Computer Architectures
Exam of 28.2.2024 - part I

First name, Last name, ID.....

A - Question #2

Let's consider a MIPS architecture using a *Branch History Table* (BHT) composed of *8 1-bit entries*. Let's assume that this architecture executes the following code: it counts the number of values in the vector *vec* that are equal to or multiples of 2 and 3 and then writes the results into the variables *mul2* and *mul3*, respectively. The calculation of the modulus, an operation used here to understand whether a value is a multiple of another, is done using the Barrett reduction, which can be described as follows:

$$a \bmod n = a - [a/n]n$$

In the code the module is calculated twice: the first time to check if the previously loaded value is a multiple of 2, the second time to check if it is a multiple of 3. In general, *a* is the value loaded from *vec* and *n* is first 2 and then 3. **Please note that some numbers may be multiples of both values.**
For every instruction, the hexadecimal address of the memory cell storing the instruction is reported. Assuming that before executing the code fragment the BHT is full of null values (corresponding to the prediction Not Taken), you are asked to compute:

- The number of mispredicted branches during the execution of the code.
- The BHT content when the execution finishes (using the third table).

For all computations, it is suggested to use the two tables on the next page. Write in the highlighted cells whether the result of the prediction of the current branch and the real behavior (result) of the software is *Taken* (T) or *Not Taken* (NT). Then, report the results on the third table.
Hint: To calculate the BHT entry corresponding to each branch instruction, remember that you should exclude the last two bits from the instruction address as they are always equal to 0.

ADDR	CODE		
	.data		
	vec:	.byte 15, 12, 9, 8, 6, 4, 3, 2	# input vector
	mul2:	.space 1	# number of values equal to or multiples of 2
	mul3:	.space 1	# number of values equal to or multiples of 3
	.text		
0x0000		daddui r1, r0, 2	# initialize the first value of n (2)
0x0004		daddui r2, r0, 3	# initialize the second value of n (3)
0x0008		daddui r3, r0, 8	# initialize the value used as a comparator
0x000c		daddui r4, r0, 0	# initialize the pointer
0x0010		daddui r5, r0, 0	# initialize the counter of values equal to or multiples of 2
0x0014		daddui r6, r0, 0	# initialize the counter of values equal to or multiples of 3
0x0018	cyc:	lbu r7, vec(r4)	# load an element from *vec*
0x001c		ddiv r8, r7, r1	# Barrett reduction (modulus calculation), *n = 2*
0x0020		dmulu r8, r8, r1	# Barrett reduction (modulus calculation), *n = 2*
0x0024		dsubu r8, r7, r8	# Barrett reduction (modulus calculation), *n = 2*
0x0028		bnez r8, m3	# jump to *m3* if the modulus is not equal to zero
0x002c		daddui r5, r5, 1	# increment the counter of values equal to or multiples of 2
0x0030	m3:	ddiv r8, r7, r2	# Barrett reduction (modulus calculation), *n = 3*
0x0034		dmulu r8, r8, r2	# Barrett reduction (modulus calculation), *n = 3*
0x0038		dsubu r8, r7, r8	# Barrett reduction (modulus calculation), *n = 3*
0x003c		bnez r8, nxt	# jump to *nxt* if the modulus is not equal to zero
0x0040		daddui r6, r6, 1	# increment the counter of values equal to or multiples of 3
0x0044	nxt:	daddui r4, r4, 1	# increment the pointer
0x0048		bne r3, r4, cyc	# condition for exiting the cycle
0x004c	term:	sb r5, mul2(r0)	# store the number of values equal to or multiples of 2
0x0050		sb r6, mul3(r0)	# store the number of values equal to or multiples of 3
0x0054		halt	# termination of the program

Computer Architectures
Exam of 28.2.2024 - part I

First name, Last name, ID.....

Address	Code	BHT	Iteration #1		Iteration #2		Iteration #3		Iteration #4		Iteration #5	
		entry #	pred	result	pred	result	pred	result	pred	result	pred	result
0x0000	daddui r1, r0, 2											
0x0004	daddui r2, r0, 3											
0x0008	daddui r3, r0, 8											
0x000c	daddui r4, r0, 0											
0x0010	daddui r5, r0, 0											
0x0014	daddui r6, r0, 0											
0x0018	cyc: lbu r7, vec(r4)											
0x001c	ddiv r8, r7, r1											
0x0020	dmulu r8, r8, r1											
0x0024	dsubu r8, r7, r8											
0x0028	bnez r8, m3	2	NT	T	T	NT	T	T	T	NT	T	NT
0x002c	daddui r5, r5, 1											
0x0030	m3: ddiv r8, r7, r2											
0x0034	dmulu r8, r8, r2											
0x0038	dsubu r8, r7, r8											
0x003c	bnez r8, nxt	7	NT	NT	NT	NT	NT	NT	NT	T	T	NT
0x0040	daddui r6, r6, 1											
0x0044	nxt: daddui r4, r4, 1											
0x0048	bne r3, r4, cyc	2	T	T	NT	T	T	T	NT	T	NT	T
0x004c	term: sb r5, mul2(r0)											
0x0050	sb r6, mul3(r0)											
0x0054	halt											

