

The unforeseen evolution of theorem proving in ARM processor verification

Mike Gordon
University of Cambridge Computer Laboratory

This talk is an overview of research building on the 2000-2004 ARM6 verification project.

It emphasises the big picture, describing changes in direction and focus that took place over the subsequent years.

Only recently has there been much impact, including applications in unforeseen directions. Some of these will be outlined.

The research described in the talk is due to Anthony Fox and Magnus Myreen. Some of the material is taken their slides.

Others have contributed to the applications.

My role has been administrative rather than technical.



Prehistory

- ▶ Leeds 1980s: Tucker and Harman model digital behaviour
 - ▶ Tucker and Harman move to Swansea
 - ▶ They apply their mathematical theory to microprocessors
 - ▶ Anthony Fox completes PhD on processor verification (1998)
 - ▶ hand proof of pipelined machine using Tucker-Harman method
 - ▶ Mike Gordon external examiner – he's very impressed
-

- ▶ I had a long collaboration with Graham Birtwistle (Calgary)
- ▶ Joyce at Calgary verified Tamarack →





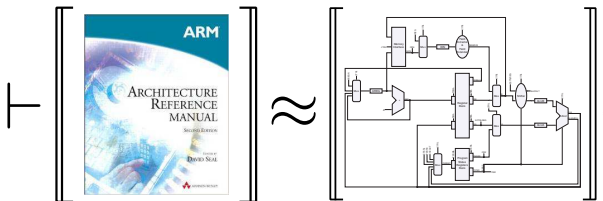
Formal Specification and Verification of ARM6

- ▶ Birtwistle (Leeds) and Gordon get grant in 2000
 - ▶ ARM a Project Partner
 - ▶ builds on earlier Tamarack collaboration
 - ▶ Leeds build ISA and processor models in ML
 - ▶ Cambridge convert to HOL and prove correctness
- ▶ Gordon hires Fox as project RA
 - ▶ mechanises Tucker-Harman method using HOL proof assistant
 - ▶ machine checks some proofs similar to those in his PhD
- ▶ ARM6 chosen after discussions with ARM (and their lawyers)
- ▶ ARM6 implementation modelled in higher order logic
- ▶ ARMv3 instruction semantics modelled in higher order logic
- ▶ Models proved equivalent using the HOL4 proof assistant



What is a formal verification of a processor

- ▶ Make a semantic model of the machine instructions
 - ▶ represented as a term in a suitable logic 
- ▶ Make a semantic model of the machine implementation
 - ▶ represented as a term in a suitable logic 
- ▶ Prove a theorem relating (\approx) the two
 - ▶ a formal proof in logic



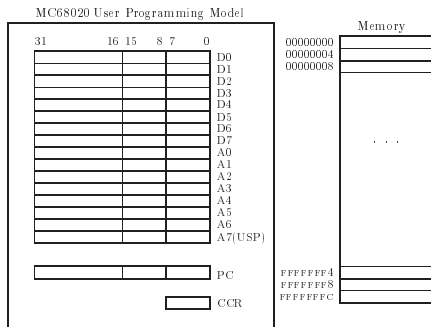
Formal Specification and Verification of ARM-Based Systems (EPSRC project, 2004-2007)

- ▶ ARM6 project a success
 - ▶ established feasibility of proving ARM-sized processors correct
 - ▶ but $\vdash \left[\text{ARM6} \right] \approx \left[\text{Circuit} \right]$ proof very labour intensive
 - ▶ low research value in doing similar bigger proofs
 - ▶ and details of more recent ARM processors unavailable
- ▶ EPSRC supported follow-on project
 - ▶ use verified ARM model as foundation for verifying software

“If successful, this project will result in possibly the first machine checked formal verification of software running on a formally verified commercial off the-shelf (COTS) processor.”

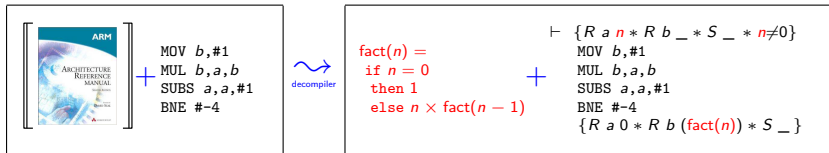
Aside on code verification

- ▶ Celebrated work by Boyer and Yu
("Automated Correctness Proofs of Machine Code Programs for a Commercial Microprocessor")



