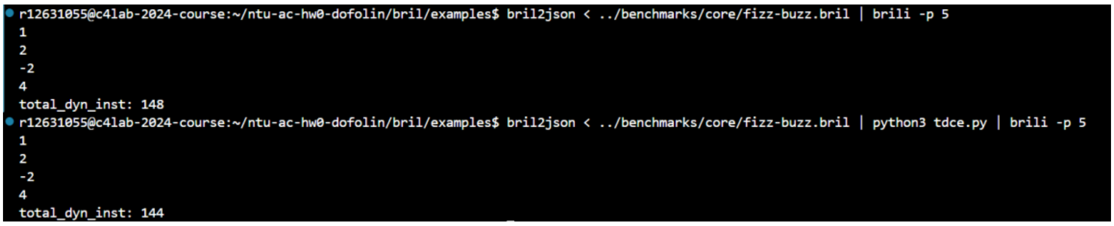
Advanced Compiler Design Case Study A. DCE and LVN

Dead Code Elimination and Local Value Numbering

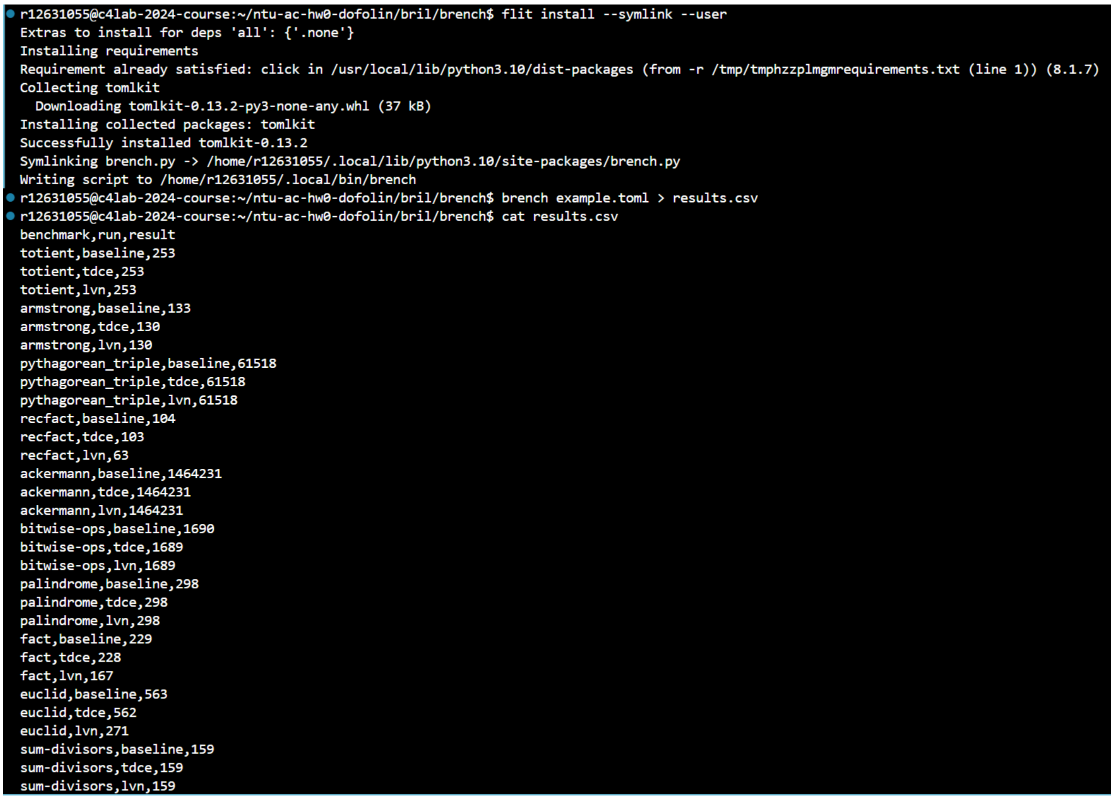
R12631055　林東甫

1.

