18-640: Foundations of Computer Architecture

Project 1: Branch Prediction

Due: 11:59 pm EDT October 4, 2015

1. **Part 0 - Setup**

   Copy `/afs/ece.cmu.edu/class/ece640/project/project1/lab1.patch` in your gem5 folder and run,

   ```
   patch -p1 < lab1.patch
   ```

   This patch applies changes to the original release so that you get started with Lab1. Please make sure you see a list of file changes in `stdout`.

   ```
   scons build/X86/gem5.opt
   ```

   Build gem5 binary again, after the patch has been applied.

   ```
   bash benchmarks-compile.sh
   ```

   Compile benchmarks again, or you won't be able to use some of the benchmarks required by Lab1.

2. **Part 1 – Analysis of Tournament and Bi-mode Predictor: (15 Points)**

   Read in detail about the Alpha 21264 Tournament Branch Predictor and Bi-mode Predictor.

   Alpha 21264 Tournament Predictor:

   Kessler, Richard E. "The alpha 21264 microprocessor." *Micro, IEEE* 19.2 (1999): 24-36.

   Bi-mode Predictor:

   Lee, Chih-Chieh, I-CK Chen, and Trevor N. Mudge. "The bi-mode branch predictor." *Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on*. IEEE, 1997.

   Go through gem5 source code for predictors at `src/cpu/pred/*`. Please understand the various functionalities implemented and relate it with the theory you have read.

   Remember, we'll be using the simulator in syscall emulation (SE) mode for this project. Example command that you will be implementing:

   ```
   build/X86/gem5.opt configs/example/se.py -c binary [-o options] --cpu-type=detailed --caches --pred-type=tournament --local-pred-size=4096 –global-pred-size=8192 --choice-pred-size=8192 --local-ctr=2 --global-ctr=2 --choice-ctr=2
   ```

Explanation:

- `build/X86/gem5.opt` - gem5 binary
- `configs/example/se.py` - simulation script
- `-c binary - benchmark binary to run in SE mode`
- `-o options - cmd line options for the binary`
- `--cpu-type=detailed - specify cpu type (In Order, Out of Order etc.)`
- `--caches - caches enable`

We added following options to make it easier for you to change predictor type and their corresponding sizes at the command line, eliminating the need to recompile gem5 binary.

- --pred-type=tournament – branch predictor to be implemented
- --local-pred-size= local predictor size
- --global-pred-size= global predictor size
- --choice-pred-size= choice predictor size

Please see Appendix at the end of the handout for examples.

`BranchPredictor.py` describes the configuration parameters pertinent to branch prediction. You can configure type and size for each predictor here. Note that all the parameters are not used in each predictor.

After simulation, the stats will be generated in `m5out/` folder.

`.config` file is the configuration of the simulation performed

`.stat` file is the statistics generated at the end of simulation. You will need to extract info from this to analyze predictors.

Understand the `m5out/*.stat` file generated; following portion of the file is specifically generated for branch predictors:

```
system.cpu.branchPred.lookups                18211747
system.cpu.branchPred.condPredicted          17199049
system.cpu.branchPred.condIncorrect           6188168
system.cpu.branchPred.BTBLookups             10359986
system.cpu.branchPred.BTBHits                 8157880
system.cpu.branchPred.BTBCorrect                    0
system.cpu.branchPred.BTBHitPct             78.744122
system.cpu.branchPred.usedRAS                  174617
```

Calculating prediction rate:

```
100 - [(sys-
tem.cpu.branchPred.condIncorrect/system.cpu.branchPred.condPredicted
)*100]
```

IPC can be found by looking for `system.cpu.ipc`

Your Task:

- Simulate the *stringsearch* benchmark with various parameters according to the configuration table of the predictors.
- Report the branch prediction rate and IPC of your simulation on *stringsearch*.
- Analyse and report how the branch prediction rate varies for different configurations.

