# COMP3211/9211 Computer Architecture
# 2004 S2 Laboratory Exercise 4 (VHDL4)
# Simple Pipelined Datapath (Draft 11/09)

## 1 Goal

This exercise is intended to get you on the road to implementing the pipelined datapath for Assignment 1. You will reuse modules you have previously designed to construct a small, pipelined datapath that comprises key elements of the datapath you will need to implement for your assignment.

If you would prefer to use your laboratory time to directly implement the pipeline you are designing for your assignment, that is also OK. This exercise is somewhat more gentle and guided, and provides you with a smaller design that you can discuss with your demonstrators as a complete unit.

## 2 Specification

The pipeline envisaged for this laboratory is depicted in Figure 1. Sequential 16-bit instructions (only ADD-type) are fetched by the Instruction Fetch Unit and stored in the Instruction Register. 4-bit addresses for the two operands to be added are presented to the Register File, from where the retrieved 8-bit data, A and B, are stored in the Operand Register. In the next stage, these operands are added using the 8-bit adder and the result is stored in the Result Register. In the fourth stage of operation, the result is written back to the Register File at the 4-bit address originally given in the instruction.

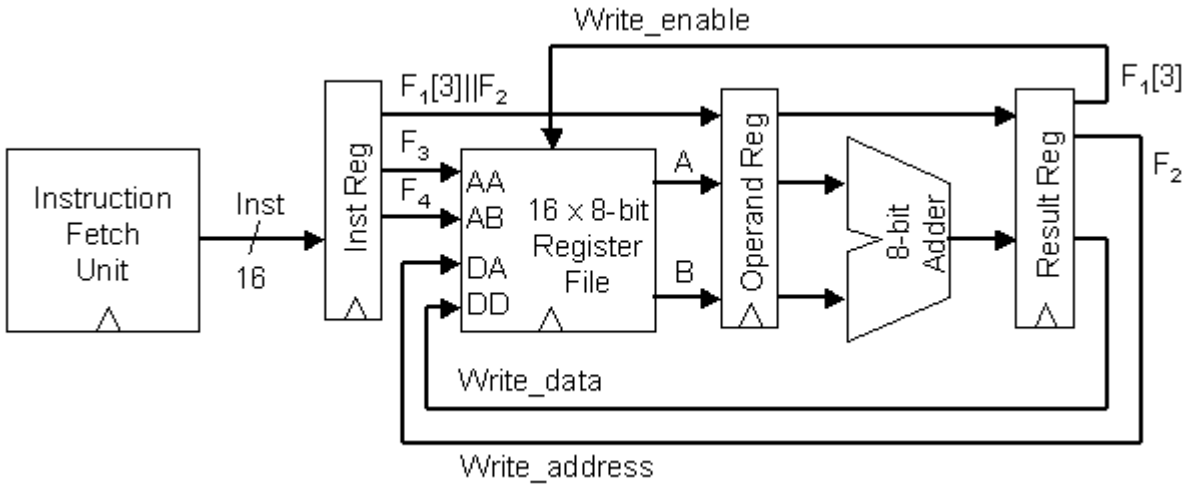


**Figure 1.** Simple addition pipeline

The Instruction Fetch Unit can be as specified for Lab 3, except that the Instruction Memory should be increased in width to 16-bits. Thus a 256 x 16-bit memory is called for. Instructions will consist of four 4-bit fields, $F_1F_2F_3F_4$. Respectively, from left to right, these fields signify the opcode, ADD = 1000, the 4-bit destination register number, the 4-bit A operand number, and the 4-bit B operand number. The branch logic can be disabled by setting `BraSat` to zero in the IFU design from Lab 3.

The Register File has three 4-bit address ports, AA, AB, and DA, two of which are for reading the 8-bit operands A and B, and one for writing the 8-bit result DD on a positive clock edge when the `Write_enable` is asserted. You can modify the memory module you investigated in Lab 1, or modify the more advanced Read/Write Memory module (rw_memory.zip) provided on the VHDL models link off the course web page. (This model allows you to write at the same time as reading from memory.)

The pipeline registers are clocked each cycle and store the data needed by downstream stages. In particular, the destination register number and the `Write_enable` bit (the most significant bit of the instruction) must be passed along the pipeline register chain.

## 3 Procedure

1. Modify the size of the Instruction Memory of the Instruction Fetch Unit (IFU) from Lab 3. Load the Instruction Memory with a sequence of instruction words that will allow you to test the design. Note the opcodes are all set to “1000”, but that the 4-bit source and destination register numbers comprising an instruction can be chosen at will.
2. Modify the memory component studied in Lab 1 or provided on the course web page (VHDL models link) to suit the purposes of the Register File. Initialize the Register file contents to interesting values that will allow you to test the functionality of the datapath. Storing zero in register number 0 and not over-writing this entry will allow you to check register contents by adding register 0 to the register whose contents you wish to inspect and writing the result back to its original location. Another approach is to extend the instruction set to include the opcode “0000”, which will perform the addition, but not store the result back to the register file.
3. Modify a register description to implement the Instruction Register, Operand Register, and Result Register. Note that it may help to provide a reset signal for these registers to initialise your datapath for testing.
4. Describe a structural implementation of the datapath using components from above and your adder from Lab 2.
5. Test your datapath by providing an initial reset and then manually toggling the clock for a sequence of cycles. Ensure that your simulation observes the contents of all significant buses.

## 4 Hand-in

There is no hand-in for this lab. Discuss your progress with your demonstrator before you leave the lab.