BPU: examples

E. Sanchez, M. Sonza Reorda

Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)

Torino - Italy

BHT Example

Considering a 2-bit saturating counter BHT of 1K entries; and assuming that the processor executes the following code fragment, determine the BHT final state and calculate the misprediction ratio for the following case.

```
for(i=1;i<=100;i++){
  read_data();
  if (aa==2)
     aa = 0;
  if (bb==2)
     bb = 0;
  if (aa == bb)
  {…}
}
```

BHT Example

Considering a 2-bit saturating counter BHT of 1K entries; and assuming that the processor executes the following code fragment, determine the BHT final state and calculate the misprediction ratio for the following case.

```
for(i=1;i<=100;i++){
  read_data();
  if (aa==2)
     aa = 0;
  if (bb==2)
     bb = 0;
  if (aa == bb)
  {…}
}
```

```
L0:       …
          DADDUI  R3, R1, #-2
          BNEZ  R3, L1
          DADD  R1, R0, R0
L1:       DADDUI  R3, R2, #-2
          BNEZ  R3, L2
          DADD  R2, R0, R0
L2:       DSUB  R3, R1, R2
          BNEZ  R3, L3
          …
L3:       …
          DADDI R4, R4, #-1
          BNEZ R4, L0
          …
```

BHT

aa is stored in R1
bb is stored in R2

**Consider ounter BHT of 1K entries;
and assu or executes the following
code fragment, determin BHT final state and calculate
the misprediction ratio for th ollowing case.**

```
for(i=1;i<=100;i++){
  read_data();
  if (aa==2)
     aa = 0;
  if (bb==2)
     bb = 0;
  if (aa == bb)
  {…}
}
```

```
L0:     …
        DADDUI  R3, R1, #-2
        BNEZ  R3, L1
        DADD  R1, R0, R0
L1:     DADDUI  R3, R2, #-2
        BNEZ  R3, L2
        DADD  R2, R0, R0
L2:     DSUB  R3, R1, R2
        BNEZ  R3, L3
        …
L3:     …
        DADDI R4, R4, #-1
        BNEZ R4, L0
        …
```

BHT Example

Considering a 2-bit saturating counter BHT of 1K entries; and assuming that the processor executes the following code fragment, determine the BHT final state and calculate the misprediction ratio for the following case.

Case 1:

- **The program inputs for `aa` and `bb` are always different than 2.**

Iteration 1

```
aa = 4
bb = 3
```

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	00	NT	
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	00	NT	
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	00	NT	
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	00	NT	
0x0040		…	00	NT	

Iteration 1

```
aa = 4
bb = 3
```

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	01	NT	1
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	00	NT	
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	00	NT	
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	00	NT	
0x0040		…	00	NT	

Iteration 1

```
aa = 4
bb = 3
```

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	01	NT	1
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	01	NT	1
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	00	NT	
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	00	NT	
0x0040		…	00	NT	

Iteration 1

```
aa = 4
bb = 3
```

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	01	NT	1
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	01	NT	1
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	01	NT	1
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	01	NT	1
0x0040		…	00	NT	

Iteration 2

```
aa = 4
bb = 3
```

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	10	NT	2
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	10	NT	2
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	10	NT	2
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	10	NT	2
0x0040		…	0	NT	

Iteration 3

```
aa = 4
bb = 3
```

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	11	T	2
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	11	T	2
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	11	T	2
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	11	T	2
0x0040		…	00	NT	

Iteration 4

```
aa = 4
bb = 3
```

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	11	T	2
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	11	T	2
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	11	T	2
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	11	T	2
0x0040		…	00	NT	

Iteration 100

aa = 4
bb = 3

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	11	T	2
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	11	T	2
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	11	T	2
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	10	T	3
0x0040		…	00	NT	

BHT Example – Case 1

Considering a 2-bit saturating counter BHT of 1K entries; and assuming that the processor executes the following code fragment, determine the BHT final state and calculate the misprediction ratio for the following case.

Case 1:

- The program inputs for `aa` and `bb` are always different than 2.

Misprediction ratio =

Number of mispredicted branches / Total branches

Misprediction ratio = 9 / 400

= 2.25%

BHT Example - Case 2

Considering a 2-bit saturating counter BHT of 1K entries; and assuming that the processor executes the following code fragment, determine the BHT final state and calculate the misprediction ratio for the following case.

