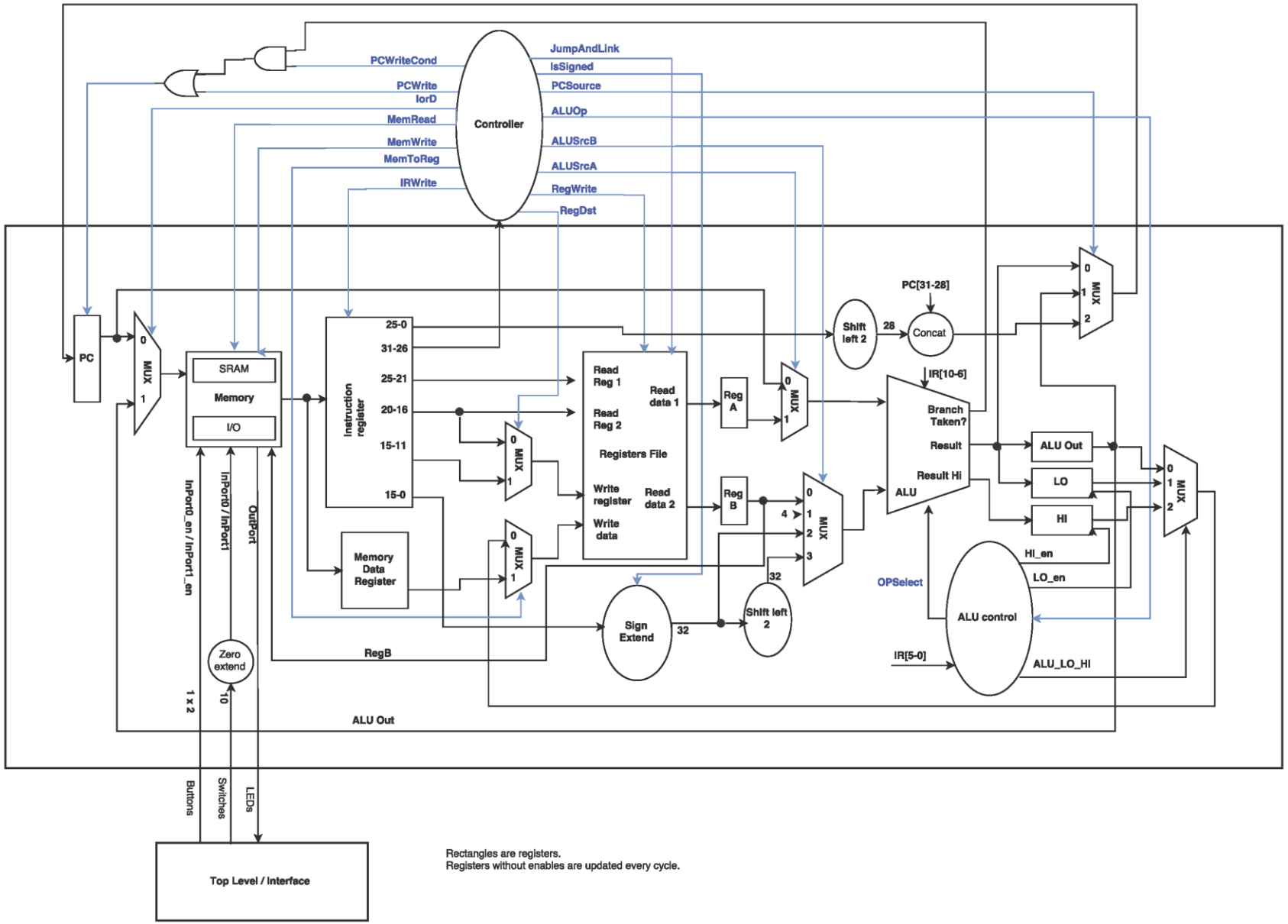
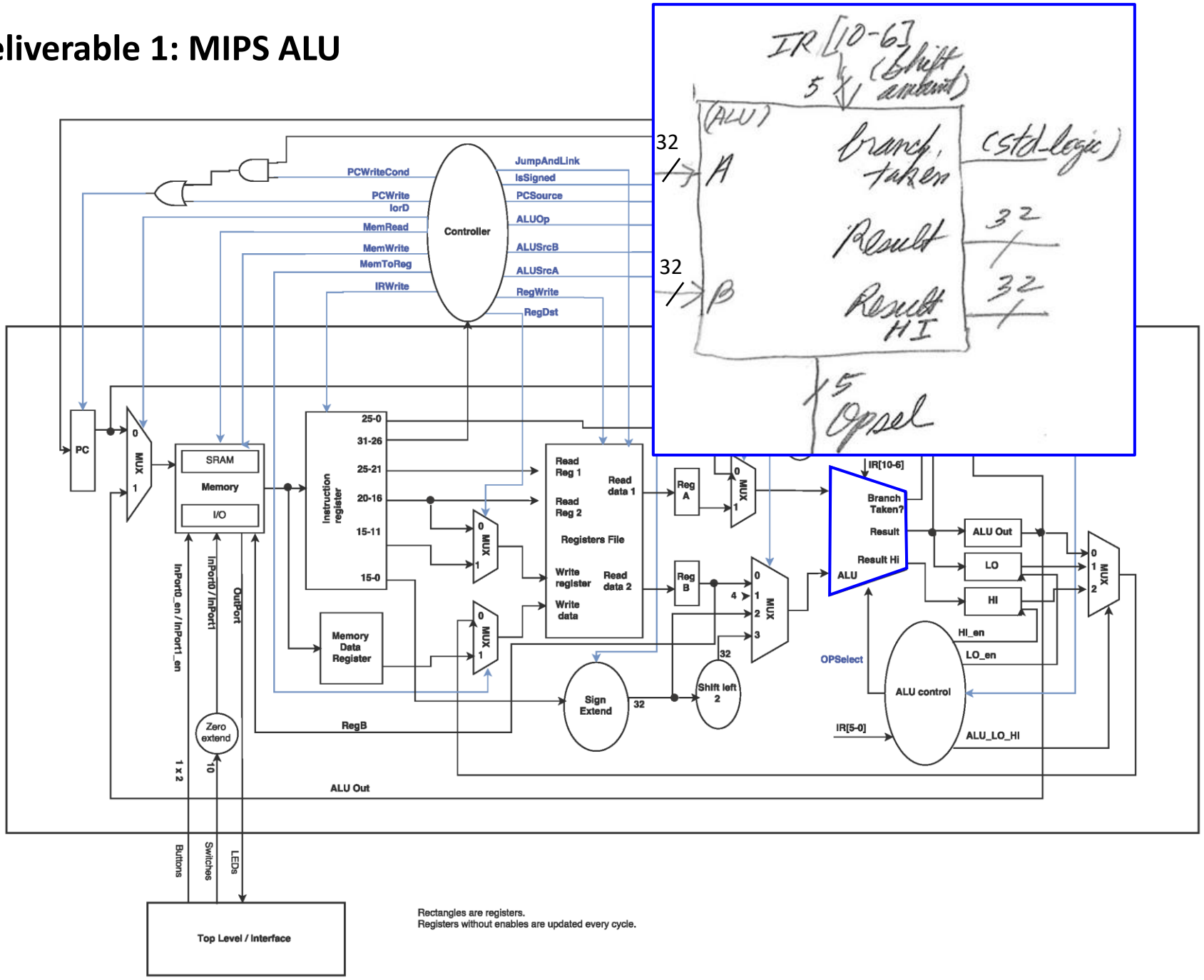
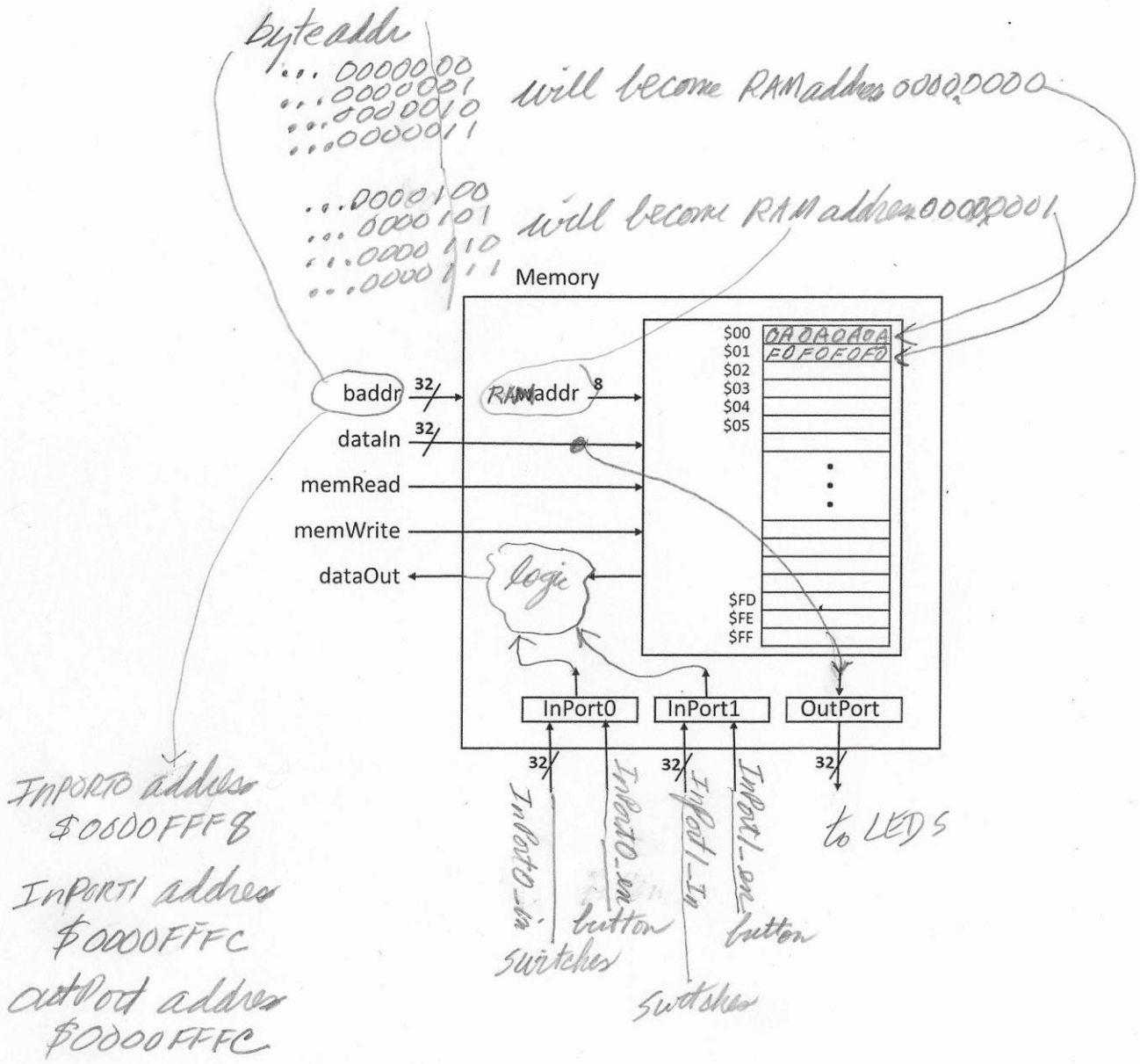
MIPS-4712 General Architecture



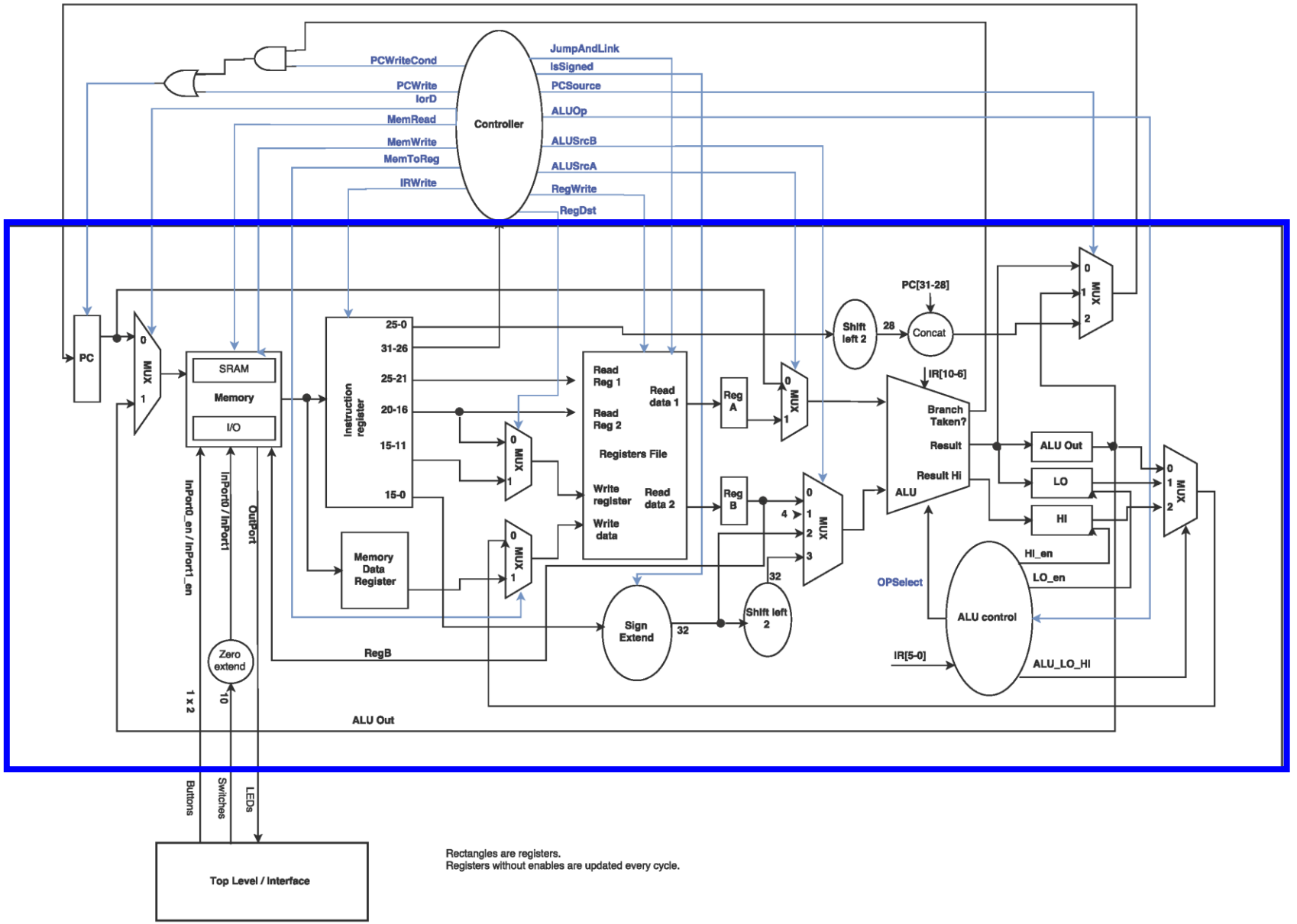
Deliverable 1: MIPS ALU



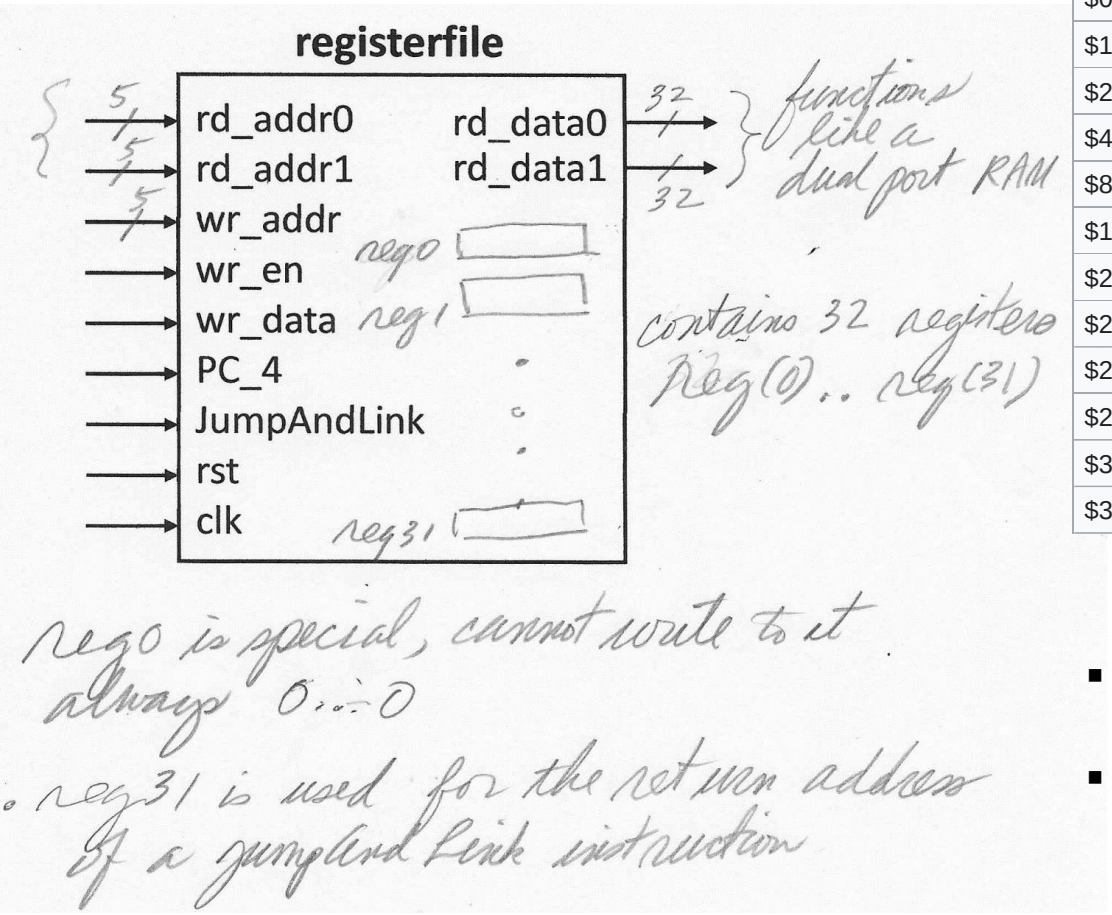
Deliverable 2: MIPS-4712 Memory/Port Module



Deliverable 3: Datapath



MIPS-4712 Register File



Number	Name	Comments
