

Assignment_Branch_Prediction

November 6, 2020

1 CS4200

2 Assignment 4 Branching

Total 100 points

This assignment is based off of one-bit and two-bit branch predictors. To simulate instructions and whether branches will occur or not, the provided methods `next_branch_outcome_random` and `next_branch_outcome_loop` will be used. These methods will simulate a completely random prediction outcome, and a set of outcomes that would more closely resemble a series of loops. A return of `True` represents taking a branch, and a `False` represents not taking a branch.

The class `Predictor` represents the predictor. It is best practice to set the initial state to 0.

```
In [ ]: from random import paretovariate
        from random import random

        def next_branch_outcome_loop():
            alpha = 2
            outcome = paretovariate(alpha)
            outcome = outcome > 2
            return outcome

        def next_branch_outcome_random():
            outcome = random()
            outcome = outcome > 0.5
            return outcome

        class Predictor:

            def __init__(self):
                self.state = 0

            def next_predict(self):
                """
                Use this method to return the prediction based off of the current
                state.
                """
                raise NotImplementedError("Implement this method")
```

```

def incorrect_predict(self):
    """
    Use this method to set the next state if an incorrect predict
    occurred. (self.state = next_state)
    """
    raise NotImplementedError("Implement this method")

def correct_predict(self):
    """
    Use this method to set the next state if an incorrect predict
    occurred. (self.state = next_state)
    """
    raise NotImplementedError("Implement this method")

```

2.1 One Bit Predictor

Complete the OneBitPredictor class by implementing the next_predict, incorrect_predict, and correct_predict methods. This instantiation will be used to compute the prediction accuracy. Use the next_predict method of the class to predict the next branch state. If the predict is incorrect, use the incorrect_predict method to set the next state. If the predict is correct, use the correct_predict method to set the next state.

```

In [ ]: class OneBitPredictor(Predictor):

    def next_predict(self):
        # YOUR CODE HERE
        raise NotImplementedError()

    def incorrect_predict(self):
        # YOUR CODE HERE
        raise NotImplementedError()

    def correct_predict(self):
        # YOUR CODE HERE
        raise NotImplementedError()

```

2.1.1 Random Branch Prediction

Use the next_branch_outcome_random method to generate branch outcomes. Use the previously implemented methods to compute a prediction rate. (25 points)

```

In [ ]: # YOUR CODE HERE
        raise NotImplementedError()

```

2.1.2 Loop Branch Prediction

Use the next_branch_outcome_loop method to generate branch outcomes. Use the previously implemented methods to compute a prediction rate. (25 points)

```
In [ ]: # YOUR CODE HERE
        raise NotImplementedError()
```

2.2 Two Bit Predictor

Complete the TwoBitPredictor class by implementing the next_predict, incorrect_predict, and correct_predict methods. This instantiation will be used to compute the prediction accuracy. Use the next_predict method of the class to predict the next branch state. If the predict is incorrect, use the incorrect_predict method to set the next state. If the predict is correct, use the correct_predict method to set the next state.

```
In [ ]: class TwoBitPredictor(Predictor):

        def next_predict(self):
            # YOUR CODE HERE
            raise NotImplementedError()

        def incorrect_predict(self):
            # YOUR CODE HERE
            raise NotImplementedError()

        def correct_predict(self):
            # YOUR CODE HERE
            raise NotImplementedError()
```

2.2.1 Random Branch Prediction

Use the next_branch_outcome_random method to generate branch outcomes. Use the previously implemented methods to compute a prediction rate. (25 points)

```
In [ ]: # YOUR CODE HERE
        raise NotImplementedError()
```

2.2.2 Loop Branch Prediction

Use the next_branch_outcome_loop method to generate branch outcomes. Use the previously implemented methods to compute a prediction rate. (25 points)

```
In [ ]: # YOUR CODE HERE
        raise NotImplementedError()
```

2.3 N-Bit Predictor

Extra credit: 30 points. Inherit the Predictor class and implement it's methods just like before. Now, implement an n-bit predictor that represents a higher confidence prediction.

```
In [ ]: # YOUR CODE HERE
        raise NotImplementedError()
```

2.3.1 Random Branch Prediction

Use the `next_branch_outcome_random` method to generate branch outcomes. Use the previously implemented methods to compute a prediction rate. (10 points)

```
In [ ]: # YOUR CODE HERE
        raise NotImplementedError()
```

2.3.2 Loop Branch Prediction

Use the `next_branch_outcome_loop` method to generate branch outcomes. Use the previously implemented methods to compute a prediction rate. (10 points)

```
In [ ]: # YOUR CODE HERE
        raise NotImplementedError()
```