



IBM

Gnu PowerPC support in 2015

Michael Meissner
GCC Compiler group
IBM Linux Technology Center

Gnu Cauldron, August 7-9, 2015

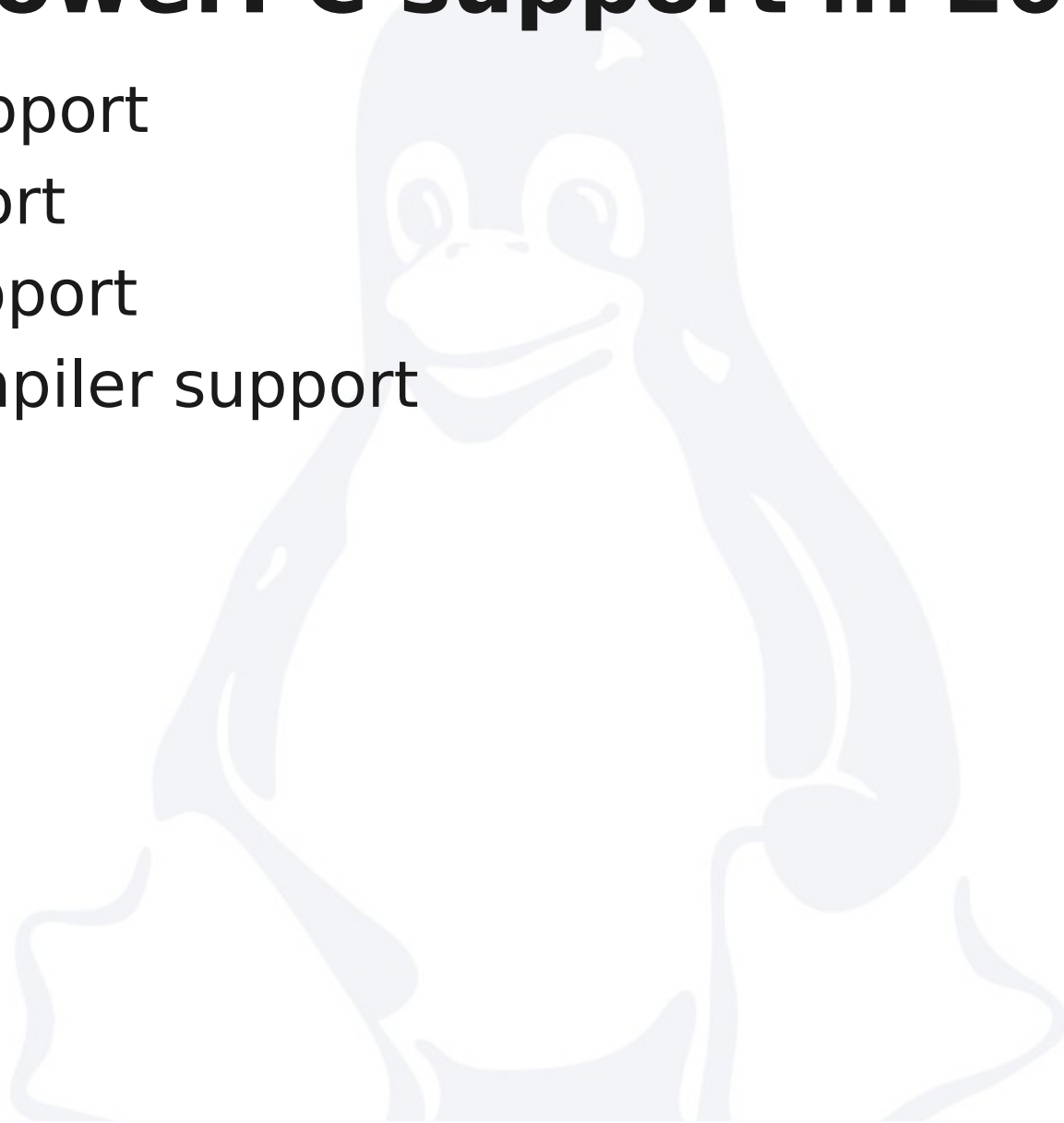
Linux is a registered trademark of Linus Torvalds.

IBM



Gnu PowerPC support in 2015

- LLVM support
- Go support
- Glibc support
- GNU compiler support



LLVM support

- In the last year, LLVM has moved to being fully supported on the PowerPC platform.
- LLVM had two major releases, 3.6 and 3.7.
- The major work has been to add power7 and power8 server support and to add little endian 64-bit support.