$0	$zero, $r0	Always zero
$1	$at	Reserved for assembler
$2, $3	$v0, $v1	First and second return values, respectively
$4, ..., $7	$a0, ..., $a3	First four arguments to functions
$8, ..., $15	$t0, ..., $t7	Temporary registers
$16, ..., $23	$s0, ..., $s7	Saved registers
$24, $25	$t8, $t9	More temporary registers
$26, $27	$k0, $k1	Reserved for kernel (operating system)
$28	$gp	Global pointer
$29	$sp	Stack pointer
$30	$fp	Frame pointer
$31	$ra	Return address

- **Name** is the symbolic name used in the MIP assembly language instruction.
- **Number** is the actual register number.

Example:

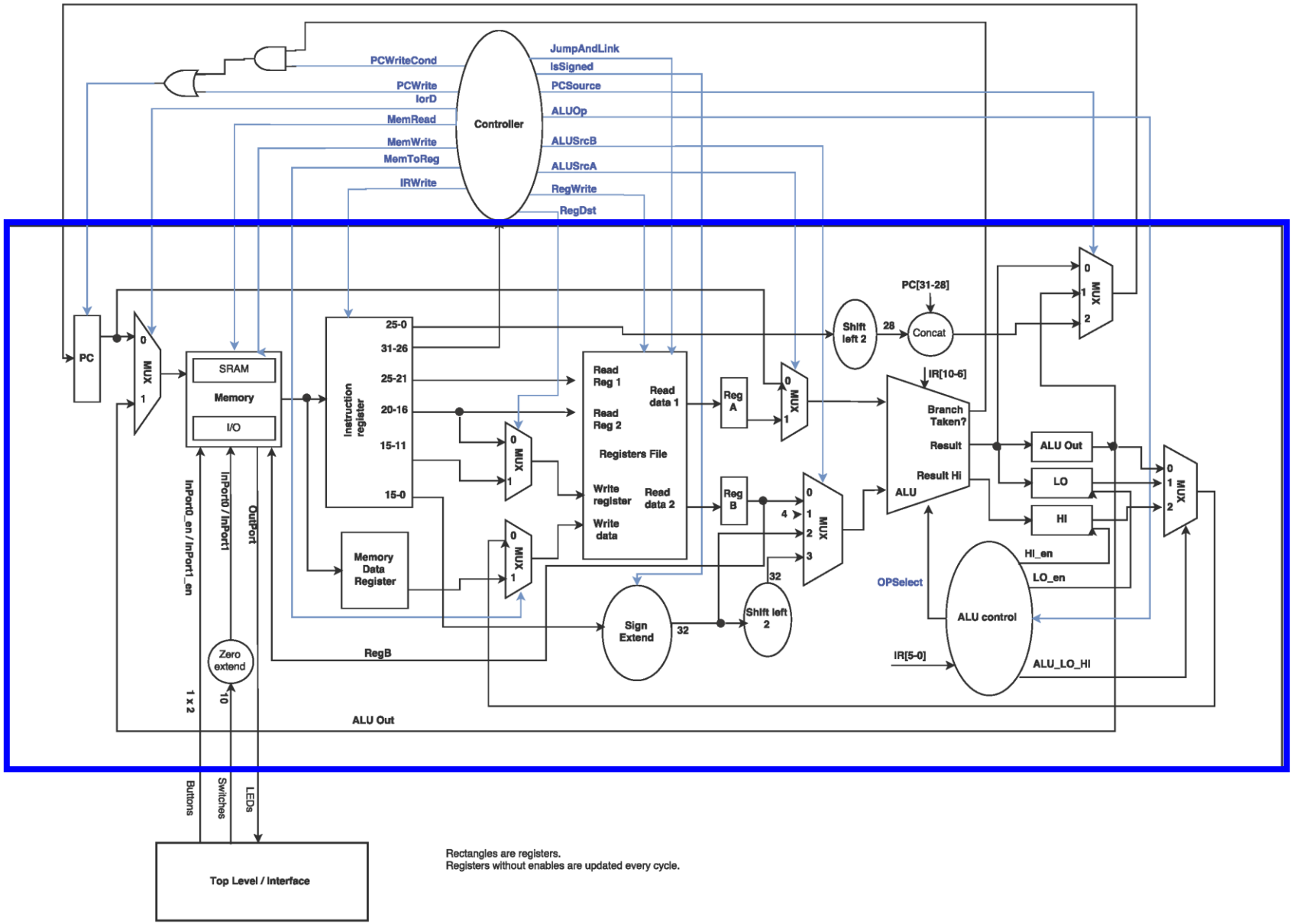
addu $s3, $s1, $s2

means (reg19) = (reg17) + (reg18)

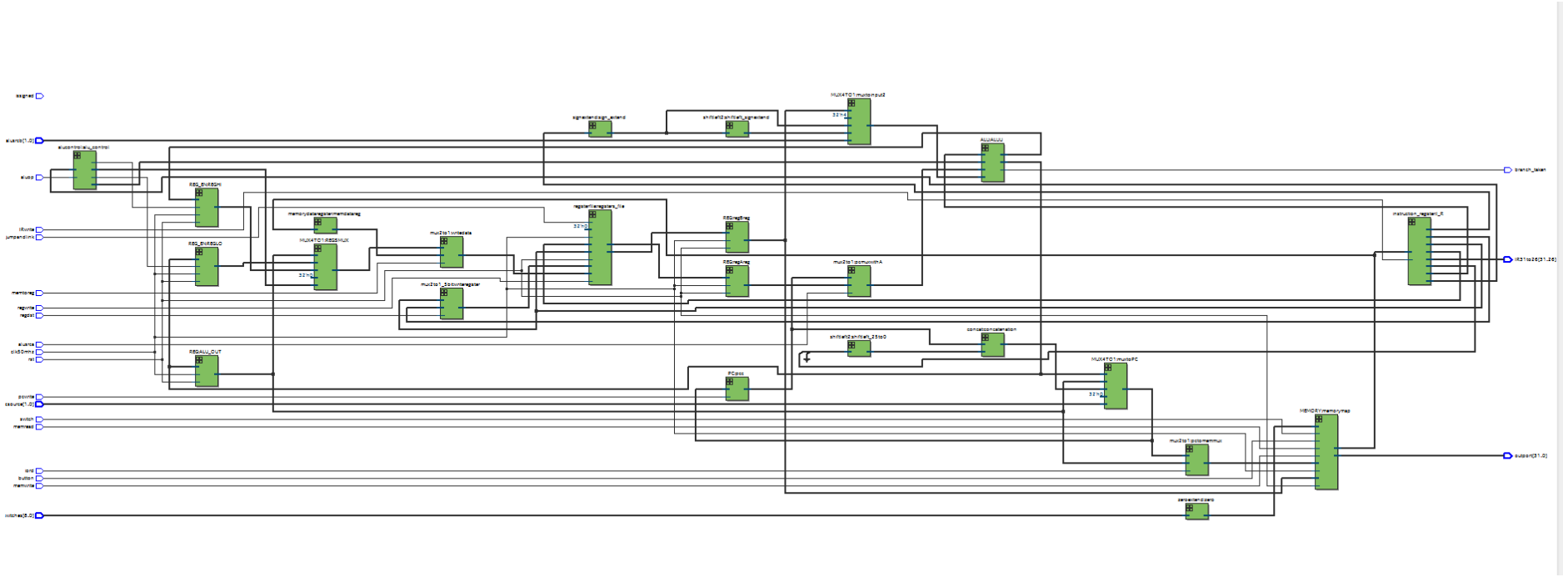
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity registerfile is
