Porting OpenVMS using LLVM

vms Software

John Reagan

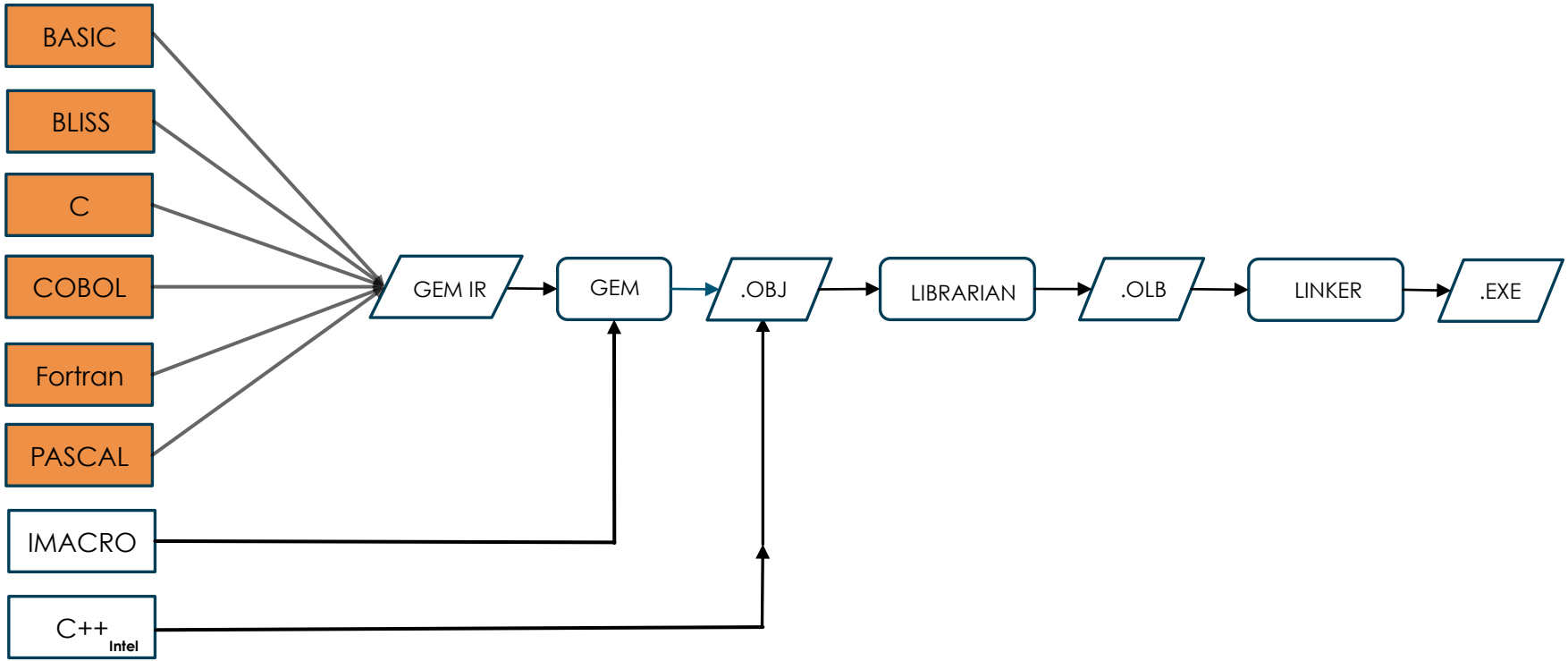
VAX/VMS

- Digital introduced the VAX-11, a 32-bit CISC architecture, 40 years ago in 1977 (before most of you were born)
- Modeled after the PDP-11 with 16 general purpose registers used for integer and floating
- While Berkeley and Bill Joy did 3BSD and beyond, Digital's OS "VAX/VMS" was the main operating system for VAX
- Many compilers on VAX/VMS (Ada, BASIC, BLISS, C, C++, COBOL, Fortran, Pascal) with assorted code-generators
- OS was written in VAX assembly (called Macro-32) and VAX BLISS

OpenVMS Alpha and Itanium

- In 1992, Digital introduced Alpha, a 64-bit RISC architecture
- New single common code generator named GEM with a target independent IR and symbol table
- New Macro-32 Alpha compiler to allow OS to use existing VAX “assembly”
- In 1998/2002, Compaq bought DEC, HP bought Compaq, killing Alpha
- New GEM target for Itanium; switched to ELF/DWARF; switched C++ compilers to Intel/EDG-based compiler; another Macro-32 compiler
- Most Itanium compilers (BASIC, BLISS, C, COBOL, Fortran, Pascal) still use the common GEM backend
- OS now a mixture of BLISS, C, and Macro-32

