EN3021 Group Project - Progress as of OCT17

210415N Navarathne D.M.G.B.
210594J Senevirathne I.U.B.
210609M Silva M.K.Y.U.N.

October 17, 2024

Current status: 45% Complete

Progress this week

We started the following parts of the project for the week OCT10 - OCT17:

- Almost completed the integration of the modules into the processor
- Following modules were completed
  - ALU control
  - Immediate generator

ALU Control

The ALU control was made to get the input from the control signal ALUOp, and func3 and func7 from the instruction to control the ALU. Further this passes on a branch instruction type to the ALU to get a branch flag according to the type.

Table 1: Instruction Set and corresponding ALU operation

Inst.	**Type**	**ALUOp**	**F3**	**F7**	**OP**
ADD	R	10	000	0	ADD
SUB	R	10	000	1	SUB
SLL	R	10	001	0	SLL
SLT	R	10	010	0	SLT
SLTU	R	10	011	0	SLTU
XOR	R	10	100	0	XOR
SRL	R	10	101	0	SRL
SRA	R	10	101	1	SRA
OR	R	10	110	0	OR
AND	R	10	111	0	AND
ADDI	I	10	000	X	ADD
SLTI	I	10	010	X	SLT
SLTIU	I	10	011	X	SLTU
XORI	I	10	100	X	XOR
ORI	I	10	110	X	OR
ANDI	I	10	111	X	AND

