Laboratório de Sistemas Embarcados
Embedded Systems Laboratory

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

INTRINSICS-HMC

An automatic Trace-Generator for simulations of Processing-in-Memory Instructions

Aline S. Cordeiro - alinesantanacordeiro@gmail.com
Diego G. Tomé
Tiago R. Kepe
Eduardo C. Almeida
Marco A. Z. Alves

1.

Processing-in-Memory Introduction

Background

- New **P**rocessing-**i**n-**M**emory architectures are emerging;

- PIM is a new technology for processing data in the same chip as DRAM memory.

Background

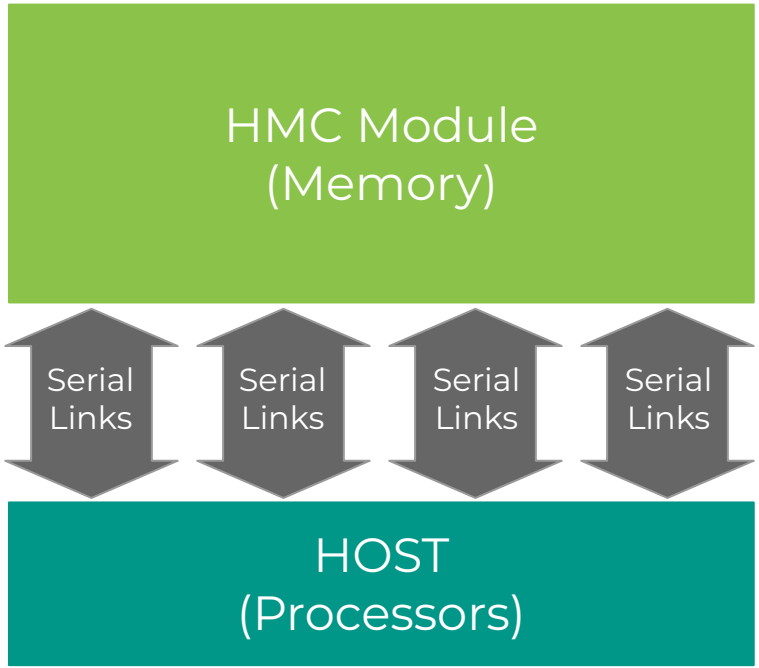
- Due less data transfer, this new concept allows fast processing and less energy consumption;

- Two main projects (HMC / HBM) are already selling this kind of hardware:
  - instructions are executed near the DRAM.

2.