LLVM 3.6

- Add VSX (power7) support
- Add power8 scheduling
- AddressSanitizer (ASan) support is now fully functional.
- Performance of simple atomic accesses has been greatly improved.
- Atomic fences now use light-weight syncs where possible, again providing significant performance benefit.



LLVM 3.7 #1

- LLVM 3.7 adds power8 hardware support:
 - Direct moves between GPR & vector registers
 - Hardware transactional memory (HTM)
 - Missing power7 support added
- Code generation for the local-dynamic and global-dynamic thread-local storage models has been improved.
- Loops may be restructured to leverage pre-increment loads and stores.
- Loads from the TOC area are now correctly treated as invariant.
- PowerPC now has support for i128 and v1i128 types. The types differ in how they are passed in registers for the ELFv2 ABI.



LLVM 3.7 #2

- Disassembly will now print shorter mnemonic aliases when available.
- Optional register name prefixes for VSX and QPX registers are now supported in the assembly parser.
- The back end now contains a pass to remove unnecessary vector swaps from POWER8 little-endian code generation.
- The undefined-behavior sanitizer (UBSan) is now supported for PowerPC.
- Vector library improvements in altivec.h.
- Improved -mrecip support.
- PowerPC supports `__builtin_call_with_static_chain`
- More improvements are planned for 3.8.



GO

- There are two implementations of GO, golang and gccgo. Golang does not currently provide full PowerPC/system Z support.
- IBM's interest in a Go compiler began when trying to port Docker to the PowerPC64 bit little endian environment. At that time gccgo had support for PowerPC64 but golang did not.
- Most of the go users on the x86 platform use golang. Gccgo has been bringing more users on other platforms.
- Tools are being added to the gccgo environment to provide the same capabilities as golang.
- 1.4.2 is the current version in trunk and GCC 5.x branches. Go 1.5 is coming.



