

Question 1)

This table represents the total number of micro and macro instructions count. When we look at these numbers, there is not that much difference and the difference between iterative and recursive method caused by the extra instructions in recursive method. As we know, there are more instructions in recursive method rather than iterative method due to in recursive method ,there are store and load operations.

Code part:

Actually due to connection between first and second question, I used the same way for calculating the number of instruction. Whenever an instruction arrives, I checked that if it is a branch or memory operations, if it is , I increased the responsible counter , if it is not, it should be arithmetic operation because there are three class and if an instruction doesn't belongs to two of them , it should be in set of third one.

	0!	4!	8!	12!
Iterative	183085	183093	183220	183334
Factorial				
Recursive	183138	183220	183345	183549
Factorial				

Question 2)

In this part, there are two table. First table shows the number of instructions in each three class for different factorial numbers and they are tested with iterative and recursive factorial code. The difference between these two methods is not significant. There is a little difference between them and as I expected, memory operations of recursive method is slightly more than iterative method because recursive method has more read/write operations in itself.

Code part:

I found result for this section in part 1 and I explained it in detail. I just take ratio of each class and I printed them out to the output file.

		0!	4!	8!	12!
Iterative	Memory	54370	54395	54428	54477
	Branch	39496	39501	39522	39546
	Arithmetic	89219	89197	89270	89311
Recursive	Memory	54392	54428	54479	54542
	Branch	39501	39514	39527	39559
	Arithmetic	89245	89278	89339	89448

Here is the usage ratio of each class.

		0!	4!	8!	12!
Iterative	Memory	29.6966%	29.7089%	29.7064%	29.7146%
	Branch	21.5725%	21.5743%	21.5708%	21.5705%
	Arithmetic	48.7309%	48.7168%	48.7228%	48.7149%
Recursive	Memory	29.7%	29.7064%	29.7139%	29.7152%
	Branch	21.569%	21.5664%	21.5588%	21.5523%
	Arithmetic	48.731%	48.7272%	48.7273%	48.7325%

Question 3)

In this part , I found which register used how many times and I take their output and reported here. Also , I found data dependency when I run my code.

Code part:

I will explain what I did in order to find data dependency. I searched when data dependency occurs and I found that there are three cases and they are: Read after Write , Write after Read , Write after Write. In order to determine when these cases occurs, I saved the data of last instruction. What I mean is I saved the info of read and write registers of last instruction and when new instruction executed , I checked whether one of these three data dependency case has occurred and with this info , I found the total number of data dependency.

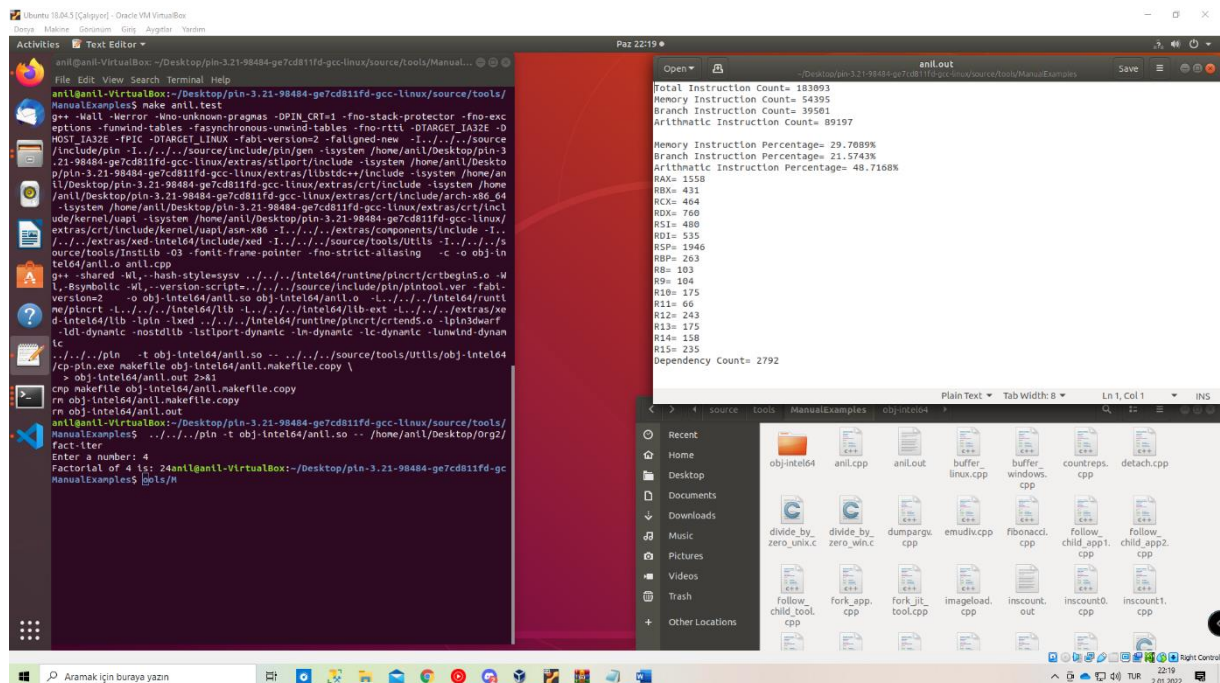
Here is the info table of iterative factorial register usage and dependency results.

