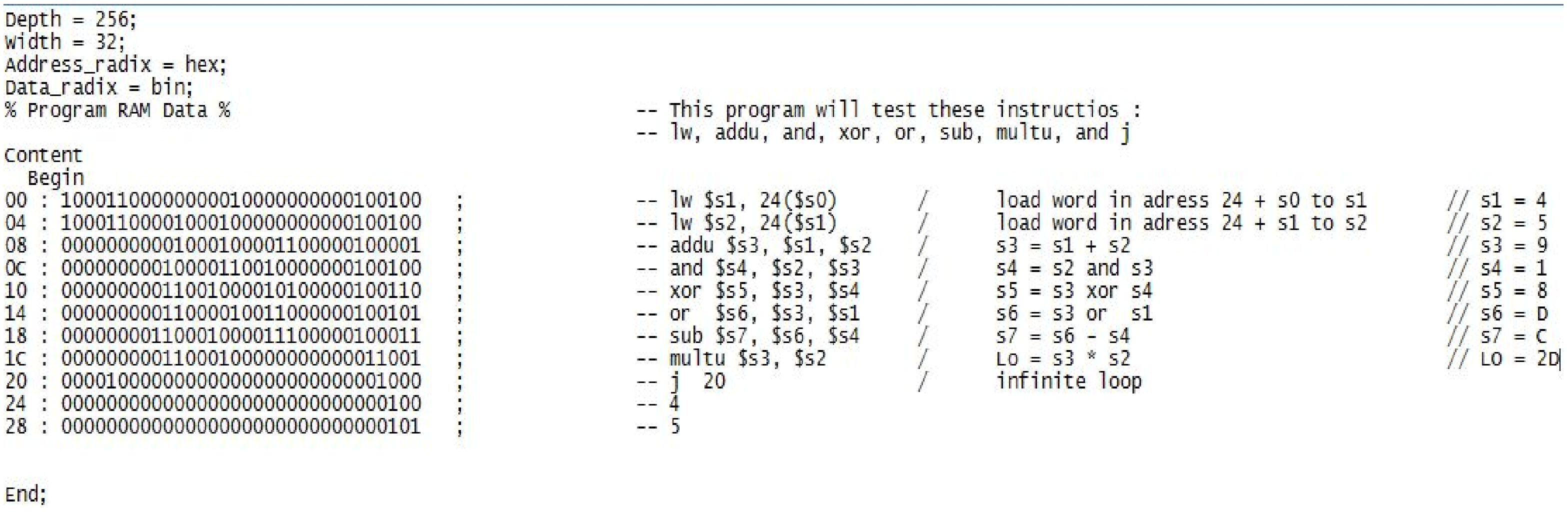
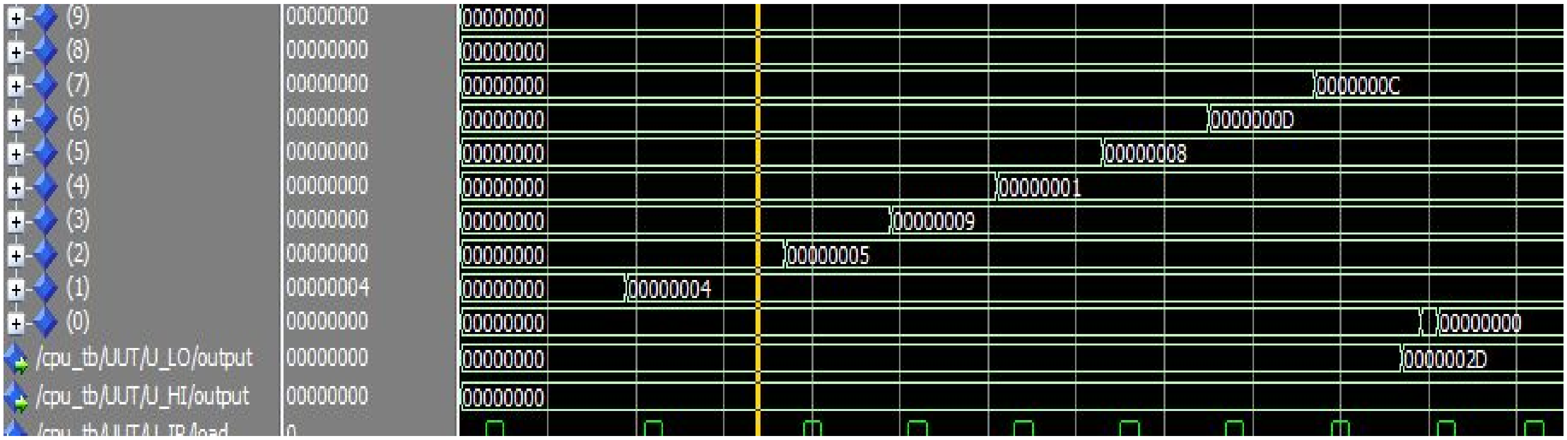
TestCase 1:

