The added value of CompCert lies not in the compilation technology implemented, but in the fact that each of the source, intermediate and target languages has formally defined semantics, and that each of the transformation and optimization passes is proved to preserve semantics...[1]

[1] Xavier Leroy - Formal Verification of a realistic compiler (2009)

Modeling Register Pairs in CompCert

Alex Loitzl[1,2] Florian Zuleger[1]

[1]Tu Wien, [2]Institute of Science and Technology Austria (ISTA)

November, 2024

CompCert - What, Who, When & Where

Authors

Xavier Leroy - **Formal verification** of a realistic **compiler** (2009)

Sandrine Blazy, Zaynah Dargaye, Jacques-Henri Jourdan, Michael Schmidt, Bernhard Schommer, and Jean-Baptiste Tristan

ACM Software System Award 2021

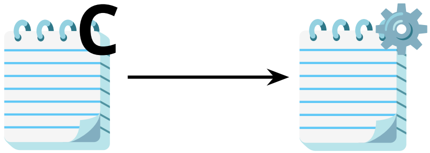
ACM SIGPLAN Programming Languages Software Award 2022

https://www.inria.fr

https://xavierleroy.org

https://www.absint.com

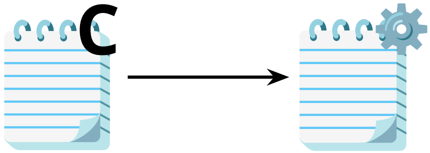
Correct Compiler



Correct Compiler

Verified Compiler

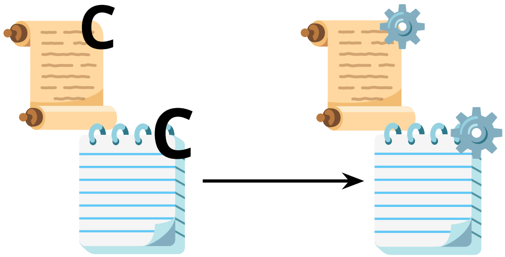
$$\forall S, P, \quad Comp(S) = \mathtt{OK}(P) \implies S \approx P$$



Correct Compiler

Verified Compiler

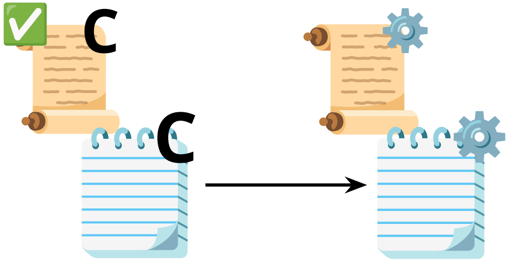
$\forall S, P, \quad Comp(S) = \mathtt{OK}(P) \implies S \approx P$



Correct Compiler

Verified Compiler

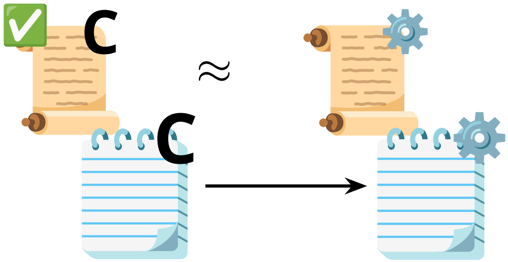
$$\forall S, P, \quad Comp(S) = \mathtt{OK}(P) \implies S \approx P$$



Correct Compiler

Verified Compiler

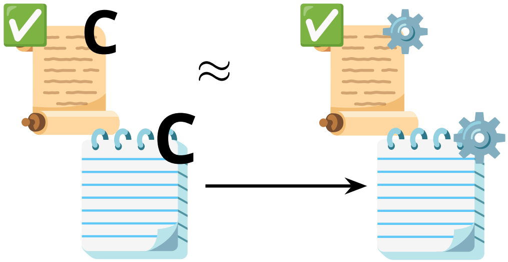
$$\forall S, P, \quad Comp(S) = \mathtt{OK}(P) \implies S \approx P$$