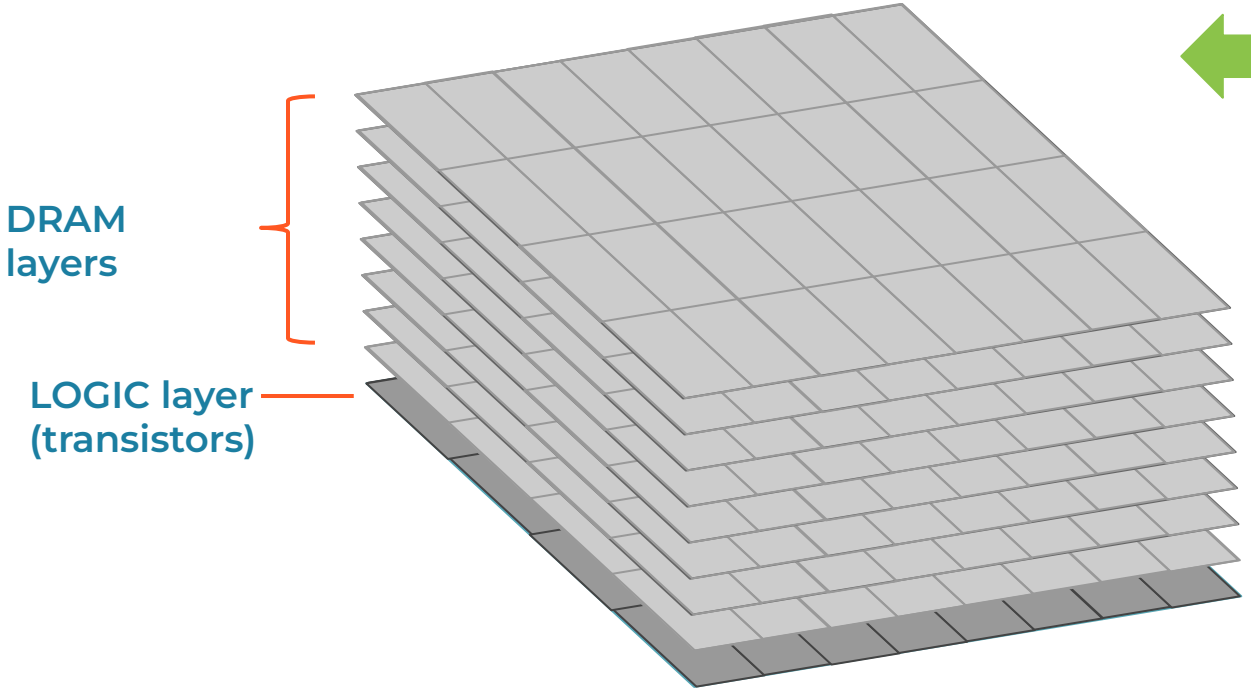
Hybrid Memory Cube

HMC Architecture

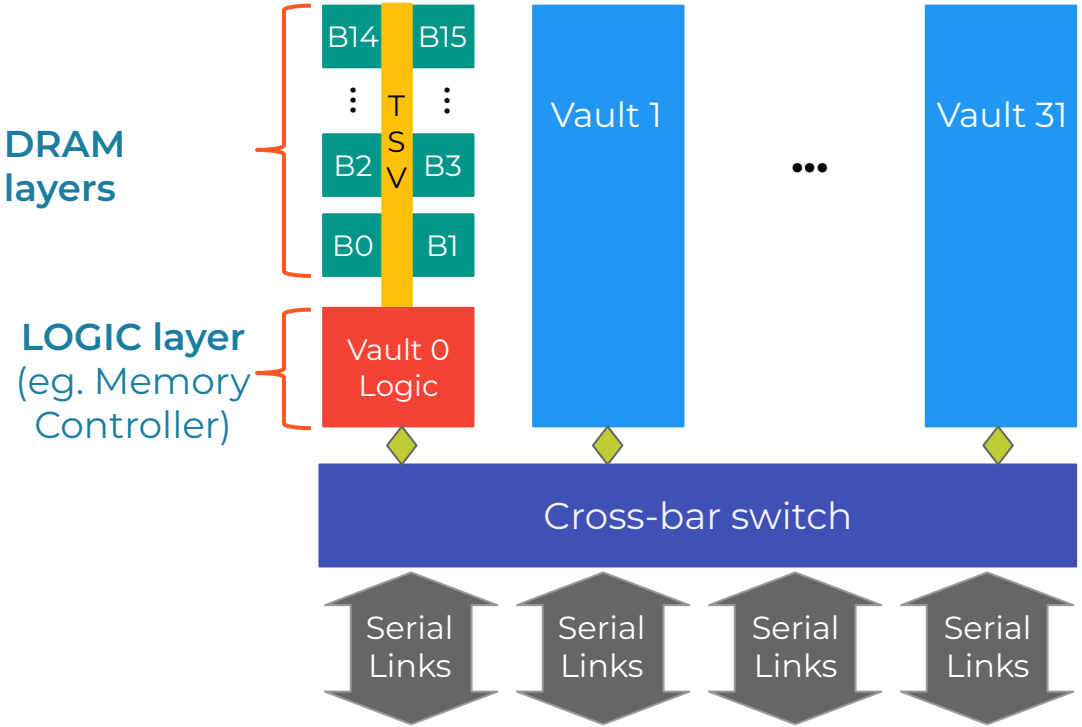


HMC Architecture

HMC Module (Memory)

