COE608 Computer Organization and Architectures
Winter 2017
Lab 5: CPU Control Unit Design

Due Date: Week 10 (Before the Start of your Lab Session)

1. Objectives

The purpose of this laboratory is to design and implement the control unit to provide control signals to the 32-bit CPU data-path designed in Lab 4. The block diagram of the control unit with all the inputs and outputs is illustrated in Figure 1.

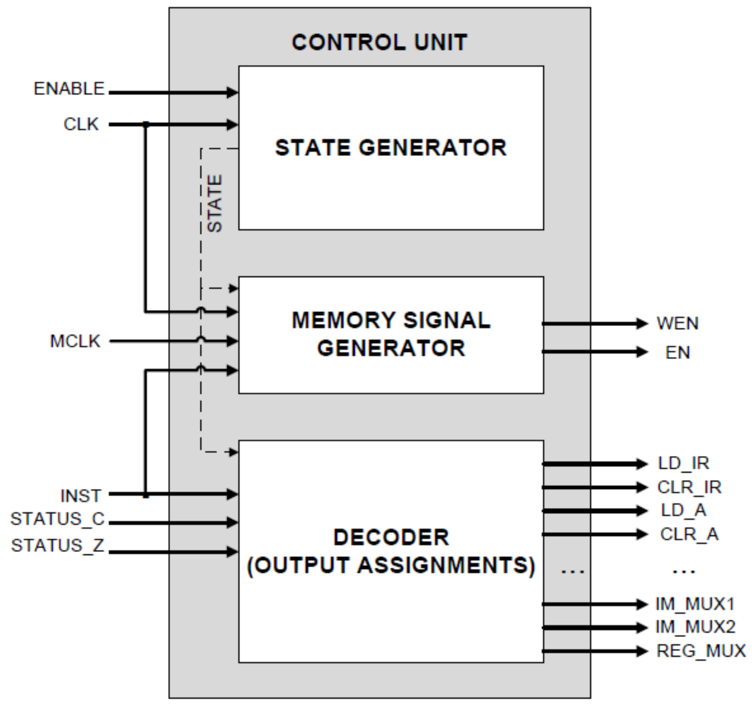

Figure 1: CPU Control Unit

Most of the control unit outputs all the control signals for the Registers, ALU and MUXes in the data-path of Lab4. It accepts as input the status bits (Zero and Carry) and the INST for instruction decisions (Note: the diagram shows the entire INST as input to the control unit. However, if we refer back to the CPU specification document, only INST[31..28] - *opcode* - and INST[27..24] - *function code* - are required). All instructions require 3 clock cycles to execute. Moreover, in situations when we desire to initialize or reset the CPU, an enable/reset input is needed.

Please note that a control signal required by the data-path, CLR PC is not produced by the control unit. The clearing of the program counter occurs during initialization. Reset signal for the CPU and CLR PC are generated by a reset circuit, which will be designed as part of Lab 6.

When we investigate the internal structure of the control unit, it can be divided it into three parts (VHDL processes), namely a sequential state generator (for T0, T1 and T2), a memory signal generator (for wen and en setup and hold times), and a combinational circuit for the decoding operations. A brief description of these processes is given below.

2. State Generator

The state generator circuit is the synchronous, sequential component of the control unit. It generates appropriate state signals based on the clock and the current state of the system. It also generates a set of pulse signals T that can be used to indicate the current state of the instruction being executed. The components of T take the following form as shown in Figure 2 (shown here for three states).

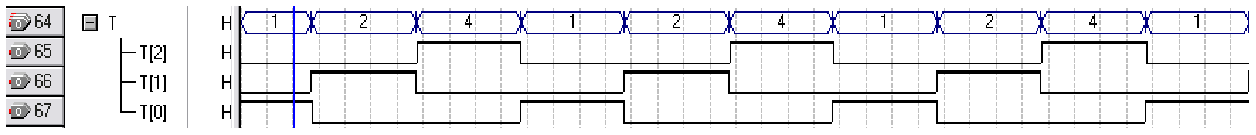

Figure 2: T Pulse Waveform

Note: When *ENABLE* signal (shown in Figure 1) is not asserted, the circuit should go to state T0 and remain in this state until the *ENABLE* signal is asserted again. At that point, the circuit resumes normal operation.

