Computer Architecture

Assignment 1 - Report
Student Number - s1368177

Functionality for inputting what predictor to use.

For this extension, we added a parameter in line 72 (case 'b') of the file simulator.cc and added another argument in the function at line 93. (Can be located with the aid of comments in the file)

1. Always Not Taken Predictor

As the name suggests this predictor always predicts not taken and was implemented by default. Relocated to lines 47-54 in stages.cc because of the added functionality to input which predictor to use. (Procedure in comments)

2. Always Taken Predictor

As the name suggests this predictor always predicts taken and is exactly the opposite of the predictor mentioned above. This predictor is located in lines 55-64. Firstly, it checks if the instruction is a branch, if it is it predicts by default that the branch is taken, sets the program counter to the branch target address and moves on as normal.

3. 2 Bit Predictor

Implemented in:
Lines 65-91, 147-149, 243-266 and 309-326 of stages.cc
Lines 28 in cpu.h
Lines 40, 42-43 and 52-54 in stages.h

This predictor first initialises a table of 1024 entries all set to 0 that correspond to the states of the finite state machine (00, 01, 10, 11) represented as (0-strong not taken, 1-weak not taken, 2-weak taken, 3- strong taken) respectively in our implementation. Index for this is the program counter shifted by 3 (because instruction has last 3 bits as zeros) modulus 1024 because our table is 1024 long.

During Instruction Fetch
The predictor takes the index and makes a prediction based on what state the finite state machine is in (0,1 lead to not taken and 2,3 lead to taken) and updates the program counter / stores it in recovery as required.

During Instruction Decode
Recovery program counter, states of finite state machine, the index we're using is all carried over to be used in execution.

During Execution
If the branch was mis-predicted the following takes place depending on what state is it in:

- State 0 (Strong Not Taken) - If this was mis-predicted then that means that the branch was actually taken, set the program counter to target branch address and change the State to 1 (Weak Not Taken)
- State 1 (Weak Not Taken) - If this was mis-predicted then that means that the branch was actually taken, set the program counter to target branch address and change the State to 2 (Weak Taken)
- State 2 (Weak Taken) - If this was mis-predicted then that means that the branch was actually not taken, set the program counter to recovery program counter and change the State to 1 (Weak Not Taken)
- State 3 (Strong Taken) - If this was mis-predicted then that means that the branch was actually not taken, set the program counter to recovery program counter and change the State to 2 (Weak Not Taken)

If the branch was correctly predicted then the following takes place depending on the state:
- State 0 (Strong Not Taken) - Stays in the same state.
- State 1 (Weak Not Taken) - Move to Strong Not Taken.
- State 2 (Weak Taken) - Move to Strong Taken.
- State 3 (Strong Taken) - Stays in the same state.

4. 2 Level Predictor

Implemented in:
Lines 93-125, 147-150, 268-302 and 330-347 in stages.cc
Lines 29 in cpu.h
Lines 40-45 and 52-56 in stages.h

This predictor first initialises a table of 1024 entries all set to 0 that correspond to the states of the finite state machine (00, 01, 10, 11) represented as (0-strong not taken, 1-weak not taken, 2-weak taken, 3- strong taken) respectively in our implementation. The index is a global history buffer of the last 10 branch outcomes, also initialised to 0, represented by a 10 bit- bitset. (#include <bitset>) can be seen in multiple files for this.

During Instruction Fetch
Initially, the index is set to 0b0000000000 which means it accesses the first state in the table of states as mentioned in the 2Bit predictor and makes a prediction based on the state retrieved and then shifts the bit left by 1 and changes the last bit to 0 or 1 depending on the prediction. If it predicts not taken then shift it by 1 and let it be because shift includes a 0, update program counter and move on. If it predicts taken then store the updated program counter in a recovery program counter, left shift by 1, perform a logical OR operation by a single 1 to change the last bit to 1 and change the program counter to the target branch address. Before this, a copy of this buffer is made, incase our prediction was wrong later in the execution stage.

During Instruction Decode
Recovery program counter, states of finite state machine, the copy of the index we're using is all carried over to be used in execution.

During Execution
If the branch was mis-predicted the following takes place depending on what state it is in:
Use the index copy we made earlier to restore our global history buffer.
- State 0 (Strong Not Taken) - If this was mis-predicted then that means that the branch was actually taken, set the program counter to target branch address, Shift in a bit 1, denoting the branch was taken. and change the State (indexed by global history buffer) to 1 (Weak Not Taken)
- State 1 (Weak Not Taken) - If this was mis-predicted then that means that the branch was actually taken, set the program counter to target branch address, Shift in a bit 1, denoting the branch was taken. and change the State (indexed by global history buffer) to 2 (Weak Taken)
- State 2 (Weak Taken) - If this was mis-predicted then that means that the branch was actually not taken, set the program counter to recovery program counter, Shift in a bit 0, denoting the branch was not taken. and change the State (indexed by global history buffer) to 1 (Weak Not Taken)
- State 3 (Strong Taken) - If this was mis-predicted then that means that the branch was actually not taken, set the program counter to recovery program counter, Shift in a bit 0, denoting the branch was not taken and change the State (indexed by global history buffer) to 2 (Weak Not Taken)

If the branch was correctly predicted then the following takes place depending on the state:
- State 0 (Strong Not Taken) - Stays in the same state.
- State 1 (Weak Not Taken) - Move to Strong Not Taken.
- State 2 (Weak Taken) - Move to Strong Taken.
- State 3 (Strong Taken) - Stays in the same state.

The End.