- ▶ execute code on model to show it implements Nqthm function
- ▶ separately verify function
- ▶ Impressive small examples done, but didn't scale
 - ▶ frame problem: need to prove what doesn't change
 - ▶ leads to complicated invariants

Myreen decompilation solves scaling problem



- ▶ Generate functionality + certification theorem automatically
 - ▶ functionality is a function definition in higher-order logic (HOL)
 - ▶ certification is a Hoare logic theorem $\vdash \{P\} C \{Q\}$
 - ▶ properties P, Q specify machine states and C is code
 - ▶ $\{P\} C \{Q\}$ means Q holds after running C in a state P
- ▶ Separating conjunction $*$ used to manage frame problem
 - ▶ P determines footprint of 'small' Hoare triple $\{P\} C \{Q\}$
 - ▶ Frame rule for unchanged stuff R :

$$\frac{\vdash \{P\} C \{Q\}}{\vdash \{P * R\} C \{Q * R\}}$$

Myreen's Hoare Logic for machine code



$\{R a x * R b _ * R p c p\}$
└─ MOV b, a
 $\{R a x * R b x * R p c (p+1)\}$



$\{R a_1 x_1 * R a_2 x_2 * R b _ * R p c p\}$
└─ MUL b, a_1, a_2
 $\{R a_1 x_1 * R a_2 x_2 * R b (x_1 \times x_2) * R p c (p+1)\}$



$\{R a_1 x_1 * R a_2 x_2 * R b _ * S _ * R p c p\}$
└─ SUBS b, a_1, a_2
 $\{R a_1 x_1 * R a_2 x_2 * R b (x_1 - x_2) * S (x_1 = x_2) * R p c (p+1)\}$



$\{S b * R p c p\}$
└─ BNE $\#-n$
 $\{R p c (\text{if } b \text{ then } p-n \text{ else } p+1)\}$



$\{R a n * R b _ * S _ * R p c p * n \neq 0\}$
└─ MOV $b, \#1$
 MUL b, a, b
 SUBS $a, a, \#1$
 BNE $\#-4$
 $\{R a 0 * R b (\text{fact}(n)) * S _ * R p c (p+4)\}$

+ $\text{fact}(n) =$
if $n = 0$
then 1
else $n \times \text{fact}(n-1)$

State by about 2007 after two 3-year EPSRC projects

- ▶ Myreen devised his new method to verify machine code
 - ▶ ARM model provides machine code semantics
 - ▶ method combines **decompilation** with **separation logic**
 - ▶ impressive small examples: multiplier, garbage collector
- ▶ Fox created unverified 'high fidelity' model of ARMv4T
 - ▶ not then totally obsolete
 - ▶ input/output and exceptions (interrupts) accurately handled
 - ▶ processor, memory, coprocessors are separate components
- ▶ Another EPSRC proposal submitted
 - ☹ proposal rejected
 - 😊 unforeseen funder offers to support the work anyway
 - 😊 support for research on ARM models and tools continues

Model and tool engineering continues at Cambridge

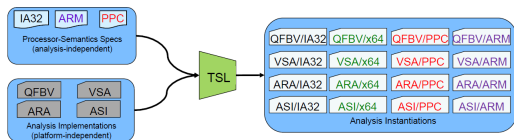
- ▶ New ARMv7 model developed
 - ▶ supported by the latest processors at the time
- ▶ Models tested against ARM hardware
 - ▶ random instructions executed in theorem prover and on ARM
 - ▶ bugs found in formal models
- ▶ New ISA models: x86, PPC, MIPS
 - ▶ Fox and Myreen create general decompilation infrastructure
- ▶ Verified LISP REPL: Myreen (Cambridge) and Davis (Texas)
 - ▶ `Julawa`: verified x86 implementation of Lisp read-eval-print-loop
- ▶ First impact outside Cambridge emerges
 - ▶ ARM models → GrammaTech TSL
 - ▶ correctness of seL4 binary
 - ▶ KTH PROSPER hypervisor information flow security

GammaTech evaluate ARM model

- ▶ CodeSonar “a static analysis tool for source code and binaries”

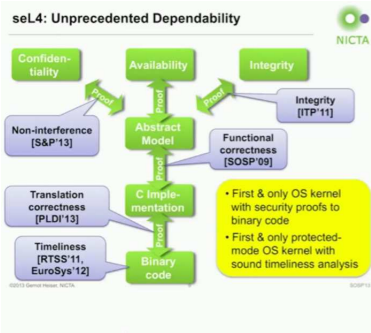
TSL/ISAL: Automatic Analysis Retargeting

- Separate processor semantics specs from analysis implementations
- Automatically generate platform-specific analysis instantiations
- Benefits:
 - › **Independence** of semantics and analyses
 - Validation of each ISA semantics is separate from static analyses
 - Validation of each static analysis is separate from ISA definitions
 - › **Consistency**. All analyses for given ISA driven off of same definition
 - › **Completeness**. Full analysis generated for all instructions.



