→ Implementation of classical 5-stage pipeline.

In order for the control circuit to work properly, the control signals must be set to the correct values at each stage for each instruction. The simplest way to do this is to extend the pipeline registers to include control information.

Data hazards can be solved by forwarding and have classifications:

- ID hazard: the contents of a register are required by a 'beq' and has been modified by a prior instruction and it's updated value is stored in a ID/EX, EX/MEM or MEM/WB pipeline register.
- EX hazard: the contents of a register are required by an instruction and it's updated value is in EX/MEM or MEM/WB pipeline register.
- MEM hazard: " "

" " "

" MEM/WB only pipeline register.

Not all data hazards can be solved with forwarding. There is still the need for stall when the instruction after a 'load' tries to read the same register which is written by it, which is only solved by a stall. If the next instruction is a [illegible] stall is needed. If it's a beq, two stalls

ALU, our [illegible]
are needed.

→ Exceptions

Control is the most challenging part of processor design. It needs allowing exceptions or interrupts other than branches and jumps.

These exceptions can be classified in 5 semi-independent categories:

- synchronous vs asynchronous: sync events occur at the same place every time the program is executed. Async may occur anywhere within the program and are handled after the execution of the current instruction.

- user requested vs coerced: user exceptions are predicted. Coerced are caused by hardware event that the program does not control.

- maskable vs non-maskable: some exceptions allow the program to choose the moment the hardware responds to them.

- within vs between instructions: an event may prevent the execution of the function (software or hardware level), and they are sync. Async that occur within instructions cause program termination.

- resume vs terminate: exceptions that do not require the program to resume often handled are easy to implement because there is no need to restart the program.

When an exception is serviced, the pipeline control must take the following steps to save the program state and allow resume:

1. Save PC value. All maskable exceptions at the same or lower priority are disabled.

2. All succeeding instructions and itself (if its within) are turned into no-op instructions. The others are allowed to complete.

3. When the exception service routine starts executing, the saved PC value gets updated to the correct one

4. After the execution, return from exception instructions restore the PC value and mode of execution.

Two main methods for communicating the reason of the exception:

- cause register: status register holding a field describing the cause for an exception.

- vectored exceptions: multiple entry point addresses for service of the exceptions. Each entry point address is associated with a particular exception.

→ Multicycle operations in classical 5-stage pipeline.

FP operations generally take more than one cycle to execute, as well as integer multiplication and divisions.

FP instructions have the same pipeline with 2 differences:

- the EX cycle may be repeated as many times needed.
- there may be multiple function units.

Four seperate functional units will be considered:

- main integer unit (basic ALU operations)
- FP and integer multiplier
- FP adder (additions, subtractions and data type conversions)
- FP and integer divider.

The following observations are in order:

→ FP division / integer division unit is not pipelined, structural hazards can occur.

→ not all instructions have the same execution time, the number of register writes in the same block may be larger.

→ data hazards are possible and more frequent

→ instructions completing in different order will give rise to problems when dealing with exceptions.

In certain cases of data hazards with there can be wrong results. For example, a branch followed by a division before the target, in a delayed branch scheme there