Assignment 3 - Appendix A
Hints for Assignment 3

Hints

A VHDL description for the RAM has been placed on the Web page. You may use this description if you add comments to it.

The design of this CPU allows one instruction to be executed per clock cycle. This means the instruction decoder is a combinational circuit, not a state machine.

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 work. This directory will not be backed up so you will be responsible for backing up the contents onto a floppy.

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 location and then try different alternatives until the error message disappears.

The MaxPlusII VHDL analyzer doesn't consider attributes of variables to be "locally static." For example, if `a` is signal of type `dword` you must use the expression: `a = conv_unsigned(0,dword'length)` rather than `a = conv_unsigned(0,a'length)`. Because of this bug you may include numeric constants in the code, e.g. `a = conv_unsigned(0,5)`.

Remember that VHDL is very strict about operand types and lengths. As stated above, you may not get meaningful error messages from the compiler. You will have to make frequent use of the `conv_integer()`, `conv_unsigned()`, `unsigned()`, and `std_logic_vector()` conversion functions. For example, the `and` and `not` operators are only defined on `std_logic_vector` types and you need to convert integer constants to `unsigned` constants in order to use the `unsigned` addition operator (+).

You may want to choose the types to be used for all of the entity inputs and outputs before you start coding.

Corrected Assembly-Language Code

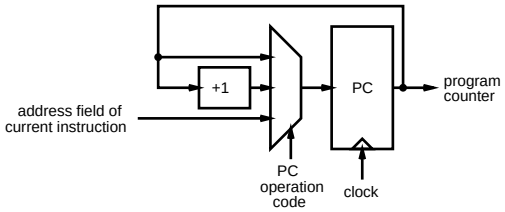
The program given in the assignment is wrong (the line numbers are wrong and the CPU has no JNZ instruction). The following listing is correct. It loads the accumulator with -3, increments it three times until it becomes non-negative and then loops forever on the last line. The code fragment shown below has been assembled into a format suitable for including in a selected assignment:

```
Instruction   Address   Opcode  Operand
"01000000" when 0,  --  LOADI   0
"00100000" when 1,  --  STORE   0
"01000001" when 2,  --  LOADI   1
"00100001" when 3,  --  STORE   1
"01000010" when 4,  --  LOADI   2
"10000000" when 5,  --  NOT
"01100001" when 6,  --  ADD     1
"11100110" when 7,  --  JN      6
"11001000" when 8,  --  JZ      8
```

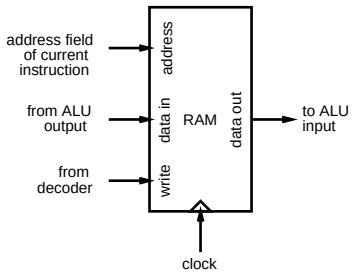
Data Path Diagrams

The following diagrams show the structure of the five entities in the computer. Although the same information is available in text form in the assignment, diagrams may make it easier to visualize the operation of the computer. The top-level entity (not shown) simply instantiates one instance of each of the other five blocks.

PC Datapath

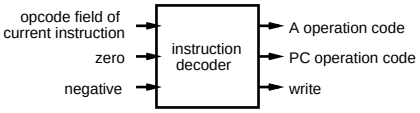


RAM



ROM

Controller



A Datapath