COMPUTERARCHITECTURE

PROGRAM 4

BY MOHIT KRISHNAMURTHY

Running the code:

python3 mars2.py

It asks the following in the terminal :

>>Enter the File Name(eg. - binary1.txt)

Enter the file name

example:

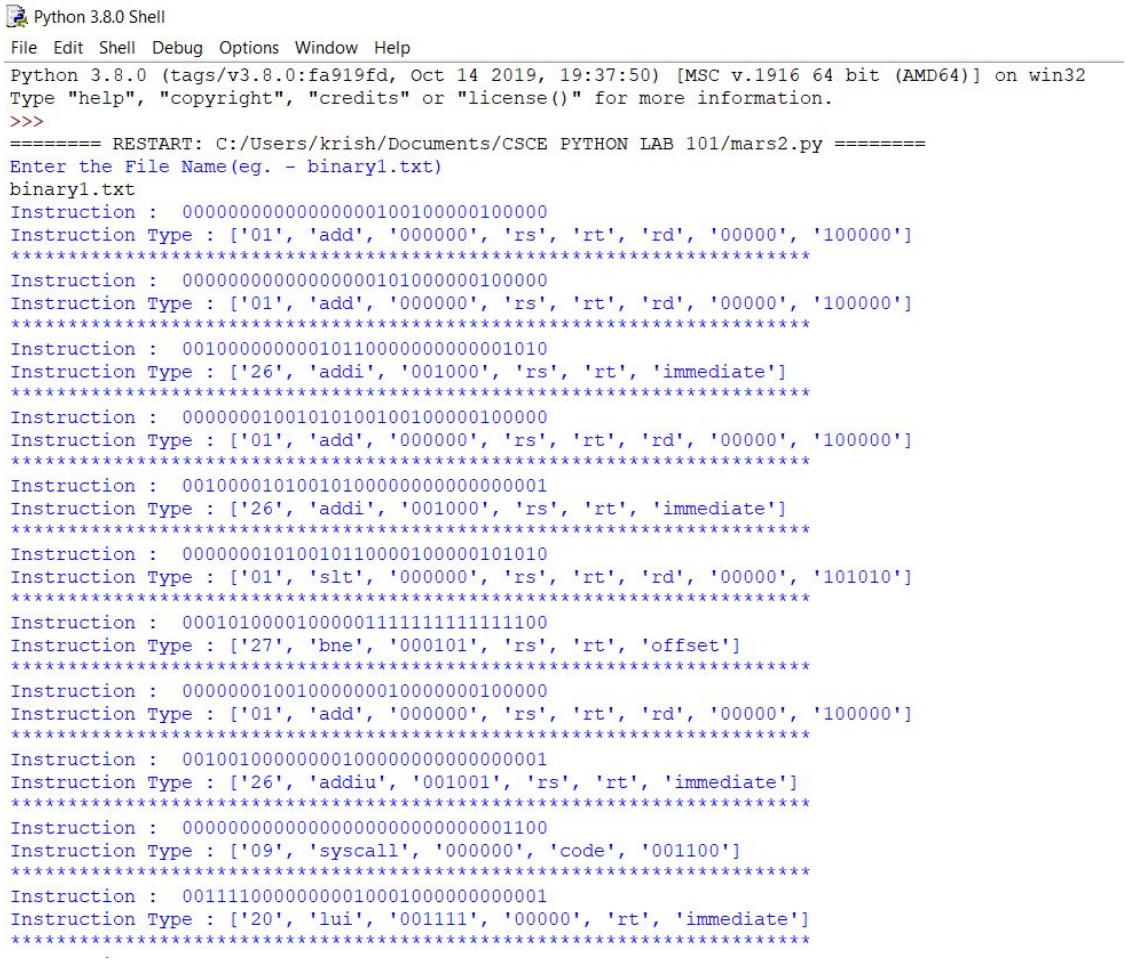
>>binary1.txt

Note: Python 3.8 was used.