Correct Compiler

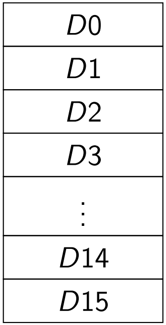
Verified Compiler

$\forall S, P, \quad Comp(S) = \texttt{OK}(P) \implies S \approx P$



Register Model

Model



Addressable:
$D0, D1, \ldots, D15$

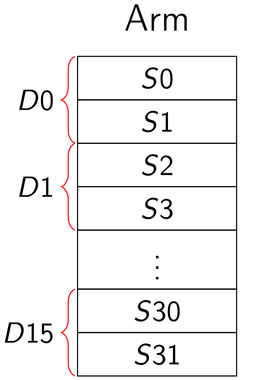
Lemma 1

$$\forall r, v : \; rs[r \leftarrow v](r) = v.$$

Lemma 2

$$\forall r, r', v : \; r \neq r' \Rightarrow rs[r' \leftarrow v](r) = rs(r).$$

Register Reality



Addressable:
$S0, S1, \ldots, S31$
$D0, D1, \ldots, D15$

Lemma 1

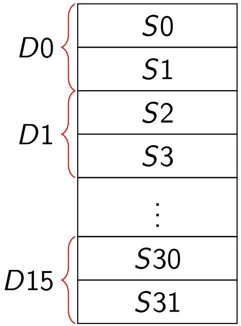
$$\forall r, v : \ rs[r \leftarrow v](r) = v.$$

Lemma 2

$$\forall r, r', v : \ r \neq r' \Rightarrow rs[r' \leftarrow v](r) = rs(r).$$

Register Reality

Arm



Addressable:
$S0, S1, \ldots, S31$
$D0, D1, \ldots, D15$

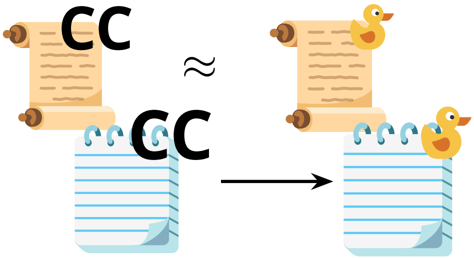
Lemma 1

$$\forall r, v: \; rs[r \leftarrow v](r) = v.$$

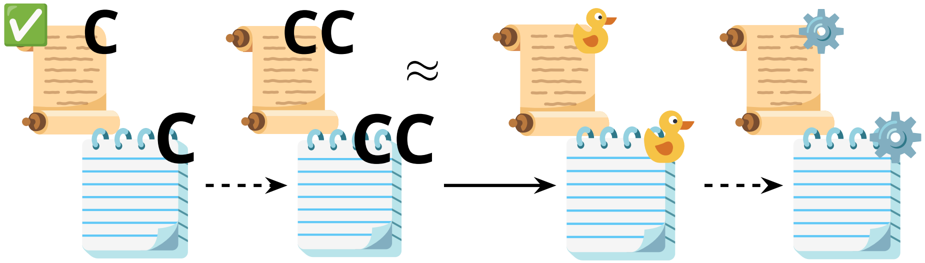
~~**Lemma 2**~~

~~$$\forall r, r', v: \; r \neq r' \Rightarrow rs[r' \leftarrow v](r) = rs(r).$$~~

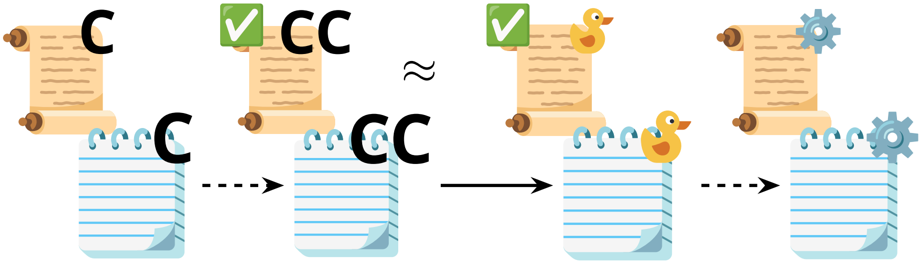
Correct Compiler ? - The whole picture



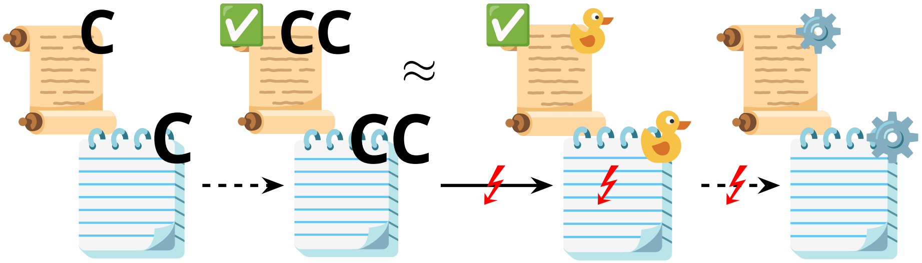
Correct Compiler ? - The whole picture



Correct Compiler ? - The whole picture

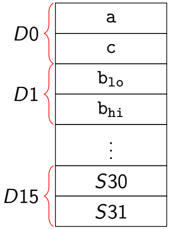


Correct Compiler ? - The whole picture



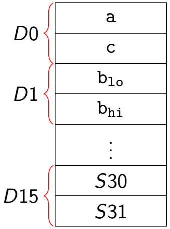
Consequences

```
double sum(single a, double b, single c){
  double d = (double) (a + c);
  return d + b;
}

int main(){
  sum (0.f, 1.0, 2.f);
  return 0;
}
```



