**Pipeline Documentation:**

* **void** iplc\_sim\_dump\_pipeline()
  + Dumps the current contents of the pipeline
  + Utilizes switch case architecture
* **void** iplc\_sim\_push\_pipeline\_stage()
  + Checks whether various stages of the pipeline require stalls or forwarding
  + Checks writeback stage
  + Checks for branch and branch prediction
  + Checks for LW delays and data hits or misses
    - Adds delay cycles as necessary
  + Checks for SW memory access and data misses
    - Adds delay cycles as necessary
  + Increments number of pipeline cycles by 1
  + Pushes pipeline stages
  + Resets fetch stage to NOP
* **void** iplc\_sim\_process\_pipeline\_rtype(char \*instruction, int dest\_reg, int reg1, int reg2\_or\_constant)
  + Example implementation of pipeline process with type “r”
* **void** iplc\_sim\_process\_pipeline\_lw(int dest\_reg, int base\_reg, unsigned int data\_address)
  + Assigns destination register, base register, and data address arguments to fetch instruction variables
* **void** iplc\_sim\_process\_pipeline\_sw(int src\_reg, int base\_reg, unsigned int data\_address)
  + Assigns source register, base register and data address arguments to fetch instruction variables
* **void** iplc\_sim\_process\_pipeline\_branch(int reg1, int reg2)
  + Assigns both register arguments to fetch instruction variables
* **void** iplc\_sim\_process\_pipeline\_jump(char \*instruction)
  + Assigns instruction character string to fetch instruction variable
* **void** iplc\_sim\_process\_pipeline\_syscall()
  + Assigns fetch instruction variable to syscall instruction
* **void** iplc\_sim\_process\_pipeline\_nop()
  + Assigns fetch instruction variable to NOP instruction