Alex Smith, Vishal Parikh, Vitaly Slabada

ECE 366: Computer Organization

UIC - Spring 2019

Project 2: MIPS SIM on Python

Activity log:

Time / Location	**Activity**	**Achieved / To-Do**	**Member(s)**
Feb 22, 2019 SEL 2nd floor Computer lab	In-person group meeting	discussed overall meeting schedule, various approaches for PRPG selection. Arithmetic, logic functions, github setup	Alex, Vishal, Vitaly
Feb 24, 2019 WhatsApp Group	Remotely,	arrays for instructions, test cases, some transformations of code to allow for branching instructions.	Alex, Vishal, Vitaly
Feb 26, 2019 Library	In-person group discussion	discuss the lw/sw formula and worked on jump and some test cases	Alex, Vishal, Vitaly
Feb 25, 2019 After class, classroom	In-person group meeting	LW, SW, jump, or, xor, test cases, bug fixing	Alex, Vishal, Vitaly
Feb 27, 2019 After class, classroom	In-person group meeting	Discussion of responsibilities for the remainder of the project. PRPG code, Test cases, Report	Alex, Vishal, Vitaly
Feb 28, 2019 WhatsApp Group	Remotely,	Finishing touches, report, testing	Alex, Vishal, Vitaly

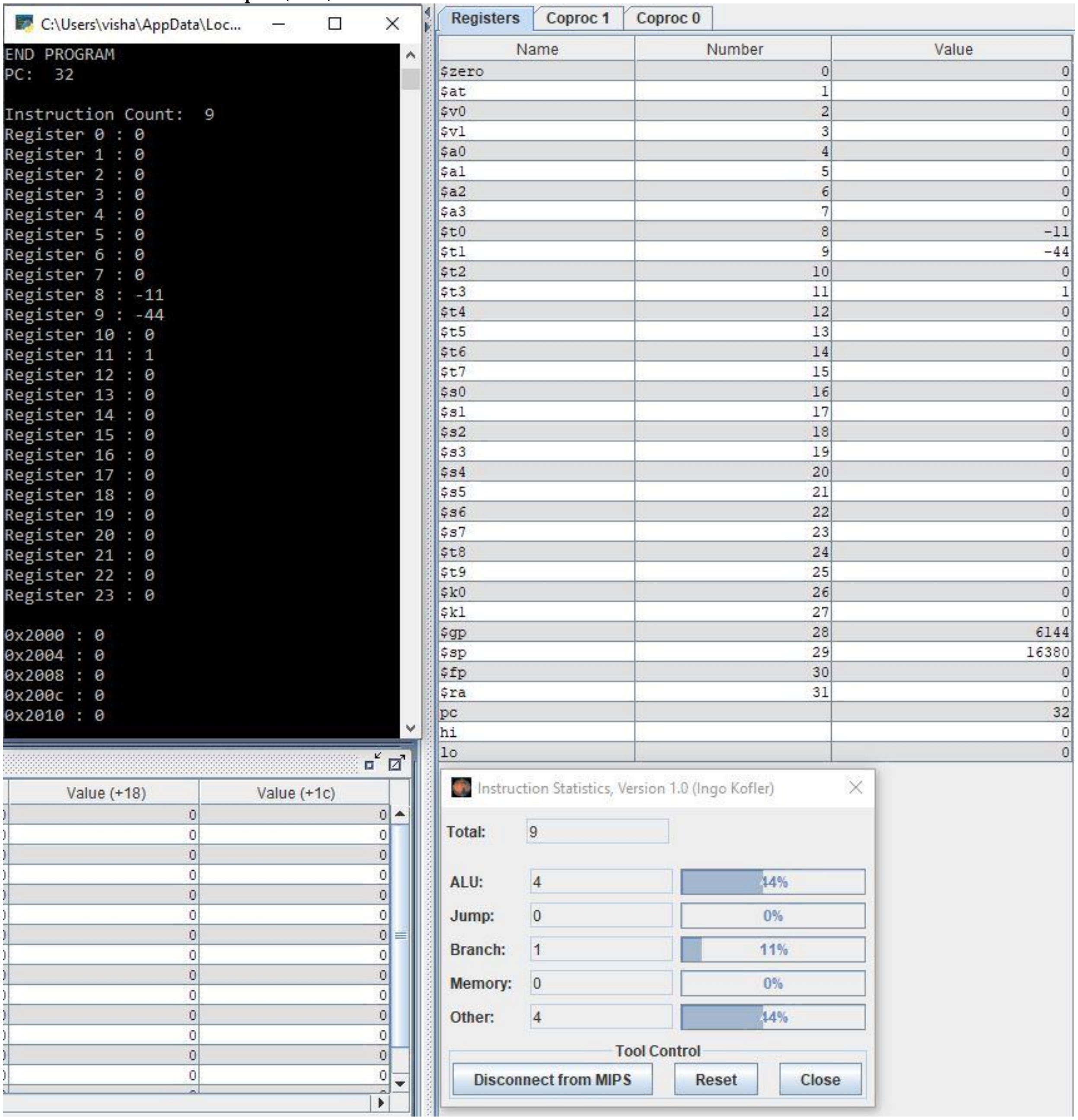
List of activities by yourself:

VISHAL
Mon Feb 25: we had a meeting before class, after class started working on lw/sw memory instructions, spent 4-5 hours.
Tue Feb 26: had some issues with lw/sw address calculations, I went to see prof. Rao at 3pm By the end of the day figured out the right equation to calculate right address in python Also worked on some test cases.
Wed Feb 27: worked on test case 4a and 4b also figured out issues with program and spent few hours debugging, also I took my project 1 code and changed into Hex code and started running prpg program on python which led to more issues, I spent almost half of my day working on project. Eventually started testing seeds and was able to get expected result for 4 seeds by the end of the day
Thu Feb 28: as I had issues with prpg, I spent a lot of time debugging in mips as well as in python, solved many issues with help of group partners, at the end I was able to get simulator working. Then, I started working on report and double checked all test cases and organized repository.
In short, I implemented memory instructions, add instruction, part of prpg program and a lot of debugging and few test cases.

I) Test cases

#1A

```
0x20080019 #addi $8, $0, 25
0x2108ffdc #addi $8, $8, -36
0x01084821 #addu $9, $8, $8
0x01284821 #addu $9, $9, $8
0x00095022 #sub $10, $0, $9
0x010a4822 #sub $9, $8, $10
0x0009502a #slt $10, $0, $9
0x010a582a #slt $11, $8, $10
0x1000ffff #end: beq $0, $0, end
```



C:\Users\visha\AppData\Loc...
Registers
Coproc 1
Coproc 0
Instruction Statistics, Version 1.0 (Ingo Kofler)
Tool Control
Disconnect from MIPS
Reset
Close

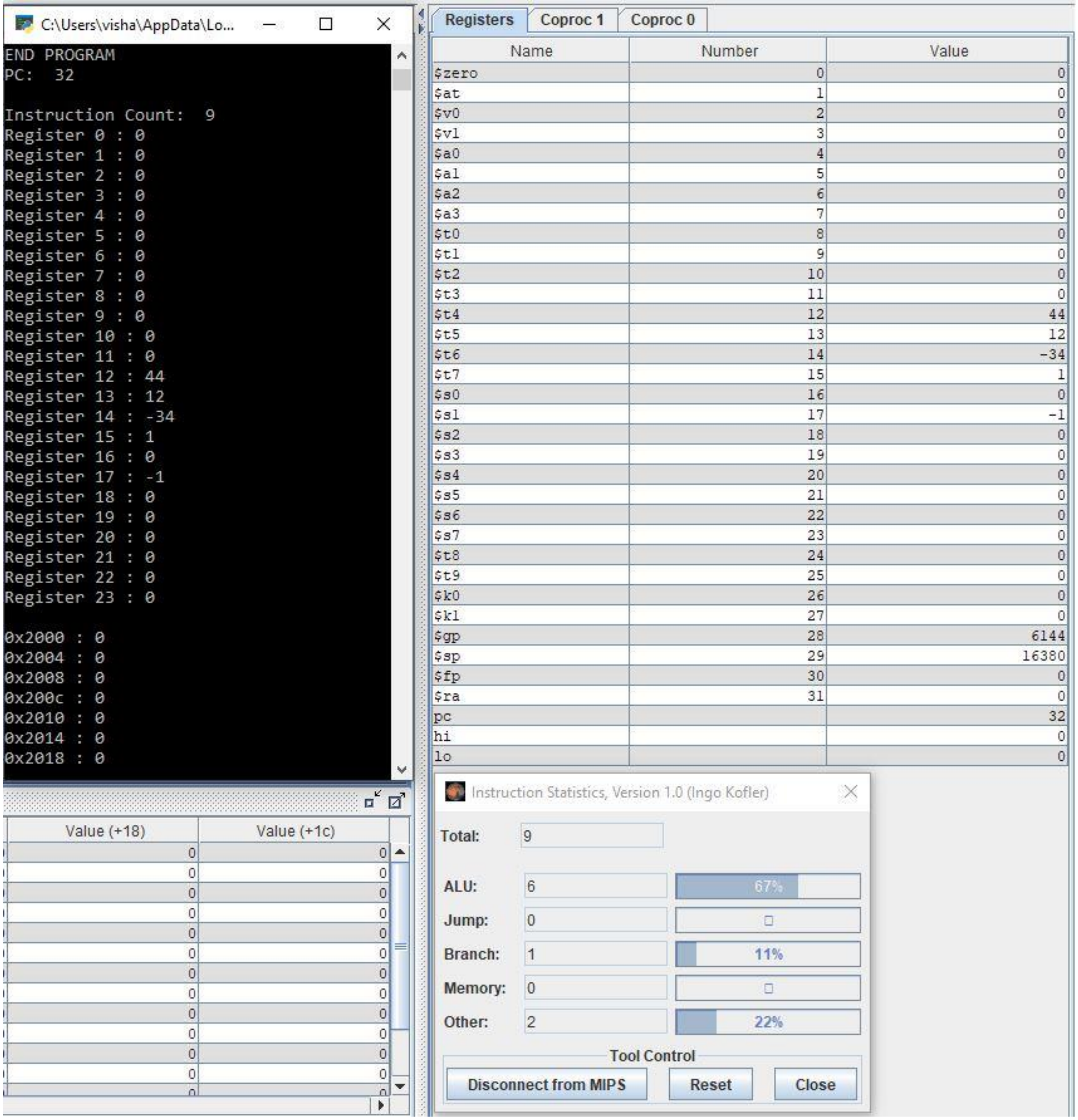
```
END PROGRAM
PC:  32

Instruction Count:  9
Register 0 : 0
Register 1 : 0
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : -11
Register 9 : -44
Register 10 : 0
Register 11 : 1
Register 12 : 0
Register 13 : 0
Register 14 : 0
Register 15 : 0
Register 16 : 0
Register 17 : 0
Register 18 : 0
Register 19 : 0
Register 20 : 0
Register 21 : 0
Register 22 : 0
Register 23 : 0

0x2000 : 0
0x2004 : 0
0x2008 : 0
0x200c : 0
0x2010 : 0
```

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	-11
$t1	9	-44
$t2	10	0
$t3	11	1
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		32
hi		0
lo		0

Value (+18)	Value (+1c)
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Total:	9	
ALU:	4	44%
Jump:	0	0%
Branch:	1	11%
Memory:	0	0%
Other:	4	44%

```
#1B
0x200c0038 #addi $12, $0, 56
0x200d000c #addi $13, $0, 12
0x200effea #addi $14, $0, -22
0x01cd7022 #sub  $14, $14, $13
0x018e802a #slt  $16, $12, $14
0x018d6022 #sub  $12, $12, $13
0x01c0782a #slt  $15, $14, $0
0x020f8822 #sub  $17, $16, $15
0x1000ffff  #end: beq $0, $0, end
```



```
END PROGRAM
PC:   32

Instruction Count:  9
Register 0 : 0
Register 1 : 0
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 0
Register 9 : 0
Register 10 : 0
Register 11 : 0
Register 12 : 44
Register 13 : 12
Register 14 : -34
Register 15 : 1
Register 16 : 0
Register 17 : -1
Register 18 : 0
Register 19 : 0
Register 20 : 0
Register 21 : 0
Register 22 : 0
Register 23 : 0

0x2000 : 0
0x2004 : 0
0x2008 : 0
0x200c : 0
0x2010 : 0
0x2014 : 0
0x2018 : 0
```