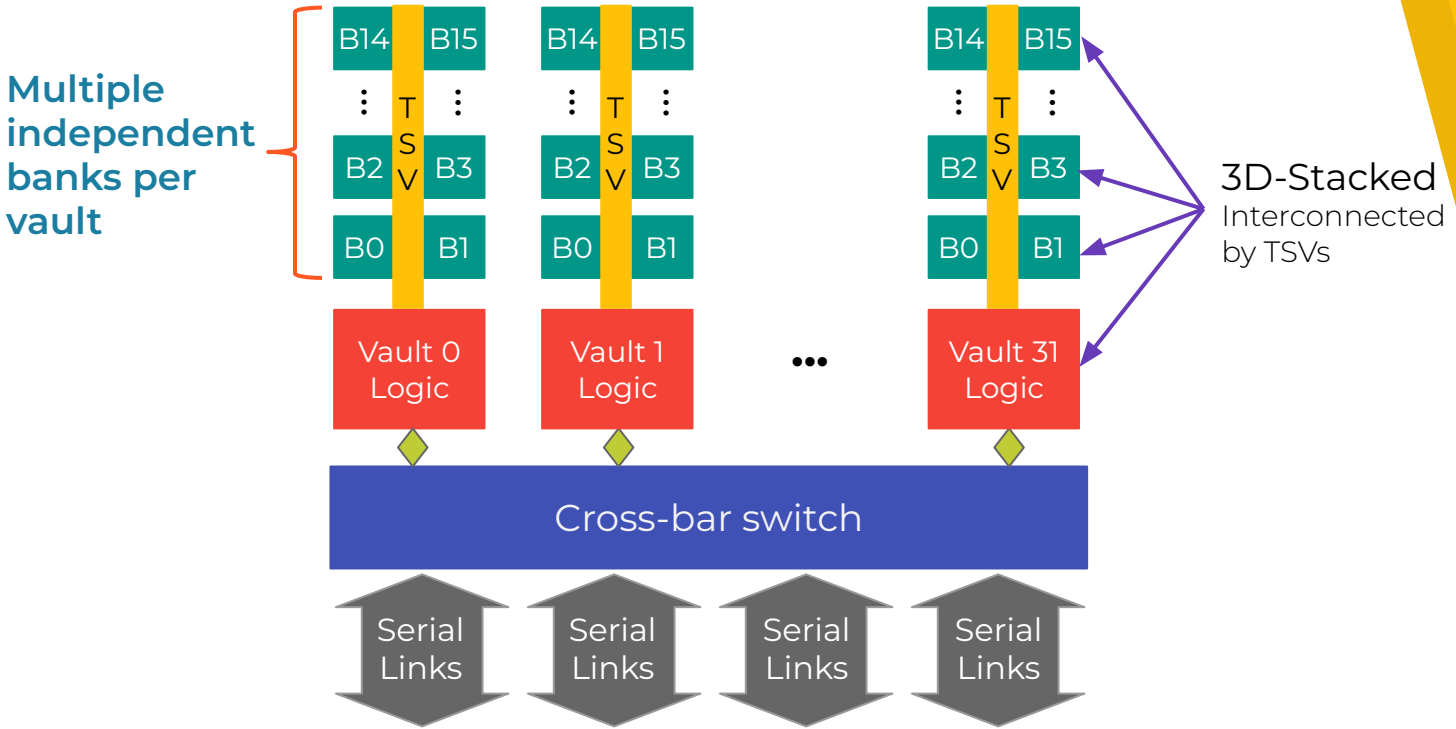


HMC Architecture



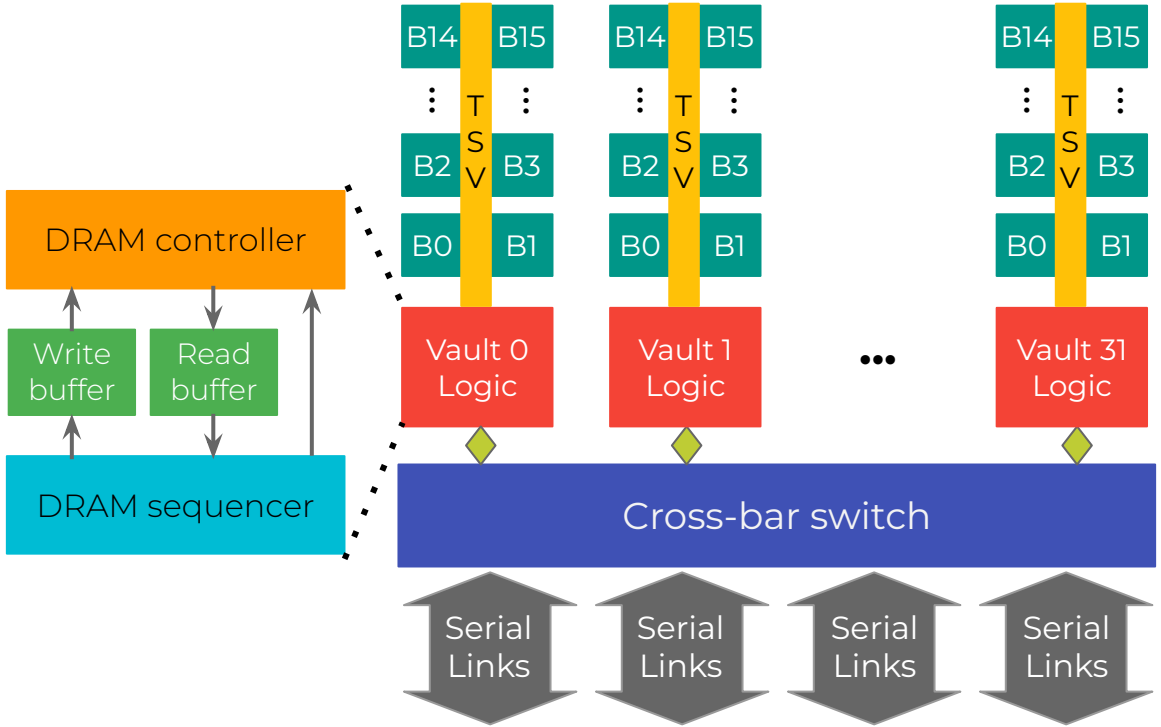
DRAM/logic layers split into **32 logical partitions**

