eXpOS Report

Stage 2:

Question 1:
When a file is created entries are made in the Inode table as well as the Root file. What is the need for this duplication
Solution:
Inode table is a data structure which is accessible only in Kernel mode, whereas Root file is accessible both in Kernel and User mode. This enables the user to search for a file from an application program itself by reading the Root file.

Assignment 1:

Copy the contents of Root File (from Block 5 of XFS disk) to a UNIX file $HOME/myexpos/root_file.txt and verify that an entry for sample.dat is made in it also.

Solution:

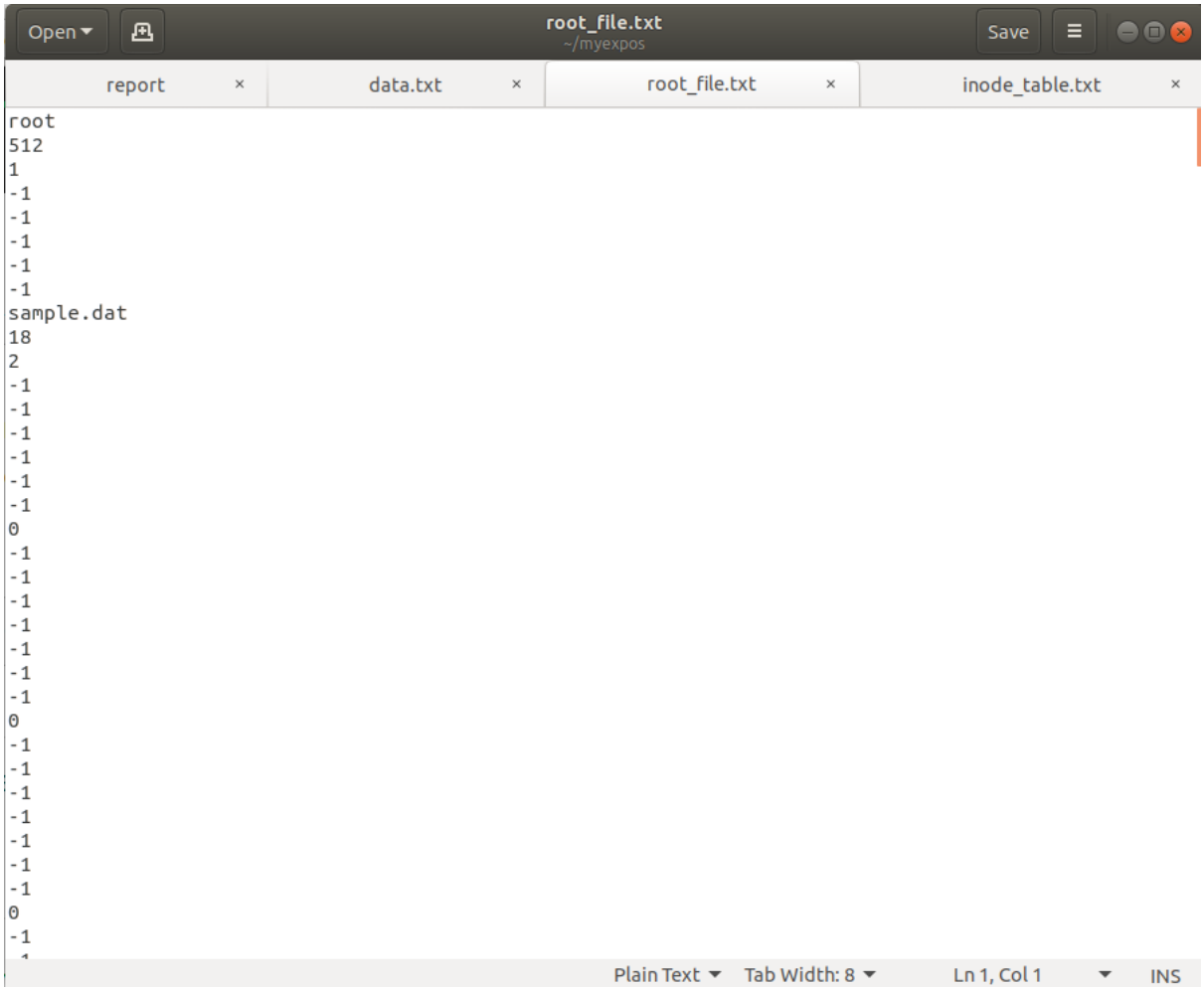
First we need to open the xfs interface using the following command in the terminal with home directory being

```
./xfs-interface
```

