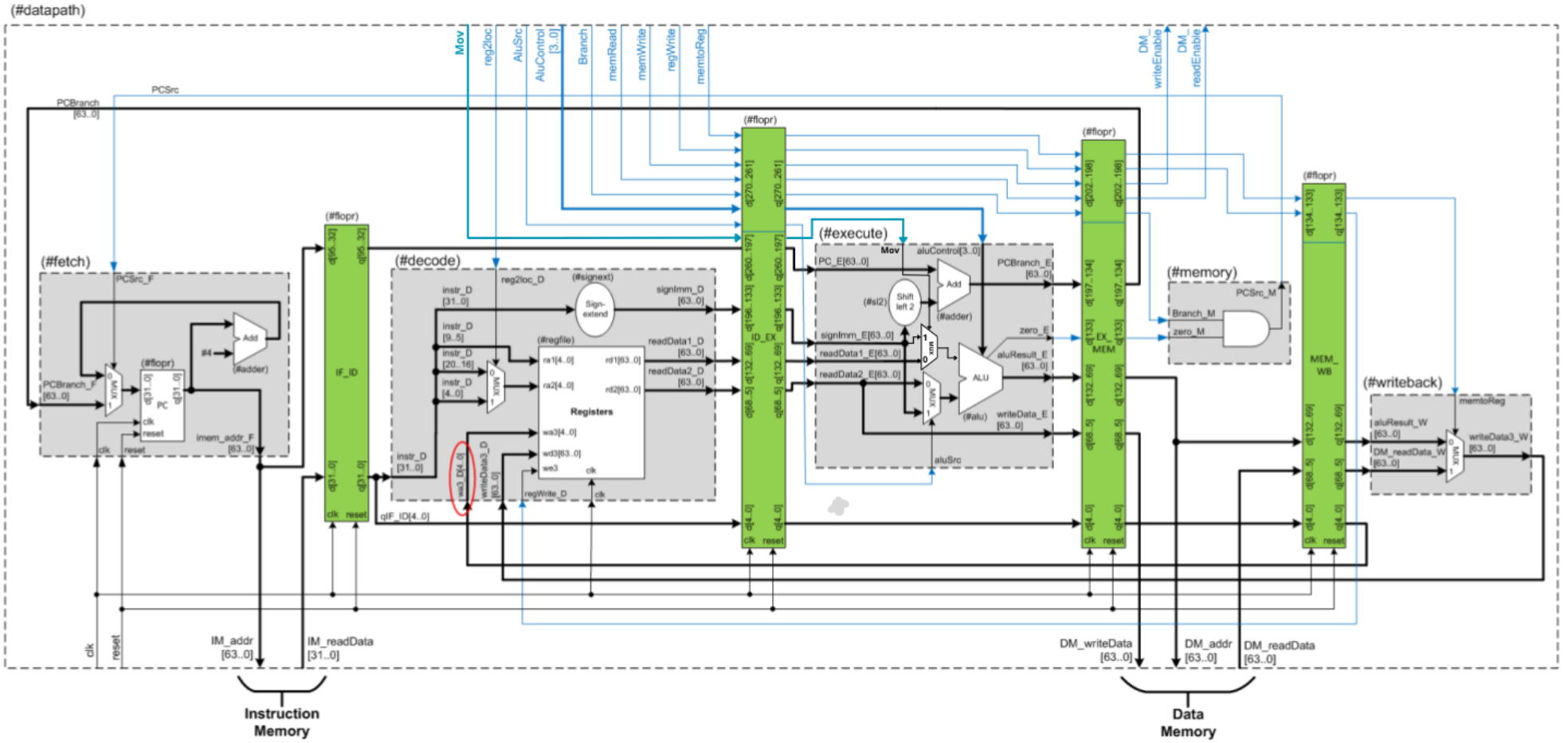
INFORME LAB 1
ARQUITECTURA DEL COMPUTADOR

Integrantes

- Cristian Ariel Muñoz
- Santiago León Torres Banner
- Damián Feigelmüller

Diagramas:




(#datapath)
Mov
reg2loc
AluSrc
AluControl [3..0]
Branch
memRead
memWrite
regWrite
memtoReg
DM_writeEnable
DM_readEnable
PCSrc
PCBranch [63..0]
(#flopr)
(#fetch)
(#decode)
(#execute)
(#memory)
(#writeback)
PCSrc_F
Add
#4
(#adder)
PCBranch_F [63..0]
PC
clk
reset
imem_addr_F [63..0]
IF_ID
instr_D [31..0]
reg2loc_D
(#signext)
Sign-extend
signImm_D [63..0]
instr_D [9..5]
instr_D [20..16]
instr_D [4..0]
(#regfile)
ra1[4..0]
ra2[4..0]
wa3[4..0]
wd3[63..0]
we3
rd1[63..0]
rd2[63..0]
Registers
readData1_D [63..0]
readData2_D [63..0]
writeData3_D [63..0]
regWrite_D
qIF_ID[4..0]
ID_EX
EX_MEM
MEM_WB
PC_E[63..0]
aluControl[3..0]
PCBranch_E [63..0]
(#sl2)
Shift left 2
signImm_E[63..0]
readData1_E[63..0]
readData2_E[63..0]
zero_E
aluResult_E [63..0]
writeData_E [63..0]
ALU
(#alu)
aluSrc
PCSrc_M
Branch_M
zero_M
memtoReg
aluResult_W [63..0]
DM_readData_W [63..0]
writeData3_W [63..0]
IM_addr [63..0]
IM_readData [31..0]
DM_writeData [63..0]
DM_addr [63..0]
DM_readData [63..0]
Instruction Memory
Data Memory