HMC Architecture



3D-Stacked layers using **TSVs**

HMC Architecture

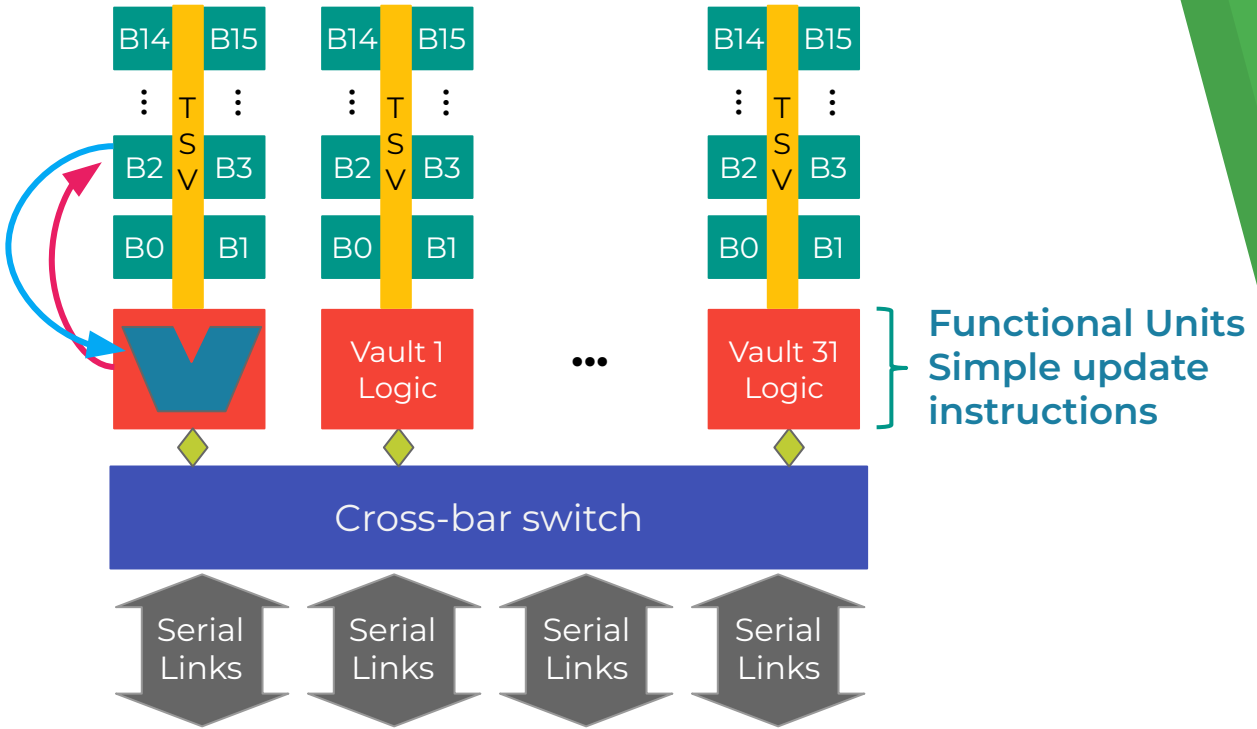


Logic Layer provides **DRAM controller** sending signals (eg. CAS, RAS)

Comparison

- HMC has bandwidth 25% higher than HBM2
  - HMC reaches 320 Gb/s
  - HBM2 reaches 256 Gb/s
- HMC uses a simple and well documented ISA

HMC Architecture



Functional Units
Simple update instructions

Logic Layer also supports **atomic instructions (eg. ADDi, SWP)**

Hybrid Memory Cube **ISA**

- New Processing-in-Memory architectures are emerging;
- HMC uses a simple and well documented ISA
- However, **we need a way to prototype/evaluate new ISA's**

How to evaluate new ISAs?

- Buy a HMC
  - We need the hardware and compiler.
  - No new ISA supported

- Prototype using FPGA or ASIC
  - Time consuming, expensive and error prone

How to evaluate new ISAs? (cont'd)

- Analytical modeling
  - Low precision for complex systems

- **Simulation**
  - **Seems the better option!**

3.

Simulation Overview

Architectural Simulators

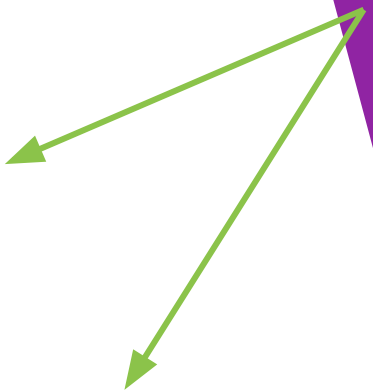
- Full system
  - Executes the binary + OS
  - We need a trusted compiler or hand write the code.

- Trace driven
  - Needs a simulation trace
  - Trace contains dynamic behavior
  - How to write this trace? Manually?

Architectural Simulators

- Full system
  - Executes the binary + OS
  - We need a trusted compiler or hand write the code.

