ipPROCESS

# ARCHITECTURE SPECIFICATION

32-bit $\mu$DLX Core Processor

Universidade Federal da Bahia

**Version: 1.0**

# GNU LGPL License

# Revision History

| Date | Description | Author(s) |
|---|---|---|
| 04/27/2014 | Conception | João Carlos Bittencourt |
| 04/30/2014 | Instruction layout description | João Carlos Bittencourt |
| 05/09/2014 | • Text revision;<br>• Update diagrams and instruction layout;<br>• Update instruction fetch I/O definitions;<br>• Missing pictures inclusion;<br>• Include memory access and write back pin/port definitions; | João Carlos Bittencourt |
| 04/13/2014 | Missing pictures | João Carlos Bittencourt |
| 04/15/2014 | • Fix instruction fetch pin definitions;<br>• Include instruction fetch datapath;<br>• Include pipeline registers definitions;<br>• Fix memory access stage pin/port definitions;<br>• Include write back data path; | João Carlos Bittencourt |
| 04/15/2014 | Add execute block diagram | Igo Amauri Luz |
| 04/16/2014 | • Add branch prediction signal to ID and IF blocks and pin definitions;<br>• Add table for pin definitions in Execute stage<br>• Add branch prediction signal in pipeline registers definitions;<br>• Include architecture interface figure; | João Carlos Bittencourt |
| 04/18/2014 | Add execute datapath diagram | Igo Amauri Luz |
| 04/23/2014 | Formating; Stakeholders listing | João Carlos Bittencourt |

# CONTENTS


1 **Introduction** 5

1.1 Purpose . . . . . 5

1.2 Stakeholders . . . . . 5

1.3 Document Outline Description . . . . . 5

1.4 Acronyms and Abbreviations . . . . . 5

2 **Architecture Overview** 7

2.1 Interface Architecture . . . . . 7

2.2 Block Diagram . . . . . 7

2.3 Pin/Port Definitions . . . . . 8

2.4 Parameters and Configurations . . . . . 8

3 **Instructions Layout** 9

3.1 Instructions Set . . . . . 9

3.2 I-type Instruction . . . . . 9

3.3 R-type Instruction . . . . . 10

3.4 J-type Instruction . . . . . 10

4 **Architecture Description** 11

4.1 Instruction Fetch . . . . . 11

4.1.1 Block Diagram . . . . . 11

4.1.2 Pin/Port Definitions . . . . . 11

4.1.3 Internal Datapath . . . . . 11

4.2 Instruction Decode/Register Fetch . . . . . 13

4.2.1 Block Diagram . . . . . 13

4.2.2 Pin/Port Definitions . . . . . 13

4.2.3 Internal Datapath . . . . . . . . . . . . . . . . . . . . . . . . . 14
4.3 Execute/Address Calculate . . . . . . . . . . . . . . . . . . . . . . . 15
4.3.1 Block Diagram . . . . . . . . . . . . . . . . . . . . . . . . . 15
4.3.2 Pin/Port Definitions . . . . . . . . . . . . . . . . . . . . . . . . 15
4.3.3 Internal Datapath . . . . . . . . . . . . . . . . . . . . . . . . . 16
4.4 Memory Access . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
4.4.1 Block Diagram . . . . . . . . . . . . . . . . . . . . . . . . . 18
4.4.2 Pin/Port Definitions . . . . . . . . . . . . . . . . . . . . . . . . 18
4.4.3 Internal Datapath . . . . . . . . . . . . . . . . . . . . . . . . . 18
4.5 Write Back . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
4.5.1 Block Diagram . . . . . . . . . . . . . . . . . . . . . . . . . 20
4.5.2 Pin/Port Definitions . . . . . . . . . . . . . . . . . . . . . . . . 20
4.5.3 Internal Datapath . . . . . . . . . . . . . . . . . . . . . . . . . 20
4.6 Pipeline Register Description . . . . . . . . . . . . . . . . . . . . . . . 22
4.6.1 Instruction Fetch/Instruction Decode . . . . . . . . . . . . . . . . . 22
4.6.2 Instruction Decode/Execute . . . . . . . . . . . . . . . . . . . . . 22
4.6.3 Execute/Memory Access . . . . . . . . . . . . . . . . . . . . . . 22
4.6.4 Memory Access/Write Back . . . . . . . . . . . . . . . . . . . . . 23
4.7 Memory and Device Interface . . . . . . . . . . . . . . . . . . . . . . 24
4.8 SRAM Controller . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
4.8.1 Block Diagram . . . . . . . . . . . . . . . . . . . . . . . . . 25
4.8.2 Pin/Port Definitions . . . . . . . . . . . . . . . . . . . . . . . . 25
4.9 SDRAM Controller . . . . . . . . . . . . . . . . . . . . . . . . . . 27
4.9.1 Block Diagram . . . . . . . . . . . . . . . . . . . . . . . . . 27
4.9.2 Pin/Port Definitions . . . . . . . . . . . . . . . . . . . . . . . . 27
4.10 Control Micro-instructions Description . . . . . . . . . . . . . . . . . . 29
4.10.1 Block Diagram . . . . . . . . . . . . . . . . . . . . . . . . . 29
4.10.2 Pin/Port Definitions . . . . . . . . . . . . . . . . . . . . . . . . 29
4.11 Bootloader . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30

# 1. Introduction

## 1.1. Purpose

The main purpose of this document is to define specifications of a uDLX implementation and to provide a full overview of the design. This specifications defines all implementation parameters that composes the general uDLX requirements and specification. This definitions include processor operation modes, instruction set (ISA) and internal registers characteristics. This document also include detailed information of pipeline stages architecture, buses and other supplemental units.

## 1.2. Stakeholders

| Name | Roles/Responsibilities |
|---|---|
| Igo Amauri Luz | TBD |
| João Carlos Bittencourt | TBD |
| Lauê Rami Costa | TBD |
| Linton Thiago Esteves | TBD |
| Victor Valente | TBD |

## 1.3. Document Outline Description

This document is outlined as follow:

- Section 2: This section presents the core processor block diagram, Pin/Port definitions and global parameters and configuration directives.
- Section 3: This section presents the $\mu$DLX instruction layout and specifications.
- Section 4: This section presents a description of each pipeline stage block, including pin definitions, signals and internal datapath.

## 1.4. Acronyms and Abbreviations

Along this and other documents part of this project, it will be recurrent the usage of some acronyms and abbreviations. In order to keep track of this elements the Table 1 presents a set of abbreviations used and its corresponding meaning.

Table 1: Acronym and descriptions of elements in this document.