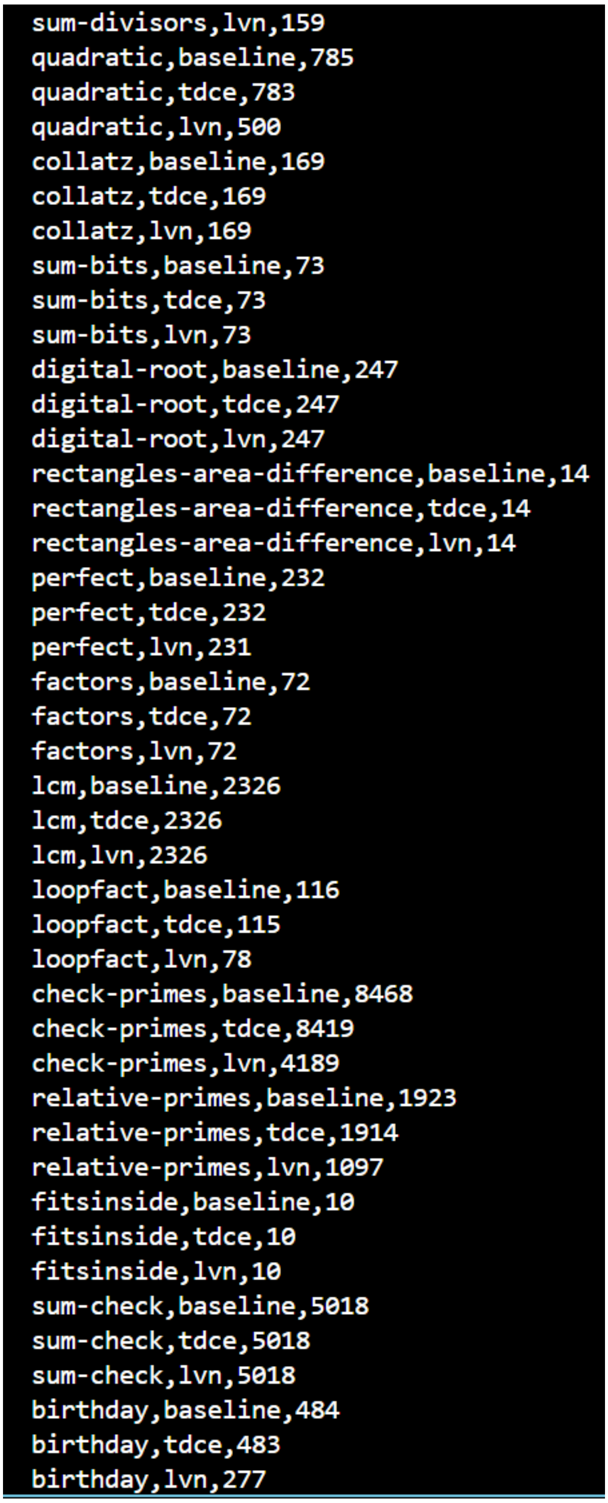


```
r12631055@c4lab-2024-course:~/ntu-ac-hw0-dofolin/bril/examples$ bril2json < ../benchmarks/core/fizz-buzz.bril | brili -p 5
1
2
-2
4
total_dyn_inst: 148
r12631055@c4lab-2024-course:~/ntu-ac-hw0-dofolin/bril/examples$ bril2json < ../benchmarks/core/fizz-buzz.bril | python3 tdce.py | brili -p 5
1
2
-2
4
total_dyn_inst: 144
```