Consequences

```c
double sum(single a, double b, single c){
  double d = (double) (a + c);
  return d + b;
}

int main(){
  sum (0.f, 1.0, 2.f);
  return 0;
}
```

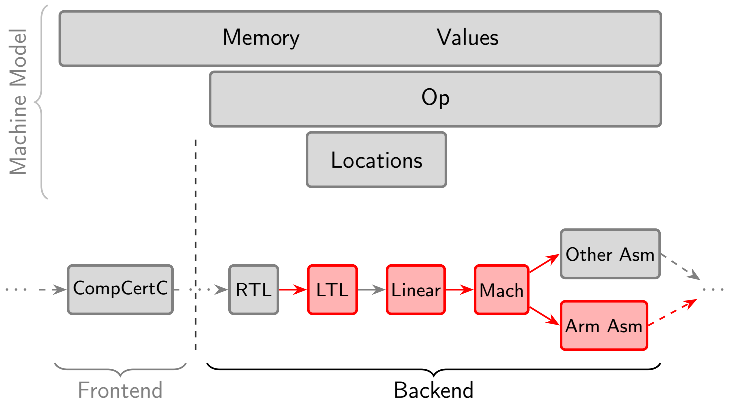


```asm
sum:
  vmov.f32 S4, S1
  ...
  vadd.f32 S0, S0, S4
  vcvt.f64.f32 D0, S0
  vadd.f64 D0, D0, D1
  ...
main:
  ...
  vmov.f32 S0, #0.
  vmov.f64 D1, #1.
  vmov.f32 S4, #2.
  vmov.f32 S1, S4
  bl   sum
  mov  R0, #0
  ...
```

Summary

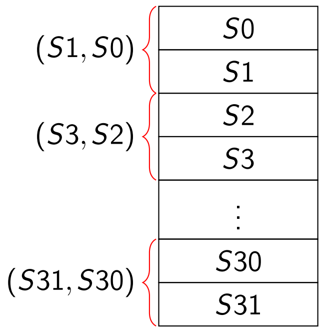
- Formally verified compiler
  - Proof covers all optimizations
  - Correct w.r.t. the modeled semantics
- Discrepancies between hardware and model
  - Cannot implement correct calling conventions
  - Cannot support TriCore architecture
- Suboptimal code generation
  - Inserted moves
  - Higher register pressure

CompCert's Machine Model



(New) Register Model

Arm



Addressable:
$S0, S1, \ldots, S31$
$(S1, S0), \ldots, (S31, S30)$

Lemma 1

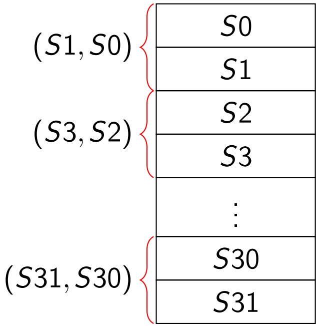
$$\forall p, v : \; rs[p \leftarrow v](p) = v.$$

Lemma 2

$$\forall p, p', v : \; p \cap p' = \emptyset \Rightarrow rs[p' \leftarrow v](p) = rs(p).$$

(New) Register Model

Arm



Addressable:
$S0, S1, \ldots, S31$
$(S1, S0), \ldots, (S31, S30)$

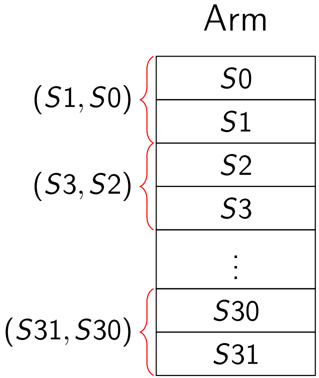
~~Lemma 1~~

$$\cancel{\forall p, v : \ rs[p \leftarrow v](p) = v.}$$

Lemma 2

$$\forall p, p', v : \ p \cap p' = \emptyset \Rightarrow rs[p' \leftarrow v](p) = rs(p).$$