ITERATIVE	0!	4!	8!	12!
RAX	1553	1558	1559	1558
RBX	431	431	432	431
RCX	468	464	465	465
RDX	760	760	760	760
RSI	480	480	481	480
RDI	533	535	535	534
RSP	1939	1946	1941	1945
RBP	262	263	262	263
R8	106	103	103	103
R9	104	104	104	104
R10	179	175	179	175
R11	66	66	67	66
R12	241	243	241	243
R13	174	175	174	175
R14	156	158	156	158
R15	234	235	234	235
DEPENDENCY	2793	2792	2793	2791

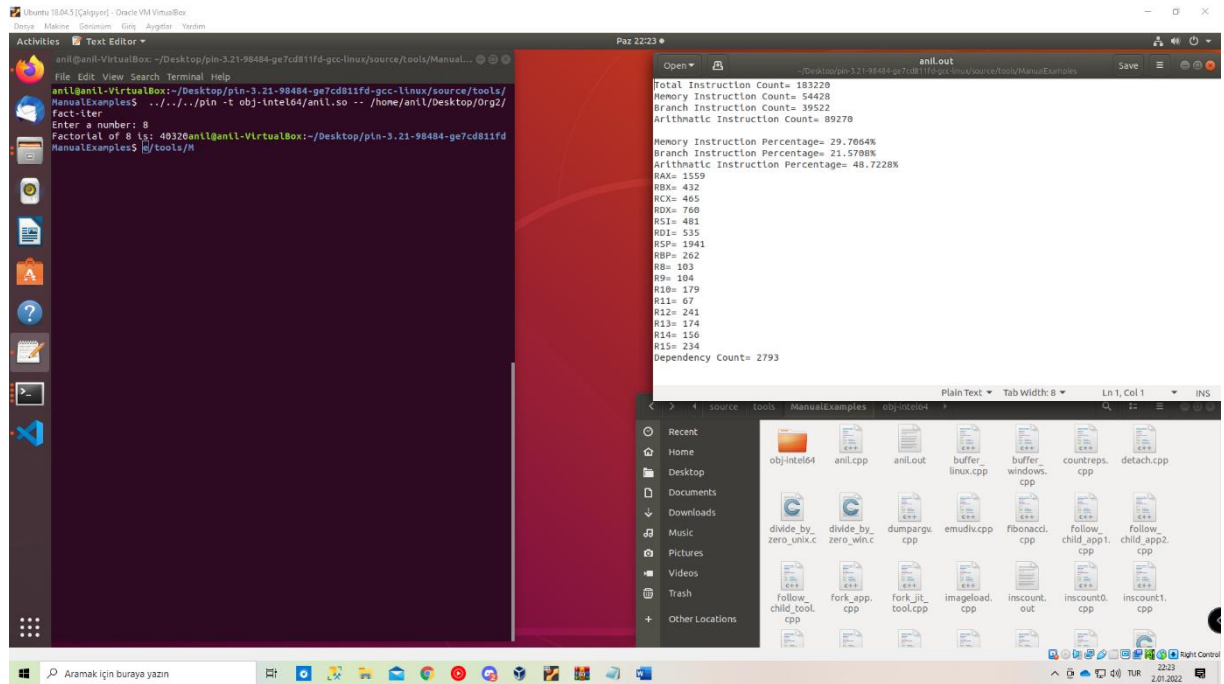
Here is the info table of recursive factorial register usage and dependency results.

