REPORT

In this assignment the following instructions were added to the processer made in assignment 4: j, beq, bne, blez, bgtz, jal and jr.

Clock Cycles Required:

Each of unconditional branch instructions took 2 cycles each and the conditional branch instructions took 1 cycle if the branch was not taking place and 2 cycles if the branch was taking place.

Successful Branch:

Whenever a branch took place in any of the 7 branch instructions 2 clock cycles were taken for execution in the first clock cycle the program counter was updated to the new instruction that needs to be executed and in the next clock cycle that instruction was fetched for execution in the subsequent clock cycle(s).

Unsuccessful Branch:

Whenever a conditional branch instuction is to be executed but the branch condition is false then the execution of that instruction took only 1 clock cycle in which the next instruction to be executed was fetched.

Testing:

		lable(if any)	CASE CHECKED
1.	add 2 0 1		
2.	sub 3 0 1		
3.	sll 4 0 1		
4.	srl 5 0 1		
5.	sw 5 6 1000		
6.	lw 6 6 1000		--Load stored value checked
7.	add 6 6 6		--Same register read write checked
8.	add 7 6 5		
9.	add 7 7 5		
10.	sub 8 8 7		
11.	sub 9 9 8		
12.	sll 9 9 1		
13.	srl 10 9 1		
14.	sw 7 31 1001		

15. lw 11 31 1001
16. lw 12 31 1000
17. add 11 12 11
18. add 28 28 28 --Assignment 3,4 Checked
19. j 21 =Here
20. 000000
21. 000000
22. add 12 12 12 :Here --j Checked
23. beq 12 11 25 =done :loop --while $12!=$11 beq both
24. add 12 12 5
25. j 22 =loop
26. bne 12 11 22 =loop :done --bne fail
27. bne 12 10 29 =Here4 :Here1 --bne success
28. j 35 =Here6 :Here2
29. j 31 =Here5 :Here3
30. blez 9 28 =Here3 :Here4 --blez fail
31. blez 8 28 =Here3 --blez success
32. bgtz 8 27 =Here2 :Here5 --bgtz fail
33. bgtz 9 27 =Here2 --bgtz success
34. 00000
35. 00000
36. add 12 12 12 :Here6
37. srl 1 1 1
38. sll 2 1 1 --$0->8 $1->1 $2->2
39. jal 40 =add_all -- Stores Arguement in $0 -- returns Sum(1,2,..n) in $30 -- $28 as $zero -- $29 is sp
40. 00000
41. sub 29 29 2 :add_all --Create Stack
42. sw 0 29 0 --Push Arguement
43. sw 31 29 1 --Push $ra
44. beq 0 28 45 =zero
45. j 48 =nonzero

46. add 30 28 28 :zero
47. add 29 29 2 --Pop from stack
48. jr 31
49. sub 0 0 1 :nonzero
50. jal 40 =add_all
51. lw 0 29 0 --Restore arguement
52. lw 31 29 1 --Restore $ra
53. add 30 30 0
54. add 29 29 2 --Pop from stack
55. jr 31

Generated Binary Code:

1. 00000000000000010001000000100000,
2. 00000000000000010001100000100010,
3. 00000000000000000010000001000000,
4. 00000000000000000010100001000010,
5. 10101100110001010000001111101000,
6. 10001100110001100000001111101000,
7. 00000000110001100011000000100000,
8. 00000000110001010011100000100000,
9. 00000000111001010011100000100000,
10. 00000001000001110100000000100010,
11. 00000001001010000100100000100010,
12. 00000000000010010100100001000000,
13. 00000000000010010101000001000010,
14. 10101111111001110000001111101001,
15. 10001111111010110000001111101001,
16. 10001111111011000000001111101000,
17. 00000001100010110101100000100000,
18. 00000011100111001110000000100000,
19. 00001000000000000000000000010101,
20. 00000000000000000000000000000000,

21. 00000000000000000000000000000000,
22. 00000001100011000110000000100000,
23. 00010001100010110000000000011001,
24. 00000001100001010110000000100000,
25. 00001000000000000000000000010110,
26. 00010101100010110000000000010110,
27. 00010101100010100000000000011101,
28. 00001000000000000000000000100011,
29. 00001000000000000000000000011111,
30. 00011001001000000000000000011100,
31. 00011001000000000000000000011100,
32. 00011101000000000000000000011011,
33. 00011101001000000000000000011011,
34. 00000000000000000000000000000000,
35. 00000000000000000000000000000000,
36. 00000001100011000110000000100000,
37. 00000000000000010000100001000010,
38. 00000000000000010001000001000000,
39. 00001100000000000000000000101000,
40. 00000000000000000000000000000000,
41. 00000011101000101110100000100010,
42. 10101111101000000000000000000000,
43. 10101111101111110000000000000001,
44. 00010000000111000000000000101101,
45. 00001000000000000000000000110000,
46. 00000011100111001111000000100000,
47. 00000011101000101110100000100000,
48. 00000011111000000000000000001000,
49. 00000000000000010000000000100010,
50. 00001100000000000000000000101000,
51. 10001111101000000000000000000000,
52. 10001111101111110000000000000001,

53. 00000011110000001111000000100000,
54. 00000011101000101110100000100000,
55. 00000011111000000000000000001000;

Expected Output:

The non-leaf procedure add_all:

1. took 8 in register 0 as arguement
2. gave 36 (= 8 + 7+ . . . + 1) in the register 30
3. the execution terminated at line 40 where the non-leaf procedure returned after execution