CS520 Computer Architecture

Project 1 – Spring 2022

Due date: 3/7/2022

1. RULES

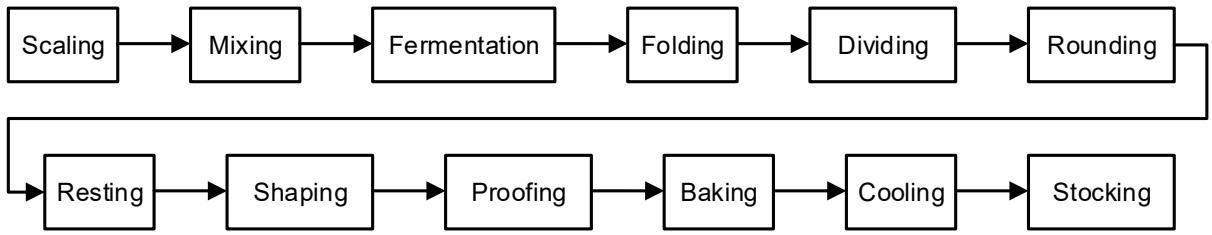
(1) All students must work alone. Cooperation is not allowed.
(2) Sharing of code between students is considered cheating and will receive appropriate action in accordance with University policy. The TAs will scan source code through various tools available to us for detecting cheating. Source code that is flagged by these tools will be dealt with severely.
(3) You must do all your work in the C/C++.
(4) Your code must be compiled on remote.cs.binghamton.edu or the machines in the EB-G7 and EB-Q22. This is the platform where the TAs will compile and test your simulator. They all have the same software environment.

2. Project Description

In this project, you will construct a baking pipeline simulator.

3. Baking Pipeline Simulator

The baking simulator has the following 12 stages.



Stage	Description
Scaling	All ingredients are measured and lined up in order of use.
Mixing	Ingredients are combined into a dough.
Fermentation	The dough is allowed to ferment.
Folding	The dough is folded to be degassed.
Dividing	The dough is divided into the desired individual portions.
Rounding	The portioned dough is loosely shaped into round balls.
Resting	The dough is rested for a while to relax the gluten
Shaping	The dough is formed into its final shape.
Proofing	The dough goes through one final fermentation.
Baking	The dough is baked. Note. The oven needs to be cleaned for 1 minute after every 10 baking.
Cooling	The loaves are cooled on racks.
Stocking	The baked bread is placed on shelves.

The baking pipeline receives one request at every minute. Suppose that every stage needs 1 minute except the baking stage. Note that the scaling stage takes a 10-minute break after every 1000 requests to prepare new materials. Also, the baking stage needs a 1-minute break after every ten baking. If the stage takes a break, the stage is stalled, and all the requests need the stage must wait until it becomes available again.

Unlike other stages, the baking stage needs 1 or 2 minutes, depending on the bread type. A bagel needs 1-minute baking, and a baguette needs 2-minute baking at the baking stage.

The no-request is there to inject a bubble into the pipeline. Thereby, even though the pipeline stage is stalled, the bubble still can proceed. For example, during the scaling stage's break, no-request operations can still be read from the trace file.

The simulator receives one request among three request types as follows every minute.

> No-Request
> Bake-Bagel
> Bake-Baguette

The request traces (trace1, trace2, trace3, and trace4) are already provided. You must complete your projects in the provided project01.c and project01.h files. **You cannot add extra files.**

4. Validation and Other Requirements

4.1. Validation requirements

Sample simulation outputs are provided. You must run your simulator and debug it until it matches the simulation outputs. Your simulator must print the final results correctly as follows.

Baking count:
- Bagel baking:
- Baguette baking:
No request:

How many minutes to bake:

Performance (bakes/minutes):

Your output must match both numerically and in terms of formatting, because the TAs will "diff" your output with the correct output. You must confirm correctness of your simulator by following these two steps for each program:

1) Redirect the console output of your simulator to a temporary file. This can be achieved by placing "> your_output_file" after the simulator command.

2) Test whether or not your outputs match properly, by running this unix command:
“diff –iw <your_output_file> <posted_output_file>”

The –iw flags tell “diff” to treat upper-case and lower-case as equivalent and to ignore the amount of whitespace between words. Therefore, you do not need to worry about the exact number of spaces or tabs as long as there is some whitespace where the sample outputs have whitespace. Both your outputs and final memory_map.txt must be the same as the solution.

3) Your simulator must run correctly not only with the given traces. Note that TA will validate your simulator with hidden traces.

4.2. Compiling and running simulator

You will hand in source code and the TA will compile and run your simulator. As such, you must be able to compile and run your simulator on machines in EB-G7 and EB-Q22. This is required so that the TAs can compile and run your simulator. You also can access the machine with the same environment remotely at remote.cs.binghamton.edu via SSH.

The pipeline receives a program name.

e.g. ./baking_sim traces/trace1 > results/traces/output_trace1

5. What to submit

You must hand in project1.c. Also, you must submit a cover page with the project title, the Honor Pledge, and your full name as electronic signature of the Honor Pledge. A cover page is posted on the project website.

6. Penalties

Various deductions (out of 100 points):
-5 points for each date late during the first 6 days.
-10 points for each date late after the first 6 days.

Up to -10 points for not complying with specific procedures. Follow all procedures very carefully to avoid penalties.

Cheating: Source code that is flagged by tools available to us will be dealt with according to University Policy. This includes a 0 for the project and other disciplinary actions.