(New) Register Model



Addressable:
$S0, S1, \ldots, S31$
$(S1, S0), \ldots, (S31, S30)$

~~Lemma 1~~

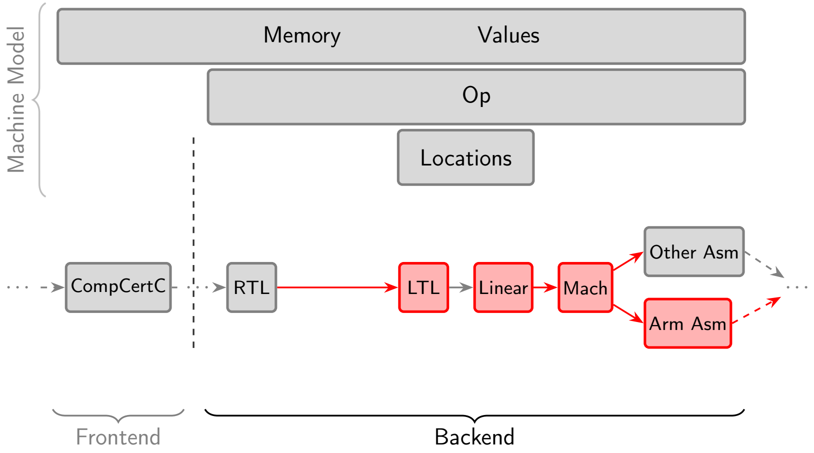
$$\cancel{\forall p, v : \ rs[p \leftarrow v](p) = v.}$$

Combining & Splitting

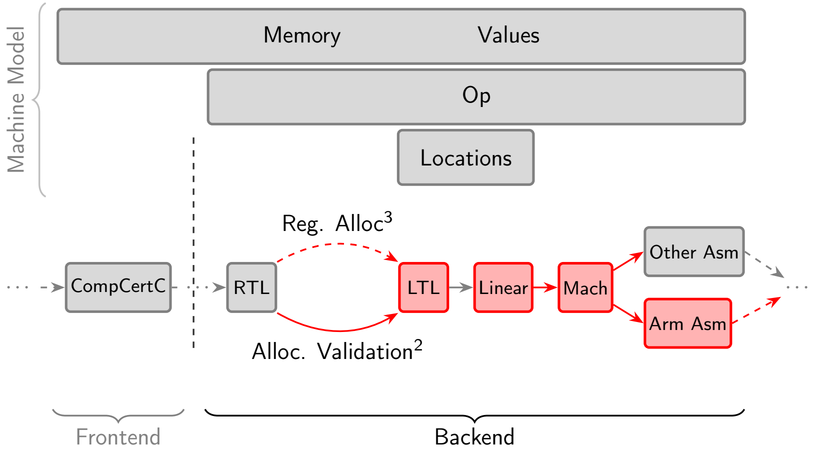
$$R((r1, r2)) \overset{def}{=} \texttt{combine}(R(r1), R(r2)), \ \textit{and}$$

$$R[(r1, r2) \leftarrow v] \overset{def}{=} R[r1 \leftarrow \texttt{hiwd}(v)][r2 \leftarrow \texttt{lowd}(v)].$$

CompCert's Machine Model



CompCert's Machine Model



[2] Rideau S., and Leroy X. Validating register allocation and spilling. (2010)

[3] Smith M.D., et al. A generalized algorithm for graph-coloring reg. alloc. (2004)

RTL to LTL

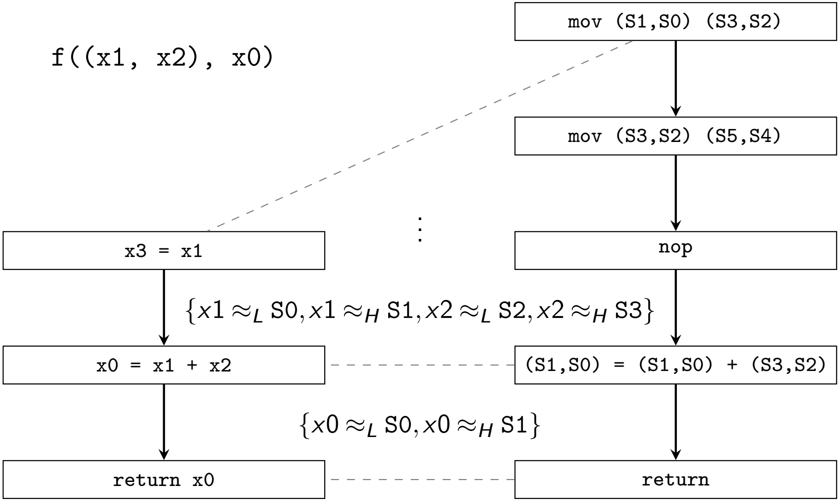
RTL Transition

$$\frac{c(pc) = \lfloor \mathrm{op}_{RTL}(op, \vec{x}, y, pc') \rfloor \qquad \texttt{eval\_op}(\_, \_, op, M(\vec{x})) = \lfloor v \rfloor}{\_ \vdash S(\_, \_, \_, pc, M, \_) \xrightarrow{\varepsilon} S(\_, \_, \_, pc', M[y \leftarrow v], \_)}$$