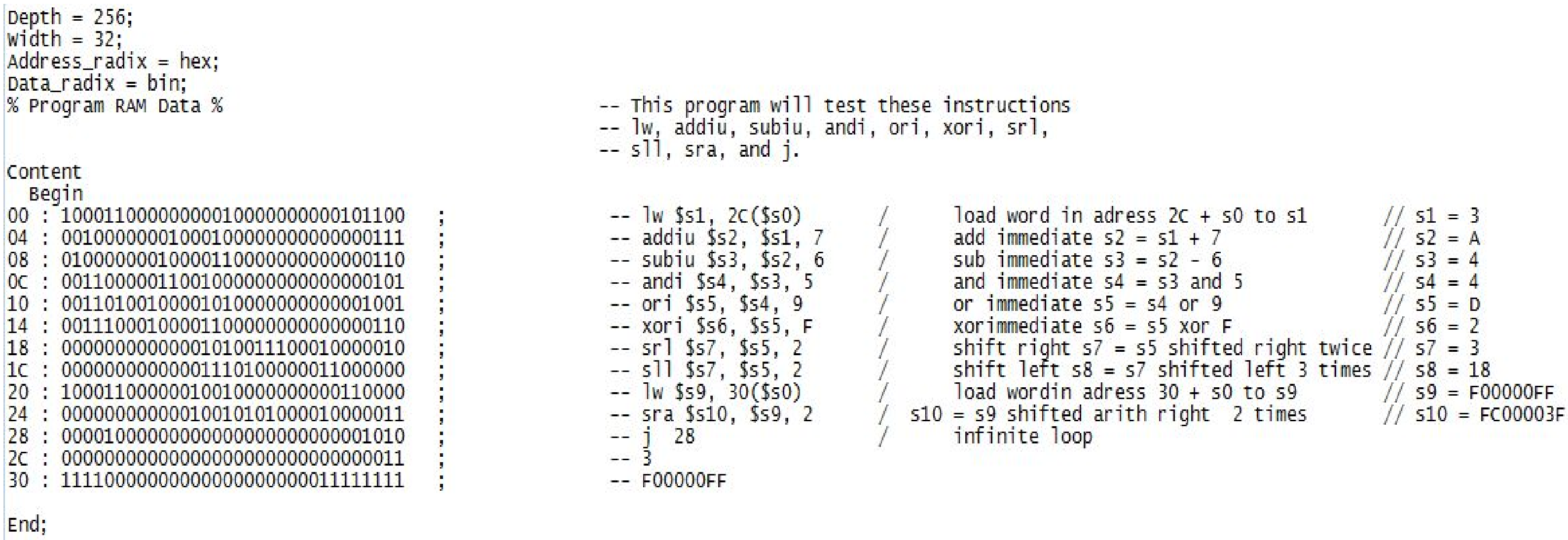


```
Depth = 256;
Width = 32;
Address_radix = hex;
Data_radix = bin;
% Program RAM Data %                                       -- This program will test these instructios :
                                                           -- lw, addu, and, xor, or, sub, multu, and j
Content
  Begin
00 : 10001100000000010000000000100100   ;               -- lw $s1, 24($s0)      /     load word in adress 24 + s0 to s1       // s1 = 4
04 : 10001100001000100000000000100100   ;               -- lw $s2, 24($s1)      /     load word in adress 24 + s1 to s2       // s2 = 5
08 : 00000000001000100001100000100001   ;               -- addu $s3, $s1, $s2   /     s3 = s1 + s2                            // s3 = 9
0C : 00000000010000110010000000100100   ;               -- and $s4, $s2, $s3    /     s4 = s2 and s3                          // s4 = 1
10 : 00000000011001000010100000100110   ;               -- xor $s5, $s3, $s4    /     s5 = s3 xor s4                          // s5 = 8
14 : 00000000011000010011000000100101   ;               -- or  $s6, $s3, $s1    /     s6 = s3 or  s1                          // s6 = D
18 : 00000000110001000011100000100011   ;               -- sub $s7, $s6, $s4    /     s7 = s6 - s4                            // s7 = C
1C : 00000000011000100000000000011001   ;               -- multu $s3, $s2       /     Lo = s3 * s2                            // LO = 2D
20 : 00001000000000000000000000001000   ;               -- j  20                /     infinite loop
24 : 00000000000000000000000000000100   ;               -- 4
28 : 00000000000000000000000000000101   ;               -- 5


End;
```

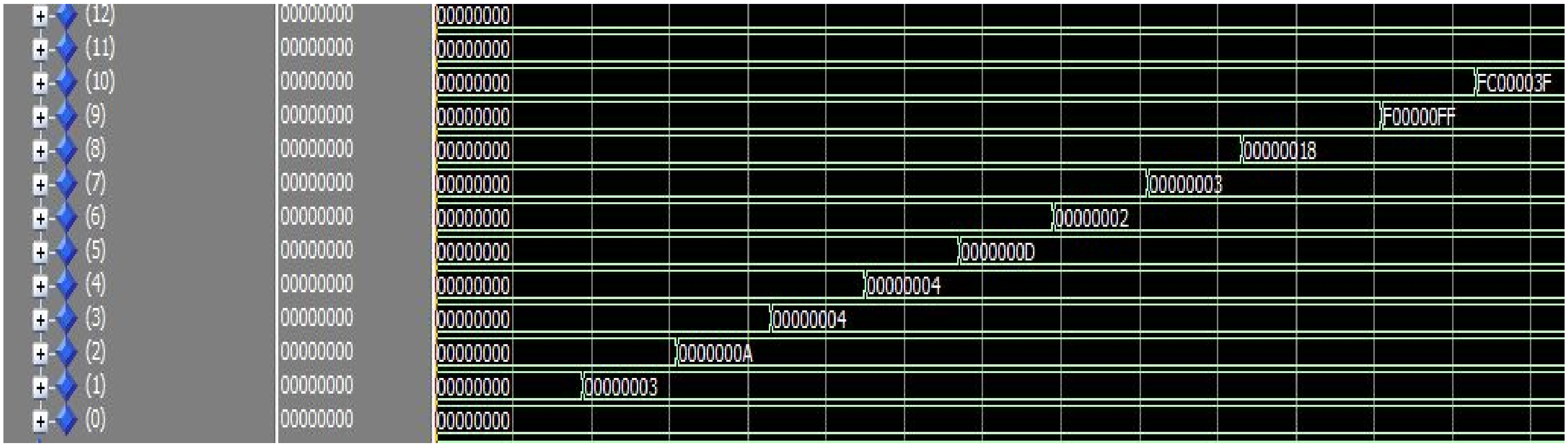


```
+ (9)                    00000000   00000000
+ (8)                    00000000   00000000
+ (7)                    00000000   00000000                                  0000000C
+ (6)                    00000000   00000000                             0000000D
+ (5)                    00000000   00000000                        00000008
+ (4)                    00000000   00000000                   00000001
+ (3)                    00000000   00000000              00000009
+ (2)                    00000000   00000000         00000005
+ (1)                    00000004   00000000    00000004
+ (0)                    00000000   00000000                                          00000000
/cpu_tb/UUT/U_LO/output  00000000   00000000                                     0000002D
/cpu_tb/UUT/U_HI/output  00000000   00000000
/cpu_tb/UUT/U_IR/load    0
```