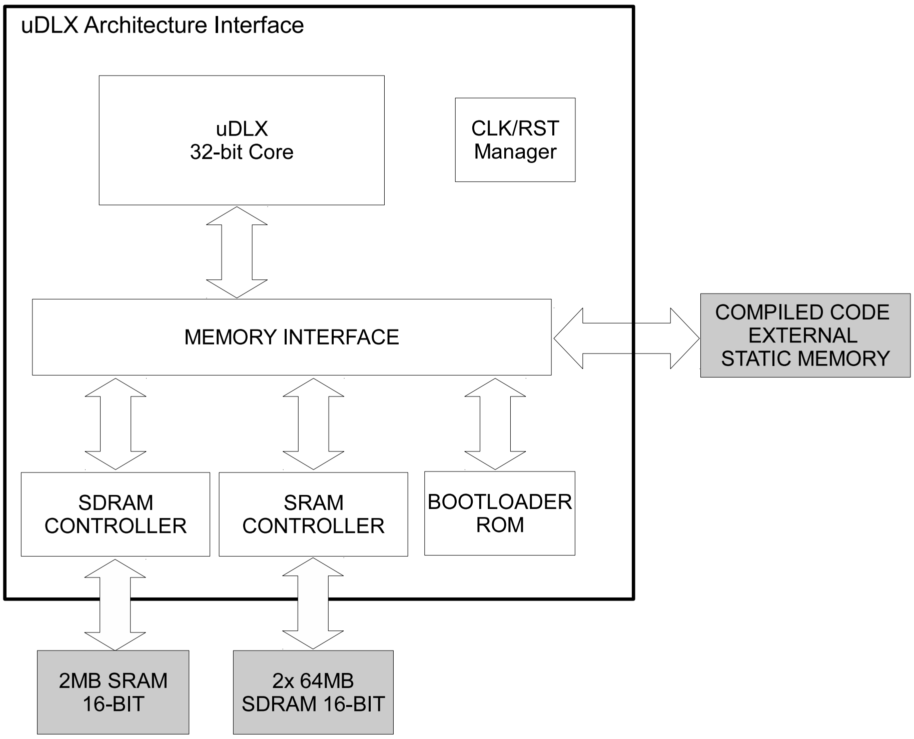
| Acronym | Description |
|---|---|
| RISC | Reduced Instruction Set Computer |
| GPR | General Purpose Registers |
| FPGA | Field Gate Programmable Array |
| GPPU | General Purpose Processing Unit |
| SDRAM | Synchronous Dynamic Random Access Memory |
| HDL | Hardware Description Language |
| RAW | Read After Write |
| CPU | Central Processing Unit |
| ISA | Instruction Set Architecture |
| ALU | Arithmetic and Logic Unit |
| PC | Program Counter |
| RFlags | Flags Register |
| Const | Constant |
| BPM | Branch Prediction Buffer |

# 2. Architecture Overview

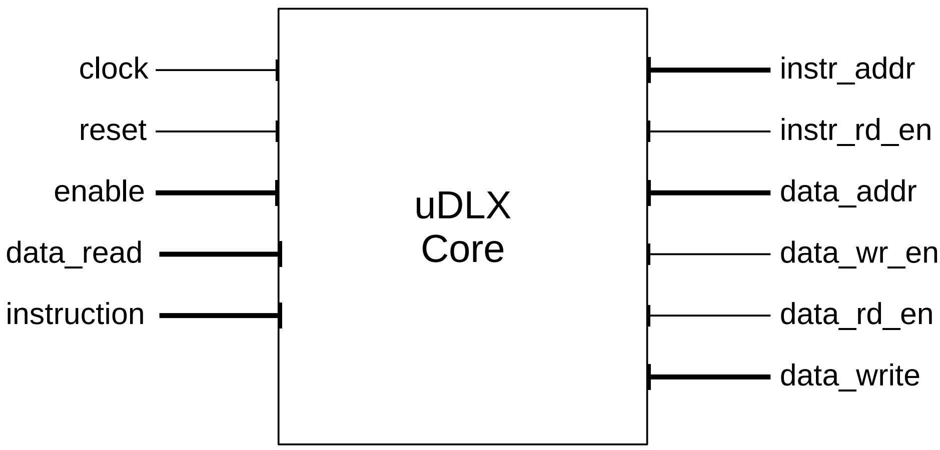
## 2.1. Interface Architecture

The $\mu$DLX architecture interface is composed by the following components.

- **$\mu$DLX 32-bit Core:** The core four-deep pipeline processor.
- **Memory Interface:** Provides a middle layer between the core processor and the external memories. This interface also controls the bootloader process.
- **SDRAM Controller:** Provides the interface for controlling the external SDRAM.
- **SRAM Controller:** Provides the interface for controlling the external SRAM.




## 2.2. Block Diagram



## 2.3. Pin/Port Definitions

| Name | Length | Direction | Description |
|---|---|---|---|
| clock | 1 | input | CPU core clock |
| reset | 1 | input | CPU core reset |
| instruction | 32 | input | SRAM instruction data |
| data_read | 32 | input | SDRAM read data |
| instr_addr | 20 | input | SRAM address |
| instr_rd_en | 1 | output | SRAM read enable |
| data_addr | 13 | output | SDRAM address |
| data_wr_en | 1 | output | SDRAM write enable |
| data_rd_en | 1 | output | SDRAM read enable |
| data_write | 32 | output | SDRAM write data |

## 2.4. Parameters and Configurations

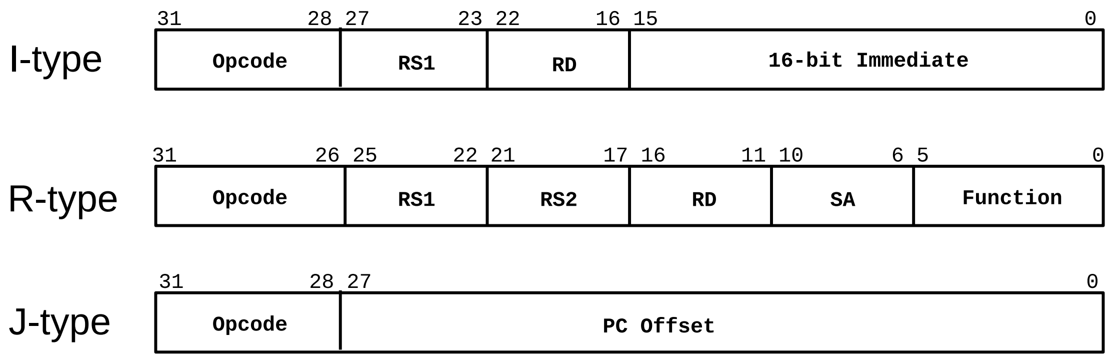
| Name | Value | Description |
|---|---|---|
| | | |

## 3. Instructions Layout

In order to be the most close to the DLX and also be as compatible as possible to MIPS R2000 and R3000 architecture few instructions were added to the instruction set. Instructions that are not present in DLX and MIPS architecture are added in not used OPCODES.

### 3.1. Instructions Set

DLX instruction structure has 5 types of instructions, the floating point instructions were removed. The 3 DLX instructions types supported in the project are shown in figure below.




NOP instruction has zero in all bits and will not be mapped to an instruction type.

### 3.2. I-type Instruction

Immediate instructions use the immediate data to address a load and store operation, make arithmetic operations and make brachs using Some arithmetic immediate instructions were added because they make easier assembly programming. The conditional branch operations BEQZ and BNEQZ were added to enable some compiler compatibility and future core improvement.

| OPCODE | Mneumonic | Operation |
|---|---|---|
| 100011/0x23 | lw | $R_D = mem$ |
| 101011/0x2b | sw | $mem = R_D$ |
| 101010/0x2A | brfl | $TBD$ |
| 001000/0x08 | addi | $R_D = R_S1 + Sext(imm)$ |
| 001010/0x10 | subi | $R_D = R_S1 - Sext(imm)$ |
| 001101/0x12 | andi | $R_D = R_S1 \wedge Sext(imm)$ |
| 001101/0x13 | ori | $R_D = R_S1 \vee Sext(imm)$ |
| 000100/0x04 | beqz | $PC = PC + 4 + (R_S1 = 0\ ?\ SExt(imm) : 0)$ |
| 000101/0x05 | bnez | $Fixme$ |
| 000111/0x16 | jr | $PC = R_S1$ |

Only LW, SW, BRFL, and JR instructions are in the p?oject requirements, the other were added to make compatibility and some codes easier.

