Assignment 3
RTL Design

Part 1 due Monday, February 22 1999
8:30 AM
Part 2 due Monday, March 1 1999
8:30 AM

Specifications

Overview

In this two-part assignment you will design, program and test a simple microcontroller.

The CPU datapath has three registers: a program counter (PC), an instruction register (I) and an accumulator register (A).

The device also has a 32-byte memory. The first 16 bytes of the memory are ROM to store a program and constants. The second 16 bytes are RAM to store variables.

The CPU controller (instruction decoder) executes each instruction in two clock cycles. The CPU controller loads the instruction from memory into the I register in the first clock cycle (the fetch state) and executes one of the eight possible instructions in the second cycle.

All instructions are one byte long. Each instruction is encoded as a 3-bit instruction type in the MS 3 bits and a 5-bit operand address field in the LS 5 bits.

The microcontroller has been divided into 5 components:

- 32-byte memory
- A datapath
- I datapath
- PC datapath
- controller

You are to design and test the five components independently before combining them into the final top-level design.

Data Types

Create two subtypes, of the std_logic_vector data type, an 8-bit `word` type, and a 5-bit `address` type. Put these type definitions in a package. Use these data types throughout your design so that you can change the CPU and memory widths relatively easily.

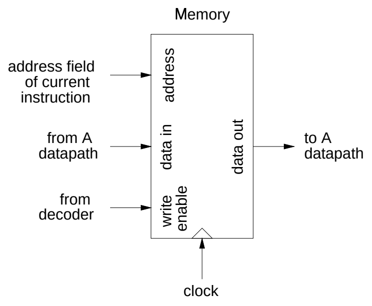
Memory

Define an entity and architecture for the memory. Use a selected assignment statement to implement the ROM portion and an array to implement the RAM portion.

The memory inputs are: data in (type `word`), address (type `address`), write enable, and clock.

The output is: data out (type `word`).

The memory interface is:



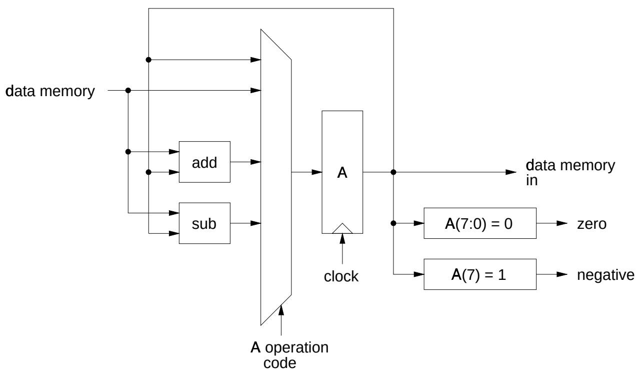
A Datapath

The A (accumulator) datapath updates A as instructed by the controller. A can be loaded with: (1) A, (2) the data memory output (during a LOAD instruction), (3) the result of adding or subtracting the data memory output and A.

The A datapath inputs are: the memory output (`word`), a signal to control the A input multiplexer, the clock.

The outputs are: A (`word`), a signal that indicates A is zero, a signal that indicates the MS bit of A is set (A is negative).

The structure of the A datapath is:



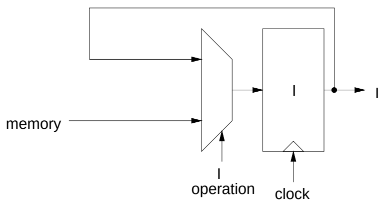
I Datapath

The I (instruction register) datapath updates I as instructed by the controller. I can be loaded with: (1) I, (2) the data memory output (during a fetch cycle).

The I datapath inputs are: the memory output (`word`), a signal to control the I input multiplexer, the clock.

The output is: I (`word`).