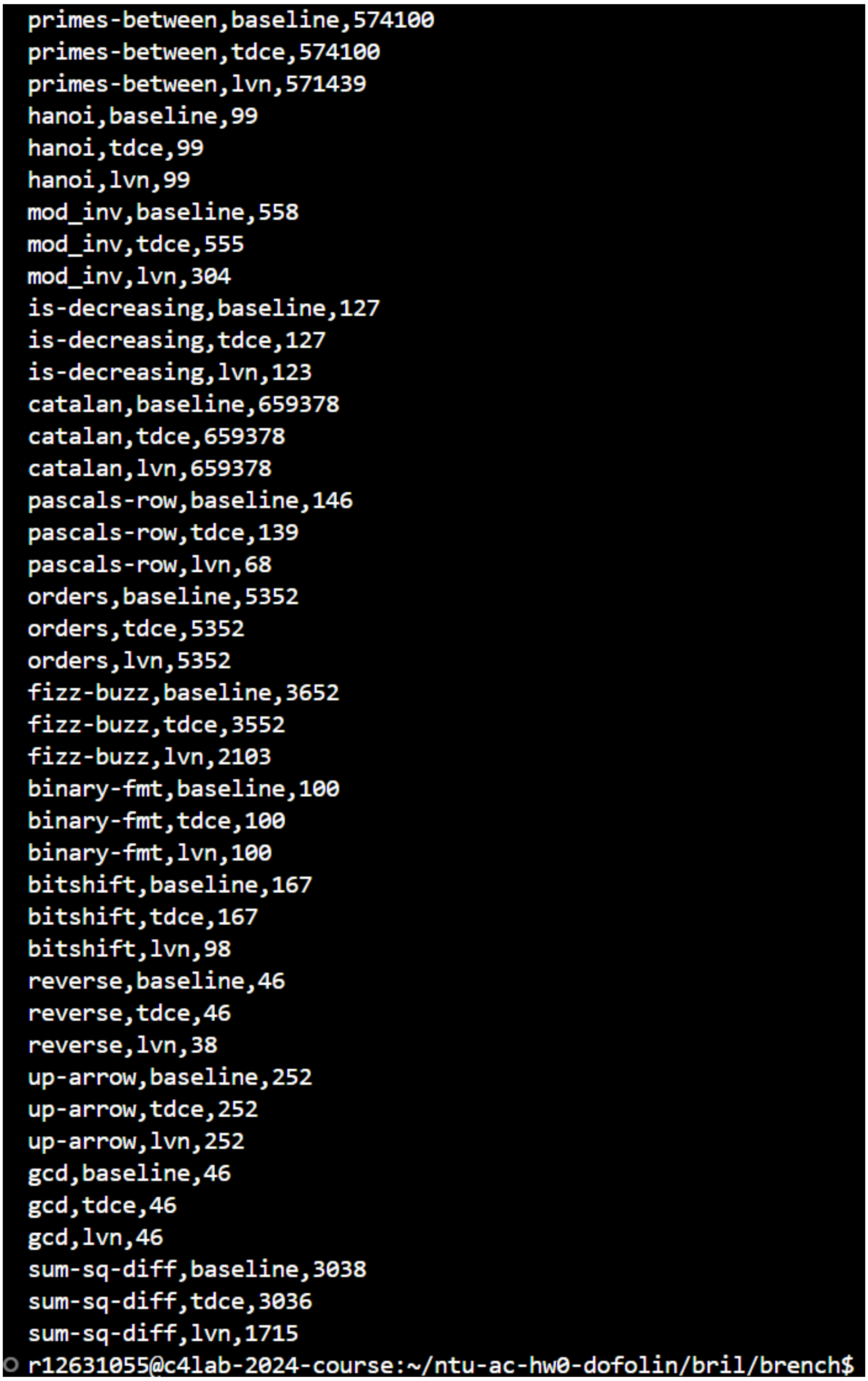
2.



```
r12631055@c4lab-2024-course:~/ntu-ac-hw0-dofolin/bril/brench$ flit install --symlink --user
Extras to install for deps 'all': {'.none'}
Installing requirements
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from -r /tmp/tmphzzplmgmrequirements.txt (line 1)) (8.1.7)
Collecting tomlkit
  Downloading tomlkit-0.13.2-py3-none-any.whl (37 kB)
Installing collected packages: tomlkit
Successfully installed tomlkit-0.13.2
Symlinking brench.py -> /home/r12631055/.local/lib/python3.10/site-packages/brench.py
Writing script to /home/r12631055/.local/bin/brench
r12631055@c4lab-2024-course:~/ntu-ac-hw0-dofolin/bril/brench$ brench example.toml > results.csv
r12631055@c4lab-2024-course:~/ntu-ac-hw0-dofolin/bril/brench$ cat results.csv
benchmark,run,result
totient,baseline,253
totient,tdce,253
totient,lvn,253
armstrong,baseline,133
armstrong,tdce,130
armstrong,lvn,130
pythagorean_triple,baseline,61518
pythagorean_triple,tdce,61518
pythagorean_triple,lvn,61518
recfact,baseline,104
recfact,tdce,103
recfact,lvn,63
ackermann,baseline,1464231
ackermann,tdce,1464231
ackermann,lvn,1464231
bitwise-ops,baseline,1690
bitwise-ops,tdce,1689
bitwise-ops,lvn,1689
palindrome,baseline,298
palindrome,tdce,298
palindrome,lvn,298
fact,baseline,229
fact,tdce,228
fact,lvn,167
euclid,baseline,563
euclid,tdce,562
euclid,lvn,271
sum-divisors,baseline,159
sum-divisors,tdce,159
sum-divisors,lvn,159
```



```
sum-divisors,lvn,159
quadratic,baseline,785
quadratic,tdce,783
quadratic,lvn,500
collatz,baseline,169
collatz,tdce,169
collatz,lvn,169
sum-bits,baseline,73
sum-bits,tdce,73
sum-bits,lvn,73
digital-root,baseline,247
digital-root,tdce,247
digital-root,lvn,247
rectangles-area-difference,baseline,14
rectangles-area-difference,tdce,14
rectangles-area-difference,lvn,14
perfect,baseline,232
perfect,tdce,232
perfect,lvn,231
factors,baseline,72
factors,tdce,72
factors,lvn,72
lcm,baseline,2326
lcm,tdce,2326
lcm,lvn,2326
loopfact,baseline,116
loopfact,tdce,115
loopfact,lvn,78
check-primes,baseline,8468
check-primes,tdce,8419
check-primes,lvn,4189