## 3.3. R-type Instruction

This instructions realize registers operations, most operations are arithmetic. All arithmetic opcodes are zero, differing only in the function value.

| OPCODE | Mneumonic | Operation |
|---|---|---|
| 100000/0x20 | add | $R_D = R_S1 + R_S2$ |
| 100010/0x22 | sub | $R_D = R_S1 - R_S2$ |
| 100100/0x24 | and | $R_D = R_S1 \wedge R_S2$ |
| 100101/0x25 | or | $R_D = R_S1 \vee R_S2$ |
| 011000/0x18 | mult | $R_D = R_S1\ .\ R_S2$ |
| 011010/0x1A | div | $R_D = R_S1\ /\ R_S2$ |
| 011100/0x1C | cmp | $R_D = R_S1\ cmp\ R_S2$ |
| 011101/0x1D | not | $R_D = \neg R_S2$ |

The MULT, DIV, CMP, and NOT instructions were added in DLX instruction set

## 3.4. J-type Instruction

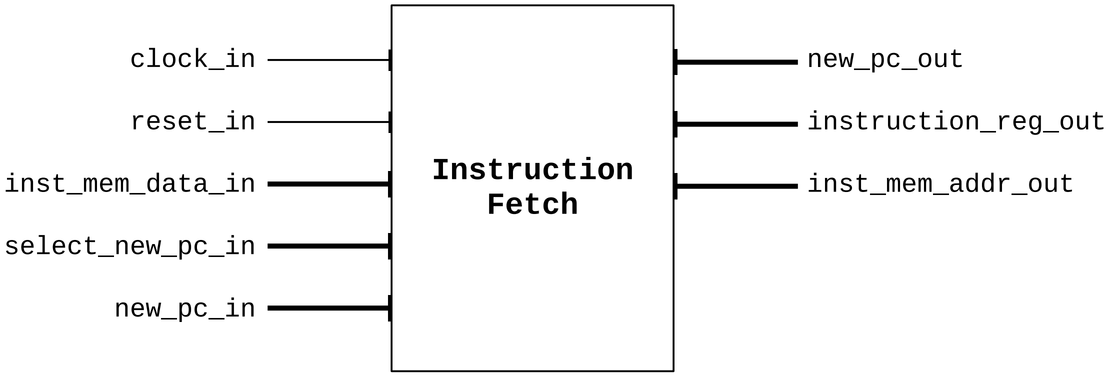
Instructions related with instruction memory jump. The instructions are JPC (J),RET (RFE), CALL (TRAP).

| OPCODE | Mneumonic | Operation |
|---|---|---|
| 000010/0x02 | jpc(j) | PC = PC + 4 |
| 111110/0x3e | call(trap) | trap = 1;<br>EPC = PC;<br>PC = SISR;<br>ESR = SR;<br>ECA = masked CA;<br>SR = 0;<br>EDATA = SExt(imm);<br>Clear CA but catch new interrupt events |
| 111111/0x3f | ret(rfe) | SR = ESR;<br>PC = EPC |

# 4. Architecture Description

## 4.1. Instruction Fetch

### 4.1.1. Block Diagram



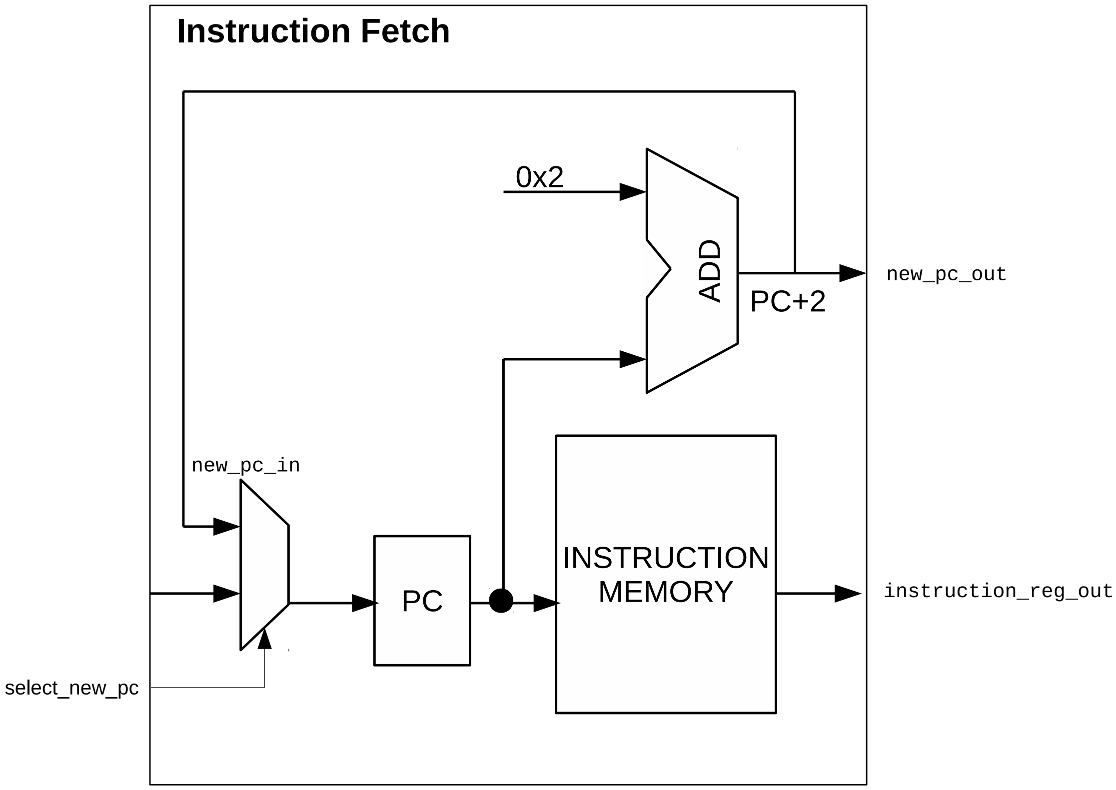

### 4.1.2. Pin/Port Definitions

| Name | Length | Direction | Description |
|---|---|---|---|
| clock_in | 1 | input | CPU core clock |
| reset_in | 1 | input | CPU core reset |
| inst_mem_data_in | TBD | input | SRAM data |
| select_new_pc_in | 1 | input | Signal used for branch not taken |
| new_pc_in | 20 | input | New value of PC |
| new_pc_out | 20 | output | Updated value of PC |
| instruction_reg_out | 32 | output | CPU core instruction |
| inst_mem_addr_out | 20 | output | SRAM address |

### 4.1.3. Internal Datapath

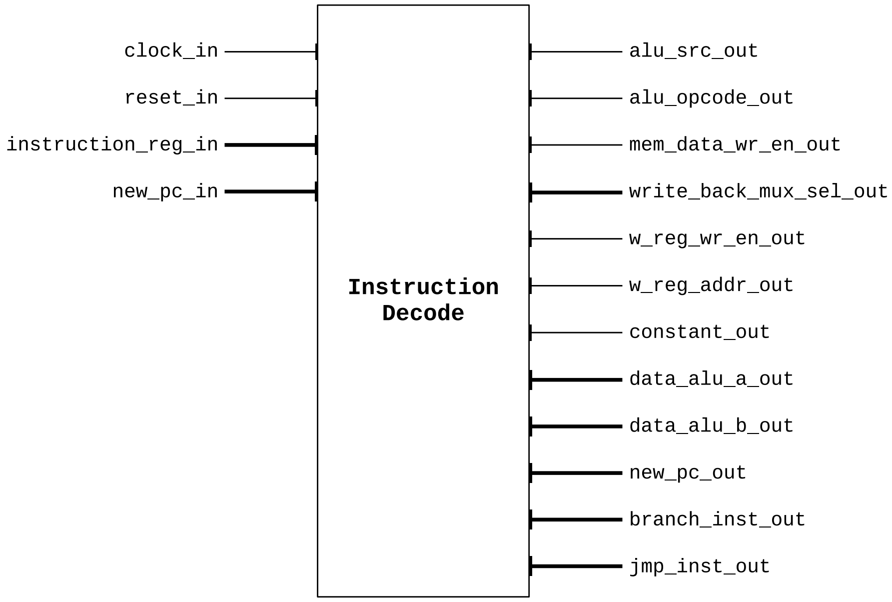
The internal data path is composed by the following components.

