



COMPUTER ENGINEERING PROGRAM



California Polytechnic State University

©Copyright: 2007 by Bryan Mealy



CPE 269 Experiment 7



Completing a Simple Computer: Adding Program Memory to the CPU

Objectives:

- To design an instruction unit to drive the CPU (control unit & ALU) designed previously
- To understand the relation between program memory and the control unit
- To model a ROM device in VHDL to act as program memory
- To understand the interaction between memory, CPU, and input/output

Somewhat Meaningful Comments: In Experiment 6, you added a control unit and an ALU to your datapath design of the previous week. In this experiment, you'll be completing a simple computer that is capable of executing a modest set of instructions relatively meaningful instructions.

The program memory for this experiment will be a ROM modeled in VHDL. Although in real life, the program memory would be filled by the assembler or compiler, for this experiment, you'll need to directly specify the machine code in the VHDL model of the ROM. Figure 1 shows an example of a VHDL model for a ROM. Note that you can initialize this ROM with whatever data you want or also change the storage capacity of the ROM.

```

entity romrom is
  Port ( addr : in std_logic_vector(3 downto 0);
        out_val : out std_logic_vector (3 downto 0));
end romrom;

architecture Behavioral of romrom is
  type vector_array is ARRAY (0 to 15) of std_logic_vector(3 downto 0);
  constant memory: vector_array := ( "0000", "0000", "0000", "0000", "0000",
                                     "0000", "0000", "0000", "0000", "0000",
                                     "0000", "0000", "0000", "0000", "0000",
                                     "0000" );
begin
  out_val <= memory(conv_integer(addr));
end Behavioral;

```

Figure 1: VHDL model of a ROM.

For this experiment, you'll want to add what could be referred to as an *instruction unit* to your previous design. For this experiment, the instruction unit consists of a program counter (PC) and instruction register (IR). As shown in Figure 2, the output of the program counter is used to index words stored in the ROM. The instruction register latches instructions stored in the program memory. Note that there is a VHDL model for a counter listed on the back side of the VHDL cheat sheet.

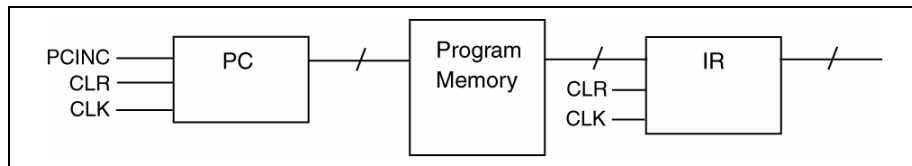


Figure 2: Instruction Unit Architecture

Once again, the control unit is going to be in charge of making the hardware do the right thing. More specifically, the control unit sequences the operations required in order for you to obtain a working (yet simple) computer. Figure 3 shows the general form (required) of the control unit. Your computer is required to implement the instructions shown in Table 1.

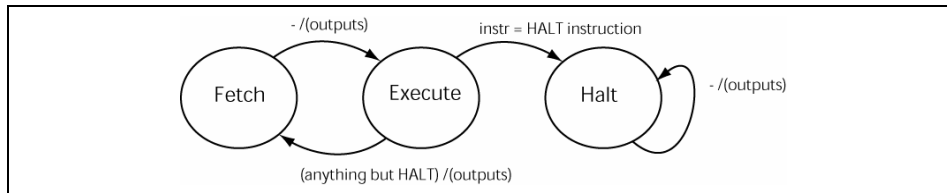


Figure 3: The general form for the control unit.

