

In imaginea de mai sus se pot observa modificarile aduse in program. Pentru NO OP am ales sa folosesc operatia add &0,&0,&0.

Programul initial este:

0: add $1,$0,$0 #B"000\_000\_000\_001\_0\_000"

1: addi $4,$0,5 #B"001\_000\_100\_0000101"

2: add $2,$0,$0 #B"000\_000\_000\_010\_0\_000"

3: add $5,$0,$0 #B"000\_000\_000\_101\_0\_000"

4: addi &6,&0,2 #B"001\_000\_110\_0000010"

5: beq $1,$4,12 #B"100\_001\_100\_0001100"

6: lw: $3,20($2) #B"010\_010\_011\_0010100"

7: sw $3,20($2) #B"011\_010\_011\_0010100"

8: bne $3,$6,5 #B"101\_011\_110\_0000101"

9: add $5,$5,$3 #B"000\_101\_011\_101\_0\_000"

10: add $5,$5,$3 #B"000\_101\_011\_101\_0\_000"

11: addi $2,$2,2 #B"001\_010\_010\_0000010"

12: addi $1,$1,1 #B"001\_001\_001\_0000001"

13: j 5 #B"111\_0000000000101"

14: add $5,$5,$3 #B"000\_101\_011\_101\_0\_000"

15: addi $2,$2,2 #B"001\_010\_010\_0000010"

16: addi $1,$1,1 #B"001\_001\_001\_0000001"

17: j 5 #B"111\_0000000000101"

18: sw $5,10($0) #B"011\_000\_101\_0001010"

Hazardurile de control sunt intalnite la cele doua instructiuni j5 (pentru a le solutiona am adaugat o operatie NO OP dupa fiecare jump 5 dupa cum se vede si in imagine) si la instructiunile de branch beq si bne (pentru a le solutiona am adaugat 3 operatii NO OP dupa acestea, dupa cum se vede si in imagine).

Hazardul de date dintre add $5,$5,$3 si add $5,$5,$3 a fost rezolvat cu adaugarea a doua operatii NO OP dupa prima instructiune, iar astfel am intarziat cu 2 cicluri de ceas programul. Problema a fost datorata incercarii citirii dintr-un registru (&5) care nu a fost inca scris in memorie (read after write).

Hazardul structural care urmeaza dupa solutionarea hazardului de date a fost reparat cu ajutorul codului VHDL, alegand sa fac citirea asincrona in blocul de memorie.

Datorita acestor modificari, am schimbat si imediatul din instructiunile de branch pentru a se sari la instructiunea corespunzatoare.

Codul final este prezentat mai jos, in partea stanga fiind codul masina, iar in partea dreapta instructiunea in hexa si in limbaj de asamblare.

0: B"000\_000\_000\_010\_0\_000", --X0020 add $2,$0,$0

1: B"000\_000\_000\_101\_0\_000", --X0050 add $5,$0,$0

2: B"001\_000\_110\_0000010", --X2302 addi &6,&0,2

3: B"000\_000\_000\_001\_0\_000", --X0010 add $1,$0,$0

4: B"001\_000\_100\_0000101", --X2205 addi $4,$0,5

5: B"100\_001\_100\_0010110", --beq $1,$4,22

6: B"000\_000\_000\_000\_0\_000", --NO OP-- add &0,&0,&0

7: B"000\_000\_000\_000\_0\_000", --NO OP-- add &0,&0,&0

8: B"000\_000\_000\_000\_0\_000", --NO OP-- add &0,&0,&0

9: B"010\_010\_011\_0000000", --X4980 lw: $3,0($2)

10: B"011\_010\_011\_0000000", --X6980 sw $3,0($2)

11: B"101\_011\_110\_0001010", --bne $3,$6,10

12: B"000\_000\_000\_000\_0\_000", --NO OP-- add &0,&0,&0

13: B"000\_000\_000\_000\_0\_000", --NO OP-- add &0,&0,&0

14: B"000\_000\_000\_000\_0\_000", --NO OP-- add &0,&0,&0

15: B"000\_101\_011\_101\_0\_000", --X15D0 add $5,$5,$3

16: B"000\_000\_000\_000\_0\_000", --NO OP-- add &0,&0,&0

17: B"000\_000\_000\_000\_0\_000", --NO OP-- add &0,&0,&0

18: B"000\_101\_011\_101\_0\_000", --X15D0 add $5,$5,$3

19: B"001\_010\_010\_0000001", --X2901 addi $2,$2,1

20: B"001\_001\_001\_0000001", --X2481 addi $1,$1,1

21: B"111\_0000000000101", --XE005 j 5

22: B"000\_000\_000\_000\_0\_000", --NO OP-- add &0,&0,&0

23: B"000\_101\_011\_101\_0\_000", --X15D0 add $5,$5,$3

24: B"001\_010\_010\_0000001", --X2901 addi $2,$2,1

25: B"001\_001\_001\_0000001", --X2481 addi $1,$1,1

26: B"111\_0000000000101", --XE005 j 5

27: B"000\_000\_000\_000\_0\_000", --NO OP-- add &0,&0,&0

28: B"011\_000\_101\_0000000", --X6280 sw $5,0($0)