% Defining my program lines

line(1, put(1, r5)).          % Initialize r5 to 1

line(2, put(0, r1)).          % Initialize r1 to 0 (starting point)

line(3, put(10, r2)).         % Set r2 to 10 (limit for comparison)

line(4, jmpe(r1, r2, 8)).     % If r1 equals r2 (10), jump to halt (line 8)

line(5, prn(r1)).             % Print the value of r1

line(6, add(r5, r1)).         % r1 = r1 + r5 (increment)

line(7, jmpu(4)).             % Jump back to line 4

line(8, halt(stop)).          % Stop the program

% Initial register values

initial\_state([r1=0, r2=10, r5=1]).  % r1 is the counter, r2 is the limit, r5 is the increment

% To run the program

run :-

    initial\_state(State),        % Get the initial state

    execute(1, State).          % Start execution at line 1

% The line based on the current line number

execute(LineNumber, State) :-

    line(LineNumber, Action),     % Get the action from the current line

    execute\_action(Action, LineNumber, State).  % Execute the action

% The 'put' action

execute\_action(put(Value, Register), LineNumber, State) :-

    select(Register=\_, State, Register=Value, NewState),  % Update the register

    NextLine is LineNumber + 1,  % Move to the next line

    execute(NextLine, NewState).  % Continue execution

% The 'prn' action

execute\_action(prn(Register), LineNumber, State) :-

    member(Register=Value, State),  % Get the current value of the register

    write(Value), nl,                % Print the value

    NextLine is LineNumber + 1,      % Move to the next line

    execute(NextLine, State).        % Continue execution

% The 'add' action

execute\_action(add(RegisterAdd, RegisterDest), LineNumber, State) :-

    member(RegisterAdd=ValueAdd, State),              % Get the value of the register to add

    member(RegisterDest=ValueDest, State),            % Get the value of the destination register

    NewValue is ValueDest + ValueAdd,                 % Calculate the new value

    select(RegisterDest=ValueDest, State, RegisterDest=NewValue, NewState),  % Update the state

    NextLine is LineNumber + 1,                       % Move to the next line

    execute(NextLine, NewState).                       % Continue execution

% The 'jmpe' action

execute\_action(jmpe(Register1, Register2, Target), LineNumber, State) :-

    member(Register1=Value1, State),                  % Get the value of r1

    member(Register2=Value2, State),                  % Get the value of r2

    (Value1 >= Value2 -> NextLine is Target ; NextLine is LineNumber + 1),  % Check condition

    execute(NextLine, State).  % Continue execution

% The 'jmpu' action

execute\_action(jmpu(Target), \_, State) :-

    execute(Target, State).  % Jump to the specified line

% Handle the 'halt' or ‘stop’ action

execute\_action(halt(\_), \_, \_) :-

    !.  % Stop execution