LTL Transition

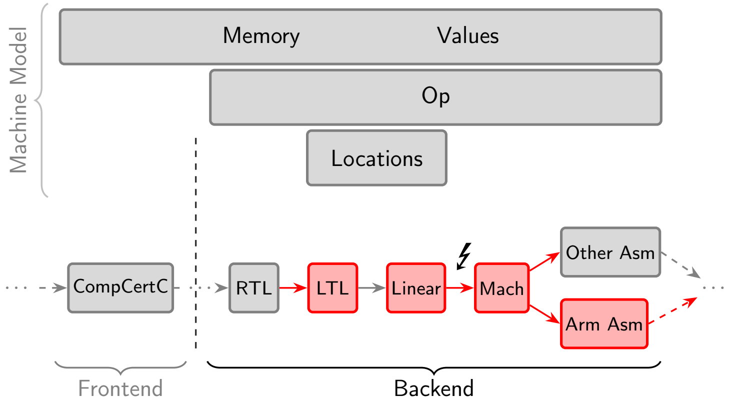
$$\frac{\texttt{eval\_op}(\_, \_, \mathrm{op}, L(\vec{p})) = \lfloor v \rfloor}{\_ \vdash B(\_, \_, \_, \mathrm{op}_{LTL}(op, \vec{p}, q) :: bb, L, \_) \xrightarrow{\varepsilon} B(\_, \_, \_, bb, L[q \leftarrow v], \_)}$$

Allocation Validation

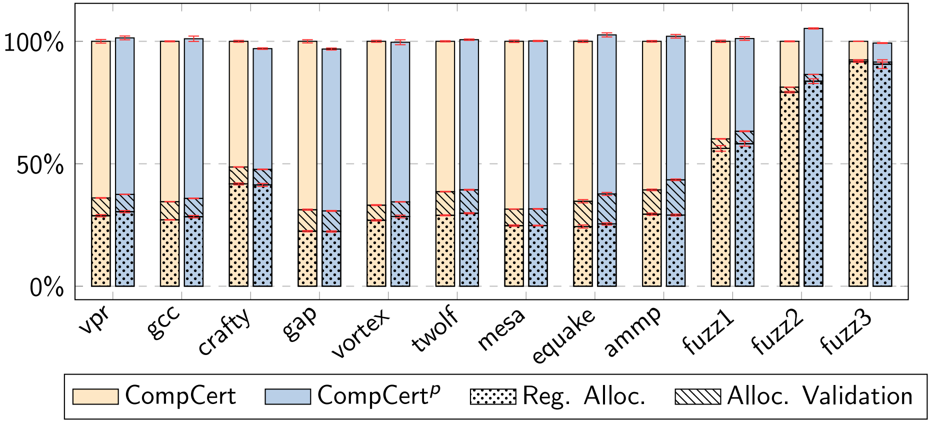


f((x1, x2), x0)
mov (S1,S0) (S3,S2)
mov (S3,S2) (S5,S4)
x3 = x1
⋮
nop
$\{x1 \approx_L \mathrm{S0}, x1 \approx_H \mathrm{S1}, x2 \approx_L \mathrm{S2}, x2 \approx_H \mathrm{S3}\}$
x0 = x1 + x2
(S1,S0) = (S1,S0) + (S3,S2)
$\{x0 \approx_L \mathrm{S0}, x0 \approx_H \mathrm{S1}\}$
return x0
return

CompCert's Machine Model



Results - Compile Time (Arm hard float)



Results - Code Size

	vpr	mesa	fuzz1	fuzz2	fuzz3
arm_hard	-0.83%	-1.77%	-4.78%	-4.7%	-4.7%
arm_soft	-0.2%	-0.71%	-0.2%	+0.19%	+0.27%

Contributions

- Improved model of the Arm assembly semantics
- Proved all architectures correct w.o. changing their semantics
- New and more general register allocator
- Enable future support for TriCore architecture
- Small positive impact on code generation

Questions?

1. Background on CompCert
2. Register Model & Reality
3. Example Code
4. New Register File
5. RTL to LTL
6. Evaluation