Retargetable Binary Utilities

Design Automation Conference

Maghsoud Abbaspour, Jianwen Zhu

Electrical and Computer Engineering
University of Toronto

June 12th, 2002

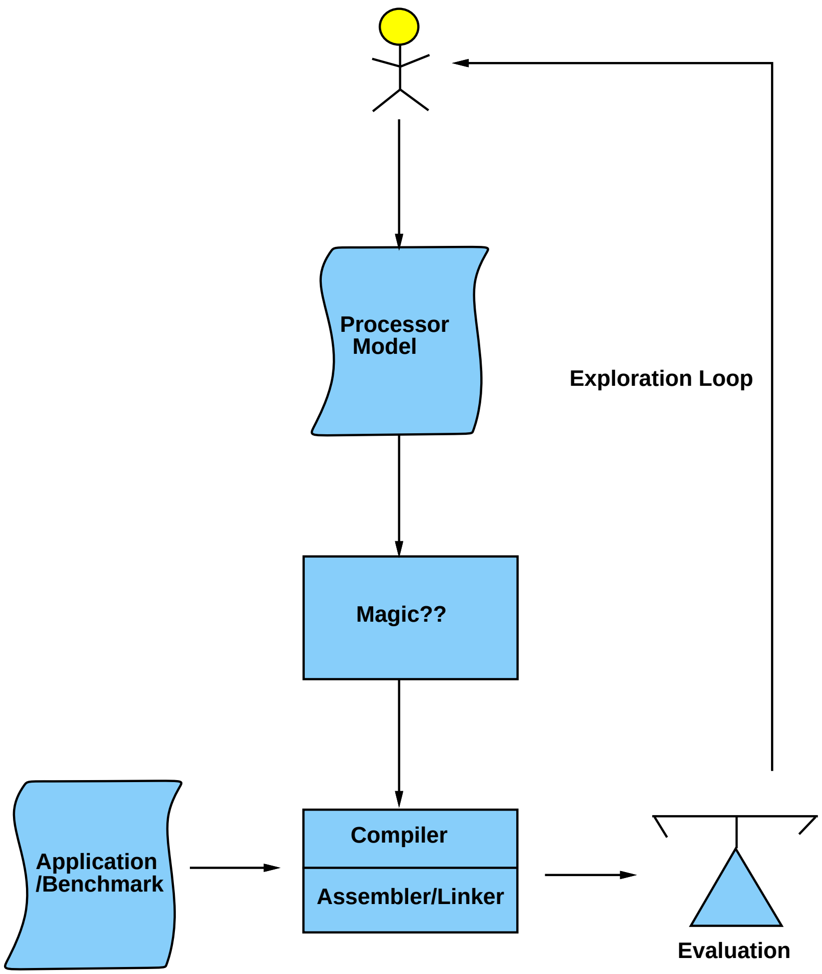
`jzhu@eecg.toronto.edu`
`http://www.eecg.toronto.edu/~jzhu`

Outline

- *Background*
- Architecture Modeling and Description
- Implementation and Results
- Conclusions

Problem Definition

- System-On-Chip dominated by software
- Is software really cheap?
- Yes if development tool readily available
- Software development suite
  - Compiler
  - Backend tools: assembler/linker etc.
  - Simulator
  - Debugger



Wasn't it a Solved Problem in 1960s?

- Considered trivial except perhaps a few black magic tricks
  - Virtually no paper published on linker/assembler construction
  - No treatment in CS/CE curriculum except lab instructions
- Not true anymore for modern computer system
- Linker example
  - Driven by OS: shared library/dynamic linking
  - Driven by Language (C++): constructor/destructor, template, inline
  - Driven by compiler technology: whole program analysis/optimization

Isn’t It Just Engineering?

