MANCHESTER 1824
The University of Manchester

MAMBO
Dynamic Binary Modification for RISC-V

FOSDEM 2024, Brussels, 4 February 2024

John Alistair Kressel
Igor Wodiany

University of Manchester
<firstname>.<lastname>@manchester.ac.uk

MANCHESTER 1824
The University of Manchester

Introduction to Dynamic Binary Modification (DBM) and MAMBO

What is DBM / DBI / DBT?

What is DBM?

MANCHESTER 1824
The University of Manchester

Example DBM / DBI / DBT

Valgrind

QEMU

MANCHESTER 1824
The University of Manchester

What is DBM / DBI / DBT?

- **D**ynamic - Working at runtime
- **B**inary - Natively compiled user-space code
- **M**odification - Alteration of applications

MANCHESTER 1824
The University of Manchester

What is DBM / DBI / DBT?

- **D**ynamic - Working at runtime
- **B**inary - Natively compiled user-space code
- **M**odification - Alteration of applications

- **I**nstrumentation – Inserting additional functionality
- **T**ranslation- Translating one instruction set into another

MANCHESTER 1824
The University of Manchester

Uses of DBM / DBI / DBT tools

- Program analysis
  - Callgrind (Valgrind)
- Memory error detection / debugging
  - Memcheck (Valgrind), Dr. Memory (DynamoRIO), Memcheck (MAMBO)
- Dynamic binary translation
  - QEMU, Apple Rosetta, TANGO

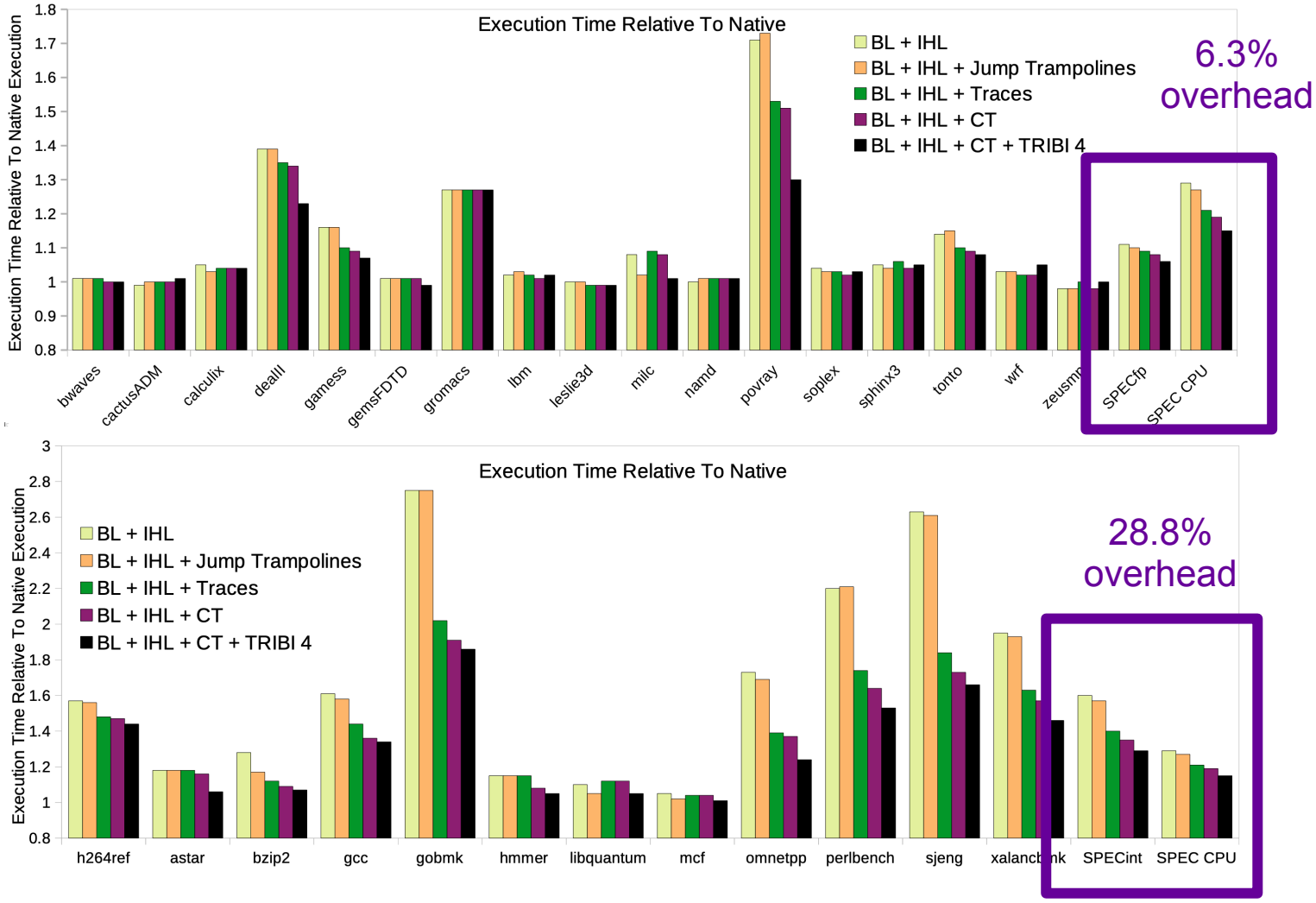
MANCHESTER 1824
The University of Manchester

Why MAMBO?

- Optimized for RISC-V 64-bit, ARM 32-bit & ARM 64-bit
  - Low overhead
  - **Only available DBM optimized for RISC-V**
- Low complexity
  - Relatively small codebase (~20k LoC)
- Simple plugin API
  - Architecture agnostic helper functions for portable plugins
- Not a toy

MANCHESTER 1824
The University of Manchester

Why MAMBO on RISC-V?



Slowdown relative to native execution for SPEC CPU2006 – RISC-V 64GC.

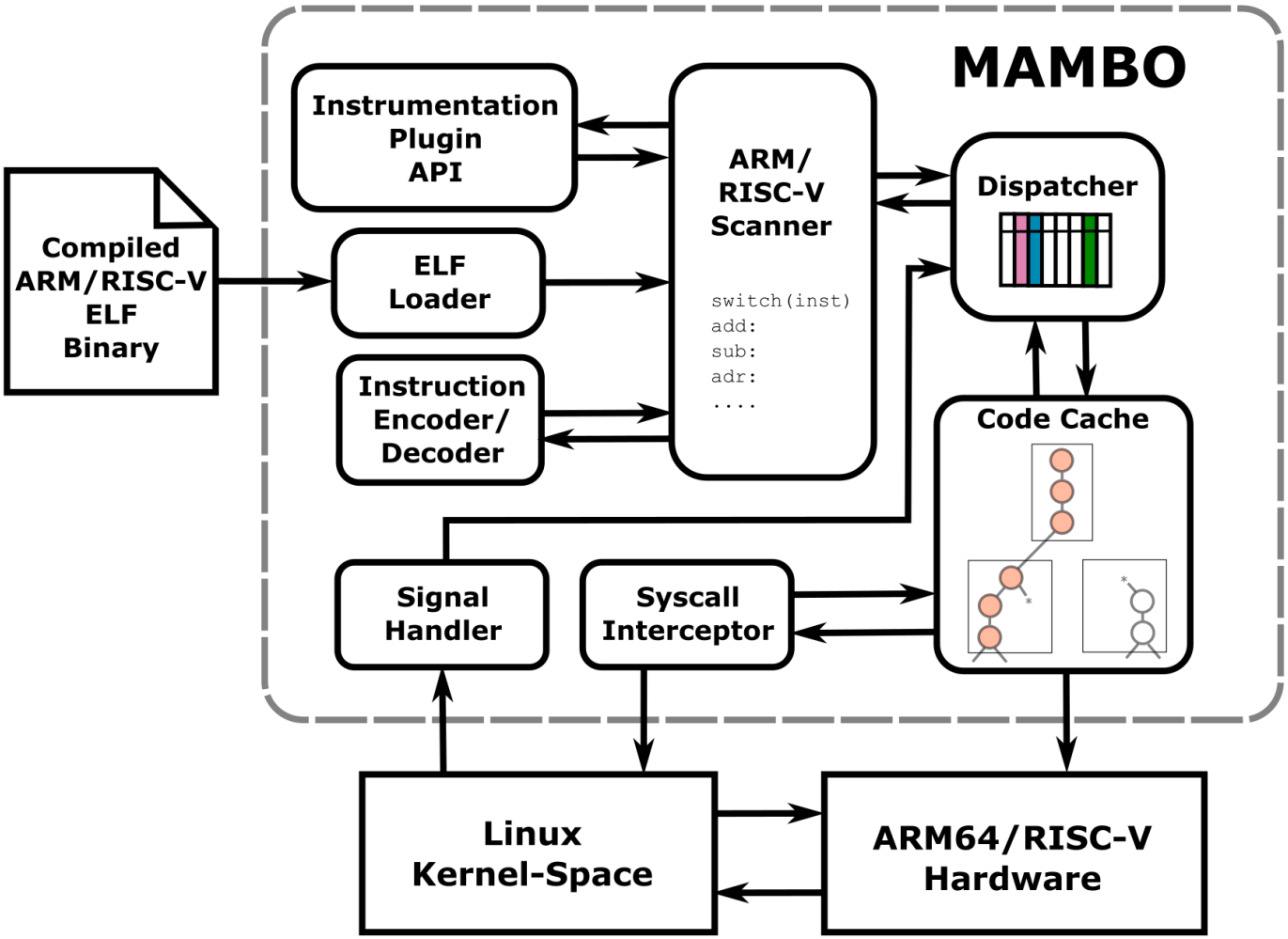
Kressel et al. Evaluating the Impact of Optimizations for Dynamic Binary Modification on 64-bit RISC-V.

