COL216 Computer Architecture

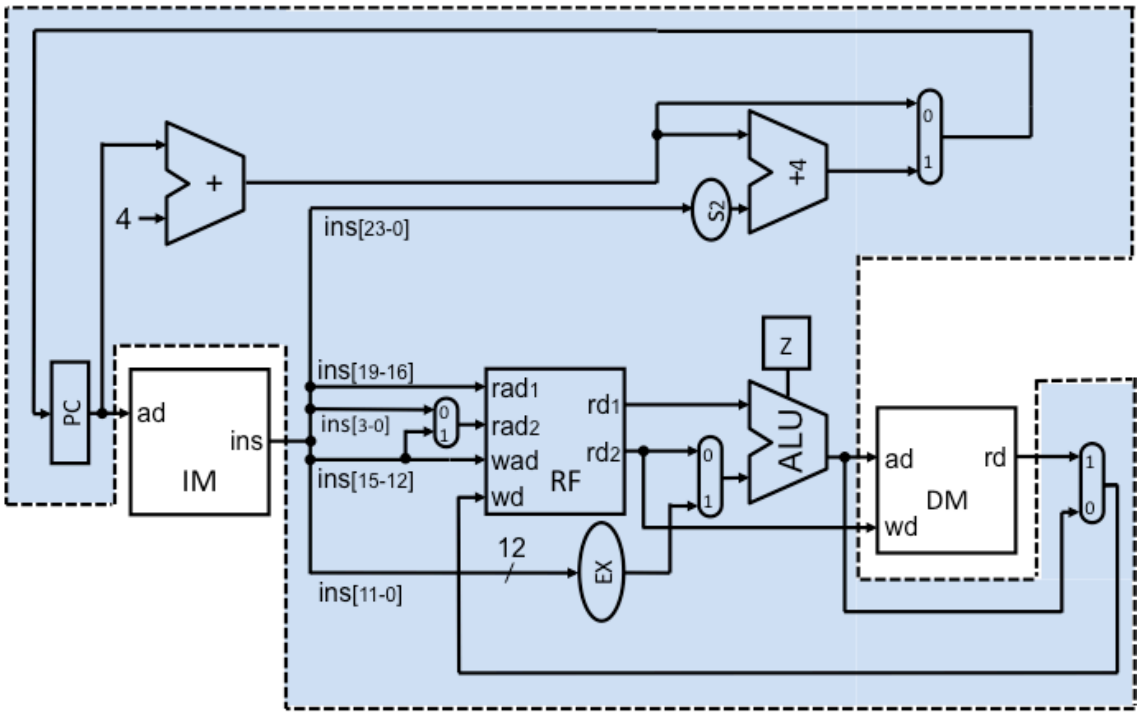
Lab Assignments 4 : CPU for a small ARM instructions subset

As a starting point for ARM processor design exercise, this assignment aims to build a CPU for the following instructions, using a very simple design approach.

{add, sub, cmp, mov, ldr, str, beq, bne, b}

Encoding of these instructions, for all the variants to be implemented, is shown in file "Instruction encoding for Lab4.pdf". Check all the fields shown shaded in order to ensure that all incorrectly encoded instructions are rejected.

The design approach to be followed is same as that discussed in Lecture 5. The focus here is on CPU only; memory modules will be added in the next assignment. In other words, the scope of this assignment is limited to only the shaded portion shown in the figure below.



The CPU entity to be designed is expected to have the following ports.

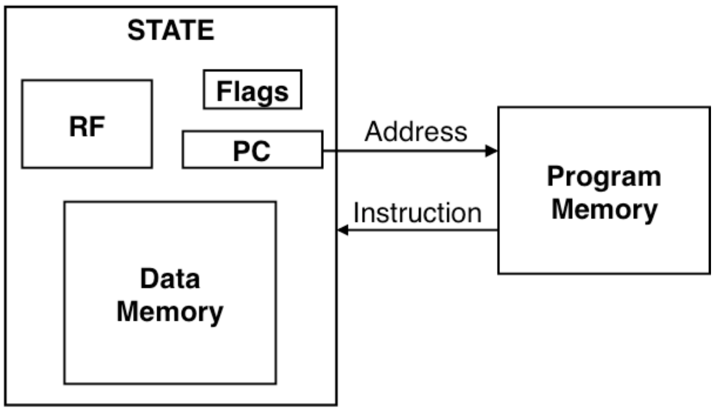
Input ports:

Clock, Reset (to initialize program counter to zero), Instruction from program memory, Data from data memory.

