How to tell compilers to optimize for A64FX on Ookami

Tony Curtis

IACS

Stony Brook University
OUGM: Compilers

• A walk-through of the info in the Ookami FAQ
  – https://www.stonybrook.edu/commcms/ookami/support/faq/Vectorization_Flags

• How to tell compilers to generate a64fx code
  – SVE = Scalar Vector Extensions

• For C, C++, and Fortran
OUGM: Compilers

A64FX 700 architecture

CMG #0

CMG #3

Local disk for OS (SBU addition)

Infiniband
OUGM: Compilers

• N.B. login nodes vs. compute nodes
  – Login nodes are ARM, but ThunderX2
    • Many more cores, much more memory
  – You can compile for a64fx on them, but they cannot run that code

```
login1$ ./a.out
Illegal instruction (core dumped)
```
OUGM: Compilers

- GNU Compilers
  - GCC = GNU Compiler Collection
    - Originally GCC = only GNU C compiler
    - Now also added C++, Fortran, Go, Ada, D, …
  - A64fx vectorization supported from v 10.x
    - Latest release is 11.2.0
OUGM: Compilers

• ARM Compilers
  – C, C++, Fortran
  – Fork from LLVM 12 with …
    • … vendor-added vectorization
    • Optimized math library (ARMPL)
  – ARM are now upstreaming their changes to LLVM
    • In github, guessing will be in release 14.0.0
OUGM: Compilers

- HPE/Cray Compilers
  - Storied HPC compiler chain
    - 2 available
      - 1 with SVE support
      - 1 without (version of LLVM)
    - SVE version has support for a64fx
    - Optimized math library (scilib)

cc, CC, ftn
OUGM: Compilers

- Fujitsu Compilers
  - Vendor of a64fx chip
  - Compiler has long history (SPARC)
  - Strong optimizations for a64fx
    - Tuning environment variables for data layout
  - Scientific Subroutine Library (SSL) math library
  - Also provides MPI implementation
    - Based on Open-MPI 4.0

fcc, FCC, frt

mpifcc, mpiFCC, mpifrt
OUGM: Compilers

- NVIDIA (formerly PGI) Compilers
  - Generally intended for GPU systems
    - Can be used on both x86_64 and aarch64
  - No SVE vectorization at present

nvc, nvc++, nvfortran
OUGM: Compilers

• Summary
  – Matrix of compilers and important options
    • https://www.stonybrook.edu/commcms/ookami/support/faq/Vectorization_Flags
  – Get the compilers to tell you what they are (or are not) doing
  – Quick examples coming up...
Example: GNU

```sh
fj-debug1$ module add gcc/11.2.0
fj-debug1$ gcc -fopenmp -O3 -mcpu=a64fx loop.c -lm
fj-debug1$ objdump -d a.out | grep 'z[0-9]'
```

```
400728: 85c0e004 ld1rd {z4.d}, p0/z, [x0]
400740: a54046c0 ld1w {z0.s}, p1/z, [x22, x0, lsl #2]
400744: a54046e2 ld1w {z2.s}, p1/z, [x23, x0, lsl #2]
400748: 05a06001 zip1 z1.s, z0.s, z0.s
        ...  
400774: 65caa000 fcvt z0.s, p0/m, z0.d
400778: 05a06820 uzp1 z0.s, z1.s, z0.s
40077c: e5404420 st1w {z0.s}, p1, [x1, x0, lsl #2]
```

Quick check for SVE instructions!
OUGM: Compilers

• Example: ARM

```bash
fj-debug1$ module load arm-modules/21.1
fj-debug1$ armclang -fopenmp -O3 -mcpu=a64fx -armp1 loop.c
fj-debug1$ objdump -d a.out | grep 'z[0-9]'
```

```
400918: 05282000 mov    z0.d, d0
400930: a54d4921 ld1w   {z1.s}, p2/z, [x9, x13, lsl #2]
400934: a54d4943 ld1w   {z3.s}, p2/z, [x10, x13, lsl #2]
400938: 05f23822 uunpklo z2.d, z1.s
...
400abc: 65caa042 fcvt   z2.s, p0/m, z2.d
400ac0: 05a16841 uzp1   z1.s, z2.s, z1.s
400ac4: e540e5c1 st1w   {z1.s}, p1, [x14]
```
OUGM: Compilers

• Example: HPE/Cray

```bash
fj-debug1$ module load CPE
fj-debug1$ cc -h omp -h msgs -O3 -h vector3 loop.c
... 
CC-6005 craycc: SCALAR File = loop.c, Line = 35
A loop was unrolled 4 times.

CC-6204 craycc: VECTOR File = loop.c, Line = 35
A loop was vectorized.
```

```bash
fj-debug1$ objdump -d a.out | grep 'z[0-9]' 
400bfc: 05a08000 mov z0.s, p0/m, s0
400c0c: a540a221 ld1w {z1.s}, p0/z, [x17]
...  
400c28: a54f4210 ld1w {z16.s}, p0/z, [x16, x15, lsl #2]
400c2c: 65a10002 fmla z2.s, p0/m, z0.s, z1.s
400c30: 65a30004 fmla z4.s, p0/m, z0.s, z3.s
...  
400c50: e54f4210 st1w {z16.s}, p0, [x16, x15, lsl #2]
```
Example: Fujitsu

```bash
fj-debug1$ module add fujitsu/compiler
fj-debug1$ fcc -Kfast -Kopenmp -KSVE -SSL2BLAMP loop.c
fj-debug1$ objdump -d a.out | grep 'z[0-9]'
```

```bash
40109c: 04d5a231 uxtw z17.d, p0/m, z17.d
...  
4010cc: c574c041 ld1w {z1.d}, p0/z, [x2, z20.d, lsl #2]
...  
4010e8: c577c07d ld1w {z29.d}, p0/z, [x3, z23.d, lsl #2]
4010ec: 05a0395e mov z30.s, w10
4010f0: 05a038e8 mov z8.s, w7
```

Important! [https://www.stonybrook.edu/commcms/ookami/support/faq/ookami-fujitsu-compilers](https://www.stonybrook.edu/commcms/ookami/support/faq/ookami-fujitsu-compilers)
OUGM: Compilers

• Wrap-up
  – Which compiler is right for me?
    • Sadly, no magic bullet
      – We’re all learning as we go…
    • Fujitsu and HPE/Cray often produce good code
      – But do not play well with cmake/autoconf
    • ARM and GCC can also generate good code
      – But play better with cmake/autoconf
OUGM: Compilers

• Wrap-up
  – Got questions?
    • Come to office hours Zoom, and/or Slack channel, and ask!
      – OpenMP
      – MPI
      – Compilation / Configuration
      – Performance
      – Interconnect
      – ...

https://www.stonybrook.edu/commcms/ookami/support/index.php