Address	Code	BHT	Iteration #6		Iteration #7		Iteration #8		Iteration #9		Iteration #10	
		entry #	pred	result	pred	result	pred	result	pred	result	pred	result
0x0000	daddui r1, r0, 2											
0x0004	daddui r2, r0, 3											
0x0008	daddui r3, r0, 8											
0x000c	daddui r4, r0, 0											
0x0010	daddui r5, r0, 0											
0x0014	daddui r6, r0, 0											
0x0018	cyc: lbu r7, vec(r4)											
0x001c	ddiv r8, r7, r1											
0x0020	dmulu r8, r8, r1											
0x0024	dsubu r8, r7, r8											
0x0028	bnez r8, m3	2	T	NT	T	T	T	NT				
0x002c	daddui r5, r5, 1											
0x0030	m3: ddiv r8, r7, r2											
0x0034	dmulu r8, r8, r2											
0x0038	dsubu r8, r7, r8											
0x003c	bnez r8, nxt	7	NT	T	T	NT	NT	T				
0x0040	daddui r6, r6, 1											
0x0044	nxt: daddui r4, r4, 1											
0x0048	bne r3, r4, cyc	2	NT	T	T	T	NT	NT				
0x004c	term: sb r5, mul2(r0)											
0x0050	sb r6, mul3(r0)											
0x0054	halt											

Computer Architectures
Exam of 28.2.2024 - part I

First name, Last name, ID.....

The number of mispredicted branches during the execution of the code is: ___15___

BHT - Final content

Entry 0	0	Entry 4	0
Entry 1	0	Entry 5	0
Entry 2	0	Entry 6	0
Entry 3	0	Entry 7	1

Computer Architectures
Exam of 28.2.2024 - part I

First name, Last name, ID.....

B - Question #2

Let's consider a MIPS architecture using a *Branch History Table* (BHT) composed of *8 1-bit entries*. Let's assume that this architecture executes the following code: it counts the number of values in the vector *vec* that are equal to or multiples of 2 and 3 and then writes the results into the variables *mul2* and *mul3*, respectively. The calculation of the modulus, an operation used here to understand whether a value is a multiple of another, is done using the Barrett reduction, which can be described as follows:

$$a \bmod n = a - [a/n]n$$

In the code the module is calculated twice: the first time to check if the previously loaded value is a multiple of 2, the second time to check if it is a multiple of 3. In general, *a* is the value loaded from *vec* and *n* is first 2 and then 3. **Please note that some numbers may be multiples of both values.**
For every instruction, the hexadecimal address of the memory cell storing the instruction is reported. Assuming that before executing the code fragment the BHT is full of null values (corresponding to the prediction Not Taken), you are asked to compute:

- The number of mispredicted branches during the execution of the code.
- The BHT content when the execution finishes (using the third table).

For all computations, it is suggested to use the two tables on the next page. Write in the highlighted cells whether the result of the prediction of the current branch and the real behavior (result) of the software is *Taken* (T) or *Not Taken* (NT). Then, report the results on the third table.
Hint: To calculate the BHT entry corresponding to each branch instruction, remember that you should exclude the last two bits from the instruction address as they are always equal to 0.