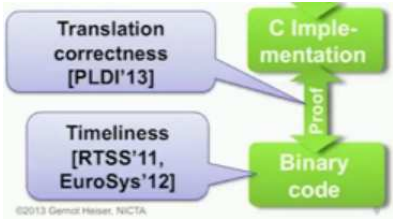
- ▶ GammaTech have a proprietary ISA description language TSL
- ▶ ARM model translated into TSL by Fox
- ▶ GammaTech also made their own port from L3 (see later)

Binary verification of seL4 OS microkernel



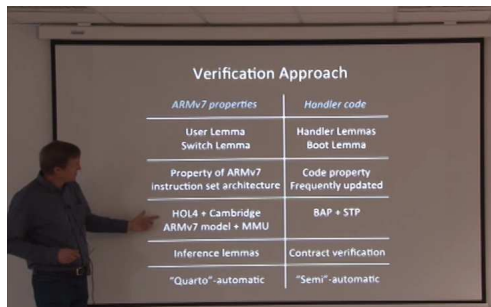
← Gernot Heiser

- ▶ Verified 'translation correctness' by Myreen at NICTA



KTH PROSPER project

Mads Dam →

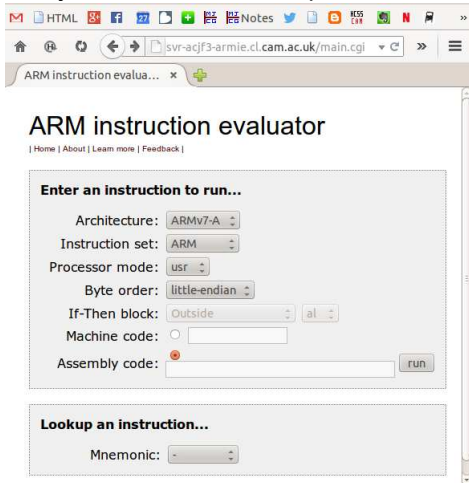


<i>ARMv7 properties</i>	<i>Handler code</i>
User Lemma Switch Lemma	Handler Lemmas Boot Lemma
Property of ARMv7 instruction set architecture	Code property Frequently updated
HOL4 + Cambridge ARMv7 model + MMU	BAP + STP
Inference lemmas	Contract verification
"Quarto"-automatic	"Semi"-automatic

- ▶ **PRO**vably **S**ecure Execution **P**latforms for **E**mbedded **S**ystems
- ▶ Open source hypervisor verification project
- ▶ ISA isolation lemmas proved for user mode execution on ARM

Meanwhile back in Cambridge Fox develops more models

- ▶ ARMv4, ARMv4T, ARMv5T, ARMv5TE, ARMv6, ARMv6K, ARMv6T2, ARMv7-A, ARMv7-R, ARMv8 (new for KTH), ARM-M0 (new for Edinburgh REMS)
- ▶ many ISA versions and options



The screenshot shows a web browser window with the URL `svr-acjf3-armie.cl.cam.ac.uk/main.cgi`. The page title is "ARM instruction evaluator". Below the title are navigation links: [Home](#) | [About](#) | [Learn more](#) | [Feedback](#).

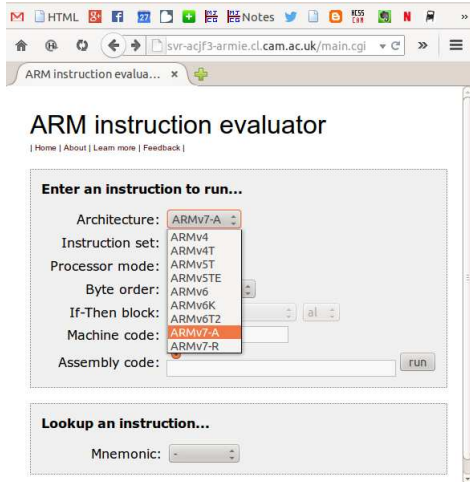
The main content area is titled "Enter an instruction to run...". It contains several dropdown menus and input fields:

- Architecture: `ARMv7-A`
- Instruction set: `ARM`
- Processor mode: `usr`
- Byte order: `little-endian`
- If-Then block: `Outside` and `al`
- Machine code:
- Assembly code:

Below this section is another section titled "Lookup an instruction...". It contains a dropdown menu for "Mnemonic:".

