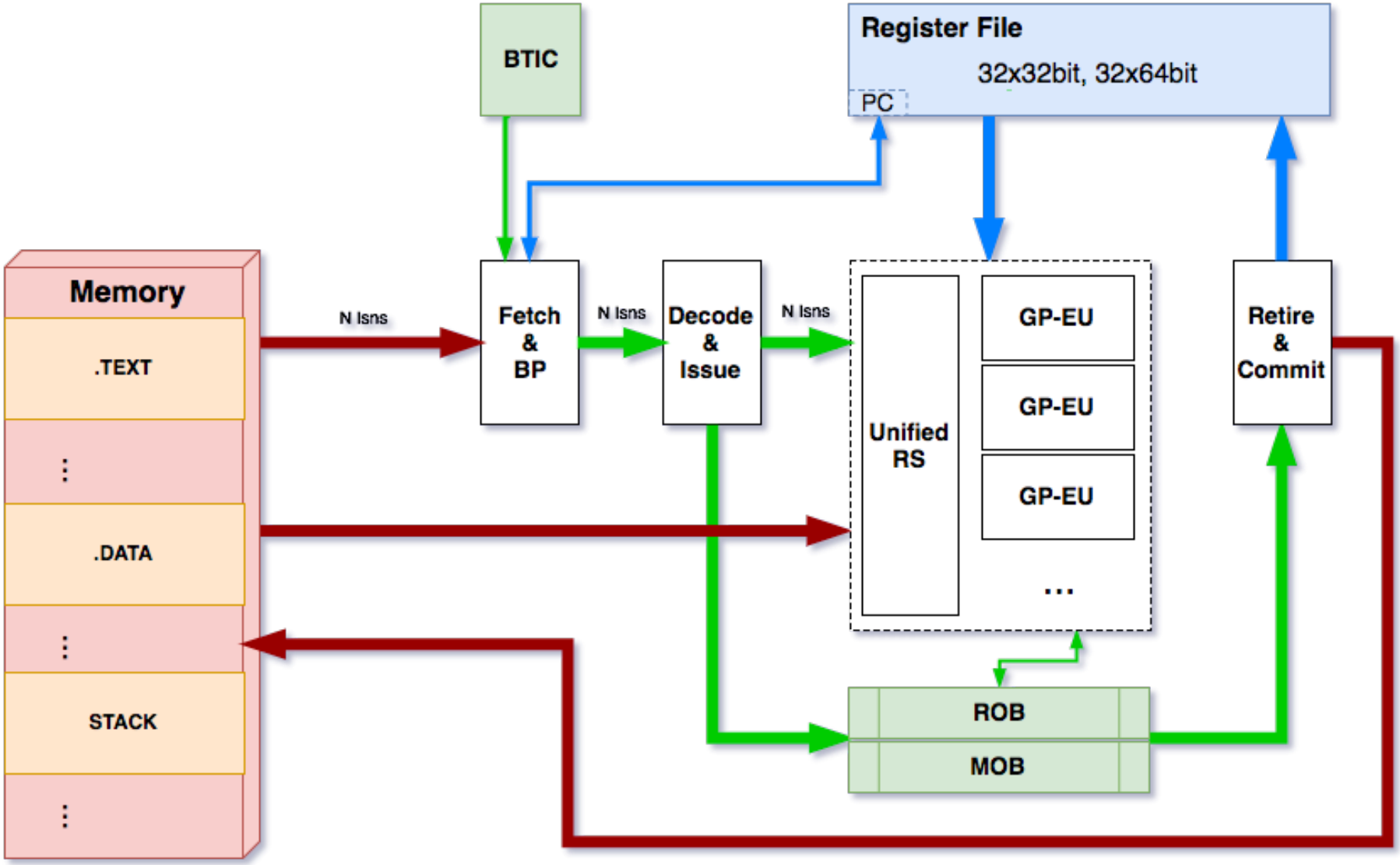
Kraken

Joy Yeh
cy13308



Avg. IPC: 3.9
Max IPC: 6

Features 1

- **Pipelined**:
  - 4 major stages: Fetch, Decode, Execute, Commit
  - Sub-pipelines
- **Superscalar**:
  - OoO Execute, In-Order completion, arbitrary number of execution units (assumed GP).
  - Speculative Execution – arbitrary level*
- **Branch Prediction**:
  - Fixed, Static, Dynamic n-bits
  - Branch Target Instruction Cache (BTIC) with Relative Instruction Pointers (RIPs)