The structure of the I datapath is:



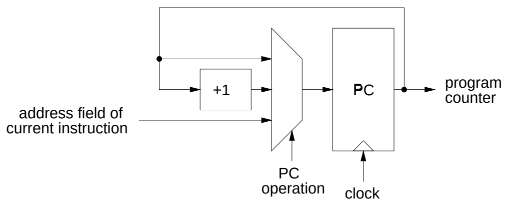
PC Datapath

The PC (program counter) datapath updates PC as instructed by the controller. PC can be loaded with: (1) PC, (2) PC+1 (to point to the next instruction), (3) the current instruction's address field (to branch to another instruction), or (4) zero (to reset the computer).

The PC datapath inputs are: the address field of I (type `address`), a signal to control the PC input multiplexer, a reset signal, the clock.

The output is: PC (5 bits).

The structure of the PC datapath is :



Controller (Instruction Decoder)

The instruction decoder is state machine whose outputs control the datapath multiplexers.

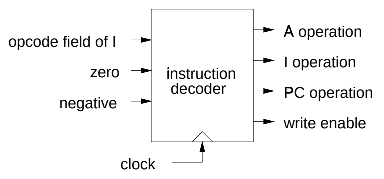
It's inputs are: the MS 3 bits of I, the zero and negative status signals from the A datapath, the clock.

It's outputs are: the signals to control the A datapath, the signals to control the I datapath, the signals to control the PC datapath, the memory write enable signal.

Design the controller by choosing controller states, controller outputs for each state, and the state machine state transition conditions so that the CPU fetches and executes the following instructions:

	code	description
LOAD	000	load - load A from the memory location addressed by the instruction address field
STORE	001	store - store A in the memory location addressed by the instruction address field
ADD	010	add - load A with the sum of A and the memory location addressed by the instruction address field
SUB	011	subtract - load A with the difference of A and the memory location addressed by the instruction address field
JMP	100	jump - load PC with the current instruction's address field
JNZ	101	jump on zero - if A is non-zero load PC with the current instruction's address field
JN	110	jump on negative - if A is negative load PC with the current instruction's address field

The interface of the controller is:



Assignment

Write VHDL descriptions for each of the five components.

The first 8 bytes of memory should contain the following program. This test program sets memory location 16 to 1, loads A with 3 and decrements A until it reaches zero. Then it goes into an infinite loop.

```
     Assembled
    Instruction   Mnemonic     80x86 syntax

  0  000 00101    LOAD   7         MOVE A,[7]
  1  001 10000    STORE 16         MOVE [16],A
  2  000 00110    LOAD   6         MOVE A,[6]
  3  011 10000    SUB   16     L1: SUB  A,[16]
  4  101 00011    JNZ    3         JNZ  L1
  5  100 00101    JMP    5     L2: JMP  L2
  6  000 00011           3         DB   3
  7  000 00001           1         DB   1
```

Assemble the above program into the CPU's machine language and use it to design the instruction memory. Unused locations should be set to zero.

Test each part of your design *separately*. As a minimum, demonstrate that:

- the memory gives the correct output for each program address input
- you can write 1, `0FFH 0EEH` in memory locations 16, 17 and 31 and read these values back
- A can hold, load, add, and subtract
- I can hold and load
- PC can hold, increment, load and reset
- the controller generates the correct sequence of outputs for each of the 7 instructions, including the fetch state and the two possible outputs for each conditional jump instruction

Write a top-level entity that combines the five components. This entity should have two inputs: reset and clock, and three outputs: PC, the memory output, A. Simulate the operation of your computer from reset until it executes two iterations of the infinite loop (11 instructions, 22 clock cycles).

For Part 1 of this assignment submit the VHDL code and simulation results for the memory, A datapath and I datapath. For Part 2 you will submit VHDL code and simulation results for the complete design (a total of 6 parts).

Hints

If you are using the Sun workstations to do this assignment you will need to ask Rob Ross (`robr@ee.ubc.ca`) to create a new directory for you where you can compile the project since your default disk quota is not large enough. This directory will not be backed up so you will be responsible for backing up the contents onto a floppy or your home directory.

The compiler's error message are sometimes vague as to the cause and location of the error. If the built-in help does not explain the cause of the problem you will have to remove (comment out) sections of the code to identify the error location and then try different alternatives until the error message disappears.