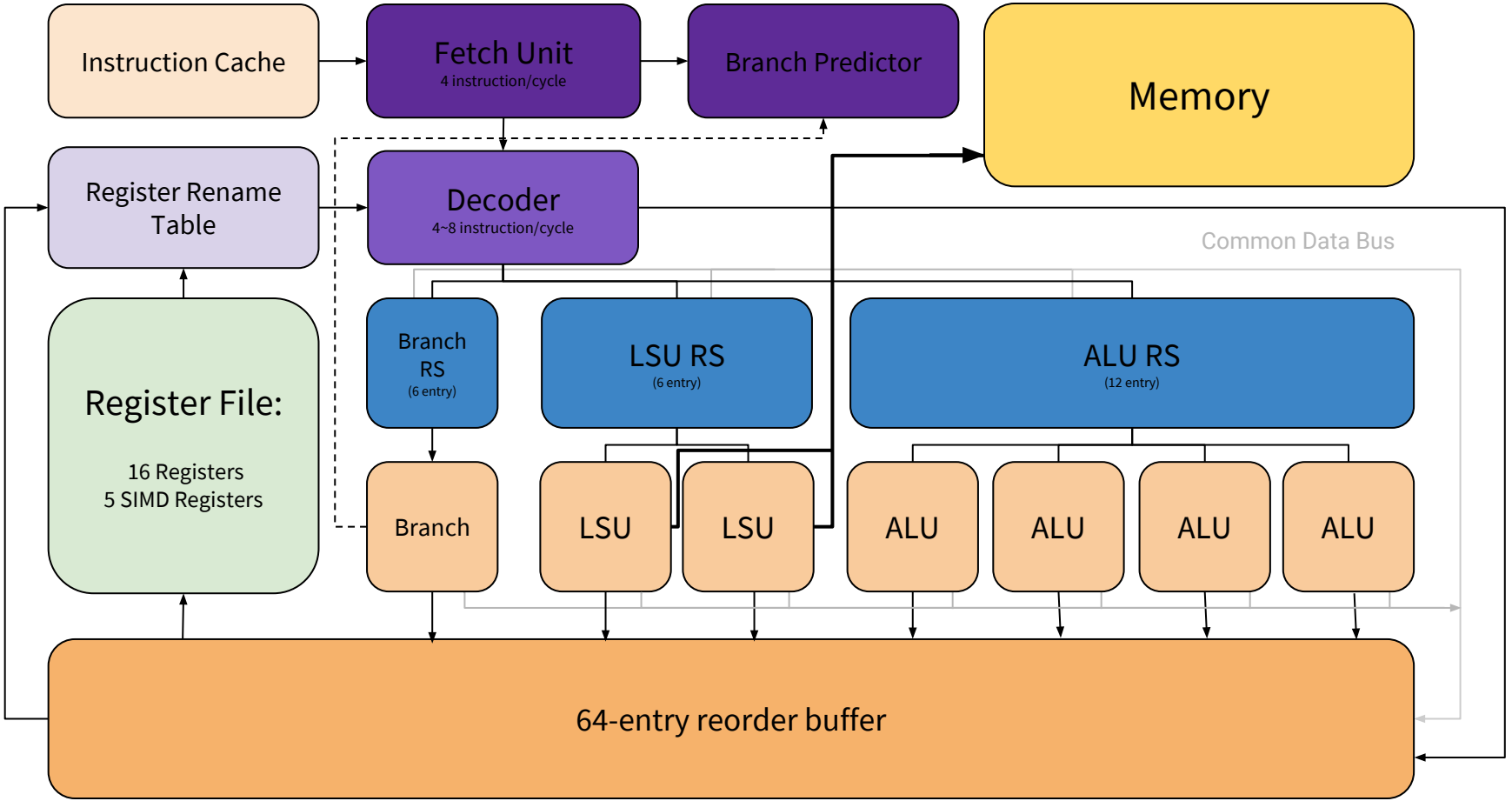
Superscalar Processor Simulator

Abdu Elturki
ae15920@bristol.ac.uk


Instruction Cache
Fetch Unit
4 instruction/cycle
Branch Predictor
Memory
Register Rename Table
Decoder
4~8 instruction/cycle
Common Data Bus
Register File:
16 Registers
5 SIMD Registers
Branch RS
(6 entry)
LSU RS
(6 entry)
ALU RS
(12 entry)
Branch
LSU
LSU
ALU
ALU
ALU
ALU
64-entry reorder buffer

Architectural Features

Parallel execution pipelines:

- 4 ALU Units, 2 Load/Store Unit, and 1 Branch Unit. (All configurable)
- Supports SIMD Operations.
- Non-blocking issue.

Decoder:

- Decodes and issue 4 instructions/cycle (configurable).
- Issues upto 8 instructions/cycle if misprediction happened.
- Performs register renaming.

Reservation stations:

- Grouped Reservation Stations.
- Out-of-Order issue when operands are valid.
- Bypass when empty and operands are valid.