MANCHESTER 1824
The University of Manchester

MAMBO Architecture

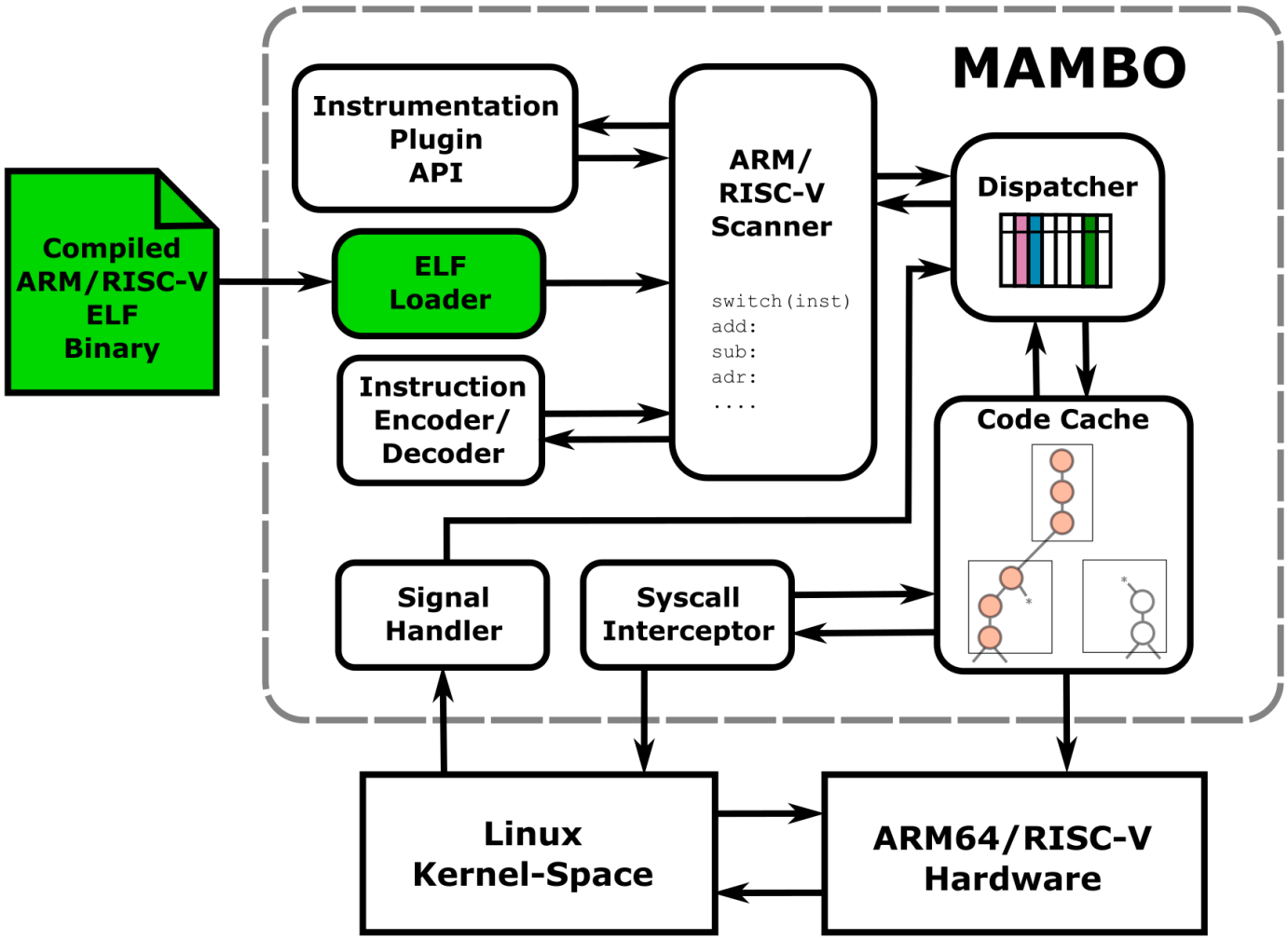
MANCHESTER 1824
The University of Manchester

MAMBO Architecture



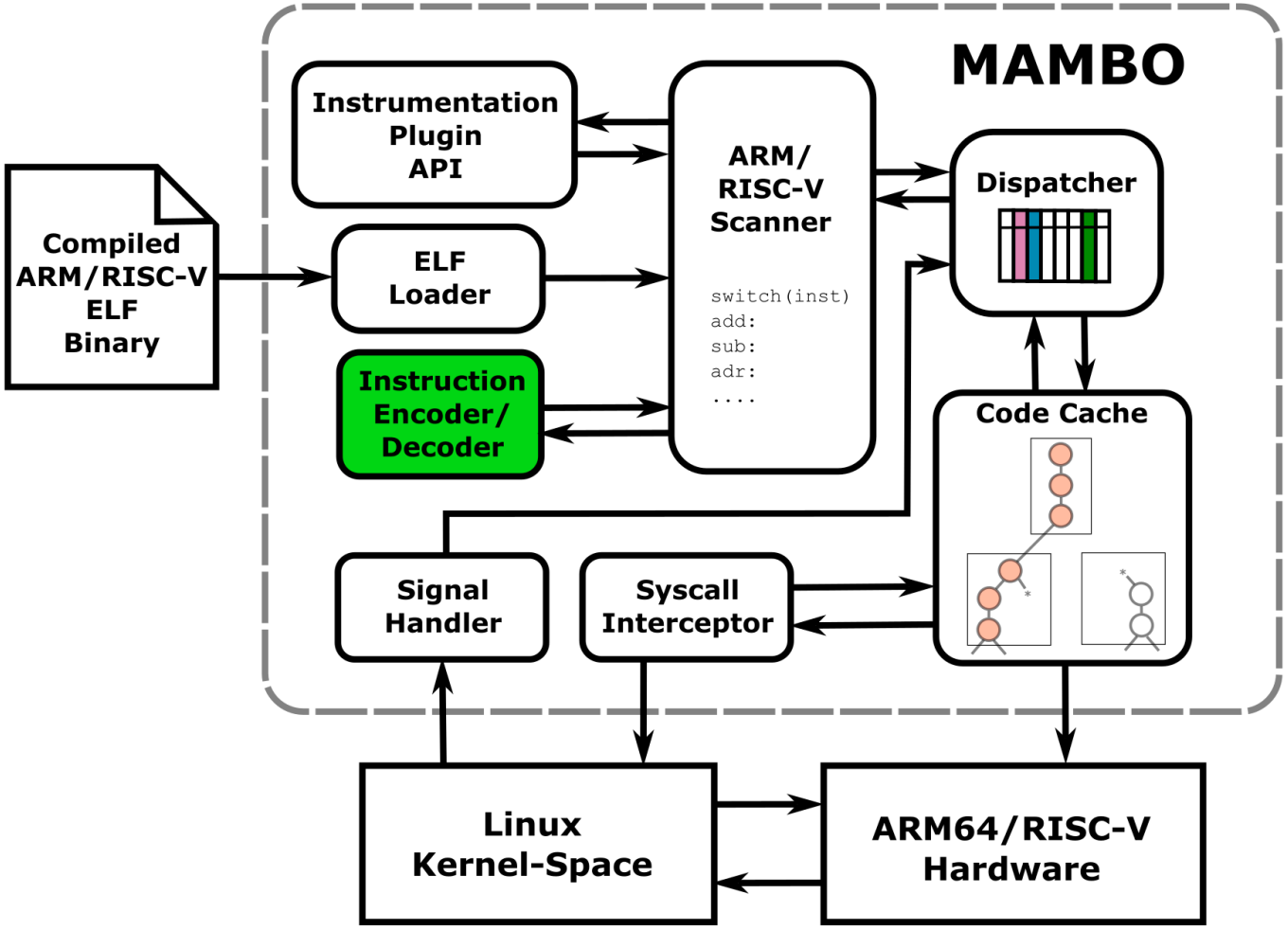
MANCHESTER 1824
The University of Manchester