ADDR	CODE		
	.data		
	vec:	.byte 3, 4, 6, 15, 2, 8, 12, 9	# input vector
	mul2:	.space 1	# number of values equal to or multiples of 2
	mul3:	.space 1	# number of values equal to or multiples of 3
	.text		
0x0000		daddui r1, r0, 2	# initialize the first value of n (2)
0x0004		daddui r2, r0, 3	# initialize the second value of n (3)
0x0008		daddui r3, r0, 8	# initialize the value used as a comparator
0x000c		daddui r4, r0, 0	# initialize the pointer
0x0010		daddui r5, r0, 0	# initialize the counter of values equal to or multiples of 2
0x0014		daddui r6, r0, 0	# initialize the counter of values equal to or multiples of 3
0x0018	cyc:	beq r3, r4, term	# condition for exiting the cycle
0x001c		lbu r7, vec(r4)	# load an element from *vec*
0x0020		ddiv r8, r7, r1	# Barrett reduction (modulus calculation), *n = 2*
0x0024		dmulu r8, r8, r1	# Barrett reduction (modulus calculation), *n = 2*
0x0028		dsubu r8, r7, r8	# Barrett reduction (modulus calculation), *n = 2*
0x002c		bnez r8, m3	# jump to *m3* if the modulus is not equal to zero
0x0030		daddui r5, r5, 1	# increment the counter of values equal to or multiples of 2
0x0034	m3:	ddiv r8, r7, r2	# Barrett reduction (modulus calculation), *n = 3*
0x0038		dmulu r8, r8, r2	# Barrett reduction (modulus calculation), *n = 3*
0x003c		dsubu r8, r7, r8	# Barrett reduction (modulus calculation), *n = 3*
0x0040		bnez r8, nxt	# jump to *nxt* if the modulus is not equal to zero
0x0044		daddui r6, r6, 1	# increment the counter of values equal to or multiples of 3
0x0048	nxt:	daddui r4, r4, 1	# increment the pointer
0x004c		j cyc	# next cycle
0x0050	term:	sb r5, mul2(r0)	# store the number of values equal to or multiples of 2
0x0054		sb r6, mul3(r0)	# store the number of values equal to or multiples of 3
0x0058		halt	# termination of the program

Computer Architectures
Exam of 28.2.2024 - part I

First name, Last name, ID.....

Address	Code	BHT	Iteration #1		Iteration #2		Iteration #3		Iteration #4		Iteration #5	
		entry #	pred	result	pred	result	pred	result	pred	result	pred	result
0x0000	daddui r1, r0, 2											
0x0004	daddui r2, r0, 3											
0x0008	daddui r3, r0, 8											
0x000c	daddui r4, r0, 0											
0x0010	daddui r5, r0, 0											
0x0014	daddui r6, r0, 0											
0x0018	cyc: beq r3, r4, term	6	NT	NT	NT	NT	NT	NT	NT	NT	NT	NT
0x001c	lbu r7, vec(r4)											
0x0020	ddiv r8, r7, r1											
0x0024	dmulu r8, r8, r1											
0x0028	dsubu r8, r7, r8											
0x002c	bnez r8, m3	3	NT	T	T	NT	NT	NT	NT	T	T	NT
0x0030	daddui r5, r5, 1											
0x0034	m3: ddiv r8, r7, r2											
0x0038	dmulu r8, r8, r2											
0x003c	dsubu r8, r7, r8											
0x0040	bnez r8, nxt	0	NT	NT	NT	T	T	NT	NT	NT	NT	T
0x0044	daddui r6, r6, 1											
0x0048	nxt: daddui r4, r4, 1											
0x004c	j cyc											
0x0050	term: sb r5, mul2(r2)											
0x0054	sb r6, mul3(r2)											
0x0058	halt											

Address	Code	BHT	Iteration #6		Iteration #7		Iteration #8		Iteration #9		Iteration #10	
		entry #	pred	result	pred	result	pred	result	pred	result	pred	result
0x0000	daddui r1, r0, 2											
0x0004	daddui r2, r0, 3											
0x0008	daddui r3, r0, 8											
0x000c	daddui r4, r0, 0											
0x0010	daddui r5, r0, 0											
0x0014	daddui r6, r0, 0											
0x0018	cyc: beq r3, r4, term	6	NT	NT	NT	NT	NT	NT	NT	T		
0x001c	lbu r7, vec(r4)											
0x0020	ddiv r8, r7, r1											
0x0024	dmulu r8, r8, r1											
0x0028	dsubu r8, r7, r8											
0x002c	bnez r8, m3	3	NT	NT	NT	NT	NT	T				
0x0030	daddui r5, r5, 1											
0x0034	m3: ddiv r8, r7, r2											
0x0038	dmulu r8, r8, r2											
0x003c	dsubu r8, r7, r8											
0x0040	bnez r8, nxt	0	T	T	T	NT	NT	NT				
0x0044	daddui r6, r6, 1											
0x0048	nxt: daddui r4, r4, 1											
0x004c	j cyc											
0x0050	term: sb r5, mul2(r2)											
0x0054	sb r6, mul3(r2)											
0x0058	halt											

Computer Architectures
Exam of 28.2.2024 - part I

First name, Last name, ID.....

The number of mispredicted branches during the execution of the code is: ___10___

BHT - Final content

Entry 0	0	Entry 4	0
Entry 1	0	Entry 5	0
Entry 2	0	Entry 6	1
Entry 3	1	Entry 7	0