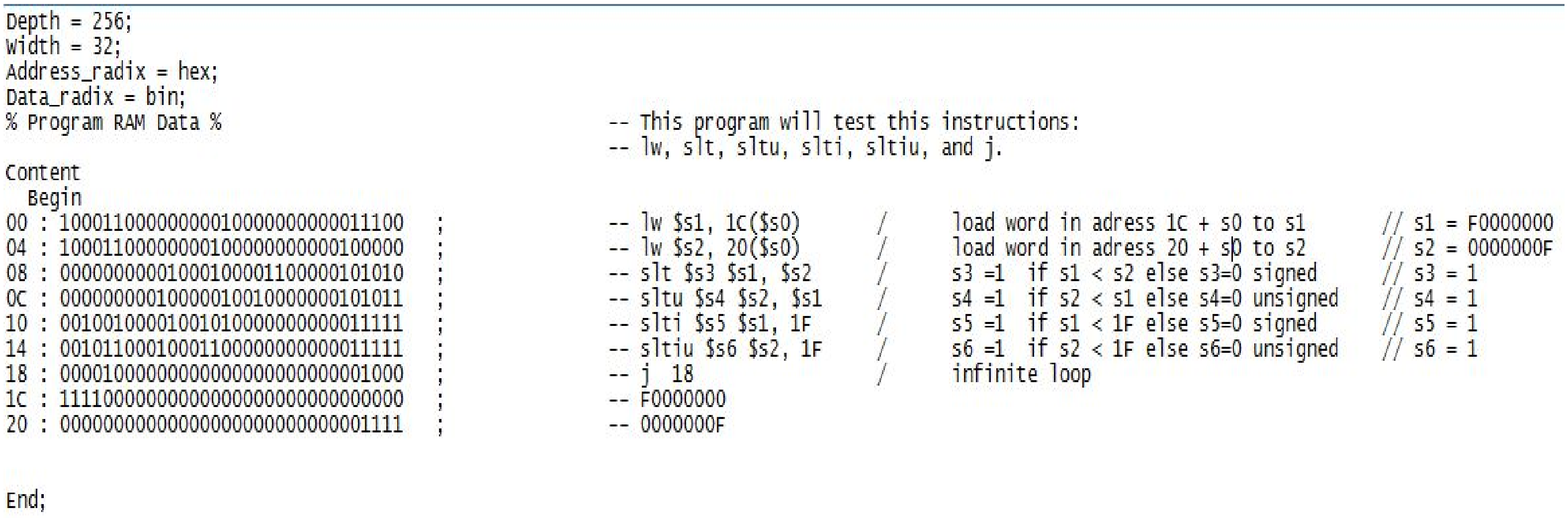
TestCase 2:



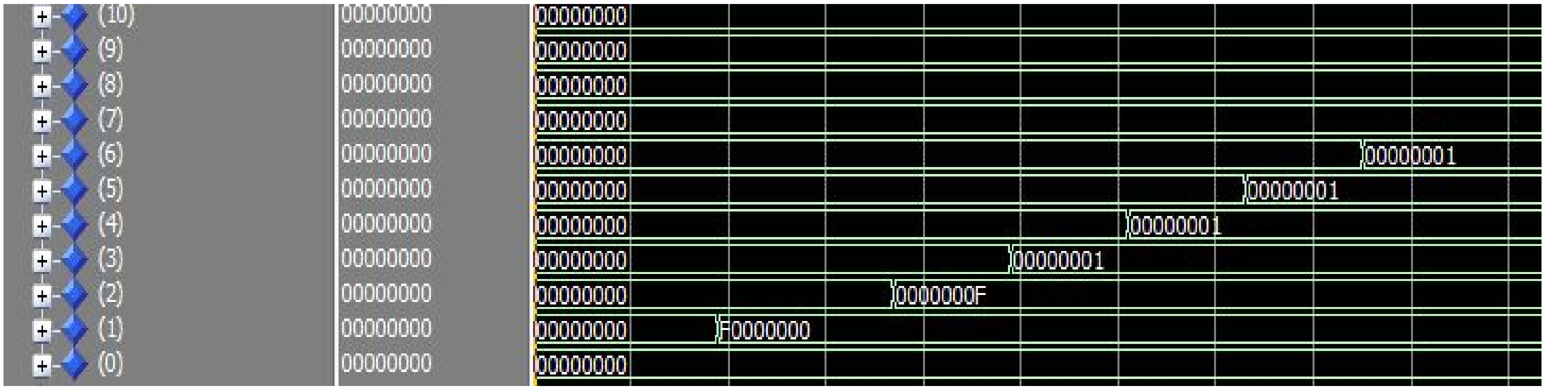
```
Depth = 256;
Width = 32;
Address_radix = hex;
Data_radix = bin;
% Program RAM Data %                                    -- This program will test these instructions
                                                        -- lw, addiu, subiu, andi, ori, xori, srl,
                                                        -- sll, sra, and j.

Content
  Begin
00 : 10001100000000001000000000101100   ;              -- lw $s1, 2C($s0)       /      load word in adress 2C + s0 to s1       // s1 = 3
04 : 00100000001000100000000000000111   ;              -- addiu $s2, $s1, 7     /      add immediate s2 = s1 + 7               // s2 = A
08 : 01000000010000110000000000000110   ;              -- subiu $s3, $s2, 6     /      sub immediate s3 = s2 - 6               // s3 = 4
0C : 00110000011001000000000000000101   ;              -- andi $s4, $s3, 5      /      and immediate s4 = s3 and 5             // s4 = 4
10 : 00110100100001010000000000001001   ;              -- ori $s5, $s4, 9       /      or immediate s5 = s4 or 9               // s5 = D
14 : 00111000101001100000000000000110   ;              -- xori $s6, $s5, F      /      xorimmediate s6 = s5 xor F              // s6 = 2
18 : 00000000000001010011100010000010   ;              -- srl $s7, $s5, 2       /      shift right s7 = s5 shifted right twice // s7 = 3
1C : 00000000000001110100000011000000   ;              -- sll $s7, $s5, 2       /      shift left s8 = s7 shifted left 3 times // s8 = 18
20 : 10001100000010010000000000110000   ;              -- lw $s9, 30($s0)       /      load wordin adress 30 + s0 to s9        // s9 = F00000FF
24 : 00000000000010010101000010000011   ;              -- sra $s10, $s9, 2      /  s10 = s9 shifted arith right  2 times       // s10 = FC00003F
28 : 00001000000000000000000000001010   ;              -- j  28                 /      infinite loop
2C : 00000000000000000000000000000011   ;              -- 3
30 : 11110000000000000000000011111111   ;              -- F00000FF

End;
```



