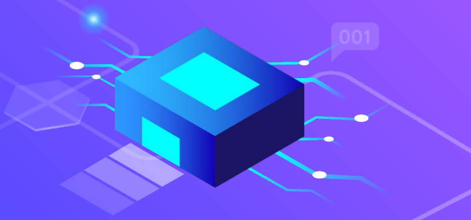
Performance Effects of TLB Inclusion for Address Translation

By Christopher Brant, Quan Pham, and Nick Poindexter

T.L.B. : Team Let's Ball

Introduction



- A Translation Lookaside buffer is a hardware module that stores recent virtual address to physical address translations
- TLBs are typically used to reduce the overhead related to accessing virtual memory and therefore increase performance
- Since CPUs access virtual memory via both instruction fetch and loads/stores, there can be both instruction and data tlbs

Experiment Description

- 5 different TLB configurations:
  - No TLB for instructions nor for data
  - Instruction TLB and no data TLB
  - Data TLB and no instruction TLB
  - Both instruction TLB and data TLB
  - Both TLBs with full associativity and random replacement policy

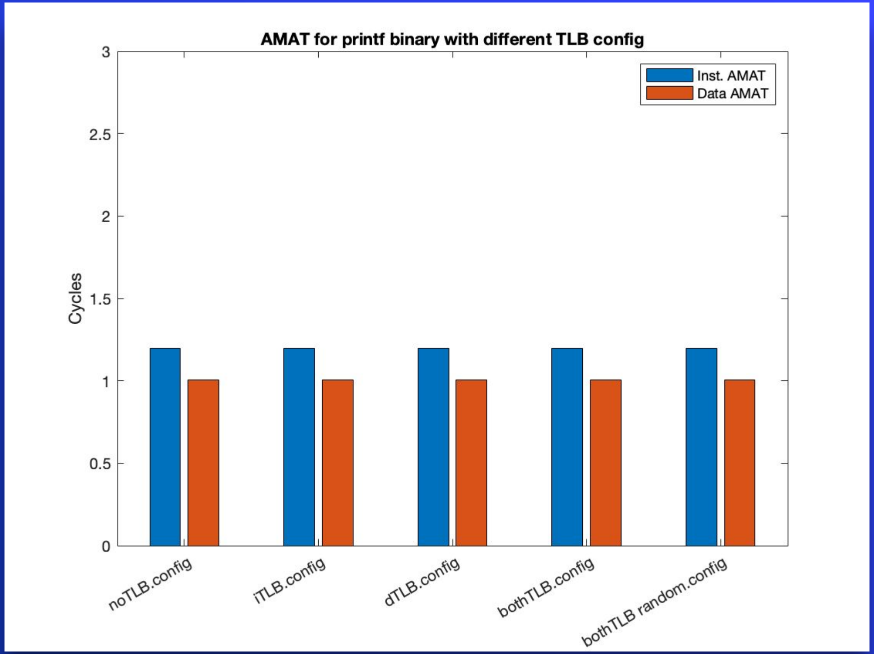
Experiment Description

- 5 pre-compiled benchmark binary files:
  - fmath: floating point math
  - llong: long long math
  - lswlr: print ‘hello world’
  - math: integer math
  - printf: print extended output to terminal

Experiment Description

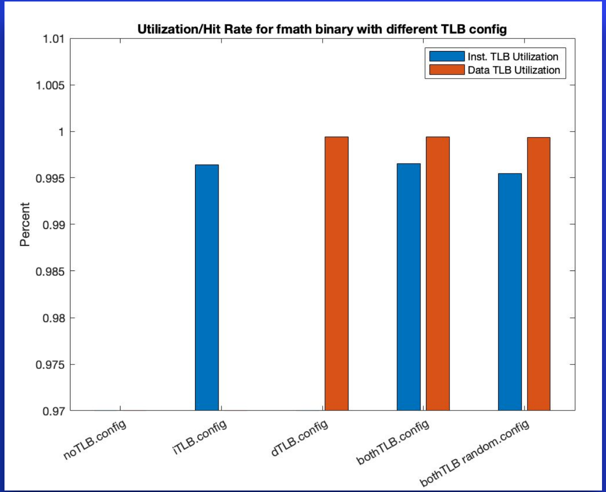
- Simulate every combination of TLB configuration and binary file
- Intentionally have high TLB miss latency to exaggerate consequences
- Metrics of interest:
  - AMAT
  - TLB hit rate/utilization
  - CPI