Web interface to ARM models

- ▶ ARMv4, ARMv4T, ARMv5T, ARMv5TE, ARMv6, ARMv6K, ARMv6T2, ARMv7-A, ARMv7-R, ARMv8 (new for KTH), ARM-M0 (new for Edinburgh REMS)
- ▶ many ISA versions and options: [architecture](#)



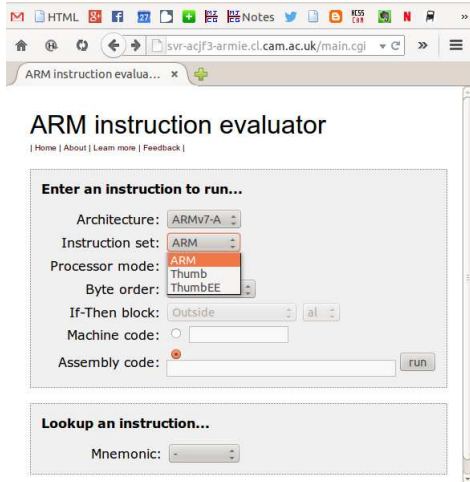
The screenshot shows a web browser window with the URL `svr-acjf3-armie.cl.cam.ac.uk/main.cgi`. The page title is "ARM instruction evaluator". Below the title are links for "Home", "About", "Learn more", and "Feedback". The main content area is titled "Enter an instruction to run..." and contains several input fields:

- Architecture:** A dropdown menu currently showing "ARMv7-A".
- Instruction set:** A dropdown menu with options: ARMv4, ARMv4T, ARMv5T, ARMv5TE, ARMv6, ARMv6K, ARMv6T2, ARMv7-A, and ARMv7-R.
- Processor mode:** A dropdown menu.
- Byte order:** A dropdown menu.
- If-Then block:** Two dropdown menus, one showing "al".
- Machine code:** A text input field.
- Assembly code:** A text input field.

A "run" button is located to the right of the assembly code field. Below this section is another section titled "Lookup an instruction..." with a "Mnemonic:" dropdown menu.

Web interface to ARM models

- ▶ ARMv4, ARMv4T, ARMv5T, ARMv5TE, ARMv6, ARMv6K, ARMv6T2, ARMv7-A, ARMv7-R, ARMv8 (new for KTH), ARM-M0 (new for Edinburgh REMS)
- ▶ many ISA versions and options: [instruction set](#)



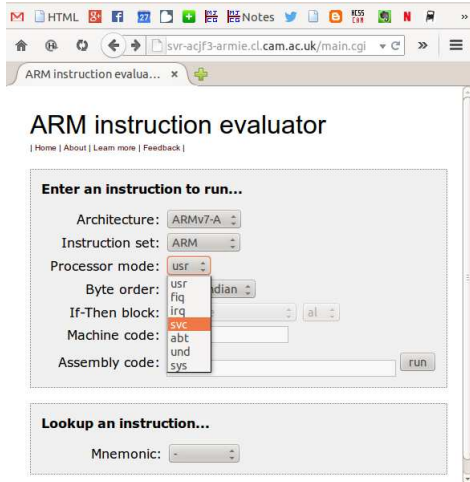
The screenshot shows a web browser window with the URL `svr-acjf3-armie.cl.cam.ac.uk/main.cgi`. The page title is "ARM instruction evaluator". Below the title are links for "Home", "About", "Learn more", and "Feedback". The main content area is titled "Enter an instruction to run..." and contains several form fields:

- Architecture:
- Instruction set: (with a dropdown menu open showing "ARM" selected, "Thumb", and "ThumbEE")
- Processor mode:
- Byte order:
- If-Then block:
- Machine code:
- Assembly code:

Below this section is another section titled "Lookup an instruction..." with a "Mnemonic:" label and a dropdown menu.

Web interface to ARM models

- ▶ ARMv4, ARMv4T, ARMv5T, ARMv5TE, ARMv6, ARMv6K, ARMv6T2, ARMv7-A, ARMv7-R, ARMv8 (new for KTH), ARM-M0 (new for Edinburgh REMS)
- ▶ many ISA versions and options: [processor mode](#)



The screenshot shows a web browser window with the URL `svr-acjf3-armie.cl.cam.ac.uk/main.cgi`. The page title is "ARM instruction evaluator". Below the title are navigation links: [Home](#) | [About](#) | [Learn more](#) | [Feedback](#).

