```verilog
module DECODE
(
   input [15:0] instr,
   input FETCH,
   input EXEC1,
   input EXEC2,
   input COND_result,
   output R0_count,
   output R0_en,
   output R1_en,
   output R2_en,
   output R3_en,
   output R4_en,
   output R5_en,
   output R6_en,
   output R7_en,
   output [2:0] s1,
   output [2:0] s2,
   output [2:0] s3,
   output s4,
   output RAMd_wren,
   output RAMd_en,
   output RAMi_en,
   output ALU_en,
   output E2,
   output stack_en,
   output stack_rst,
   output stack_rw,
   output s5,
   output s6,
   output ADD1_en
);

   wire msb = instr[15];                //MSB of the instruction word
   wire ls = instr[14];                 //LDA or STA bit
   wire [2:0] Rls = instr[13:11];       //Register in the LDA/STA operation
   wire [10:0] addr = instr[10:0];      //Memory address in the LDA/STA operation
   wire [5:0] op = instr[14:9];         //Opcode in regular instructions
   wire [2:0] Rd = instr[8:6];          //Destination register in command
   wire [2:0] Rs1 = instr[5:3];         //Source register 1 in command
   wire [2:0] Rs2 = instr[2:0];         //Source register 2 in command

   //Different opcodes (refer to documentation):
   wire LDA =  msb & ~ls;
   wire STA =  msb &  ls;
   wire JMP = ~msb & ~op[5] & ~op[4] & ~op[3] & ~op[2] & ~op[1] & ~op[0];
   wire JMA = ~msb & ~op[5] & ~op[4] & ~op[3] & ~op[2] & ~op[1] &  op[0];
   wire JCX = ~msb & ((~op[5] & ~op[4] & ~op[3] & op[2]) | (~op[5] & ~op[4] & op[3] & ~op[2]));
   wire MUL = ~msb & ~op[5] &  op[4] &  op[3] &  op[2] & ~op[1] & ~op[0];
   wire MLA = ~msb & ~op[5] &  op[4] &  op[3] &  op[2] & ~op[1] &  op[0];
   wire MLS = ~msb & ~op[5] &  op[4] &  op[3] &  op[2] &  op[1] & ~op[0];
   wire PSH = ~msb &  op[5] & ~op[4] &  op[3] & ~op[2] & ~op[1] & ~op[0];
   wire POP = ~msb &  op[5] & ~op[4] &  op[3] & ~op[2] & ~op[1] &  op[0];
   wire LDR = ~msb &  op[5] & ~op[4] &  op[3] & ~op[2] &  op[1] & ~op[0];
   wire STR = ~msb &  op[5] & ~op[4] &  op[3] & ~op[2] &  op[1] &  op[0];
   wire CLL = ~msb &  op[5] & ~op[4] & ~op[3] &  op[2] &  op[1] & ~op[0];
```

```verilog
    wire RTN = ~msb &  op[5] & ~op[4] & ~op[3] &  op[2] &  op[1] &  op[0];
    wire NOP = ~msb &  op[5] &  op[4] &  op[3] &  op[2] &  op[1] & ~op[0];
    wire STP = ~msb &  op[5] &  op[4] &  op[3] &  op[2] &  op[1] &  op[0];

    assign R0_count = (FETCH & ~STP) | (EXEC1 & ~(JMP | JMA | (JCX & COND_result) | STP | LDR | LDA | MUL | MLA | MLS | POP | RTN | CLL)) | (EXEC2 & (LDR | LDA | MUL | MLA | MLS | POP));
    assign R0_en = (EXEC1 & (~(STA | NOP | STP | LDA | PSH | LDR | CLL | RTN) & ~Rd[2] & ~Rd[1] & ~Rd[0] | JMP | (JCX & COND_result) | JMA)) | (EXEC2 & LDA & ~Rls[2] & ~Rls[1] & ~Rls[0]) | (EXEC2 & (MUL | MLA | MLS | POP | STR | LDR) & ~Rd[2] & ~Rd[1] & ~Rd[0]) | (EXEC2 & RTN) | (EXEC1 & CLL);
    assign R1_en = (EXEC1 & ~(JMP | JMA | JCX | STA | LDA | MUL | MLA | MLS | NOP | STP | POP | PSH | LDR | CLL | RTN) & ~Rd[2] & ~Rd[1] &  Rd[0]) | (EXEC2 & LDA & ~Rls[2] & ~Rls[1] &  Rls[0]) | (EXEC2 & (MUL | MLA | MLS | POP | LDR) & ~Rd[2] & ~Rd[1] &  Rd[0]);