Case 2:

- The program inputs for `aa` and `bb` alternate values different than 2 and equal to 2.

Case 2 - Iteration 1

```
aa = 4
bb = 3
```

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	00	NT	
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	00	NT	
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	00	NT	
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	00	NT	
0x0040		…	00	NT	

Case 2 - Iteration 1

```
aa = 4
bb = 3
```

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	01	NT	1
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	01	NT	1
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	01	NT	1
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	01	NT	1
0x0040		…	00	NT	

Case 2 - Iteration 2

```
aa = 2
bb = 2
```

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	00	NT	1
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	00	NT	1
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	00	NT	1
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	10	NT	2
0x0040		…	00	NT	

Case 2 - Iteration 3

```
aa = 4
bb = 3
```

address	Instruction	BHT	Prediction	misP. counter
0x0000	L0: …	00	NT	
…	…	00	NT	
0x0010	DADDUI R3, R1, #-2	00	NT	
0x0014	BNEZ R3, L1	01	NT	2
0x0018	DADD R1, R0, R0	00	NT	
0x001C	L1: DADDUI R3, R2, #-2	00	NT	
0x0020	BNEZ R3, L2	01	NT	2
0x0024	DADD R2, R0, R0	00	NT	
0x0028	L2: DSUB R3, R1, R2	00	NT	
0x002C	BNEZ R3, L3	01	NT	2
0x0030	…	00	NT	
0x0034	L3: …	00	NT	
0x0038	DADDI R4, R4, #-1	00	NT	
0x003C	BNEZ R4, L0	11	T	2
0x0040	…	00	NT	

Case 2 - Iteration 4

```
aa = 2
bb = 2
```

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	00	NT	2
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	00	NT	2
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	00	NT	2
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	11	T	2
0x0040		…	00	NT	

Case 2 - Iteration 100

aa = 2
bb = 2

address	Instruction		BHT	Prediction	misP. counter
0x0000	L0:	…	00	NT	
…		…	00	NT	
0x0010		DADDUI R3, R1, #-2	00	NT	
0x0014		BNEZ R3, L1	00	NT	50
0x0018		DADD R1, R0, R0	00	NT	
0x001C	L1:	DADDUI R3, R2, #-2	00	NT	
0x0020		BNEZ R3, L2	00	NT	50
0x0024		DADD R2, R0, R0	00	NT	
0x0028	L2:	DSUB R3, R1, R2	00	NT	
0x002C		BNEZ R3, L3	00	NT	50
0x0030		…	00	NT	
0x0034	L3:	…	00	NT	
0x0038		DADDI R4, R4, #-1	00	NT	
0x003C		BNEZ R4, L0	10	T	3
0x0040		…	00	NT	

BHT Example - Case 2

Considering a 2-bit saturating counter BHT of 1K entries; and assuming that the processor executes the following code fragment, determine the BHT final state and calculate the misprediction ratio for the following case.

Case 2:

- **The program inputs for `aa` and `bb` alternate values different than 2 and equal to 2.**

Misprediction ratio =

Number of mispredicted branches / Total branches

Misprediction ratio = 153 / 400

= 38.25%

(2,2) Example

Considering a (2,2) correlating predictor of 1K entries, and assuming that the processor executes the following code fragment, determine the (2,2) final state and calculate the misprediction ratio for the following case.

```
for(i=1;i<=100;i++){
  read_data();
  if (aa==2)
     aa = 0;
  if (bb==2)
     bb = 0;
  if (aa == bb)
  {…}
}
```

```
L0:     …
        DADDUI  R3, R1, #-2
        BNEZ  R3, L1
        DADD  R1, R0, R0
L1:     DADDUI  R3, R2, #-2
        BNEZ  R3, L2
        DADD  R2, R0, R0
L2:     DSUB  R3, R1, R2
        BNEZ  R3, L3
        …
L3:     …
        DADDI R4, R4, #-1
        BNEZ R4, L0
        …
```

(2,2) Example

Considering a (2,2) correlating predictor of 1K entries; and assuming that the processor executes the following code fragment, determine the (2,2) final state and calculate the misprediction ratio for the following case.