RECURSIVE	0!	4!	8!	12!
RAX	1567	1570	1570	1571
RBX	430	434	434	435
RCX	474	466	466	467
RDX	768	764	764	764
RSI	482	482	482	482
RDI	534	535	535	534
RSP	1945	1959	1959	1967
RBP	263	266	266	268
R8	108	105	105	105
R9	104	104	104	104
R10	175	175	175	175
R11	67	67	67	67
R12	244	246	246	248
R13	175	176	176	177
R14	157	159	159	161
R15	237	238	238	239
DEPENDENCY	2803	2806	2806	2812

Iterative 0!



Iterative 8!

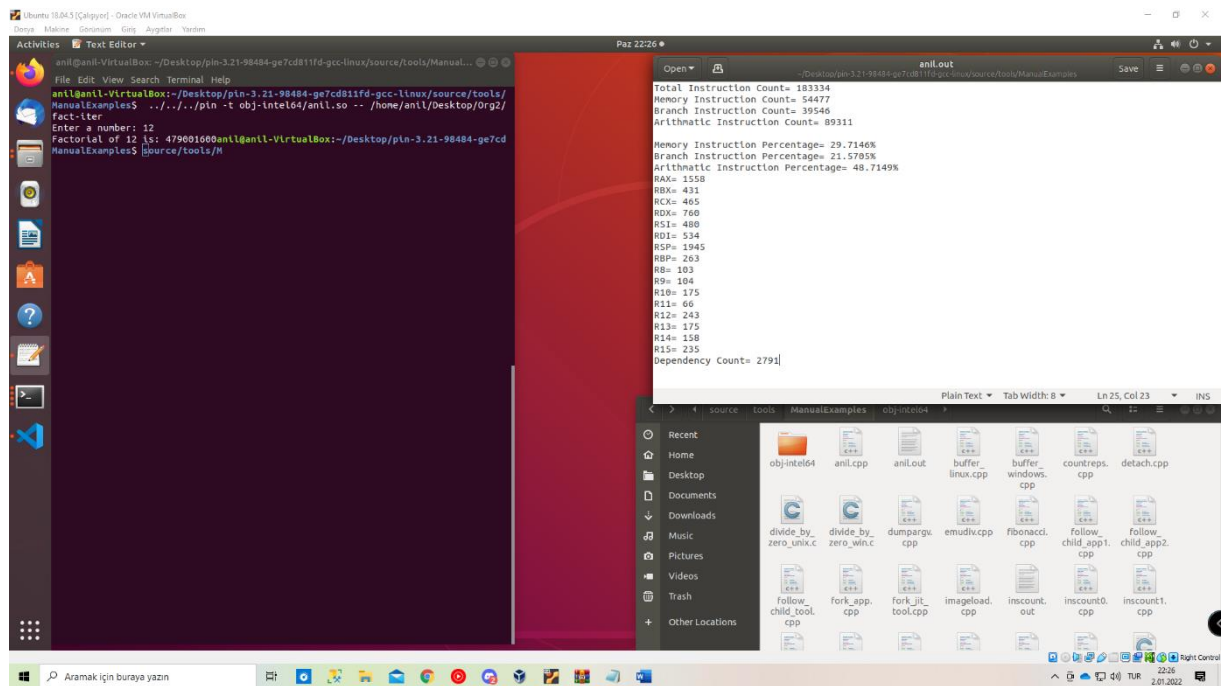


anil@anil-VirtualBox: ~/Desktop/pin-3.21-98484-ge7cd811fd-gcc-linux/source/tools/Manual...
File Edit View Search Terminal Help
anil@anil-VirtualBox:~/Desktop/pin-3.21-98484-ge7cd811fd-gcc-linux/source/tools/
ManualExamples\$../../pin -t obj-intel64/anil.so -- /home/anil/Desktop/Org2/
Fact-iter
Enter a number: 8
Factorial of 8 is: 40320anil@anil-VirtualBox:~/Desktop/pin-3.21-98484-ge7cd811fd
ManualExamples\$ cd tools/M