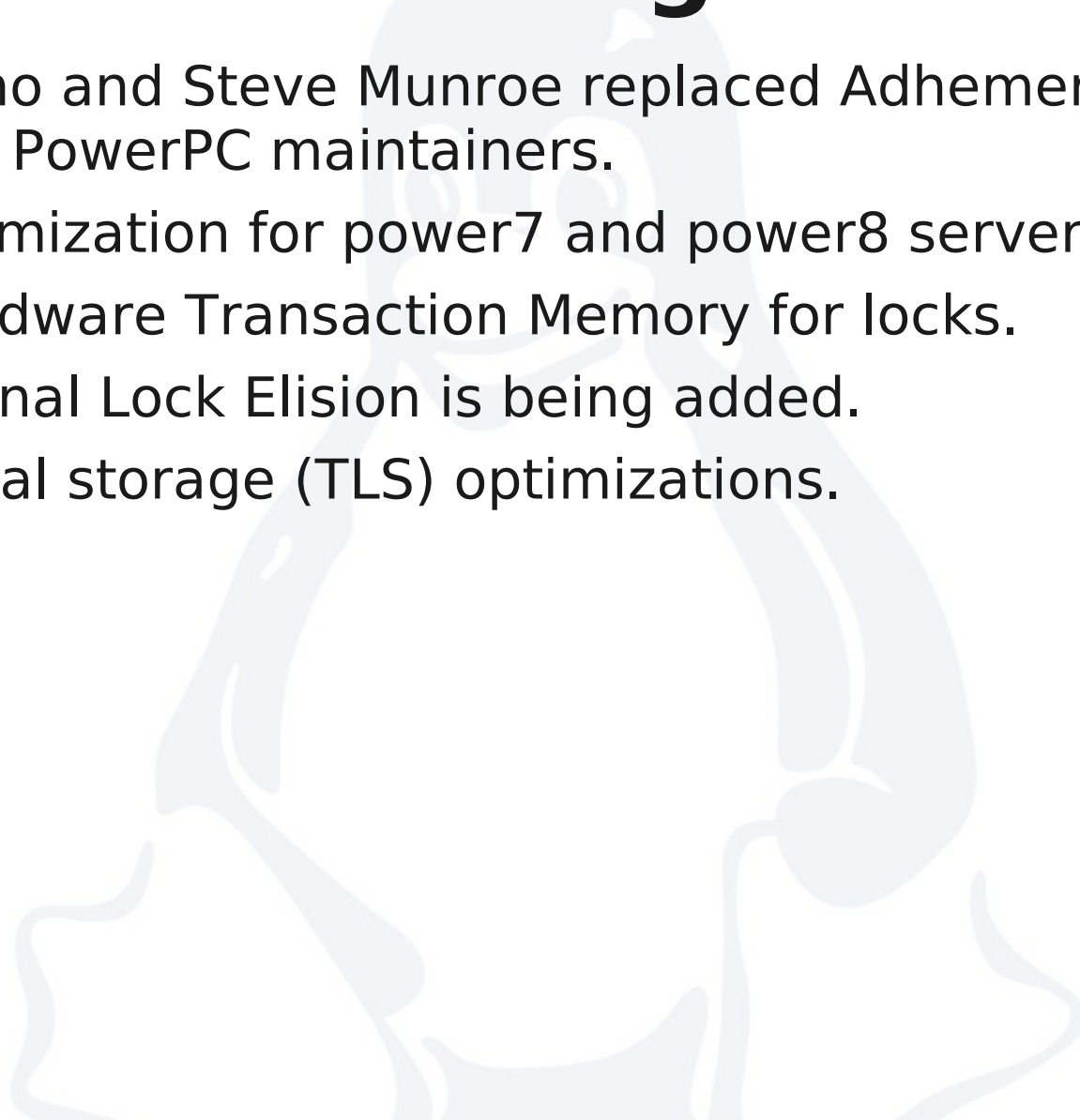
Go #2

- PowerPC specific changes include:
 - Split stack support.
- Language independent changes to gccgo (note work was done by various people in different organizations, not just IBM):
 - Incorporated build of the go tool in gccgo.
 - 'go get' did not work with gccgo.
 - An alternate netgo package is provided.
 - Header file differences were resolved.
 - Added go information to 'go version'.
 - Added escape analysis which is a feature that was always in golang, and helps optimize garbage collection.