Modificaciones:

Se agregó la señal de control *mov* que se activará cuando hay una instrucción MOVZ, la cuál llegará a un **mux2** en **execute** que determinará qué valor entra al primer término de la **alu**, si el *signImm_E* o el *readData1_E*. Las entidades modificadas al agregar la señal fueron las siguientes:

- **processor_arm**
- **controller**
- **maindec**
- **datapath**
- **ID_EX**
- **execute**

Para poder interpretar la instrucción MOVZ se agregó el caso necesario en el **maindec** con el opcode provisto, donde el *aluop* se asigna a 0b11. Además de modificar el **aludec** agregando el caso de *aluop* = 0b11 para obtener el alucontrol necesario y así poder realizar la operación que necesitamos en la **alu**.

En la **alu** se agregó el caso de *Opcode* = 0b1000 donde se pasa directamente el valor a, el cuál tendrá calculado lo que se escribirá en el registro de la instrucción MOVZ.

En la entidad **signext** se agregó el caso del opcode de MOVZ, en donde, en base al LSL y al *MOV_immediate*, se forma el valor que se guardara en el registro.

En la entidad **execute** se agregó un **mux2** llamado **muxi**, el cuál determina, en base a la señal de control *mov,* qué valor ingresa en la **alu**. Siendo, en el caso de *mov* igual a 1, el valor calculado que iría en el registro a escribir de la instrucción MOVZ.

Programa en assembler LEGv8 modificado:

```
STUR X1, [X0, #0]
STUR X2, [X0, #8]
STUR X3, [X16, #0]
ADD X3, X4, X5
ADD X29, XZR, XZR
ADD X29, XZR, XZR
STUR X3, [X0, #24]
SUB X3, X4, X5
ADD X29, XZR, XZR
ADD X29, XZR, XZR
STUR X3, [X0, #32]
SUB X4, XZR, X10
ADD X29, XZR, XZR
ADD X29, XZR, XZR
STUR X4, [X0, #40]
ADD X4, X3, X4
ADD X29, XZR, XZR
ADD X29, XZR, XZR
STUR X4, [X0, #48]
SUB X5, X1, X3
ADD X29, XZR, XZR
ADD X29, XZR, XZR
STUR X5, [X0, #56]
AND X5, X10, XZR
ADD X29, XZR, XZR
ADD X29, XZR, XZR
STUR X5, [X0, #64]
AND X5, X10, X3
ADD X29, XZR, XZR
ADD X29, XZR, XZR
STUR X5, [X0, #72]
AND X20, X20, X20
ADD X29, XZR, XZR
ADD X29, XZR, XZR
STUR X20, [X0, #80]
MOVZ X10, #43981, LSL 16
ADD X29, XZR, XZR
ADD X29, XZR, XZR
STUR X10, [X0, #208]
ORR X6, X11, XZR
ADD X29, XZR, XZR
ADD X29, XZR, XZR
STUR X6, [X0, #88]
ORR X6, X11, X3
ADD X29, XZR, XZR
ADD X29, XZR, XZR
```

```
    STUR X6, [X0, #96]
    MOVZ X10, #48830, LSL 32
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    STUR X10, [X0, #216]
    LDUR X12, [X0, #0]
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    ADD X7, X12, XZR
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    STUR X7, [X0, #104]
    STUR X12, [X0, #112]
    ADD XZR, X13, X14
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    STUR XZR, [X0, #120]
    CBZ X0, L1
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    STUR X21, [X0, #128]
L1:
    STUR X21, [X0, #136]
    ADD X2, XZR, X1
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
L2:
    SUB X2, X2, X1
    ADD X24, XZR, X1
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    STUR X24, [X0, #144]
    ADD X0, X0, X8
CBZ X2, L2
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    STUR X30, [X0, #144]
    ADD X30, X30, X30
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    SUB X21, XZR, X21
    ADD X30, X30, X20
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    LDUR X25, [X30, #-8]
    ADD X30, X30, X30
```

```
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    ADD X30, X30, X16
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    STUR X25, [X30, #-8]
    MOVZ X10, #65535, LSL 0
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    STUR X10, [X0, #160]
    MOVZ X10, #51966, LSL 16
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    STUR X10, [X0, #168]
    MOVZ X10, #49354, LSL 32
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    STUR X10, [X0, #176]
    MOVZ X10, #0, LSL 48
    ADD X29, XZR, XZR
    ADD X29, XZR, XZR
    STUR X10, [X0, #184]

finloop:
    CBZ XZR, finloop
```