**Program Counter** : During the instruction time of an instruction this is the address of the instruction word. The address of the instruction that occurs during the next instruction time is determined by assigning a value to PC during an instruction time. If no value is assigned to PC during an instruction time by any pseudocode statement, it is automatically incremented by 2 before the next instruction time.


Instruction Fetch
0x2
ADD
PC+2
new_pc_out
new_pc_in
PC
INSTRUCTION MEMORY
instruction_reg_out
select_new_pc

## 4.2. Instruction Decode/Register Fetch

### *4.2.1. Block Diagram*




### *4.2.2. Pin/Port Definitions*

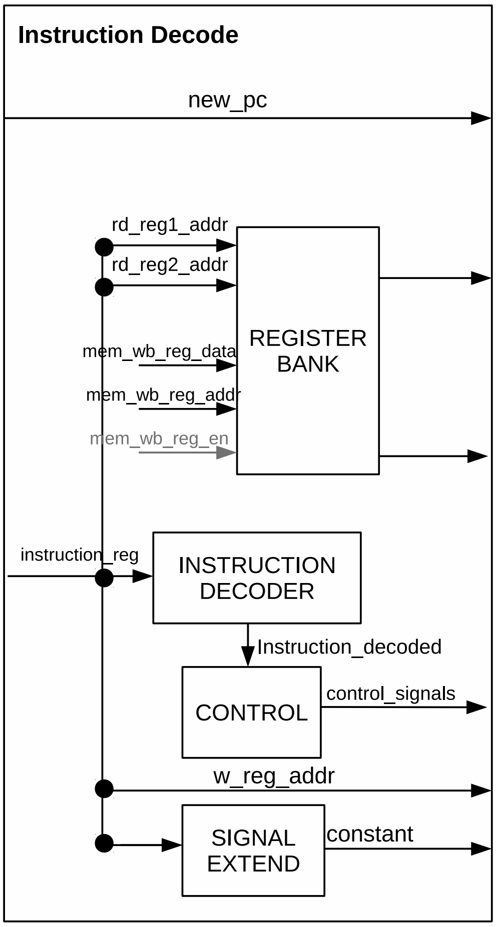
| Name | Length | Direction | Description |
|---|---|---|---|
| clock_in | 1 | input | CPU core clock |
| reset_in | 1 | input | CPU core reset |
| instruction_reg_in | 32 | input | CPU core instruction |
| new_pc_in | 20 | input | Updated value of PC |
| alu_src_out | 1 | output | Alu mux source selector |
| alu_opcode_out | 3 | output | Alu opcode |
| mem_data_wr_en_out | 1 | output | Data memory write enable |
| write_back_mux_sel_out | 1 | output | Write back mux selector |
| w_reg_wr_en_out | 1 | output | GPR bank write enable signal |
| w_reg_addr_out | TBD | output | GPR bank destiny address |
| constant_out | 32 | output | 32-bit Sign-extended constant |
| data_alu_a_out | 32 | output | ALU input A data |
| data_alu_b_out | 32 | output | ALU input B data |
| new_pc_out | 20 | output | Updated value of PC delayed |
| branch_inst_out | 1 | output | Conditional branch instruction |

continued on next page

| continued from previous page | | | |
|---|---|---|---|
| **Name** | **Length** | **Direction** | **Description** |
| jmp_inst_out | 1 | output | Incoditional branch instruction |

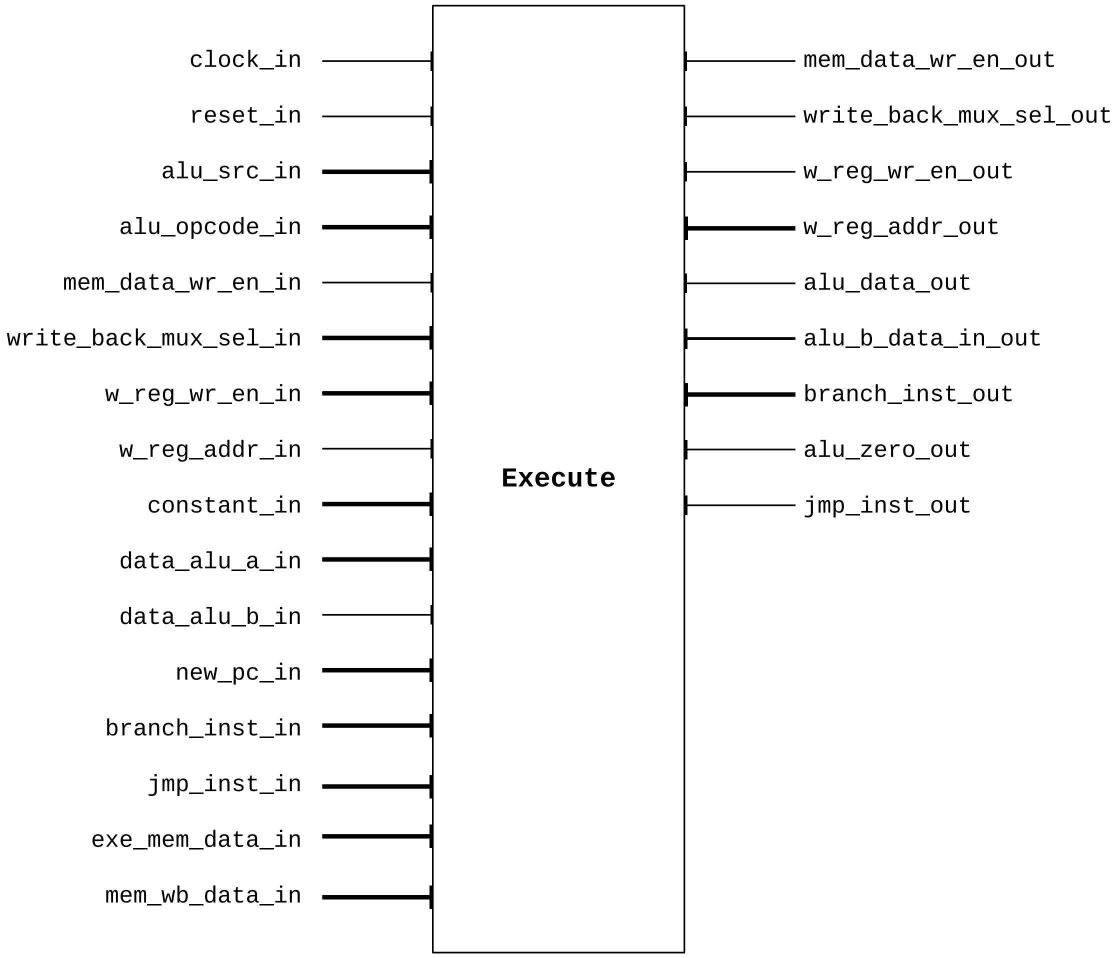
### 4.2.3. *Internal Datapath*

The internal data path is composed by the following components.