(Take number of counter bits as 2 for all data structures)

Configuration Table for Tournament Predictor:

	Config 1	**Config 2**	**Config 3**	**Config 4**
Local Predictor Size	2048	4096	4096	4096
Global Predictor Size	8192	4096	8192	8192
Choice Predictor Size	8192	8192	4096	8192

Configuration Table for Bimode Predictor:

	Config 1	**Config 2**	**Config 3**	**Config 4**
Global Predictor Size	2048	4096	8192	8192
Choice Predictor Size	4096	8192	4096	8192

Optional: We have provided two scripts, `bi-mode.sh` and `tournament.sh`. These scripts will help you try out different configurations (apart from the ones mentioned if you learn to modify the scripts by yourself) and other benchmarks to do better analysis.

```
bash bi-mode.sh
```

The above command runs simulation for a subset of Mibench benchmarks (mentioned in Part 3) for the above mentioned bi-mode configurations. The simulation results (runtime, prediction rate and ipc) will be present in tabular form in output `bi-mode.csv`.

```
bash tournament.sh
```

Similar to what `bi-mode.sh` does, the above command runs simulation of tournament predictor. The simulation results will be present in output `tournament.csv`.

3. Part 2 – Implement gshare and YAGS Predictor (25+25 Points):

In this part you will be implementing the gshare branch predictor and YAGS branch predictor. You should refer to the following papers:

Gshare:

Scott McFarling, “Combining Branch Predictors” - Technical Note, *http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-TN-36.pdf*

YAGS Predictor:

Eden, Avinoam N., and Trevor Mudge. "The YAGS branch prediction scheme."*Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture*. IEEE Computer Socie-ty Press, 1998.

We have added dummy files such as {`gshare.cc, gshare.hh`} and {`yags.cc,`

`yags.hh`} for you. You may take help from the existing predictors implemented in the distribution.

Your Task:

- Briefly, describe your predictor implementation design.
- Similar to the Part 1, analyze using *stringsearch* and report the results and analysis for the following configurations:

Gshare Predictor:

	Config – 1	Config – 2	Config – 3	Config – 4
Local Predictor Size	2048	4096	8192	16384

YAGS Predictor:

	Config 1	Config 2	Config 3	Config 4
Global Predictor Size	2048	4096	8192	8192
Choice Predictor Size	4096	8192	4096	8192

Note: Direct mapped cache for YAGS predictor would suffice for this part of the project.

4. Part 3 – Challenge (15 Points):

Improve your predictor (gshare and YAGS) keeping the configuration:

Gshare Predictor:

	Config
Local Predictor Size	4096

YAGS Predictor:

	Config
Global Predictor Size	4096
Choice Predictor Size	8192

You will be graded based on your Branch Prediction Rate (average of gshare and YAGS) relative to the rest of the class. It will be tested on the following set of MiBench benchmarks

Benchmarks for the challenge part

patricia	typeset	susan-smooth	susan-edge
susan-corner	gsm-toast	gsm-untoast	fft-inv

Run the command,

```
bash yags.sh
```

The above command runs simulation for yags predictor for the above mentioned subset of benchmarks and also for different configurations. The simulation results will be present in tabular form in output `yags.csv`.

Run the command,
`bash gshare.sh`

Similar to what `yags.sh` does, the above command runs simulation of gshare predictor. The simulation results will be present in output `gshare.csv`.

Specifically, if you are in the best quartile, you will receive the full 15 of 15 points. If you are in the second best quartile, you will receive 10 of 15 points. If you are in the third best quartile, you will receive 5 of 15 points. If you are in the lowest quartile, you will receive 0 of 15 points.

An exception is that you will receive the score of a better quartile if your branch misprediction rate (i.e 100 - prediction rate) is within 0.2% of the highest misprediction rate in that quartile.

Note:
You are allowed to deviate from the baseline design subject to the following restrictions.

- If its gshare, it should have a global history register and xor-ing operation, or if its YAGS, it should have taken-not taken cache, choice PHT, global history register and no other data structure.
- If you employ associativity, it must be 8-way or less.
- Extravagant logic functions (such as dividing/multiplying by non-constants or non-powers of 2) are not allowed.

If you have any doubt, feel free to ask us!

5. Questions (15 Points):

Answer the following questions.