ELF Loading



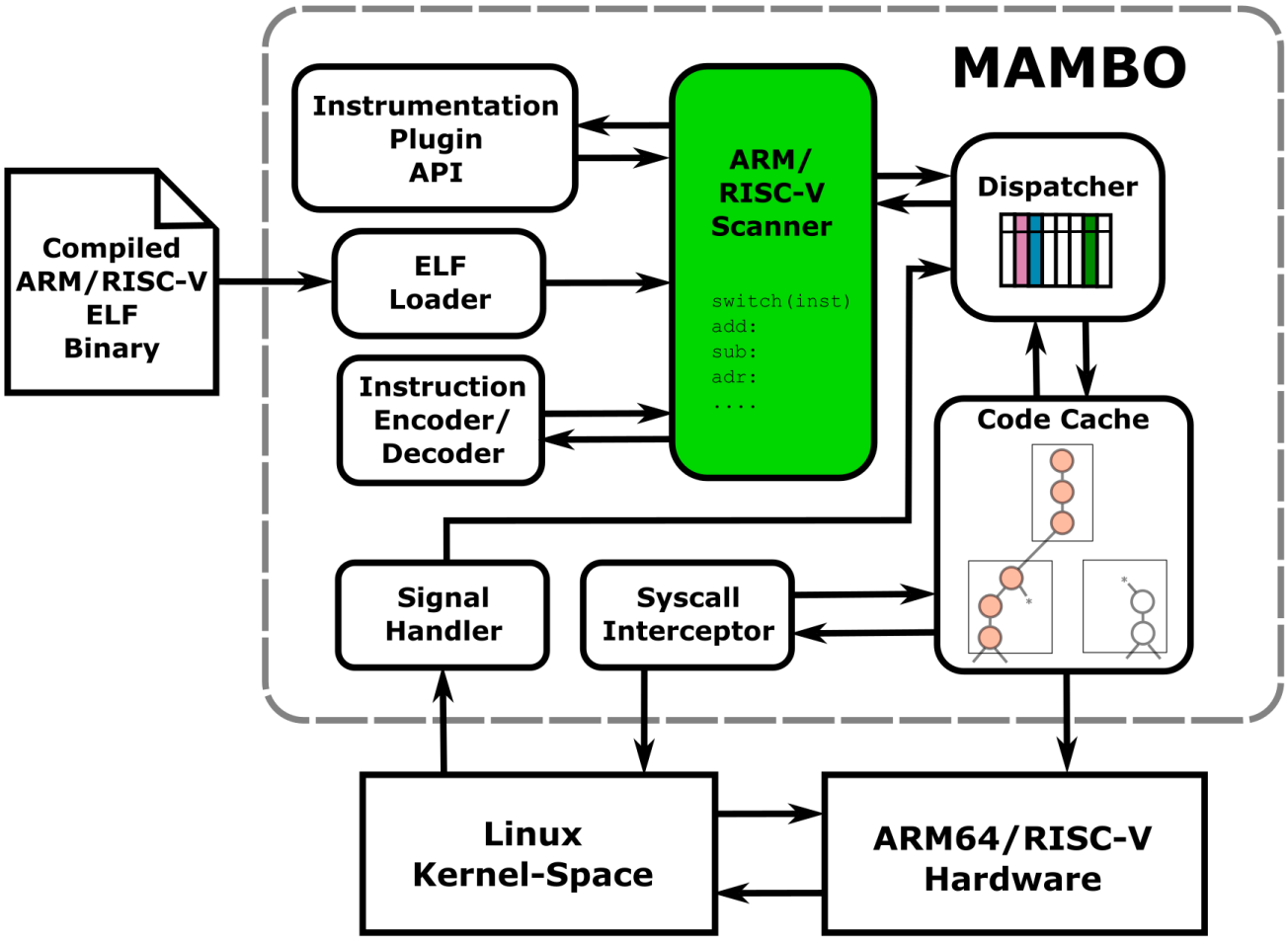
MANCHESTER 1824
The University of Manchester

Instruction Encoder / Decoder



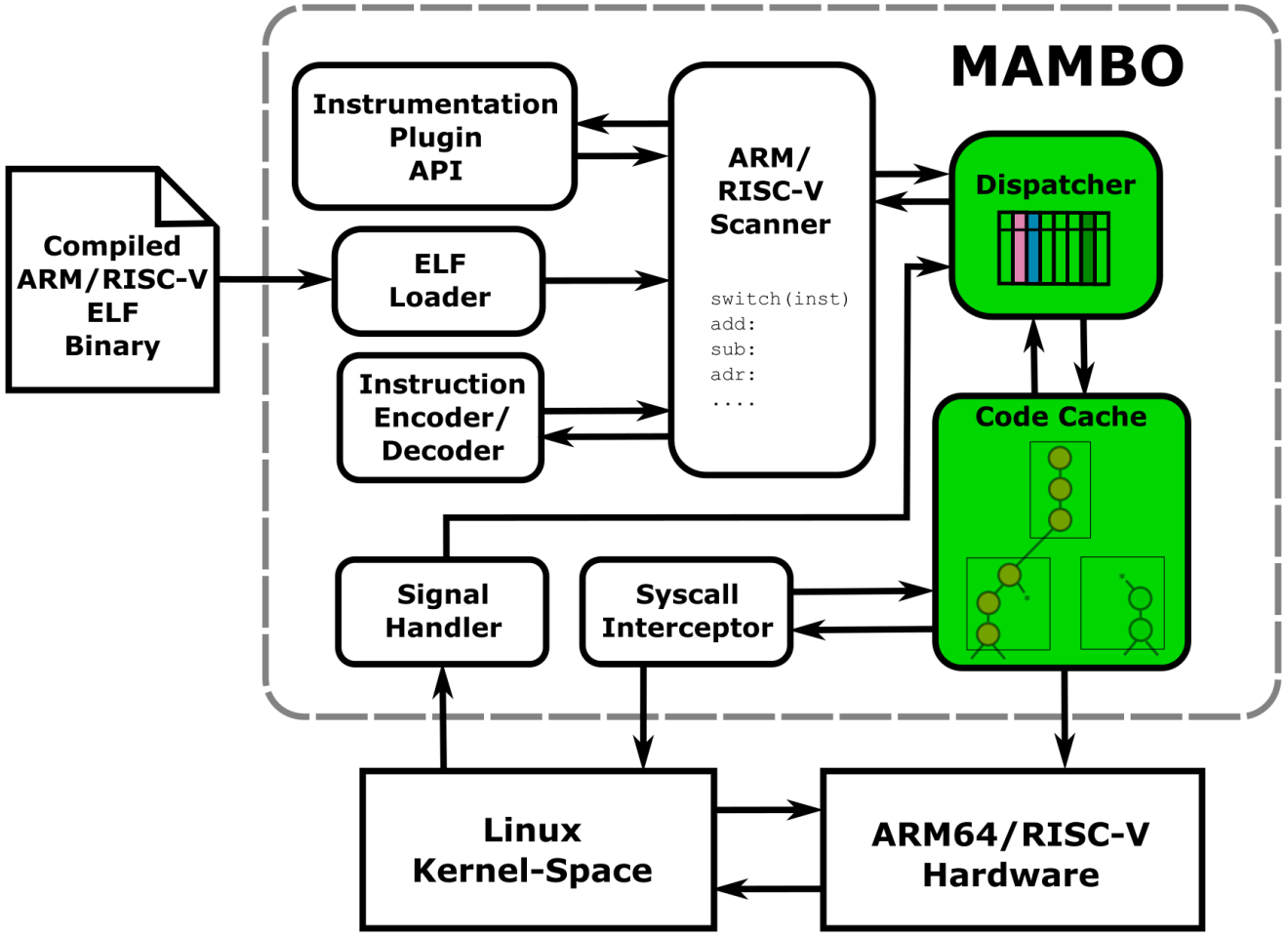
MANCHESTER 1824
The University of Manchester