Value (+18)	Value (+1c)
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

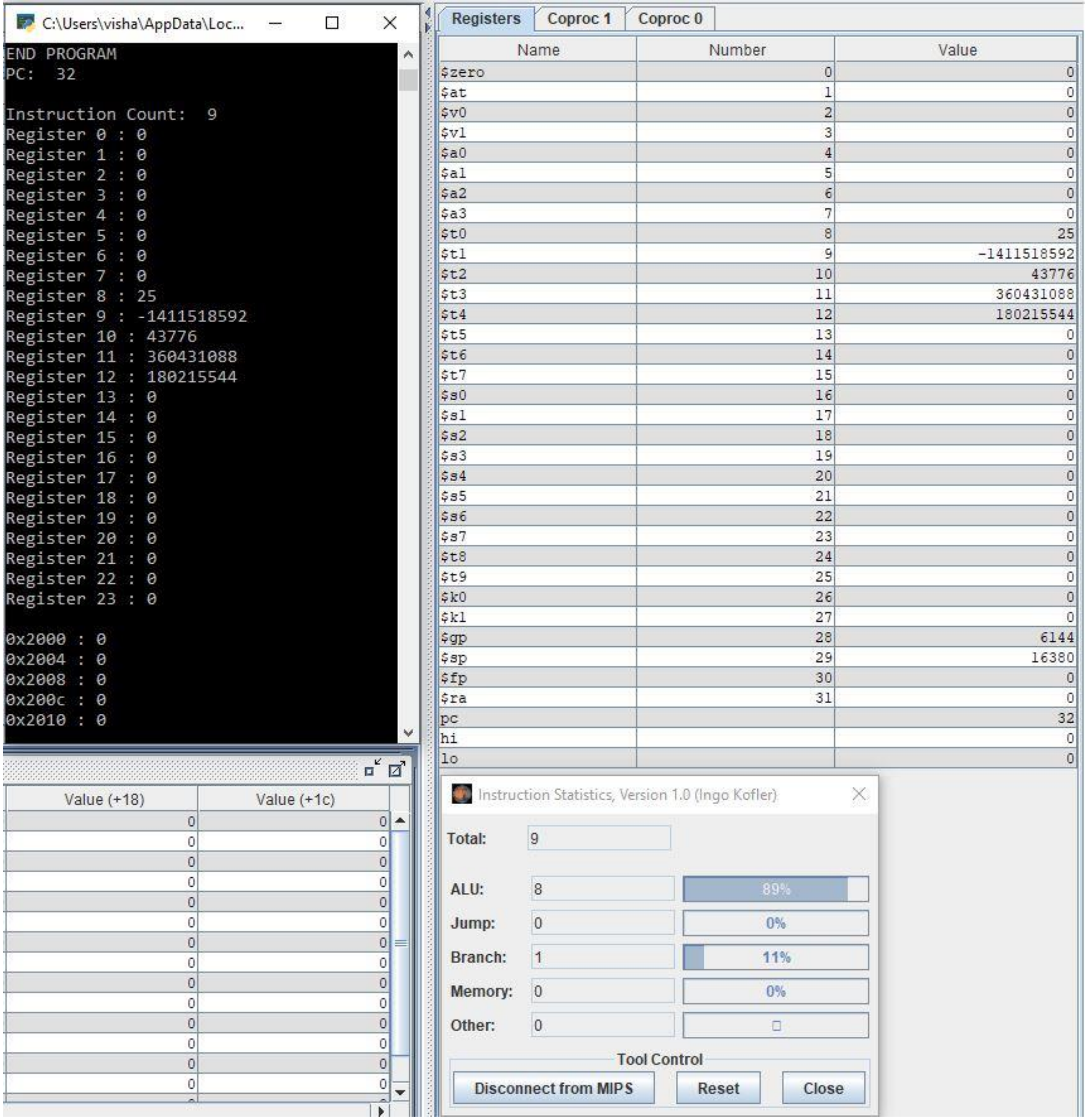
Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	44
$t5	13	12
$t6	14	-34
$t7	15	1
$s0	16	0
$s1	17	-1
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		32
hi		0
lo		0

Instruction Statistics, Version 1.0 (Ingo Kofler)

Total: 9

ALU: 6 67%
Jump: 0 □
Branch: 1 11%
Memory: 0 □
Other: 2 22%

#2A
0x34080019 #ori $8, $0, 25
0x3509abcd #ori $9, $8, 0xabcd
0x340aff00 #ori $10, $0, 0xff00
0x012a5024 #and $10, $9, $10
0x00094c00 #sll $9, $9, 16
0x3529ef80 #ori $9, $9, 0xEF80
0x000958c2 #srl $11, $9, 3
0x000b6042 #srl $12, $11, 1
0x1000ffff #end: beq $0, $0, end


C:\Users\visha\AppData\Loc...
Registers
Coproc 1
Coproc 0
Instruction Statistics, Version 1.0 (Ingo Kofler)
Tool Control
Disconnect from MIPS
Reset
Close

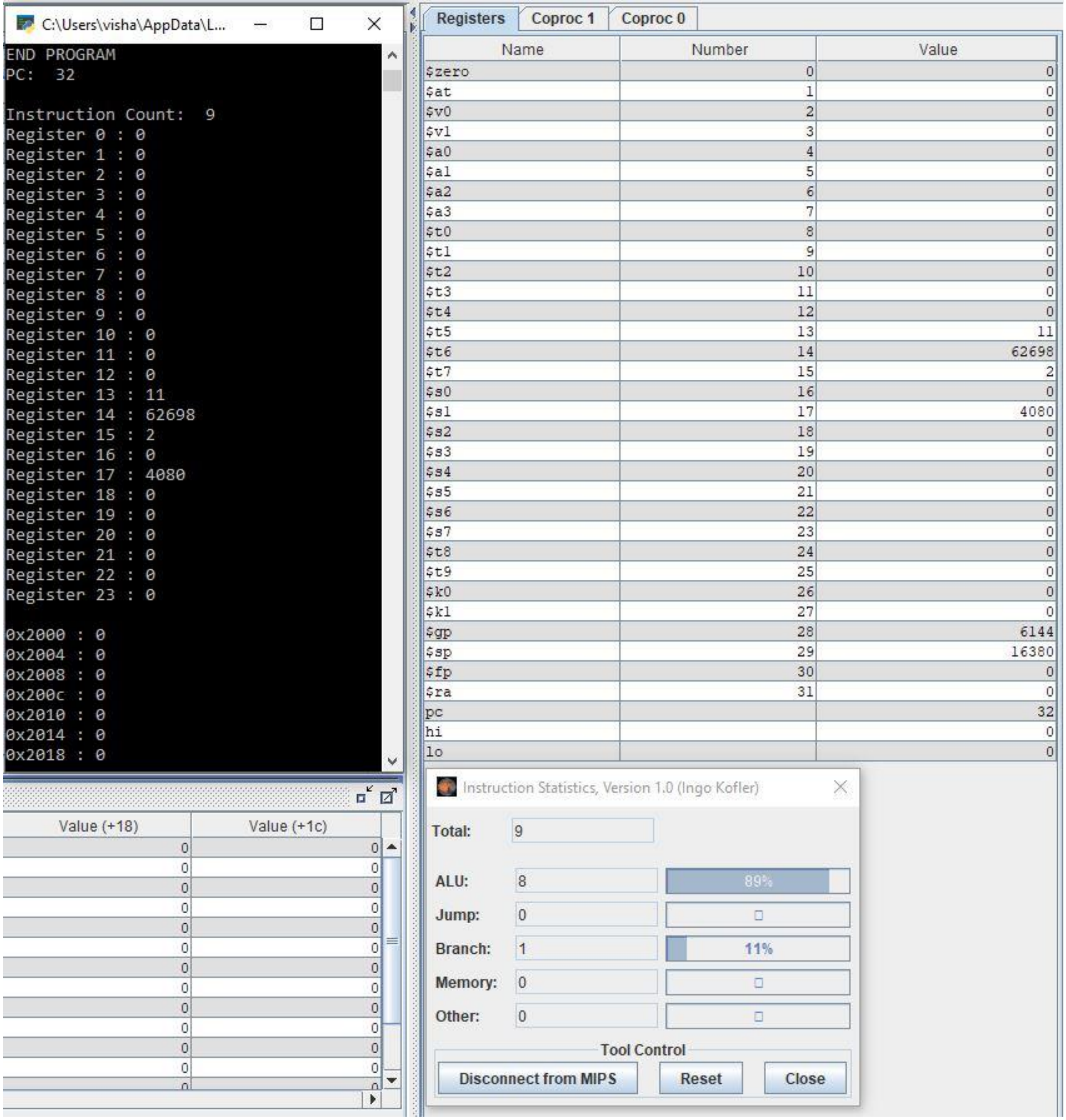
```
END PROGRAM
PC:   32

Instruction Count:  9
Register 0 : 0
Register 1 : 0
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 25
Register 9 : -1411518592
Register 10 : 43776
Register 11 : 360431088
Register 12 : 180215544
Register 13 : 0
Register 14 : 0
Register 15 : 0
Register 16 : 0
Register 17 : 0
Register 18 : 0
Register 19 : 0
Register 20 : 0
Register 21 : 0
Register 22 : 0
Register 23 : 0

0x2000 : 0
0x2004 : 0
0x2008 : 0
0x200c : 0
0x2010 : 0
```

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	25
$t1	9	-1411518592
$t2	10	43776
$t3	11	360431088
$t4	12	180215544
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		32
hi		0
lo		0

Value (+18)	Value (+1c)
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Total:	9	
ALU:	8	89%
Jump:	0	0%
Branch:	1	11%
Memory:	0	0%
Other:	0	□

#2B
0x340d002c #ori $13,$0,44
0x340ef4ea #ori $14,$0,62698
0x01ae7824 #and $15,$13,$14
0x000f7902 #srl $15,$15,4
0x01ed8024 #and $16,$15,$13
0x00108200 #sll $16,$16,8
0x000d6882 #srl $13,$13,2
0x36110ff0 #ori $17,$16,4080
0x1000ffff #end: beq $0, $0, end



```
END PROGRAM
PC:  32

Instruction Count:  9
Register 0 : 0
Register 1 : 0
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 0
Register 9 : 0
Register 10 : 0
Register 11 : 0
Register 12 : 0
Register 13 : 11
Register 14 : 62698
Register 15 : 2
Register 16 : 0
Register 17 : 4080
Register 18 : 0
Register 19 : 0
Register 20 : 0
Register 21 : 0
Register 22 : 0
Register 23 : 0

0x2000 : 0
0x2004 : 0
0x2008 : 0
0x200c : 0
0x2010 : 0
0x2014 : 0
0x2018 : 0
```

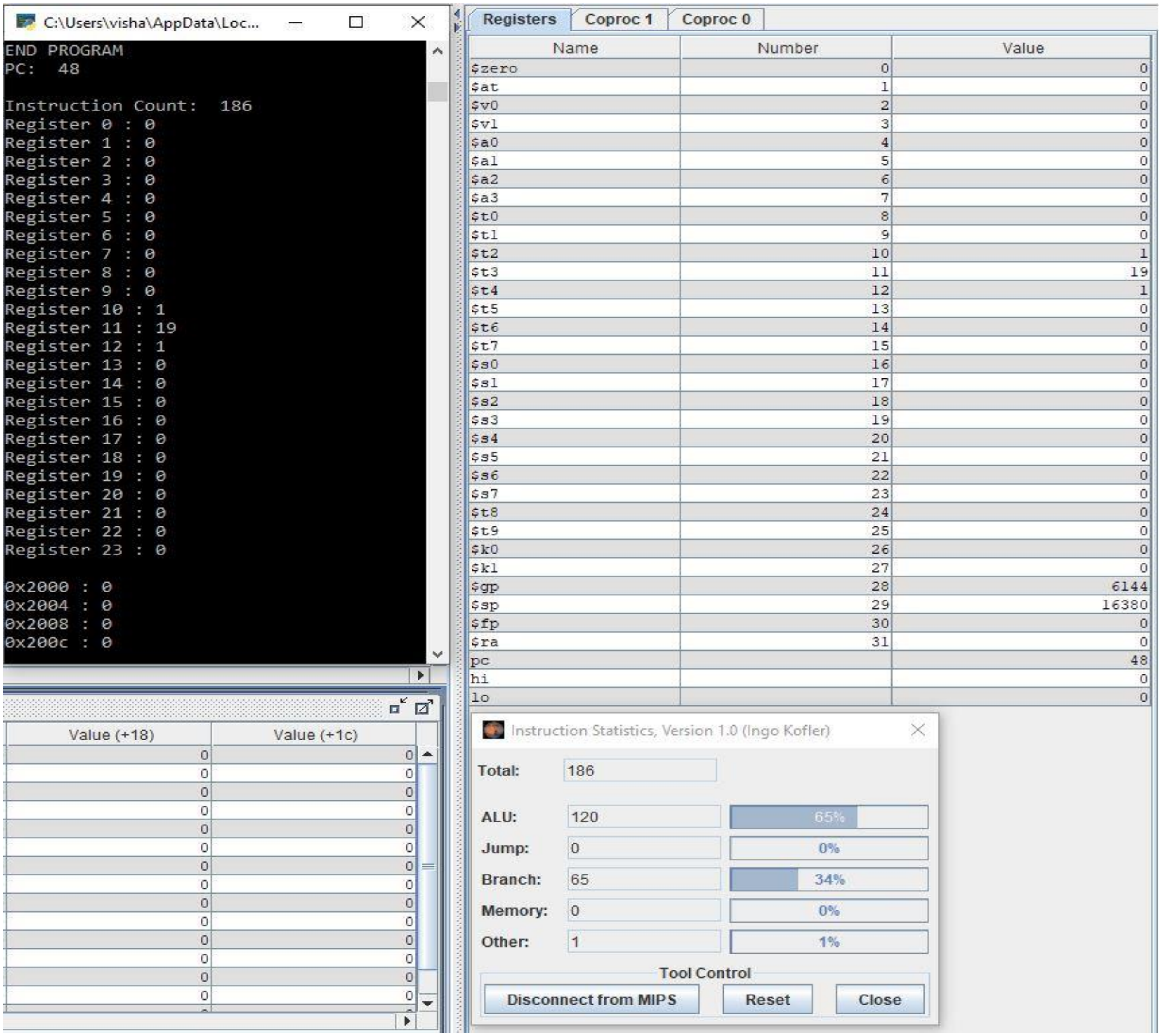
Value (+18)	Value (+1c)
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	11
$t6	14	62698
$t7	15	2
$s0	16	0
$s1	17	4080
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		32
hi		0
lo		0