Using the Copy command in xfs interface as follows

```
copy 5 5 $HOME/myexpos/root_file.txt
```

Now the data in root_file.txt is changed as follows:




```
root
512
1
-1
-1
-1
-1
-1
sample.dat
18
2
-1
-1
-1
-1
-1
-1
0
-1
-1
-1
-1
-1
-1
-1
0
-1
-1
-1
-1
-1
-1
-1
0
-1
```

Assignment 2:

Delete the sample.dat from the XSM machine using xfs-interface and note the changes for the entries for this file in *inode table, root file and disk free list*.

Solution:

First let's remove the sample.dat using the following command

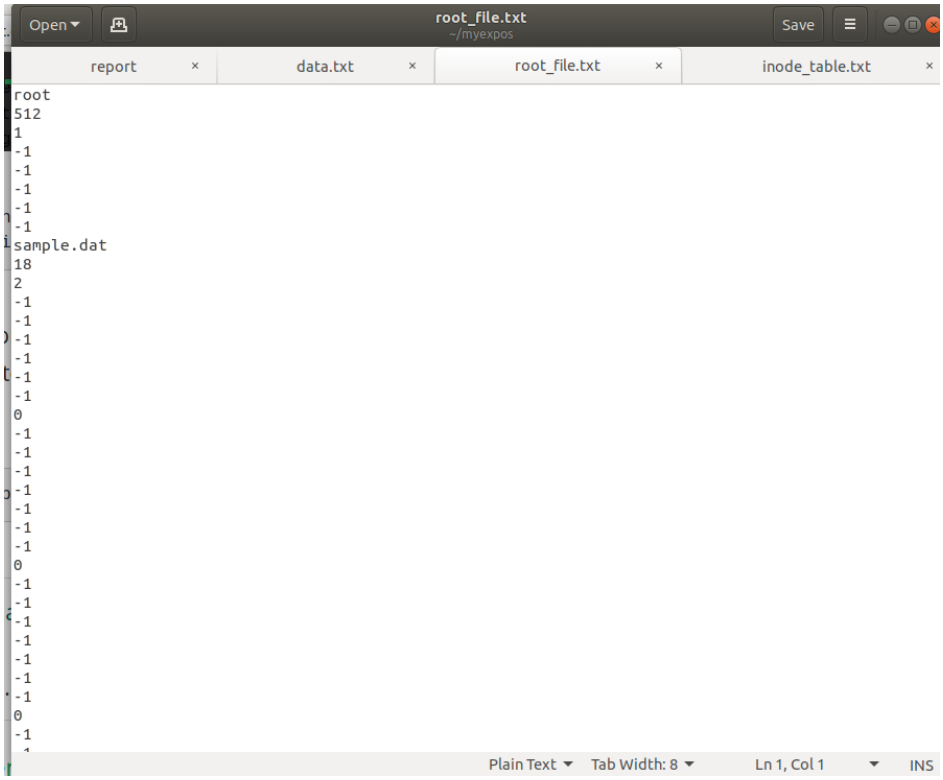
```
rm sample.dat
```

To see the changes in root file and inode table, we need to use these commands in the xfs-interface:

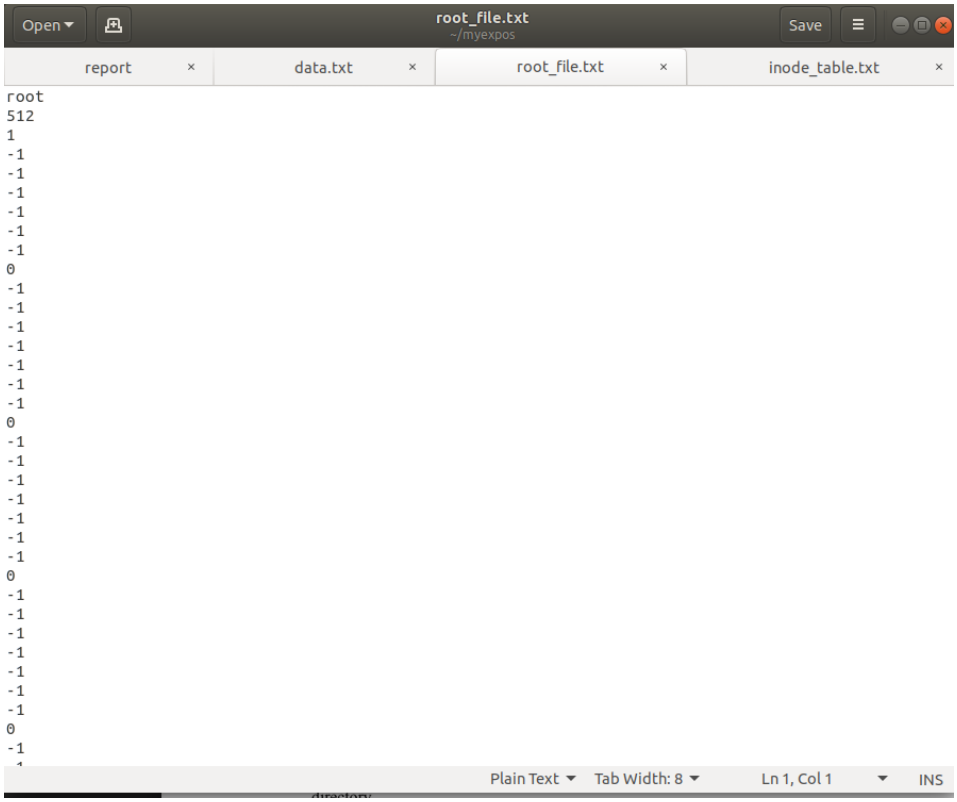
```
copy 3 4 $HOME/myexpos/inode_table.txt
copy 5 5 $HOME/myexpos/root_file.txt
```

Changes in root_file.txt are as follows

Before:

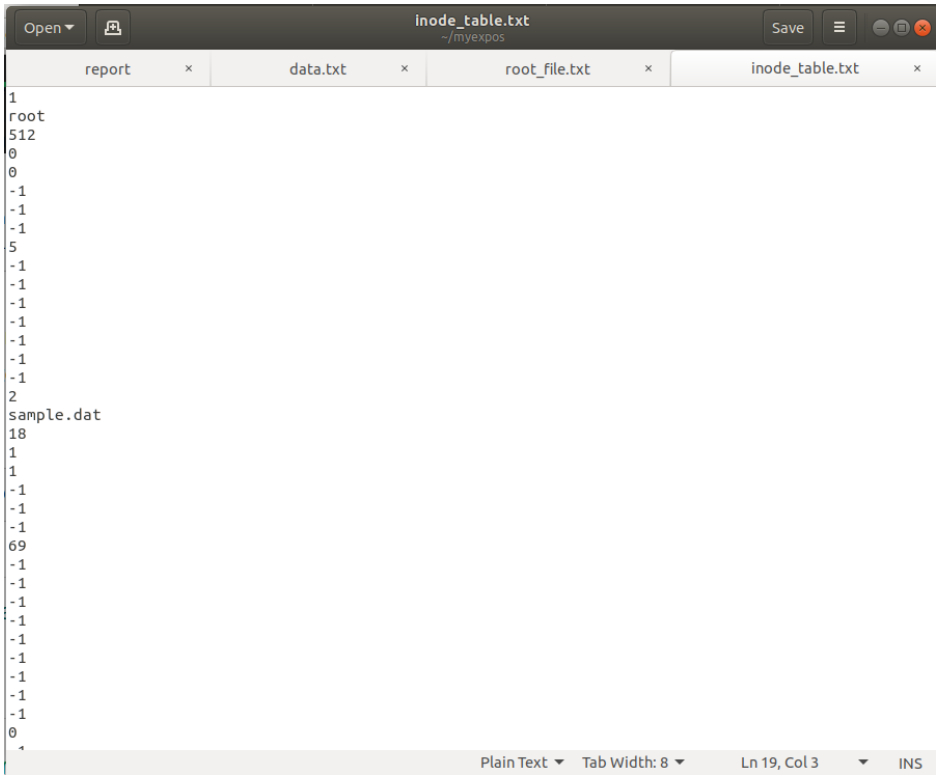


After:



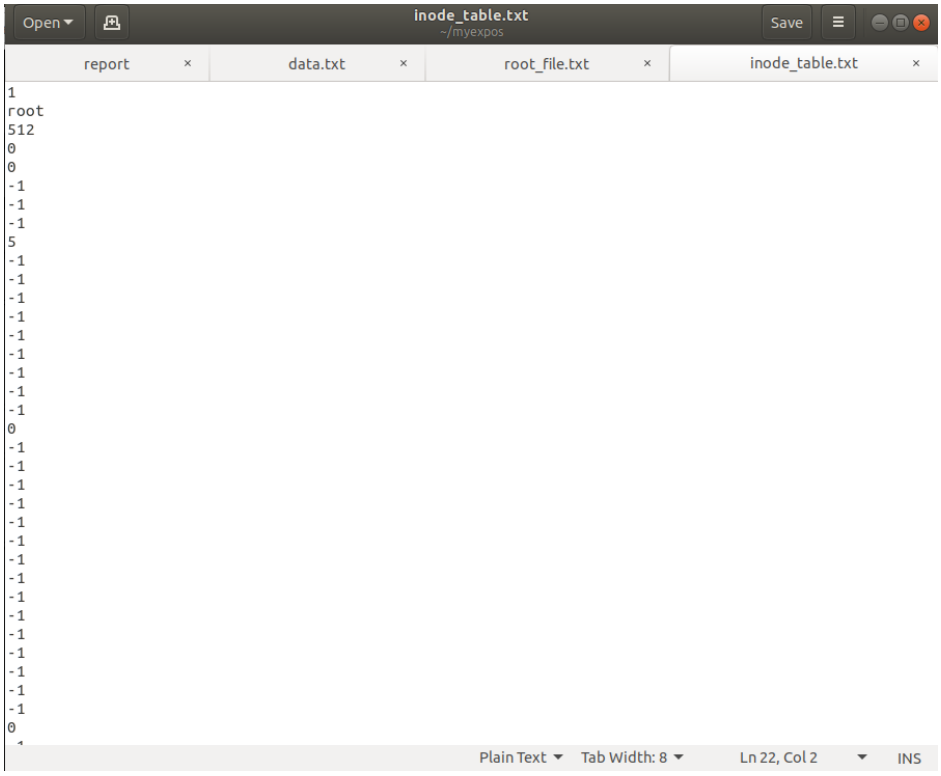
Changes in inode_table.txt are as follows

Before:




```
1
root
512
0
0
-1
-1
-1
5
-1
-1
-1
-1
-1
-1
-1
2
sample.dat
18
1
1
-1
-1
-1
69
-1
-1
-1
-1
-1
-1
-1
-1
-1
0
```

After:




```
1
root
512
0
0
-1
-1
-1
5
-1
-1
-1
-1
-1
-1
-1
-1
-1
0
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
0
```

Changes in disk free list are as follows:

Before:

67	-	1
68	-	1
69	-	1
70	-	0
71	-	0
72	-	0

After:

67	-	1
68	-	1
69	-	0
70	-	0
71	-	0
72	-	0

We can observe that the 69th block changed to '0'.

Stage 3:

Question 1:

Q1. If the OS Startup Code is loaded to some other page other than Page 1, will XSM work fine?

Solution:

No. This is because after the execution of the ROM Code, IP points to **512** which is the 1st instruction of Page 1. So, if the OS Startup Code is not loaded to Page 1, it results in an exception and leads to system crash.