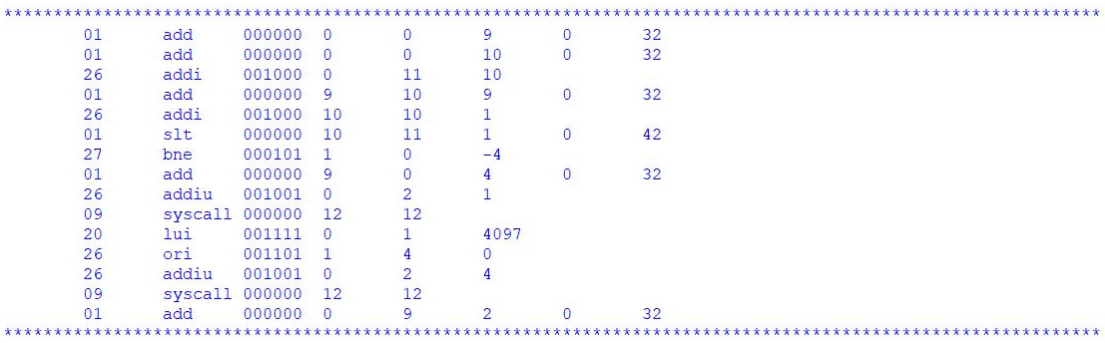
The working of the program is described in the mars2.py file itself, through the use of comments.

Explanation of Output:

>>binary1.txt



The above is the identification of the binary instruction. “Instruction“ above is referring to our binary input and “Instruction Type is its classification”



The above image is the decoded instructions we get from the file.

```
Instruction Fetch
PC : 0
NPC : 4
Instruction Register : ['01', 'add', '000000', 'rs', 'rt', 'rd', '00000', '100000']


Instruction Decode :


Execute Stage :


Memory Stage :


Write Back Stage :
********************************************************************************************************************************
In Clock  1
Instruction Fetch
PC : 4
NPC : 8
Instruction Register : ['01', 'add', '000000', 'rs', 'rt', 'rd', '00000', '100000']


Instruction Decode :
Instruction : ['01', 'add', '000000', 'rs', 'rt', 'rd', '00000', '100000']
Temporary Register A : 0
Temporary Register B : 0
Immediate : None
Decoded Instruction : ['01', 'add', '000000', 0, 0, 9, 0, 32]


Execute Stage :


Memory Stage :


Write Back Stage :
********************************************************************************************************************************
```

The above image is the 1st and 2nd clock cycle . In 1st clock cycle i.e. clock only instruction fetch is performed, but in 2nd clock cycle both instruction fetch and instruction decode is performed .

In instruction fetch the PC,NPC and the instruction being fetched is displayed.

In instruction decode the temporary A and B register values, immediate values and decoded instruction is displayed.

```
********************************************************************************
******************************
In Clock  4
Instruction Fetch
PC : 16
NPC : 20
Instruction Register : ['26', 'addi', '001000', 'rs', 'rt', 'immediate']


Instruction Decode :
STALL


Execute Stage :
Instruction : ['26', 'addi', '001000', 0, 11, 10]
ALUOutput : 10


Memory Stage :
Instruction : ['01', 'add', '000000', 0, 0, 10, 0, 32]
NPC : 8


Write Back Stage :
Instruction : ['01', 'add', '000000', 0, 0, 9, 0, 32]
Registers : {0: 0, 1: 901, 2: 902, 3: 903, 4: 904, 5: 905, 6: 906, 7: 907, 8: 90
8, 9: 0, 10: 910, 11: 911, 12: 912, 13: 913, 14: 914, 15: 915, 16: 916, 17: 917,
 18: 918, 19: 919, 20: 920, 21: 921, 22: 922, 23: 923, 24: 924, 25: 925, 26: 926
, 27: 927, 28: 928, 29: 929, 30: 930, 31: 931}
********************************************************************************
```

In clock 4 , an instruction is being fetched but instruction decode is stalled. In execute stage the “Instruction” addi was executed and the resulting ALUOutput is displayed. There is an instruction in Memory and in writeback stage as well. The add instruction is modifying the registers, in this case register 9 and setting it to zero.

```
******************************
In Clock  49
Instruction Fetch
PC : 52
NPC : 56
Instruction Register : ['09', 'syscall', '000000', 'code', '001100']


Instruction Decode :
Instruction : ['01', 'slt', '000000', 'rs', 'rt', 'rd', '00000', '101010']
Temporary Register A : 6
Temporary Register B : 10
Immediate : None
Decoded Instruction : ['01', 'slt', '000000', 10, 11, 1, 0, 42]


Execute Stage :
STALL


Memory Stage :
STALL


Write Back Stage :
Instruction : ['26', 'addi', '001000', 10, 10, 1]
Registers : {0: 0, 1: 1, 2: 902, 3: 903, 4: 904, 5: 905, 6: 906, 7: 907, 8: 908,
 9: 15, 10: 6, 11: 10, 12: 912, 13: 913, 14: 914, 15: 915, 16: 916, 17: 917, 18:
 918, 19: 919, 20: 920, 21: 921, 22: 922, 23: 923, 24: 924, 25: 925, 26: 926, 27
: 927, 28: 928, 29: 929, 30: 930, 31: 931}
********************************************************************************
******************************
```