Total: 9

ALU: 8 89%
Jump: 0
Branch: 1 11%
Memory: 0
Other: 0

#3A
0x20080020
0x3409abcd
0x00094c00
0x3529ef12
0x200a0001
0x00005821
0x012a6024
0x11800001
0x216b0001
0x00094842
0x2108ffff
0x1500fffa
0x1000ffff



```
END PROGRAM
PC:   48

Instruction Count:   186
Register 0 : 0
Register 1 : 0
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 0
Register 9 : 0
Register 10 : 1
Register 11 : 19
Register 12 : 1
Register 13 : 0
Register 14 : 0
Register 15 : 0
Register 16 : 0
Register 17 : 0
Register 18 : 0
Register 19 : 0
Register 20 : 0
Register 21 : 0
Register 22 : 0
Register 23 : 0

0x2000 : 0
0x2004 : 0
0x2008 : 0
0x200c : 0
```

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	1
$t3	11	19
$t4	12	1
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		48
hi		0
lo		0

Value (+18)	Value (+1c)
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

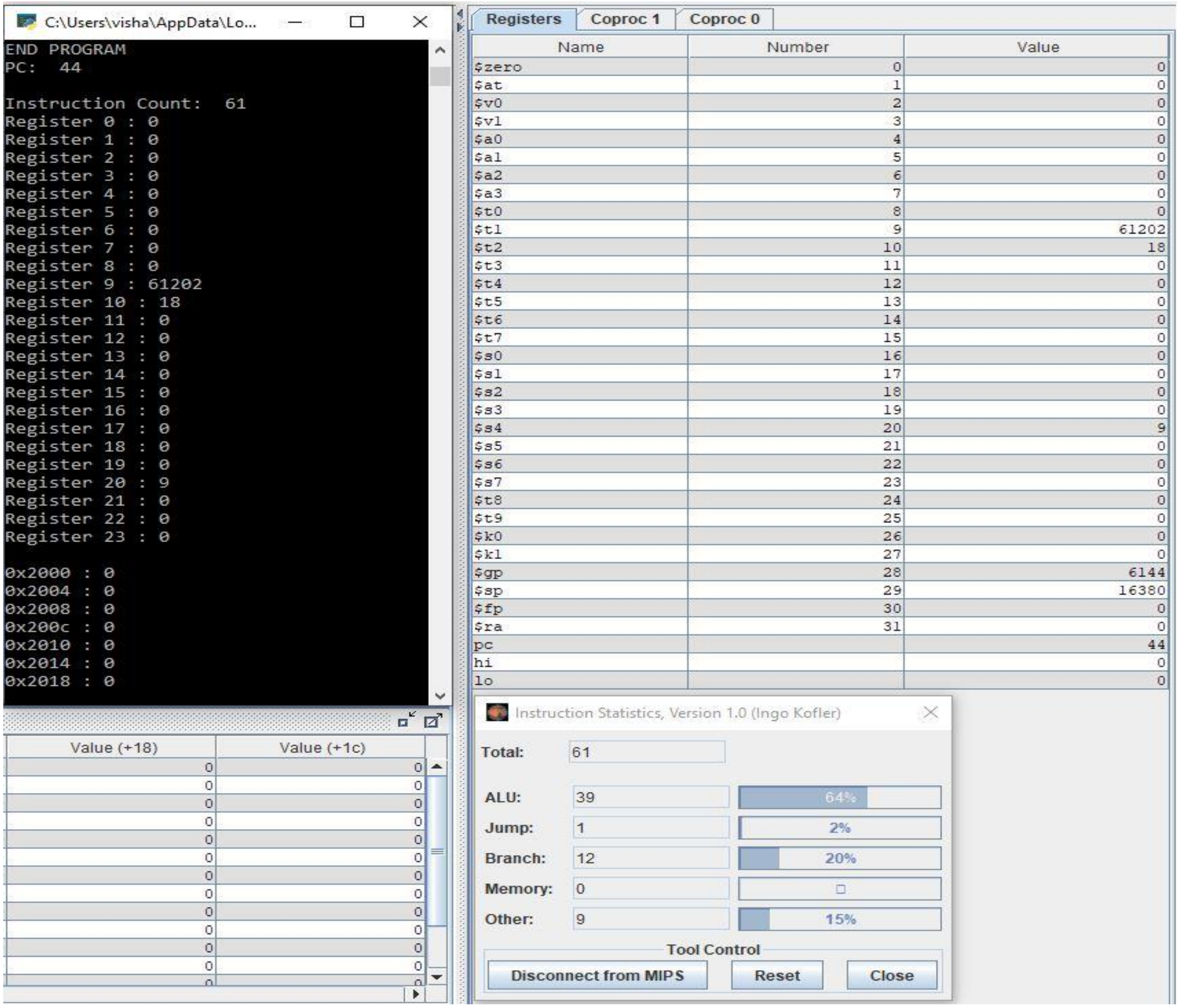
Instruction Statistics, Version 1.0 (Ingo Kofler)

Total: 186

ALU:	120	65%
Jump:	0	0%
Branch:	65	34%
Memory:	0	0%
Other:	1	1%

Tool Control

```
#3B
0x08000002 #j jump
0x200d000d #addi $13, $0, 13
#jump:
0x2008000a #addi $8, $0, 10
0x20140000 #addi $20, $0,0
0x11800004 #beq $12, $0, skip
#loop:
0x3529ef12 #ori $9, $9, 0xef12
0x214a0002 #addi $10, $10, 2
0x00005821 #addu $11, $0, $0
0x22940001 #addi $20, $20,1
#skip:
0x2108ffff #addi $8, $8, -1
0x1500fffa #bne $8, $0, loop
0x1000ffff
```


C:\Users\visha\AppData\Lo...
Registers
Coproc 1
Coproc 0
Instruction Statistics, Version 1.0 (Ingo Kofler)
Tool Control
Disconnect from MIPS
Reset
Close

```
END PROGRAM
PC:  44

Instruction Count:  61
Register 0 : 0
Register 1 : 0
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 0
Register 9 : 61202
Register 10 : 18
Register 11 : 0
Register 12 : 0
Register 13 : 0
Register 14 : 0
Register 15 : 0
Register 16 : 0
Register 17 : 0
Register 18 : 0
Register 19 : 0
Register 20 : 9
Register 21 : 0
Register 22 : 0
Register 23 : 0

0x2000 : 0
0x2004 : 0
0x2008 : 0
0x200c : 0
0x2010 : 0
0x2014 : 0
0x2018 : 0
```

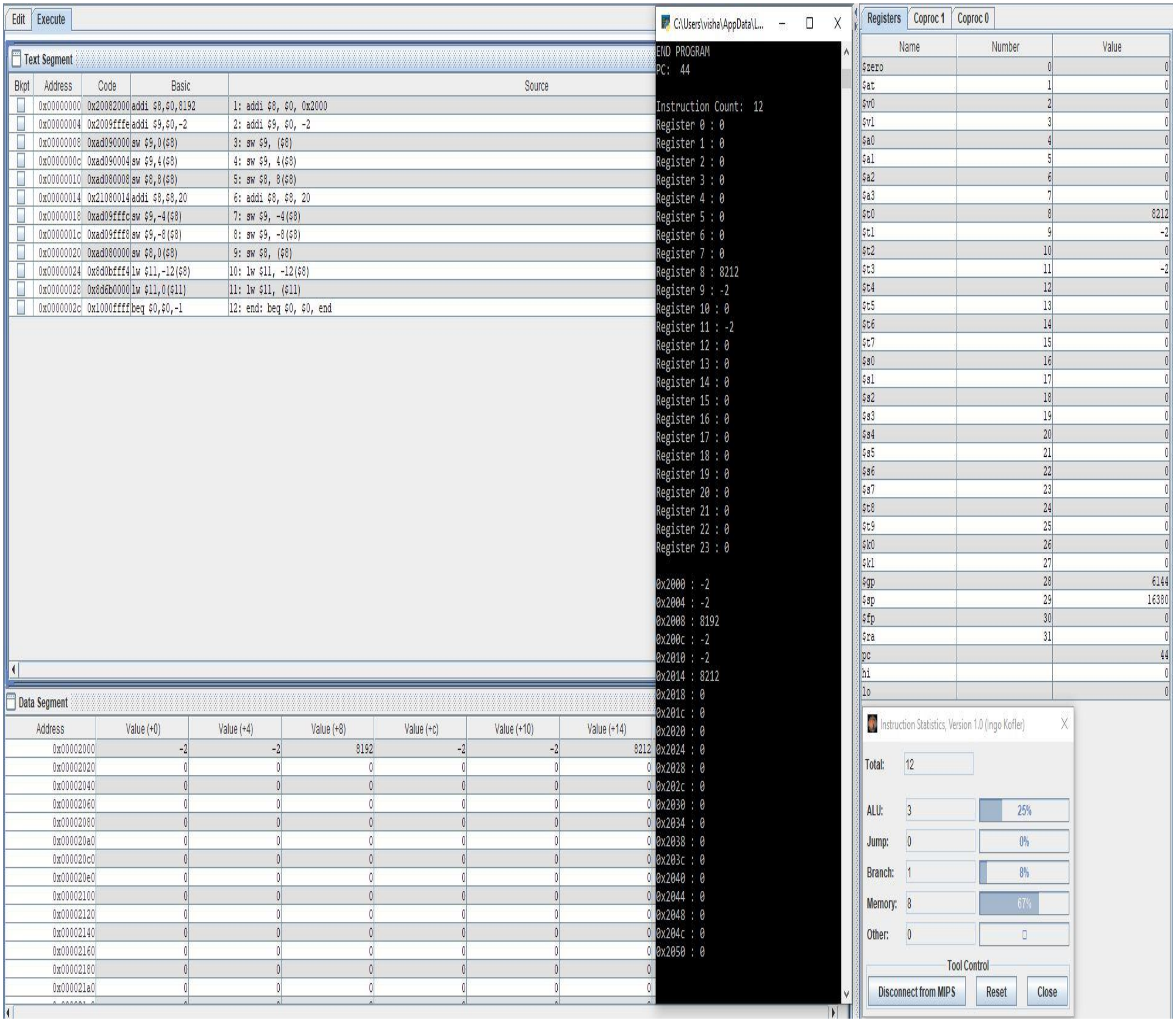
Value (+18)	Value (+1c)
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	61202
$t2	10	18
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	9
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		44
hi		0
lo		0

Total: 61

ALU:	39	64%
Jump:	1	2%
Branch:	12	20%
Memory:	0	□
Other:	9	15%

```
#4A        #.asm for 4A
0x20082000 #addi $8, $0, 0x2000
0x2009fffe #addi $9, $0, -2
0xad090000 #sw $9, ($8)
0xad090004 #sw $9, 4($8)
0xad080008 #sw $8, 8($8)
0x21080014 #addi $8, $8, 20
0xad09fffc #sw $9, -4($8)
0xad09fff8 #sw $9, -8($8)
0xad080000 #sw $8, ($8)
0x8d0bfff4 #lw $11, -12($8)
0x8d6b0000 #lw $11, ($11)
0x1000ffff #end: beq $0, $0, end
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00000000	0x20082000	addi $8,$0,8192	1: addi $8, $0, 0x2000
□	0x00000004	0x2009fffe	addi $9,$0,-2	2: addi $9, $0, -2
□	0x00000008	0xad090000	sw $9,0($8)	3: sw $9, ($8)
□	0x0000000c	0xad090004	sw $9,4($8)	4: sw $9, 4($8)
□	0x00000010	0xad080008	sw $8,8($8)	5: sw $8, 8($8)
□	0x00000014	0x21080014	addi $8,$8,20	6: addi $8, $8, 20
□	0x00000018	0xad09fffc	sw $9,-4($8)	7: sw $9, -4($8)
□	0x0000001c	0xad09fff8	sw $9,-8($8)	8: sw $9, -8($8)
□	0x00000020	0xad080000	sw $8,0($8)	9: sw $8, ($8)
□	0x00000024	0x8d0bfff4	lw $11,-12($8)	10: lw $11, -12($8)
□	0x00000028	0x8d6b0000	lw $11,0($11)	11: lw $11, ($11)
□	0x0000002c	0x1000ffff	beq $0,$0,-1	12: end: beq $0, $0, end

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x00002000	-2	-2	8192	-2	-2	8212
0x00002020	0	0	0	0	0	0
0x00002040	0	0	0	0	0	0
0x00002060	0	0	0	0	0	0
0x00002080	0	0	0	0	0	0
0x000020a0	0	0	0	0	0	0
0x000020c0	0	0	0	0	0	0
0x000020e0	0	0	0	0	0	0
0x00002100	0	0	0	0	0	0
0x00002120	0	0	0	0	0	0
0x00002140	0	0	0	0	0	0
0x00002160	0	0	0	0	0	0
0x00002180	0	0	0	0	0	0
0x000021a0	0	0	0	0	0	0

C:\Users\visha\AppData\L...