   port(
      clk : in std_logic;
      rst : in std_logic;

      rd_addr0 : in std_logic_vector(4 downto 0); --read reg 1
      rd_addr1 : in std_logic_vector(4 downto 0); --read reg 2

      wr_addr : in std_logic_vector(4 downto 0); --write register
      wr_en : in std_logic;
      wr_data : in std_logic_vector(31 downto 0); --write data

      rd_data0 : out std_logic_vector(31 downto 0); --read data 1
      rd_data1 : out std_logic_vector(31 downto 0); --read data 2
            --JAL
            PC_4 : in std_logic_vector(31 downto 0);
            JumpAndLink : in std_logic
      );
end registerfile;
```

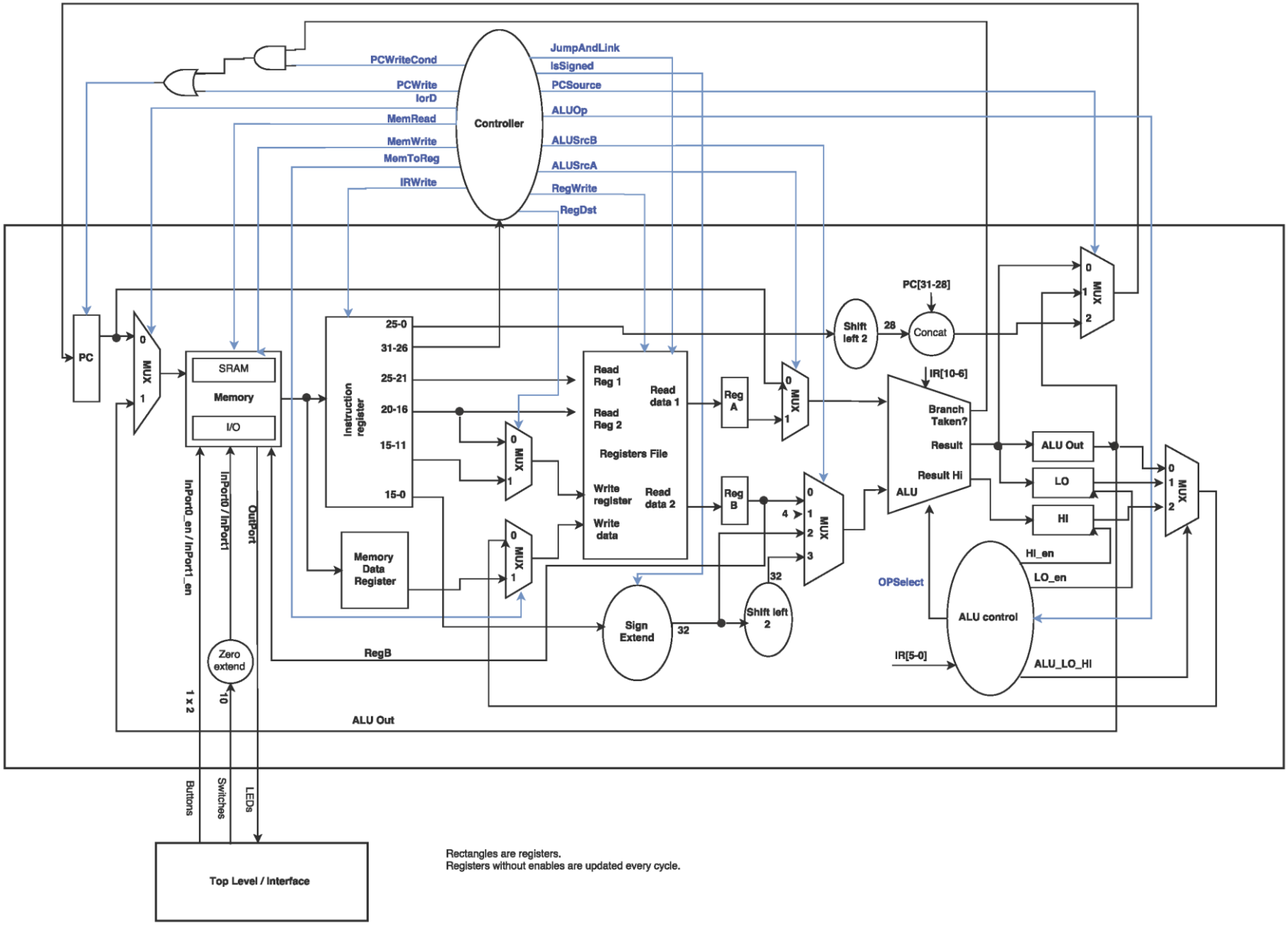
```
architecture sync_read of registerfile is
   type reg_array is array(0 to 31) of std_logic_vector(31 downto 0);
