```
with CODE,
     EXCEPTIONS,
     INTERFACE,
     INSTRUCTION_UNIT,
     EXECUTION_UNIT;

procedure SIMULATE is

    pragma MAIN;

    CURRENT_INSTRUCTION : CODE.INSTRUCTION;

    procedure CYCLE is
    begin
        INSTRUCTION_UNIT.FETCH   (CURRENT_INSTRUCTION);
        EXECUTION_UNIT.DISPATCH (CURRENT_INSTRUCTION);
        INTERFACE.CONVERSE (INTERFACE.AFTER_DISPATCH,
                            CURRENT_INSTRUCTION);
    exception
        when INTERFACE.END_SIMULATION =>
            raise;

        when EXCEPTIONS.IN_SIMULATION.RAISING_EXCEPTION =>
            INTERFACE.CONVERSE (INTERFACE.AFTER_EXCEPTION,
                                CURRENT_INSTRUCTION);

        when EXCEPTIONS.IN_SIMULATION.DECLARED_OUTER_MODULE =>
            INTERFACE.CONVERSE (INTERFACE.AFTER_DECLARATION,
                                CURRENT_INSTRUCTION);

        when EXCEPTIONS.IN_SIMULATION.ACTIVATED_OUTER_MODULE =>
            INTERFACE.CONVERSE (INTERFACE.AFTER_ACTIVATION,
                                CURRENT_INSTRUCTION);

        when EXCEPTIONS.IN_SIMULATION.TERMINATING_OUTER_MODULE =>
            INTERFACE.CONVERSE (INTERFACE.AFTER_TERMINATION,
                                CURRENT_INSTRUCTION);

        when EXCEPTIONS.IN_SIMULATION.SCHEDULER_QUEUE_EMPTY =>
            INTERFACE.CONVERSE (INTERFACE.WITH_SCHEDULER_EMPTY,
                                CURRENT_INSTRUCTION);

        when EXCEPTIONS.IN_SIMULATION.SIMULATION_ERROR =>
            INTERFACE.CONVERSE (INTERFACE.AT_SIM_ERROR,
                                CURRENT_INSTRUCTION);

        when EXCEPTIONS.IN_SIMULATION.UNHANDLED_EXCEPTION =>
            INTERFACE.CONVERSE
                (INTERFACE.UNHANDLED_EXCEPTION_REACHES_MAIN_PROGRAM,
                 CURRENT_INSTRUCTION);

        when EXCEPTIONS.IN_SIMULATION.UNIMPLEMENTED_OPERATION =>
            INTERFACE.CONVERSE (INTERFACE.AT_UNIMPLEMENTED_OP,
                                CURRENT_INSTRUCTION);

        when INSTRUCTION_UNIT.BREAK_OCCURRED =>
            INTERFACE.CONVERSE (INTERFACE.AT_PC_VALUE,
                                CURRENT_INSTRUCTION);

        when others =>
```

```
          INTERFACE.CONVERSE (INTERFACE.AFTER_EXCEPTION,
                              CURRENT_INSTRUCTION);
    end CYCLE;
```

```
begin
    INTERFACE.CONVERSE (INTERFACE.AT_INITIALIZATION);
    loop
        CYCLE;
    end loop;
exception
    when INTERFACE.END_SIMULATION =>
        null;

    when others =>
        null;
end SIMULATE;
```