```
END PROGRAM
PC:  44

Instruction Count:  12
Register 0 : 0
Register 1 : 0
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 8212
Register 9 : -2
Register 10 : 0
Register 11 : -2
Register 12 : 0
Register 13 : 0
Register 14 : 0
Register 15 : 0
Register 16 : 0
Register 17 : 0
Register 18 : 0
Register 19 : 0
Register 20 : 0
Register 21 : 0
Register 22 : 0
Register 23 : 0

0x2000 : -2
0x2004 : -2
0x2008 : 8192
0x200c : -2
0x2010 : -2
0x2014 : 8212
0x2018 : 0
0x201c : 0
0x2020 : 0
0x2024 : 0
0x2028 : 0
0x202c : 0
0x2030 : 0
0x2034 : 0
0x2038 : 0
0x203c : 0
0x2040 : 0
0x2044 : 0
0x2048 : 0
0x204c : 0
0x2050 : 0
```

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	8212
$t1	9	-2
$t2	10	0
$t3	11	-2
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		44
hi		0
lo		0

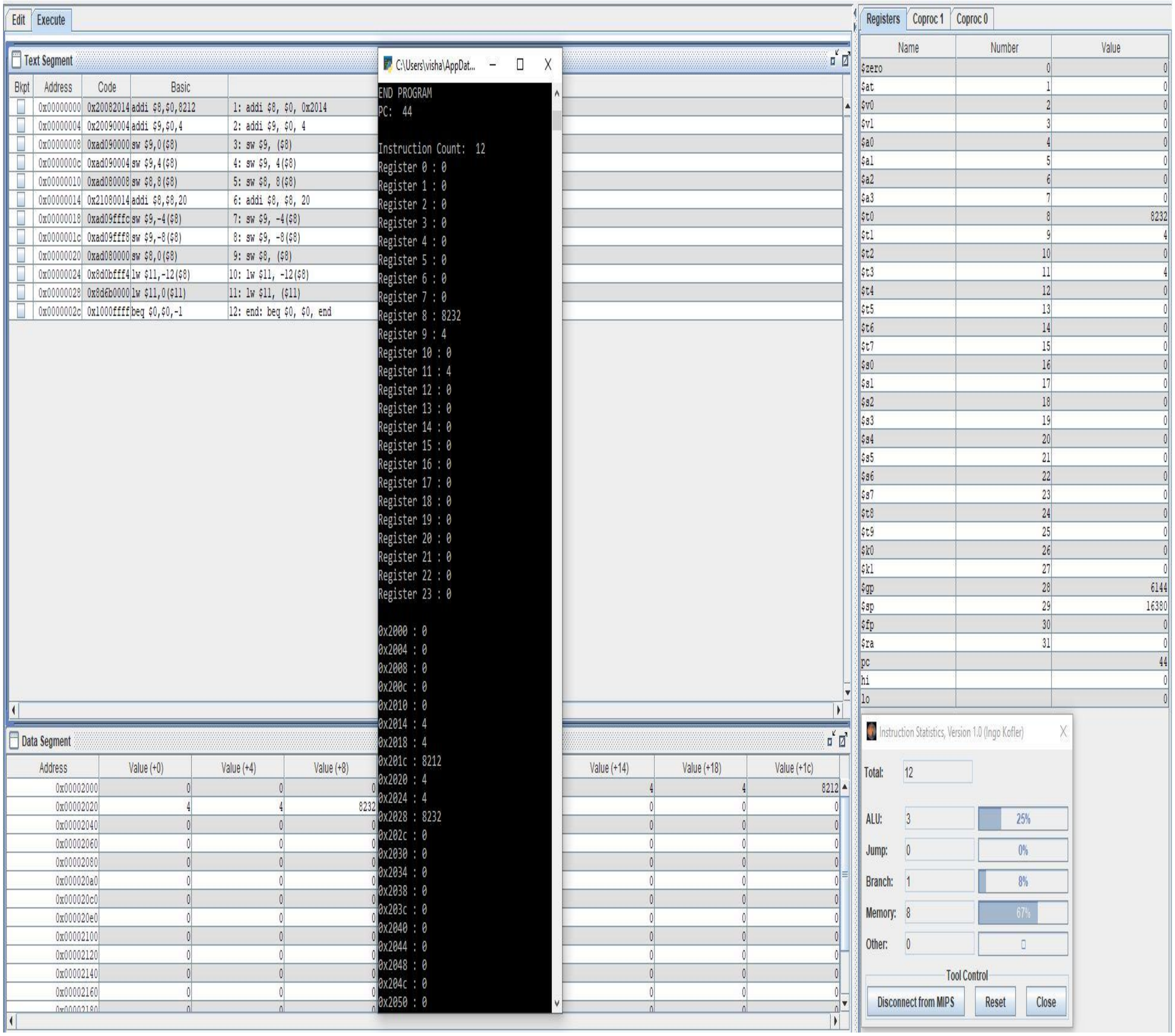
Instruction Statistics, Version 1.0 (Ingo Kofler)

Total: 12

ALU:	3	25%
Jump:	0	0%
Branch:	1	8%
Memory:	8	67%
Other:	0	□

Tool Control

```
#4B       #.asm for 4B
0x20082014 #addi $8, $0, 0x2014
0x20090004 #addi $9, $0, 4
0xad090000 #sw $9, ($8)
0xad090004 #sw $9, 4($8)
0xad080008 #sw $8, 8($8)
0x21080014 #addi $8, $8, 20
0xad09fffc #sw $9, -4($8)
0xad09fff8 #sw $9, -8($8)
0xad080000 #sw $8, ($8)
0x8d0bfff4 #lw $11, -12($8)
0x8d6b0000 #lw $11, ($11)
0x1000ffff #end: beq $0, $0, end
```



Text Segment

Bkpt	Address	Code	Basic	
□	0x00000000	0x20082014	addi $8,$0,8212	1: addi $8, $0, 0x2014
□	0x00000004	0x20090004	addi $9,$0,4	2: addi $9, $0, 4
□	0x00000008	0xad090000	sw $9,0($8)	3: sw $9, ($8)
□	0x0000000c	0xad090004	sw $9,4($8)	4: sw $9, 4($8)
□	0x00000010	0xad080008	sw $8,8($8)	5: sw $8, 8($8)
□	0x00000014	0x21080014	addi $8,$8,20	6: addi $8, $8, 20
□	0x00000018	0xad09fffc	sw $9,-4($8)	7: sw $9, -4($8)
□	0x0000001c	0xad09fff8	sw $9,-8($8)	8: sw $9, -8($8)
□	0x00000020	0xad080000	sw $8,0($8)	9: sw $8, ($8)
□	0x00000024	0x8d0bfff4	lw $11,-12($8)	10: lw $11, -12($8)
□	0x00000028	0x8d6b0000	lw $11,0($11)	11: lw $11, ($11)
□	0x0000002c	0x1000ffff	beq $0,$0,-1	12: end: beq $0, $0, end

```
END PROGRAM
PC:  44

Instruction Count:  12
Register 0 : 0
Register 1 : 0
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 8232
Register 9 : 4
Register 10 : 0
Register 11 : 4
Register 12 : 0
Register 13 : 0
Register 14 : 0
Register 15 : 0
Register 16 : 0
Register 17 : 0
Register 18 : 0
Register 19 : 0
Register 20 : 0
Register 21 : 0
Register 22 : 0
Register 23 : 0