The main content area is titled "Enter an instruction to run...". It contains several dropdown menus and input fields:

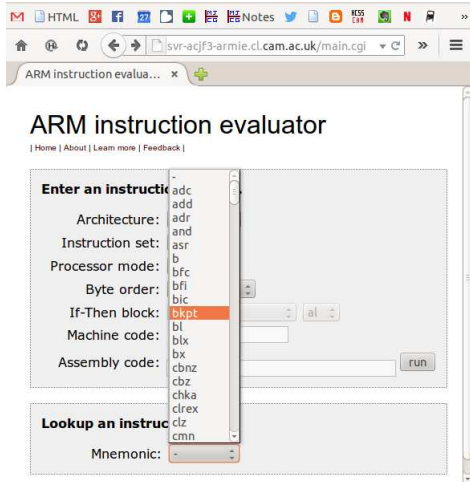
- Architecture: `ARMv7-A`
- Instruction set: `ARM`
- Processor mode: `usr` (with a dropdown menu open showing options: `usr`, `fiq`, `irq`, `svc`, `abt`, `und`, `sys`)
- Byte order: `usr` and `little endian`
- If-Then block: `irq` and `al`
- Machine code:
- Assembly code:

A "run" button is located to the right of the assembly code input field.

Below this section is another section titled "Lookup an instruction...". It contains a "Mnemonic:" dropdown menu.

Web interface to ARM models

- ▶ ARMv4, ARMv4T, ARMv5T, ARMv5TE, ARMv6, ARMv6K, ARMv6T2, ARMv7-A, ARMv7-R, ARMv8 (new for KTH), ARM-M0 (new for Edinburgh REMS)
- ▶ many ISA versions and options: [instructions](#)



The screenshot shows a web browser window with the URL `svr-acjf3-armie.cl.cam.ac.uk/main.cgi`. The page title is "ARM instruction evaluator". Below the title are navigation links: [Home](#) | [About](#) | [Learn more](#) | [Feedback](#).

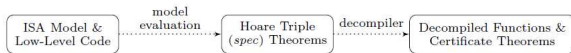
The main interface is divided into two sections:

- Enter an instruction:** This section contains several dropdown menus for configuration:
 - Architecture: `adc`, `add`, `adr`, `and`
 - Instruction set: `asr`, `b`
 - Processor mode: `bfc`, `bfi`, `bic`
 - Byte order: `bkpt` (highlighted), `bl`
 - If-Then block: `blx`
 - Machine code: `bx`
 - Assembly code: `cbnz`, `cbz`, `chka`, `clrex`, `clz`, `cmn`
- Lookup an instruction:** This section has a "Mnemonic:" dropdown menu.

To the right of the dropdowns, there are input fields for registers (e.g., `al`) and a "run" button.

To manage models Fox designs L3 (Low Levell Language)

- ▶ L3 created for easier authoring and managing logic models
 - ▶ engineer friendly syntax based on ARM ISA manuals
 - ▶ tool for generating logic models for proof
 - ▶ tool for generating code for testing
- ▶ Diagram pinched from a draft paper by Fox





- ▶ Slide from a recent talk

L3 Instruction Set Models

Model	Supported Modes	Coverage	Lines of L3
ARMv4 through to ARMv7-A	ARM and Thumb	Partial VFP, no Advanced SIMD & coprocessor	9238 + 7687
ARMv6-M	Thumb only	No coprocessor	1996 + 2095
ARMv8	AArch64 only	No VFP, Advanced SIMD and only partial system	2434 + 4097
x86-64	64-bit only	Only 40 instructions	1357 + 1579
MIPS (in HOL)		No floating-point, partial coprocessor & system.	2080 + 700
CHERI		Enough to boot FreeBSD	5203

Applications of L3

- ▶ GrammaTech do their own port of L3 ARM model to TSL
- ▶  Technology Readiness Level (TRL) evaluation
- ▶  bytecode to x86 binary verification
- ▶ BERI and CHERI testing and design exploration


TRL evaluation project

- ▶ Assess TRL of L3-based decompilation
- ▶ Compare decompiled binaries from different compilers
 - ▶ compile tiny example of same functionality from C and Ada
 - ▶ decompile binaries
 - ▶ verify equivalent function by theorem proving
- ▶ Success – project features in the video on the D-RisQ website