Basic Block Scanning



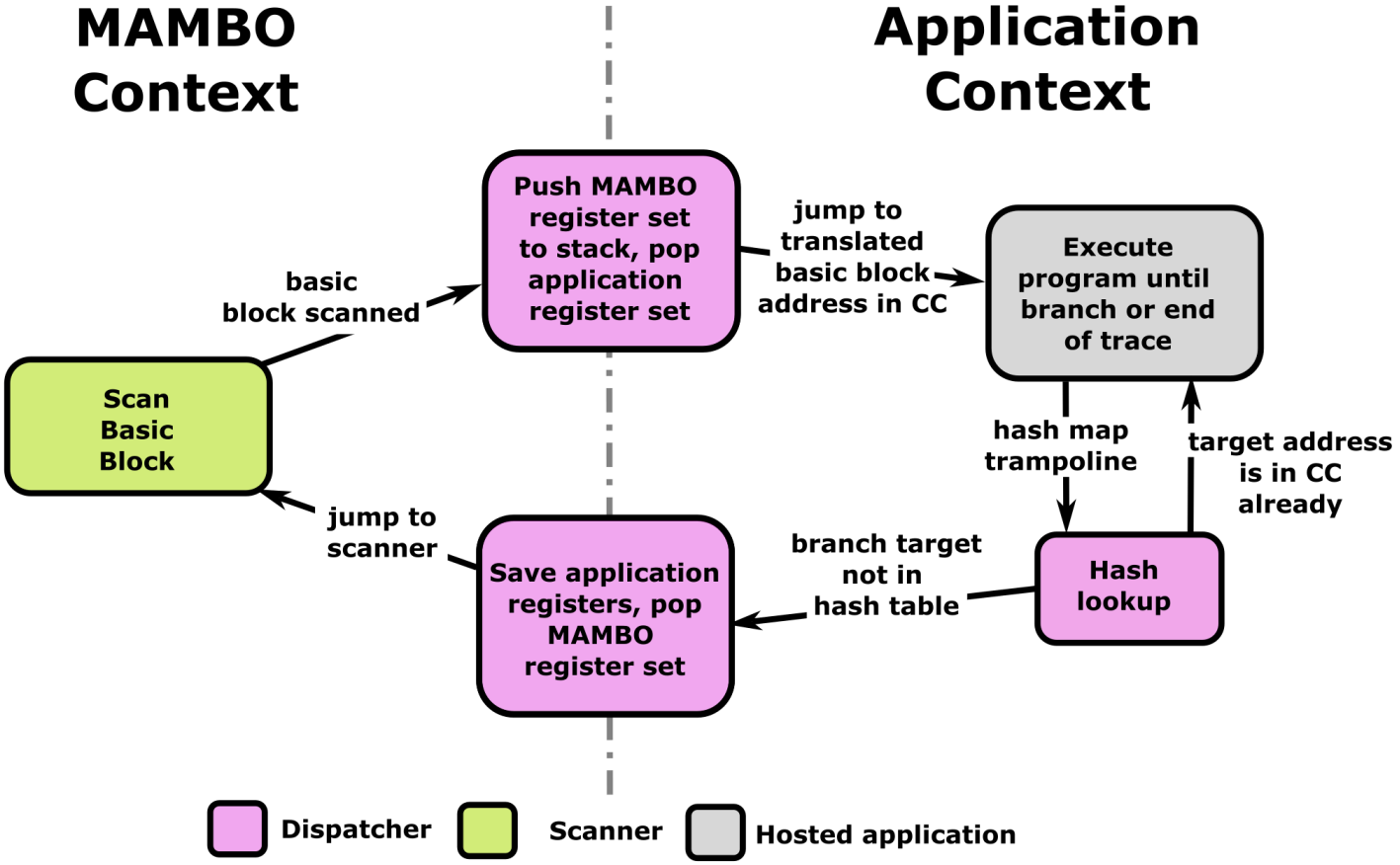
MANCHESTER 1824
The University of Manchester

Dispatch and Code-Cache



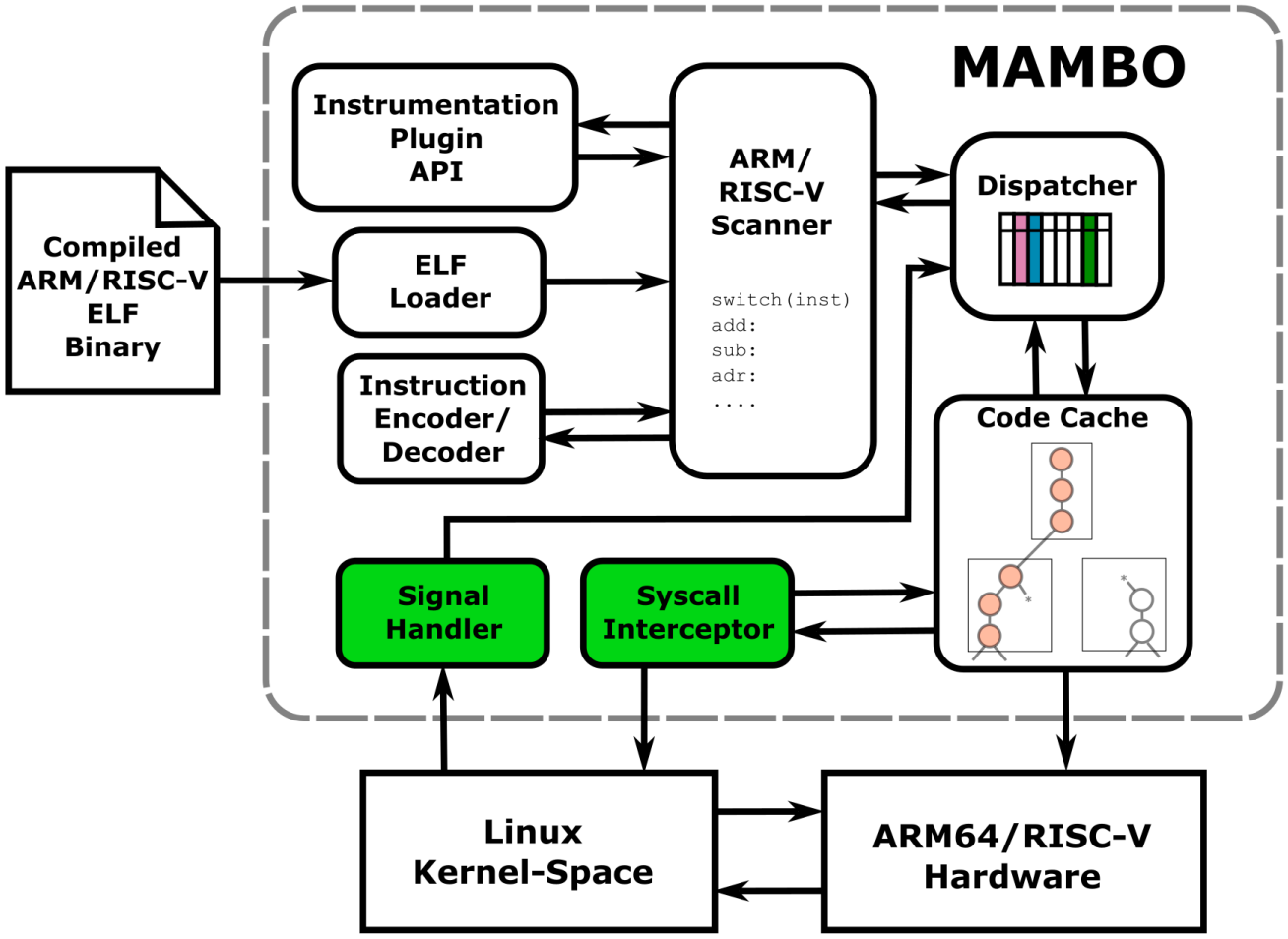
MANCHESTER 1824
The University of Manchester