0x2000 : 0
0x2004 : 0
0x2008 : 0
0x200c : 0
0x2010 : 0
0x2014 : 4
0x2018 : 4
0x201c : 8212
0x2020 : 4
0x2024 : 4
0x2028 : 8232
0x202c : 0
0x2030 : 0
0x2034 : 0
0x2038 : 0
0x203c : 0
0x2040 : 0
0x2044 : 0
0x2048 : 0
0x204c : 0
0x2050 : 0
```

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+14)	Value (+18)	Value (+1c)
0x00002000	0	0	0	4	4	8212
0x00002020	4	4	8232	0	0	0
0x00002040	0	0	0	0	0	0
0x00002060	0	0	0	0	0	0
0x00002080	0	0	0	0	0	0
0x000020a0	0	0	0	0	0	0
0x000020c0	0	0	0	0	0	0
0x000020e0	0	0	0	0	0	0
0x00002100	0	0	0	0	0	0
0x00002120	0	0	0	0	0	0
0x00002140	0	0	0	0	0	0
0x00002160	0	0	0	0	0	0
0x00002180	0	0	0	0	0	0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	8232
$t1	9	4
$t2	10	0
$t3	11	4
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		44
hi		0
lo		0

Instruction Statistics, Version 1.0 (Ingo Kofler)

Total:	12	
ALU:	3	25%
Jump:	0	0%
Branch:	1	8%
Memory:	8	67%
Other:	0	

II) PRPG

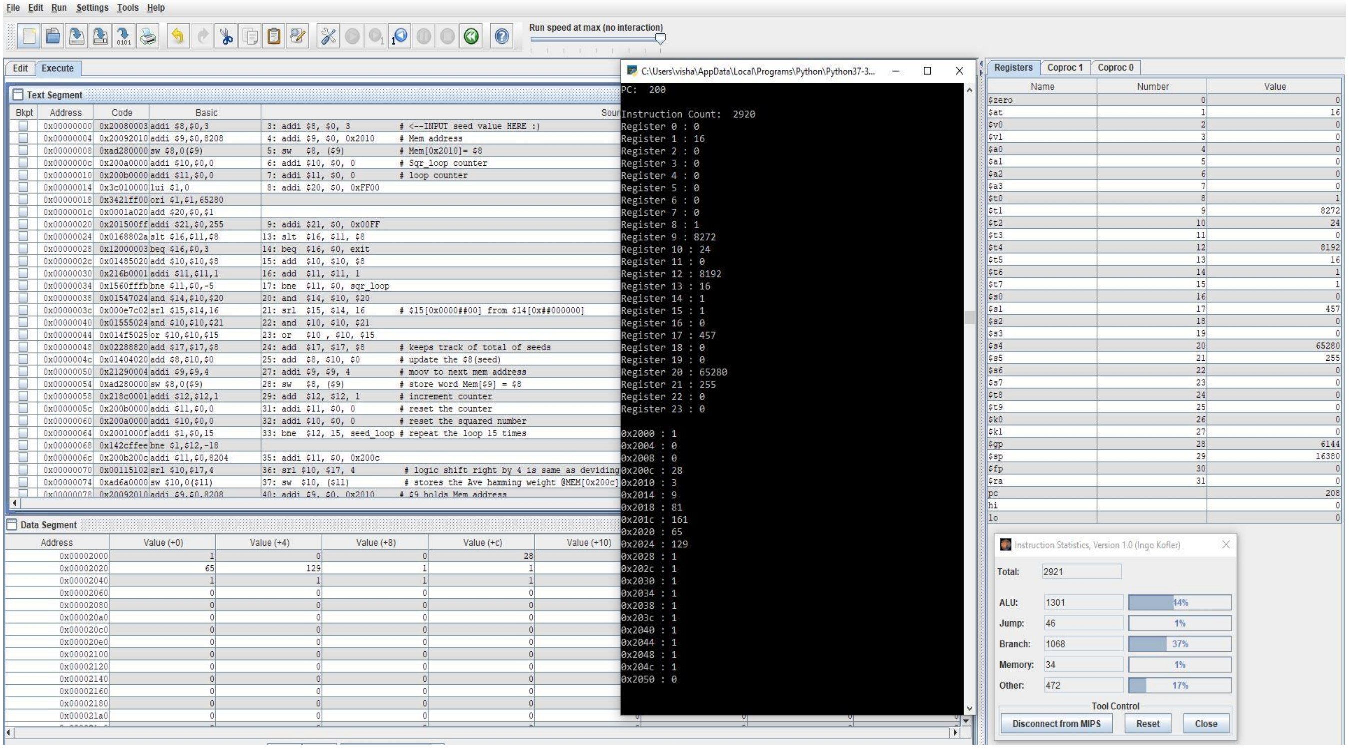
Q1: Which level does your project achieve? Which PRPG algorithm did your group choose? Why? (up to 5pts will be given to the group(s) which chose the rarest PRPG algorithm.)

Our project achieves all levels except the "special instruction". Our program can simulate all the required instructions in Python as well as a few others in order to allow our PRPG implementation to work properly. The PRPG algorithm we used to be the same algorithm from Project 1, squaring the number, dropping the middle 16 bits, and then combining the first and last 8 bits to create another 16-bit number. We chose this method mainly because it was already a working implementation and it allowed us to focus more on the unfinished parts of the Python Simulator.

Q2: How do you verify the correctness of your python simulator, and PRPG code? What kind of resources (productivity tools, collaboration tools) did your group use to work on this project? Give an example of a bug (either in MIPS or python) that your group found out.

To verify the correctness of our Python Simulator, we constantly referred to MARS with our specific test cases from our project 1 throughout the entirety of the project. Every time we would implement a new section of code for an instruction, we would create a small test case in assembly code to run it in MARS and then verify that we would get the same result from our Python Simulator. Our PRPG code was verified in a similar way. Since we used the original algorithm for our PRPG from Project 1, we already knew that it was fully functional. All we had to do was convert this code from assembly to hexadecimal and then use that in our Python Simulator to verify that the results were the same in both cases. A few collaboration tools our group used to work on the project included GitHub for our version control and WhatsApp for communication. One bug that we encountered during the project was dealing with some of the edge cases for the SLL instruction. In our previous implementations, we always ran into a problem with positive numbers not being shifted correctly because of how bin () works in Python. We did not consider that the leading 0's in the binary representation of a number would be left out when the length of the binary string was less than 32 bits. Because of this, the MSB in each binary string would always be a 1, something we obviously did not intend to happen because our code treated all these numbers as negative. To get around this issue, we simply always extended the binary string to 32-bits to allow our code to detect both positive and negative numbers correctly.
Also, while I was running the hex code of PRPG on python I encountered issue with large number for example mips tends to convert addi instruction with 34567 immediate into 3 different instructions also mips uses the regular binary bandwidth not the 2's complimented. So as our addi instruction was designed for negative and positive number But for some reason our code was treating 34567 number as 2's complimented negative number. To assure that I commented out 3 lines of code which was treating large number as 2's complemented signed negative number. I tested seed being 34567 and it produced correct output. In the end we all did very good job. It was fun experience working with Alex and vitaly.

Show the results and screenshots of your python simulator for your PRPG code.
S0 = 3


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit
Execute
Text Segment

Bkpt	Address	Code	Basic	Sour
□	0x00000000	0x20080003	addi $8,$0,3	3: addi $8, $0, 3 # <--INPUT seed value HERE :)
□	0x00000004	0x20092010	addi $9,$0,8208	4: addi $9, $0, 0x2010 # Mem address
□	0x00000008	0xad280000	sw $8,0($9)	5: sw $8, ($9) # Mem[0x2010]= $8
□	0x0000000c	0x200a0000	addi $10,$0,0	6: addi $10, $0, 0 # Sqr_loop counter
□	0x00000010	0x200b0000	addi $11,$0,0	7: addi $11, $0, 0 # loop counter
□	0x00000014	0x3c010000	lui $1,0	8: addi $20, $0, 0xFF00
□	0x00000018	0x3421ff00	ori $1,$1,65280	
□	0x0000001c	0x0001a020	add $20,$0,$1	
□	0x00000020	0x201500ff	addi $21,$0,255	9: addi $21, $0, 0x00FF
□	0x00000024	0x0168802a	slt $16,$11,$8	13: slt $16, $11, $8
□	0x00000028	0x12000003	beq $16,$0,3	14: beq $16, $0, exit
□	0x0000002c	0x01485020	add $10,$10,$8	15: add $10, $10, $8
□	0x00000030	0x216b0001	addi $11,$11,1	16: add $11, $11, 1
□	0x00000034	0x1560fffb	bne $11,$0,-5	17: bne $11, $0, sqr_loop
□	0x00000038	0x01547024	and $14,$10,$20	20: and $14, $10, $20
□	0x0000003c	0x000e7c02	srl $15,$14,16	21: srl $15, $14, 16 # $15[0x0000##00] from $14[0x##000000]
□	0x00000040	0x01555024	and $10,$10,$21	22: and $10, $10, $21
□	0x00000044	0x014f5025	or $10,$10,$15	23: or $10 , $10, $15
□	0x00000048	0x02288820	add $17,$17,$8	24: add $17, $17, $8 # keeps track of total of seeds
□	0x0000004c	0x01404020	add $8,$10,$0	25: add $8, $10, $0 # update the $8(seed)
□	0x00000050	0x21290004	addi $9,$9,4	27: addi $9, $9, 4 # moov to next mem address
□	0x00000054	0xad280000	sw $8,0($9)	28: sw $8, ($9) # store word Mem[$9] = $8
□	0x00000058	0x218c0001	addi $12,$12,1	29: add $12, $12, 1 # increment counter
□	0x0000005c	0x200b0000	addi $11,$0,0	31: addi $11, $0, 0 # reset the counter
□	0x00000060	0x200a0000	addi $10,$0,0	32: addi $10, $0, 0 # reset the squared number
□	0x00000064	0x2001000f	addi $1,$0,15	33: bne $12, 15, seed_loop # repeat the loop 15 times
□	0x00000068	0x142cffee	bne $1,$12,-18	
□	0x0000006c	0x200b200c	addi $11,$0,8204	35: addi $11, $0, 0x200c
□	0x00000070	0x00115102	srl $10,$17,4	36: srl $10, $17, 4 # logic shift right by 4 is same as deviding
□	0x00000074	0xad6a0000	sw $10,0($11)	37: sw $10, ($11) # stores the Ave hamming weight @MEM[0x200c]
□	0x00000078	0x20092010	addi $9,$0,8208	40: addi $9, $0, 0x2010 # $9 holds Mem address

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x00002000	1	0	0	28	
0x00002020	65	129	1	1	
0x00002040	1	1	1	1	
0x00002060	0	0	0	0	
0x00002080	0	0	0	0	
0x000020a0	0	0	0	0	
0x000020c0	0	0	0	0	
0x000020e0	0	0	0	0	
0x00002100	0	0	0	0	
0x00002120	0	0	0	0	
0x00002140	0	0	0	0	
0x00002160	0	0	0	0	
0x00002180	0	0	0	0	
0x000021a0	0	0	0	0	

```
PC:  200

Instruction Count:  2920
Register 0 : 0
Register 1 : 16
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 1
Register 9 : 8272
Register 10 : 24
Register 11 : 0
Register 12 : 8192
Register 13 : 16
Register 14 : 1
Register 15 : 1
Register 16 : 0
Register 17 : 457
Register 18 : 0
Register 19 : 0
Register 20 : 65280
Register 21 : 255
Register 22 : 0
Register 23 : 0

0x2000 : 1
0x2004 : 0
0x2008 : 0
0x200c : 28
0x2010 : 3
0x2014 : 9
0x2018 : 81
0x201c : 161
0x2020 : 65
0x2024 : 129
0x2028 : 1
0x202c : 1
0x2030 : 1
0x2034 : 1
0x2038 : 1
0x203c : 1
0x2040 : 1
0x2044 : 1
0x2048 : 1
0x204c : 1
0x2050 : 0
```

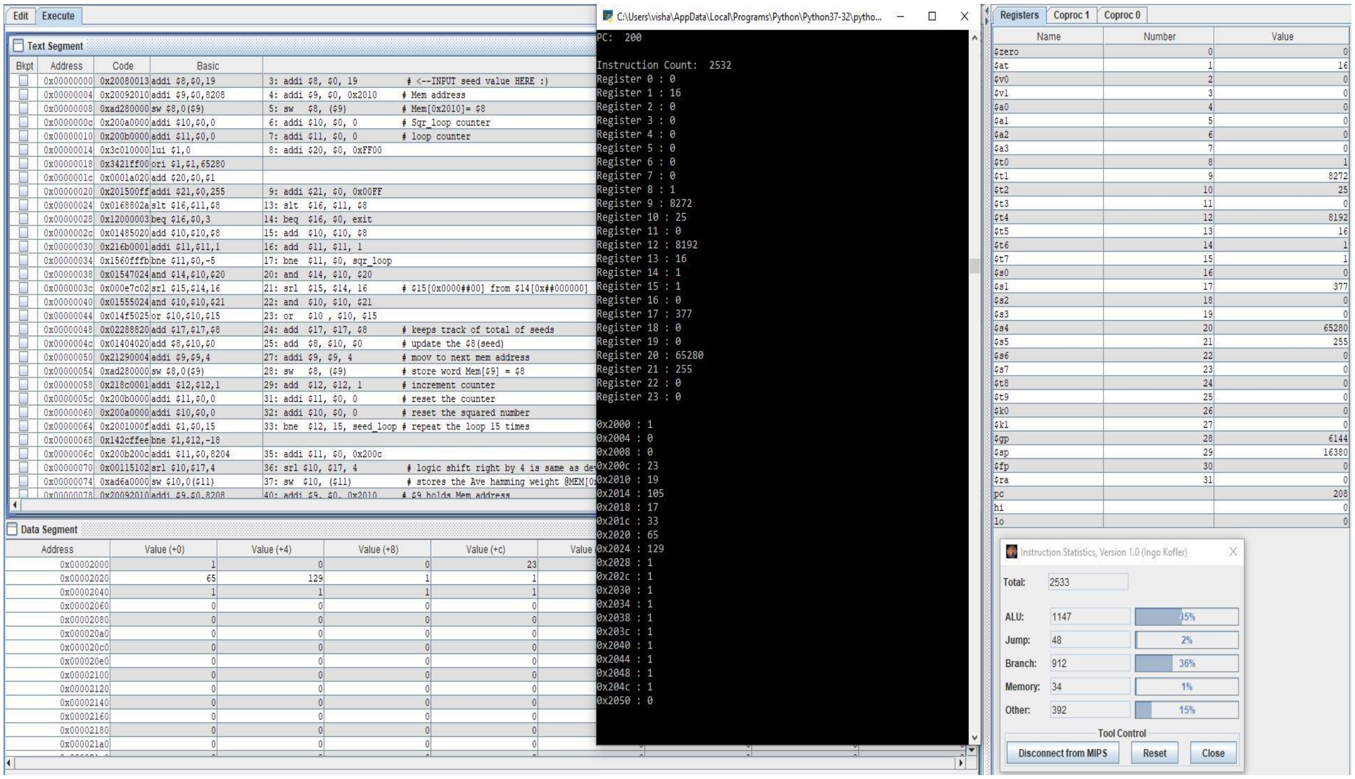
Name	Number	Value
$zero	0	0
$at	1	16
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	1
$t1	9	8272
$t2	10	24
$t3	11	0
$t4	12	8192
$t5	13	16
$t6	14	1
$t7	15	1
$s0	16	0
$s1	17	457
$s2	18	0
$s3	19	0
$s4	20	65280
$s5	21	255
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		208
hi		0
lo		0

Instruction Statistics, Version 1.0 (Ingo Kofler)

Total: 2921

ALU: 1301 44%
Jump: 46 1%
Branch: 1068 37%
Memory: 34 1%
Other: 472 17%

S0 = 19



Bkpt	Address	Code	Basic	
□	0x00000000	0x20080013	addi $8,$0,19	3: addi $8, $0, 19 # <--INPUT seed value HERE :)
□	0x00000004	0x20092010	addi $9,$0,8208	4: addi $9, $0, 0x2010 # Mem address
□	0x00000008	0xad280000	sw $8,0($9)	5: sw $8, ($9) # Mem[0x2010]= $8
□	0x0000000c	0x200a0000	addi $10,$0,0	6: addi $10, $0, 0 # Sqr_loop counter
□	0x00000010	0x200b0000	addi $11,$0,0	7: addi $11, $0, 0 # loop counter
□	0x00000014	0x3c010000	lui $1,0	8: addi $20, $0, 0xFF00
□	0x00000018	0x3421ff00	ori $1,$1,65280	
□	0x0000001c	0x0001a020	add $20,$0,$1	
□	0x00000020	0x201500ff	addi $21,$0,255	9: addi $21, $0, 0x00FF
□	0x00000024	0x0168802a	slt $16,$11,$8	13: slt $16, $11, $8
□	0x00000028	0x12000003	beq $16,$0,3	14: beq $16, $0, exit
□	0x0000002c	0x01485020	add $10,$10,$8	15: add $10, $10, $8
□	0x00000030	0x216b0001	addi $11,$11,1	16: add $11, $11, 1
□	0x00000034	0x1560fffb	bne $11,$0,-5	17: bne $11, $0, sqr_loop
□	0x00000038	0x01547024	and $14,$10,$20	20: and $14, $10, $20
□	0x0000003c	0x000e7c02	srl $15,$14,16	21: srl $15, $14, 16 # $15[0x0000##00] from $14[0x##000000]
□	0x00000040	0x01555024	and $10,$10,$21	22: and $10, $10, $21
□	0x00000044	0x014f5025	or $10,$10,$15	23: or $10 , $10, $15
□	0x00000048	0x02288820	add $17,$17,$8	24: add $17, $17, $8 # keeps track of total of seeds
□	0x0000004c	0x01404020	add $8,$10,$0	25: add $8, $10, $0 # update the $8(seed)
□	0x00000050	0x21290004	addi $9,$9,4	27: addi $9, $9, 4 # moov to next mem address
□	0x00000054	0xad280000	sw $8,0($9)	28: sw $8, ($9) # store word Mem[$9] = $8
□	0x00000058	0x218c0001	addi $12,$12,1	29: add $12, $12, 1 # increment counter
□	0x0000005c	0x200b0000	addi $11,$0,0	31: addi $11, $0, 0 # reset the counter
□	0x00000060	0x200a0000	addi $10,$0,0	32: addi $10, $0, 0 # reset the squared number
□	0x00000064	0x2001000f	addi $1,$0,15	33: bne $12, 15, seed_loop # repeat the loop 15 times
□	0x00000068	0x142cffee	bne $1,$12,-18	
□	0x0000006c	0x200b200c	addi $11,$0,8204	35: addi $11, $0, 0x200c
□	0x00000070	0x00115102	srl $10,$17,4	36: srl $10, $17, 4 # logic shift right by 4 is same as de
□	0x00000074	0xad6a0000	sw $10,0($11)	37: sw $10, ($11) # stores the Ave hamming weight @MEM[0x
□	0x00000078	0x20092010	addi $9,$0,8208	40: addi $9, $0, 0x2010 # $9 holds Mem address

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value
0x00002000	1	0	0	23	
0x00002020	65	129	1	1	
0x00002040	1	1	1	1	
0x00002060	0	0	0	0	
0x00002080	0	0	0	0	
0x000020a0	0	0	0	0	
0x000020c0	0	0	0	0	
0x000020e0	0	0	0	0	
0x00002100	0	0	0	0	
0x00002120	0	0	0	0	
0x00002140	0	0	0	0	
0x00002160	0	0	0	0	
0x00002180	0	0	0	0	
0x000021a0	0	0	0	0	

C:\Users\visha\AppData\Local\Programs\Python\Python37-32\pytho...

