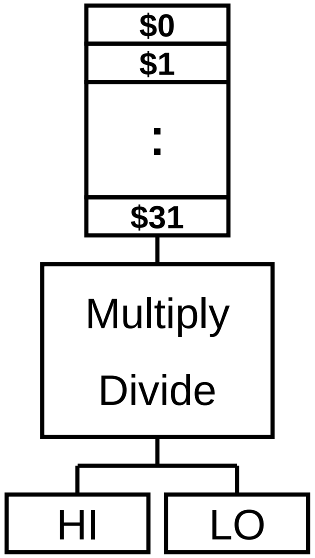
Extension: Multiplication & Division

Specifications

Extension Description

- Main Goal
  - Add a computation unit and control unit to support multiplication and division instructions
  - Add two special registers (**`$HI`** & **`$LO`**) for these instructions
- 4 New R-type Instructions
  - **`mult $rt $rs`** (**{$HI,$LO}=$rt*$rs**)
  - **`div $rt $rs`** (**$HI=$rt/$rs**, **$LO=$rt%$rs**)
  - **`mfhi $rd`** (**$rd=$HI**)
  - **`mflo $rd`** (**$rd=$LO**)



- **+define+MultDiv** in ncverilog simulation command

Instructions

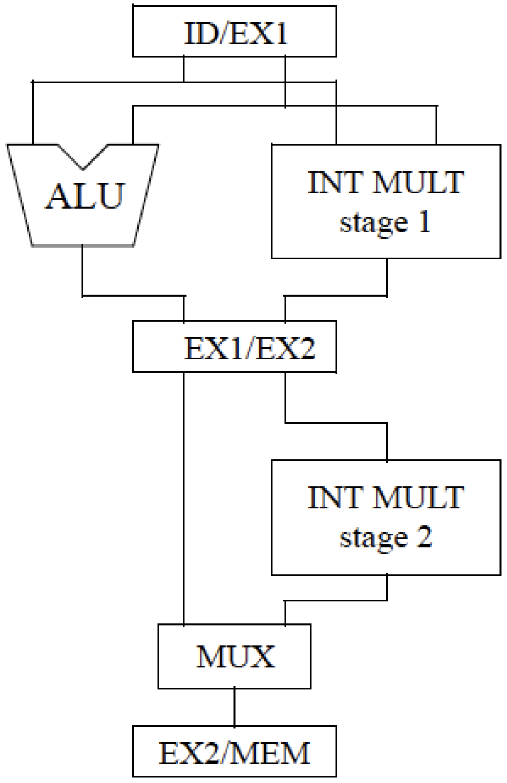
Instrcution	op/func	Meaning
mult \$rs \$rt	0/24	Multiply \$rs with \$rt, and store upper 32 bits in \$HI, lower 32 bits in \$LO
div \$rs \$rt	0/26	Divide \$rs by \$rt, and store the remainder in \$HI, the quotient in \$LO
mfhi \$rd	0/16	Move the data from \$HI to \$rd
mflo \$rd	0/18	Move the data from \$LO to \$rd

Architectures (1/2)

- There are two implementation styles
- 1. Iterative Approach
  - Use multiple cycles for ALU computation of `mult` and `div`, and one cycle for other ALU computation
  - During mult/div iterations, the successive instructions should be stalled to avoid hazards (Longer execution time)
  - Simple control signals for hazard free execution
  - Check "`IterMultDiv.pdf`" for more

Architectures (2/2)

- 2. Pipeline Approach
  - Use pipelined multiplication and division units
  - ALU stage will be further separated into multiple pipeline stages (for example, 2 stages of ALU in right figure)
  - No stalls during successive mult/div (Shorter execution time)
  - Complicated forwarding schemes



Comparison Metrics

- Base on the test program "I_mem_MultDiv"
- Score 1 (MD_S1): Total synthesis area (um^2)
  - MD_S1 = total area of MIPS core
- Score 2 (MD_S2): Total execution time (ns)
  - MD_S2 = total execution time of test program
- Score 3 (MD_S3): Minimum clock period (ns)
  - MD_S3 = clock period of MIPS core