## 4.3. Execute/Address Calculate

### 4.3.1. Block Diagram




### 4.3.2. Pin/Port Definitions

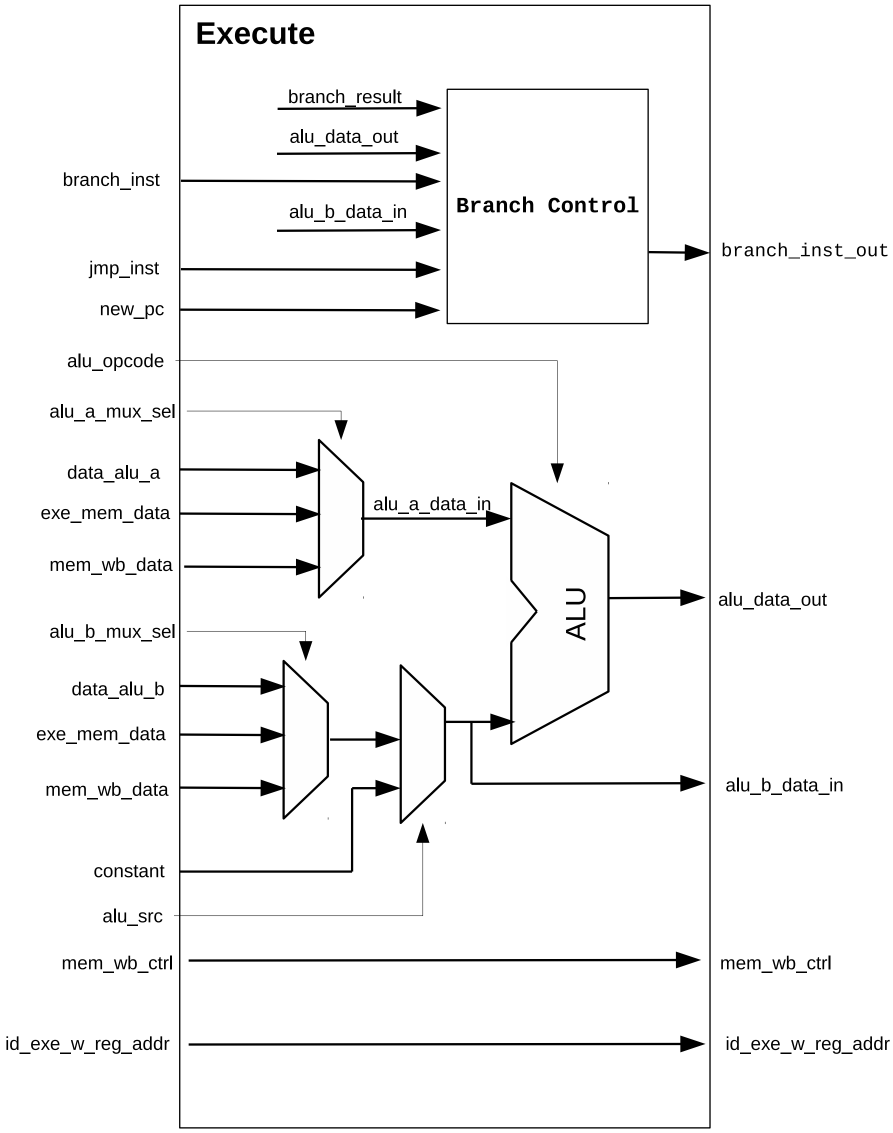
| Name | Length | Direction | Description |
|---|---|---|---|
| clock_in | 1 | input | CPU core clock |
| reset_in | 1 | input | CPU core reset |
| alu_src_in | TBD | input | TBD |
| alu_opcode_in | 3 | input | ALU opperation code |
| data_alu_a_in | 32 | input | ALU input A data |
| data_alu_b_in | 32 | input | ALU input B data |
| constant_in | 32 | input | 32-bit Sign-extended constant |
| write_back_mux_sel_in | TBD | input | Write back mux select |

continued on next page

| continued from previous page | | | |
|---|---|---|---|
| **Name** | **Length** | **Direction** | **Description** |
| w_reg_wr_en_in | 1 | input | GPR bank write enable |
| new_pc_in | 20 | input | Updated value of PC |
| mem_data_wr_en_in | TBD | input | TBD |
| w_reg_addr_in | TBD | input | GPR bank destiny address |
| branch_inst_in | TBD | input | TBD |
| jmp_inst_in | TBD | input | TBD |
| exe_mem_data_in | TBD | input | TBD |
| mem_wb_data_in | TBD | input | TBD |
| mem_data_wr_en_out | 1 | output | TBD |
| alu_b_data_in_out | 32 | output | ALU input B data |
| alu_data_out | 32 | output | ALU data output |
| write_back_mux_sel_out | TBD | output | Write back mux select |
| w_reg_wr_en_out | 1 | output | GPR bank write enable |
| w_reg_addr_out | 4 | output | GPR bank destiny address |
| branch_inst_out | 1 | output | Branch result after flag check |
| alu_zero_out | 1 | output | TBD |
| jmp_inst_out | 1 | output | TBD |

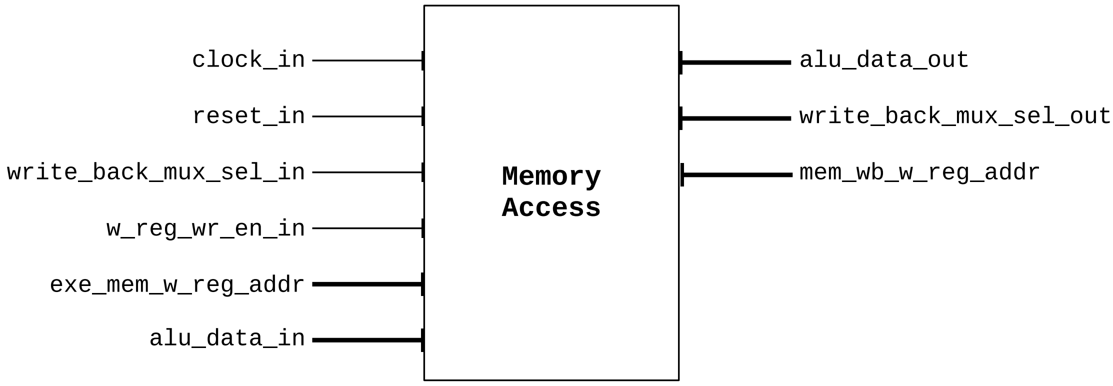
#### 4.3.3. *Internal Datapath*

The internal data path is composed by the following components.


Execute
branch_result
alu_data_out
branch_inst
alu_b_data_in
Branch Control
branch_inst_out
jmp_inst
new_pc
alu_opcode
alu_a_mux_sel
data_alu_a
exe_mem_data
mem_wb_data
alu_a_data_in
ALU
alu_data_out
alu_b_mux_sel
data_alu_b
exe_mem_data
mem_wb_data
alu_b_data_in
constant
alu_src
mem_wb_ctrl
mem_wb_ctrl
id_exe_w_reg_addr
id_exe_w_reg_addr