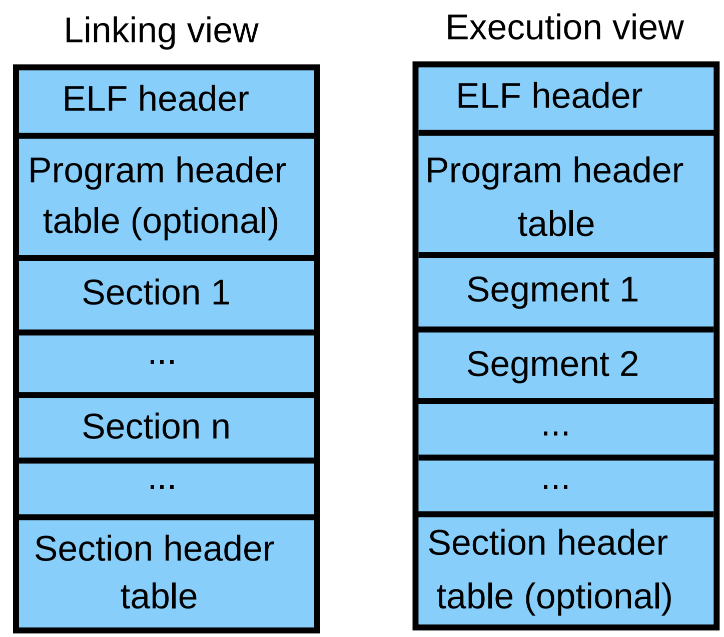
- Why can’t I manually port the “portable” GNU tools?
- Why can’t I hack legacy tool from Company XYZ?
  - GNU’s binutils has 250K lines of code!
  - John Levine: Backend gurus can be “packed in one single room”.
  - Target-dependent interface not cleanly defined
  - Target-dependent implementations scattered in many C files
  - Target-dependent information overlaps with what needed for other tools like compiler/simulator/debuggers.
- Need “automatic hacking”

Wasn’t it a Solved Problem in 1990s?

- Yes! Related work:
  - Code generator generators: IBURG etc
  - Compiler community: Zephyr, MDES
  - CAD community: MIMOLA, CHESS, EXPRESSION, FLEXWARE, LISA, ISDL, MESCAL
  - Commercial ARM, ARC, Tensilica etc
- An No! Our contribution:
  - Formal model of ISA: independent of ADL
  - Application Binary Interface (ABI)
  - Open-source, production quality package

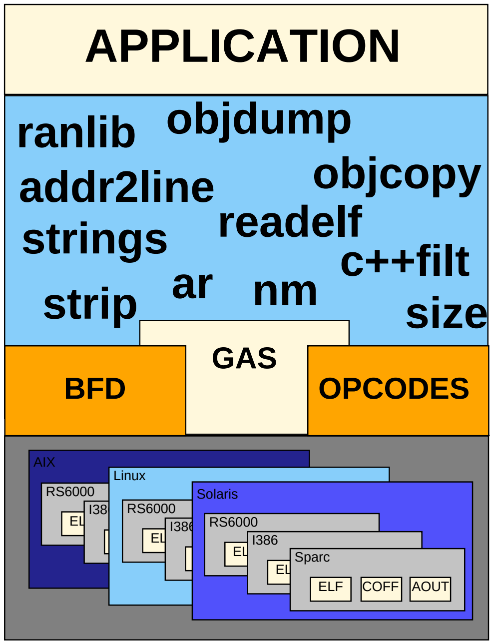
Background: Anatomy of Object Files

- Three types:
  - Relocatable object files
  - Executable files
  - Shared object files
- Many legacy formats
  - a.out
  - Common Object File Format (COFF)
  - Portable Executable (PT)
  - (ELF)



Background: Anatomy of GNU's binutils Package

- Binary File Descriptor (BFD) Library: libbfd
  - Relocatable object files
  - Executable files
  - Shared object files
- opcodes Library
- gas
- Target-independent components



Outline

- Background
- *Architecture Modeling and Description*
- Implementation and Results
- Conclusions

Definitions

- Architecture Modeling:

  Defining an abstraction (model) of the target-dependent information as well as their relationship

  - Instruction Set Architecture (ISA)
  - Application Binary Interface (ABI)
  - Micro-architecture

- Architecture Description Language

  A language that can help capture the architecture model in a human-friendly form.

