Computer Architectures
Exam of 4.7.2023 - part I

First name, Last name, ID..

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, which, taking a *vec* vector, calculates the number of even and odd characters it contains with respect to the relative index of the ASCII table (for example, the character '*a*' in the ASCII table is *61* and is therefore odd, the character '*b*' is *62* and is therefore even, and so on) and then writes the result into the variables *res0* and *res1*. The calculation of the remainder is performed using the following formula: *a mod n = a – (a / n) * n*. For every instruction, the hexadecimal address of the memory cell storing the instruction is reported.
Assuming that when the execution of 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 table reported on the next page).

For all computations, it is suggested the usage of the table 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).
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.

```
vec:     . asciiz   "coding"          # input vector with the termination character
rese:    .space     1                 # counter of even elements
reso:    .space     1                 # counter of odd elements
…
0x0000              daddui r1, r0, 2  # initialize the divisor
0x0004              daddi r2, r0, -1  # initialize the pointer
0x0008              daddui r5, r0, 0  # initialize the counter of even elements
0x000c              daddui r6, r0, 0  # initialize the counter of odd elements
0x0010   cyc:       daddui r2, r2, 1  # increment the pointer
0x0014              lb r3, vec(r2)    # load an element from vec
0x0018              beqz r3, term     # check if the loaded char is the termination one
0x001c              daddu r4, r3, r0  # create a copy of the previously loaded char
0x0020              ddivu r3, r3, r1  # calculate the remainder - step 1: a / n
0x0024              dmulu r3, r3, r1  # calculate the remainder - step 2: (a / n) * n
0x0028              dsubu r4, r4, r3  # calculate the remainder - step 3: a – (a / n) * n
0x002c              bnez r4, nxt      # check if the remainder is not equal to zero (odd char)
0x0030              daddui r5, r5, 1  # increment the counter of even elements
0x0034              j cyc             # next cycle
0x0038   nxt:       beqz r4, cyc      # check if the remainder is equal to zero (even char)
0x003c              daddui r6, r6, 1  # increment the counter of odd elements
0x0040              j cyc             # next cycle
0x0044   term:      sb r5, rese(r0)   # store the result
0x0048              sb r6, reso(r0)   # store the result
0x004c   halt                         # termination of the program
```

Computer Architectures
Exam of 4.7.2023 - part I

First name, Last name, ID..

Address	Code	BHT	Iteration #1		Iteration #2		Iteration #3		Iteration #4	
		entry #	prediction	result	prediction	result	prediction	result	prediction	result
0x0000	daddui r1, r0, 2									
0x0004	daddi r2, r0, -1									
0x0008	daddui r5, r0, 0									
0x000c	daddui r6, r0, 0									
0x0010	cyc: daddui r2, r2, 1									
0x0014	lb r3, vec(r2)									
0x0018	beqz r3, term	6	NT	NT	NT	NT	NT	NT	NT	NT
0x001c	daddu r4, r3, r0									
0x0020	ddivu r3, r3, r1									
0x0024	dmulu r3, r3, r1									
0x0028	dsubu r4, r4, r3									
0x002c	bnez r4, nxt	3	NT	T	T	T	T	NT	NT	T
0x0030	daddui r5, r5, 1									
0x0034	j cyc									
0x0038	nxt: beqz r4, cyc	6	NT	NT	NT	NT			NT	NT
0x003c	daddui r6, r6, 1									
0x0040	j cyc									
0x0044	term: sb r5, rese(r0)									
0x0048	sb r6, reso(r0)									
0x004c	halt									

Address	Code	BHT	Iteration #5		Iteration #6		Iteration #7		Iteration #8	
		entry #	prediction	result	prediction	result	prediction	result	prediction	result
0x0000	daddui r1, r0, 2									
0x0004	daddi r2, r0, -1									
0x0008	daddui r5, r0, 0									
0x000c	daddui r6, r0, 0									
0x0010	cyc: daddui r2, r2, 1									
0x0014	lb r3, vec(r2)									
0x0018	beqz r3, term	6	NT	NT	NT	NT	NT	T		
0x001c	daddu r4, r3, r0									
0x0020	ddivu r3, r3, r1									
0x0024	dmulu r3, r3, r1									
0x0028	dsubu r4, r4, r3									
0x002c	bnez r4, nxt	3	T	NT	NT	T				
0x0030	daddui r5, r5, 1									
0x0034	j cyc									
0x0038	nxt: beqz r4, cyc	6			NT	NT				
0x003c	daddui r6, r6, 1									
0x0040	j cyc									
0x0044	term: sb r5, rese(r0)									
0x0048	sb r6, reso(r0)									
0x004c	halt									

The number of mispredicted branches during the execution of the code is: ___6___

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