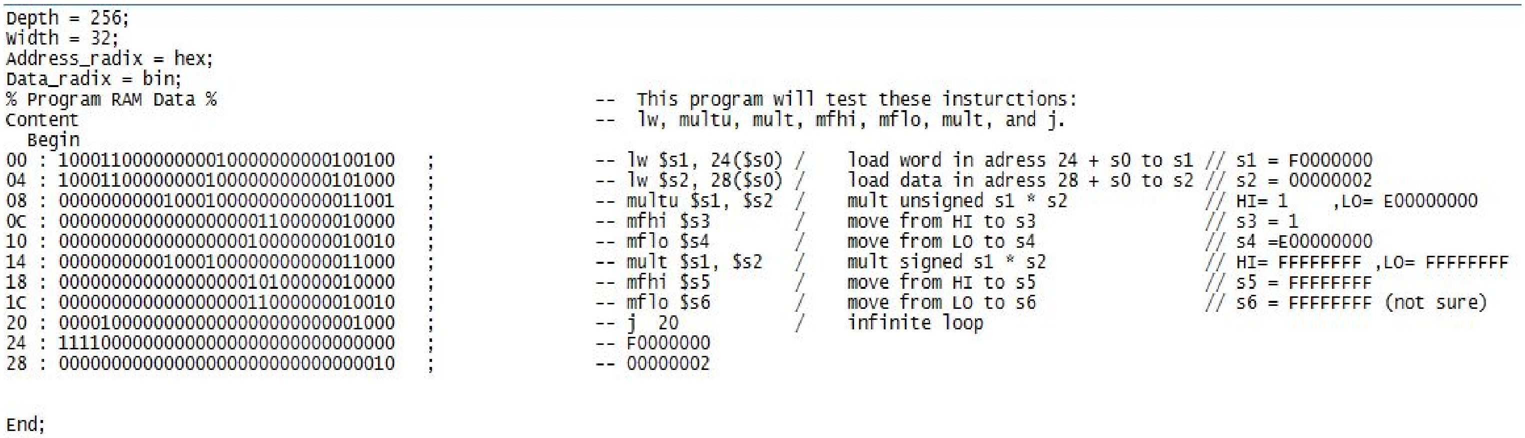
TestCase 3:



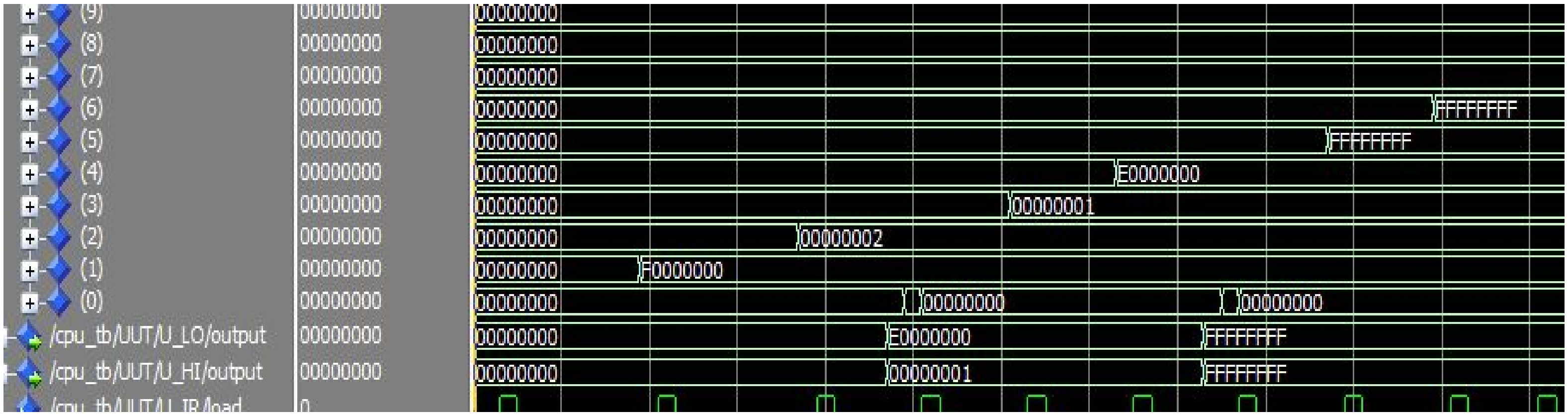
```
Depth = 256;
Width = 32;
Address_radix = hex;
Data_radix = bin;
% Program RAM Data %                                    -- This program will test this instructions:
                                                        -- lw, slt, sltu, slti, sltiu, and j.
Content
  Begin
00 : 10001100000000010000000000011100   ;               -- lw $s1, 1C($s0)      /     load word in adress 1C + s0 to s1        // s1 = F0000000
04 : 10001100000000100000000000100000   ;               -- lw $s2, 20($s0)      /     load word in adress 20 + s0 to s2        // s2 = 0000000F
08 : 00000000001000100001100000101010   ;               -- slt $s3 $s1, $s2     /     s3 =1  if s1 < s2 else s3=0 signed       // s3 = 1
0C : 00000000010000010010000000101011   ;               -- sltu $s4 $s2, $s1    /     s4 =1  if s2 < s1 else s4=0 unsigned     // s4 = 1
10 : 00100100001001010000000000011111   ;               -- slti $s5 $s1, 1F     /     s5 =1  if s1 < 1F else s5=0 signed       // s5 = 1
14 : 00101100010001100000000000011111   ;               -- sltiu $s6 $s2, 1F    /     s6 =1  if s2 < 1F else s6=0 unsigned     // s6 = 1
18 : 00001000000000000000000000001000   ;               -- j  18                /     infinite loop
1C : 11110000000000000000000000000000   ;               -- F0000000
20 : 00000000000000000000000000001111   ;               -- 0000000F


End;
```



TestCase 4:

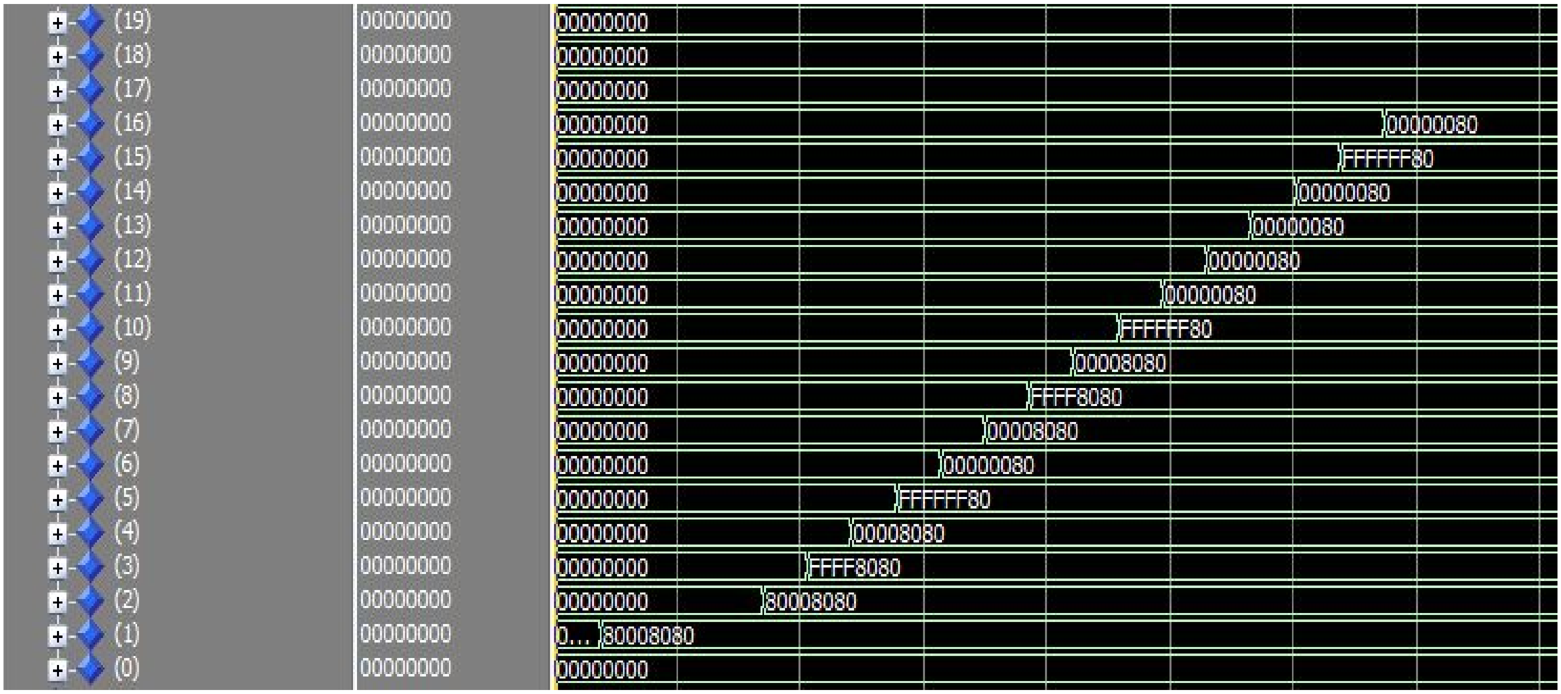


```
Depth = 256;
Width = 32;
Address_radix = hex;
Data_radix = bin;
% Program RAM Data %                                      --  This program will test these insturctions:
Content                                                   --  lw, multu, mult, mfhi, mflo, mult, and j.
  Begin
00 : 10001100000000010000000000100100   ;                 -- lw $s1, 24($s0) /    load word in adress 24 + s0 to s1 // s1 = F0000000
04 : 10001100000000100000000000101000   ;                 -- lw $s2, 28($s0) /    load data in adress 28 + s0 to s2 // s2 = 00000002
08 : 00000000001000100000000000011001   ;                 -- multu $s1, $s2  /    mult unsigned s1 * s2             // HI= 1    ,LO= E00000000
0C : 00000000000000000001100000010000   ;                 -- mfhi $s3        /    move from HI to s3                // s3 = 1
10 : 00000000000000000010000000010010   ;                 -- mflo $s4        /    move from LO to s4                // s4 =E00000000
14 : 00000000001000100000000000011000   ;                 -- mult $s1, $s2   /    mult signed s1 * s2               // HI= FFFFFFFF ,LO= FFFFFFFF
18 : 00000000000000000010100000010000   ;                 -- mfhi $s5        /    move from HI to s5                // s5 = FFFFFFFF
1C : 00000000000000000011000000010010   ;                 -- mflo $s6        /    move from LO to s6                // s6 = FFFFFFFF (not sure)
20 : 00001000000000000000000000001000   ;                 -- j  20           /    infinite loop
24 : 11110000000000000000000000000000   ;                 -- F0000000
28 : 00000000000000000000000000000010   ;                 -- 00000002


End;
```


(9) 00000000 00000000
(8) 00000000 00000000
(7) 00000000 00000000
(6) 00000000 00000000 FFFFFFFF
(5) 00000000 00000000 FFFFFFFF
(4) 00000000 00000000 E0000000
(3) 00000000 00000000 00000001
(2) 00000000 00000000 00000002
(1) 00000000 00000000 F0000000
(0) 00000000 00000000 00000000 00000000
/cpu_tb/UUT/U_LO/output 00000000 00000000 E0000000 FFFFFFFF
/cpu_tb/UUT/U_HI/output 00000000 00000000 00000001 FFFFFFFF
/cpu_tb/UUT/U_IR/load 0