```
PC:  200

Instruction Count:  2532
Register 0 : 0
Register 1 : 16
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 1
Register 9 : 8272
Register 10 : 25
Register 11 : 0
Register 12 : 8192
Register 13 : 16
Register 14 : 1
Register 15 : 1
Register 16 : 0
Register 17 : 377
Register 18 : 0
Register 19 : 0
Register 20 : 65280
Register 21 : 255
Register 22 : 0
Register 23 : 0

0x2000 : 1
0x2004 : 0
0x2008 : 0
0x200c : 23
0x2010 : 19
0x2014 : 105
0x2018 : 17
0x201c : 33
0x2020 : 65
0x2024 : 129
0x2028 : 1
0x202c : 1
0x2030 : 1
0x2034 : 1
0x2038 : 1
0x203c : 1
0x2040 : 1
0x2044 : 1
0x2048 : 1
0x204c : 1
0x2050 : 0
```

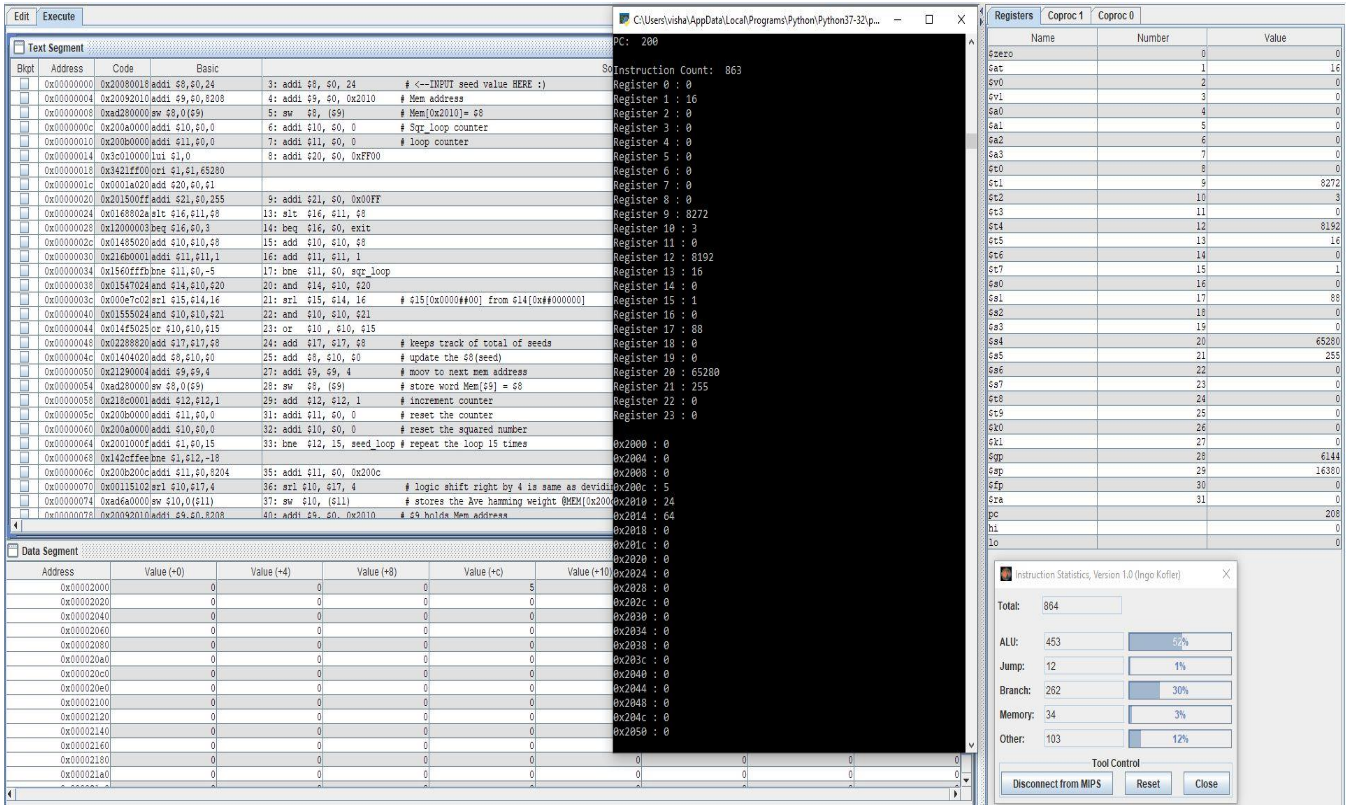
Name	Number	Value
$zero	0	0
$at	1	16
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	1
$t1	9	8272
$t2	10	25
$t3	11	0
$t4	12	8192
$t5	13	16
$t6	14	1
$t7	15	1
$s0	16	0
$s1	17	377
$s2	18	0
$s3	19	0
$s4	20	65280
$s5	21	255
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		208
hi		0
lo		0

Instruction Statistics, Version 1.0 (Ingo Kofler)

Total: 2533

ALU: 1147 45%
Jump: 48 2%
Branch: 912 36%
Memory: 34 1%
Other: 392 15%

S0 = 24



Bkpt	Address	Code	Basic	So
☐	0x00000000	0x20080018	addi $8,$0,24	3: addi $8, $0, 24 # <--INPUT seed value HERE :)
☐	0x00000004	0x20092010	addi $9,$0,8208	4: addi $9, $0, 0x2010 # Mem address
☐	0x00000008	0xad280000	sw $8,0($9)	5: sw $8, ($9) # Mem[0x2010]= $8
☐	0x0000000c	0x200a0000	addi $10,$0,0	6: addi $10, $0, 0 # Sqr_loop counter
☐	0x00000010	0x200b0000	addi $11,$0,0	7: addi $11, $0, 0 # loop counter
☐	0x00000014	0x3c010000	lui $1,0	8: addi $20, $0, 0xFF00
☐	0x00000018	0x3421ff00	ori $1,$1,65280	
☐	0x0000001c	0x0001a020	add $20,$0,$1	
☐	0x00000020	0x201500ff	addi $21,$0,255	9: addi $21, $0, 0x00FF
☐	0x00000024	0x0168802a	slt $16,$11,$8	13: slt $16, $11, $8
☐	0x00000028	0x12000003	beq $16,$0,3	14: beq $16, $0, exit
☐	0x0000002c	0x01485020	add $10,$10,$8	15: add $10, $10, $8
☐	0x00000030	0x216b0001	addi $11,$11,1	16: add $11, $11, 1
☐	0x00000034	0x1560fffb	bne $11,$0,-5	17: bne $11, $0, sqr_loop
☐	0x00000038	0x01547024	and $14,$10,$20	20: and $14, $10, $20
☐	0x0000003c	0x000e7c02	srl $15,$14,16	21: srl $15, $14, 16 # $15[0x0000##00] from $14[0x##000000]
☐	0x00000040	0x01555024	and $10,$10,$21	22: and $10, $10, $21
☐	0x00000044	0x014f5025	or $10,$10,$15	23: or $10 , $10, $15
☐	0x00000048	0x02288820	add $17,$17,$8	24: add $17, $17, $8 # keeps track of total of seeds
☐	0x0000004c	0x01404020	add $8,$10,$0	25: add $8, $10, $0 # update the $8(seed)
☐	0x00000050	0x21290004	addi $9,$9,4	27: addi $9, $9, 4 # moov to next mem address
☐	0x00000054	0xad280000	sw $8,0($9)	28: sw $8, ($9) # store word Mem[$9] = $8
☐	0x00000058	0x218c0001	addi $12,$12,1	29: add $12, $12, 1 # increment counter
☐	0x0000005c	0x200b0000	addi $11,$0,0	31: addi $11, $0, 0 # reset the counter
☐	0x00000060	0x200a0000	addi $10,$0,0	32: addi $10, $0, 0 # reset the squared number
☐	0x00000064	0x2001000f	addi $1,$0,15	33: bne $12, 15, seed_loop # repeat the loop 15 times
☐	0x00000068	0x142cffee	bne $1,$12,-18	
☐	0x0000006c	0x200b200c	addi $11,$0,8204	35: addi $11, $0, 0x200c
☐	0x00000070	0x00115102	srl $10,$17,4	36: srl $10, $17, 4 # logic shift right by 4 is same as devidi
☐	0x00000074	0xad6a0000	sw $10,0($11)	37: sw $10, ($11) # stores the Ave hamming weight @MEM[0x200
☐	0x00000078	0x20092010	addi $9,$0,8208	40: addi $9, $0, 0x2010 # $9 holds Mem address

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x00002000	0	0	0	5	
0x00002020	0	0	0	0	
0x00002040	0	0	0	0	
0x00002060	0	0	0	0	
0x00002080	0	0	0	0	
0x000020a0	0	0	0	0	
0x000020c0	0	0	0	0	
0x000020e0	0	0	0	0	
0x00002100	0	0	0	0	
0x00002120	0	0	0	0	
0x00002140	0	0	0	0	
0x00002160	0	0	0	0	
0x00002180	0	0	0	0	0
0x000021a0	0	0	0	0	0

C:\Users\visha\AppData\Local\Programs\Python\Python37-32\p...

```
PC:  200

Instruction Count:  863
Register 0 : 0
Register 1 : 16
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 0
Register 9 : 8272
Register 10 : 3
Register 11 : 0
Register 12 : 8192
Register 13 : 16
Register 14 : 0
Register 15 : 1
Register 16 : 0
Register 17 : 88
Register 18 : 0
Register 19 : 0
Register 20 : 65280
Register 21 : 255
Register 22 : 0
Register 23 : 0