Case 1:

- The program inputs for `aa` and `bb` alternate values different than 2 and equal to 2.

(2,2) - Iteration 1

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	00	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	00	00	00	00		
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	00	00	00		
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	00		
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	00	00	00	00		
0x0040	…	00	00	00	00		

(2,2) - Iteration 1

```
aa = 3
bb = 4
```

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	00	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	01	00	00	00	01	1
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	00	00	00		
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	00		
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	00	00	00	00		
0x0040	…	00	00	00	00		

(2,2) - Iteration 1

```
aa = 3
bb = 4
```

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	00	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	01	00	00	00	01	1
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	01	00	00	11	1
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	00		
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	00	00	00	00		
0x0040	…	00	00	00	00		

(2,2) - Iteration 1

```
aa = 3
bb = 4
```

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	00	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	01	00	00	00	01	1
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	01	00	00	11	1
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	01	11	1
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	00	00	00	01	11	1
0x0040	…	00	00	00	00		

(2,2) - Iteration 2

```
aa = 2
bb = 2
```

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	11	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	01	00	00	00	10	1
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	01	00	00	00	1
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	01	00	1
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	01	00	00	01	01	2
0x0040	…	00	00	00	00		

(2,2) - Iteration 3

```
aa = 3
bb = 4
```

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	01	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	01	01	00	00	11	2
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	01	00	01	11	2
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	10	11	2
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	01	00	00	10	11	3
0x0040	…	00	00	00	00		

(2,2) - Iteration 4

aa = 2
bb = 2

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	11	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	01	01	00	00	10	2
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	01	00	01	00	2
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	10	00	2
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	10	00	00	10	01	4
0x0040	…	00	00	00	00		

(2,2) - Iteration 5

```
aa = 3
bb = 4
```

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	01	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	01	10	00	00	11	3
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	01	00	10	11	3
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	11	11	2
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	10	00	00	11	11	4
0x0040	…	00	00	00	00		

(2,2) - Iteration 6

```
aa = 2
bb = 2
```

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	11	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	01	10	00	00	10	3
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	01	00	10	00	3
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	11	00	2
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	11	00	00	11	01	4
0x0040	…	00	00	00	00		

(2,2) - Iteration 7

```
aa = 3
bb = 4
```

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	01	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	01	11	00	00	11	3
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	01	00	11	11	3
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	11	11	2
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	11	00	00	11	11	4
0x0040	…	00	00	00	00		

(2,2) - Iteration 8

```
aa = 2
bb = 2
```

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	11	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	01	11	00	00	10	3
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	01	00	11	00	3
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	11	00	2
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	11	00	00	11	01	4
0x0040	…	00	00	00	00		

(2,2) - Iteration 100

aa = 2
bb = 2

address	Instruction	00	01	10	11	2-bit shift reg	misP. counter
0x0000	L0: …	00	00	00	00	11	
…	…	00	00	00	00		
0x0010	DADDUI R3, R1, #-2	00	00	00	00		
0x0014	BNEZ R3, L1	01	11	00	00	10	3
0x0018	DADD R1, R0, R0	00	00	00	00		
0x001C	L1: DADDUI R3, R2, #-2	00	00	00	00		
0x0020	BNEZ R3, L2	00	01	00	11	00	3
0x0024	DADD R2, R0, R0	00	00	00	00		
0x0028	L2: DSUB R3, R1, R2	00	00	00	00		
0x002C	BNEZ R3, L3	00	00	00	11	00	2
0x0030	…	00	00	00	00		
0x0034	L3: …	00	00	00	00		
0x0038	DADDI R4, R4, #-1	00	00	00	00		
0x003C	BNEZ R4, L0	10	00	00	11	00	5
0x0040	…	00	00	00	00		

(2,2) Example

Considering a (2,2) correlating predictor of 1K entries; and assuming that the processor executes the following code fragment, determine the (2,2) final state and calculate the misprediction ratio for the following case.

Case 1:

- The program inputs for `aa` and `bb` alternate values different than 2 and equal to 2.

Misprediction ratio =

Number of mispredicted branches / Total branches

Misprediction ratio = 13 / 400

= 3.25%