Assignment 1:

Write an assembly program to print numbers from 1 to 20 and run it as the OS Startup code.

Solution:

Write and Save the program print_twenty_num.xsm as follows in spl/spl_progs:

Open ▾ print_twenty_num.xsm ~/myexpos/spl/spl_progs Save

report × print_twenty_num.xsm ×

```
MOV R0,0
MOV R1,20
L1:
INR R0
PORT P1,R0
OUT
MOV R2,R0
LT R2,R1
JNZ R2,L1
HALT
```

Then use the following command in xfs interface as follows:

```
# load --os $HOME/myexpos/spl/spl_progs/print_twenty_num.xsm
# exit
```

Now for the program to be executed go to xsm folder in terminal and use the following command

```
./xsm
```

To get the following results

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Machine is halting.
```

Stage 4:

Assignment 1:
Write the spl program to print sum of squares of the first 20 natural numbers. Load it using xfs interface and run the in the machine.
Solution:
Write and Save the program print_twenty_squares.spl as follows in spl/spl_progs:

print_twenty_squares.spl
~/myexpos/spl/spl_progs

```
alias sum R0;
alias count R1;
sum = 0;
count = 1;
while(count <= 20) do
  sum = sum + count * count;
  count = count + 1;
endwhile;
print sum;
```

Compile the following program with spl compiler using the following command

./spl spl_progs/print_twenty_squares.spl

This has generated a new file in spl_progs as print_twenty_squares.xsm as:

print_twenty_squares.xsm
~/myexpos/spl/spl_progs

report | print_twenty_squares.spl | print_twenty_squares.xsm

```
MOV R0, 0
MOV R1, 1
_L1:
MOV R16, 20
GE R16, R1
JZ R16, _L2
MOV R16, R1
MUL R16, R1
ADD R16, R0
MOV R0, R16
MOV R16, R1
ADD R16, 1
MOV R1, R16
JMP _L1
_L2:
MOV R16, R0
PORT P1, R16
OUT
HALT
```

Loading file "/home/guna/myexpos/spl/spl_progs/print_twen... Plain Text Tab Width: 8 Ln 1, Col 1 INS

Now load the load the program in xfs interface like we did before and exit and change directory to xsm and run ./xsm to get the following result.

```
2870
Machine is halting.
```

Stage 6:

Assignment 1:

Change virtual memory model such that code occupies logical pages 4 and 5 and the stack lies in logical page 8. You will have to modify the user program as well as the os startup code.

Solution:

User Code:

squares.xsm
~/myexpos/expl/expl_progs

report | print_twenty_squares.spl | print_twenty_squares.xsm | squares.xsm

```
MOV R0, 1
MOV R2, 5
GE R2, R0
JZ R2, 18
MOV R1, R0
MUL R1, R0
BRKP
ADD R0, 1
JMP 2
INT 10
```

Save this code in eps/eps_progs as squares.xsm and load it in xfs interface as follows

```
load --init $HOME/myexpos/expl/expl_progs/squares.xsm
```

Create another file for halt with just one line “halt;” and save it as halt.spl in spl_progs and then compile it with spl compiler and load it as follows

```
load --int=10 $HOME/myexpos/spl/spl_progs/halt.xsm
load --exhandler $HOME/myexpos/spl/spl_progs/halt.xsm
```

Create an os startup file as follows

os_startup.spl
~/myexpos/spl/spl_progs

```
loadi(4,7);
loadi(5,8);
loadi(22,35);
loadi(23,36);
loadi(2, 15);
loadi(3, 16);
PTBR = PAGE_TABLE_BASE;
PTLR = 3;
[PTBR+0] = 4;
[PTBR+1] = "0100";
[PTBR+2] = 5;
[PTBR+3] = "0100";
[PTBR+4] = 8;
[PTBR+5] = "0110";
[8*512] = 0;
SP = 2*512;
ireturn;
```

Save it as os_startup.spl and then compile it spl compiler as discussed before and load in it xfs as follows