Speculative Execution:

- Dual 2-bit Dynamic (Smith) Branch predictor. One for forward branches and other for backwards.
- Sequential and Target Instruction buffers to reduce recovery time from misprediction.
- Store buffer used to keep data that is planned to store in memory during speculation. Execution Units load from the buffer during speculation as well.

Out-of-order execution:

- Dynamic dispatch and renaming with Tomasulo' algorithm.
- ROB's completion is In-order and commits renamed registers.
- Dispatches results to reservation stations.

Benchmarks

Kernel	Description
Sorting	Sorting done in memory; available algorithms Bubble and Selection Sort
Fibonacci	Calculates Fibonacci sequence
GCD	Calculates the greatest common divisor of two values
Matrix Multiplication	Performs multiplication of two matrices using SIMD instructions
Edge Detection	Performs convolution on an image using edge filter and SIMD instructions

Experiment: RS Bypass

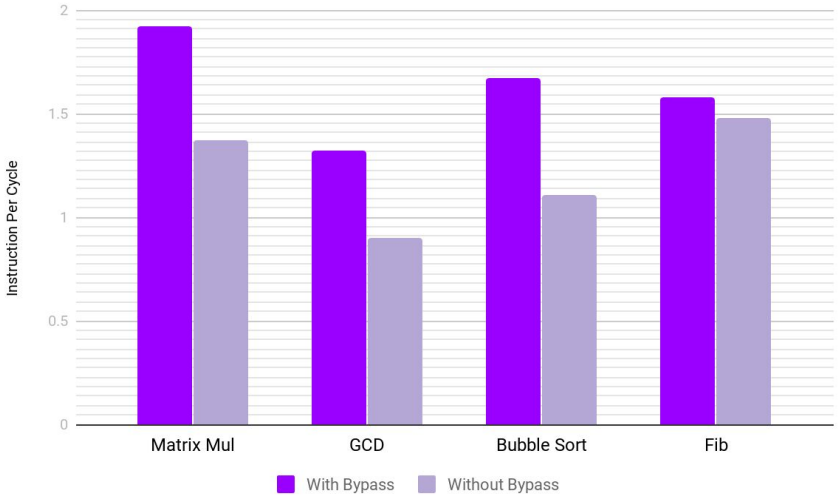
Hypothesis:

Adding decode into the reservation station typically takes a cycle. The feature of bypassing reservation station can increase IPC.

Experiment:

The kernels Matrix Multiplication, GCD, Bubble Sort, and Fibonacci were tested. The effect on IPC were measured.

Result:



For matrix multiplication, which generally makes memory access to load the matrices. It was noticed that bypass made the program have 28% higher IPC compared to non-bypass. The results are similar for GCD and Bubble sort. Fibonacci hasn't got significant difference, and it doesn't do any memory access instructions and GCD doesn't either. However, GCD uses more expensive instructions such as division which could mean that applications with expensive instructions benefits greatly from Reservation Station Bypass feature.

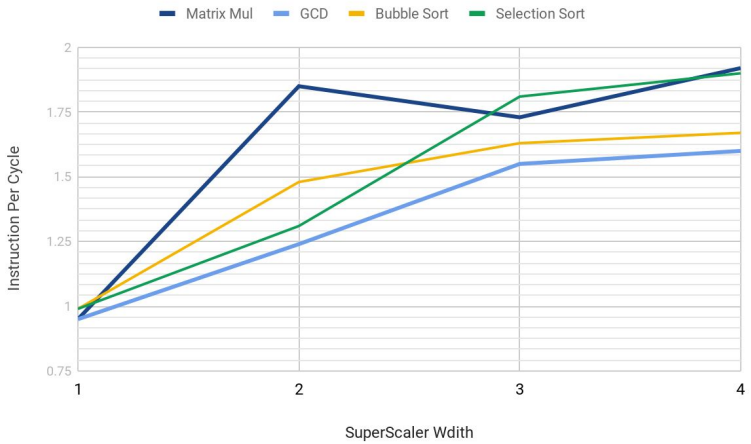
Experiment: Instruction Throughout

Hypothesis:

Having higher instruction throughout should provide higher instruction per cycle executed.

Experiment:

The kernels Matrix Multiplication, GCD, Bubble Sort, and Selection were tested. The SuperScalar Width was modified for each run and the effect on IPC were measured.



Result:

As expected, the higher the instruction throughout the higher IPC. The Matrix Multiplication gave an interesting result, as it can be seen that SuperScalar with width of 2 performs better than width 3. This perhaps could be due to speculative execution, as instruction order might have caused it to perform better.