Inst.	**Type**	**ALUOp**	**F3**	**F7**	**OP**
SLLI	I	10	001	X	SLL
SRLI	I	10	101	X	SRL
SRAI	I	10	101	X	SRA
JALR	I	10	000	X	ADD
LUI	U	XX	XXX	X	-
AUIPC	U	XX	XXX	X	-
JAL	UJ	XX	XXX	X	-
SB	S	00	000	X	ADD
SH	S	00	001	X	ADD
SW	S	00	010	X	ADD
LB	I	00	000	X	ADD
LH	I	00	001	X	ADD
LW	I	00	010	X	ADD
LBU	I	00	100	X	ADD
LHU	I	00	101	X	ADD
BEQ	SB	01	000	X	SUB
BNE	SB	01	001	X	SUB
BLT	SB	01	100	X	SLT
BGE	SB	01	101	X	SLT
BLTU	SB	01	110	X	SLTU
BGEU	SB	01	111	X	SLTU
FENCE	I	00	000	X	-
ECALL	I	00	000	X	-
EBREAK	I	00	000	X	-

```
//
//   +--------------------------------------------------+
//   |  Instruction set                                 |
//   +---------+--------+---------+------+-----+--------+
//   |  Inst.  |  Type  |  ALUOp  |  F3  |  F7 |  OP    |
//   +---------+--------+---------+------+-----+--------+
//   |  ADD    |   R    |   10    |  000 |   0 |  ADD   |
//   |  SUB    |   R    |   10    |  000 |   1 |  SUB   |
//   |  SLL    |   R    |   10    |  001 |   0 |  SLL   |
//   |  SLT    |   R    |   10    |  010 |   0 |  SLT   |
//   |  SLTU   |   R    |   10    |  011 |   0 |  SLTU  |
//   |  XOR    |   R    |   10    |  100 |   0 |  XOR   |
//   |  SRL    |   R    |   10    |  101 |   0 |  SRL   |
//   |  SRA    |   R    |   10    |  101 |   1 |  SRA   |
//   |  OR     |   R    |   10    |  110 |   0 |  OR    |
//   |  AND    |   R    |   10    |  111 |   0 |  AND   |
//   +---------+--------+---------+------+-----+--------+
//   |  ADDI   |   I    |   10    |  000 |   X |  ADD   |
//   |  SLTI   |   I    |   10    |  010 |   X |  SLT   |
//   |  SLTIU  |   I    |   10    |  011 |   X |  SLTU  |
//   |  XORI   |   I    |   10    |  100 |   X |  XOR   |
//   |  ORI    |   I    |   10    |  110 |   X |  OR    |
//   |  ANDI   |   I    |   10    |  111 |   X |  AND   |
//   |  SLLI   |   I    |   10    |  001 |   X |  SLL   |
//   |  SRLI   |   I    |   10    |  101 |   X |  SRL   |
//   |  SRAI   |   I    |   10    |  101 |   X |  SRA   |
//   |  JALR   |   I    |   10    |  000 |   X |  ADD   |
//   +---------+--------+---------+------+-----+--------+
//   |  LUI    |   U    |   XX    |  XXX |   X |  -     |
//   |  AUIPC  |   U    |   XX    |  XXX |   X |  -     |
//  |  JAL    |   UJ   |   XX    |  XXX |   X |  -     |
//  +---------+--------+---------+------+-----+--------+
//  |  SB     |   S    |   00    |  000 |   X |  ADD   |
//  |  SH     |   S    |   00    |  001 |   X |  ADD   |
//  |  SW     |   S    |   00    |  010 |   X |  ADD   |
//  +---------+--------+---------+------+-----+--------+
//  |  LB     |   I    |   00    |  000 |   X |  ADD   |
//  |  LH     |   I    |   00    |  001 |   X |  ADD   |
//  |  LW     |   I    |   00    |  010 |   X |  ADD   |
//  |  LBU    |   I    |   00    |  100 |   X |  ADD   |
//  |  LHU    |   I    |   00    |  101 |   X |  ADD   |
//  +---------+--------+---------+------+-----+--------+
//  |  BEQ    |   SB   |   01    |  000 |   X |  SUB   |
//  |  BNE    |   SB   |   01    |  001 |   X |  SUB   |
//  |  BLT    |   SB   |   01    |  100 |   X |  SLT   |
//  |  BGE    |   SB   |   01    |  101 |   X |  SLT   |
//  |  BLTU   |   SB   |   01    |  110 |   X |  SLTU  |
//  |  BGEU   |   SB   |   01    |  111 |   X |  SLTU  |
//  +---------+--------+---------+------+-----+--------+
//  |  FENCE  |   I    |   00    |  000 |   X |  -     |
//  |  ECALL  |   I    |   00    |  000 |   X |  -     |
//  |  EBREAK |   I    |   00    |  000 |   X |  -     |
//  +---------+--------+---------+------+-----+--------+


module alu_ctrl(
    input  [1:0] ALUOp,
    input  [2:0] FUNC3,
    input FUNC7,
    output reg [3:0] ALUCTRL,
    output reg [2:0] BRANCHCONDITION
  );


  parameter ADD = 4'b0000,  SUB = 4'b0001,  SLL = 4'b0010,
         SRL = 4'b0011,  SRA = 4'b0100,  AND = 4'b0101,
         OR  = 4'b0110,  XOR = 4'b0111,  SLT = 4'b1000,
         SLTU = 4'b1001;


  always@(*) begin
    if (ALUOp == 2'b10) begin // ----------------- R type / I type
      case (FUNC3)
        3'b000:
          begin
            if ((ALUOp == 2'b00)&(FUNC7 == 1'b1))
              ALUCTRL <= SUB;
            else
              ALUCTRL <= ADD;
          end

        3'b001:
          ALUCTRL <= SLL;

        3'b010:
          ALUCTRL <= SLT;

        3'b011:
          ALUCTRL <= SLTU;

        3'b100:
          ALUCTRL <= XOR;

        3'b101:
          begin
            if (FUNC7 == 1'b0)
              ALUCTRL <= SRL;
            else
```

```
              ALUCTRL <= SRA;
          end

        3'b110:
          ALUCTRL <= OR;

        3'b111:
          ALUCTRL <= AND;

        default:
          ALUCTRL <= 4'bx;

      endcase
    end else if (ALUOp == 2'b00) begin
      ALUCTRL <= ADD;

    end else if (ALUOp == 2'b01) begin
      case(FUNC3[2:1])
        2'b00:
          ALUCTRL <= SUB;

        2'b10:
          ALUCTRL <= SLT;

        2'b11:
          ALUCTRL <= SLTU;

        default:
          ALUCTRL <= 4'bx;

      endcase
    end else begin
      ALUCTRL <= 4'bx;

    end

    if (ALUOp == 2'b01) begin
      BRANCHCONDITION <= FUNC3;
    end else begin
      BRANCHCONDITION <= 3'bx;
    end


  end
endmodule
```