Dispatch and Code-Cache



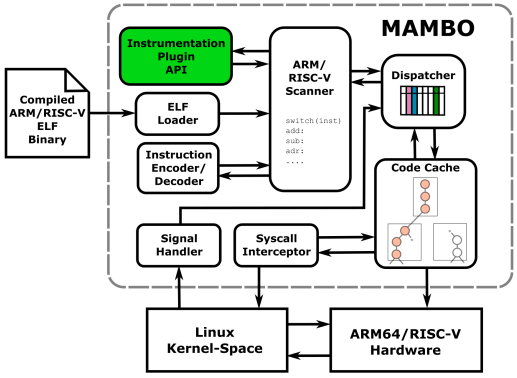
MANCHESTER 1824
The University of Manchester

Kernel Interaction



MANCHESTER 1824
The University of Manchester

Introduction to MAMBO plugin API



MANCHESTER 1824
The University of Manchester

MAMBO Plugins

Example use cases for plugins:

- Code analysis (e.g., building CFG)
- Code generation (e.g., insertion of new functionality)
- Code modification (e.g., reimplementation of library functions)
- Code instrumentation (e.g., performance counters and metrics)
- Runtime event handling (e.g., tracking thread creation/destruction)

MANCHESTER 1824
The University of Manchester

Event Driven Programming Model

- User defined functions are registered as callbacks
- MAMBO executes callbacks when the event is encountered
- Two categories of events:
  - Hosted application runtime events (e.g. system calls).
  - MAMBO scan-time events (used for analysis and instrumentation)
- Fine-grained (e.g., per-instruction) and coarse-grained (e.g., thread creation/destruction) instrumentation

MANCHESTER 1824
The University of Manchester

Event Points

PRE-INSTRUCTION

```
ld a0, -24(s0)
```

POST-INSTRUCTION

MANCHESTER 1824
The University of Manchester

Event Points

PRE-BASIC BLOCK

PRE-INSTRUCTION

```
ld a0, -24(s0)
```

POST-INSTRUCTION

```
ld a0, -24(s0)
mul a0, a0, a0
ld ra, 24(sp)
ld s0, 16(sp)
addi sp, sp, 32
ret
```

POST-BASIC BLOCK

MANCHESTER 1824
The University of Manchester

Event Points

PRE-INSTRUCTION

```
ld a0, -24(s0)
```

POST-INSTRUCTION

PRE-BASIC BLOCK

```
ld a0, -24(s0)
mul a0, a0, a0
ld ra, 24(sp)
ld s0, 16(sp)
addi sp, sp, 32
ret
```

POST-BASIC BLOCK

PRE-THREAD

```
addi sp, sp, -32
sd ra, 24(sp)
sd s0, 16(sp)
addi s0, sp, 32
sd a0, -24(s0)
ld a1, -24(s0)
.Lpcrel_hi0:
auipc a0, %pcrel_hi(.L.str)
addi a0, a0, %pcrel_lo(.Lpcrel_hi0)
call printf@plt
ld a0, -24(s0)
mul a0, a0, a0
ld ra, 24(sp)
ld s0, 16(sp)
addi sp, sp, 32
ret
```

POST-THREAD

PRE-THREAD

```
addi sp, sp, -32
sd ra, 24(sp)
sd s0, 16(sp)
addi s0, sp, 32
sd a0, -24(s0)
ld a1, -24(s0)
.Lpcrel_hi0:
auipc a0, %pcrel_hi(.L.str)
addi a0, a0, %pcrel_lo(.Lpcrel_hi0)
call printf@plt
ld a0, -24(s0)
mul a0, a0, a0
ld ra, 24(sp)
ld s0, 16(sp)
addi sp, sp, 32
ret
```