3. Operation Decoder

The operation decoder is responsible for correctly setting the control signals being fed to the data-path during instruction execution. It requires the current state, status bits (C and Z) and the INST contents to determine which instruction to execute. Essentially, the operation decoder is nothing more than an if-else type of statement (MUX), which sets the control signals appropriately. We have determined the correct settings of the control signals for each operation in the data-path lab. Note that it is wise to use **case statements** to successfully synthesize the VHDL for decoding all the CPU instructions.

4. Memory Signal Generator

To support the load and store instructions, the Write Enable (wen) and Enable (en) signals have been included in the control unit specification. Assume the signal is active high; the signal must be asserted correctly during the store and load operations as specified in memory lab manual and **cpu_specification** document. The wen and en are sensitive to the Clk, mclk, and INST given. The process template is given in the code on the next page for setting up "en" and "wen" in the "Data Memory Instructions" process. Fill in the code with the appropriate values for "en" and "wen" according to the template given in the specifications to achieve correct setup and hold times for your CPU's memory operations.

5. VHDL Implementation

The declaration below is provided as a possible reference:

```
library ieee;
use ieee.std_logic_1164.ALL;
ENTITY control IS
        PORT(
                clk, mclk                       : IN STD_LOGIC;
                enable                                  : IN STD_LOGIC;
                statusC, statusZ                : IN STD_LOGIC;
                INST                                    : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
                A_Mux, B_Mux                    : OUT STD_LOGIC;
                IM_MUX1, REG_Mux                : OUT STD_LOGIC;
                IM_MUX2, DATA_Mux               : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
                ALU_op                          : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
                inc_PC, ld_PC                   : OUT STD_LOGIC;
                clr_IR                                  : OUT STD_LOGIC;
                ld_IR                                   : OUT STD_LOGIC;
                clr_A, clr_B, clr_C, clr_Z      : OUT STD_LOGIC;
                ld_A, ld_B, ld_C, ld_Z          : OUT STD_LOGIC;
                T                                        : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
                wen, en                          : OUT STD_LOGIC);
END control;

ARCHITECTURE description OF control IS
        TYPE STATETYPE IS (state_0, state_1, state_2, etc...);
        SIGNAL present_state: STATETYPE;
BEGIN

-------- OPERATION DECODER ---------
  PROCESS (present_state, INST, statusC, statusZ, enable)
  BEGIN
-------- YOU FILL IN WHAT GOES IN HERE (DON'T FORGET TO CHECK FOR ENABLE)
-------- OUTPUT ASSIGNMENTS
   END process;


-------- STATE MACHINE ---------
   PROCESS (clk, enable)begin
-------- YOU FILL IN WHAT GOES IN HERE
   END process;


-------- DATA MEMORY INSTRUCTIONS ---------
PROCESS (mclk, clk, INST)
BEGIN
        IF(mclk'EVENT and mclk = '0') THEN
                IF(present_state = state_1 AND clk = '0') THEN
                        --LDA and LDB Signals
                        --STA and STB Signals
                        --Default Case Signals
                ELSIF(present_state = state_2 AND clk = '1') THEN
                                --LDA and LDB
                                --STA and STB
                                --Default Case
                ELSIF(present_state = state_1) THEN --or alternatively just an ELSE statement
                        --fill in
                END IF;
        END IF;
END process;
END description;
```

6. What to Hand In

Submit the VHDL implementation of the control unit along with testing (timing simulations) to demonstrate that it works correctly. For testing, you should show the correct settings of the control lines for states T0 and T1 (which should always be the same. See Figure 4 of the CPU specification document). For state T2, show that the control lines are set correctly for each of the operations supported by the data-path (See Table 1 of the CPU_specification document and refer to Table 1 of your datapath lab). Ensure that you have tested all the instructions needed for your final CPU implementation.

Your TA may quiz you on the implementation of your control unit at the time of your demo.