CS/ECE 752 Spring 2025: HOMEWORK # 2
(Due by **12:30 PM** on **Friday, Feb 21**, via upload of PDF to Canvas)

Contact Rajesh Srivatsav (rsuresh6@wisc.edu)

1. Problem 1 (6 points)
Assume that in a particular program, 20% of the instructions are branches. The remaining 80% of the instructions has a CPI of 1.4. Consider a microarchitecture design A that does not have a branch predictor and has to expend 2 cycles for every branch instruction. Determine the CPI of design A.

Consider another design B that has a branch predictor with an accuracy of 92%. While a correct prediction (in B) requires only 1 cycle, a misprediction requires 3 cycles for the branch to execute. Determine the CPI of design B.

Consider yet another design C that has a branch predictor with an accuracy of 95%, and a branch execution time of 1 cycle for correct prediction and 5 cycles for misprediction. Determine the CPI of design C.

Rank the three designs in order from best to worst.

2. Problem 2 (14 points, 4+5+5)
Answer the following questions based on the code sequence given in Figure 1.

```
Loop: LD     F2,0(Rx)
I0:   MULTD  F2,F0,F2
I1:   DIVD   F8,F2,F0
I2:   LD     F4,0(Ry)
I3:   ADDD   F4,F0,F4
I4:   ADDD   F10,F8,F2
I5:   SD     F4,0(Ry)
I6:   ADDI   Rx,Rx,#8
I7:   ADDI   Ry,Ry,#8
I8:   SUB    R20,R4,Rx
I9:   BNZ    R20,Loop
```

Latencies beyond single cycle	
Memory LD	+3
Memory SD	+1
Integer ADD, SUB	+0
Branches	+1
ADDD	+2
MULTD	+4
DIVD	+10

Figure 1. Code and latencies for Problem 2

(a) What would be the baseline performance (in cycles per loop iteration) if no new instruction execution could be initiated until the previous instruction execution had completed and that only one instruction can be issued per cycle. Assume that the loop branch is taken and that execution does not stall for fetching an instruction.

(b) Now consider a multiple-issue design. Suppose you have two execution pipelines, each capable of beginning execution of one instruction per cycle, and enough fetch/decode bandwidth in the front end does not stall execution. Assume that results can be immediately forwarded from one execution unit to another, or to itself. Further assume that the only reason an execution pipeline would stall is to observe a true data dependence. Now how many cycles does it take to execute a loop?

(c) Unroll the loop once, merge the two iterations, and schedule the instructions to eliminate as many bubbles as possible. Now how many cycles does the loop take for the multiple issue design of part (b)?

3. Problem 3 (12 points) Introduction to *ChampSim*
ChampSim Github Repository **- *https://github.com/ChampSim/ChampSim***

Step 1: Cloning ChampSim

Running **git clone https://github.com/ChampSim/ChampSim** should create a directory named ChampSim within your current working directory, with all the files and directories. Then you should work from the newly created directory, while following the steps below.

Step 2: Installing Dependencies

Go through the README of the ChampSim Github repository. It primarily involves installation of vcpkg, which is used by ChampSim to manage dependencies.

NOTE: If you are using CSL machines (highly recommended), then the necessary dependencies may not be installed by default and you will have to install them. Installation **does not require** root privileges.

Step 3: Compile ChampSim

Specified in the README. This involves first setting up the configuration (specified via config file in JSON), followed by running the make command. Note that there exist defaults for all options specified via config file, but feel free to look at parameters that are easily configured via the json config file.

For this assignment, the most important parts are:

- downloading and building ChampSim,
- Setting up a simple configuration.
- how to run ChampSim
- how to parse the ChampSim output and understand the statistics, and
- using/modifying the default configuration file (**champsim.json**).

Step 4: Running Benchmarks on ChampSim

In this part, we will run a few applications from the SPEC 2017 benchmark suite on ChampSim using ***champsim.json*** config file.

- Change directories to your ChampSim directory.
- Running benchmark traces on ChampSim is specified in the README, mentioned again below. (run command from ChampSim directory)
  - `bin/champsim --warmup_instructions 2000000 --simulation_instructions 500000000 ~/path/to/traces/483.xalancbmk-127B.champsimtrace.xz`
- All SPEC 2017 traces are present in **/u/s/o/sohi/public/html/cs752/Spring2025/traces**

You will perform a simple characterization of one of
(a) 400.perlbench-41B.champsimtrace.xz [if your last name begins with A-I]
(b) 403.gcc-16B.champsimtrace.xz [if your last name begins with J-Q]
(c) 429.mcf-184B.champsimtrace.xz [if your last name begins with R-Z]

NOTE: champsim.json has lots of parameters relating to the processor core and the memory hierarchy which can be varied. Feel free to explore it.

Submit the following.

(a) Simulate using the default OOO CPU core with the following cache sizes (those cache sizes not mentioned here can be left as default size, for example LLC) and report the IPC, L1-Data Cache and L2 Cache Load miss rates for:

L1 instruction cache: 32KB 8-way set associative
L1 data cache: 32KB 2-way set associative
L2 cache: 256 KB 8-way set associative

(b) Simulate using the default OOO CPU core with the following cache sizes and report the IPC, L1-Data Cache and L2 Cache Load miss rates for:

L1 instruction cache: 16KB 4-way set associative
L1 data cache: 64KB 4-way set associative
L2 cache: 1MB 8-way set associative

(c) How much time (roughly) did the simulations take for (a) and (b)?

For all of the above tests, warmup for 1M instructions, and simulate for next 50M instructions. These options are specified in the command line used to run ChampSim on a particular trace.

Optional resources :

1) Paper **-** **https://arxiv.org/pdf/2210.14324.pdf**