1. On the basis of your analysis, which predictor has the highest prediction rate? And, why?
2. When is gshare more accurate than bimodal and why? Also, when is bimodal more accurate than gshare and why?
3. How does the length of history bits in gshare predictor affect the accuracy? Does increasing the length of history bits improve accuracy?
4. How does Yags predictor reduce destructive aliasing?
5. In Yags predictor, the size of the direction caches is tuned according to the overall size of the predictor. Small predictor sizes need small direction caches and large predictor sizes need large direction caches. Is the statement true or false? Also, explain the reason behind your answer.

6. Feedback (5 Points):

Please provide us with a short feedback for this lab.

7. Hand-in Details:

Note: Please use **pdf** only.

Part 3 is the optimized version of what you implemented in Part 2, so you just need to hand-in the optimized version.

You will need to hand-in

`src/cpu/pre/gshare.*`
`src/cpu/pred/yags.*` and
`report.pdf` to
`/afs/ece/class/ece640/submission/<andrew_id>/mygroup/project1/`

8. Appendix:

Sample list of commands you may try running: (Append size parameters to commands appropriately)

```
#network/patricia
build/X86/gem5.opt --stats-file=patricia.stat --dump-config= patricia.config
configs/example/se.py -c mibench/network/patricia/patricia -o
"mibench/network/patricia/small.udp" --
output=mibench/network/dijkstra/output_small.dat -- cpu-type=detailed –
caches


#automotive/typeset
build/X86/gem5.opt --stats-file= susan-smooth.stat --dump-config= susan-
smooth.config configs/example/se.py -c mibench/consumer/typeset/lout-
3.24/lout
-o "-I mibench/consumer/typeset/lout-3.24/include -D
mibench/consumer/typeset/lout-3.24/data -F mibench/consumer/typeset/lout-
3.24/font -C mibench/consumer/typeset/lout-3.24/maps -H
mibench/consumer/typeset/lout-3.24/hyph mibench/consumer/typeset/small.lout"
-- cpu-type=detailed --caches

#consumer/susan-smooth
build/X86/gem5.opt --stats-file= susan-smooth.stat --dump-config= susan-
smooth.config configs/example/se.py -c mibench/automotive/susan/susan -o
“mibench/automotive/susan/input_small.pgm
mibench/automotive/susan/output_small.smoothing.pgm -s" --cpu-type=detailed
-- caches

#consumer/susan-edge
build/X86/gem5.opt --stats-file= susan-smooth.stat --dump-config= susan-
smooth.config configs/example/se.py -c mibench/automotive/susan/susan -o
“mibench/automotive/susan/input_large.pgm
mibench/automotive/susan/output_large.edges.pgm -e" --cpu-type=detailed
– caches


#automotive/susan-corner
build/X86/gem5.opt --stats-file= susan-smooth.stat --dump-config= susan-
smooth.config configs/example/se.py -c mibench/automotive/susan/susan -o
“mibench/automotive/susan/input_large.pgm
mibench/automotive/susan/output_large.corners.pgm -c" --cpu-type=detailed
-- caches

#office/stringsearch
build/X86/gem5.opt --stats-file=strsearch.stat --dump-
config=strsearch.config configs/example/se.py -c
mibench/office/stringsearch/search_large --
output=mibench/office/stringsearch/output_large.txt --cpu-type=detailed
--caches


#telecomm/gsm-toast
build/X86/gem5.opt --stats-file= gsm-toast.stat --dump-config= gsm-
toast.config configs/example/se.py -c mibench/telecomm/gsm/bin/toast –o “-
fps -c mibench/telecomm/gsm/data/small.au” --cpu-type=detailed -- caches
```

```
#telecomm/gsm-untoast
build/X86/gem5.opt --stats-file= gsm-untoast.stat --dump-config= gsm-
untoast.config configs/example/se.py –c mibench/telecomm/gsm/bin/untoast –o
"-fps -c mibench/telecomm/gsm/data/small.au.run.gsm” --cpu-type=detailed –
caches

#telecomm/FFT-inv
build/X86/gem5.opt --stats-file=fft_inv.stat --dump- config=fft_inv.config
configs/example/se.py -c mibench/telecomm/FFT/fft
-o "4 8192 -i" --output=mibench/telecomm/FFT/output_small.inv.txt -- cpu-
type=detailed --caches
```