```
load --os $HOME/myexpos/spl/spl_progs/osstartup.xsm
```

Exit the interface and then use the follow command to run the program

```
./xsm --debug --timer 0
```

At each step of entering reg observe the value in register R1 at each breakpoint

```
Previous instruction at IP = 12: BRKP
Mode: USER       PID: 0
Next instruction at IP = 14, Page No. = 0: ADD R0,1
debug> reg
R0: 1   R1: 1   R2: 1   R3:     R4:
R5:     R6:     R7:     R8:     R9:
R10:    R11:    R12:    R13:    R14:
R15:    R16: 1024       R17:    R18:    R19:
P0:     P1:     P2:     P3:
BP:     SP: 1023        IP: 14  PTBR: 29696     PTLR: 3
EIP:    EC:     EPN:    EMA:
debug> c
Previous instruction at IP = 12: BRKP
Mode: USER       PID: 0
Next instruction at IP = 14, Page No. = 0: ADD R0,1
debug> reg
R0: 2   R1: 4   R2: 1   R3:     R4:
R5:     R6:     R7:     R8:     R9:
R10:    R11:    R12:    R13:    R14:
R15:    R16: 1024       R17:    R18:    R19:
P0:     P1:     P2:     P3:
BP:     SP: 1023        IP: 14  PTBR: 29696     PTLR: 3
EIP:    EC:     EPN:    EMA:
debug> c
Previous instruction at IP = 12: BRKP
Mode: USER       PID: 0
Next instruction at IP = 14, Page No. = 0: ADD R0,1
debug> reg
R0: 3   R1: 9   R2: 1   R3:     R4:
R5:     R6:     R7:     R8:     R9:
R10:    R11:    R12:    R13:    R14:
R15:    R16: 1024       R17:    R18:    R19:
P0:     P1:     P2:     P3:
BP:     SP: 1023        IP: 14  PTBR: 29696     PTLR: 3
EIP:    EC:     EPN:    EMA:
debug> c
Previous instruction at IP = 12: BRKP
Mode: USER       PID: 0
Next instruction at IP = 14, Page No. = 0: ADD R0,1
debug> reg
R0: 4   R1: 16  R2: 1   R3:     R4:
R5:     R6:     R7:     R8:     R9:
R10:    R11:    R12:    R13:    R14:
R15:    R16: 1024       R17:    R18:    R19:
P0:     P1:     P2:     P3:
BP:     SP: 1023        IP: 14  PTBR: 29696     PTLR: 3
EIP:    EC:     EPN:    EMA:
debug> c
Previous instruction at IP = 12: BRKP
Mode: USER       PID: 0
Next instruction at IP = 14, Page No. = 0: ADD R0,1
debug> reg
R0: 5   R1: 25  R2: 1   R3:     R4:
R5:     R6:     R7:     R8:     R9:
R10:    R11:    R12:    R13:    R14:
```

```
R15:    R16: 1024       R17:    R18:    R19:
P0:     P1:     P2:     P3:
BP:     SP: 1023        IP: 14  PTBR: 29696     PTLR: 3
EIP:    EC:     EPN:    EMA:
debug> c
Machine is halting.
```

Stage 7:

Assignment 1:

Change the user program to compute cubes of the first five numbers.

Solution:

Create a new file with name cubes.xsm and type the following code in it

```
0
2056
0
0
0
0
0
0
MOV R0, 1
MOV R2, 5
GE R2, R0
JZ R2, 2076
MOV R1, R0
MUL R1, R0
MUL R1, R0
BRKP
ADD R0, 1
JMP 2058
INT 10
```

Save this code in eps/eps_progs as cube.xsm and load it in xfs interface as follows

load --init $HOME/myexpos/expl/expl_progs/cubes.xsm

Now let's create an another os startup file with name os_startup_cubes.spl and save it in spl_progs.

os_startup_cubes.spl
~/myexpos/spl/spl_progs

cubes.xsm | os_startup_cubes.spl

```
loadi(63,13);
loadi(64,14);
loadi(65,7);
loadi(66,8);
loadi(22,35);
loadi(23,36);
loadi(2, 15);
loadi(3, 16);