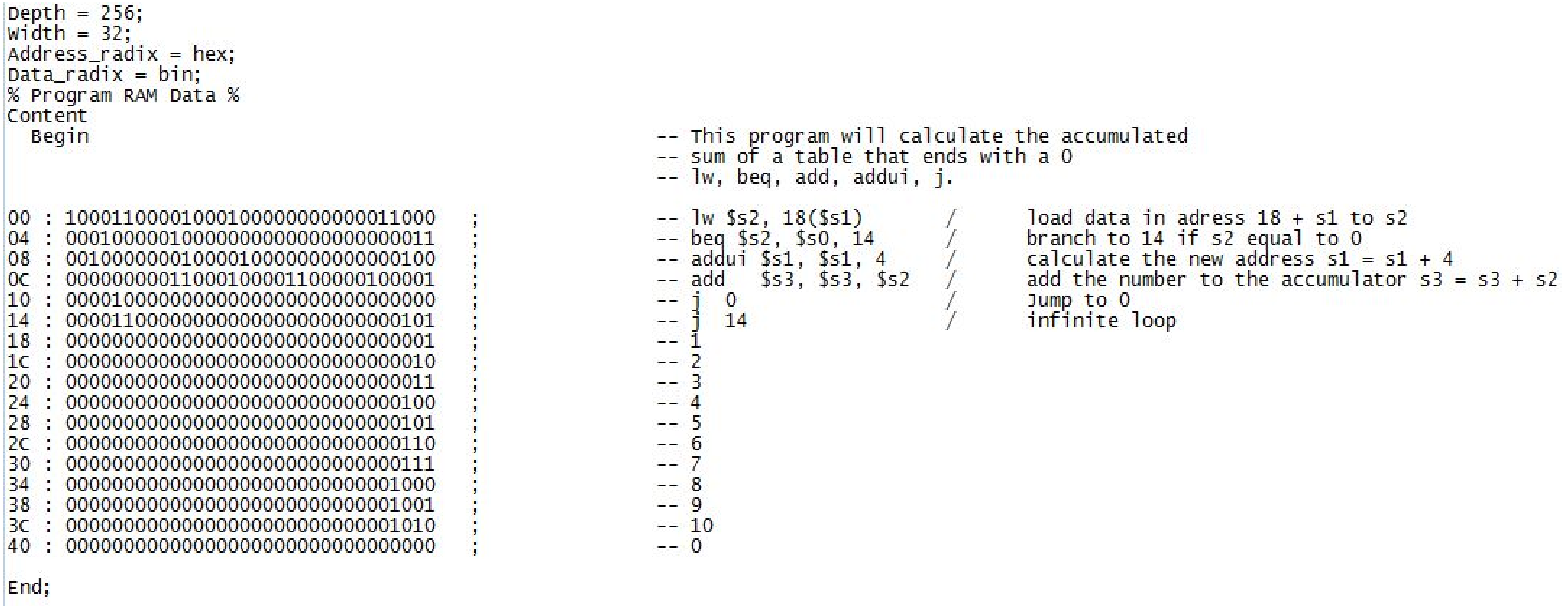
TestCase 5:

```
Depth = 256;
Width = 32;
Address_radix = hex;
Data_radix = bin;
% Program RAM Data %                                 -- This program will test these instrunctions:
                                                     -- lw, lh, lhu, lb, lbu, sw, sh, and sb.
Content
  Begin
00 : 10001100000000010000000001100100   ;        -- lw $s1, 64($s0)       /      load word in adress 64 + s0 to s1                // s1 = 80008080
04 : 10101100000000010000000001101000   ;        -- sw $s1, 68($s0)       /      store s1 in the adress s0 + 68
08 : 10100100000000010000000001101100   ;        -- sh $s1, 6C($s0)       /      store half s1 in the adress s0 + 6C
0C : 10100000000000010000000001110000   ;        -- sb $s1, 70($s0)       /      store byte of s1 in the adress s0 + 70

10 : 10001100000000100000000001101000   ;        -- lw  $s2, 68($s0)      /      load word in adress 68 + s0 to s2                // s2 = 80008080
14 : 10000100000000110000000001101000   ;        -- lh  $s3, 68($s0)      /      load half word in adress 68 + s0 to s3           // s3 = FFFF8080
18 : 10010100000001000000000001101000   ;        -- lhu $s4, 68($s0)      /      load half word unsigned in adress 68 + s0 to s4 // s4 = 00008080
1C : 10000000000001010000000001101000   ;        -- lb  $s5, 68($s0)      /      load byte word in adress 68 + s0 to s5           // s5 = FFFFFF80
20 : 10010000000001100000000001101000   ;        -- lbu $s6, 68($s0)      /      load byte unsigned unsigned adress 68 +s0 to s6 // s6 = 00000080

24 : 10001100000001110000000001101100   ;        -- lw  $s7, 6C($s0)      /      load word in adress 6C + s0 to s7                // s7 = 00008080
28 : 10000100000010000000000001101100   ;        -- lh  $s8, 6C($s0)      /      load half word in adress 6C + s0 to s8           // s8 = FFFF8080
2C : 10010100000010010000000001101100   ;        -- lhu $s9, 6C($s0)      /      load half word unsigned in adress 6C + s0 to s9 // s9 = 00008080
30 : 10000000000010100000000001101100   ;        -- lb  $s10, 6C($s0)     /      load byte word in adress 6C + s0 to s10          // s10 = FFFFFF80
34 : 10010000000010110000000001101100   ;        -- lbu $s11 6C($s0)      /      load byte unsigned unsigned adress 6C+s0 to s11 // s11 = 00000080

38 : 10001100000011000000000001110000   ;        -- lw  $s12, 70($s0)     /      load word in adress 70 + s0 to s12               // s12 = 00000080
3C : 10000100000011010000000001110000   ;        -- lh  $s13, 70($s0)     /      load half word in adress 70 + s0 to s13          // s13 = 00000080
40 : 10010100000011100000000001110000   ;        -- lhu $s14, 70($s0)     /      load half word unsigned in adress 70 +s0 to s14 // s14 = 00000080
44 : 10000000000011110000000001110000   ;        -- lb  $s15, 70($s0)     /      load byte word in adress 70 + s0 to s15          // s15 = FFFFFF80
48 : 10010000000100000000000001110000   ;        -- lbu $s16, 70($s0)     /      load byte unsigned unsigned adress 70+s0 to s16 // s16 = 00000080

4C : 00001000000000000000000000010011   ;        -- jmp  4C               /      infinite loop

64 : 10000000000000001000000010000000   ;        -- 80008080
End;
```

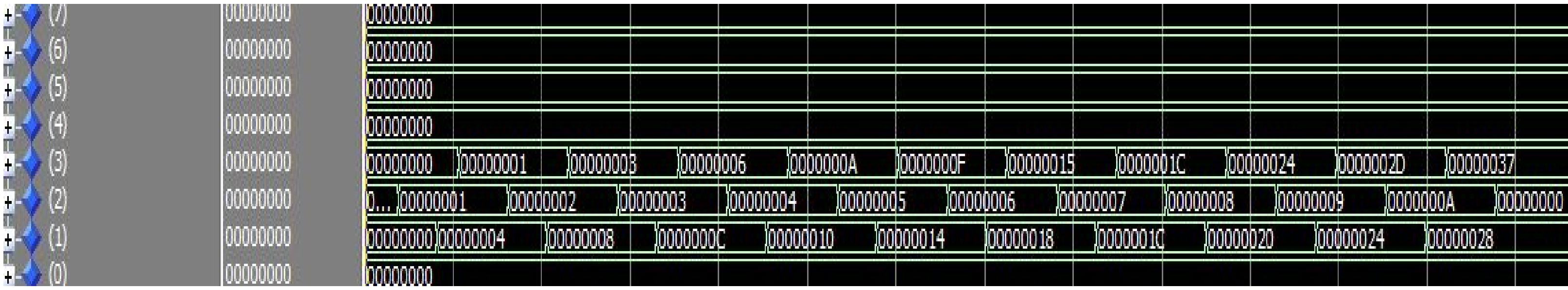


```
(19)  00000000  00000000
(18)  00000000  00000000
(17)  00000000  00000000
(16)  00000000  00000000  00000080
(15)  00000000  00000000  FFFFFF80
(14)  00000000  00000000  00000080
(13)  00000000  00000000  00000080
(12)  00000000  00000000  00000080
(11)  00000000  00000000  00000080
(10)  00000000  00000000  FFFFFF80
(9)   00000000  00000000  00008080
(8)   00000000  00000000  FFFF8080
(7)   00000000  00000000  00008080
(6)   00000000  00000000  00000080
(5)   00000000  00000000  FFFFFF80
(4)   00000000  00000000  00008080
(3)   00000000  00000000  FFFF8080
(2)   00000000  00000000  80008080
(1)   00000000  0...  80008080
(0)   00000000  00000000
```