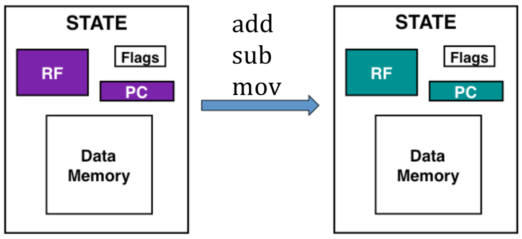
Output ports:

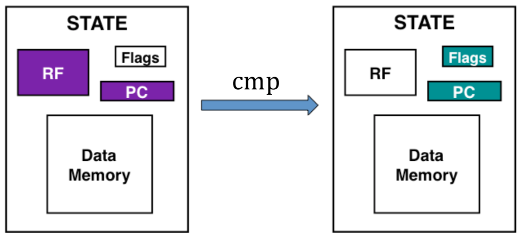
Adress to program memory, Address to data memory, Data to data memory, Write enable to data memory

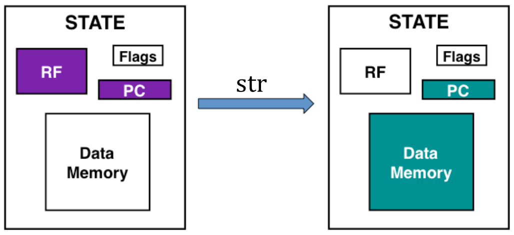
Express your CPU design in VHDL at behavioural level, viewing it as an abstract machine shown in the figure on right. The state is defined by the contents of register file, PC (same as register 15), Flags (only Z flag for the present exercise) and contents of data memory.

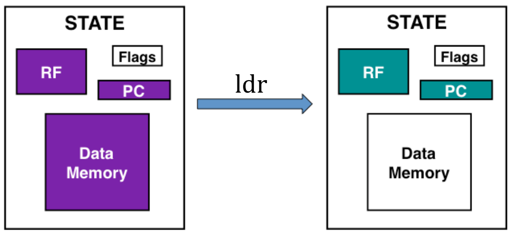


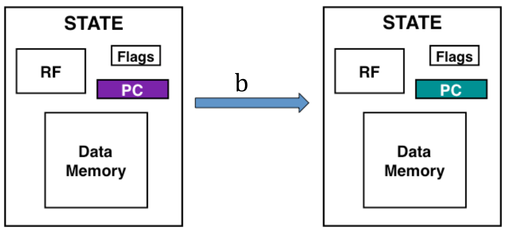
Each instruction modifies the state in a certain way. This is depicted in the series of figures below for the instructions under consideration. Here components of the state that are read are shown in purple and the components of the state that are written are shown in green.

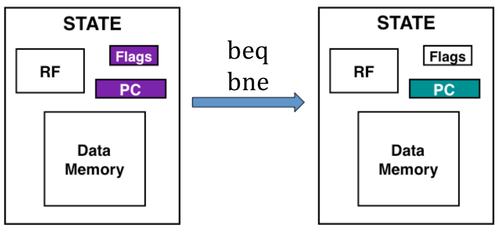












The architecture of CPU entity can be considered to have two parts -

- A set of concurrent assignments to describe the combinational circuit that determines what the next state is. Alternatively, a set of unclocked processes can be used. This part includes instruction decoding as well.
- A clocked process to actually make the change in state. Changes in RF, Flags and PC contents are to be done here. Changes in data memory contents are to be done inside

data memory component (Lab Assignment 5). CPU's role is only to activate write enable signal when that is to be done.

Some useful suggestions:

Declare signals corresponding to various instruction fields and assign these using concurrent assignments. Some examples are shown below.

```
signal cond : std_logic_vector (3 downto 0);
signal F_field : std_logic_vector (1 downto 0);
signal I_bit : std_logic;
signal shift_spec : std_logic_vector (7 downto 0);
. . . .
cond <= instruction(31 downto 28);
F_field <= instruction(27 downto 26);
I_bit <= instruction(25);
shift_spec <= instruction (11 downto 4);
```

Declare types and signals, such as those shown below, to represent outcome of instruction decoding.

```
type instr_class_type is (DP, DT, branch, unknown);
type i_decoded_type is (add,sub,cmp,mov,ldr,str,beq,bne,b,unknown);
signal instr_class : instr_class_type;
signal i_decoded : i_decoded_type;
```

Value of instr_class will be determined from F_field. Value of i_decoded will be determined from instr_class and fields such as opcode, L_bit, cond etc.