In the figure above Memory and Execute stage is stalled. SLT instruction is being decoded in the decode stage. Addi is being executed in the writeback stage and syscall instruction is being fetched in instruction fetch.

Final Result:

```
In Clock  98
Instruction Fetch


Instruction Decode :


Execute Stage :


Memory Stage :


Write Back Stage :
Instruction : ['01', 'add', '000000', 0, 9, 2, 0, 32]
Registers : {0: 0, 1: 268500992, 2: 45, 3: 903, 4: 268500992, 5: 905, 6: 906, 7:
 907, 8: 908, 9: 45, 10: 10, 11: 10, 12: 912, 13: 913, 14: 914, 15: 915, 16: 916
, 17: 917, 18: 918, 19: 919, 20: 920, 21: 921, 22: 922, 23: 923, 24: 924, 25: 92
5, 26: 926, 27: 927, 28: 928, 29: 929, 30: 930, 31: 931}
********************************************************************************
```

The figure above shows the final clockstages and the resulting register values.

Final Output : Register 2 : 45

Similarly for other files :

>>binary2.txt

```
********************************************************************************
*******************************
        26      addi    001000  0       9       3
        26      addi    001000  0       10      7
        02      sll     000000  0       10      11      4       0
        01      add     000000  11      9       11      0       32
        01      add     000000  11      0       4       0       32
        26      addiu   001001  0       2       1
        09      syscall 000000  12      12
        20      lui     001111  0       1       4097
        26      ori     001101  1       4       0
        26      addiu   001001  0       2       4
        09      syscall 000000  12      12
        01      add     000000  0       11      2       0       32
********************************************************************************
```

The above image shows the Decoded Instructions

```
*******************************
In Clock  24
Instruction Fetch


Instruction Decode :


Execute Stage :


Memory Stage :


Write Back Stage :
Instruction : ['01', 'add', '000000', 0, 11, 2, 0, 32]
Registers : {0: 0, 1: 268500992, 2: 115, 3: 903, 4: 268500992, 5: 905, 6: 906, 7
: 907, 8: 908, 9: 3, 10: 7, 11: 115, 12: 912, 13: 913, 14: 914, 15: 915, 16: 916
, 17: 917, 18: 918, 19: 919, 20: 920, 21: 921, 22: 922, 23: 923, 24: 924, 25: 92
5, 26: 926, 27: 927, 28: 928, 29: 929, 30: 930, 31: 931}
********************************************************************************
```

