HW 4 REPORT

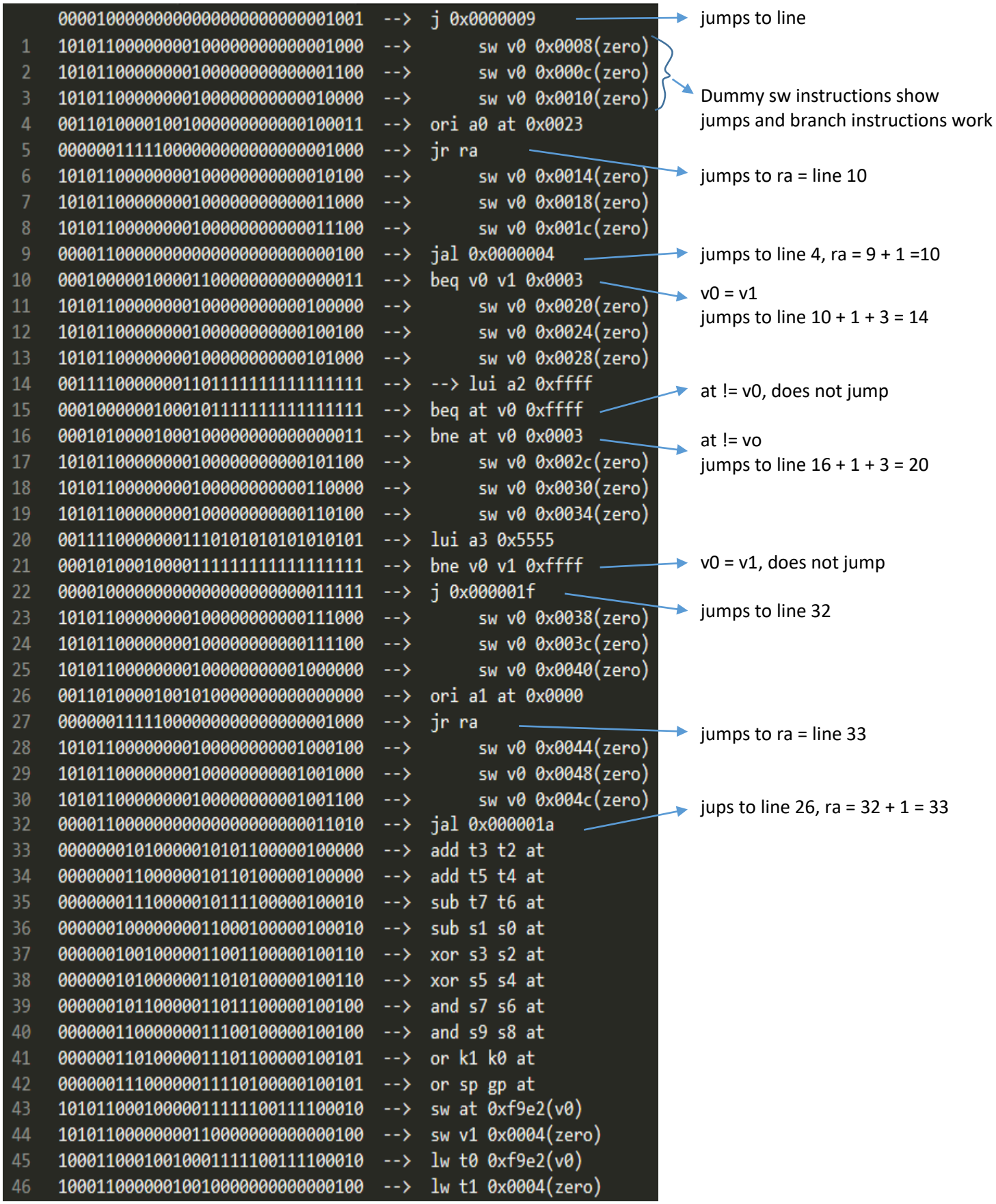
Registers before execution:

```
$ zero   00000000000000000000000000000000
$ at     00000000000000000000000001110001 = 113
$ v0     00000000000000000000011000011110 = 1566
$ v1     00000000000000000000011000011110 = 1566
$ a0     00000000000000000000000000000000
$ a1     00000000000000000000000000000000
$ a2     00000000000000000000000000000000
$ a3     00000000000000000000000000000000
$ t0     00000000000000000000000000000000
$ t1     00000000000000000000000000000000
$ t2     00000000000000000000000011000111 = 199
$ t3     00000000000000000000000000000000
$ t4     11111111111111111111111011101000 = -280
$ t5     00000000000000000000000000000000
$ t6     11111111111111111111111011101000 = -280
$ t7     00000000000000000000000000000000
$ s0     00000000000000000000001110000100 = 900
$ s1     00000000000000000000000000000000
$ s2     00000000000000000000000000000111
$ s3     00000000000000000000000000000000
$ s4     11010100000100101000001111000011
$ s5     00000000000000000000000000000000
$ s6     00000000000000001111111110001110
$ s7     00000000000000000000000000000000
$ t8     00100000100101001000000001110001
$ t9     00000000000000000000000000000000
$ k0     10001001010100111111001010111100
$ k1     00000000000000000000000000000000
$ gp     00000000000000001111111110001110
$ sp     00000000000000000000000000000000
$ fp     00000000000000000000000000000000
$ ra     00000000000000000000000000000000
```