0x2000 : 0
0x2004 : 0
0x2008 : 0
0x200c : 5
0x2010 : 24
0x2014 : 64
0x2018 : 0
0x201c : 0
0x2020 : 0
0x2024 : 0
0x2028 : 0
0x202c : 0
0x2030 : 0
0x2034 : 0
0x2038 : 0
0x203c : 0
0x2040 : 0
0x2044 : 0
0x2048 : 0
0x204c : 0
0x2050 : 0
```

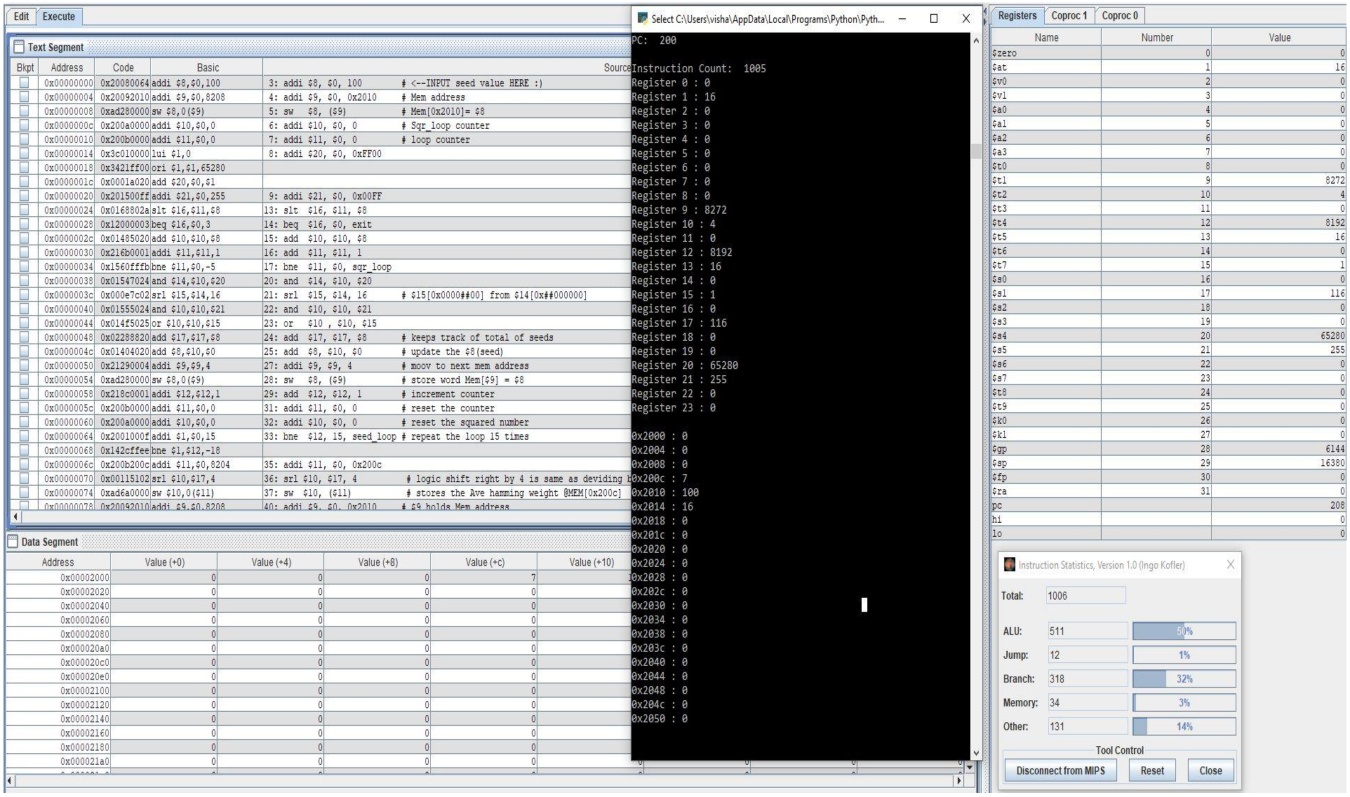
Name	Number	Value
$zero	0	0
$at	1	16
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	8272
$t2	10	3
$t3	11	0
$t4	12	8192
$t5	13	16
$t6	14	0
$t7	15	1
$s0	16	0
$s1	17	88
$s2	18	0
$s3	19	0
$s4	20	65280
$s5	21	255
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		208
hi		0
lo		0

Instruction Statistics, Version 1.0 (Ingo Kofler)

Total: 864

ALU:	453	52%
Jump:	12	1%
Branch:	262	30%
Memory:	34	3%
Other:	103	12%

S0 = 100



Bkpt	Address	Code	Basic	Source
☐	0x00000000	0x20080064	addi $8,$0,100	3: addi $8, $0, 100 # <--INPUT seed value HERE :)
☐	0x00000004	0x20092010	addi $9,$0,8208	4: addi $9, $0, 0x2010 # Mem address
☐	0x00000008	0xad280000	sw $8,0($9)	5: sw $8, ($9) # Mem[0x2010]= $8
☐	0x0000000c	0x200a0000	addi $10,$0,0	6: addi $10, $0, 0 # Sqr_loop counter
☐	0x00000010	0x200b0000	addi $11,$0,0	7: addi $11, $0, 0 # loop counter
☐	0x00000014	0x3c010000	lui $1,0	8: addi $20, $0, 0xFF00
☐	0x00000018	0x3421ff00	ori $1,$1,65280	
☐	0x0000001c	0x0001a020	add $20,$0,$1	
☐	0x00000020	0x201500ff	addi $21,$0,255	9: addi $21, $0, 0x00FF
☐	0x00000024	0x0168802a	slt $16,$11,$8	13: slt $16, $11, $8
☐	0x00000028	0x12000003	beq $16,$0,3	14: beq $16, $0, exit
☐	0x0000002c	0x01485020	add $10,$10,$8	15: add $10, $10, $8
☐	0x00000030	0x216b0001	addi $11,$11,1	16: add $11, $11, 1
☐	0x00000034	0x1560fffb	bne $11,$0,-5	17: bne $11, $0, sqr_loop
☐	0x00000038	0x01547024	and $14,$10,$20	20: and $14, $10, $20
☐	0x0000003c	0x000e7c02	srl $15,$14,16	21: srl $15, $14, 16 # $15[0x0000##00] from $14[0x##000000]
☐	0x00000040	0x01555024	and $10,$10,$21	22: and $10, $10, $21
☐	0x00000044	0x014f5025	or $10,$10,$15	23: or $10 , $10, $15
☐	0x00000048	0x02288820	add $17,$17,$8	24: add $17, $17, $8 # keeps track of total of seeds
☐	0x0000004c	0x01404020	add $8,$10,$0	25: add $8, $10, $0 # update the $8(seed)
☐	0x00000050	0x21290004	addi $9,$9,4	27: addi $9, $9, 4 # moov to next mem address
☐	0x00000054	0xad280000	sw $8,0($9)	28: sw $8, ($9) # store word Mem[$9] = $8
☐	0x00000058	0x218c0001	addi $12,$12,1	29: add $12, $12, 1 # increment counter
☐	0x0000005c	0x200b0000	addi $11,$0,0	31: addi $11, $0, 0 # reset the counter
☐	0x00000060	0x200a0000	addi $10,$0,0	32: addi $10, $0, 0 # reset the squared number
☐	0x00000064	0x2001000f	addi $1,$0,15	33: bne $12, 15, seed_loop # repeat the loop 15 times
☐	0x00000068	0x142cffee	bne $1,$12,-18	
☐	0x0000006c	0x200b200c	addi $11,$0,8204	35: addi $11, $0, 0x200c
☐	0x00000070	0x00115102	srl $10,$17,4	36: srl $10, $17, 4 # logic shift right by 4 is same as deviding b
☐	0x00000074	0xad6a0000	sw $10,0($11)	37: sw $10, ($11) # stores the Ave hamming weight @MEM[0x200c]
☐	0x00000078	0x20092010	addi $9,$0,8208	40: addi $9, $0, 0x2010 # $9 holds Mem address

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x00002000	0	0	0	7	
0x00002020	0	0	0	0	
0x00002040	0	0	0	0	
0x00002060	0	0	0	0	
0x00002080	0	0	0	0	
0x000020a0	0	0	0	0	
0x000020c0	0	0	0	0	
0x000020e0	0	0	0	0	
0x00002100	0	0	0	0	
0x00002120	0	0	0	0	
0x00002140	0	0	0	0	
0x00002160	0	0	0	0	
0x00002180	0	0	0	0	
0x000021a0	0	0	0	0	

Select C:\Users\visha\AppData\Local\Programs\Python\Pyth...

```
PC:  200

Instruction Count:  1005
Register 0 : 0
Register 1 : 16
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 0
Register 9 : 8272
Register 10 : 4
Register 11 : 0
Register 12 : 8192
Register 13 : 16
Register 14 : 0
Register 15 : 1
Register 16 : 0
Register 17 : 116
Register 18 : 0
Register 19 : 0
Register 20 : 65280
Register 21 : 255
Register 22 : 0
Register 23 : 0

0x2000 : 0
0x2004 : 0
0x2008 : 0
0x200c : 7
0x2010 : 100
0x2014 : 16
0x2018 : 0
0x201c : 0
0x2020 : 0
0x2024 : 0
0x2028 : 0
0x202c : 0
0x2030 : 0
0x2034 : 0
0x2038 : 0
0x203c : 0
0x2040 : 0
0x2044 : 0
0x2048 : 0
0x204c : 0
0x2050 : 0
```

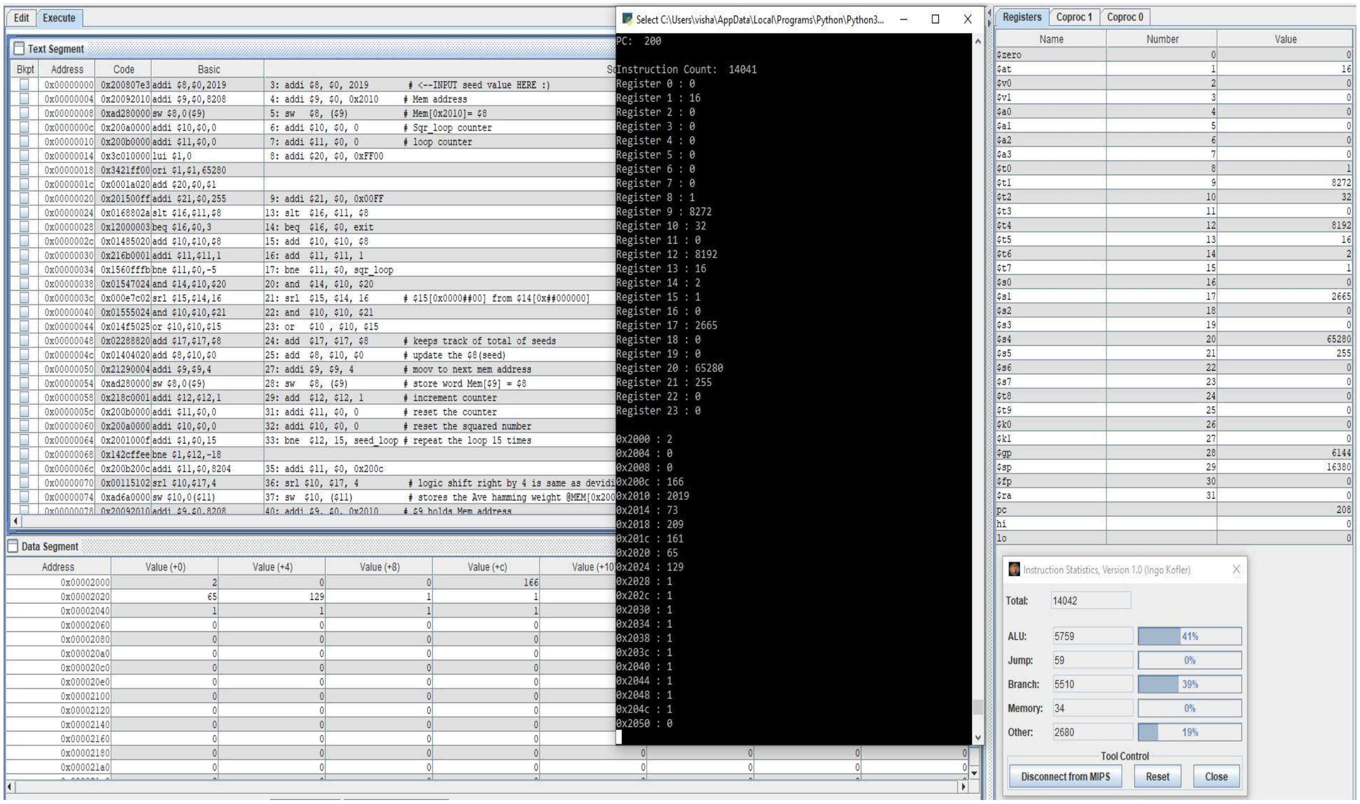
Name	Number	Value
$zero	0	0
$at	1	16
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	8272
$t2	10	4
$t3	11	0
$t4	12	8192
$t5	13	16
$t6	14	0
$t7	15	1
$s0	16	0
$s1	17	116
$s2	18	0
$s3	19	0
$s4	20	65280
$s5	21	255
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		208
hi		0
lo		0

Instruction Statistics, Version 1.0 (Ingo Kofler)

Total: 1006

ALU:	511	50%
Jump:	12	1%
Branch:	318	32%
Memory:	34	3%
Other:	131	14%

S0 = 2019



Bkpt	Address	Code	Basic	Source
☐	0x00000000	0x200807e3	addi $8,$0,2019	3: addi $8, $0, 2019 # <--INPUT seed value HERE :)
☐	0x00000004	0x20092010	addi $9,$0,8208	4: addi $9, $0, 0x2010 # Mem address
☐	0x00000008	0xad280000	sw $8,0($9)	5: sw $8, ($9) # Mem[0x2010]= $8
☐	0x0000000c	0x200a0000	addi $10,$0,0	6: addi $10, $0, 0 # Sqr_loop counter
☐	0x00000010	0x200b0000	addi $11,$0,0	7: addi $11, $0, 0 # loop counter
☐	0x00000014	0x3c010000	lui $1,0	8: addi $20, $0, 0xFF00
☐	0x00000018	0x3421ff00	ori $1,$1,65280	
☐	0x0000001c	0x0001a020	add $20,$0,$1	
☐	0x00000020	0x201500ff	addi $21,$0,255	9: addi $21, $0, 0x00FF
☐	0x00000024	0x0168802a	slt $16,$11,$8	13: slt $16, $11, $8
☐	0x00000028	0x12000003	beq $16,$0,3	14: beq $16, $0, exit
☐	0x0000002c	0x01485020	add $10,$10,$8	15: add $10, $10, $8
☐	0x00000030	0x216b0001	addi $11,$11,1	16: add $11, $11, 1
☐	0x00000034	0x1560fffb	bne $11,$0,-5	17: bne $11, $0, sqr_loop
☐	0x00000038	0x01547024	and $14,$10,$20	20: and $14, $10, $20
☐	0x0000003c	0x000e7c02	srl $15,$14,16	21: srl $15, $14, 16 # $15[0x0000##00] from $14[0x##000000]
☐	0x00000040	0x01555024	and $10,$10,$21	22: and $10, $10, $21
☐	0x00000044	0x014f5025	or $10,$10,$15	23: or $10 , $10, $15
☐	0x00000048	0x02288820	add $17,$17,$8	24: add $17, $17, $8 # keeps track of total of seeds
☐	0x0000004c	0x01404020	add $8,$10,$0	25: add $8, $10, $0 # update the $8(seed)
☐	0x00000050	0x21290004	addi $9,$9,4	27: addi $9, $9, 4 # moov to next mem address
☐	0x00000054	0xad280000	sw $8,0($9)	28: sw $8, ($9) # store word Mem[$9] = $8
☐	0x00000058	0x218c0001	addi $12,$12,1	29: add $12, $12, 1 # increment counter
☐	0x0000005c	0x200b0000	addi $11,$0,0	31: addi $11, $0, 0 # reset the counter
☐	0x00000060	0x200a0000	addi $10,$0,0	32: addi $10, $0, 0 # reset the squared number
☐	0x00000064	0x2001000f	addi $1,$0,15	33: bne $12, 15, seed_loop # repeat the loop 15 times
☐	0x00000068	0x142cffee	bne $1,$12,-18	
☐	0x0000006c	0x200b200c	addi $11,$0,8204	35: addi $11, $0, 0x200c
☐	0x00000070	0x00115102	srl $10,$17,4	36: srl $10, $17, 4 # logic shift right by 4 is same as devidi
☐	0x00000074	0xad6a0000	sw $10,0($11)	37: sw $10, ($11) # stores the Ave hamming weight @MEM[0x200
☐	0x00000078	0x20092010	addi $9,$0,8208	40: addi $9, $0, 0x2010 # $9 holds Mem address

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10			
0x00002000	2	0	0	166				
0x00002020	65	129	1	1				
0x00002040	1	1	1	1				
0x00002060	0	0	0	0				
0x00002080	0	0	0	0				
0x000020a0	0	0	0	0				
0x000020c0	0	0	0	0				
0x000020e0	0	0	0	0				
0x00002100	0	0	0	0				
0x00002120	0	0	0	0				
0x00002140	0	0	0	0				
0x00002160	0	0	0	0				
0x00002180	0	0	0	0	0	0	0	0
0x000021a0	0	0	0	0	0	0	0	0

Select C:\Users\visha\AppData\Local\Programs\Python\Python3...

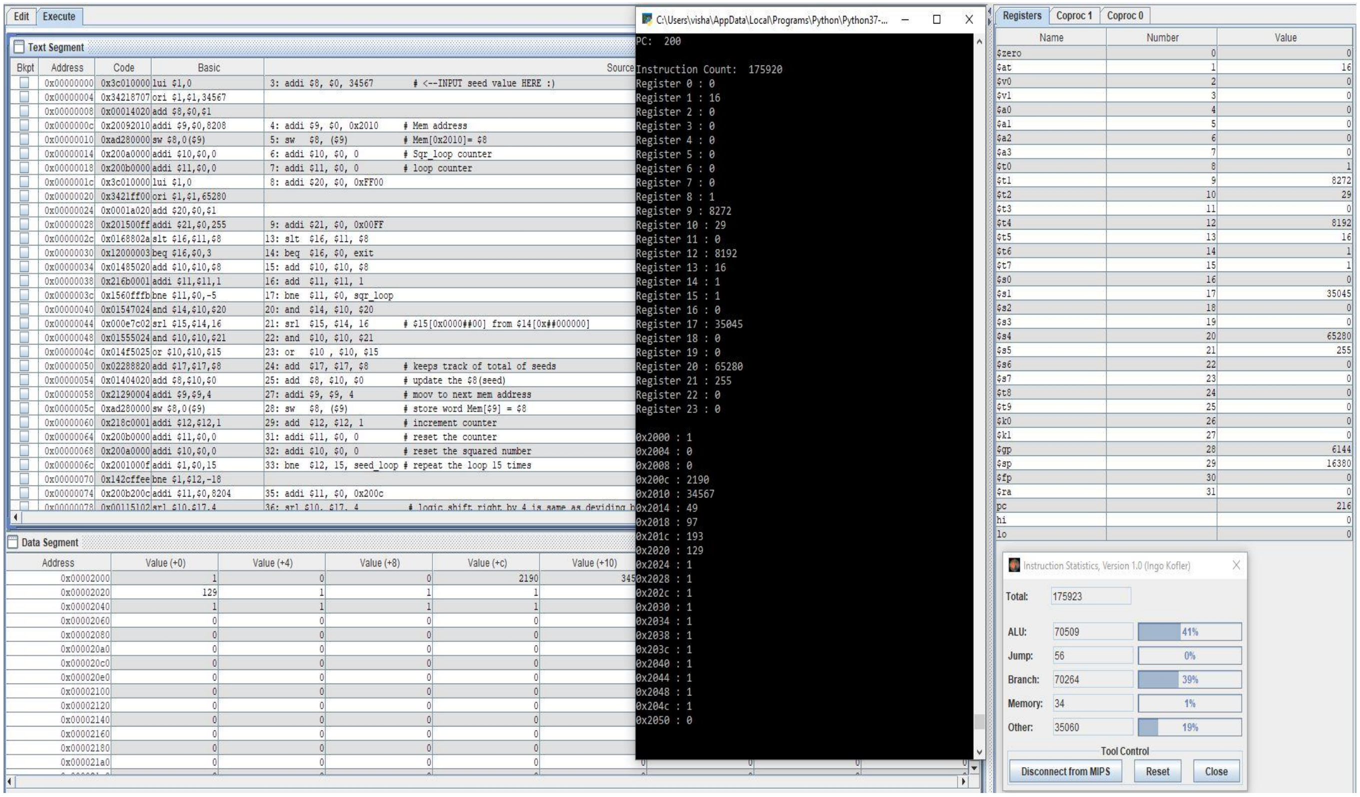
```
PC:  200

