Free Software Porting on the Elbrus Architecture

Andrew Savchenko

LVEE 2019
22 – 25 August

LVEE
Linux Vacation / Eastern Europe

1. The Elbrus Hardware
2. Toolchain
3. Typical porting problems
   - Compiler problems
   - Architecture specific issues
   - Alt status
4. Interaction with upstreams

The Elbrus Architecture

Elbrus aka E2000 aka E2K features [1]:

- VLIW: inherent parallelism (6 ALU/core)
- microcode $\Rightarrow$ compiler
- tagged memory (security)
- 3 separate hardware stacks
- x86/amd64 JIT compiler
- asynchronous array prefetch

4th generation features [2]

- 25 general instructions per cycle
- 1000 – 1300 MHz
- 1 and 8 core CPU
- 250 GFlops per CPU (8 core)
- inter-core link (up to 4 CPUs)
- 60 – 80 W/CPU
- 64 GB/CPU DDR3-1600

The E2K CPU [3]

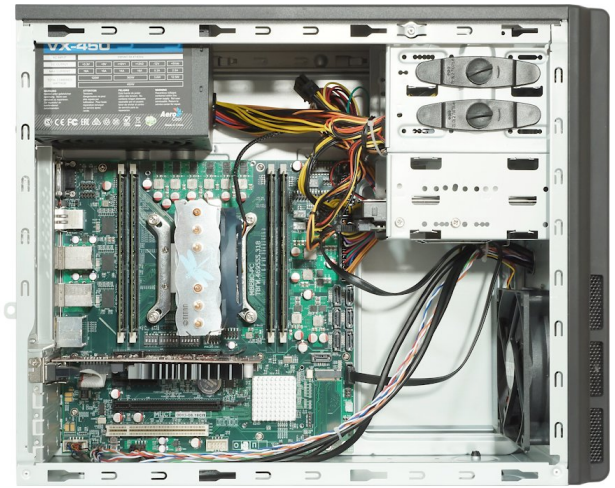
- 600 – 1000 MHz
- MGA2 (MCST)
- Vivante (OpenGL 2.1, OpenCL 1.2)
- $0.375 \cdot 10^9$ tr.

- 1000 – 1300 MHz
- VLIW 25
- DDR3-1600
- $2.73 \cdot 10^9$ tr.

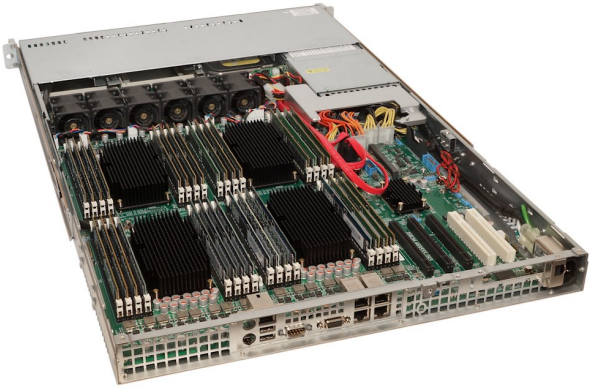
- 1500 MHz
- VLIW 50
- DDR4-2400
- $3.5 \cdot 10^9$ tr.

The E2K Devices [4]

- Elbrus 101-PC
- E1C+ CPU + MGA2/Vivante
- 1000 MHz + 800 MHz
- 16 GB RAM



- Elbrus 801-PC
- E8C CPU
- 1200 (1300) MHz
- 32 GB RAM



- Elbrus 804 1U
- 4x E8C CPU
- 4x 1200 MHz
- 256 GB RAM

The ELbrus Compiler (LCC)

- Production lcc-1.23 $\sim$ gcc-5.5
- EDG (Edisson Group) Frontend
- MCST multilayer backend

Main supported standards:

- C11
- **C++11**, partially C++14
- Fortran-2008
- OpenMP-2.5 [year 2005! :/]