POST-THREAD

MANCHESTER 1824
The University of Manchester

Event Points

PRE-INSTRUCTION

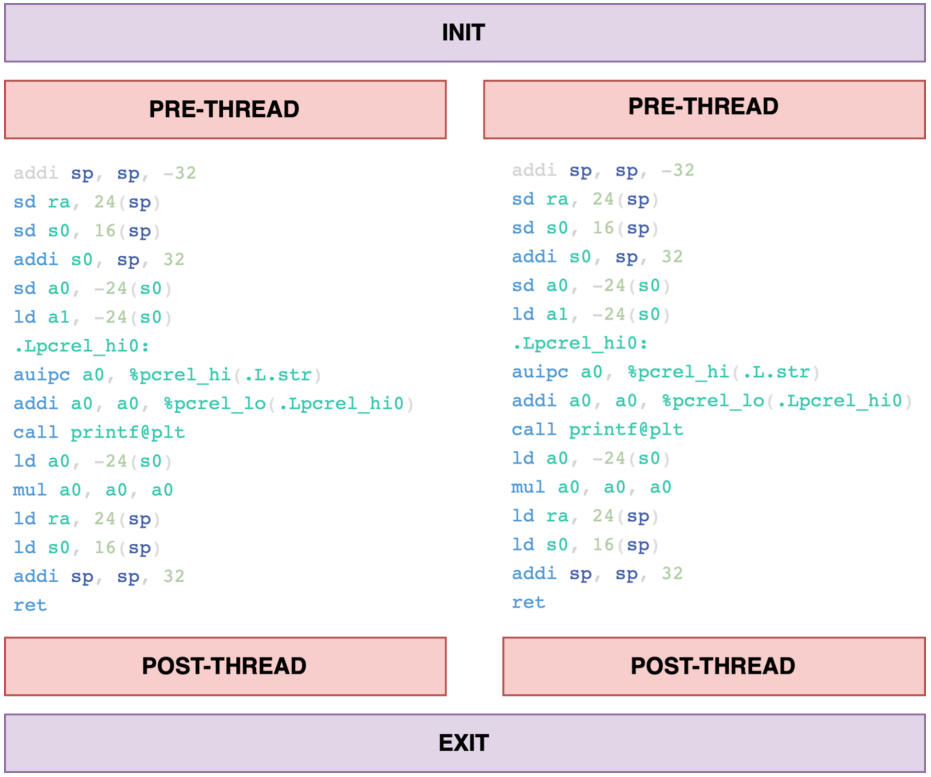
```
ld a0, -24(s0)
```

POST-INSTRUCTION

PRE-BASIC BLOCK

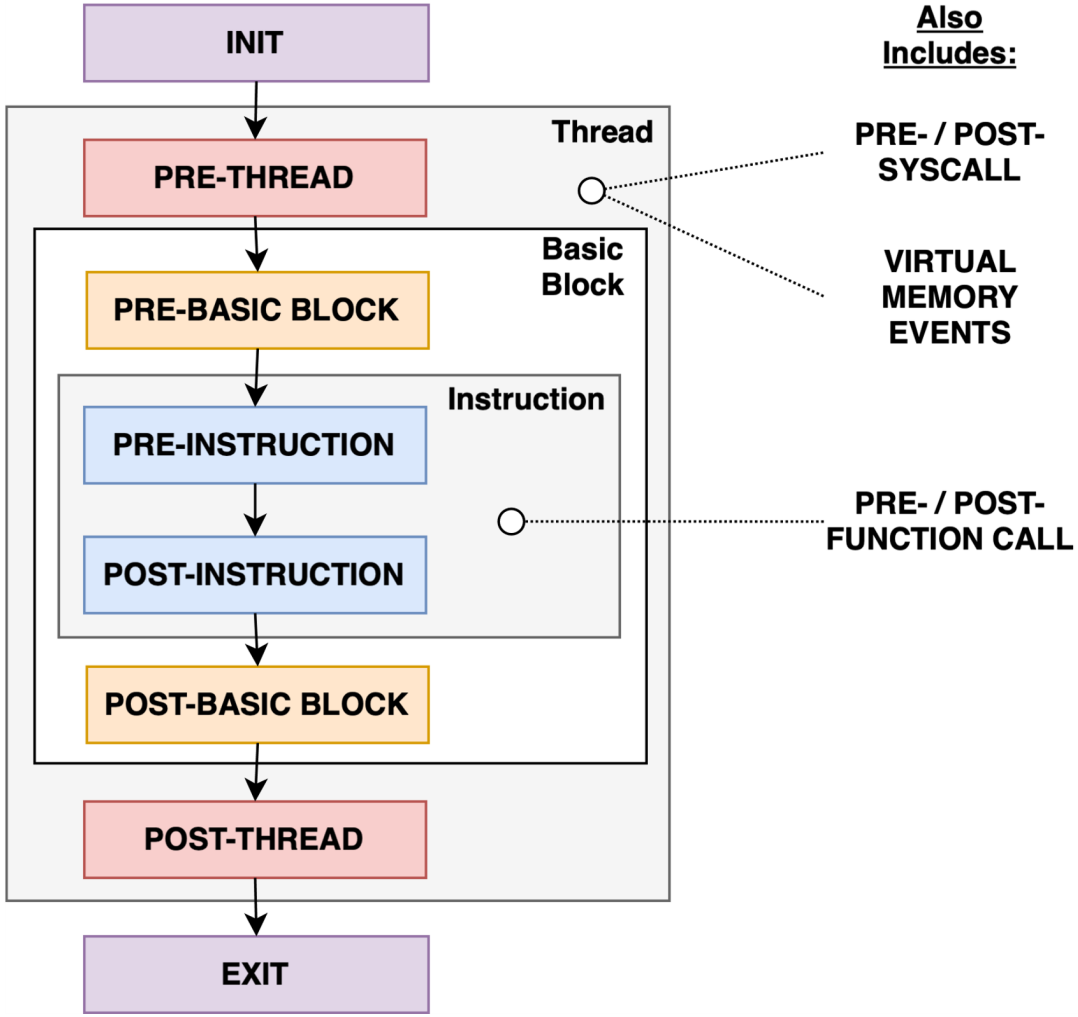
```
ld a0, -24(s0)
mul a0, a0, a0
ld ra, 24(sp)
ld s0, 16(sp)
addi sp, sp, 32
ret
```

POST-BASIC BLOCK



MANCHESTER 1824
The University of Manchester

MAMBO Event Flow



MANCHESTER 1824
The University of Manchester

MAMBO API

- Callback registering functions
- Code analysis
- Instrumentation functions
- Various helper functions

Both architecture dependent and independent functions

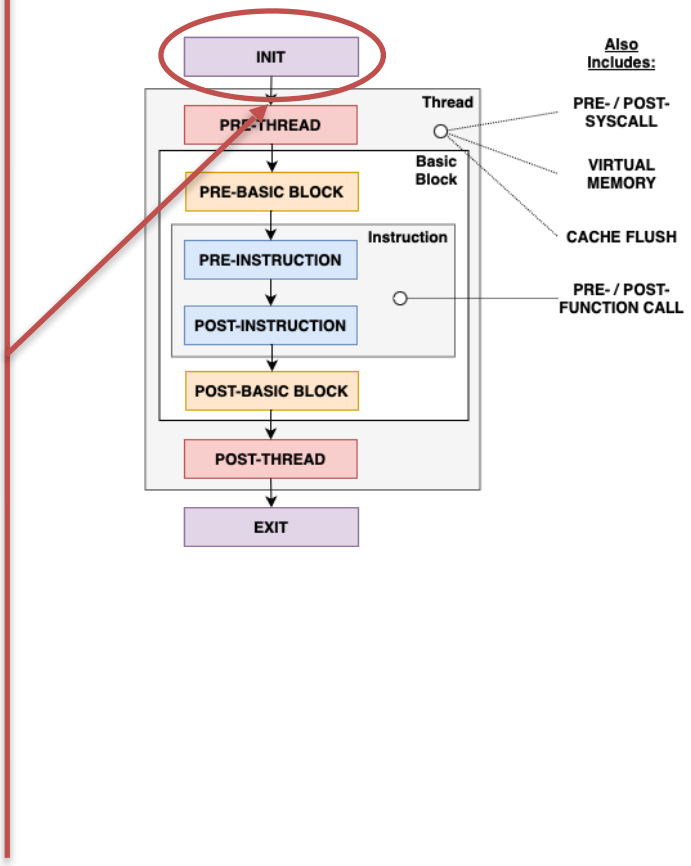
Initialisation Functions Example

1) Initialise plugin

```
__attribute__((constructor))
void <plugin name>() {
    mambo_context * ctx =
        mambo_register_plugin();
    ...
    ...
}
```

2) Register callbacks (examples)

```
int mambo_register_pre_inst_cb(
    mambo_context *ctx,
    &<user function> );

int mambo_register_pre_basic_block_cb(
    mambo_context *ctx,
    &<user function> );

int mambo_register_post_thread_cb(
    mambo_context *ctx,
    &<user function> );
```



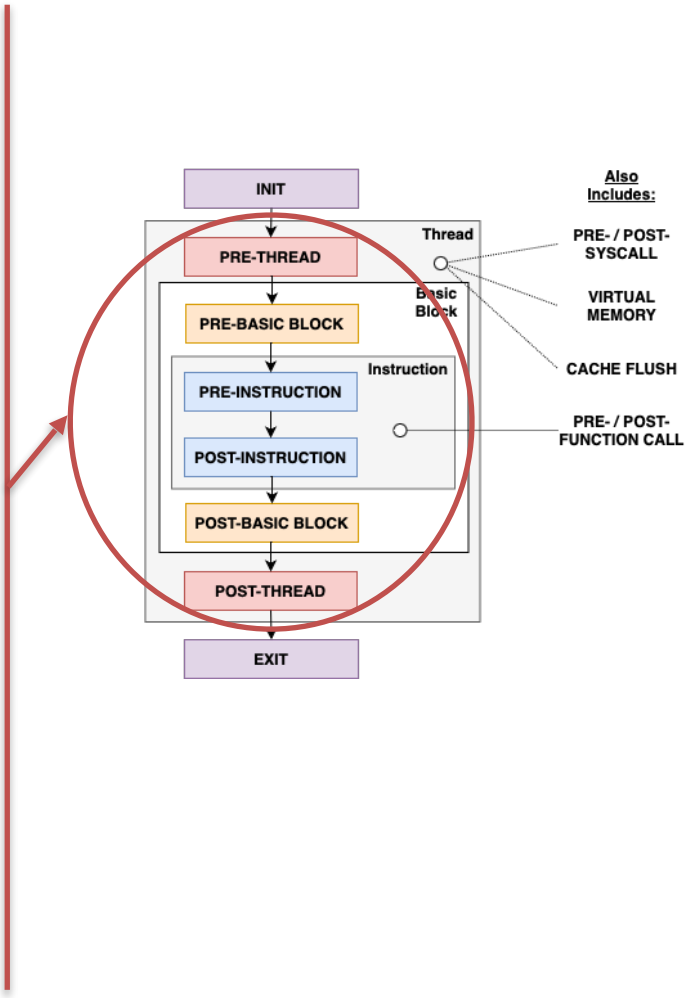
MANCHESTER 1824
The University of Manchester

Initialisation Functions Signature

Code Analysis Functions

Code analysis (examples)

```
mambo_branch_type
mambo_get_branch_type(mambo_context *ctx);

mambo_cond
mambo_get_cond(mambo_context *ctx);

bool
mambo_is_load(mambo_context *ctx);
bool
mambo_is_store(mambo_context *ctx);
```