Adding Branch flags to the ALU

The ALU was modified to indicate the satisfaction of a branch condition, from the result that was calculated from the existing ALU Operations.

```
module alu(
    input   [31:0] A, B,
    input   [3:0]  CTRL,
    input   [2:0]  BRANCHCONDITION,
    output reg [31:0] OUT,
    output reg BRANCHFLAG
  );

  reg  [31:0] A_ALU, B_ALU;
  wire [31:0] SUM, BAND, BXOR;
  reg   CIN;

  // CLA
  cla32 cla32_0(.A(A_ALU), .B(B_ALU^{4{CIN}}), .CIN(CIN), .OF(), .SUM(SUM), .BAND(BAND)
    , .BXOR(BXOR));
```

```

parameter ADD = 4'b0000,  SUB = 4'b0001,  SLL = 4'b0010,
       SRL = 4'b0011,  SRA = 4'b0100,  AND = 4'b0101,
       OR  = 4'b0110,  XOR = 4'b0111,  SLT = 4'b1000,
       SLTU = 4'b1001;

parameter BEQ = 3'b000, BNE = 3'b001, BLT = 3'b100,
       BGE = 3'b101, BLTU= 3'b110, BGEU= 3'b111;


always@(*) begin
  case (CTRL)
    ADD:
      begin
        CIN <= 1'b0;
        A_ALU <= A;
        B_ALU <= B;
        OUT <= SUM;
      end

    SUB:
      begin
        CIN <= 1'b1;
        A_ALU <= A;
        B_ALU <= B;
        OUT <= SUM;
      end

    AND:
      begin
        CIN <= 1'bx;
        A_ALU <= A;
        B_ALU <= B;
        OUT <= BAND;
      end

    OR:
      begin
        CIN <= 1'bx;
        A_ALU <= A;
        B_ALU <= B;
        OUT <= A|B;
      end

    XOR:
      begin
        CIN <= 1'bx;
        A_ALU <= A;
        B_ALU <= B;
        OUT <= BXOR;
      end

    SLL:
      begin
        CIN <= 1'bx;
        A_ALU <= 32'bx;
        B_ALU <= 32'bx;
        OUT <= A << B;
      end

    SRL:
      begin
        CIN <= 1'bx;
        A_ALU <= 32'bx;
        B_ALU <= 32'bx;
        OUT <= A >> B;
      end

```

```verilog
      SRA:
        begin
          CIN <= 1'bx;
          A_ALU <= 32'bx;
          B_ALU <= 32'bx;
          OUT <= A >>> B;
        end

      SLT:
        begin
          CIN <= 1'b1;
          A_ALU <= A;
          B_ALU <= B;
          OUT <= {31'b0, SUM[31]};
        end

      SLTU:
        begin
          CIN <= 1'bx;
          A_ALU <= 32'bx;
          B_ALU <= 32'bx;
          OUT <= (A < B) ? 32'b1 : 32'b0;
        end

      default:
        begin
          A_ALU <= 32'bx;
          B_ALU <= 32'bx;
          CIN <= 1'bx;
          OUT <= 32'bx;
        end
    endcase


    case(BRANCHCONDITION)
      BEQ:
        BRANCHFLAG <= |SUM;

      BNE:
        BRANCHFLAG <= ~|SUM;

      BLT:
        BRANCHFLAG <= OUT[0];

      BGE:
        BRANCHFLAG <= ~OUT[0];

      BLTU:
        BRANCHFLAG <= OUT[0];

      BGEU:
        BRANCHFLAG <= ~OUT[0];

      default:
        BRANCHFLAG <= 1'bx;
    endcase
  end

endmodule
```

Immediate Generator

The immediate generator formats the immediate into the proper form by using the instruction.

