* Code:
  + Code is distributed within three packages:
    - Initialization:
      * Contains initialization code
      * Initializes memory and registers
    - Pipeline stages:
      * Contains all the stages of pipeline.
    - Simulation:
      * Contains simulation process code
* Pipeline stages:
  + Fetch:
    - Takes the instruction starting from PC value starting from 4000
  + Decode:
    - Takes all the input instructions from fetch.
    - Sets sources and destinations
    - Checks for dependency
    - If dependency is present, then stalls next cycles
    - Checks whether data is available on forwarding bus
    - Checks which FU gets the instruction
  + Integer FU:
    - Calculates values according to instructions
    - Adds values to intFUBus
    - Makes it available for forwarding
  + Mul One stage:
    - Calculates MUL instruction
    - Adds values to mulFUBus
  + MUL Two stage:
    - Makes mulFUBus available for forwarding
    - Sends instruction to memory stage
  + Div One Stage:
    - Calculates Div instruction
    - Adds values to divFUBus
    - Sends instruction to div two
  + Div Two Stage:
    - Sends instruction to div thtree
  + Div Three Stage:
    - Sends instruction to div Four
  + Div Four Stage:
    - Makes divFUBus available for forwarding
  + Memory stage:
    - Stores value to memory location
    - Makes data forwarding unavailable
  + Writeback:
    - Writes the result to registers
    - Resets stall flag
* FLOW:
  + Initialize the registers and memory
  + Executes fetch stage
  + Execute decode stage
    - Checks is the register value is available and also checks if data can be forwarded from FUs last stages
    - If not available, stalls the instruction in decode stage until the required reg becomes available.
    - Set value of sources and destination.
  + Execute Int FU
    - Contains logic of all the instructions except MUL and DIV
    - Calculates the values
    - Makes it available for forwarding
    - Sends instruction to memory stage according to the priority.
  + Execute MUL one
    - Calculates the value using MUL logic
    - Sends instruction to MUL two
  + Execute MUL two
    - Sends instruction to memory stage according to the priority.
  + Execute Mem stage
    - Stores the memory
    - Sends instruction to writeback stage
  + Div One Stage:
    - Calculates Div instruction
    - Adds values to divFUBus
    - Sends instruction to div two
  + Div Two Stage:
    - Sends instruction to div thtree
  + Div Three Stage:
    - Sends instruction to div Four
  + Div Four Stage:
    - Makes divFUBus available for forwarding
    - Sends instruction to memory stage according to the priority
  + Execute writeback stage
    - Writes the calculated values to registers
    - Resets the stall flag
* Interlocking:
  + If the reg is a destination, make it unavailable and stall the stages until it is available
  + Make it available once it is being written in WB stage
* Data Forwarding:
  + Reg value is stored on forwarding bus in the last stage of FUs
  + In D/RF stage instruction checks if the reg value is available on forwarding bus
  + If available store that value in sources
  + If not available, wait till it clears WB stage.
* Registers:
  + 16 register are stored in an array
  + In the writeback stage the calculated values are stored in respective register.
* Forwarding Bus
  + Three forwarding buses gets values of registers in the last FU stage
* Instruction:
  + Instructions are fetched from a file given as a command line argument.
  + Used file reader and buffer reader.
  + Stored the instruction in a hashmap using the PC value as a key.
  + Also stored instruction numbers in a hashmap