Features 2

- **Dependency Reduction**:
  - Register file "extension" using ROB
  - Second ROB (Memory Order Buffer) for load/stores*
- **ISA**:
  - AArch64, Neon, VFP (ARM vixl decoding library)
  - Executes real ARM64 benchmark binaries compiled from C. Can't link libraries yet, just pure C with a core Maths library

Latency Model

• ADD/SUB/SHIFT/MOV…	1
• Shifted ADD/SUB	2
• Load/Store (assume L1)	4/5
• Mul	4
• Div	20
• Branches	5

(Data pieced together from different processors…)

Benchmarks

Ranging (10^(6~8) instructions):

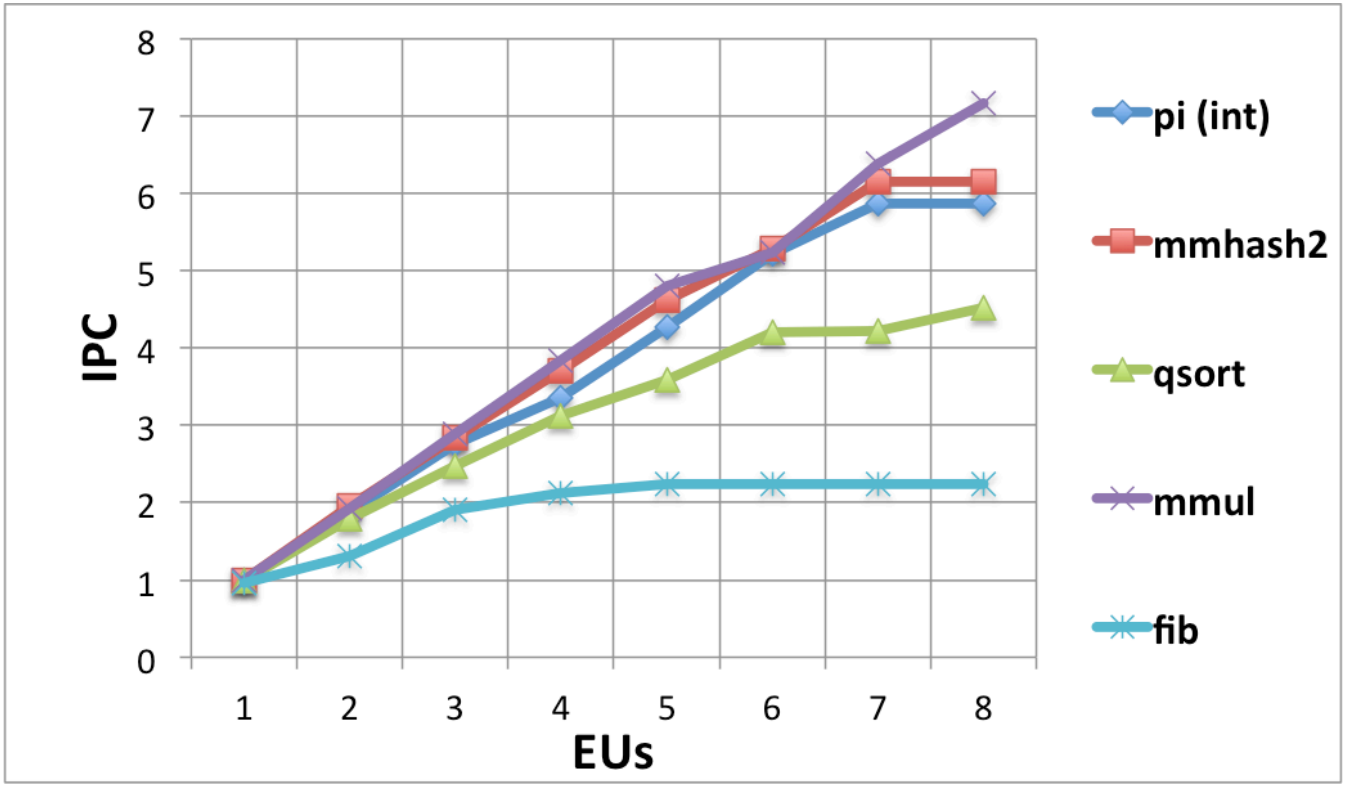
- Unrolled: Fac
- Tight loops: Fac, Fibs
- Loaded loops: QSort, Pi, MMul, BUDE
- Recursion: Fac
- Artificial/Misc.: MMHash2, Nops