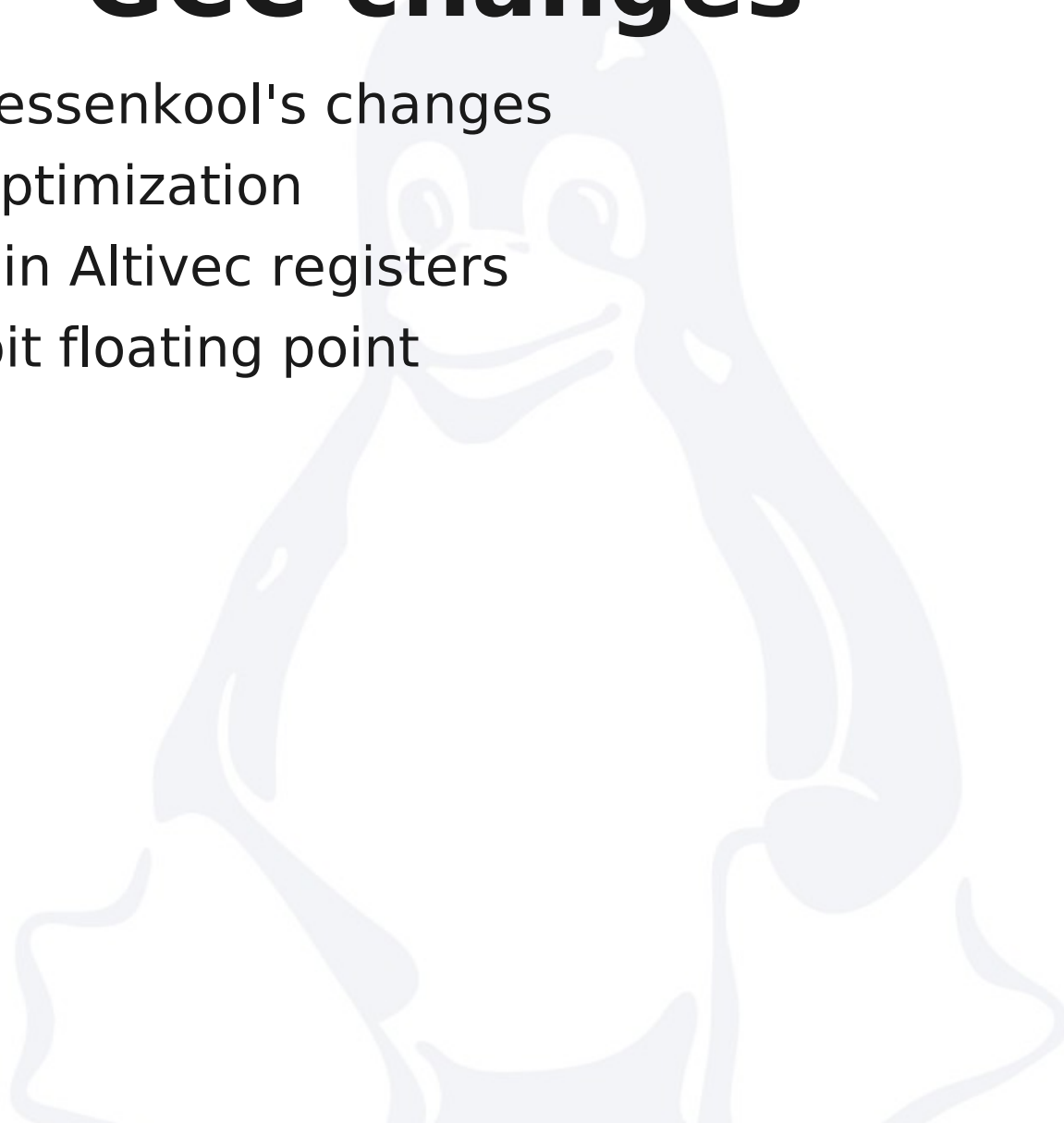
Glibc changes

- Tulio Magno and Steve Munroe replaced Adhemerval Zanella as PowerPC maintainers.
- String optimization for power7 and power8 servers.
- Use of Hardware Transaction Memory for locks.
- Transactional Lock Elision is being added.
- Thread local storage (TLS) optimizations.



GCC changes

- Segher Boessenkool's changes
- PowerPC optimization
- FP scalars in Altivec registers
- IEEE 128-bit floating point
- Fusion



Segher Boessenkool's changes

- Segher Boessenkool made a lot of changes to the PowerPC backend
 - Rework insns that have a record form to set CR registers.
 - Rework shifting, rotating and masking
 - Simplify cpu type attributes
 - Make GCC aware of the CA bit in XER and split multiple instructions for better scheduling
 - Optimize divisions by powers of 2
 - Improve boolean support



