Columbia University
Department of Electrical Engineering
EECS E4340. Laboratory #2.
PDP-8 Fetch controls and front panel.
Due: March 24, 2004

In subsequent homeworks/laboratories, you will be building up the hard-wired control logic of the PDP-8. In this homework, you will write VHDL for the fetch controls of the PDP-8 that we went over in class and design the front panel of your PDP-8. 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_ulogic;
         -- Inputs from the datapath
         cout, link: in std_ulogic;
         bus_in: in std_ulogic_vector(0 to 11);
         mb_iszero_in: in std_ulogic;

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

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

         -- Control lines to the datapath
         loadma_out, loadmb_out, loadpc_out, loadir_out: out std_ulogic;
         alusel_out, datasel_out: out std_ulogic_vector(0 to 2);
         jms0: out std_ulogic;
```

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

For now, you can "null out" the I/O signal by driving the outputs to '0' and ignoring the inputs.

In your VHDL, please assume that any switch inputs are debounced, but they must by synchronized and pulsed (if necessary). You may "deadend" your controls for now by making the `E0` state the `IDLE` state, or you could enable continuous operation by making the `E0` state the `F1` state. Create a single symbol for your control logic and wire it up to the dataflow you created in Laboratory #1. In addition, wire up three 2168 SRAM chips to the memory bus in your schematic. Each SRAM will provide four bits of the twelve bit word. You will find a component for this called `sram_2168` in the library `pdp8` in `/tools2/ee4340/cdslibs`.

Create a testbench for your design. You might want to do your testing by loading up memory with an instruction stream and verifying that each instruction in the stream is correctly fetched.

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

- `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.* The vast majority of your front-panel support is implemented in the fetch controls, but you will need some support in your execution controls; therefore, you will have to keep this in mind in the subsequent labs.