Instruction	Action	RTL Description	Comment
LDAC	Load ACC	$ACC \leftarrow REGA$	REGA loaded into lower 4-bits of the ACC; upper 4-bits do not change
ADD	Add ACC	$ACC \leftarrow ACC + REGB$	REGB is added to the lower 4-bits of ACC
AND	AND ACC	$ACC \leftarrow ACC \text{ AND } REGB$	REGB is ANDed with lower 4-bits of ACC; the upper 4-bits of ACC do not change
MLT8X	Multiply ACC x 8	$ACC \leftarrow ACC \times 8$	ACC is multiplied by 8 and the result is stored in ACC
LDA	Load SW(7:4) to REGA	$REGA \leftarrow SW(7:4)$	Load switches 7-4 into REGA
LDB	Load SW(3:0) to REGB	$REGB \leftarrow SW(3:0)$	Load switches 3-0 into REGB
LDAB	Load SW(7:4) to REGA; Load SW(3:0) to REGB	$REGA \leftarrow SW(7:4)$, $REGB \leftarrow SW(3:0)$	Load switches 7-4 into REGA, Load switches 3-0 into REGB
HALT	stop computer	N/A	stops normal operation of computer

Table 1: Instructions that your computer needs to implement.

Test your computer by configuring the ROM to execute the program shown in Table 2.

Instruction	Comment
LDA	load REGA with switch values (7:4)
LDB	load REGB with switch values (3:0)
LDAB	Load both REGA and REGB from switches
LDAC	Load ACC
ADD	ADD ACC
AND	AND ACC
MLT8X	Multiply ACC x 8
HALT	stops computer operation

Table 2: A test program for you computer.

Figure 4 shows a rough diagram of the final circuit. Notice that output signals FETCH, EXEC, and HALT have been added to the circuit outputs. These signals indicated the state of the control unit and should be connected to LD7, LD6, and LD5 on your development board, respectively.

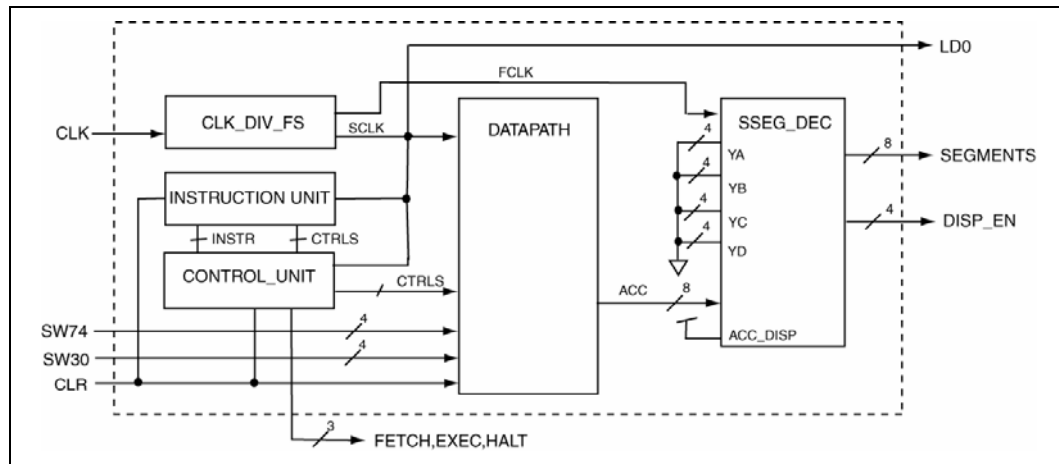


Figure 4: The somewhat final top-level circuit.

Deliverables:

1. Demonstrate your working circuit to the lab instructor or Teaching Assistant.
2. Answers to the questions below.

Questions:

1. List the instruction opcodes you chose to use in your instruction set.
2. List the complete RTL equations associated with Table 1.
3. For the ROM shown in Figure 1, state its storage capacity in the "A x B" typically associated with memory devices.
4. State what I/O devices were used in the simple computer you built in this experiment.
5. State what memory devices were used in this simple computer.
6. The switches were used to input data in this experiment. In a more useful computer, briefly describe where these inputs could have come from.
7. If one more instruction were added to the current instruction set, what would be the minimum bit-width of the instruction opcodes?