H1

IPC increases as the number of general purpose execution units, with diminishing returns

Config:

6 EU
5-bit BP
OOOSS
Clang –O0



Observations

- Matrix multiply is very easily parallised hence the continued trend, but other codes have more intra-loop depedencies so they stop around 6

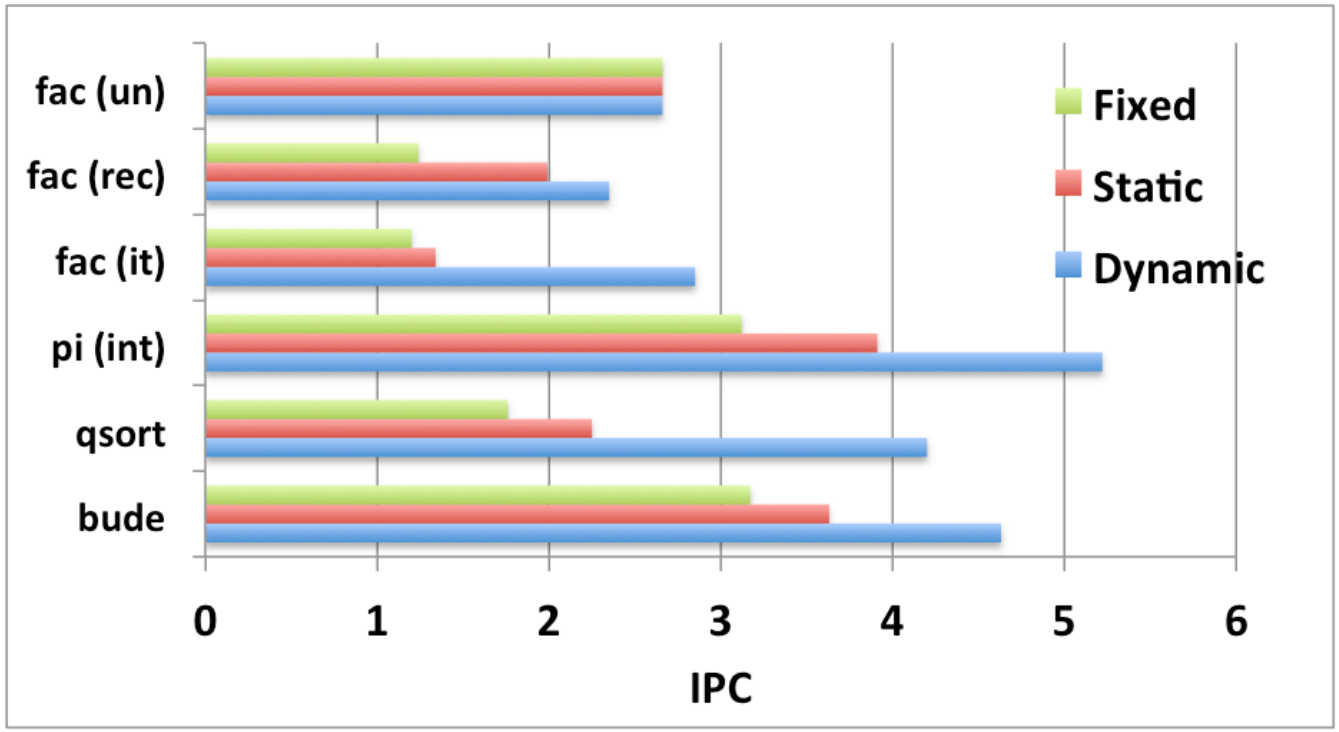
- Really tight loops like Fib suffer a lower plateau. Pi, hash and MMul have a lot more work per loop to parallelise. **Result differs depending on code generation method.

H2

Ideally, branch prediction should deliver an IPC almost equal to unrolled loop

Config:

6 EU
5-bit BP
OOOSS
Clang –O0



Observations

- Dynamic Factorial gives HIGHER IPC than unrolled Factorial!
  - Misc. instructions in the loop + speculation allowed extra instruction counts
- Large amount of work within each loop reduces the advantage of BP (BUDE – less diff.).