PTBR = PAGE_TABLE_BASE;
PTLR = 10;

[PTBR+0] = 63;
[PTBR+1] = "0100";
[PTBR+2] = 64;
[PTBR+3] = "0100";
[PTBR+4] = 78;
[PTBR+5] = "0110";
[PTBR+6] = 79;
[PTBR+7] = "0110";
[PTBR+8] = 65;
[PTBR+9] = "0100";
[PTBR+10] = 66;
[PTBR+11] = "0100";
[PTBR+12] = -1;
[PTBR+13] = "0000";
[PTBR+14] = -1;
[PTBR+15] = "0000";
[PTBR+16] = 76;
[PTBR+17] = "0110";
[PTBR+18] = 77;
[PTBR+19] = "0110";

SP = 8*512;
[76*512] = [65*512 + 1];

ireturn;
```

Compile it with spl compiler just like in stage 6 and load it in xfs interface and exit and run the program by following command

```
./xsm --debug --timer 0
```

At each step of entering reg observe the value in register R1 at each breakpoint

```
Previous instruction at IP = 2070: BRKP
Mode: USER       PID: 0
Next instruction at IP = 2072, Page No. = 4: ADD R0,1
debug> reg
R0: 1   R1: 1   R2: 1   R3:     R4:
R5:     R6:     R7:     R8:     R9:
R10:    R11:    R12:    R13:    R14:
R15:    R16: 38912      R17: 2056       R18:    R19:
P0:     P1:     P2:     P3:
BP:     SP: 4095        IP: 2072        PTBR: 29696     PTLR: 10
EIP:    EC:     EPN:    EMA:
debug> c
Previous instruction at IP = 2070: BRKP
Mode: USER       PID: 0
Next instruction at IP = 2072, Page No. = 4: ADD R0,1
debug> reg
R0: 2   R1: 8   R2: 1   R3:     R4:
R5:     R6:     R7:     R8:     R9:
R10:    R11:    R12:    R13:    R14:
R15:    R16: 38912      R17: 2056       R18:    R19:
P0:     P1:     P2:     P3:
BP:     SP: 4095        IP: 2072        PTBR: 29696     PTLR: 10
EIP:    EC:     EPN:    EMA:
debug> c
Previous instruction at IP = 2070: BRKP
Mode: USER       PID: 0
Next instruction at IP = 2072, Page No. = 4: ADD R0,1
debug> reg
R0: 3   R1: 27  R2: 1   R3:     R4:
R5:     R6:     R7:     R8:     R9:
R10:    R11:    R12:    R13:    R14:
R15:    R16: 38912      R17: 2056       R18:    R19:
P0:     P1:     P2:     P3:
BP:     SP: 4095        IP: 2072        PTBR: 29696     PTLR: 10
EIP:    EC:     EPN:    EMA:
debug> c
Previous instruction at IP = 2070: BRKP
Mode: USER       PID: 0
Next instruction at IP = 2072, Page No. = 4: ADD R0,1
debug> reg
R0: 4   R1: 64  R2: 1   R3:     R4:
R5:     R6:     R7:     R8:     R9:
R10:    R11:    R12:    R13:    R14:
R15:    R16: 38912      R17: 2056       R18:    R19:
P0:     P1:     P2:     P3:
BP:     SP: 4095        IP: 2072        PTBR: 29696     PTLR: 10
EIP:    EC:     EPN:    EMA:
debug> c
Previous instruction at IP = 2070: BRKP
Mode: USER       PID: 0
Next instruction at IP = 2072, Page No. = 4: ADD R0,1
debug> reg
R0: 5   R1: 125 R2: 1   R3:     R4:
R5:     R6:     R7:     R8:     R9:
R10:    R11:    R12:    R13:    R14:
```

```
R15:    R16: 38912      R17: 2056       R18:    R19:
P0:     P1:     P2:     P3:
BP:     SP: 4095        IP: 2072        PTBR: 29696     PTLR: 10
EIP:    EC:     EPN:    EMA:
debug> c
Machine is halting.
```