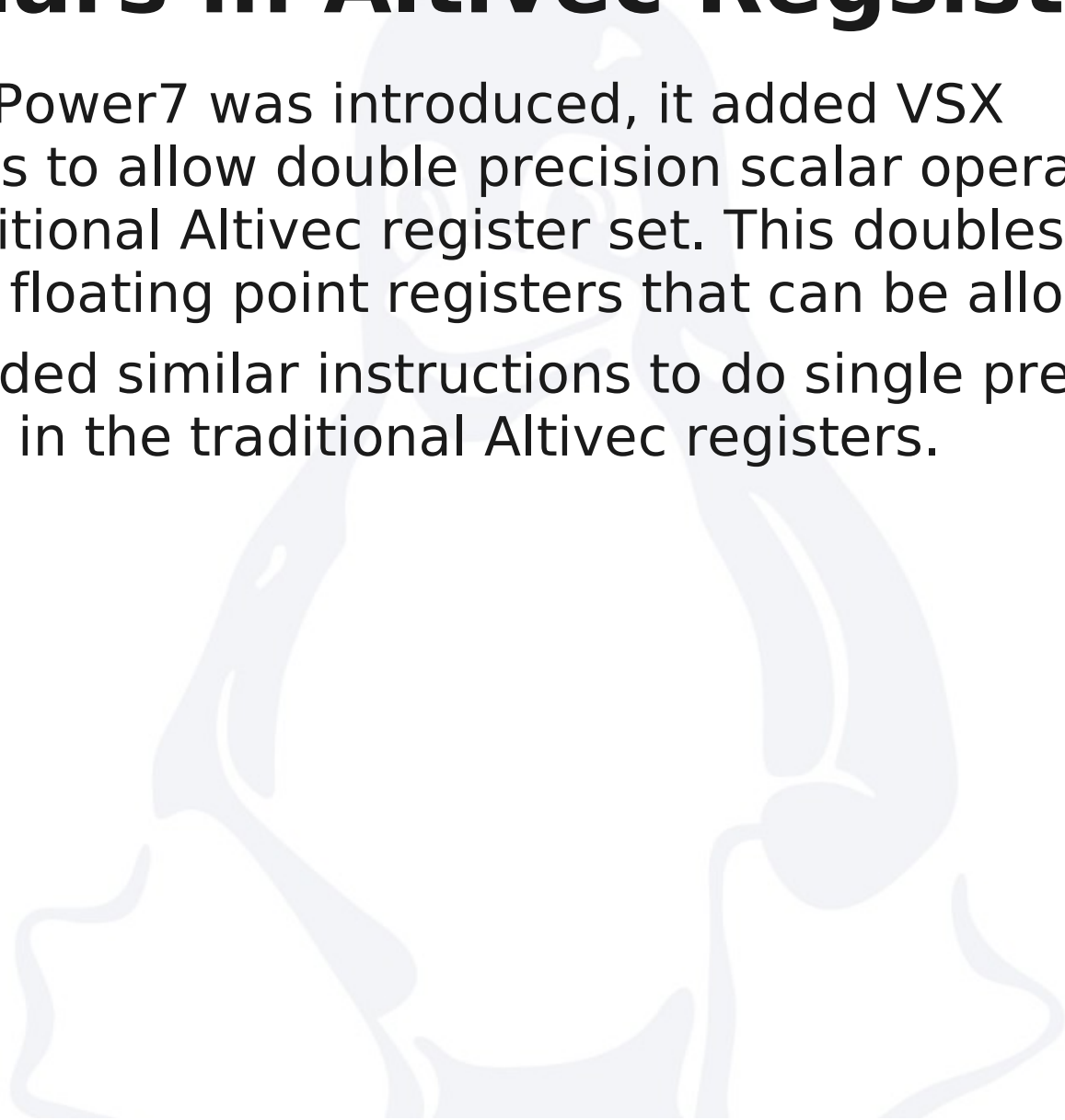
GCC PowerPC optimization

- As IBM optimizes code for the PowerPC, we are starting to use the target attribute and pragma support to change options for a particular function. Several of the PowerPC specific options were missed and are now provided.
- IBM releases the Advance Toolchain to provide customers who need access to newer compilers than the distro provides. We added a new configure switch to support building a compiler using the libraries and dynamic linker from a given Advance toolchain release instead of the system libraries.
- We optimized the bswap built-in function.



FP scalars in Altivec Registers

- When the Power7 was introduced, it added VSX instructions to allow double precision scalar operations in the traditional Altivec register set. This doubles the number of floating point registers that can be allocated.
- Power8 added similar instructions to do single precision operations in the traditional Altivec registers.



FP scalars in Altivec Regs #2

- Traditional FPR registers have either REG+OFFSET or REG+REG addressing, while traditional Altivec registers only support REG+REG addressing.
- GCC legitimate address support assumes that for a given mode, any register uses the same address format until register allocation is done.
- Adding the upper register support meant adding secondary reload support to fixup the addressing mode.
- On power7, we saw 4 benchmarks (zeusmp, namd, calculix, lbm) improve and 1 regress (bwaves).
- On power8, we saw 5 benchmarks (libquantum, gromacs, namd, calculix, lbm) improve and 2 regress (bwaves, cactusADM).



IEEE 128-bit floating point

- As I reported at the last Cauldron, we are adding IEEE 128-bit floating point support to GCC.
- We do not yet plan to switch the default for long double from IBM extended double to IEEE 128-bit floating point.
- Users will use the **`__float128`** type to access IEEE 128-bit floating point and the 'q' suffix on constants to be compatible with x86_64.
- The keyword **`__ibm128`** was added to allow library writers to support existing IBM extended code.
- Work is under way to submit IEEE 128-bit floating point before GCC 6.x freezes. About ½ of the GCC changes are in at this point, with a lot of work left to do on libraries and such.



IEEE 128-bit floating point #2

- The new IEEE 128-bit support requires the VSX instruction set. It is passed and returned like a vector type and not a scalar floating point type.
- The RTEMS PowerPC port never used IBM extended double, and it used IEEE 128-bit support for its long double. These patches should not affect that port.
- Originally you could select on Linux whether IEEE 128-bit floating point were passed in GPRs (ala RTEMS) or via the VSX vector registers (ala the new 64-bit little endian ABI), but this was removed because of support considerations. On Linux, you must use VSX for IEEE 128-bit floating point.
- 64-bit integer and Decimal conversions to/from IBM extended double used TF in some of the function names. This makes it hard to use the standard function names.



IEEE 128-bit FP steps (#3)

- Add new types internally (IFmode for IBM extended mode, KFmode for IEEE 128-bit floating point, use TFmode as a switch hitter).
- Add `__ibm128`, `__float128` types.
- Do the GCC enablement.
- Add temporary support in libgcc to provide KF mode library functions.
- Work with the glibc team to provide official support for the soft-float library and back port to libgcc.
- Work with glibc team to provide more IEEE 128-bit floating point support.
- Work on libquadmath, fixes, etc.



Fusion

- The power8 hardware will recognize adjacent instructions that are used in loads and internally combine these operations together. There may be new forms of fusion in the future.
- Currently, a peephole is used to try and fuse instructions so that they are kept adjacent.
- Work has been done on a branch to move the fusion addressing higher and do it via secondary reload support, rather than using a peephole to spot adjacent instructions.