TestCase 6:

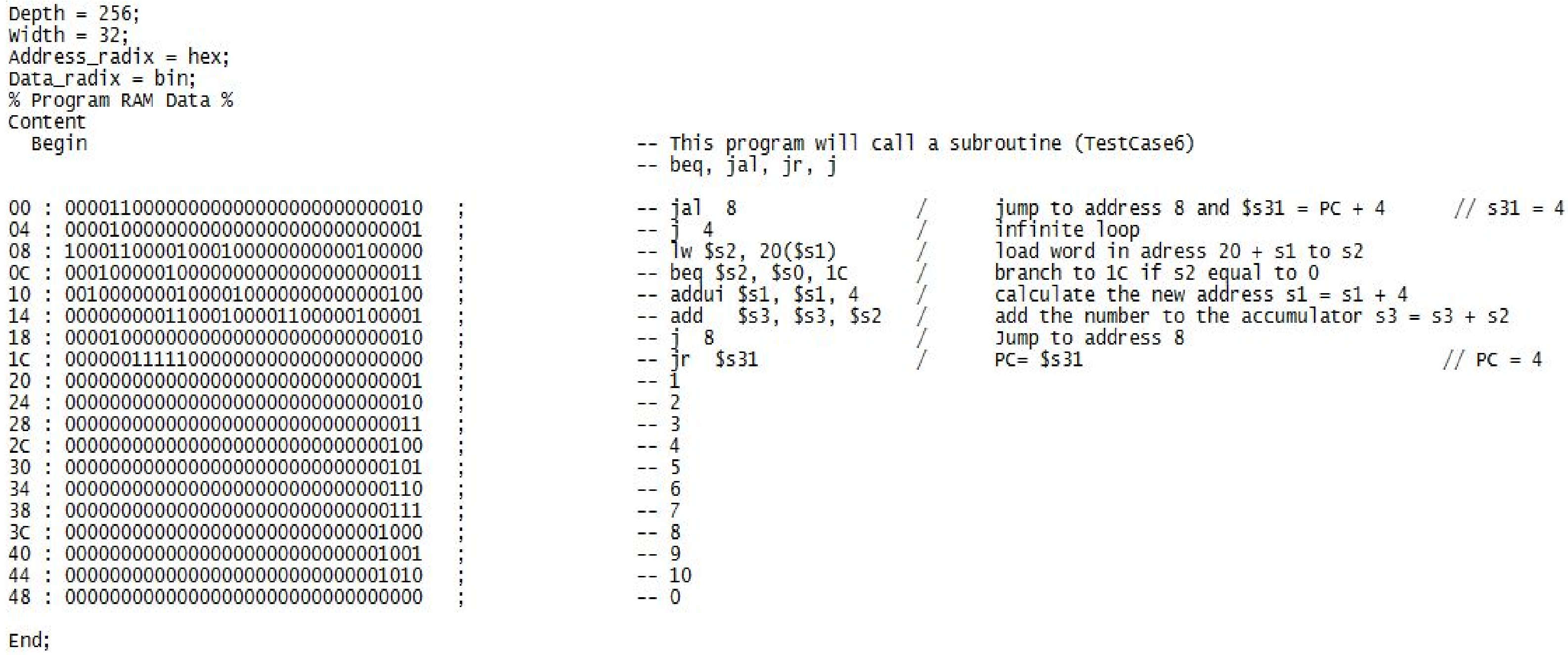


```
Depth = 256;
Width = 32;
Address_radix = hex;
Data_radix = bin;
% Program RAM Data %
Content
  Begin                                                    -- This program will calculate the accumulated
                                                           -- sum of a table that ends with a 0
                                                           -- lw, beq, add, addui, j.

00 : 10001100001000100000000000011000    ;                -- lw $s2, 18($s1)       /      load data in adress 18 + s1 to s2
04 : 00010000010000000000000000000011    ;                -- beq $s2, $s0, 14      /      branch to 14 if s2 equal to 0
08 : 00100000001000010000000000000100    ;                -- addui $s1, $s1, 4     /      calculate the new address s1 = s1 + 4
0C : 00000000011000100001100000100001    ;                -- add   $s3, $s3, $s2   /      add the number to the accumulator s3 = s3 + s2
10 : 00001000000000000000000000000000    ;                -- j  0                  /      Jump to 0
14 : 00001100000000000000000000000101    ;                -- j  14                 /      infinite loop
18 : 00000000000000000000000000000001    ;                -- 1
1C : 00000000000000000000000000000010    ;                -- 2
20 : 00000000000000000000000000000011    ;                -- 3
24 : 00000000000000000000000000000100    ;                -- 4
28 : 00000000000000000000000000000101    ;                -- 5
2C : 00000000000000000000000000000110    ;                -- 6
30 : 00000000000000000000000000000111    ;                -- 7
34 : 00000000000000000000000000001000    ;                -- 8
38 : 00000000000000000000000000001001    ;                -- 9
3C : 00000000000000000000000000001010    ;                -- 10
40 : 00000000000000000000000000000000    ;                -- 0

End;
```