Table 2: Instruction Set and Immediate format

Inst.	**Type**	**ALUOp**	**F3**	**F7**	**Immediate Location**	**Format**
ADDI	I	10	000	X	[31:20]	[11:0]
SLTI	I	10	010	X	[31:20]	[11:0]
SLTIU	I	10	011	X	[31:20]	[11:0]
XORI	I	10	100	X	[31:20]	[11:0]
ORI	I	10	110	X	[31:20]	[11:0]
ANDI	I	10	111	X	[31:20]	[11:0]
SLLI	I	10	001	X	[24:20] (no ext)	[4:0]
SRLI	I	10	101	X	[24:20] (no ext)	[4:0]
SRAI	I	10	101	X	[24:20] (no ext)	[4:0]
JALR	I	10	000	X	[31:20]	[11:0]
LUI	U	XX	XXX	X	[31:12] (up)	[31:12]
AUIPC	U	XX	XXX	X	[31:12] (up)	[31:12]
JAL	UJ	XX	XXX	X	[31:12] (up)	[20][10:1][11][19:12]
SB	S	00 (?)	000	X	[31:25][11:7]	[11:5][4:0]
SH	S	00 (?)	001	X	[31:25][11:7]	[11:5][4:0]
SW	S	00 (?)	010	X	[31:25][11:7]	[11:5][4:0]
LB	I	00 (?)	000	X	[31:20]	[11:0]
LH	I	00 (?)	001	X	[31:20]	[11:0]
LW	I	00 (?)	010	X	[31:20]	[11:0]
LBU	I	00 (?)	100	X	[31:20]	[11:0]
LHU	I	00 (?)	101	X	[31:20]	[11:0]
BEQ	SB	01	000	X	[31:25][11:7]	[12][10:5][4:1][11]
BNE	SB	01	001	X	[31:25][11:7]	[12][10:5][4:1][11]
BLT	SB	01	100	X	[31:25][11:7]	[12][10:5][4:1][11]
BGE	SB	01	101	X	[31:25][11:7]	[12][10:5][4:1][11]
BLTU	SB	01	110	X	[31:25][11:7]	[12][10:5][4:1][11]
BGEU	SB	01	111	X	[31:25][11:7]	[12][10:5][4:1][11]

```
//
//  +----------------------------------------------------------------------------------------+
//  |  Instruction set (Which use imm. gen.)                                                 |
//  +---------+--------+---------+------+-----+-----------------------+-------------------------+
//  |  Inst.  |  Type  |  ALUOp  |  F3  |  F7 |  Immediate location   |  Format                 |
//  +---------+--------+---------+------+-----+-----------------------+-------------------------+
//  |  ADDI   |   I    |   10    | 000  |  X  |    [31:20]            | [11:0]                  |
//  |  SLTI   |   I    |   10    | 010  |  X  |    [31:20]            | [11:0]                  |
//  |  SLTIU   |    I    |    10     |  011 |   X |     [31:20]             | [11:0]                    |
//  |  XORI    |    I    |    10     |  100 |   X |     [31:20]             | [11:0]                    |
//  |  ORI     |    I    |    10     |  110 |   X |     [31:20]             | [11:0]                    |
//  |  ANDI    |    I    |    10     |  111 |   X |     [31:20]             | [11:0]                    |
//  |  SLLI    |    I    |    10     |  001 |   X |     [24:20] (no ext)    | [4:0]                     |
//  |  SRLI    |    I    |    10     |  101 |   X |     [24:20] (no ext)    | [4:0]                     |
//  |  SRAI    |    I    |    10     |  101 |   X |     [24:20] (no ext)    | [4:0]                     |
//  |  JALR    |    I    |    10     |  000 |   X |     [31:20]             | [11:0]                    |
//  +----------+---------+----------+------+-----+-------------------------+---------------------------+
//  |  LUI     |    U    |    XX     |  XXX |   X |     [31:12]   (up)      | [31:12]                   |
//  |  AUIPC   |    U    |    XX     |  XXX |   X |     [31:12]   (up)      | [31:12]                   |
//  |  JAL     |    UJ   |    XX     |  XXX |   X |     [31:12]   (up)      | [20][10:1][11][19:12]     |
//  +----------+---------+----------+------+-----+-------------------------+---------------------------+
//  |  SB      |    S    |    00 (?)|  000 |   X |     [31:25][11:7]       | [11:5][4:0]               |
//  |  SH      |    S    |    00 (?)|  001 |   X |     [31:25][11:7]       | [11:5][4:0]               |
//  |  SW      |    S    |    00 (?)|  010 |   X |     [31:25][11:7]       | [11:5][4:0]               |
//  +----------+---------+----------+------+-----+-------------------------+---------------------------+