- ▶ Lead to a multi-university + industry proposal
 - ▶ extract the function of binary using L3
 - ▶ transform HOL functions to CSP using Isabelle/HOL
 - ▶ verify properties with FDR3
 - ▶ driven by applications from automotive industry partner

Verified translation of CakeML bytecode to x86 binary

- ▶ Verified compiler for 
 - ▶ CakeML is a variant of Standard ML
 - ▶ L3 tools verify generated bytes correctly translated to x86
- ▶ International collaboration:
 - ▶ UK: Cambridge, Kent (CaKe)
 - ▶ international: NICTA (Australia), Chalmers (Sweden)
 - ▶ www.cakeml.org
- ▶ TRL evaluation at Rockwell Collins (USA)
 - ▶ funded by NASA
 - ▶ get verified executables from logic specifications
 - ▶ explore use of verified compilers in certification

BERI and CHERI testing and design exploration



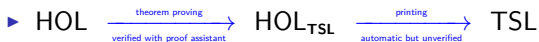
- ▶ MIPS-based clean slate processor designs
- ▶ ISA modelled in L3
- ▶ Multi-core mode FreeBSD booted using L3 simulator
- ▶ Processor designers now use L3 for architecture explorations

The evolution of theorem proving from ARM6 verification

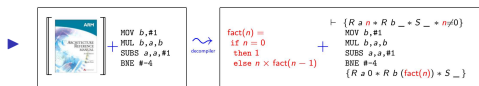
1. Traditional hardware verification



2. Translating models



3. Proving decompilation certification theorems



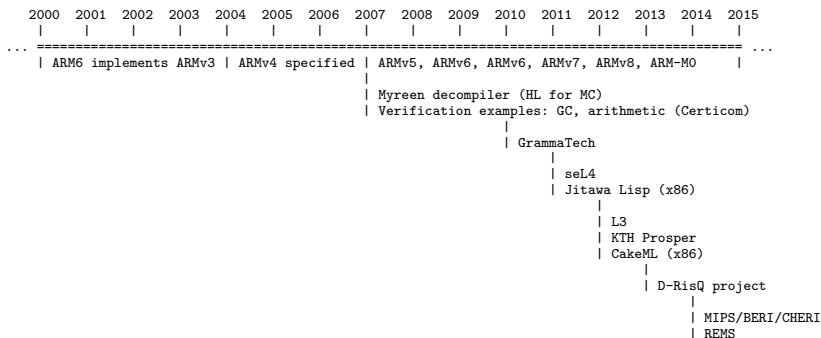
4. Specification debugging by trustworthy simulation

- ▶ comparing hardware and proof-generated execution traces

5. Language for authoring and executing HOL models

- ▶ L3 bridges worlds of specifier, verifier and design explorer

Observations



- ▶ Unforeseen impact of research
 - ▶ proofs about ARM \rightsquigarrow MIPS extensions design testing
- ▶ Results take a long time to emerge
 - ▶ 15 years developing ARM models and over 8 applying them
- ▶ Infrastructure development important
 - ▶ hard to fund using usual research grants

Response to a policy document about future funding

“It seems that immediate impact, which is likely to be incremental in nature, is overshadowing the longer-term perspective and may lead to short-term gains but a dearth of major advances for the future. EPSRC, for example, is concentrating funds into strategic areas and so squeezing the money for responsive-mode funding, and hence the space available for curiosity-driven research. Since this approach is meant to be all about achieving “impact” and plenty of impact has arisen serendipitously from high quality curiosity-driven research in the past, this could be counter-productive: even having the opposite effect to that intended.”

Response to a policy document about future funding

“It seems that **immediate impact**, which is likely to be incremental in nature, is overshadowing the longer-term perspective and may lead to **short-term gains** but a **dearth of major advances for the future**. EPSRC, for example, is concentrating funds into strategic areas and so squeezing the money for responsive-mode funding, and hence the space available for curiosity-driven research. Since this approach is meant to be all about achieving “impact” and plenty of impact has arisen serendipitously from high quality curiosity-driven research in the past, this could be counter-productive: even having the opposite effect to that intended.”

Response to a policy document about future funding

“It seems that **immediate impact**, which is likely to be incremental in nature, is overshadowing the longer-term perspective and may lead to **short-term gains** but a **dearth of major advances for the future**. EPSRC, for example, is concentrating funds into strategic areas and so squeezing the money for responsive-mode funding, and hence the space available for **curiosity-driven research**. Since this approach is meant to be all about achieving “impact” and **plenty of impact** has arisen **serendipitously** from high quality curiosity-driven research in the past, this could be counter-productive: even having the opposite effect to that intended.”