anil.out
Total Instruction Count= 183220
Memory Instruction Count= 54428
Branch Instruction Count= 39522
Arithmetic Instruction Count= 89270
Memory Instruction Percentage= 29.7064%
Branch Instruction Percentage= 21.5708%
Arithmetic Instruction Percentage= 48.7228%
RAX= 1559
RBX= 432
RCX= 465
RDX= 760
RSI= 481
RDI= 535
RSP= 1941
RBP= 262
RB= 103
R9= 104
R10= 179
R11= 67
R12= 241
R13= 174
R14= 156
R15= 234
Dependency Count= 2793

obj-intel64
anil.cpp
anil.out
buffer_linux.cpp
buffer_windows.cpp
countreps.cpp
detach.cpp
divide_by_zero_unix.c
divide_by_zero_win.c
dumpargu.cpp
emudiv.cpp
fibonacci.cpp
follow_child_app1.cpp
follow_child_app2.cpp
follow_child_tool.cpp
fork_app.cpp
fork_jit_tool.cpp
imgeload.cpp
inscount.out
inscount0.cpp
inscount1.cpp

Iterative 12!

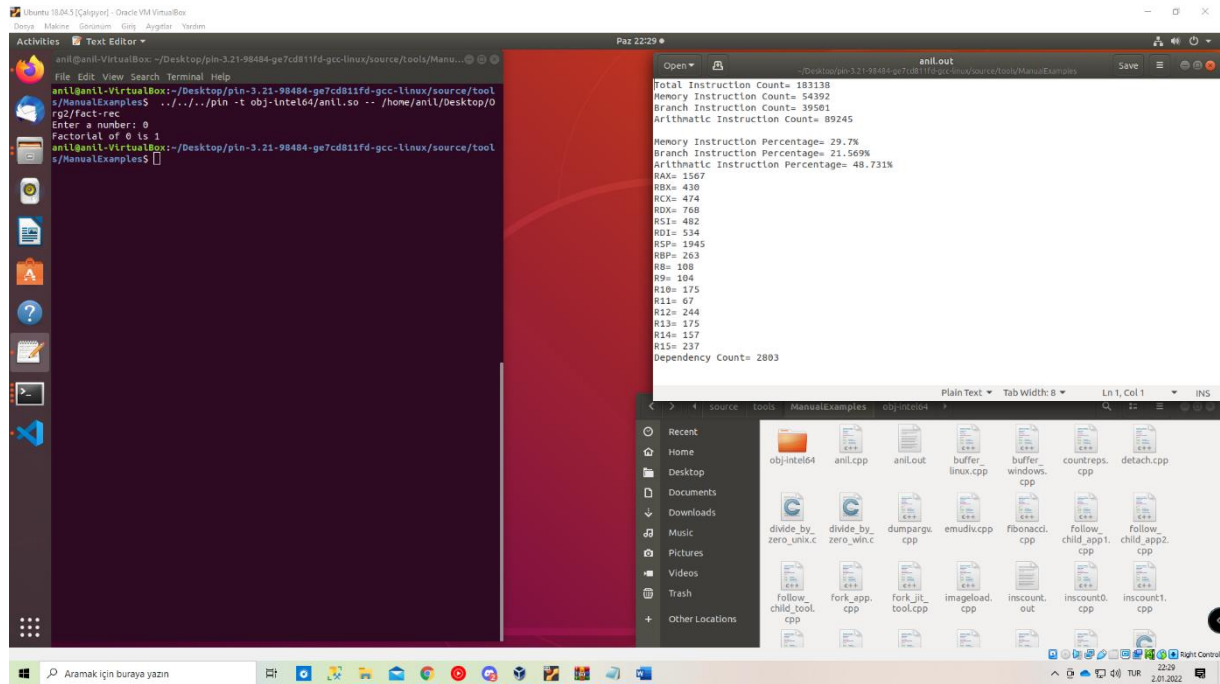


anil@anil-VirtualBox: ~/Desktop/pin-3.21-98484-ge7cd811fd-gcc-linux/source/tools/Manual...
File Edit View Search Terminal Help
anil@anil-VirtualBox:~/Desktop/pin-3.21-98484-ge7cd811fd-gcc-linux/source/tools/
ManualExamples\$../../pin -t obj-intel64/anil.so -- /home/anil/Desktop/Org2/
Fact-iter
Enter a number: 12
Factorial of 12 is: 47900160anil@anil-VirtualBox:~/Desktop/pin-3.21-98484-ge7cd
ManualExamples\$ cd source/tools/M

anil.out
Total Instruction Count= 183334
Memory Instruction Count= 54477
Branch Instruction Count= 39546
Arithmetic Instruction Count= 89311
Memory Instruction Percentage= 29.7146%
Branch Instruction Percentage= 21.5705%
Arithmetic Instruction Percentage= 48.7149%
RAX= 1558
RBX= 431
RCX= 465
RDX= 760
RSI= 480
RDI= 534
RSP= 1945
RBP= 263
RB= 103
R9= 104
R10= 175
R11= 66
R12= 243
R13= 175
R14= 158
R15= 235
Dependency Count= 2791

obj-intel64
anil.cpp
anil.out
buffer_linux.cpp
buffer_windows.cpp
countreps.cpp
detach.cpp
divide_by_zero_unix.c
divide_by_zero_win.c
dumpargu.cpp
emudiv.cpp
fibonacci.cpp
follow_child_app1.cpp
follow_child_app2.cpp
follow_child_tool.cpp
fork_app.cpp
fork_jit_tool.cpp
imgeload.cpp
inscount.out
inscount0.cpp
inscount1.cpp

Recursive 0!