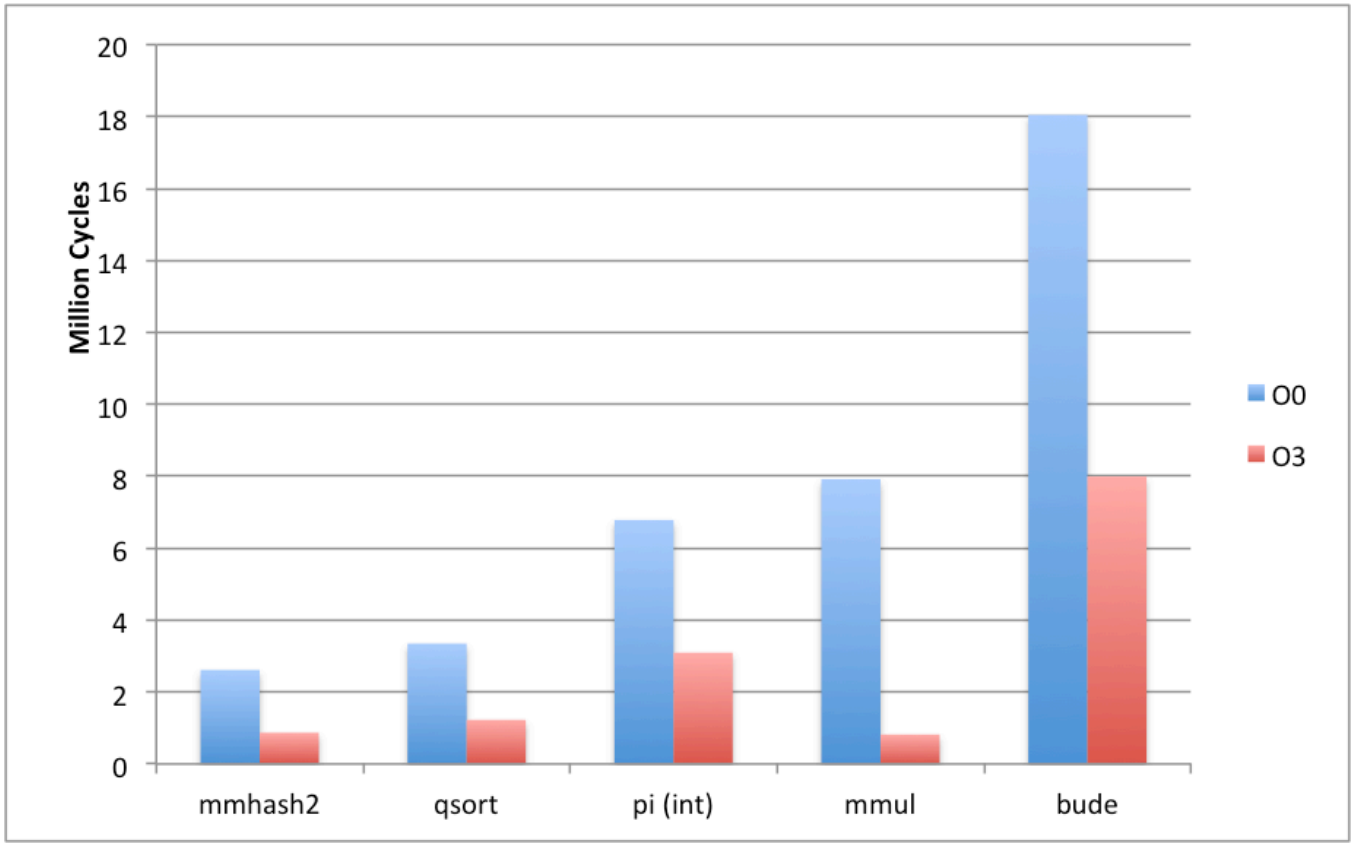
H3

Compiler optimisation should help increase IPC… or not?

Config:

6 EU
5-bit BP
OOOSS

Clang –O3
-ffast-math
-fvectorize
-mcpu=cortex-a57



Observations

- Compiler optimised for code density and ultimately minimum cycle count, and not IPC (MMHash2 and Qsort have lower IPC in O3).
- The total cycle count is a better/more important measure when considering real world application. Lower IPC can be improved by adding SMT.