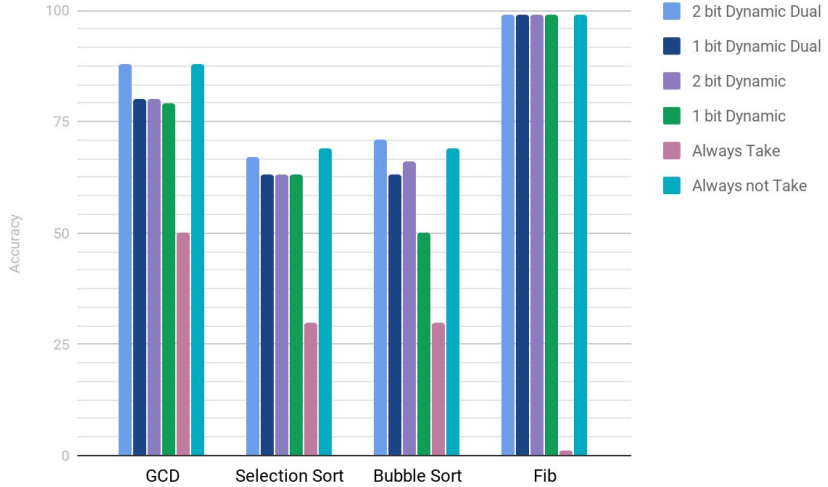
Experiment: Branch Prediction

Hypothesis:

Having separate branch predictor for forward and backward branches will have higher accuracy. 2-bit Dynamic branch predictor will have higher accuracy compared to static or 1-bit dynamic.

Experiment:

The kernels GCD, Fibonacci, Bubble Sort, and Selection were tested. The number of ALU units were adjusted for each run. There will be 1 branch and 2 Load/Store units.



Result:

As can be seen, there only minor benefit of having Dual Dynamic Branch predictions. However in instances of having static branch prediction perform with higher accuracy, the dual set up is able to match it. The fibonacci program had 99% accuracy in all branch prediction except for “Always not Taken” Branch predictor.

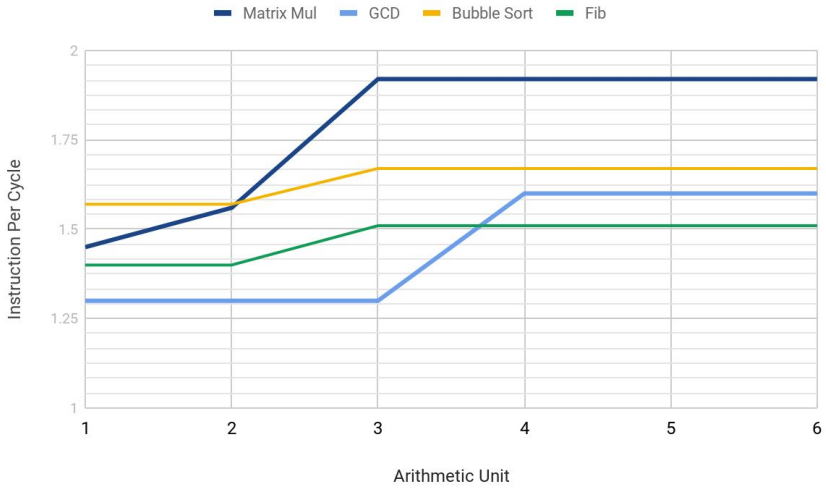
Experiment: Arithmetic Unit

Hypothesis:

Greater number of ALU can increase performance to a point. Increasing that number further may not provide any additional benefits

Experiment:

The kernels Matrix Multiplication, GCD, Bubble Sort, and Selection were tested. The number of ALU units were adjusted for each run. There will be 1 branch and 2 Load/Store units.

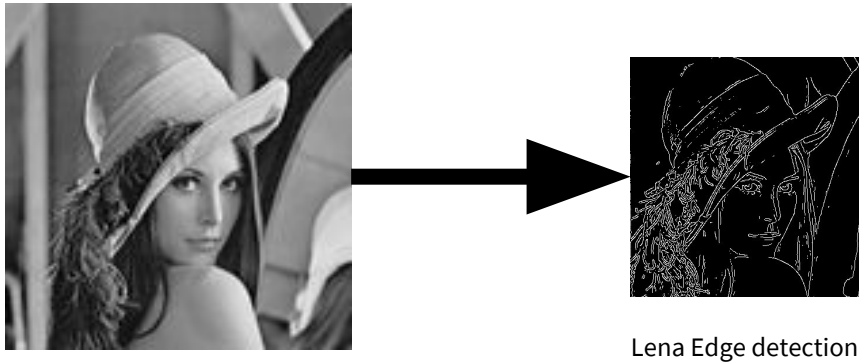


Result:

Most programs saw improvement with around 3 ALUs. GCD received noticeable benefit with 4 ALUs which makes this processor optimal with 4 ALUs. It should be noted that all IPC is higher than 1.0 due to having 2 Load/Store units

(Bonus) Experiment: Edge Detection

The processor is capable of doing convolution using SIMD instructions. The process takes about an hour to complete. FMA instruction set would probably decrease the time significantly.



Lena
(128x128)

Lena Edge detection
(62x62)

Cycles: 265758
IPC: 1.53
Prediction Rate = 99.2%