## 4.4. Memory Access

### 4.4.1. Block Diagram

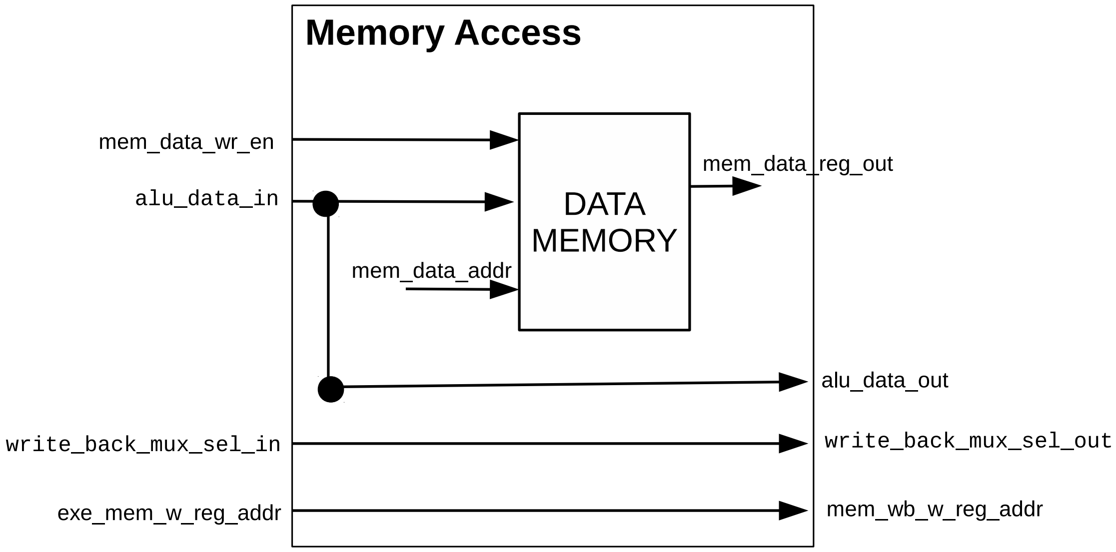



### 4.4.2. Pin/Port Definitions

| Name | Length | Direction | Description |
|---|---|---|---|
| clock_in | 1 | input | CPU core clock |
| reset_in | 1 | input | CPU core reset |
| write_back_mux_sel_in | 1 | input | Write back mux select |
| w_reg_wr_en_in | 1 | input | GPR bank write enable signal |
| exe_mem_w_reg_addr | TBD | input | GPR bank destiny address |
| alu_data_in | 32 | input | ALU data output |
| alu_data_out | 32 | output | ALU data output |
| write_back_mux_sel_out | TBD | output | Write back mux select |
| mem_wb_w_reg_addr | TBD | output | GPR bank destiny address |

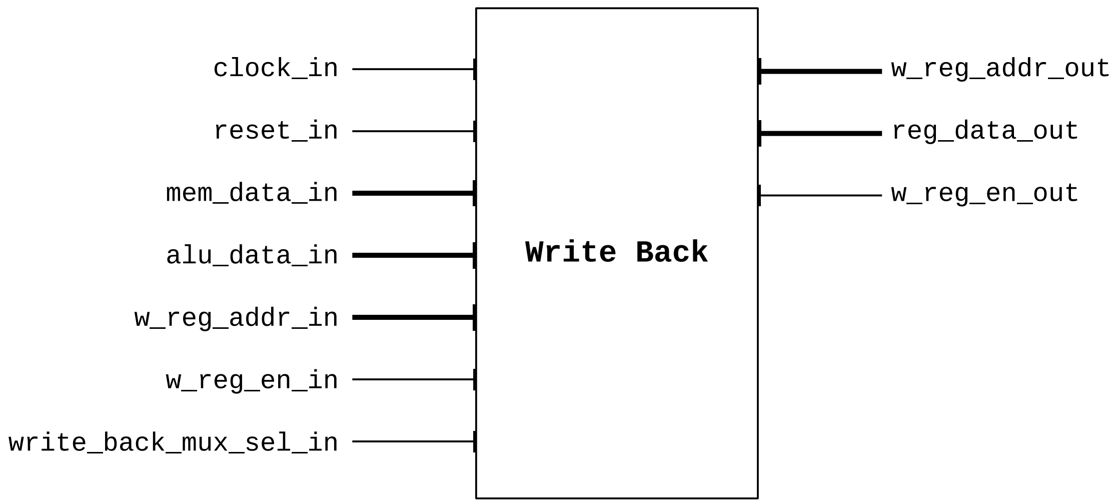
### 4.4.3. Internal Datapath

The internal data path is composed by the following components.


Memory Access
mem_data_wr_en
alu_data_in
mem_data_addr
DATA MEMORY
mem_data_reg_out
alu_data_out
write_back_mux_sel_in
write_back_mux_sel_out
exe_mem_w_reg_addr
mem_wb_w_reg_addr

## 4.5. Write Back

### 4.5.1. *Block Diagram*

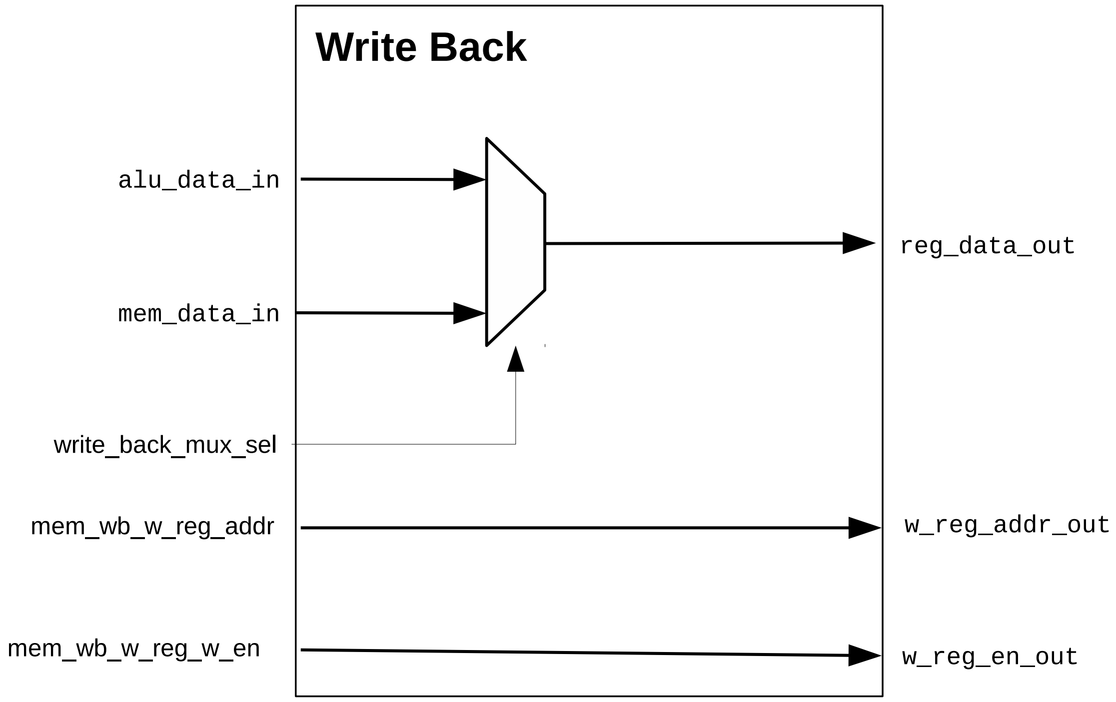



### 4.5.2. *Pin/Port Definitions*

| Name | Length | Direction | Description |
|---|---|---|---|
| clock_in | 1 | input | CPU core clock |
| reset_in | 1 | input | CPU core reset |
| mem_data_in | 32 | input | SDRAM data output |
| alu_data_in | 32 | input | ALU data output |
| w_reg_en_in | 4 | input | GPR bank write enable signal |
| w_reg_addr_in | 1 | input | GPR bank destiny address |
| write_back_mux_sel | TBD | input | Write back mux select |
| w_reg_addr_out | 4 | output | GPR bank destiny address |
| reg_data_out | 32 | output | GPR bank write data |
| w_reg_en_out | 1 | output | GPR bank write enable signal |

### 4.5.3. *Internal Datapath*

The internal data path is composed by the following components.


