The implementation of AArch64 NEON™ Instruction Set

Ana Pazos

Senior Staff Engineer, QuIC (Qualcomm Innovation Center)

Jiangning Liu

Principal software engineer, ARM

Implementation Goals

- Support all instructions in AArch64 NEON™
- Support all ACLE intrinsics
- Provide ARM 32-bit backward compatibility
- Integrated assembler on by default
  - MC Layer completeness: instruction printing, asm parsing, assembling, disassembling.

Implementation Status

Instruction Class	100% complete	Complete except community code review	Under active development
AdvSIMD (imm)	14		
AdvSIMD (3 same)	80		
AdvSISD (3 same)	32		
AdvSIMD (lselem)	14		
AdvSIMD (lsone)			36
AdvSIMD (3 diff)	26		
AdvSIMD (misc)			62
AdvSIMD (across)	23		
AdvSIMD (insdup)	24	2	
AdvSIMD (by element)	18		
AdvSIMD (shift)	28		
AdvSIMD (table)		8	
AdvSIMD (perm)		6	
AdvSIMD (extract)		2	
AdvSISD (3 diff)	3		
AdvSISD (misc)	31	5	
AdvSISD (pairwise)	6		
AdvSISD (copy)			4
AdvSISD (by element)			10
AdvSISD (shift)	22	2	
AdvSIMD (lselem-post)		14	
AdvSIMD (lsone-post)			36
AdvSIMD Crypto (aes)		4	
AdvSIMD Crypto (3 sha)		7	
AdvSIMD Crypto (sha)		3	
	296	51	175
	61.49%	10.15%	28.35%

Implementation Details

- Minimize LLVM IR intrinsics
  - Reusing ARM definitions when possible
- SISD support is implemented
  - Defined v1ix and v1fx vector types
    - to distinguish NEON™ scalar types from integer/FP types
  - To be reworked when global instruction selection is available
- Shared arm_neon.h between ARM and AArch64
- MI–based scheduler turned on by default
  - Overloaded memory cost model in Selection DAG-based scheduler causes issues
- Using RegisterOperand instead of Register class when defining Neon registers.

Testing

- LLVM Regression tests
  - Assembling, disassembling, codegen (.ll, .c, .txt)
- LLVM auto-generated ACLE tests
  - arm_neon_test.h, arm_neon_sema.h
- ARM's MC Hammer test
  - MC layer test against golden implementation
- ARM's Emperor tests
  - ACLE composition test
- Other vector workloads to test pattern matching

Future Work

- Complete NEON™ implementation by end of Nov. 2013
- Performance tuning
  - Investigate regressions compared to the most recent GCC
    - Use ARM's foundation model available at http://www.linaro.org/engineering/engineering-projects/armv8
  - Add micro-architecture description to MI-based scheduler
    - Based on ARM's code generation guidelines
  - Explore sub-word level parallelism (SLP)
    - Improve pattern matching quality for vectorizers.

Acknowledgments

- Chad Rosier (QuIC Inc.)
- Hao Liu (ARM Ltd. Shanghai)
- Kevin Qin (ARM Ltd. Shanghai)
- Tim Northover (Apple)
- Clang and LLVM community reviewers