relative-primes,baseline,1923
relative-primes,tdce,1914
relative-primes,lvn,1097
fitsinside,baseline,10
fitsinside,tdce,10
fitsinside,lvn,10
sum-check,baseline,5018
sum-check,tdce,5018
sum-check,lvn,5018
birthday,baseline,484
birthday,tdce,483
birthday,lvn,277
```



```
primes-between,baseline,574100
primes-between,tdce,574100
primes-between,lvn,571439
hanoi,baseline,99
hanoi,tdce,99
hanoi,lvn,99
mod_inv,baseline,558
mod_inv,tdce,555
mod_inv,lvn,304
is-decreasing,baseline,127
is-decreasing,tdce,127
is-decreasing,lvn,123
catalan,baseline,659378
catalan,tdce,659378
catalan,lvn,659378
pascals-row,baseline,146
pascals-row,tdce,139
pascals-row,lvn,68
orders,baseline,5352
orders,tdce,5352
orders,lvn,5352
fizz-buzz,baseline,3652
fizz-buzz,tdce,3552
fizz-buzz,lvn,2103
binary-fmt,baseline,100
binary-fmt,tdce,100
binary-fmt,lvn,100
bitshift,baseline,167
bitshift,tdce,167
bitshift,lvn,98
reverse,baseline,46
reverse,tdce,46
reverse,lvn,38
up-arrow,baseline,252
up-arrow,tdce,252
up-arrow,lvn,252
gcd,baseline,46
gcd,tdce,46
gcd,lvn,46
sum-sq-diff,baseline,3038
sum-sq-diff,tdce,3036
sum-sq-diff,lvn,1715
r12631055@c4lab-2024-course:~/ntu-ac-hw0-dofolin/bril/brench$
```

3.

Because LVN framework tracks values based on their computed results, rather than relying on variable names, so it can handle DCE, CSE, copy propagation, and constant propagation simultaneously.

This tracking allows LVN to perform all of these optimizations as part of the same framework without the need for separate optimization passes:

CSE by assigning the same value number to redundant subexpressions.

Copy propagation by assigning the same value number to variables that are simple copies of each other.

Constant propagation by recognizing variables assigned constant values and substituting those constants where applicable.

DCE by eliminating instructions that produce values with no subsequent use.

The LVN framework handles all these optimizations in a unified manner by maintaining a single table of value numbers for expressions, variables, constants, and copies within a basic block.