//  |  LB      |    I    |    00 (?)|  000 |   X |     [31:20]             | [11:0]                    |
//  |  LH      |    I    |    00 (?)|  001 |   X |     [31:20]             | [11:0]                    |
//  |  LW      |    I    |    00 (?)|  010 |   X |     [31:20]             | [11:0]                    |
//  |  LBU     |    I    |    00 (?)|  100 |   X |     [31:20]             | [11:0]                    |
//  |  LHU     |    I    |    00 (?)|  101 |   X |     [31:20]             | [11:0]                    |
//  +----------+---------+----------+------+-----+-------------------------+---------------------------+
//  |  BEQ     |    SB   |    01     |  000 |   X |     [31:25][11:7]       | [12][10:5][4:1][11]       |
//  |  BNE     |    SB   |    01     |  001 |   X |     [31:25][11:7]       | [12][10:5][4:1][11]       |
//  |  BLT     |    SB   |    01     |  100 |   X |     [31:25][11:7]       | [12][10:5][4:1][11]       |
//  |  BGE     |    SB   |    01     |  101 |   X |     [31:25][11:7]       | [12][10:5][4:1][11]       |
//  |  BLTU    |    SB   |    01     |  110 |   X |     [31:25][11:7]       | [12][10:5][4:1][11]       |
//  |  BGEU    |    SB   |    01     |  111 |   X |     [31:25][11:7]       | [12][10:5][4:1][11]       |
//  +----------+---------+----------+------+-----+-------------------------+---------------------------+

module immediate_gen(
    input  [31:0] INSTRUCTION,
```

```verilog
    output reg [31:0] IMMEDIATE_OUT
  );

  always@(*) begin
    case(INSTRUCTION[6:0])
      7'b0110111:              // LUI
        begin
          IMMEDIATE_OUT[31:12] <= INSTRUCTION[31:21];
          IMMEDIATE_OUT[11:0]  <= 12'b0;
        end

      7'b0010111:              // AUIPC
        begin
          IMMEDIATE_OUT[31:12] <= INSTRUCTION[31:21];
          IMMEDIATE_OUT[11:0]  <= 12'b0;
        end

      7'b1101111:              // JAL
        begin
          IMMEDIATE_OUT[31:20] <= {12{INSTRUCTION[31]}};
          IMMEDIATE_OUT[19:12] <= INSTRUCTION[19:12];
          IMMEDIATE_OUT[11]    <= INSTRUCTION[20];
          IMMEDIATE_OUT[10:1]  <= INSTRUCTION[30:21];
          IMMEDIATE_OUT[0]     <= 1'b0;
        end

      7'b1100111:              // JALR
        begin
          IMMEDIATE_OUT[31:12] <= {20{INSTRUCTION[31]}};
          IMMEDIATE_OUT[11:0]  <= INSTRUCTION[31:20];
        end

      7'b0010011:              // SLLI / SRLI / SRAI
        begin
          if ((INSTRUCTION[14:12] == 001)|(INSTRUCTION[14:12] == 001)) begin
            // SLLI / SRLI / SRAI
            IMMEDIATE_OUT[31:5]  <= 27'b0;
            IMMEDIATE_OUT[4:0]   <= INSTRUCTION[24:20];

          end else begin
            // Other I type
            IMMEDIATE_OUT[31:12] <= {12{INSTRUCTION[31]}};
            IMMEDIATE_OUT[11:0]  <= INSTRUCTION[31:20];
          end
        end

      7'b0100011:   // Store
        begin
          IMMEDIATE_OUT[31:12] <= {20{INSTRUCTION[31]}};
          IMMEDIATE_OUT[11:5]  <= INSTRUCTION[31:25];
          IMMEDIATE_OUT[4:0]   <= INSTRUCTION[11:7];
        end

      7'b0000011:   // Load
        begin
          IMMEDIATE_OUT[31:12] <= {20{INSTRUCTION[31]}};
          IMMEDIATE_OUT[11:0]  <= INSTRUCTION[31:20];
        end

      7'b1100011:   // Branch
        begin
          IMMEDIATE_OUT[31:13] <= {19{INSTRUCTION[31]}};
          IMMEDIATE_OUT[12]    <= INSTRUCTION[31];
          IMMEDIATE_OUT[11]    <= INSTRUCTION[7];
          IMMEDIATE_OUT[10:5]  <= INSTRUCTION[30:25];
          IMMEDIATE_OUT[4:1]   <= INSTRUCTION[11:8];
          IMMEDIATE_OUT[0]     <= 1'b0;
        end

      default:
        IMMEDIATE_OUT[31:0] <= 32'bx;

    endcase
  end
endmodule
```