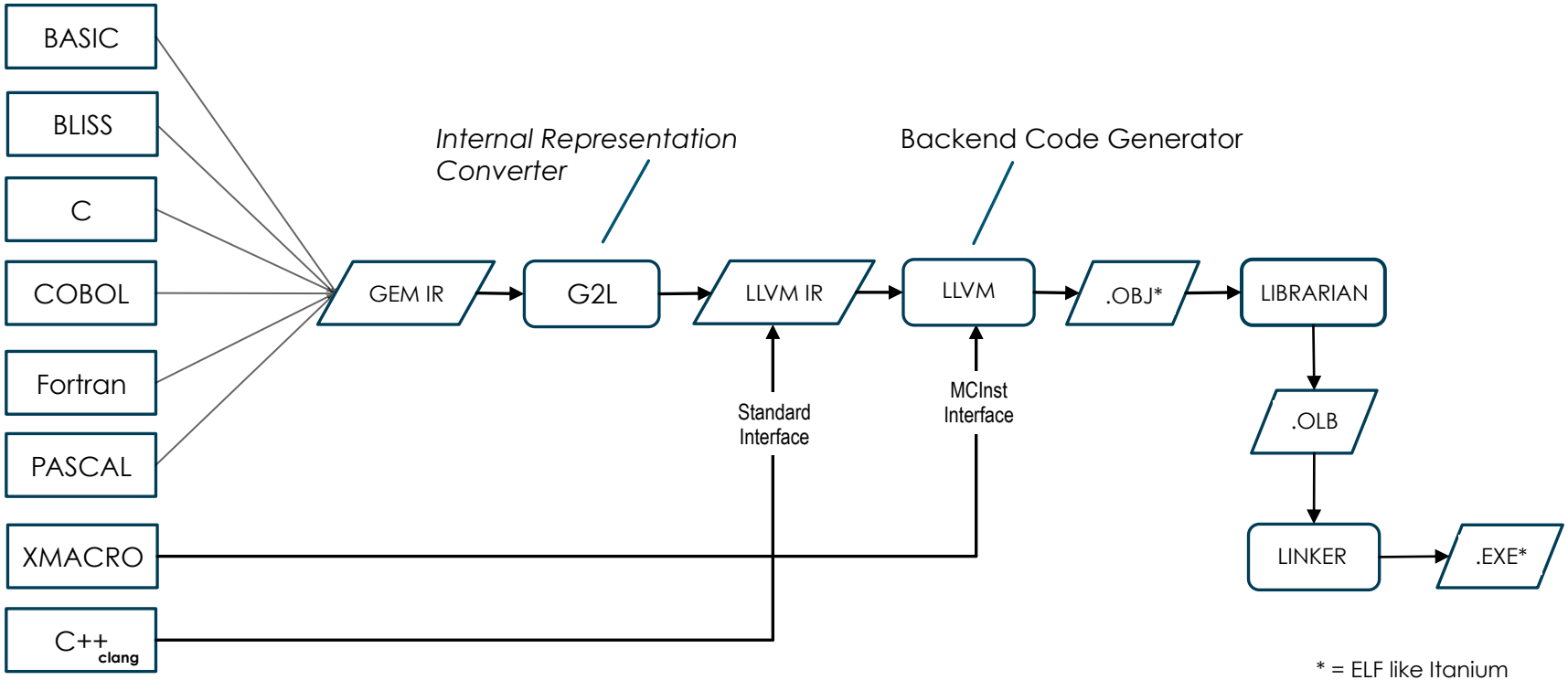
OpenVMS Itanium Compilers



OpenVMS x86-64

- HP licensed OpenVMS to VMS Software Inc in 2014 for future platform support which includes porting to x86-64
- GEM is stale and doesn't know x86-64
- Instead of throwing money at GEM, we picked LLVM to get modern code base and not have to chase all possible chip features
- Still need to provide "recompile and go" for customers
- Leverage existing frontends which generate GEM IR/symtab
- Create a GEM IR to LLVM IR converter (G2L)
- Leverage clang as our C++ offering

OpenVMS x86-64 Compilers



GEM Meets LLVM

- Map GEM IR nodes (~275) to LLVM IR nodes
  - About 240 GEM IR nodes have been mapped sufficient for BLISS and C
  - Remainder used by other frontends and will get mapped as needed
  - GEM has a richer set of primary datatypes so support for things like COBOL packed decimal arithmetic not yet implemented
  - Many GEM nodes have nice simple mappings, but some result in interesting IR sequences or clever converter abstractions (strings for example)
  - Converter currently about 15K lines of C++ (source & headers & comments)
  - Debug info, long double, VAX floating support is TBD

ABI and OpenVMS-isms

- Will use the industry standard AMD64 ABI with a few upward compatible extensions for things like argument count and to assist in performing VMS-style argument list homing
- Will use libc++ and some of libc++ ABI
- Continue to use OpenVMS' debugger and linker
- Continue to use parts of GEM for command line, source file management, etc

LLVM Meets OpenVMS

- A handful of OpenVMS additions to LLVM
  - Mixed pointer size linker relocations
  - Complex LTCE linker relocations
  - “.note” section generation for module name, compilation date/time, etc
  - ABI additions for argument count and arg-info in RAX register
  - Additional DWARF language tags
  - Additional EH unwind descriptors for Macro-32 VAX register emulation
  - Hooks for machine code listings

Clang Meets OpenVMS

- Attempt to be compatible with the C++ compiler on OpenVMS Itanium
- Need to add/modify some behaviors like dual-sized pointers; interface with GEM for command line processing, include file processing, error message generation, listing file generation, etc
- Only in initial investigation stage

Challenges

- Itanium C++ compiler is only C++03 so we're using LLVM 3.4.2 and will have to bootstrap to newer LLVM once on native system with clang in place
- The OpenVMS versions of bash, make, etc are mostly ok for the job but will have to port CMake eventually
- Headers and CRTL needs to become compatible with both clang and our traditional C compiler
- OpenVMS doesn't have "compiler drivers" in the traditional UNIX sense
- Variable length argument list in AMD ABI doesn't mesh well with OpenVMS style which might force some source code changes for customers

Current Status

- Cross C and BLISS compilers are effectively 99% complete with just a few issues with variable length argument lists to iron out
- The cross Macro-32 compiler is a little farther behind. The compiler was designed around mapping VAX CISC instructions to Alpha and Itanium RISC style machines. Getting reasonable x86-64 code has resulted in some deeper changes especially with emulating VAX condition codes
- The OS can currently boot into the initial kernel level debugger with many other components are building and linking
- Our current schedule for OS boot to the shell and issue a directory command is for Q1CY18