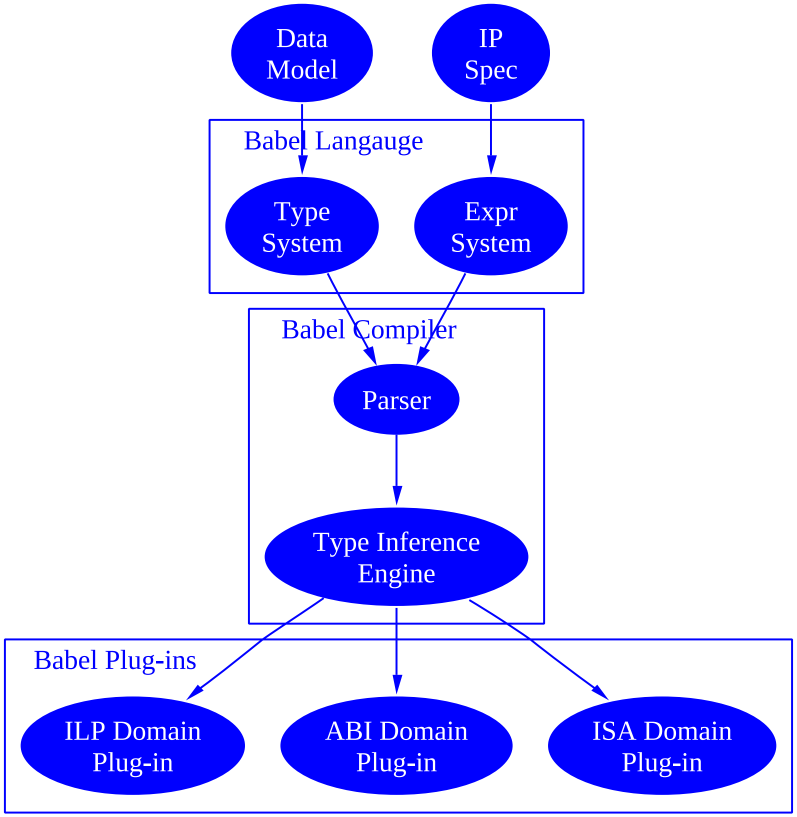
- Architecture Description

Babel Specification Language

- A general-purpose IP specification language
  - Frontend to the IPSUITE infrastructure
  - Captures arbitrary graph of data
- Help capture multiple "facets" of an IP
  - Behavioral facet
  - Architectural: ISA, ABI, ILP facets
  - Physical: Plan, Mask facets

Babel Specification Language Architecture

- Extensible: Type system to define domain-specific data model
- Specification are checked using type inference engine
- Domain-specific plug-ins to compile spec into IP



Specifying Behavior

- The need
  - Instruction selection
  - Simulator generation
  - Dynamic linking
  - Calling convention specification
- The form
  - Field: logical registers
  - Method: logical instructions

- SPARC behavioral specification

```
facet Sparc::beh {
  unsigned[32] g0, g1, g2, g3, g4, g5, g6, g7;
  unsigned[32] l0, l1, l2, l3, l4, l5, l6, l7;
  unsigned[32] i0, i1, i2, i3, i4, i5, i6, i7;
  unsigned[32] o0, o1, o2, o3, o4, o5, o6, o7;

  unsigned[32] fp, sp;

  ...
  byte lodb( byte* src1, int offset ) {
      return *(src1+offset);
      }

  short lods( short* src1, int offset ) {
      return *(src1+offset);
      }
  ...
  }
```

Specifying ISA and ABI

- Data Model:

```
class domain.ISA {
    {}Store             stores;
    {}CellGroup         groups;
    {}Field             fields;
    {}Format            formats;
    {}Instrn            instrns;
    {}string            properties;
    }

class domain.ABI {
    // ... register convention
    // ... argument passing scheme
    // ... frame layout
    // ... view change
    {}Reloc relocs;
    method plt;
    }
```

- SPARC ISA and ABI facets

```
facet Sparc::isa = new domain.ISA {
    stores = { ... }
    fields = { ... }
    formats = { ... }
    instrns = { ... }
    properties = { ... }
    }


facet Sparc:abi = new domain.ABI {
    ...
    relocs = { ... }
    ...
    }
```

Specifying Storages

- Size and granularity
- Register window
- Mapping to logical registers
- Data modeling

```
typedef int*int     Cell;

typedef []field    CellGroup;

class Store {
    Store( int gran, int size );

    int         depth;
    int         overlap;
    field       pointer;
    {}CellGroup cells;
    }
```