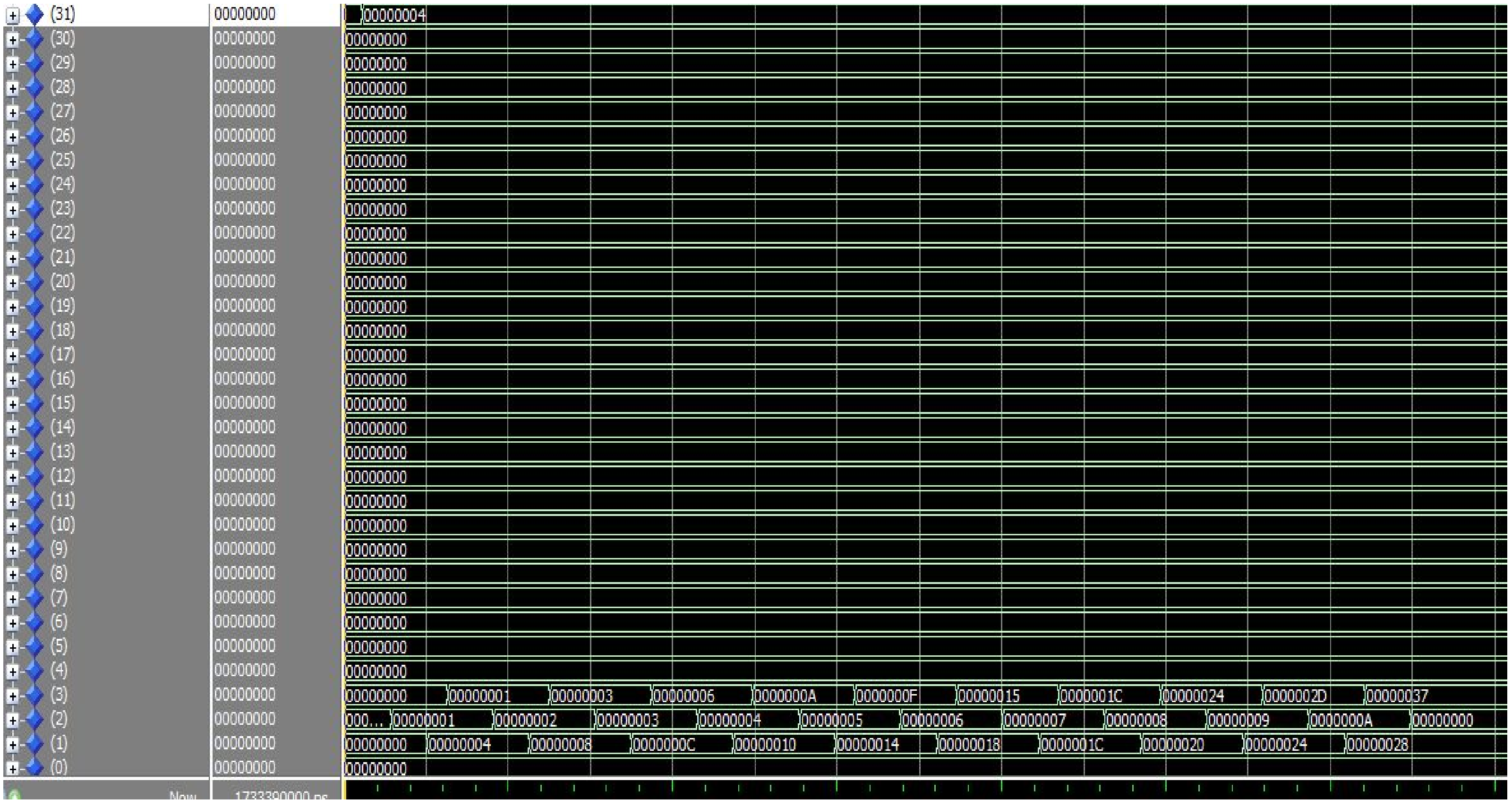


```
(7)  00000000  00000000
(6)  00000000  00000000
(5)  00000000  00000000
(4)  00000000  00000000
(3)  00000000  00000000 00000001 00000003 00000006 0000000A 0000000F 00000015 0000001C 00000024 0000002D 00000037
(2)  00000000  0... 00000001 00000002 00000003 00000004 00000005 00000006 00000007 00000008 00000009 0000000A 00000000
(1)  00000000  00000000 00000004 00000008 0000000C 00000010 00000014 00000018 0000001C 00000020 00000024 00000028
(0)  00000000  00000000
```

TestCase 7:



```
Depth = 256;
Width = 32;
Address_radix = hex;
Data_radix = bin;
% Program RAM Data %
Content
  Begin                                                -- This program will call a subroutine (TestCase6)
                                                       -- beq, jal, jr, j

00 : 00001100000000000000000000000010    ;            -- jal  8                  /      jump to address 8 and $s31 = PC + 4      // s31 = 4
04 : 00001000000000000000000000000001    ;            -- j  4                    /      infinite loop
08 : 10001100001000100000000000100000    ;            -- lw $s2, 20($s1)         /      load word in adress 20 + s1 to s2
0C : 00010000010000000000000000000011    ;            -- beq $s2, $s0, 1C        /      branch to 1C if s2 equal to 0
10 : 00100000001000010000000000000100    ;            -- addui $s1, $s1, 4       /      calculate the new address s1 = s1 + 4
14 : 00000000011000100001100000100001    ;            -- add   $s3, $s3, $s2     /      add the number to the accumulator s3 = s3 + s2
18 : 00001000000000000000000000000010    ;            -- j  8                    /      Jump to address 8
1C : 00000011111000000000000000000000    ;            -- jr  $s31                /      PC= $s31                               // PC = 4
20 : 00000000000000000000000000000001    ;            -- 1
24 : 00000000000000000000000000000010    ;            -- 2
28 : 00000000000000000000000000000011    ;            -- 3
2C : 00000000000000000000000000000100    ;            -- 4
30 : 00000000000000000000000000000101    ;            -- 5
34 : 00000000000000000000000000000110    ;            -- 6
38 : 00000000000000000000000000000111    ;            -- 7
3C : 00000000000000000000000000001000    ;            -- 8
40 : 00000000000000000000000000001001    ;            -- 9
44 : 00000000000000000000000000001010    ;            -- 10
48 : 00000000000000000000000000000000    ;            -- 0

End;
```



```
(31)  00000000   00000004
(30)  00000000   00000000
(29)  00000000   00000000
(28)  00000000   00000000
(27)  00000000   00000000
(26)  00000000   00000000
(25)  00000000   00000000
(24)  00000000   00000000
(23)  00000000   00000000
(22)  00000000   00000000
(21)  00000000   00000000
(20)  00000000   00000000
(19)  00000000   00000000
(18)  00000000   00000000
(17)  00000000   00000000
(16)  00000000   00000000
(15)  00000000   00000000
(14)  00000000   00000000
(13)  00000000   00000000
(12)  00000000   00000000
(11)  00000000   00000000
(10)  00000000   00000000
(9)   00000000   00000000
(8)   00000000   00000000
(7)   00000000   00000000
(6)   00000000   00000000
(5)   00000000   00000000
(4)   00000000   00000000
(3)   00000000   00000000 00000001 00000003 00000006 0000000A 0000000F 00000015 0000001C 00000024 0000002D 00000037
(2)   00000000   000... 00000001 00000002 00000003 00000004 00000005 00000006 00000007 00000008 00000009 0000000A 00000000
(1)   00000000   00000000 00000004 00000008 0000000C 00000010 00000014 00000018 0000001C 00000020 00000024 00000028
(0)   00000000   00000000
Now   1733390000 ps
```