

Coding Tips for Sun[™] Studio on AMD64 Technology

Sun Studio Compiler Optimization Flags

Most aggressive

```
-fast -xarch=sse3a -m64 -xchip=native -xipo=2
-xprofile=collect/use
```

- › Enables instruction level tuning for AMD Opteron™ processor, O5 level optimizations, sse scalar and vector code generation.
- › Be sure to include `-m64` to generate 64-bit code.
- › Recommended to be used with profile feedback.

Middle ground

```
-xO2 -xarch=sse2a -m64 -xchip=native
```

- › Enables all of the most aggressive except loop reduction, automatic inlining and more aggressive optimization such as scheduling analysis.

Least aggressive

```
-m64 -xchip=native
```

- › No optimizations are performed. No SSE2 instructions are generated.

Sun Studio Compiler Notable Functionality Flags

AMD specific flags

- › `-xarch=sse3a` enables use of SSE3 instructions plus 3DNow!™ technology.
- › `-xtarget=opteron` generates best performance on AMD Opteron processor-based systems. Add `-m64` to generate 64-bit code.
- › `-xtarget=barcelona` enables Third Generation AMD Opteron processor support to take full advantage of the latest processor features.
- › `cc -fast` also gives the right values for `-xarch`, `-xchip`, and `-xcache` for Third Generation AMD Opteron processor support.

Memory model

- › `-xmodel=[a]` where `a=small`, `kernel`, or `medium`
- › Small memory model is used by default if `-xmodel` is not specified.

Parallelism

- › `-xopenmp` parallelizes loops based on openMP pragmas
- › `-xautopar` automatic parallelism, does not accept openMP pragmas

Feedback

- › `-xprofile=collect` collects profile feedback data
- › `-xprofile=use` uses the feedback data

More Sun Studio Tips

- › `lint`: detect potential 64-bit problems, make code 64-bit safe.
- › On Solaris 10, use `-xarch=sse2 -m64` option on code intended to be run on AMD64 platforms to ensure 64-bit code generation.
- › When setting environment variables for dynamic linker programs on the AMD64 architecture, use this code for 32-bit applications:


```
/usr/lib/ld.so.1 -L$(LIB_PATH_32)
```

 and use this code for 64-bit applications:


```
/usr/lib/amd64/ld.so.1
-L$(LIB_PATH_64)
```
- › When repacking a structure for 64-bits, the AMD64 ABI (application binary interface) aligns all types of structures to at least the size of the largest quantity within them.

Learn more at:

http://developer.amd.com/solaris_zone.jsp

AMD 
Smarter Choice