Metrics: AMAT

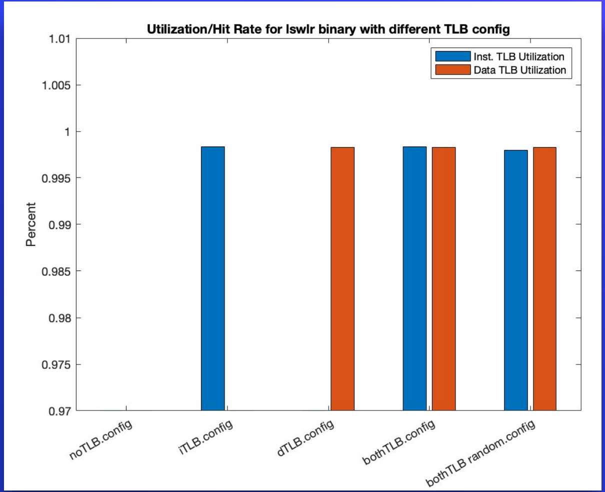


printf Binary File Used in the Above Test

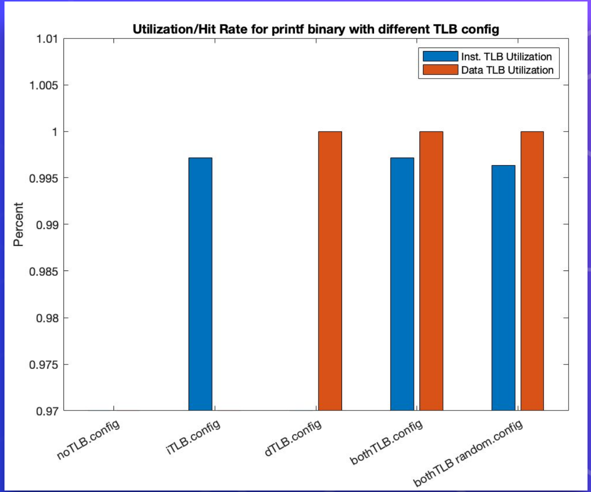
Metrics: TLB Hit Rate/Utilization



fmath Binary Test File

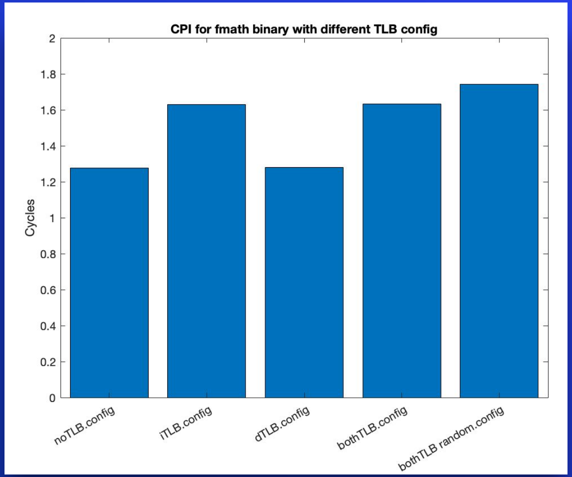


lswlr Binary Test File

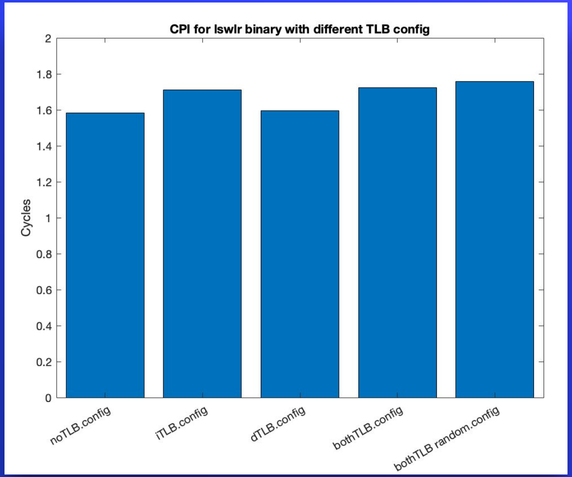


printf Binary Test File

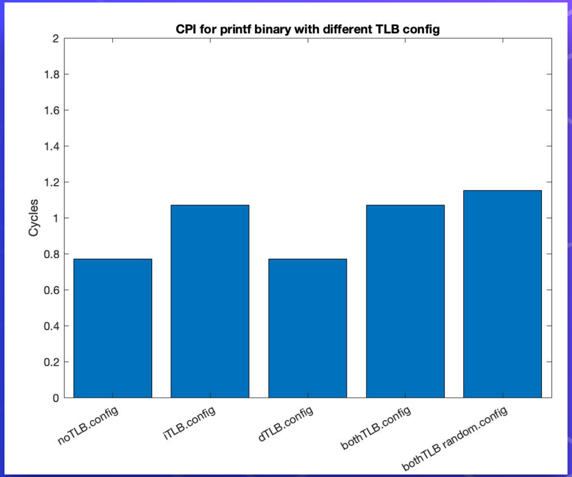
Metrics: CPI



fmath Binary Test File



lswlr Binary Test File



printf Binary Test File

Discussion of Results

Utilization

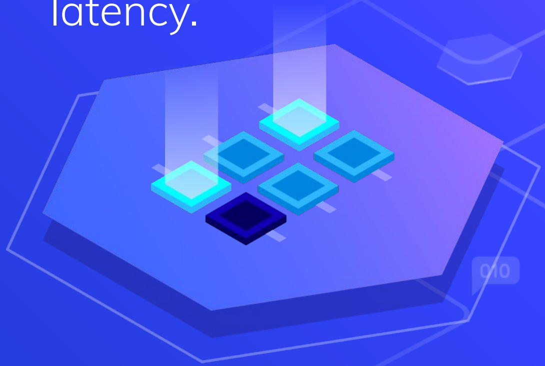
- High utilization is typical across test cases where TLBs are included.
- More complexity of programs yield lower utilization/hit rate

AMAT

- AMAT will stay the same because the config of the caches is not changing

CPI

- Variance of CPI is a result of the change in TLB hit rate and a high TLB miss latency.



>99%

Hit rate of every TLB configuration we tested in SimpleScalar.

Discussion of Results (cont.)

- Including a TLB will increase the complexity of the design but will also decrease CPI and AMAT
- As seen in our CPI results slide, it can be observed that the CPI increases in the case that the TLB replacement policy is changed from LRU to random.
- The changes in results between the different TLB configurations with regards to performance in the metrics that we recorded was relatively small, even when we changed the associativity and replacement policy for our final configuration.

Conclusions

- It can be concluded that introducing a TLB for fast memory address translation does have an impact on system performance, as our results were able illustrate with our high TLB miss latency. If that latency were to be reduced we would expect that performance would be better than not having a TLB at all.
- Just as well, it can be concluded that changing the TLB replacement policy to be random replacement instead of LRU will reduce the performance of the overall system.

Future Work and Limitations

- Prior to finishing our report we would like to run the same tests again, but with a much lower TLB miss latency than we have currently.
- One thing we could have done differently is also run tests that contained more data accesses.
- The reason we could not is that the cross compiler no longer could be compiled/installed correctly on our VMs, and therefore we were unable to compile our own programs for testing purposes.

Thanks for Watching!

Any questions?

Please contact us via email

quanpham@ufl.edu

cdbrant@ufl.edu

npoindexter@ufl.edu