Instruction Count:  14041
Register 0 : 0
Register 1 : 16
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 1
Register 9 : 8272
Register 10 : 32
Register 11 : 0
Register 12 : 8192
Register 13 : 16
Register 14 : 2
Register 15 : 1
Register 16 : 0
Register 17 : 2665
Register 18 : 0
Register 19 : 0
Register 20 : 65280
Register 21 : 255
Register 22 : 0
Register 23 : 0

0x2000 : 2
0x2004 : 0
0x2008 : 0
0x200c : 166
0x2010 : 2019
0x2014 : 73
0x2018 : 209
0x201c : 161
0x2020 : 65
0x2024 : 129
0x2028 : 1
0x202c : 1
0x2030 : 1
0x2034 : 1
0x2038 : 1
0x203c : 1
0x2040 : 1
0x2044 : 1
0x2048 : 1
0x204c : 1
0x2050 : 0
```

Name	Number	Value
$zero	0	0
$at	1	16
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	1
$t1	9	8272
$t2	10	32
$t3	11	0
$t4	12	8192
$t5	13	16
$t6	14	2
$t7	15	1
$s0	16	0
$s1	17	2665
$s2	18	0
$s3	19	0
$s4	20	65280
$s5	21	255
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		208
hi		0
lo		0

Instruction Statistics, Version 1.0 (Ingo Kofler)

Total:	14042	
ALU:	5759	41%
Jump:	59	0%
Branch:	5510	39%
Memory:	34	0%
Other:	2680	19%

S0 = 34567



Bkpt	Address	Code	Basic	Source
☐	0x00000000	0x3c010000	lui $1,0	3: addi $8, $0, 34567 # <--INPUT seed value HERE :)
☐	0x00000004	0x34218707	ori $1,$1,34567	
☐	0x00000008	0x00014020	add $8,$0,$1	
☐	0x0000000c	0x20092010	addi $9,$0,8208	4: addi $9, $0, 0x2010 # Mem address
☐	0x00000010	0xad280000	sw $8,0($9)	5: sw $8, ($9) # Mem[0x2010]= $8
☐	0x00000014	0x200a0000	addi $10,$0,0	6: addi $10, $0, 0 # Sqr_loop counter
☐	0x00000018	0x200b0000	addi $11,$0,0	7: addi $11, $0, 0 # loop counter
☐	0x0000001c	0x3c010000	lui $1,0	8: addi $20, $0, 0xFF00
☐	0x00000020	0x3421ff00	ori $1,$1,65280	
☐	0x00000024	0x0001a020	add $20,$0,$1	
☐	0x00000028	0x201500ff	addi $21,$0,255	9: addi $21, $0, 0x00FF
☐	0x0000002c	0x0168802a	slt $16,$11,$8	13: slt $16, $11, $8
☐	0x00000030	0x12000003	beq $16,$0,3	14: beq $16, $0, exit
☐	0x00000034	0x01485020	add $10,$10,$8	15: add $10, $10, $8
☐	0x00000038	0x216b0001	addi $11,$11,1	16: add $11, $11, 1
☐	0x0000003c	0x1560fffb	bne $11,$0,-5	17: bne $11, $0, sqr_loop
☐	0x00000040	0x01547024	and $14,$10,$20	20: and $14, $10, $20
☐	0x00000044	0x000e7c02	srl $15,$14,16	21: srl $15, $14, 16 # $15[0x0000##00] from $14[0x##000000]
☐	0x00000048	0x01555024	and $10,$10,$21	22: and $10, $10, $21
☐	0x0000004c	0x014f5025	or $10,$10,$15	23: or $10 , $10, $15
☐	0x00000050	0x02288820	add $17,$17,$8	24: add $17, $17, $8 # keeps track of total of seeds
☐	0x00000054	0x01404020	add $8,$10,$0	25: add $8, $10, $0 # update the $8(seed)
☐	0x00000058	0x21290004	addi $9,$9,4	27: addi $9, $9, 4 # moov to next mem address
☐	0x0000005c	0xad280000	sw $8,0($9)	28: sw $8, ($9) # store word Mem[$9] = $8
☐	0x00000060	0x218c0001	addi $12,$12,1	29: add $12, $12, 1 # increment counter
☐	0x00000064	0x200b0000	addi $11,$0,0	31: addi $11, $0, 0 # reset the counter
☐	0x00000068	0x200a0000	addi $10,$0,0	32: addi $10, $0, 0 # reset the squared number
☐	0x0000006c	0x2001000f	addi $1,$0,15	33: bne $12, 15, seed_loop # repeat the loop 15 times
☐	0x00000070	0x142cffee	bne $1,$12,-18	
☐	0x00000074	0x200b200c	addi $11,$0,8204	35: addi $11, $0, 0x200c
☐	0x00000078	0x00115102	srl $10,$17,4	36: srl $10, $17, 4 # logic shift right by 4 is same as dividing b

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x00002000	1	0	0	2190	345
0x00002020	129	1	1	1	
0x00002040	1	1	1	1	
0x00002060	0	0	0	0	
0x00002080	0	0	0	0	
0x000020a0	0	0	0	0	
0x000020c0	0	0	0	0	
0x000020e0	0	0	0	0	
0x00002100	0	0	0	0	
0x00002120	0	0	0	0	
0x00002140	0	0	0	0	
0x00002160	0	0	0	0	
0x00002180	0	0	0	0	
0x000021a0	0	0	0	0	

C:\Users\visha\AppData\Local\Programs\Python\Python37-...

```
PC:  200

Instruction Count:  175920
Register 0 : 0
Register 1 : 16
Register 2 : 0
Register 3 : 0
Register 4 : 0
Register 5 : 0
Register 6 : 0
Register 7 : 0
Register 8 : 1
Register 9 : 8272
Register 10 : 29
Register 11 : 0
Register 12 : 8192
Register 13 : 16
Register 14 : 1
Register 15 : 1
Register 16 : 0
Register 17 : 35045
Register 18 : 0
Register 19 : 0
Register 20 : 65280
Register 21 : 255
Register 22 : 0
Register 23 : 0

0x2000 : 1
0x2004 : 0
0x2008 : 0
0x200c : 2190
0x2010 : 34567
0x2014 : 49
0x2018 : 97
0x201c : 193
0x2020 : 129
0x2024 : 1
0x2028 : 1
0x202c : 1
0x2030 : 1
0x2034 : 1
0x2038 : 1
0x203c : 1
0x2040 : 1
0x2044 : 1
0x2048 : 1
0x204c : 1
0x2050 : 0
```

Name	Number	Value
$zero	0	0
$at	1	16
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	1
$t1	9	8272
$t2	10	29
$t3	11	0
$t4	12	8192
$t5	13	16
$t6	14	1
$t7	15	1
$s0	16	0
$s1	17	35045
$s2	18	0
$s3	19	0
$s4	20	65280
$s5	21	255
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	16380
$fp	30	0
$ra	31	0
pc		216
hi		0
lo		0

Instruction Statistics, Version 1.0 (Ingo Kofler)

Total:	175923	
ALU:	70509	41%
Jump:	56	0%
Branch:	70264	39%
Memory:	34	1%
Other:	35060	19%

Appendix

Link to all files:

https://drive.google.com/open?id=1UNxpsjGiUnXR3a7GlyPbui5rhCf-JpTB