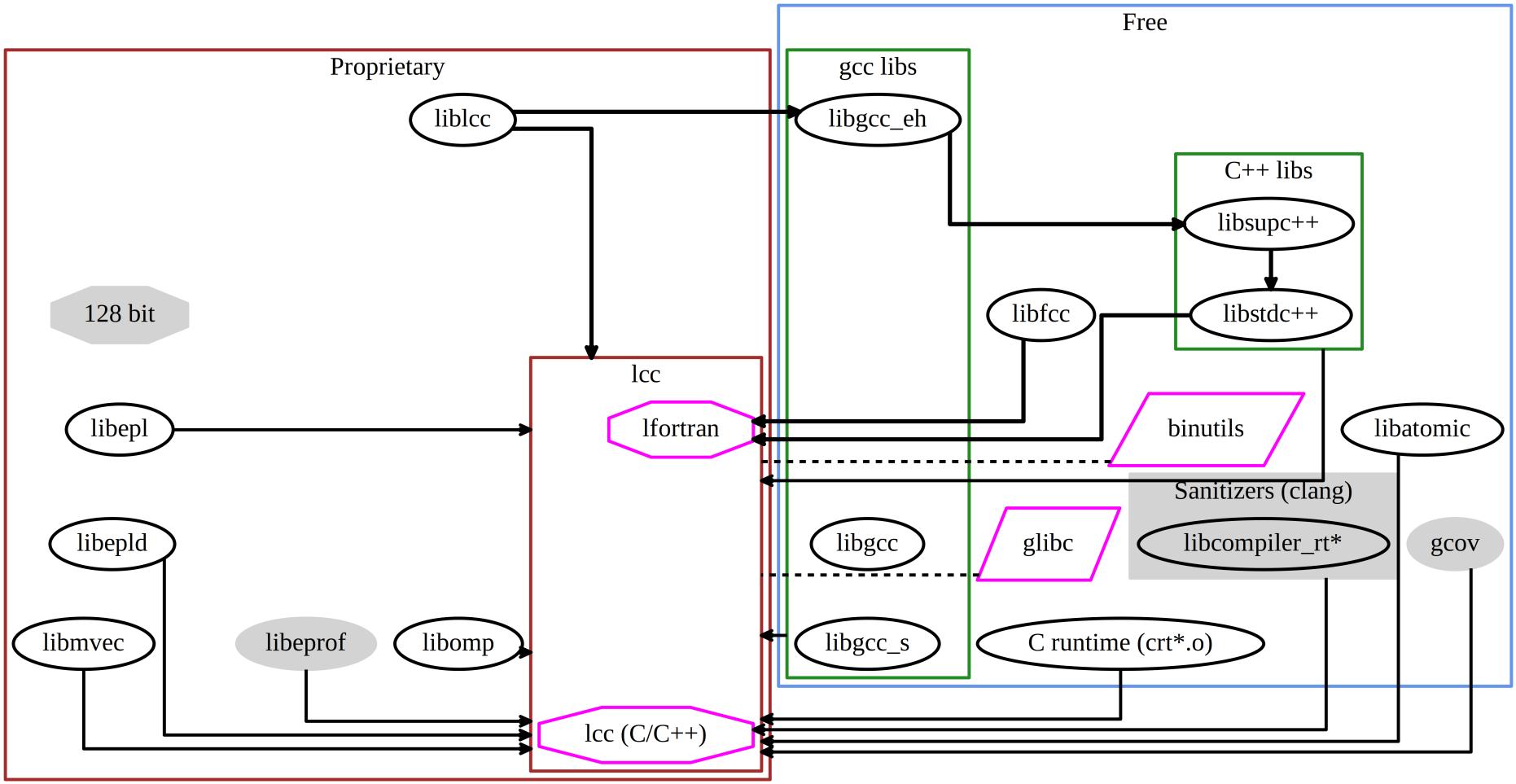
The ELbrus Compiler (LCC)

- Production lcc-1.23 $\sim$ gcc-5.5
- EDG (Edisson Group) Frontend
- MCST multilayer backend