Write Back
alu_data_in
mem_data_in
write_back_mux_sel
mem_wb_w_reg_addr
mem_wb_w_reg_w_en
reg_data_out
w_reg_addr_out
w_reg_en_out

## 4.6. Pipeline Register Description

### 4.6.1. *Instruction Fetch/Instruction Decode*

| Name | Length | Description |
|---|---|---|
| new_pc | 20 | Stores the next program counter value. |
| instruction_reg | 32 | Stores the intruction word. |
| bpb_branch_taken | 1 | Stores BPB result. |

### 4.6.2. *Instruction Decode/Execute*

| Name | Length | Description |
|---|---|---|
| new_pc | 20 | Stores the next program counter value. |
| data_alu_reg_a | 32 | Stores the value of ALU input port A. |
| data_alu_reg_b | 32 | Stores the value of ALU input port B. |
| constant | 32 | Stores the signed extended integer constant. |
| instruction_reg | 32 | Stores the intruction word. |
| select_rd_reg | 1 | TBD |
| reg_file_w_en_reg | 1 | Stores the signal to enable GPR write back. |
| write_back_mux_sel_reg | TBD | Stores the select signal for write back Multiplexer. |
| alu_opcode | 3 | Stores the ALU opperation code. |
| select_mux_alu_a | TBD | Stores the ALU input data select signal |
| select_mux_alu_b | TBD | Stores the ALU input data select signal |
| bpb_branch_taken | 1 | Stores BPB result. |

### 4.6.3. *Execute/Memory Access*

| Name | Length | Description |
|---|---|---|
| instruction_reg | 32 | Stores the intruction word. |
| select_rd_reg | 1 | TBD |
| reg_file_w_en_reg | 1 | Stores the signal to enable GPR write back. |
| write_back_mux_sel_reg | TBD | Stores the select signal for write back Multiplexer. |

continued on next page

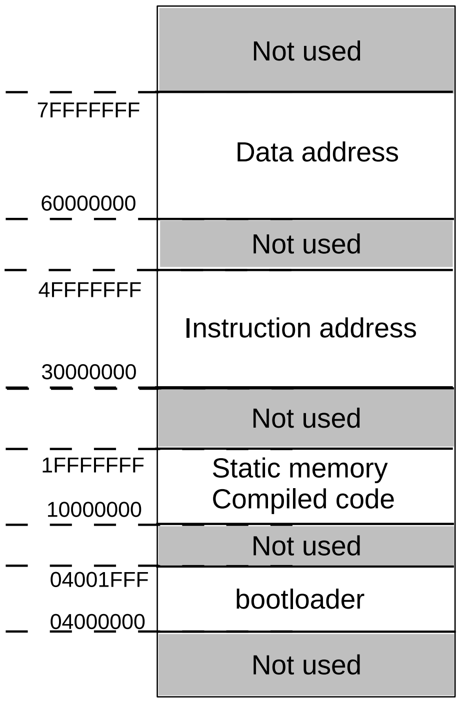
| continued from previous page | | |
|---|---|---|
| **Name** | **Length** | **Description** |
| data_alu_a | 32 | Stores the ALU input data A for memory addressing. |
| alu_data_reg | 32 | Stores the ALU output data. |

### *4.6.4. Memory Access/Write Back*

| Name | Length | Description |
|---|---|---|
| instruction_reg | 32 | Stores the intruction word. |
| select_rd_reg | 1 | TBD |
| reg_file_w_en_reg | 1 | Stores the signal to enable GPR write back. |
| write_back_mux_sel_reg | TBD | Stores the select signal for write back Multiplexer. |
| mem_data_reg | 32 | Stores the memory output data. |
| alu_data_reg | 32 | Stores the ALU output data. |
| w_reg_addr_reg | 4 | Stores the GPR data write address. |

## 4.7. Memory and Device Interface

The memory and device interface is responsible for memory mapping to the processor. Depending on the address accessed a specific memory will be accessed by the memory device interface. If it is a read operation in the next clock cycle the read data will be directed to the DLX input read data port.

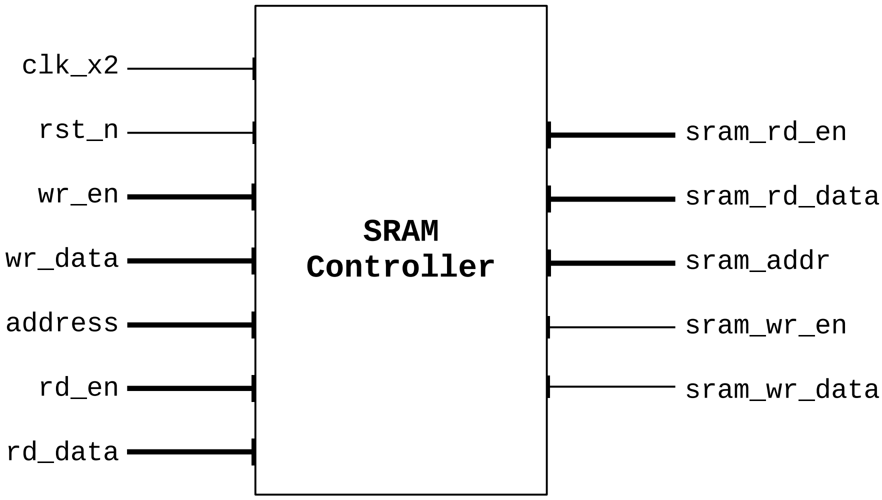



The memory map shown was created with separate address to show that devices have no relation between them and to show that the memory used in this project will be a small fraction for the 4GBytes possible.

## 4.8. SRAM Controller

The SRAM will store the instructions, as the SRAM has only 16 bits of word and an instruction has 32 bits the read and write operation must be done in a faster clock domain to avoid stopping the microprocessor. The SRAM has instruction code this way this memory is not written in normal operation. The SRAM will be written by the processor just during the bootloader code is running and during this operation no read will be done. Not having read and write simultaneously make easier the control not needing to deal with write and read operations at the same time.

### 4.8.1. *Block Diagram*




### 4.8.2. *Pin/Port Definitions*

| Name | Length | Direction | Description |
|---|---|---|---|
| clk_x2 | 1 | input | SRAM clock that is twice the DLX clock |
| rst_n | 1 | input | System asynchronous reset |
| wr_en | 1 | input | Data write enable |
| wr_data | 32 | input | Data to be written in the memory |
| address | 24 | input | Memory address from the DLX |
| rd_en | 1 | input | Data read enable |
| rd_data | 32 | output | Read data from SRAM to the DLX |
| sram_rd_en | 1 | output | TBD |
| sram_rd_data | 16 | output | TBD |
| sram_addr | 10 | output | Memory address that goes to the SRAM |
| sram_wr_en | 1 | output | TBD |

