FAQ

- I got the following error. What should I do?

common.h:8:18: error: zlib.h: No such file or directory

> The system that you are using does not have the gzip library. You have to install it.

- How do I check the correctness of my solution?

> You can print out trace information and you can check manually for the first few cycles. Later, We will provide a sample test and test output file so that you can double check.

- What is for the Op valid field?

> Op valid bit indicates when an op is valid or not. If you get an op from trace file, all ops are valid. For the first assignment, you probably have only all valid ops. Once you implement a speculative execution, you might have invalid ops.

- Can I add my header files or modify Makefile ?

> In order to grade your homework, we have to replace all your make files to something else. so please do not modify Makefile. Do not modify header files other than sim.h.

- How do I update PC value in the simulator?

> First, if you are asking about modeling the hardware, the simulator updates the pc latch when a branch instruction is in the MEM stage and the front end can start to fetch from the next clock cycle.
> If you are asking about how to set a correct PC value to get a correct op, you do not need to worry about that. Remember, this is a trace-driven simulator. Traces contain only the correct instructions. Whenever the simulator calls the get_op function, it always returns the next correct path information. so if the PC value is not correct for some reason such as branch misprediction which is in the later assignment, then the simulator should not call the get_op function. The front-end should wait until the correct PC value is available (i.e. until the branch is resolved) and then it starts to call the get_op function. For the first assignment, you do not worry about how to set a PC value. You just need to worry about when the simulator should call the get_op_function.

- I didn't modify my code at all and it generates "ERROR! OP_POOL SIZE is too small!!

".!!

You need to call free_op function in the retirement stage. If you do not call free_op function, the default simulator will generate this error message.

- I get "NO_OP_TYPE" assertion. What did I do wrong?
    - You might have not corrected your code. (sim_end_condition)
    - You might be running your code on different gcc version or 32-bit. Please check your code in killerbee machines
    - You might be doing something wrong with your op pointers.

- When there is a branch instruction, when can the front-end start to fetch a new instruction from the correct path?

    The front-end receives a new fetch address and the new PC address is stored in the PC latch at the end of the cycle when a branch instruction is at the MEM stage. Hence, at the following cycle, (i.e., when a branch instruction is in the WB stage), the front-end can fetch an instruction using the new updated PC address.

- When to update data hazard and control hazard?

    Whenever a pipeline is stalled due to data dependences and control dependences, you increment the corresponding counters in the ID stage

    The same instruction can increment both data hazard counter and control hazard counter.
    If one instruction has two data dependences (i.e., both source operands are dependent on other instructions), data hazard counter is only incremented by one.
    If the same instruction needs to wait for several cycles in the ID stage due to long latency operations, you still increment the data hazard counter multiple cycles. (i.e., while an instruction is waiting in the ID stage, the data hazard counter is continuously incremented.) To simplify the hardware counter design, you can increment the control hazard counter multiple times, if a branch instruction is waiting in the ID stage.

- Do I need to implement sim_end_condition?

    Yes, you must implement sim_end_condition.
    The success variable at "success = (gzread(stream, &trace_op, sizeof(Trace_op)) > 0);" tells the end of trace. If the success value is false, it means the end of trace.
    When the trace reaches the end, you must send an op that indicates the end of

trace by using some conditions such as (op->last_op = TRUE). If that op reaches WB_stage, the simulator sets sim_end_condition as TRUE.
Note that last_op field is not provided in the simulator frame. You must add such kind of field.

- Do I need to understand all the files?

  Please focus on sim.cpp sim.h file. trace.cpp is to generate traces and *knob* files are just to support knob features.

- What is Op_pool?

  To avoid of calling malloc all the time, op structures are already created.

- What is the difference between max_insts and max_cycles knobs?

  max_insts: a simulation ends after executing max_insts number of instructions.
  max_cycles: a simulation ends after executing max_cycles number of cycles.

- How do we handle instructions that take multiple cycles?

  EX_latch should not be updated until the execution is finished.
  FE_stage, ID_stage should be stalled during the additional cycle time.

- How can we check a branch instruction?

  if ((op->cf_type) >= CF_BR)

- Do we need to differentiate branch types?

  No. All branches are treated equally in this assignment.

- Do we need to use dcache_access, icache_access functions?

  No, you don't need to use them.

- Do store instructions also write to register files?

  Yes, Store instructions can write the register files.

- Can I use write_flg to check whether an instruction writes a value to a register file or not?

  No, write_flg is for EFLAGS in x86. You need to check op->dst to see whether there is a write access or not. if op->dst is "-1", the instruction does not write to a register file.

- Can I use windows programming? Or 32-bit machine?

  You could. But you need to make it sure your code compiles with g++4.1 and the specified testing machine. The trace format is written in a binary. That might cause some issues in other operating systems. So we do not grantee any support for other operating systems.