Final Result : Register 2 : 115

>>binary4.txt

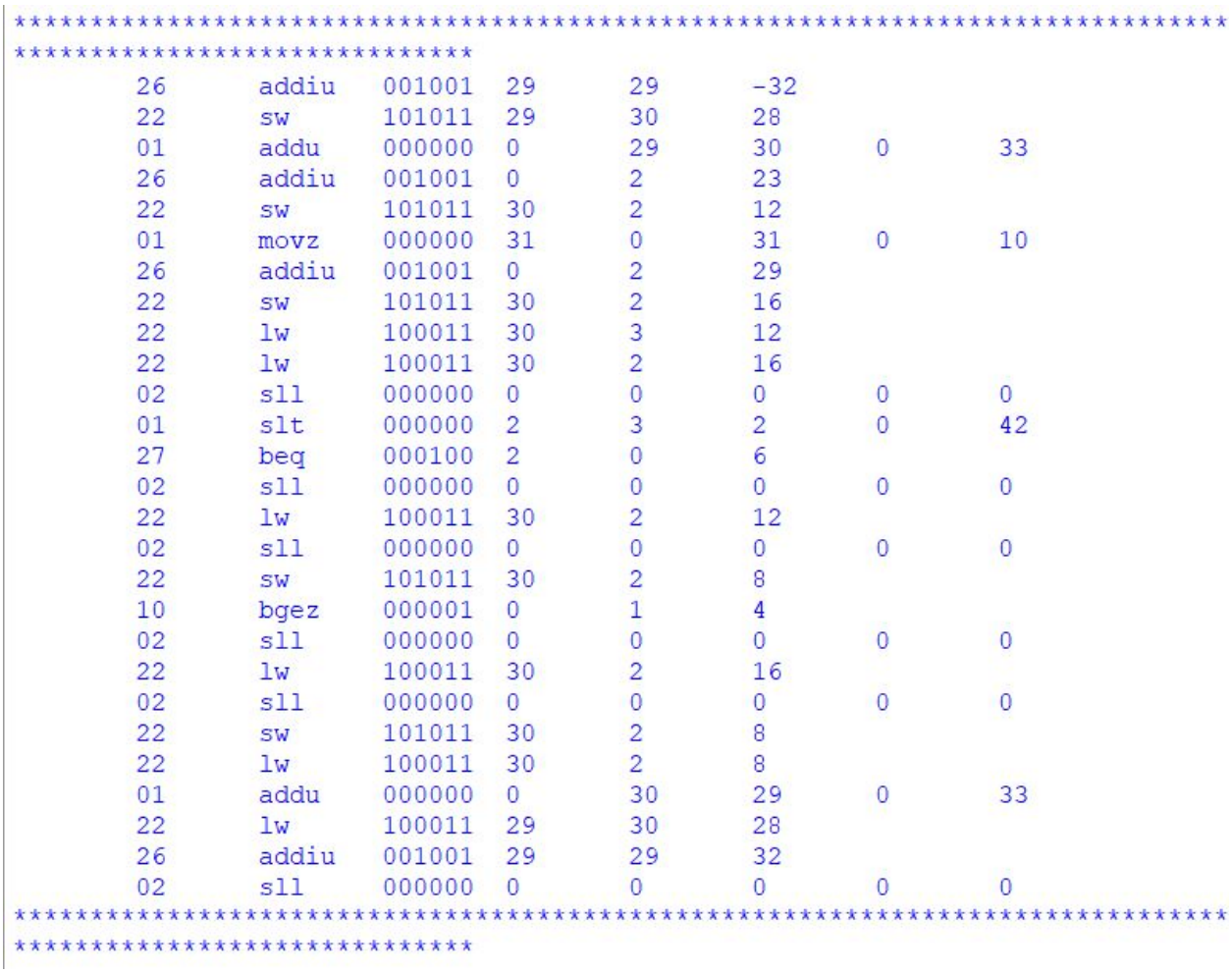
```
*******************************
        26      addi    001000  0       8       23
        26      addi    001000  0       9       29
        01      add     000000  0       0       10      0       32
        01      slt     000000  8       9       1       0       42
        27      bne     000101  1       0       2
        01      add     000000  9       0       10      0       32
        24      j       000010  1048584
        01      add     000000  9       0       10      0       32
        01      add     000000  0       10      2       0       32
*********************************************************************************************
```

The above image shows the Decoded Instructions

```
*******************************
In Clock  15
Instruction Fetch


Instruction Decode :


Execute Stage :


Memory Stage :


Write Back Stage :
Instruction : ['01', 'add', '000000', 0, 10, 2, 0, 32]
Registers : {0: 0, 1: 1, 2: 29, 3: 903, 4: 904, 5: 905, 6: 906, 7: 907, 8: 23, 9
: 29, 10: 29, 11: 911, 12: 912, 13: 913, 14: 914, 15: 915, 16: 916, 17: 917, 18:
 918, 19: 919, 20: 920, 21: 921, 22: 922, 23: 923, 24: 924, 25: 925, 26: 926, 27
: 927, 28: 928, 29: 929, 30: 930, 31: 931}
*********************************************************************************
```