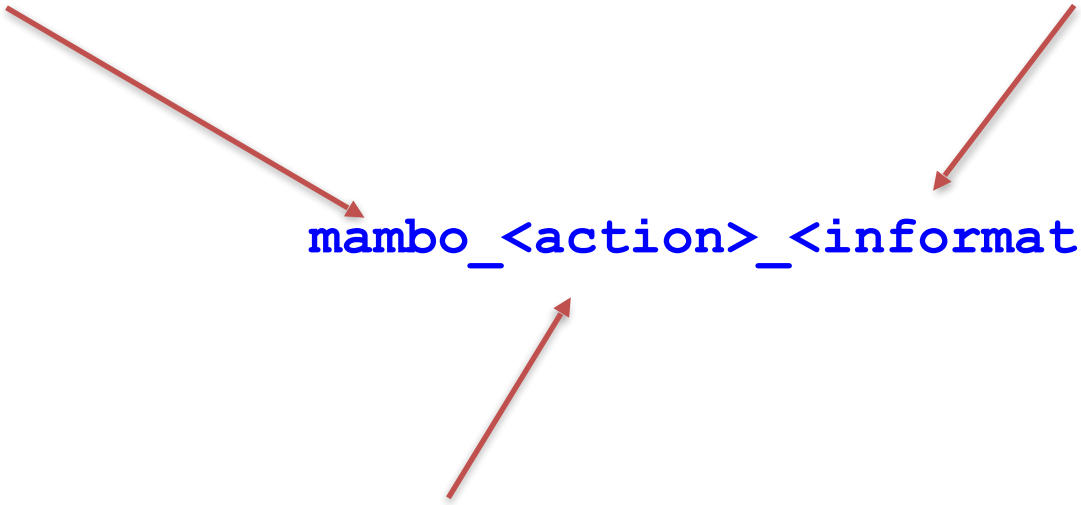


MANCHESTER 1824
The University of Manchester

Analysis Functions Signature

"Namespace"

Type of information being accessed / modified



```
mambo_<action>_<information>
```

Access or set (`get` / `set` / `is`) current state of the execution

MANCHESTER 1824
The University of Manchester

Coding Our First Plugin - `branchplugin`

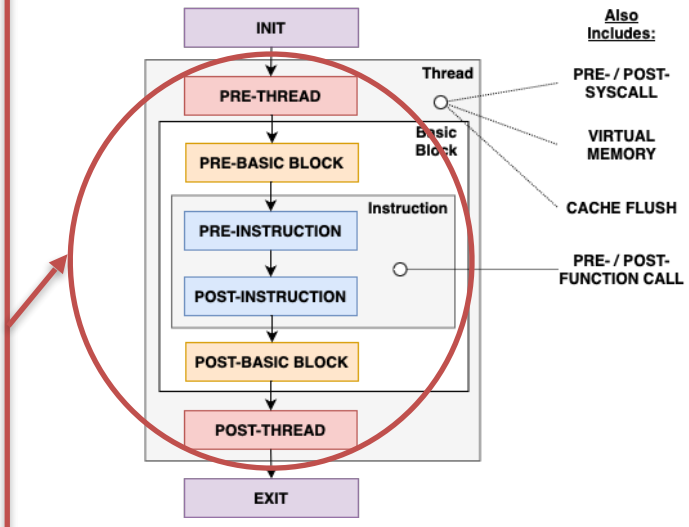
```
__attribute__((constructor))
void branchplugin_init() {
    mambo_context * ctx = mambo_register_plugin();

   mambo_register_pre_inst_callback(ctx, &branchplugin_pre_inst_handler);
}

int branchplugin_pre_inst_handler(mambo_context *ctx) {

    mambo_branch_type type = mambo_get_branch_type(ctx);

    if (type & BRANCH_RETURN) {
        // ...
    } else if (type & BRANCH_DIRECT) {
        // …
    }
    else if (type & BRANCH_INDIRECT) {
        // ...
    }
}
```

MANCHESTER 1824
The University of Manchester

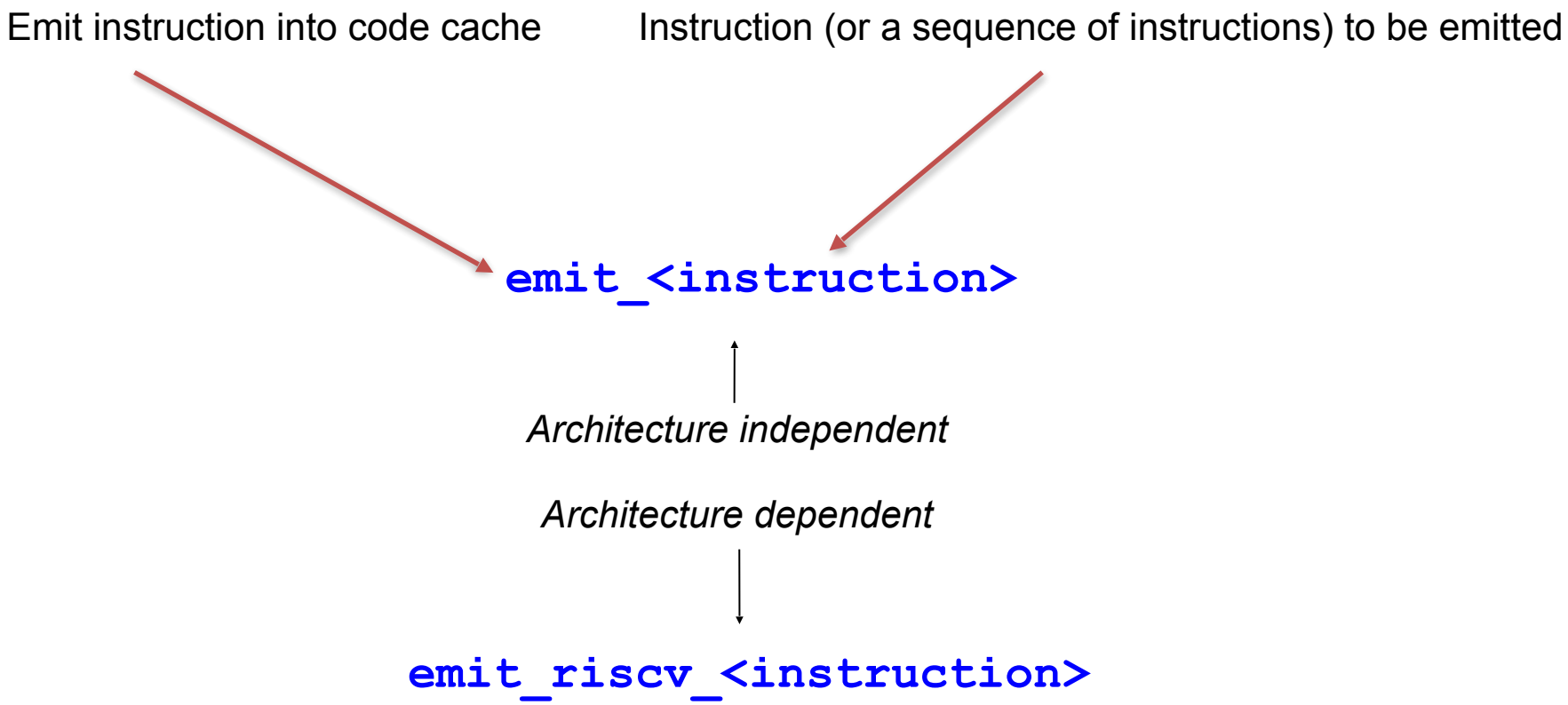
Code Instrumentation Functions

Code instrumentations (examples)

```
void emit_counter64_incr(
    mambo_context *ctx,
    void *counter,
    unsigned incr);

void emit_push(
    mambo_context *ctx, uint32_t regs);
void emit_pop(
    mambo_context *ctx, uint32_t regs);

void emit_set_reg(
    mambo_context *ctx, enum reg reg,
    uintptr_t value);

void emit_fcall(
    mambo_context *ctx,
    void *function_ptr);

int  emit_safe_fcall(mambo_context *ctx,
void *function_ptr,
    int argno);
```