Main supported standards:

- C11
- **C++11**, partially C++14
- Fortran-2008
- OpenMP-2.5 [year 2005! :/]

Toolchain


Free
Proprietary
gcc libs
liblcc
libgcc_eh
C++ libs
libsupc++
128 bit
libfcc
libstdc++
lcc
libepl
lfortran
binutils
libatomic
Sanitizers (clang)
libepld
libgcc
glibc
libcompiler_rt*
gcov
libmvec
libeprof
libomp
libgcc_s
C runtime (crt*.o)
lcc (C/C++)

Compiler features

lcc often detects problems missed by gcc:

- especially for warnings
- we have to disable -Werror

Preprocessor assumes that input is C/C++:

- indentation tabs are being replaced by spaces
- Makefile, etc processing is broken
- fixed by using -xassembler or changing preprocessor handling logic.

Compiler features

Icc often detects problems missed by gcc:

- especially for warnings
- we have to disable -Werror

Preprocessor assumes that input is C/C++:

- indentation tabs are being replaced by spaces
- Makefile, etc processing is broken
- fixed by using -xassembler or changing preprocessor handling logic.

Compiler features

- Nested functions are not supported.
- -O2 $\Rightarrow$ -O3
- Debugging:
  - -O0
  - -g0
  - -fcontrol-spec

Compiler specific issues [5]

Not all gcc-5.5 data types are supported:

- `__(u)int128_t` — will be in lcc-1.24
- `_Decimal64`

Not all gcc-5.5 builtins are present (and with the same behaviour):

- `__builtin_*_overflow` — will be in lcc-1.24
- `__builtin_*_overflow_p`
- `__bultin_constant_p` — in corner cases behaviour is different from gcc
- `__builtin_shuffle`
- ...

Compiler specific issues [5]

Not all gcc-5.5 data types are supported:

- `__(u)int128_t` — will be in lcc-1.24
- `_Decimal64`

Not all gcc-5.5 builtins are present (and with the same behaviour):

- `__builtin_*_overflow` — will be in lcc-1.24
- `__builtin_*_overflow_p`
- `__bultin_constant_p` — in corner cases behaviour is different from gcc
- `__builtin_shuffle`
- ...

Undocumented gcc: VLAIS

- Variable length arrays (VLA, C99) are supported:

```
void f (int n) {
  int a[n];
}
```

- Variable length arrays in structures (VLAIS) are not supported:

```
#define assert_cc(expr)                                    \
  DISABLE_WARNING_DECLARATION_AFTER_STATEMENT;              \
  struct CONCATENATE(_assert_struct_, __COUNTER__) {  \
          char x[(expr) ? 0 : -1];                         \
  };                                                       \
  REENABLE_WARNING
```

C++ as C

- Code is written in C++
- but without C++ runtime (-nostdlib -fno-rtti ...)
- gcc can link this as C, but not lcc:

```
/usr/lib64/libharfbuzz.so.0: undefined
      reference to ‘__cxa_vec_dtor’
```

- Workaround: link with C++ subset:

```
LIBS+=”-lsupc++ -lgcc_eh -llcc”
```

UTF-8

- Default output is KOI8-R
  - but respects LANG (en, ru)
- *Some* output is still in KOI8-R
  - some software (like python) chokes on parsing
- Preprocessor can’t parse UTF-8 BOM (byte order mark)
  - strip it
- UTF-8 literals:

```
lcc: ”static_unicode_sets.h”, строка 111: ошибка:
     слишком много символов в символьной константе
       {RUPEE_SIGN, u’₨’},
                    ^
```

  - -finput-charset=utf8

Compiler's future

Less common languages are not supported:

- Go
- Rust
- Haskell

Possible solutions:

- LLVM backend
- in progress...
- problems with "non-standard compiler" should be mitigated

Free compiler?