   signal regs : reg_array;
begin
   process (clk, rst) is
   begin
      if (rst = '1') then
         for i in regs'range loop
            regs(i) <= (others => '0');
         end loop;
      elsif (rising_edge(clk)) then
         if (wr_en = '1') then
            regs(to_integer(unsigned(wr_addr))) <= wr_data;
                                    regs(0) <= (others => '0');
         end if;
         if(JumpAndLink = '1') then
             regs(31) <= PC_4;
         end if;
         rd_data0 <= regs(to_integer(unsigned(rd_addr0)));
         rd_data1 <= regs(to_integer(unsigned(rd_addr1)));
      end if;
   end process;
end sync_read;
```

Deliverable 3: Datapath

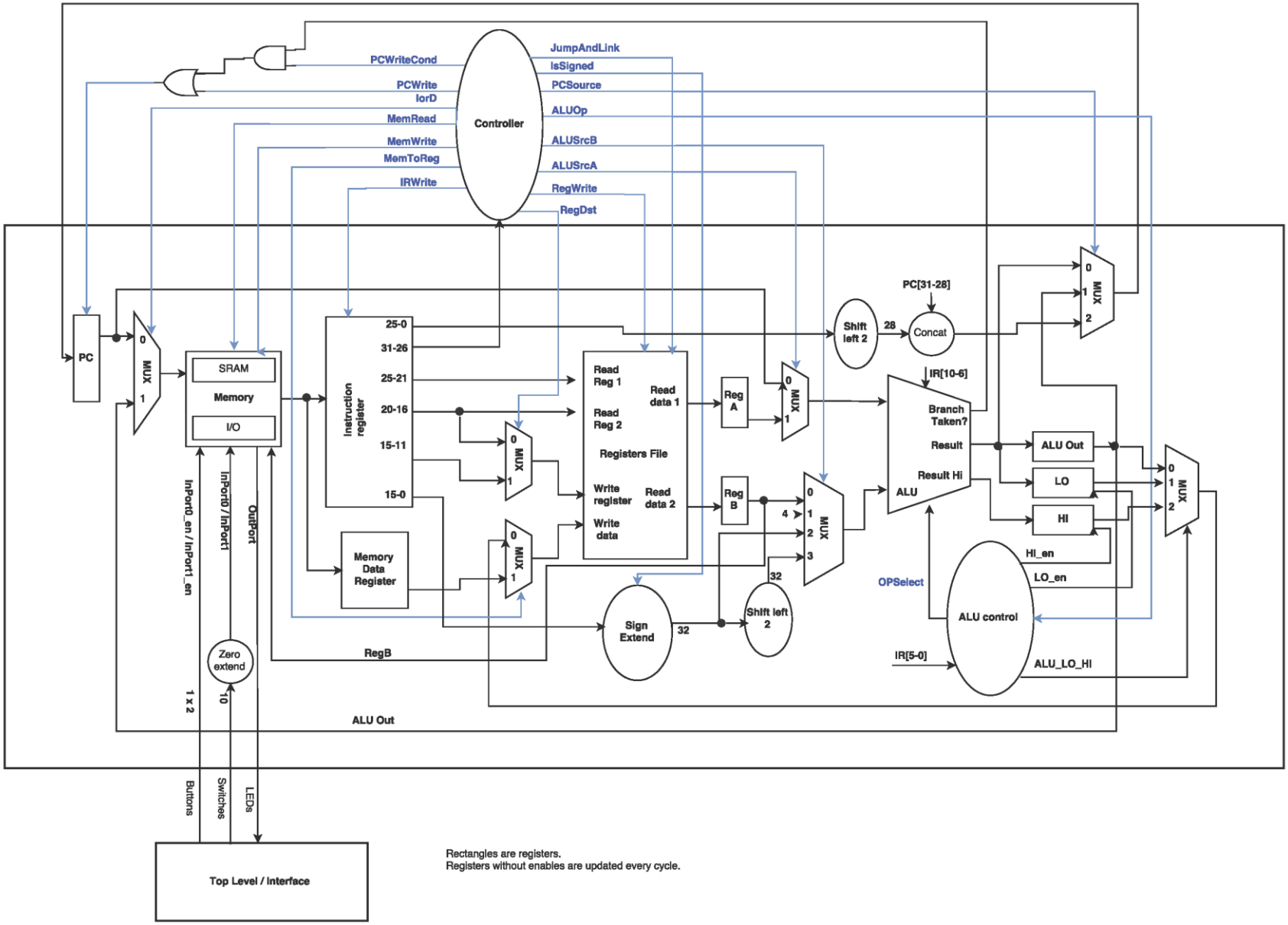


Deliverable 3: RTL View of Datapath