Final Result : Register 2 : 29

>>hexcodebinary4.txt



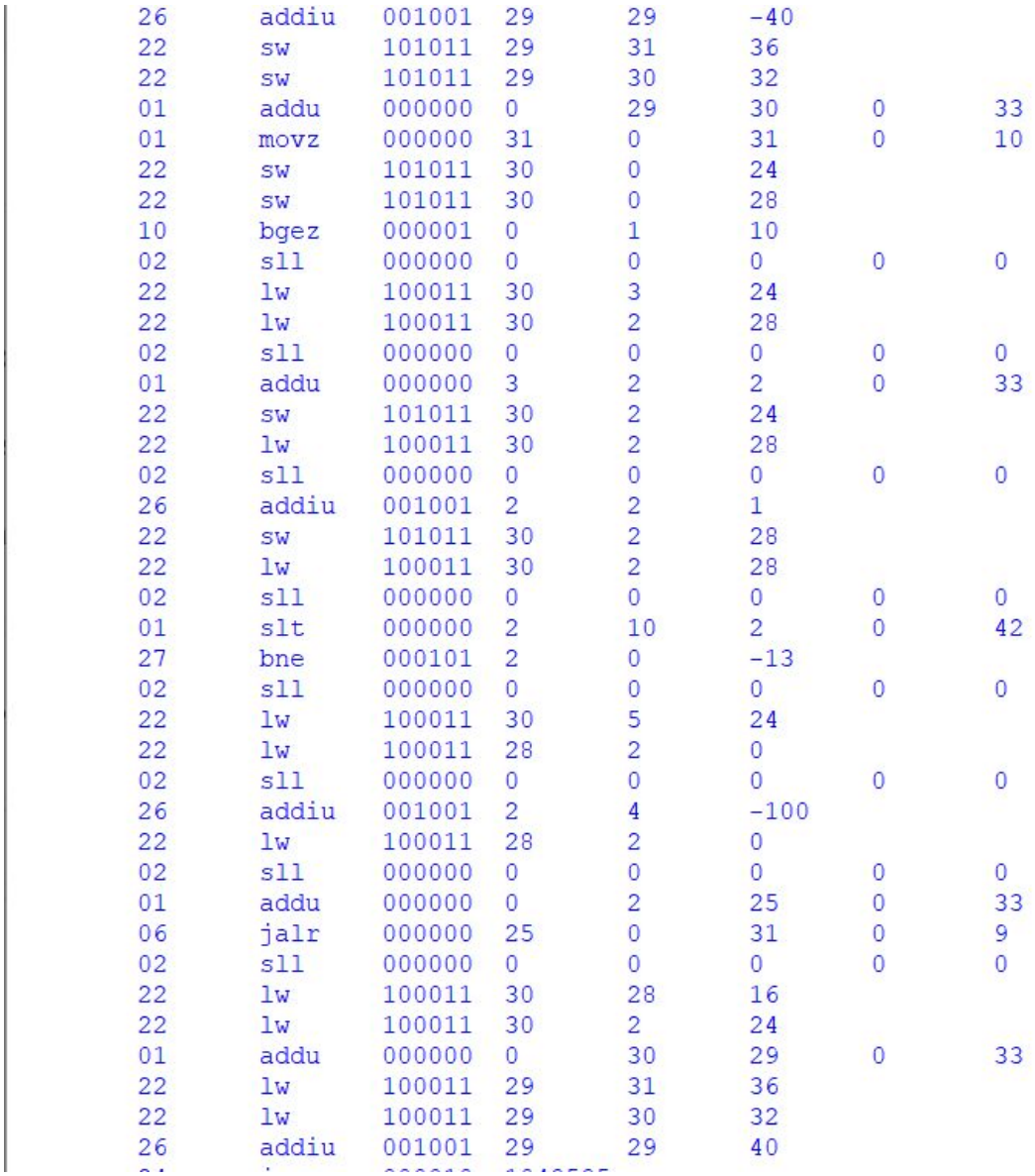
```
********************************************************************************
*******************************
        26      addiu   001001  29      29      -32
        22      sw      101011  29      30      28
        01      addu    000000  0       29      30      0       33
        26      addiu   001001  0       2       23
        22      sw      101011  30      2       12
        01      movz    000000  31      0       31      0       10
        26      addiu   001001  0       2       29
        22      sw      101011  30      2       16
        22      lw      100011  30      3       12
        22      lw      100011  30      2       16
        02      sll     000000  0       0       0       0       0
        01      slt     000000  2       3       2       0       42
        27      beq     000100  2       0       6
        02      sll     000000  0       0       0       0       0
        22      lw      100011  30      2       12
        02      sll     000000  0       0       0       0       0
        22      sw      101011  30      2       8
        10      bgez    000001  0       1       4
        02      sll     000000  0       0       0       0       0
        22      lw      100011  30      2       16
        02      sll     000000  0       0       0       0       0
        22      sw      101011  30      2       8
        22      lw      100011  30      2       8
        01      addu    000000  0       30      29      0       33
        22      lw      100011  29      30      28
        26      addiu   001001  29      29      32
        02      sll     000000  0       0       0       0       0
********************************************************************************
*******************************
```