Integration into the processor

The completed modules were integrated into the processor. As the control unit is still under completion, this wasn't added. but the rest were added.

```
module processor_main(
    input CLK
  );

  // Program counter
  wire [31:0] INS_ADDR, INS;
  wire [31:0] INS_ADDR_IN;

  ProgramCounter pc(
    .instruct_address(INS_ADDR),
    .clk(CLK),
    .instruct_address_in(INS_ADDR_IN)
   );




  // ---- Instruction memory ----

  InstructMem imem(
    .Pro_count(INS_ADDR),
    .inst_out(INS)
   );




  // ---- Control signals ----

  wire CTRL_BRANCH, CTRL_MEMREAD, CTRL_MEMTOREG, CTRL_MEMWRITE, CTRL_ALUSRC, CTRL_REGWRITE;
  wire [1:0] CTRL_ALUOP;

  //////////////////////////
  // Contol unit goes here //
  //////////////////////////



  // ---- Registry file ----

  wire [31:0] RS1_DATA, RS2_DATA, RD_DATA;

  Register_File regfile(
    .Read_reg01(INS[19:15]),      // rs1 addr
    .Read_reg02(INS[24:20]),     // rs2 addr
    .Write_reg(INS[11:7]),       // rd  addr
    .Write_data(RD_DATA),      // data written to register
    .Read_data01(RS1_DATA),      // rs1 data
    .Read_data02(RS2_DATA),      // rs2 data
    .write_signal(CTRL_REGWRITE), // control regwrite
    .clk(CLK)                // clk
  );

```

```



// ---- Sign extension ----

wire [31:0] IMM_EXT; // Sign extended / instruction correctly formatted immediate

immediate_gen immgen(
  .INSTRUCTION(INS),
  .IMMEDIATE_OUT(IMM_EXT)
);




// ---- ALU control ----

wire [3:0] ALU_OPCMD;
wire [2:0] ALU_BRANCHCMD;

alu_ctrl aluctrl(
  .ALUOp(CTRL_ALUOP),
  .FUNC3(INS[14:12]),
  .FUNC7(INS[30]),
  .ALUCTRL(ALU_OPCMD),
  .BRANCHCONDITION(ALU_BRANCHCMD)
);




// ---- ALU ----

wire [31:0] B_RS2_IMM, ALU_OUT;
wire ALU_BRANCHFLAG;
assign B_RS2_IMM = (CTRL_ALUSRC == 1'b0) ? RS2_DATA : IMM_EXT;

alu alu(
  .A(RS1_DATA),
  .B(B_RS2_IMM),
  .CTRL(ALU_OPCMD),
  .BRANCHCONDITION(ALU_BRANCHCMD),
  .OUT(ALU_OUT),
  .BRANCHFLAG(ALU_BRANCHFLAG)
);




// ---- Data memory ----

wire [31:0] DMEM_OUT;

DataMem dmem(
  .clk(CLK),
  .write_en(CTRL_MEMWRITE),
  .read_en(CTRL_MEMREAD),
  .address(ALU_OUT),     // Address bus width is 32 bits
  .data_in(RS2_DATA),    // Data bus width is 32 bits
  .data_out(DMEM_OUT)
);

assign RD_DATA = (CTRL_MEMTOREG == 1'b1) ? DMEM_OUT : ALU_OUT;




// ---- Address calculation ----
```

```

  wire [31:0] PC_ORDINARY, PC_BRANCH;

  cla32 add_0(.A(INS_ADDR), .B(32'd4), .CIN(1'b0), .OF(), .SUM(PC_ORDINARY), .BAND(),
    .BXOR());

  wire [31:0] IMM_EXT_SHIFTED;
  assign IMM_EXT_SHIFTED = IMM_EXT << 1'b1;

  cla32 add_1(.A(INS_ADDR), .B(IMM_EXT_SHIFTED), .CIN(1'b0), .OF(), .SUM(PC_BRANCH), .
    BAND(), .BXOR());

  assign INS_ADDR_IN = ((CTRL_BRANCH & ALU_BRANCHFLAG) == 1'b0) ? PC_ORDINARY : PC_BRANCH;

endmodule
```

Members' Contribution

- Nawarathna D.M.G.B - Refining control logic, program counter increment logic
- Senavirathne I.U.B. - Control unit
- Silva M.K.Y.U.N - ALU control, Immediate generation