Response to a policy document about future funding

“It seems that **immediate impact**, which is likely to be incremental in nature, is overshadowing the longer-term perspective and may lead to **short-term gains** but a **dearth of major advances for the future**. EPSRC, for example, is concentrating funds into strategic areas and so squeezing the money for responsive-mode funding, and hence the space available for **curiosity-driven research**. Since this approach is meant to be all about achieving “impact” and **plenty of impact** has arisen **serendipitously** from high quality curiosity-driven research in the past, this could be counter-productive: even having the opposite effect to that intended.”

The ARM project is an example of research that had little “immediate impact” ... but eventually after many years, and thanks to far-sighted funders, is having “plenty of impact” which indeed has “arisen serendipitously”.

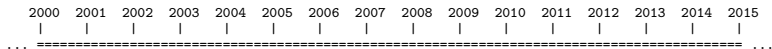
Response to a policy document about future funding

“It seems that **immediate impact**, which is likely to be incremental in nature, is overshadowing the longer-term perspective and may lead to **short-term gains** but a **dearth of major advances for the future**. EPSRC, for example, is concentrating funds into strategic areas and so squeezing the money for responsive-mode funding, and hence the space available for **curiosity-driven research**. Since this approach is meant to be all about achieving “impact” and **plenty of impact** has arisen **serendipitously** from high quality curiosity-driven research in the past, this could be counter-productive: even having the opposite effect to that intended.”

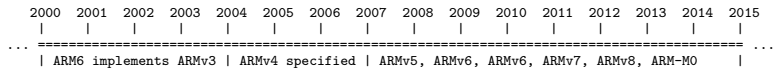
The ARM project is an example of research that had little “immediate impact” ... but eventually after many years, and thanks to far-sighted funders, is having “plenty of impact” which indeed has “arisen serendipitously”.

THE END

Additional slides reviewing what was done



Fox makes many ARM models



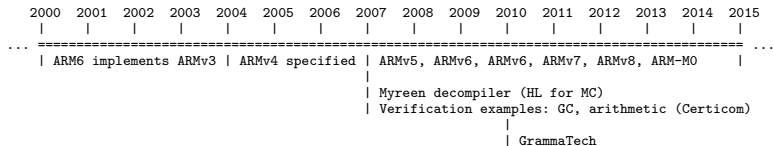
Myreen uses separation logic for decompilation

2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
...	-----														...
	ARM6 implements ARMv3				ARMv4 specified				ARMv5, ARMv6, ARMv6, ARMv7, ARMv8, ARM-MO						
								Myreen decompiler (HL for MC)							

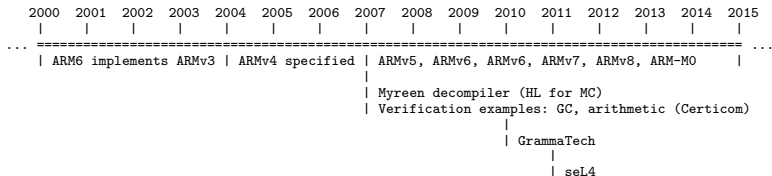
First case studies

```
2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |
...  -----  ...
| ARM6 implements ARMv3 | ARMv4 specified | ARMv5, ARMv6, ARMv6, ARMv7, ARMv8, ARM-MO |
|                               |                               |                               |
|                               |                               | Myreen decompiler (HL for MC) |
|                               |                               | Verification examples: GC, arithmetic (Certicom) |
```

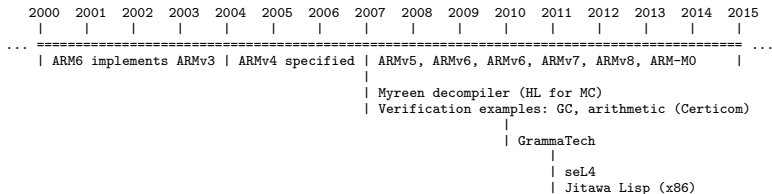
First industry interest



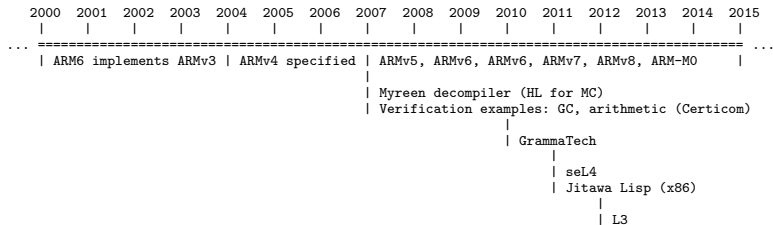
ARM binary decompilation used in sel4 verification



