- `sr(0 to 11)`. This comes from a bank of 12 DIP switches. Depending on the switch settings, these inputs will be driven to either logic '1' or logic '0'.

- `switch_enable` and `switch_display`. These are pushbutton switches that are normally logic '0', except when pressed when they become a logic '1'. In your VHDL, please assume for now that the switches are debounced, but you must still synchronize them.

- `reset_in`. This is a pushbutton switch that is normally logic '1', except when pressed when it becomes a logic '0'. It must be synchronized.

- `switches(0 to 3)` and `sing_inst_in`. This comes from a bank of 5 DIP switches. Depending on the switch settings, these inputs will be driven to either logic '1' or logic '0'.

- 12 LEDs are driven from `bus_out` of the datapath.

One way of using these components in the design of your front panel is to use `reset_in` as a reset signal that throws your machine back into the idle state. `sing_inst_in` can be used to enable a single-stepping mode for execution. `switches` can be used to encode a set of commands to be executed by the PDP-8 on hitting `switch_enable` or a set of things to display on the LEDs on hitting `switch_display`. Possible commands include load MA, load MB, load PC, load IR, load AC, load memory, read memory, examine, deposit, clear, continue. Possible things to display are MA, MB, PC, IR, AC, effective address, SR. *You are free to customize and design the front panel any way you like using the available resources and connections.*

```
        jms0: out std_logic;
        acleft, acright, ce_bar_out, read: out std_logic;
        cll, cml: out std_logic);
end;
```

The I/O interface to the UART should have the following entity:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY uart_interface IS
    PORT(
        acclr : OUT std_logic;
        ioskip : OUT std_logic;
        orac : OUT std_logic;
        uart_address : OUT std_logic_vector(0 TO 2);
        uart_cs0 : OUT std_logic;
        uart_read : OUT std_logic;
        uart_write : OUT std_logic;
        int_request : OUT std_logic;
        iop1 : IN std_logic;
        iop2 : IN std_logic;
        iop4 : IN std_logic;
        iop_device_address : IN std_logic_vector(0 TO 4);
        rxrdy_bar : IN std_logic;
        txrdy_bar : IN std_logic
    );
END uart_interface;
```

You will need to complete the ASM charts for the rest of your design, including all of the execution control. This should be followed by completing the VHDL for the entire PDP-8, including the UART interface. You should be able to simulate the complete machine, including loading the memory from the front panel and executing some simple machine language programs. The I/O should also be fully tested, including interrupt handling. You *do not* have to attempt to model the UART in VHDL, since this is far too tricky. We will debug this aspect of the machine in hardware.

The front panel components that you have available to you on the board are:

Columbia University
Department of Electrical Engineering
EE E4340. Homework/Laboratory #5.
PDP-8 Execution Controls, I/O Interface, and Front Panel.
Due: April 6, 1999

You should expect that completing the remainder of the PDP-8 design is going to take you some time. Please do not wait until last minute. As always, please do not hesitate to come to me or the TA if you get stuck.

In this laboratory, you are to complete the design of the PDP-8 minicomputer, including the interface to the 16550 UART and a front panel. In order to be compatible with the existing connections on the board, your control logic should have the following interface:

```
library ieee;
use ieee.std_logic_1164.all;
entity pdp8_control is
    port(clock: in std_logic;
         -- Inputs from the datapath
         cout, link: in std_logic;
         bus_in: in std_logic_vector(0 to 11);
         mb_iszero_in: in std_logic;

         -- Front panel support
         switch_enable, switch_display, reset_in: in std_logic;
         switches : in std_logic_vector(0 to 3);
         sing_inst_in   : in std_logic; -- toggle switch

         -- I/O signals
         ioskip, acclr, orac: in std_logic;
         int_request: in std_logic;
         iop1,iop2,iop4: out std_logic;
         iop_device_address: out std_logic_vector(0 to 5);

         -- Control lines to the datapath
         loadma_out, loadmb_out, loadpc_out, loadir_out: out std_logic;
         alusel_out, datasel_out: out std_logic_vector(0 to 2);
```