QUESTION 1:

A RISC CPU has an instruction pipeline with 3 stages:

- I (Fetch): Fetches and decodes instruction
- A (ALU): reads registers and performs ALU operation, calculates branch target address, makes branch decission
- D (Data): writes the result of ALU operation or data from memory to registers, or writes data to memory.

```
START:
      LDSU  (R0)$500, R10    ; R10 <-- A (16-bit)
      ADD   R0, 0, R11       ; R11 <-- 0
      LDSU  (R0)$502, R11    ; R11 <-- B (16-bit)
      ADD   R0, 0, R12       ; R12 <-- 0 (C)
LOOP:
      ADD   R10, R12, R12    ; R12 <--R12+A
      SUB   R11, 1, R11      ; Decrement R11
      JMPR  BHI, LOOP        ; If higher than zero
      ADD   R0,R0,R10        ;R10 <-- 0 (Clear A)
      STL   (R0)$504,R12     ;M[504]<-- Result (C)
```

The given program on the above performs a multiplication **(C=AxB)** and clears the registers that hold the source operands **A** and **B**. **A** starts in memory at the address **$500**, **B** at **$502**, and **C** at **$504**. Remember **R0** of the CPU always contains zero.

a) Examine the given program by drawing the timing diagram of the 3-stage pipeline. Is there any data conflict?

b) Explain the control (branch) conflict encountered in the given program. What happens if this branch conflict is not solved?

c) Find a software solution to the branch problem without decreasing the performance of the pipeline. Do not change the algorithm.

INSTRUCTION SET:

ADD	Rs,S2,Rd	*Rd <-- Rs* + S2	
SUB	Rs,S2,Rd	*Rd <-- Rs* - S2	
LDSU	(Rs)S2,Rd	*Rd <-- M[Rs + S2]*	Short unsigned
STL	(Rs)S2, Rm	*M[Rs + S2] <-- Rm*	Store long
JMPR	COND,Y	*PC <-- PC + Y*	Jump relative

a)

	1	2	3	4	5	6	7	8	9	10	11	12
LDSU (R0)$500,R10	I	A	D									
ADD R0,0,R11		I	A	D								
LDSU (R0)$502,R11			I	A	D							
ADD R0,0,R12				I	A	D						
NOOP					I	A	D					
LOOP: ADD R10,R12,R12						I	A	D				
SUB R11,1,R11							I	A	D			
JMPR BHI,LOOP								I	A	D		
ADD R0,R0,R10									I	A	D	
STL (R0)$504,R12										I	A	D

There is a data conflict between these lines, ADD R0,0,R12 and ADD R10,R12,R12. First instruction wants to write R12 in its D stage, while the second one wants to read R12 in its A stage at the same clock cycle if we did not put a NOOP operation between them. As in the slide 2.39, there would be no conflict if data was written in the first half of the cycle (rising edge) and read in the second half (falling edge).

ADD R0,0,R12 ve ADD R10,R12,R12 satırları arasında veri bağımlılığı vardır. Birincisi D'de sonucu R12'ye yazacak, ikincisi A'da okumaya çalışacak. Bu iki katman ayrı saat darbelerinde olmalı. Derste anlatıldığı gibi (2.39) eğer çıkan kenarda saklayıcılara yazılıp inen kenarda okunuyorsa bu durumda çatışma olmaz.

b) During the execution of the branch (JMPR), next instruction (ADD) is taken to the pipeline. Therefore, this ADD instruction is also executed in each iteration of the loop if the condition of the JMPR is true.
If this branch conflict is not solved, register R10 will be cleared in every run of the loop and the program will generate wrong result.

Dallanma komutu yürütülürken, iş hattına sıradaki komut (ADD) alınmaktadır. Bu nedenle koşul sağlandığı halde her döngüde bu komut da yürütülmektedir.
Eğer bu sorun çözülmezse R10 saklayıcısı her döngüde sıfırlandığından program yanlış sonuç üretecektir.

c) In order to solve this problem without decreasing the performance of the pipeline, “ADD R10,R12,R12” can be moved after JMPR instruction.