Registers before execution:

They dont have any value on them. They all are X.

Instructions:



Register Output:

```
$ zero   00000000000000000000000000000000
$ at     00000000000000000000000001110001
$ v0     00000000000000000000011000011110
$ v1     00000000000000000000011000011110
$ a0     00000000000000000000000001110011 --> ori a0 at 0x0023
$ a1     00000000000000000000000001110001 --> ori a1 at 0x0000
$ a2     11111111111111110000000000000000 --> lui a2 0xffff
$ a3     01010101010101010000000000000000 --> lui a3 0x5555
$ t0     00000000000000000000000001110001 --> lw t0 0xf9e2(v0)
$ t1     00000000000000000000011000011110 --> lw t1 0x0004(zero)
$ t2     00000000000000000000000100111000 --> addn t3 t2 at --> 199 + 113 = 312
$ t3     00000000000000000000000000000011 --> t2 > 0
$ t4     11111111111111111111111101011001 --> addn t5 t4 at --> -280 + 113 = -167
$ t5     00000000000000000000000000000010 --> t4 < 0
$ t6     11111111111111111111111001110111 --> subn t7 t6 at --> -280 - 113 = -413
$ t7     00000000000000000000000000000010 --> t6 < 0
$ s0     00000000000000000000001100010011 --> subn s1 s0 at --> 900 - 113 = 767
$ s1     00000000000000000000000000000011 --> s0 > 0
$ s2     00000000000000000000000001110110 --> xorn s3 s2 at
$ s3     00000000000000000000000000000011
$ s4     11010100000100101010000110010010 --> xorn s5 s4 at
$ s5     00000000000000000000000000000010
$ s6     00000000000000000000000000000000 --> andn s7 s6 at
$ s7     00000000000000000000000000000001
$ t8     00000000000000000000000001110001 --> andn s9 s8 at
$ t9     00000000000000000000000000000011
$ k0     10001001010100111111001011111101 --> orn k1 k0 at
$ k1     00000000000000000000000000000010
$ gp     00000000000000001111111111111111 --> orn sp gp at
$ sp     00000000000000000000000000000011
$ fp     00000000000000000000000000000000
$ ra     00000000000000000000000010000000
```

Memory output:

```
00000000000000000000000001110001 --> sw at 0xf9e2(v0)
00000000000000000000011000011110 --> sw v1 0x0004(zero)
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

0xf9 = -1566, v0 = 1566, adress = 0

adress = 4 + 0 = 4th byte = 1st word

The rest is all x since I didn't executed any sw instructions. The sw instructions I wrote as dummy instructions didn't work which shows that jump and branch instructions work too.