- SPARC storage specification

```
stores = {
    sGPR = new Store( 32, 32 ) {
      depth = 128;
      overlap = 24;
      pointer = cwp;
      maps = {
          gpr = [
              g0, g1, g2, g3, g4, g5, g6, g7,
              o0, o1, o2, o3, o4, o5, o6, o7,
              l0, l1, l2, l3, l4, l5, l6, l7,
              i0, i1, i2, i3, i4, i5, i6, i7
              ]
          alias = [
              x, x, x, x, x, x, x, x,
              x, x, x, x, x, x, sp, x,
              x, x, x, x, x, x, x, x,
              x, x, x, x, x, x, fp, x
              ]
          }
      }
    ...
    }
```

Specifying Instruction Format

- Instruction format:
  - Opcode field
  - Register field
  - Immediate/Relocatable filed
- Data model

```
class Field {
    Field( int size );  // opcode field
    Field( // immediate field
        int size, boolean isSigned, int argno
        );
    Field( // register field
        int size, boolean isDest,
        int argno, CellGroup group
        );
    Field( // relocatable field
        int size, boolean isSigned,
        int argno, int reloctype
        );
    }
typedef []Field     Format;
```

- SPARC instruction format

```
fields = {
  op  = new Field( 2 );
  op2 = new Field( 3 );
  rdg = new Field( 5, true, 0, gpr );
  rdf = new Field( 5, true, 0, fpr );
  rdd = new Field( 5, true, 0, dpr );
  rdfsr = new Field( 5, true, 0, fsr );
  a = new Field( 1 );
  cond = new Field( 4 );
  imm22 = new Field( 22, false, 1, 9 );
  disp22 = new Field( 22, true, 1, 8 );
  ...
  }
formats = {
  F3A    = [op,rdg,op3,rs1g,i,asi,rs2g],
  F3B    = [op,rdg,op3,rs1g,i,simm13],
  F3AF   = [op,rdf,op3,rs1g,i,unused8,rs2g],
  F3BF   = [op,rdf,op3,rs1g,i,simm13],
  F3AD   = [op,rdd,op3,rs1g,i,unused8,rs2g],
  F3BD   = [op,rdd,op3,rs1g,i,simm13],
  F3AFSR = [op,rdfsr,op3,rs1g,i,unused8,rs2g],
  F3BFSR = [op,rdfsr,op3,rs1g,i,simm13],
  ...
  }
```

Specifying Instructions

- Assembly format
  - Instruction emitting
  - Instruction assembling
  - Instruction de-assembling
- Binary format
- Opcodes
- Patterns

```
class Instrn {
    string      asmFormat;
    Format      binFormat;
    []int       opcodes;
    {}method    patterns;
    }
```

- Sparc instructions

```
instrns = {
    typedef int[0:(1<<13-1)]    int13;

    ldsb = new Instrn {
        asmFormat = "ldsb [%1],%0; [%1+%2],%0":
        asmFormat = F3A;
        opcode = [0x3, 0x9, 0x0, 0x0];
        patterns = {
            byte.OP_LOD(pointer),
            Sparc.lodb( point, int )
            }
        }
    andi = new Instrn {
        asmFormat = "and #1,%2,%0; %2,#1,%0";
        binFormat = F3B;
        opcode = [0x2, 0x1, 0x1];
        patterns = {
            uint.OP_BAND(uint, uint13)
            }
        }
    ...
    }
```

Specifying Relocations

- Relocation type: expression applied on information kept in
  - Relocation field
  - Relocation entry
  - Linkers: base, GOT, GOT offset, PLT offset, place of instruction
- Expression encoding

−GOT	+A	+B	+G	+GOT	+L	−P	+S	
0	1	0	0	0	0	1	1	=0x43

==> S+A−P

- Sparc instructions

```
enum ABIComplainKind {
    KIND_ABI_IGNORE = 0,
    KIND_ABI_BIT = 1,
    KIND_ABI_SIGN = 2,
    KIND_ABI_UNSIGNED = 3
    }
class Reloc {
  Reloc(
    int id, int exprCode,
    int rightShift, int bitSize,
    int bitPos, int complain
    );
  }
```

```
relocs = {
    r_sparc_none = new Reloc(
        0, 0x00, 0, 0, 0, KIND_ABI_IGNORE );
    r_sparc_8 = new Reloc(
        1, 0x41, 0, 8, 0, KIND_ABI_BIT );
    r_sparc_wdisp30 = new Reloc(
        7, 0x43, 2, 30, 0, KIND_ABI_SIGN );
    ...
    }
```