Sorunu çözmek için “ADD R10,R12,R12”, JMPR komutundan sonraya taşınır.

QUESTION 2:

A CPU with an instruction pipeline runs the loop given below. The loop is designed using two branch instructions: one conditional branch (BRZ ”Branch if zero”) and one unconditional branch instruction (BRA “Branch always”).

```
        Counter <-- 10 ;            Loop iterates 10 times
LOOP: ….
        ….
        Counter <-- Counter - 1;    Decrement the counter
        BRZ  OUT;                   Branch if zero to OUT
        ….
        BRA LOOP;                   Branch always to LOOP
OUT:    ….
        ….
```

a. If the prediction mechanism is “dynamic prediction with one bit”, what should the initial value of the prediction bit be to increase the performance of the pipeline? What are the total numbers of correct predictions and mispredictions with this initial value? Explain briefly.
b. If the prediction mechanism is “dynamic prediction with two bits”, what should the initial values of the prediction bits be to increase the performance of the pipeline? What are the total numbers of correct predictions and mispredictions with these initial values? Explain briefly

SOLUTION 2:

a)

Note: Prediction mechanisms are applied only to conditional branch instructions. For an unconditional branch instruction (BRA), which branches always, a prediction is not necessary.

Dikkat: *Öngörü yöntemleri sadece koşullu dallanma komutlarında uygulanırlar. Her zaman dallanan koşulsuz dallanma (BRA) komutunda zaten öngörü gereksiz ve anlamsızdır.*

BRZ OUT: If the initial decision is to take the branch (p = 1), there will be a misprediction in the first iteration. Then, p will be changed to 0. In the last iteration, there will be a misprediction because p = 0. There are mispredictions in the first and last iterations. Other predictions are correct.

İlk ve son yinelemelerde yanlış öngörü olur; diğer öngörüler doğrudur.

Correct (*Doğru*): 8 Incorrect (*yanlış*): 2

BRZ OUT: If the initial decision is not to take the branch (p = 0) , there will be a misprediction only in the last iteration Other predictions are correct.

Sadece son yinelemede yanlış öngörü olur; diğer öngörüler doğrudur.

Correct (*Doğru*): 9 Incorrect (*yanlış*): 1

CONCLUSION: We would prefer "not taking the branch" (p = 0). The number of mispredictions is 1.

SONUÇ: *Başlangıç durumu olarak "dallanma yok" (p = 0) tercih edilir. Hata sayısı 1 olur.*

b)

BRZ OUT: If the initial value of the prediction bits is 11 (possibility to take the branch is VERY high), there will be mispredictions in the first and second iterations, because the prediction is changed after two mispredictions. There will also be a misprediction in the last iteration. Other predictions are correct.

Birinci ve ikinci yinelemelerde yanlış öngörü olur, çünkü bu yöntemde karar iki hatadan sonra değişir. Ek olarak son yinelemede de yanlış öngörü olur; diğer öngörüler doğrudur.

Correct (*Doğru*): 7 Incorrect (*yanlış*): 3

BRZ OUT: If the initial value of the prediction bits is 10 (possibility to take the branch is high), there will be mispredictions in the first iteration. There will also be a misprediction in the last iteration. Other predictions are correct.

Birinci ve son yinelemelerde yanlış öngörü olur; diğer öngörüler doğrudur.

Correct (*Doğru*): 8 Incorrect (*yanlış*): 2

BRZ OUT: If the initial decision is not to take the branch (00 or 01), there will be a misprediction only in the last iteration. Other predictions are correct.

Sadece son yinelemede yanlış öngörü olur; diğer öngörüler doğrudur.

Correct (*Doğru*): 9 Incorrect (*yanlış*): 1

CONCLUSION: We would prefer "not taking the branch" (00 or 01). The number of incorrect predictions is 1.

SONUÇ: *Başlangıç durumu olarak "dallanma yok" (00 veya 01) tercih edilir. Hata sayısı 1 olur.*