Architecture specific features

- Arch-specific syscall table
- Arch-specific ioctl table
- Different stacks:
  - Separate stacks for data, functions, function arguments.
  - Special stack handling for garbage collectors, etc.
  - Arch-specific context switch: makecontext_e2k(), freecontext_e2k(), swapcontext_e2k().

Alt (Sisyphus) on E2K

Total packages: ~ 11500

Core components:

lcc	1.23	$\sim$ gcc-5.5
Linux kernel	4.9	
glibc	2.23	
gdb	8.1	
strace	4.20	
gtk	2.24	3.24
qt	3 / 4	5.9
boost	1.67	
perl	5.28	
python	2.7	3.7
firefox (+jit)	52.9	
libreoffice	5.4.3.2	

Upstream integration with MCST

- $\Leftarrow$ New versions
- $\Leftrightarrow$ Patches exchange
- $\Rightarrow$ Bug reports

Complexity overview:

- Hundreds of packages patched for e2k
- Kernel 4.9.170, diff 50 MB:

  ```
  3296 files changed, 1463488 insertions(+),
  4390 deletions(-)
  ```

Free software upstreaming

- We can upstream only own patches
  - Sometimes we have mixed patchsets and can’t upstream them
- Upstreams are different:
  - some accept gladly
  - some ignore
  - some says “give me binutils/toolchain sources first”
- Share of patches fix problems for all architectures detected on e2k

Where e2k is contributed [6]

- binutils
- bzflag
- gimagereader
- glew
- gnu-config
- gstreamer
- imake
- libtommath
- lxc
- meson
- netsurf
- nss
- pcre
- ruby
- sanlock
- spice-protocol
- transmission
- util-linux

Summary

- Elbrus is self sufficient for a full scale linux distribution
- E2K upstreaming is not easy, but possible
- Awaiting for going public
- Write standard code!

Thank you for your attention!

Meanwhile in Soviet Russia

CPUID on Intel:

```
#include <cpuid.h>
#include <stdio.h>

int main (void)
{
  int a[13];
  __cpuid (0x80000002, a[0], a[1], a[2], a[3]);
  __cpuid (0x80000003, a[4], a[5], a[6], a[7]);
  __cpuid (0x80000004, a[8], a[9], a[10], a[11]);
  a[12]=0;
  puts((char*)a);
  return 0;
}
```

Meanwhile in Soviet Russia

CPUID on Elbrus (simplified):

```
char const * const getcpu(void)
{
    if (__builtin_cpu_is(”elbrus”))
        return ”elbrus-v1”;
    else if (__builtin_cpu_is(”elbrus-2c+”))
        return ”elbrus-2c+”;
    else if (__builtin_cpu_is(”elbrus-4c”))
        return ”elbrus-4c”;
    else if (__builtin_cpu_is(”elbrus-8c”))
        return ”elbrus-8c”;
    else if (__builtin_cpu_is(”elbrus-1c+”))
        return ”elbrus-1c+”;
    else if (__builtin_cpu_is(”elbrus-8c2”))
        return ”elbrus-8c2”;
    else
        return ”unknown”;
}
```

Bibliography I

- Александр Киирович Ким, Валерий Иванович Перекатов, Сергей Геннадьевич Ермаков. Микропроцессоры и вычислительные комплексы семейства Эльбрус. — Санкт-Петербург : Издательство «Питер», 2013. — ISBN: 978-5-459-01697-0. — http://www.mcst.ru/files/511cea/886487/1a8f40/000000/book_elbrus.pdf.
- Эльбрус. Российские процессоры и вычислительные системы. — http://elbrus.ru.
- МЦСТ Эльбрус. Микропроцессоры и СБИС. — http://mcst.ru/chips.
- ИНЭУМ Встраиваемые системы. Вычислительные машины. — http://ineum.ru/computers.
- Alt wiki: Эльбрус/lcc. — https://www.altlinux.org/Эльбрус/lcc.
- Alt wiki: Эльбрус/upstream. — https://www.altlinux.org/Эльбрус/upstream.

LVEE
Linux Vacation / Eastern Europe