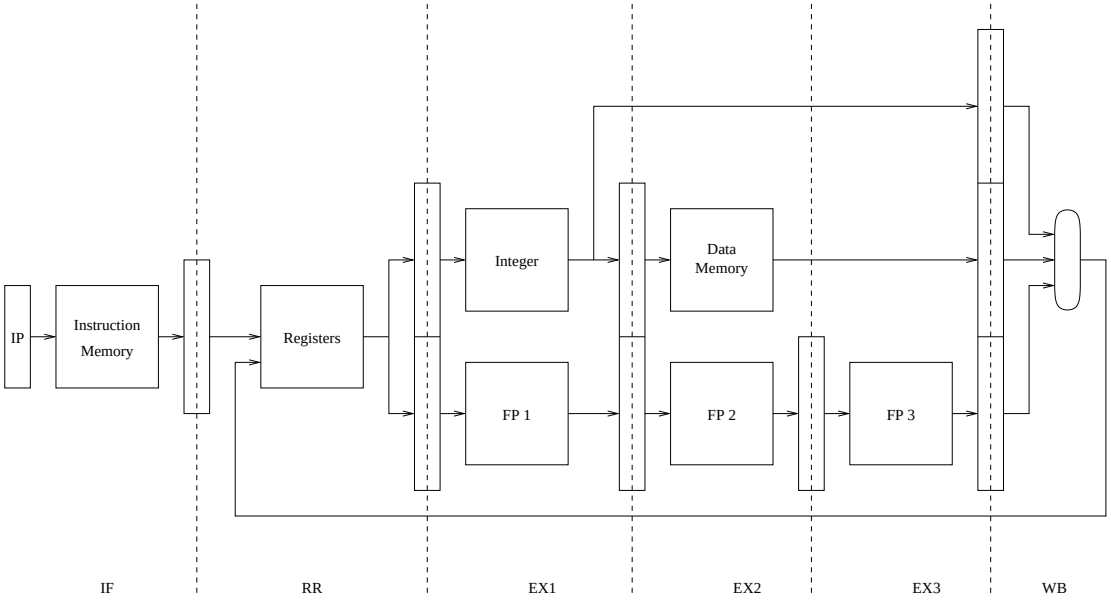
Architecture Quals Preview

11th July 2002

One of the questions on the Architecture exam will be based on the following pipeline and instruction set. Please take a look at this description in advance, so it is clear in your mind before taking the exam. Note: this only applies to a 30 point question! The rest of the exam is on other topics!

- It is a three-operand processor somewhat similar to a MIPS, except that it is capable of performing either integer or floating point operations, and the pipeline length varies according to the operation to be performed. Here's a highly schematic picture of the pipeline (it only shows the flow of instructions through the pipeline; it doesn't show all data paths):



  - There are three possible paths through the pipeline:
    - An integer instruction will use the IF, RR, EX1, and WB stages (it follows the top path through the pipeline).
    - A memory read or write will use the IF, RR, EX1, EX2, and WB stages (it follows the middle path).
    - A floating point instruction will use the IF, RR, EX1, EX2, EX3, and WB stages (it follows the bottom path).

    An instruction that does not write to a register needn't use the WB stage.
  - Instructions can stall in any of the pipeline registers. The segments of the pipeline registers are independent, so (for instance) an arithmetic instruction can be stalled after the Integer unit while a floating point instruction is proceeding after the FP3 unit.
  - The assembly language is likewise similar to the MIPS; instructions take the following form:
    - Integer and floating point instructions:
      `aluop result operand1 operand2`
      where `result` and `operand1` are the result register and a register containing the first operand, and `operand2` is either a register or a constant. Integer arithmetic instructions all begin with the letter `i` (as in

`iadd` for integer addition or `isub` for integer subtraction). Floating point arithmetic instructions all begin with the letter `f` (as in `fadd` for floating point addition or `fsub` for floating point subtraction).

* Memory instructions:
  `memop register1 index(register2)`
  where *`register1`* is either the destination register for a load or the source register for a store, and the memory address is formed by adding the *`index`* to the contents of *`register2`*. The memory instructions are `ld` (load) and `st` (store).