GIMPLE Tuples

Aldy Hernandez <aldyh@redhat.com>
Diego Novillo <dnovillo@google.com>

Outline

- What?
- Why?
- How?

Why? trees just work

- Increased separation between FE and ME
- More clearly defined hand-off sites
- More chances to reduce memory footprint (FE left overs)
- Memory reduction in the representation of statements
- Faster compile times
- Faster streaming for LTO purposes (less pickling/unpickling)

Data structures

Field	Size (bits)
code	16
subcode	8
no_warning	1
visited	1
nontemporal_move	1
plf	2
modifed	1
has_volatile_ops	1
references_memory_p	1
uid	32
location	32
num_ops	32
bb	64
block	64
Total size	32 bytes

Data structures

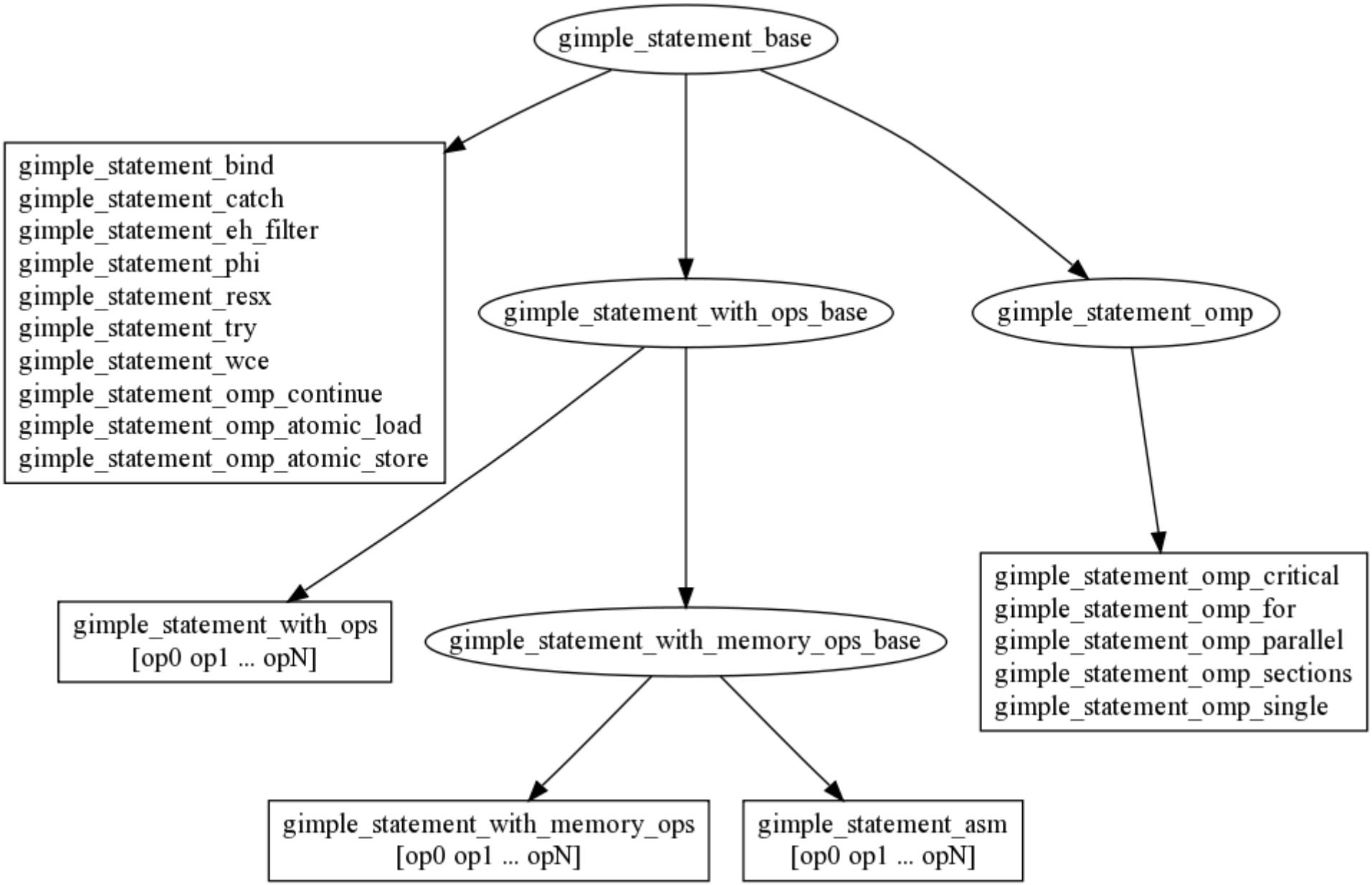
Operands

Field	Size (bits)
gsbase	256
addresses_taken	64
def_ops	64
use_ops	64
op	num_ops * 64
Total size	56 + 8 * num_ops bytes

Memory operands

Field	Size (bits)
gsbase	256
addresses_taken	64
def_ops	64
use_ops	64
vdef_ops	64
vuse_ops	64
stores	64
loads	64
op	num_ops * 64
Total size	88 + 8 * num_ops bytes

Data structures



Data structure definition and hierarchy

- Files
  - `gimple.def`
  - `gimple.h`
  - `gimple.c`
  - `gimple-iterator.c`
- Naming convention
  - `gimple_*`
  - `gimple_seq`
  - `struct gimple_statement_*`
  - `gimple_build_*`

Preliminary stats

(--enable-checking=release)

Memory (MB)

	Before	After	% change
insn-attrtab.c	420	399	- 5%
200.i	634	563	- 11%
tramp3d-v4.cc	1,688	1,700	+0.7%
FiniteElementMethod.cc	719	697	- 3%
PR12850	1,863	1,850	-0.7%

Compile time (secs)

	Before	After	% change
cc1-i-files	310	289	- 6.8%
SPEC 2000	242	200	-17.4%
tramp3d-v4	39	41	+ 5.0%
FF3D	197	193	- 2.0%
PR12850	64	66	+ 3.0%

Manipulating tuples

- Common accessors
  - `gimple_code`
  - `gimple_subcode`
  - `gimple_bb`
  - `gimple_block`
  - `gimple_locus`
  - `gimple_num_ops`
  - `gimple_op`

Manipulating tuples

- Specific accessors for frequently used codes
  - `gimple_assign_lhs`
  - `gimple_assign_rhs1`
  - `gimple_assign_rhs2`
  - `gimple_call_fn`
  - `gimple_call_arg`
  - `gimple_label_label`
  - `gimple_goto_dest`
  - `gimple_return_retval`

Manipulating sequences

- Adding/removing
  - `gimple_seq_add_stmt`
  - `gimple_seq_add_seq`
- Iterators
  - `gsi_start`
  - `gsi_start_bb`
  - `gsi_next`
  - `gsi_end_p`
  - `gsi_stmt`

Current status

- Branch builds an all primary and secondary platforms
- All primary languages converted
  - Ada still not converted
- ~10 unconverted passes
- 3 with no "owner"
  - `pass_if_conversion`
  - `pass_linear_transform`
  - `pass_vectorize`.
- Still some regressions wrt trunk
- Some TODO and cleanups
- http://gcc.gnu.org/wiki/tuples