Instrumentation Functions Signature

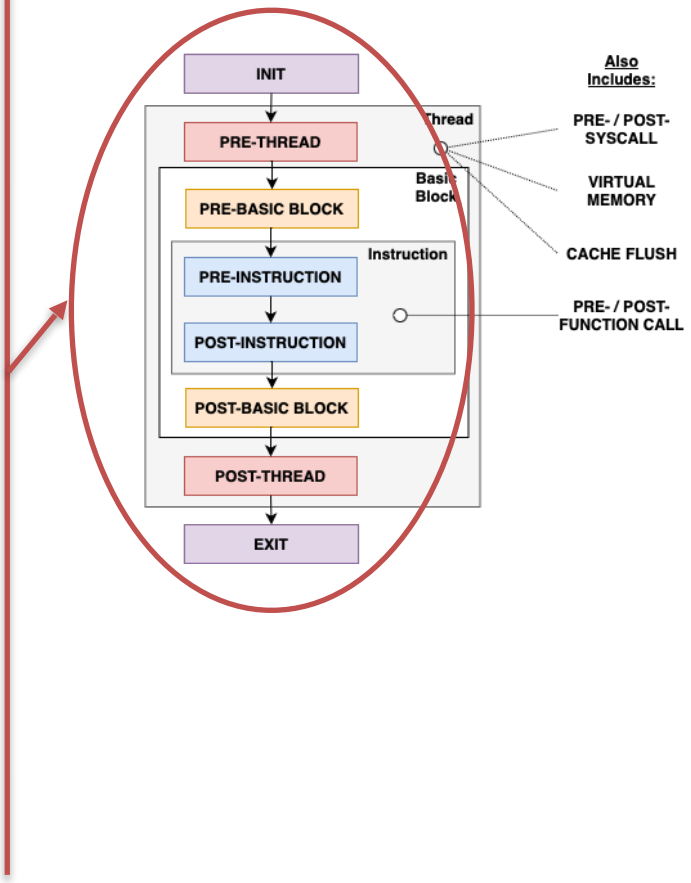


MANCHESTER 1824
The University of Manchester

Additional Helper Functions

Additional Helpers (examples)

```
void mambo_alloc(
    mambo_context *ctx,
    size_t* size);
void mambo_free(
    mambo_context *ctx, void *ptr);

int mambo_ht_init(mambo_ht_t *ht,
    size_tinitial_size, int index_shift,
    int fill_factor, bool allow_resize);
int mambo_ht_add(mambo_ht_t *ht,
    uintptr_t key, uintptr_t value);
int mambo_ht_get(mambo_ht_t *ht,
    uintptr_t key, uintptr_t *value);

int get_symbol_info_by_addrl(
    uintptr_t addr, char **sym_name,
    void **start_addr, char **filename);
```



MANCHESTER 1824
The University of Manchester

Plugin API Scan-Time vs. Runtime

– Important to remember. Most common mistake when first writing plugins:

C:

```
uint64_t run_many_times(uint64_t num) {
 return num * num;
}
```

RISC-V:

```
                run_many_times:
0x8000          mul a0, a0, a0
0x8004          ret
```

Plugin API Scan-Time vs. Runtime

```
char* message = "We are here\n";

int pre_inst_callback(mambo_context ctx*){
 if(ctx->code.read_address == 0x8000) {
    printf(message);
 }
 return 0;
}
```

```
        run_many_times:
0xFC7000   mul a0, a0, a0
0xFC7004   ret
```

```
Output:
We are here!
```

```
char* message = "We are here!\n";

int pre_inst_callback(mambo_context ctx*) {
 if(ctx->code.read_address == 0x8000) {
  emit_push(ctx, (1 << 0));
  emit_set_reg(ctx, reg0, message);
  emit_safe_fcall(ctx, my_print_fn, 1);
  emit_pop(ctx, (1 << 0));
 }
return 0;
}
```

```
        run_many_times:
0xFC7000   addi sp, sp, -8
0xFC7004   sd a0, 0(sp)
0xFC7008   auipc x0, &message
0xFC700B   jal my_print_fn
0xFC7010   ld a0, 0(sp)
0xFC7014   addi sp, sp, 8
0xFC7018   mul a0, a0, a0
0xFC701B   ret
```

```
Output:
We are here!
We are here!
We are here!
We are here!
...
```

MANCHESTER 1824
The University of Manchester

MAMBO Example Plugin

MANCHESTER 1824
The University of Manchester

The following code example can be found at

BEEHIVE-LAB > MAMBO > PLUGINS

MANCHESTER 1824
The University of Manchester

LIVE DEMO

(NOTE: VIM RUNS UNDER RISC-V MAMBO)

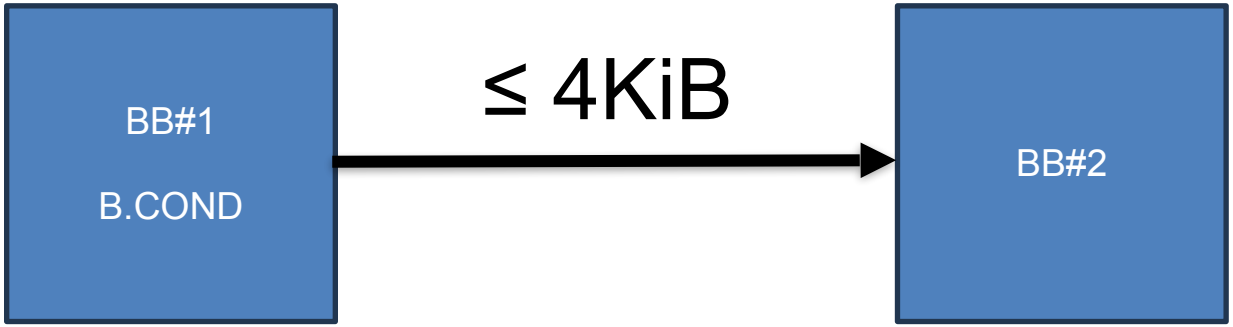
MANCHESTER 1824
The University of Manchester

Lessons Learned from RISC-V Port

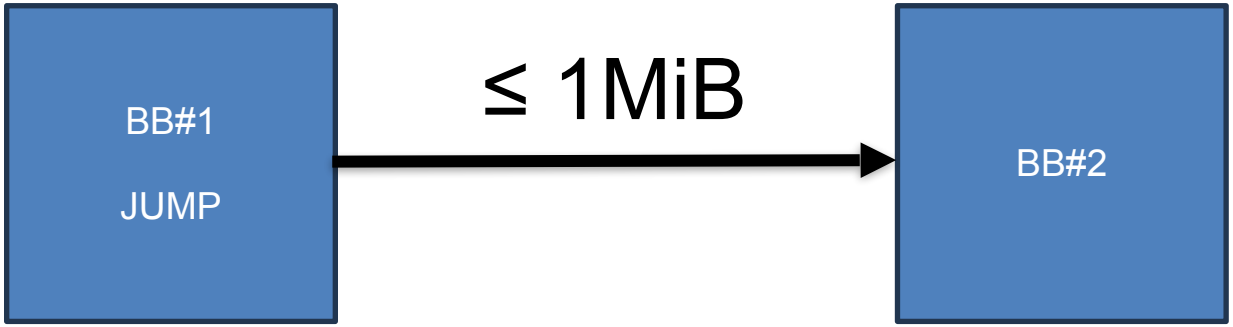
MANCHESTER 1824
The University of Manchester

Lessons Learned From RISC-V Port

Conditional Branch Range:



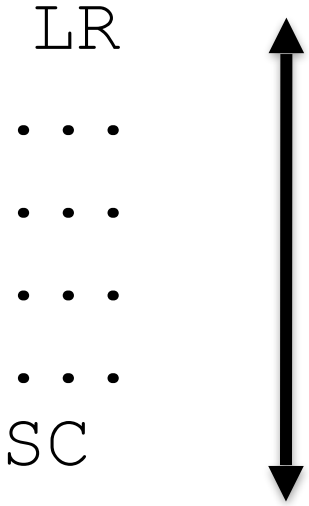
Direct Jump Range:



MANCHESTER 1824
The University of Manchester

Lessons Learned From RISC-V Port

Atomic Load Reserved/Store Conditional:



✔ At most 16 integer instructions between

✘ Other loads and stores between

MANCHESTER 1824
The University of Manchester

Lessons Learned From RISC-V Port

MANCHESTER 1824
The University of Manchester

MAMBO Roadmap

- Foster an open-source community
  - Collaborations/Contributions welcome
- Improve Documentation
- More tools
  - Data race detector
  - Call graph generator
- Keep up with RISC-V (and ARM)
- Current research projects
  - Fast architectural simulation
  - Cybersecurity
  - Binary lifting

MANCHESTER 1824
The University of Manchester

CODE OPEN SOURCE ON GITHUB

BEEHIVE-LAB / MAMBO

(APACHE 2.0 LICENSE)

Thanks!

MoatE (10017512) and Soteria (75243)