```verilog
/*
   DataPath.v stands as the top-level module, instantiating all registers, the ALU, the Bus, and Memory.
   For Phase 1/2 purposes, all control signals are simulated to validate the correctness of datapath operations.
*/

`timescale 1ns / 1ps

/*
   DESIGN DECISION:
   Our group decided to incorporate as much vectorization as possible, given the sparsity of control signals such as:
   register enables, register 'reads', and ALU operation IDs. This allows for far cleaner and legible description,
   without loss of convenience associated to referencing registers by name.

   All register and operation IDs are associated their own indices. This is achieved with the following const. definitions.

   - Enable signals for general purpose registers (like 'rXin') are grouped under GRin (One-Hot-Encoded).
   - Enable signals for datapath registers (like 'PCin', 'IRin', 'MARin', 'MDRin', etc.) are grouped under DPin (One-Hot-Encoded).
   - Read signals for general purpose registers (like 'rXout') are grouped under GRout (One-Hot-Encoded).
   - Read signals for datapath registers (like 'PCout', 'IRout', 'MARout', 'MDRout', etc.) are grouped under DPout (One-Hot-Encoded).
   - ALU control signals (like 'add', 'sub', etc.) are grouped under ALUopp (One-Hot-Encoded).
*/

/*
   PHASE 2 EDITS:
   New module instances:
   - ram: The 512x32 memory unit.
   - SelectAndEncodeLogic: IR decoding logic.
   - condition_ff_logic: branch condition decoding logic.

   New control signals were added to the module's I/O list:

   Inputs
   - CONin serves as the enable for conditional branch logic decoding.
   - Gra, Grb, Grc, Rout, Rin, BAout select the contents of the IR to be placed on the GRin / GRout register-enable Buses.
   - RAM_wr enables the memory write operation.

   Outputs:
   - CON indicates whether a branch condition as been met.

*/


`define PC 0
`define IR 1
`define Y 2
`define MAR 3
`define MDR 4
`define INPORT 5
`define OUTPORT 6
`define Z   7 //Z used for Enable
`define ZHI 8 //ZHI / ZLO used for outputs
`define ZLO 9
`define HI 10
`define LO 11
`define READ 12
`define C   13

`define ADD 0
`define SUB 1
`define NEG 2
`define MUL 3
`define DIV 4
`define AND 5
`define OR  6
`define ROR 7
```

```verilog
`define ROL 8
`define SLL 9
`define SRA 10
`define SRL 11
`define NOT 12
`define INC 13

module DataPath (
   /*******Control Signals******/

   input clk, clr, CONin,
   input Gra, Grb, Grc, Rin, Rout, BAout, RAM_wr,


   // Register Write Control
   input [15:0] DPin,

   // Register Read Control
   input [15:0] DPout,

   // ALU Control
   input [15:0] ALUopp,

   input  [31:0] INPORTin,
   output [31:0] OUTPORTout,
   output [31:27] IRop,
   output CON
);

   // Output from Bus
   wire [31:0] BusMuxOut;

   // Inputs to Bus
   wire [31:0] BusMuxInGR;
   wire [31:0] BusMuxInPC;
   wire [31:0] BusMuxInINPORT;
   wire [31:0] BusMuxInMDR;
   wire [31:0] BusMuxInHI;
   wire [31:0] BusMuxInLO;
   wire [31:0] YtoA;
   wire [63:0] CtoZ;
   wire [63:0] ZtoBusMux;
   wire [31:0] IRout;
   wire [31:0] MARout;
   wire [31:0] C;
   wire [15:0] GRin;
   wire [15:0] GRout;
   wire [31:0] Mdatain;

   assign IRop = IRout[31:27];

   wire GR_Read;
   assign GR_Read = |GRout;

   wire [31:0] MDRin;
   assign MDRin = DPin[`READ] ?  Mdatain : BusMuxOut ;

   // General Purpose Register instantiation
   R0_R15_GenPurposeRegs GR(clk, clr, BAout, BusMuxOut, GRin, GRout, BusMuxInGR);

   // All Datapath Register instantiations.
   register PC       (clr, clk, DPin[`PC], BusMuxOut, BusMuxInPC);
   register IR       (clr, clk, DPin[`IR], BusMuxOut, IRout);
   register Y        (clr, clk, DPin[`Y], BusMuxOut, YtoA);
   register MAR      (clr, clk, DPin[`MAR], BusMuxOut, MARout);
   register MDR      (clr, clk, DPin[`MDR], MDRin, BusMuxInMDR);
   register INPORT   (clr, clk, DPin[`INPORT], INPORTin, BusMuxInINPORT);
   register OUTPORT  (clr, clk, DPin[`OUTPORT], BusMuxOut, OUTPORTout);
   register HI       (clr, clk, DPin[`HI], BusMuxOut, BusMuxInHI);
   register LO       (clr, clk, DPin[`LO], BusMuxOut, BusMuxInLO);
   register Z        (clr, clk, DPin[`Z] , CtoZ, ZtoBusMux);
      defparam Z.DATA_WIDTH_IN  = 64,
               Z.DATA_WIDTH_OUT  = 64;

   conditional_ff_logic CON_FF (IRout[20:19], BusMuxOut, CONin, clk, CON);

```

```verilog
	// Bus
	Bus DataPathBus	(BusMuxInGR, BusMuxInHI, BusMuxInLO, ZtoBusMux[63:32], ZtoBusMux[31:0], BusMuxInPC, BusMuxInMDR, BusMuxInINPORT, C,
				GR_Read, DPout[`HI], DPout[`LO], DPout[`ZHI], DPout[`ZLO], DPout[`PC], DPout[`MDR], DPout[`INPORT], DPout[`C], BusMuxOut );

	// ALU
	ALU DP_ALU		(YtoA, BusMuxOut, ALUopp, clk, CtoZ);

	// Select And Encode Module

	SelectAndEncodeLogic DP_SnEL(IRout, Gra, Grb, Grc, Rin, Rout, BAout, C, GRin, GRout);

	// RAM
	ram DP_ram (clk, RAM_wr, MARout[8:0], MARout[8:0], BusMuxInMDR, Mdatain);

endmodule

```