The above image shows the Decoded Instructions

```
*******************************
In Clock  32
Instruction Fetch


Instruction Decode :


Execute Stage :


Memory Stage :


Write Back Stage :
Instruction : ['02', 'sll', '000000', 0, 0, 0, 0, 0]
Registers : {0: 0, 1: 901, 2: 29, 3: 23, 4: 904, 5: 905, 6: 906, 7: 907, 8: 908,
 9: 909, 10: 910, 11: 911, 12: 912, 13: 913, 14: 914, 15: 915, 16: 916, 17: 917,
 18: 918, 19: 919, 20: 920, 21: 921, 22: 922, 23: 923, 24: 924, 25: 925, 26: 926
, 27: 927, 28: 928, 29: 929, 30: 930, 31: 931}
********************************************************************************
-------------------------------
```

Final Result : Register 2 : 29

>>hexcodebinary1.txt



```
26      addiu   001001  29      29      -40
22      sw      101011  29      31      36
22      sw      101011  29      30      32
01      addu    000000  0       29      30      0       33
01      movz    000000  31      0       31      0       10
22      sw      101011  30      0       24
22      sw      101011  30      0       28
10      bgez    000001  0       1       10
02      sll     000000  0       0       0       0       0
22      lw      100011  30      3       24
22      lw      100011  30      2       28
02      sll     000000  0       0       0       0       0
01      addu    000000  3       2       2       0       33
22      sw      101011  30      2       24
22      lw      100011  30      2       28
02      sll     000000  0       0       0       0       0
26      addiu   001001  2       2       1
22      sw      101011  30      2       28
22      lw      100011  30      2       28
02      sll     000000  0       0       0       0       0
01      slt     000000  2       10      2       0       42
27      bne     000101  2       0       -13
02      sll     000000  0       0       0       0       0
22      lw      100011  30      5       24
22      lw      100011  28      2       0
02      sll     000000  0       0       0       0       0
26      addiu   001001  2       4       -100
22      lw      100011  28      2       0
02      sll     000000  0       0       0       0       0
01      addu    000000  0       2       25      0       33
06      jalr    000000  25      0       31      0       9
02      sll     000000  0       0       0       0       0
22      lw      100011  30      28      16
22      lw      100011  30      2       24
01      addu    000000  0       30      29      0       33
22      lw      100011  29      31      36
22      lw      100011  29      30      32
26      addiu   001001  29      29      40
```

The above image shows the Decoded Instructions

This file sends my program into an infinite loop

>>hexcodebinary2.txt

```
*******************************
        26      addiu   001001  29      29      -48
        22      sw      101011  29      31      44
        22      sw      101011  29      30      40
        01      addu    000000  0       29      30      0       33
        01      movz    000000  31      0       31      0       10
        26      addiu   001001  0       2       3
        22      sw      101011  30      2       24
        26      addiu   001001  0       2       7
        22      sw      101011  30      2       28
        22      lw      100011  30      2       28
        02      sll     000000  0       0       0       0       0
        02      sll     000000  0       2       3       4       0
        22      lw      100011  30      2       24
        02      sll     000000  0       0       0       0       0
        01      addu    000000  3       2       2       0       33
        22      sw      101011  30      2       32
        22      lw      100011  30      5       32
        22      lw      100011  28      2       0
        02      sll     000000  0       0       0       0       0
        26      addiu   001001  2       4       -100
        22      lw      100011  28      2       0
        02      sll     000000  0       0       0       0       0
        01      addu    000000  0       2       25      0       33
        06      jalr    000000  25      0       31      0       9
        02      sll     000000  0       0       0       0       0
        22      lw      100011  30      28      16
        22      lw      100011  30      2       32
        01      addu    000000  0       30      29      0       33
        22      lw      100011  29      31      44
        22      lw      100011  29      30      40
        26      addiu   001001  29      29      48
        24      j       000010  1048599
        02      sll     000000  0       0       0       0       0
*********************************************************************************
```

The above image shows the Decoded Instructions

This program crashes as there is a load which takes register 28 as base, even though register 28 is not modified in the program.