Five Stage Pipeline:

Trace File	prediction_method=0 (cycles)	prediction_method=1 (cycles)	% Reduction in Cycles
sample1.tr	1233112	1128480	8.4851984
sample2.tr	1159167	1140907	1.5752691
sample3.tr	1278927	1268969	0.7786215
sample4.tr	3671198	3538348	3.6187098
sample_large1.tr	108044161	103703599	4.0173962
sample_large2.tr	119348777	115530363	3.1993742

Eight Stage Pipeline:

Trace File	prediction_method=0 (cycles)	prediction_method=1 (cycles)	% Reduction in Cycles
sample1.tr	3514261	3501066	0.3754701
sample2.tr	3131749	3119427	0.3934543
sample3.tr	3369978	3359077	0.3234739
sample4.tr	10015948	9979533	0.3635702
sample_large1.tr	318830015	315661622	0.9937562
sample_large2.tr	321386948	315481058	1.837626

On average, (considering prediction_method = 0) the number of cycles increases by a factor of 2.760 times when moving from five-stage to eight-stage. With prediction_method set to 1, this actually increases to a factor of 2.846 times since branch prediction is less effective on the eight-stage pipeline than it is on the five-stage. Considering numbers from both the eight-stage and five-stage architectures, branch prediction reduces the number of cycles by 2.163%. What's interesting here is that on the five-stage pipeline a branch predictor reduces the number of cycles by an average of 3.612% while on the eight-stage pipeline, the branch predictor only reduces the number of cycles by 0.715%. A branch predictor isn't even 1/5 as effective on the eight-stage pipeline as it is on the five-stage.

Below are the increase factors in cycles when moving from the five-stage to the eight-stage pipeline. For example, we interpret the first number as meaning that the eight-stage pipeline takes 2.850 times as many cycles to run trace file sample1.tr when prediction_method = 0.

Eight-stage (cycles)/five-stage (cycles)

sample1.tr

3514261 cycles / 1233112 cycles = 2.850 (prediction_method = 0)
3501066 cycles / 1128480 cycles = 3.102 (prediction_method = 1)

sample2.tr

3131749 cycles / 1159167 cycles = 2.702
3119427 cycles / 1140907 cycles = 2.734

sample3.tr

3369978 cycles / 1278927 cycles = 2.635
3359077 cycles / 1268969 cycles = 2.647

sample4.tr

10015948 cycles / 3671198 cycles = 2.728
9979533 cycles / 3538348 cycles = 2.820

sample_large1.tr

318830015 cycles / 108044161 cycles = 2.951
315661622 cycles / 103703599 cycles = 3.044

sample_large2.tr

321386948 cycles / 119348777 cycles = 2.693
315481058 cycles / 115530363 cycles = 2.731

From the results of the trace files, we see that the eight-stage pipeline uses approximately 2.803 times as many cycles as the five-stage pipeline. We reached this number by finding the average value when considering all the cycle increase factors from five-stage to eight-stage (including both prediction_method = 0 and prediction_method = 1).

Since we are assuming the clock frequency of the eight-stage pipeline is double that of the five-stage pipeline, the efficiency of both programs can be calculated as follows:

Let x = clock frequency on the five-stage pipeline (cycles/second)

Let y = the number of cycles needed to run a program on the five-stage pipeline (cycles)

We can calculate the time per program with the following equation:

time per program (s) = y (cycles/s) / x (cycles)

five_stage.c:

clock frequency = x cycles/second (by definition of x);

# cycles = y cycles (by definition of y);

time per program = y/x seconds (by equation defined above);

Eight_stage.c:

clock frequency = 2x cycles/second (definition of eight-stage pipeline clock frequency in project description);

# cycles = 2.803y cycles (by calculated value of average factor of increase in cycles from five-stage to eight-stage above)

time per program = 2.803y/2x seconds = 1.402y/x seconds (by equation defined above);

From calculations performed above, we can see that the eight-stage design runs, on average, approximately 1.402 times longer than the five-stage pipeline. This leads us to our conclusion that even with twice the clock frequency, the eight-stage design is still less efficient than the five-stage design. Therefore, we recommend use of the five-stage architecture over the eight-stage architecture.

Branch Prediction Table Changes:

Sample 1

Normal

Simulation terminates at cycle :

Size 32

Simulation terminates at cycle : 1081

Size 64

Simulation terminates at cycle : 1081

Size 128

Simulation terminates at cycle : 1081

Sample 2

- Normal
    - Simulation terminates at cycle :
- Size 32
    - Simulation terminates at cycle : 1145276
- Size 64
    - Simulation terminates at cycle : 1140907
    - Improvement: 4369
- Size 128
    - Simulation terminates at cycle : 1138972
    - Improvement: 1935

Sample 3

- Normal
    - Simulation terminates at cycle :
- Size 32
    - Simulation terminates at cycle : 1283147
- Size 64
    - Simulation terminates at cycle : 1268969
    - Improvement: 14178
- Size 128
    - Simulation terminates at cycle : 1264178
    - Improvement: 4791

Sample 4

Normal

Simulation terminates at cycle :

Size 32

Simulation terminates at cycle : 3589406

Size 64

Simulation terminates at cycle : 3538348

Improvement: 51058

Size 128

Simulation terminates at cycle : 3530615

Improvement: 7733

Large Sample 1

Normal

Simulation terminates at cycle :

Size 32

Simulation terminates at cycle : 105402700

Size 64

Simulation terminates at cycle : 103703599

Improvement: 1699101

Size 128

Simulation terminates at cycle : 103703513

Improvement: 86

Large Sample 2

- Normal
  - Simulation terminates at cycle :
- Size 32
  - Simulation terminates at cycle : 116974556
- Size 64
  - Simulation terminates at cycle : 115530363
  - Improvement: 1444193
- Size 128
  - Simulation terminates at cycle : 115116953
  - Improvement: 413410