    assign R2_en = (EXEC1 & ~(JMP | JMA | JCX | STA | LDA | MUL | MLA | MLS | NOP | STP | POP | PSH | LDR | CLL | RTN) & ~Rd[2] & Rd[1] & ~Rd[0]) | (EXEC2 & LDA & ~Rls[2] &  Rls[1] & ~Rls[0]) | (EXEC2 & (MUL | MLA | MLS | POP | LDR) & ~Rd[2] &  Rd[1] & ~Rd[0]);
    assign R3_en = (EXEC1 & ~(JMP | JMA | JCX | STA | LDA | MUL | MLA | MLS | NOP | STP | POP | PSH | LDR | CLL | RTN) & ~Rd[2] & Rd[1] &  Rd[0]) | (EXEC2 & LDA & ~Rls[2] &  Rls[1] &  Rls[0]) | (EXEC2 & (MUL | MLA | MLS | POP | LDR) & ~Rd[2] &  Rd[1] &  Rd[0]);
    assign R4_en = (EXEC1 & ~(JMP | JMA | JCX | STA | LDA | MUL | MLA | MLS | NOP | STP | POP | PSH | LDR | CLL | RTN) &  Rd[2] & ~Rd[1] & ~Rd[0]) | (EXEC2 & LDA &  Rls[2] & ~Rls[1] & ~Rls[0]) | (EXEC2 & (MUL | MLA | MLS | POP | LDR) &  Rd[2] & ~Rd[1] & ~Rd[0]);
    assign R5_en = (EXEC1 & ~(JMP | JMA | JCX | STA | LDA | MUL | MLA | MLS | NOP | STP | POP | PSH | LDR | CLL | RTN) &  Rd[2] & ~Rd[1] &  Rd[0]) | (EXEC2 & LDA &  Rls[2] & ~Rls[1] &  Rls[0]) | (EXEC2 & (MUL | MLA | MLS | POP | LDR) &  Rd[2] & ~Rd[1] &  Rd[0]);
    assign R6_en = (EXEC1 & ~(JMP | JMA | JCX | STA | LDA | MUL | MLA | MLS | NOP | STP | POP | PSH | LDR | CLL | RTN) &  Rd[2] & Rd[1] & ~Rd[0]) | (EXEC2 & LDA &  Rls[2] &  Rls[1] & ~Rls[0]) | (EXEC2 & (MUL | MLA | MLS | POP | LDR) &  Rd[2] &  Rd[1] & ~Rd[0]);
    assign R7_en = (EXEC1 & ~(JMP | JMA | JCX | STA | LDA | MUL | MLA | MLS | NOP | STP | POP | PSH | LDR | CLL | RTN) &  Rd[2] & Rd[1] &  Rd[0]) | (EXEC2 & LDA &  Rls[2] &  Rls[1] &  Rls[0]) | (EXEC2 & (MUL | MLA | MLS | POP | LDR) &  Rd[2] &  Rd[1] &  Rd[0]);
    assign s1[2] = (~(JMP | JMA | STA | LDA | NOP | STP | POP | CLL | RTN) & Rs1[2]) | (STA & Rls[2]);
    assign s1[1] = (~(JMP | JMA | STA | LDA | NOP | STP | POP | CLL | RTN) & Rs1[1]) | (STA & Rls[1]);
    assign s1[0] = (~(JMP | JMA | STA | LDA | NOP | STP | POP | CLL | RTN) & Rs1[0]) | (STA & Rls[0]);
    assign s2[2] = (~(JMP | JMA | STA | LDA | NOP | STP | POP | PSH | LDR | STR | CLL | RTN) & Rs2[2]);
    assign s2[1] = (~(JMP | JMA | STA | LDA | NOP | STP | POP | PSH | LDR | STR | CLL | RTN) & Rs2[1]);
    assign s2[0] = (~(JMP | JMA | STA | LDA | NOP | STP | POP | PSH | LDR | STR | CLL | RTN) & Rs2[0]);
    assign s3[2] = (~(STA | LDA | NOP | STP | PSH | POP | RTN) & Rd[2]);
    assign s3[1] = (~(STA | LDA | NOP | STP | PSH | POP | RTN) & Rd[1]);
    assign s3[0] = (~(STA | LDA | NOP | STP | PSH | POP | RTN) & Rd[0]);
    assign s4 = ~(LDA | LDR);
    assign RAMd_wren = EXEC1 & (STA | STR);
    assign RAMd_en = EXEC1 & (STA | LDA | STR | LDR);
    assign RAMi_en = (FETCH & ~STP) | (EXEC1 & ~(LDA | LDR | MUL | MLA | MLS | POP | STP | RTN)) | (EXEC2 & (LDA | LDR | MUL | MLA | MLS | POP | RTN));
    assign ALU_en = LDA | STA;
    assign E2 = EXEC1 & (LDA | MUL | MLA | MLS | POP | LDR | RTN);
    assign stack_en = (EXEC1 & (PSH | CLL | RTN | POP));
    assign stack_rst = STP;
    assign stack_rw = EXEC1 & (PSH | CLL);
    assign s5 = EXEC1 & (STR | LDR);
    assign s6 = (EXEC1 & (JMP | JMA | (JCX & COND_result) | CLL)) | (EXEC2 & RTN);
    assign ADD1_en = (EXEC1 & (JMP | JMA | (JCX & COND_result) | CLL)) | (EXEC2 & RTN);

endmodule

```