continued on next page

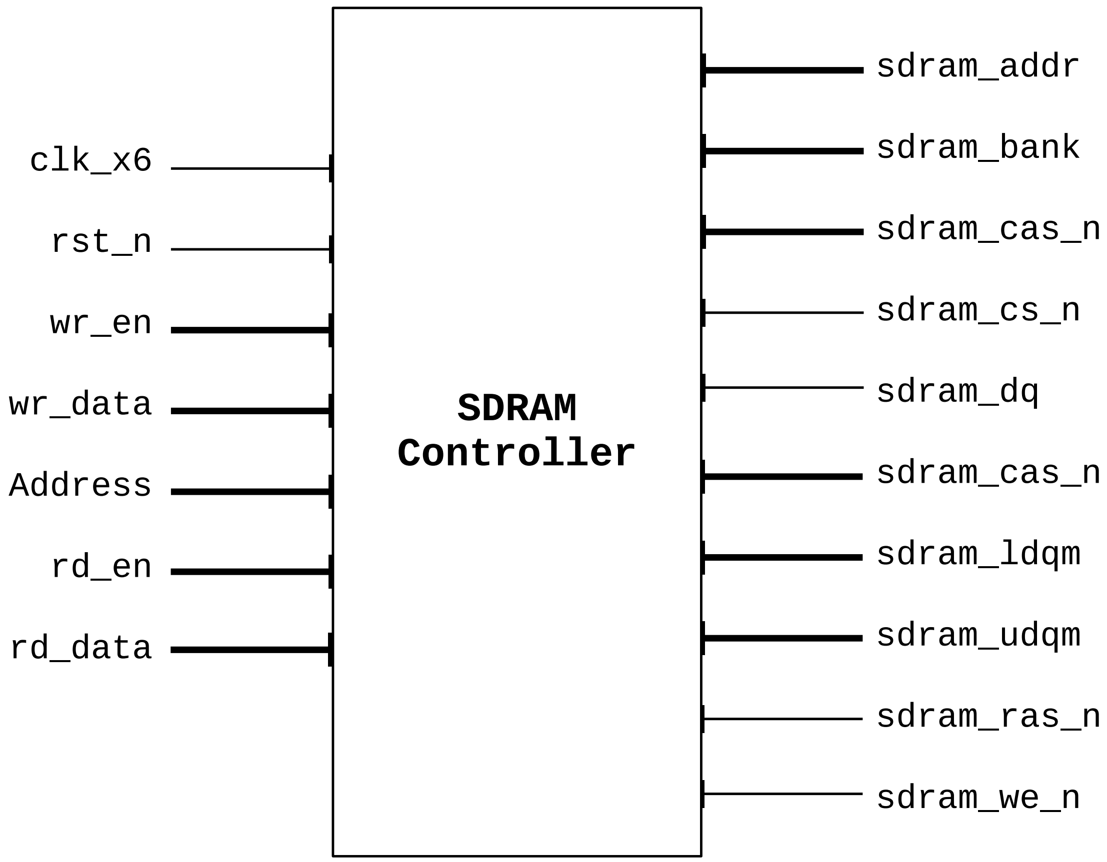
| continued from previous page | | | |
|---|---|---|---|
| **Name** | **Length** | **Direction** | **Description** |
| sram_wr_data | 16 | output | TBD |

As the SRAM has half word the controller works with a clock twice faster enabling read two address, concatenating and sending to the DLX in time.

## 4.9. SDRAM Controller

The SDRAM controller will be very simple and optimized to read or write one word that will be requested by the processor. There will be no burst, just one word write or read operation.

### 4.9.1. Block Diagram




### 4.9.2. Pin/Port Definitions

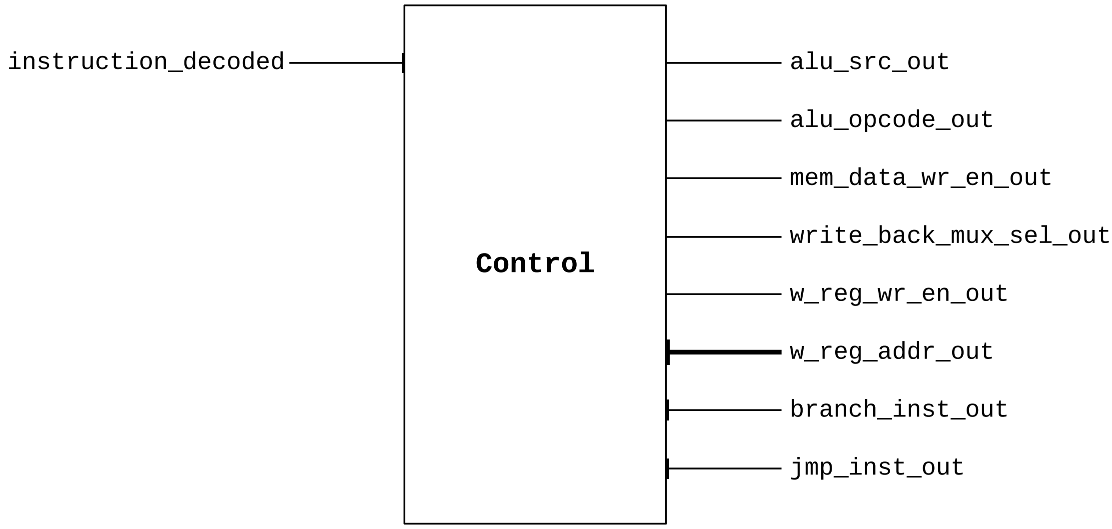
| Name | Length | Direction | Description |
|---|---|---|---|
| clk_x6 | 1 | input | SDRAM controller input clock. 6 times faster than DLX clock |
| rst_n | 1 | input | System asynchronous reset |
| wr_en | 1 | input | Data write enable |
| wr_data_in | 32 | input | Data to be written in the memory |
| address | 24 | input | Address of write or read operation |
| rd_en | 1 | input | Data read enable |

continued on next page

| continued from previous page | | | |
|---|---|---|---|
| **Name** | **Length** | **Direction** | **Description** |
| rd_data | 32 | output | Data read from the SDRAM to the memory interface |
| sdram_addr | 12 | output | SDRAM data address |
| sdram_bank | 2 | output | The SDRAM selected Bank |
| sdram_cas_n | 1 | output | SDRAM CAS |
| sdram_cs_n | 1 | output | SDRAM chip select |
| sdram_dq | 32 | output | SDRAM read data, 2 bus of 16 bits, one for each memory |
| sdram_ldqm | 1 | output | TBD |
| sdram_udqm | 1 | output | TBD |
| sdram_ras_n | 1 | output | SRAM RAS |
| sdram_we_n | 1 | output | SDRAM write enable |

## 4.10. Control Micro-instructions Description

### 4.10.1. Block Diagram

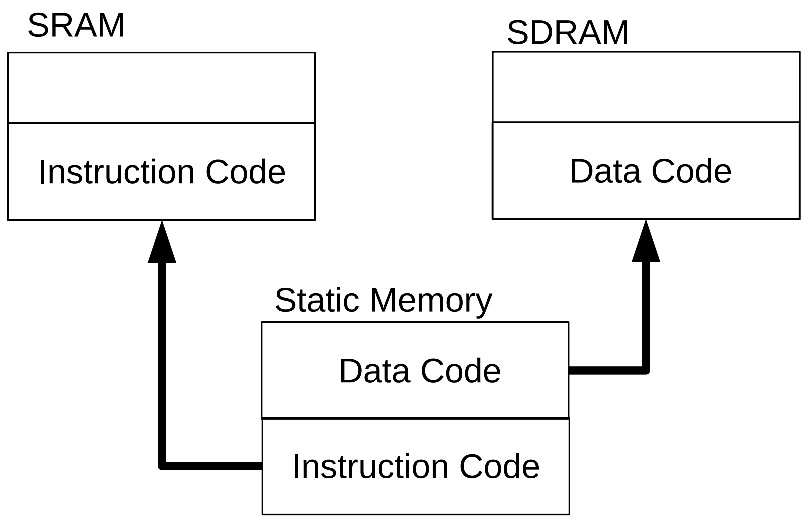



### 4.10.2. Pin/Port Definitions

| Name | Length | Direction | Description |
|---|---|---|---|
| instruction_decoded | 1 | input | TBD |
| alu_src_out_n | 1 | output | TBD |
| alu_opcode_out | 1 | output | TBD |
| mem_data_wr_en_out | 1 | output | TBD |
| write_back_mux_sel_out | 1 | output | TBD |
| w_reg_wr_en_out | 1 | output | TBD |
| w_reg_addr_out | 1 | output | TBD |
| branch_inst_out | 12 | output | TBD |
| jmp_inst_out | 1 | output | TBD |

### 4.11. Bootloader

The bootloader is the ROM memory has the code responsible to initialize the processor and write the code that will be executed in the correct address and consequently each memory device. The ROM must have the DLX reset PC register address. When the system is reseted the processor will read instructions from the ROM address. The code inside the ROM will copy the instruction and data code from a static memory to the instruction memory SRAM and the data memory SDRAM.




Usually data in code is read-only data and it is common in compiled C code. Probably there will be only instruction code to be loaded during bootloader operation.