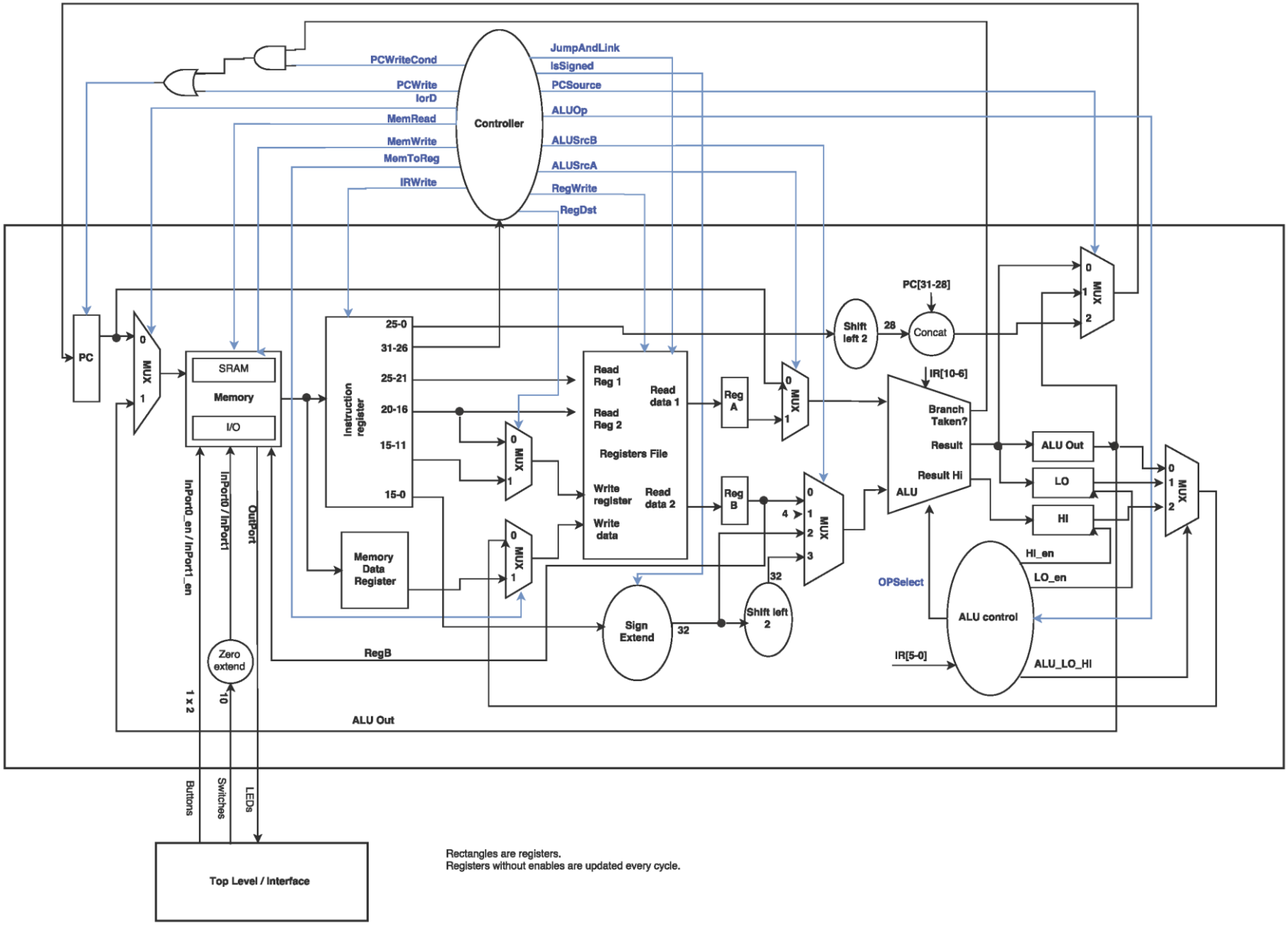




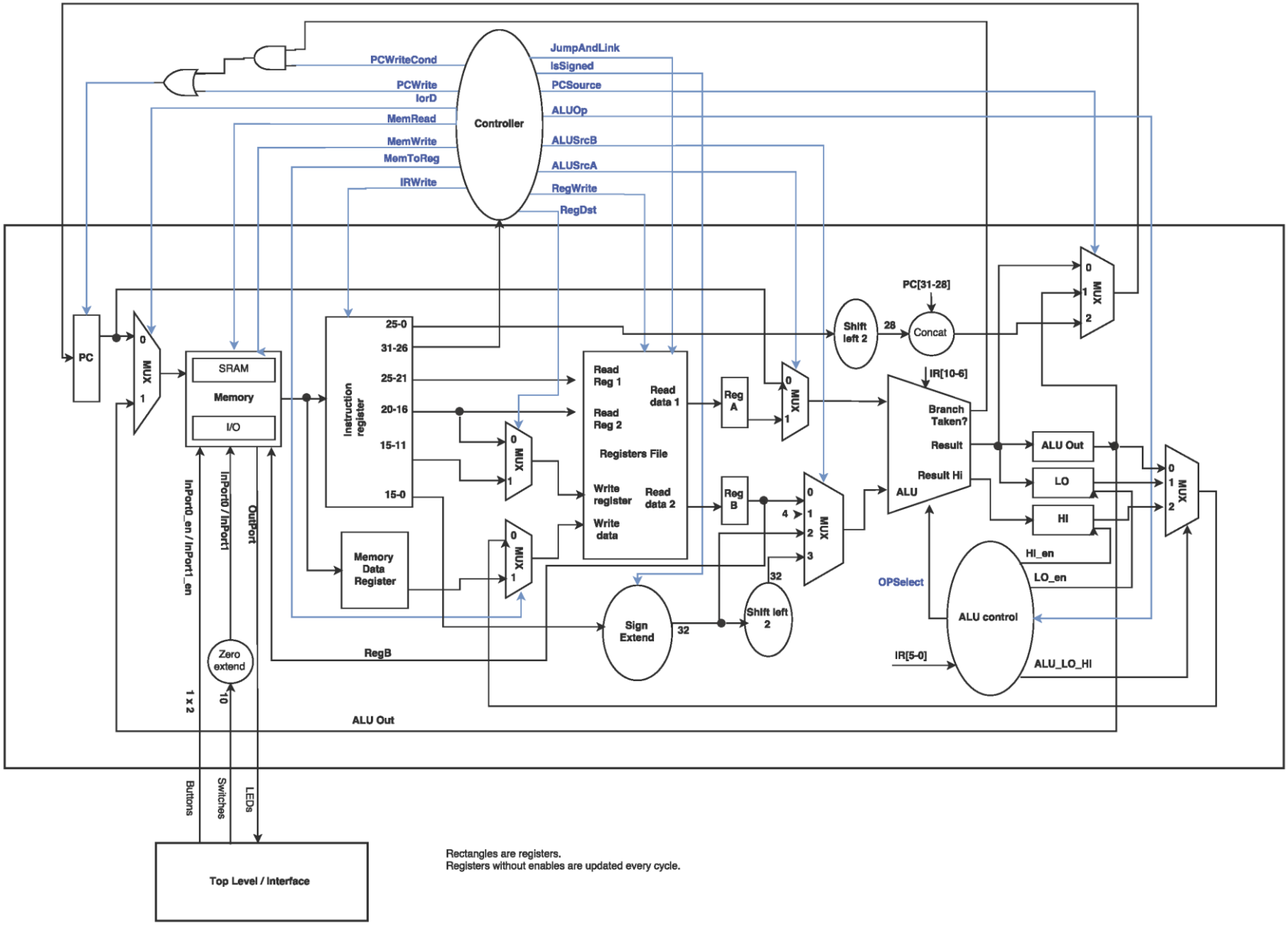
Rectangles are registers.
Registers without enables are updated every cycle.



Rectangles are registers.
Registers without enables are updated every cycle.



Rectangles are registers.
Registers without enables are updated every cycle.



Rectangles are registers.
Registers without enables are updated every cycle.