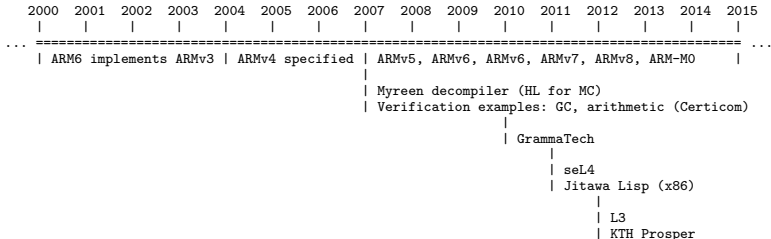
Myreen verifies JIT Lisp for Jared Davis' Milawa



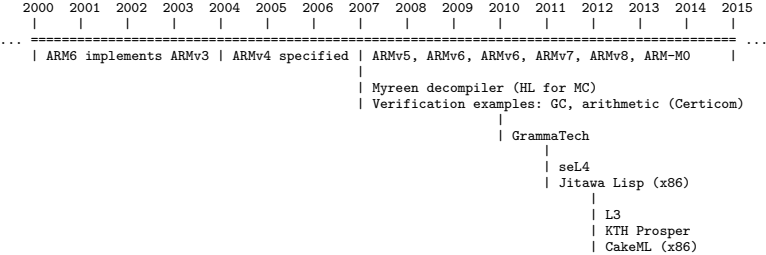
Fox develops L3 to manage ARM models

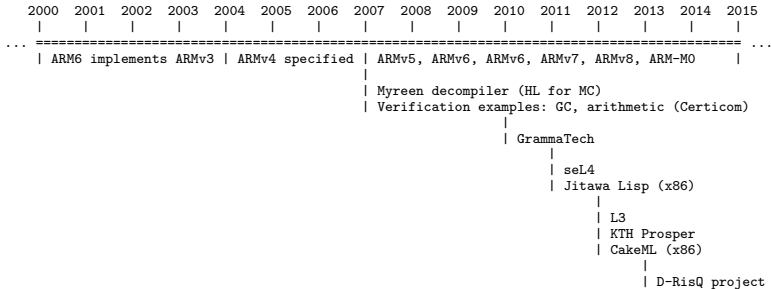


KTH adopt ARM model for security verification

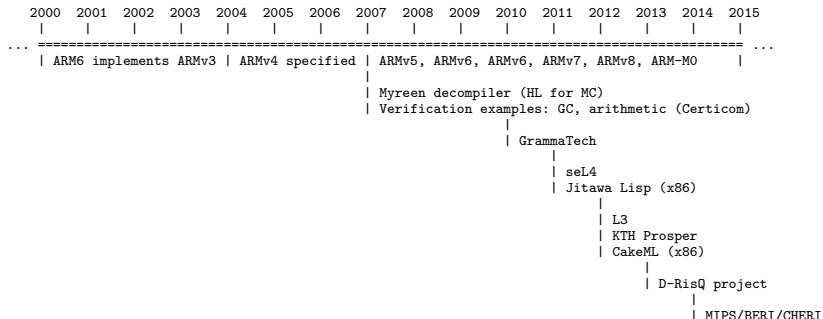


CakeML verification of correct x86 code generation

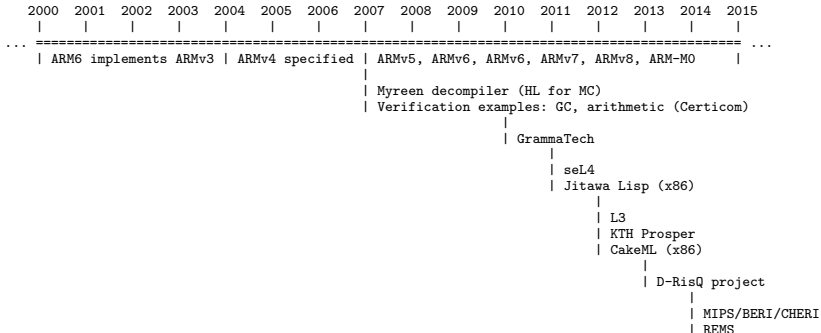




Fox L3 MIPS models used by CTSRD



REMS project hires Fox



REMS project hires Fox ... Gordon retires