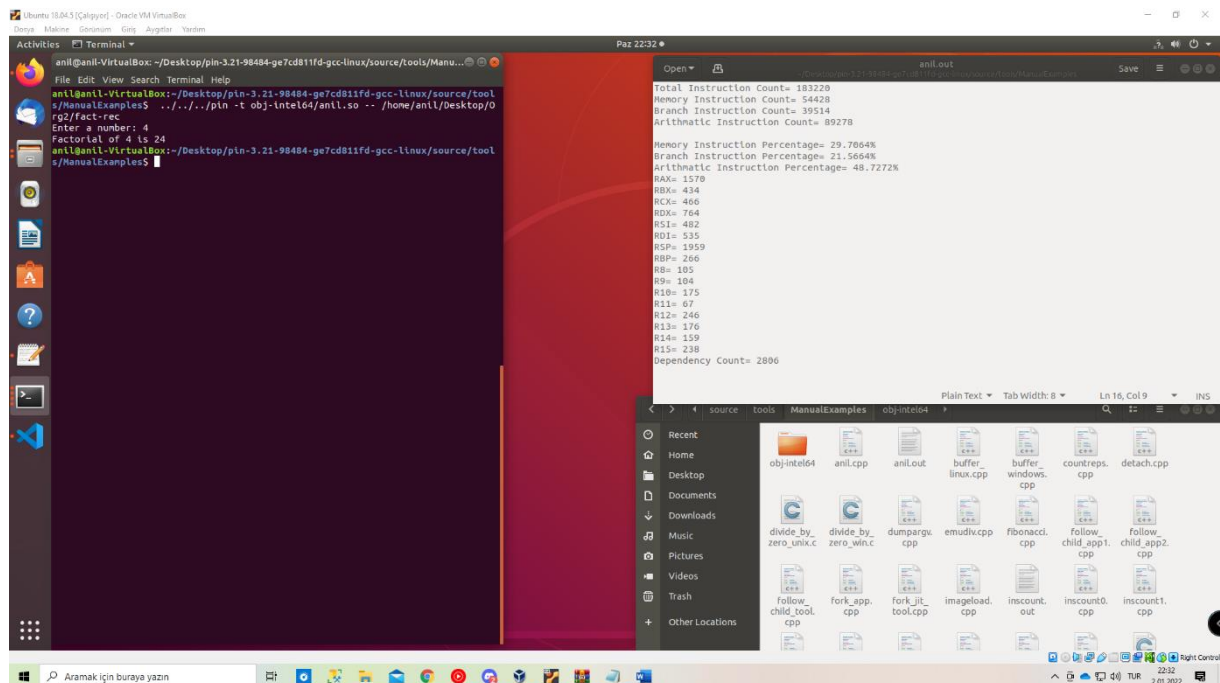


The screenshot shows a Linux desktop environment with a terminal window and a file explorer window. The terminal window displays the following commands and output:

```
anil@anil-VirtualBox: ~/Desktop/pln-3.21-98484-ge7cd811fd-gcc-linux/source/tools/ManualExamples$ ./fact-rec
Enter a number: 0
Factorial of 0 is 1
anil@anil-VirtualBox:~/Desktop/pln-3.21-98484-ge7cd811fd-gcc-linux/source/tools/ManualExamples$
```

The file explorer window shows the contents of the `ManualExamples` directory, including files like `obj-intel64`, `anil.cpp`, `anil.out`, `buffer_linux.cpp`, `buffer_windows.cpp`, `countreps.cpp`, `detach.cpp`, `divide_by_zero_unix.c`, `divide_by_zero_win.c`, `dumpargu.cpp`, `emudiv.cpp`, `fibonacci.cpp`, `follow_child_app1.cpp`, `follow_child_app2.cpp`, `follow_child_tool.cpp`, `fork_app.cpp`, `fork_jit_tool.cpp`, `imagedload.cpp`, `inscount.out`, `inscount0.cpp`, and `inscount1.cpp`.

Recursive 4!

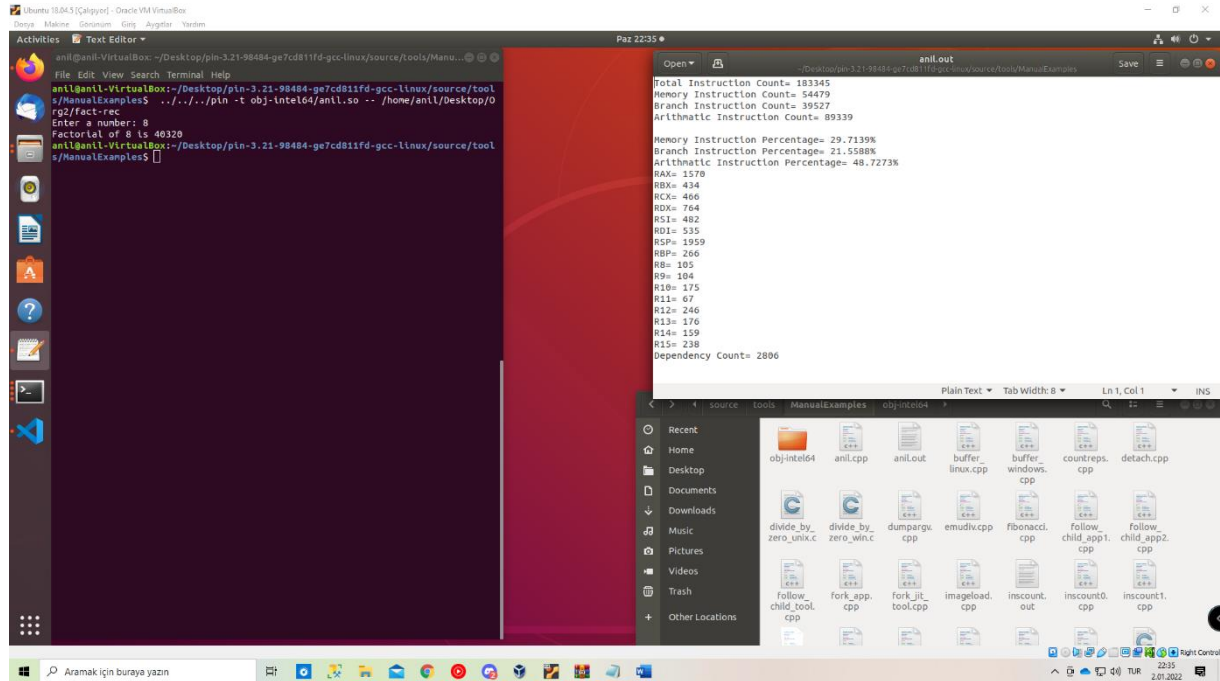


The screenshot shows a Linux desktop environment with a terminal window and a file explorer window. The terminal window displays the following commands and output:

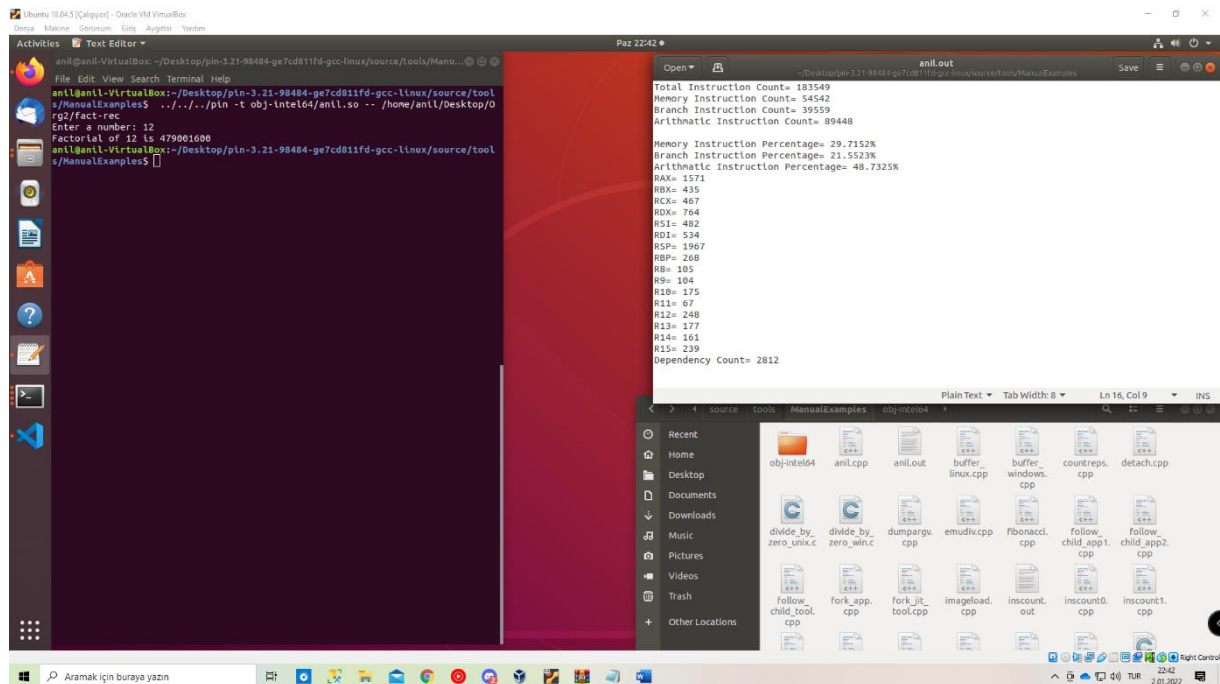
```
anil@anil-VirtualBox: ~/Desktop/pln-3.21-98484-ge7cd811fd-gcc-linux/source/tools/ManualExamples$ ./fact-rec
Enter a number: 4
Factorial of 4 is 24
anil@anil-VirtualBox:~/Desktop/pln-3.21-98484-ge7cd811fd-gcc-linux/source/tools/ManualExamples$
```

The file explorer window shows the contents of the `ManualExamples` directory, including files like `obj-intel64`, `anil.cpp`, `anil.out`, `buffer_linux.cpp`, `buffer_windows.cpp`, `countreps.cpp`, `detach.cpp`, `divide_by_zero_unix.c`, `divide_by_zero_win.c`, `dumpargu.cpp`, `emudiv.cpp`, `fibonacci.cpp`, `follow_child_app1.cpp`, `follow_child_app2.cpp`, `follow_child_tool.cpp`, `fork_app.cpp`, `fork_jit_tool.cpp`, `imagedload.cpp`, `inscount.out`, `inscount0.cpp`, and `inscount1.cpp`.

Recursive 8!



Recursive 12



!