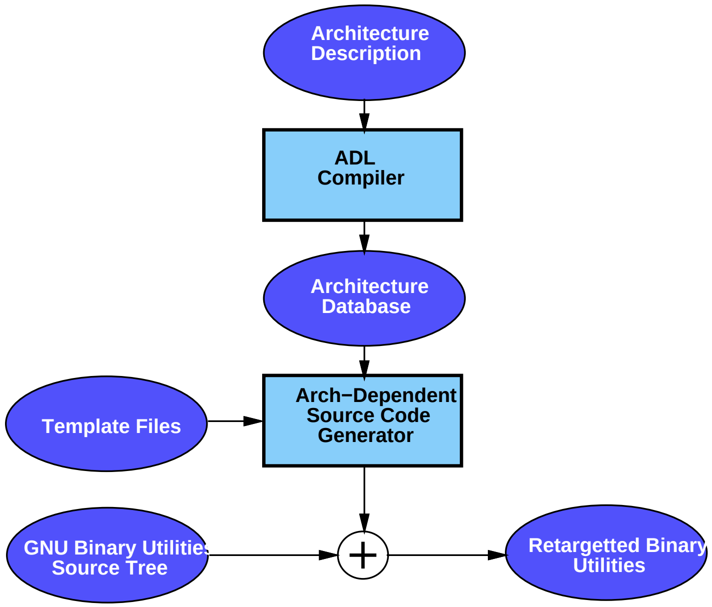
Outline

- Background
- Architecture Modeling and Description
- *Implementation and Results*
- Conclusions

Automatic Porting of Binary Utilities

- Bridge the gap between
  - Our architecture model
  - GNU's porting interface
- Template files
  - C, header, configuration, Makefile
  - Placeholder for substitution
- No change to GNU binutils distribution

- Retargeting binutils



More on Implementation

- Retargeting BFD library
  - Type definitions
  - Data generation: internal representation of relocation method and PLT
  - Function generation: relocation checking, relocator, dynamic section generators
- Retargeting opcodes library
  - Internal representation of instructions
  - Instructions in hash table keyed by mnemonics
  - Instruction disassembling
- Retargeting gas
  - Generates line parser
  - Mnemonic matching, pattern matching
  - Symbol, immediate, register field collection
  - Instruction encoding
  - Relocation generation
- Retargeting other tools
  - Depends only on BFD and opcodes library to generate line parser
  - Only configuration changes

Implementation Statistics

- Our Tools

Component	#lines
Babel compiler	3538
ISA plug-in	2124
ABI plug-in	1474
rbinutils	2785
ISA/ABI data model	207

- Processor specification

Processor	Beh (#lines)	ISA (#lines)	ABI (#lines)
SPARC	196	2300	56
SimpleScalar	90	1048	45
Alpha	1144	4511	52
I386	1234	20406	44

Result: Retargeting SPARC

- Generated BFD code

File	#line
bfd/elf32-sparc.c	1899
include/elf/sparc.h	114
include/elf/common.h	641
bfd/archures.c	1088
bfd/config.bfd	1093
configure.in	838
bfd/cpu-sparc.c	78
bfd/targets.c	1243
bfd/bfd-in2.h	3918
config.sub	1449
Total	12361

- Generated ld code

File	#line
ld/Makefile.am	985
ld/emulparams/elf32_sparc.sh	12
Total	997

- Generated opcodes code

File	#line
include/opcode/sparc.h	116
opcodes/sparc-opc.c	2128
opcodes/Makefile.am	658
opcodes/configure-in	283
Total	3185

- Generated gas code

File	#line
gas/config/sparc-tmp.h	521
gas/config/tc-sparc.c	2436
gas/config/tc-sparc.h	205
gas/configure.in	984
gas/Makefile.am	2280
Total	6424

- Verification: SPEC2000

Conclusion

- ISA model should be made formal
- ABI model should not be ignored
- Use typed, generic language as ADL
- Leverage open-source, production quality software
- Future work
  - Exploit polymorphism in specification to reduce specification size
  - Use behavioral specification for relocation calculation
  - Automatic simulator generation
  - Automatic porting of GDB