- Trace driven
  - Needs a simulation trace
  - Trace contains dynamic behavior
  - How to write this trace? Manually?



Time consuming and may generate a lot of errors!!!!

Our proposal

Automatically generate simulation traces

- Write high level code (eg. C or C++);
- Using Intrinsics-HMC library;
- Compilation and execution in x86 architectures;

Our proposal

Automatically generate simulation traces

- Write high level code (eg. C or C++);
- Using Intrinsics-HMC library;
- Compilation and execution in x86 architectures;

Our mechanism will convert x86 library functions into HMC instructions

4.

PIM Simulators

Overview of nowadays simulators

Gem5 + SMC Simulation Environment

- Smart Memory Cube is a module inserted in Gem5 simulator;
  - Enhances capabilities of the HMC logic-base;
  - State-of-the-art configuration;

It requires the binary to be executed

Can't achieve the HMC specifications

CasHMC

- Cycle-accurate simulator for HMC;
- Follows the exactly HMC architecture;

Models HMC without PIM capability;

CLAPPS

- Precise Cycle Parallel PIM Simulator;
- Provide a more precise model for PIM architectures;

There is no efficient way to provide HMC input workloads;

Need to integrate with processor simulator to obtain realistic results.

SiNUCA

- Trace-driven simulator;
- Cycle-accurate;
- Based on x86 architecture;
- Easily extensible.
- Models HMC and its PIM capability

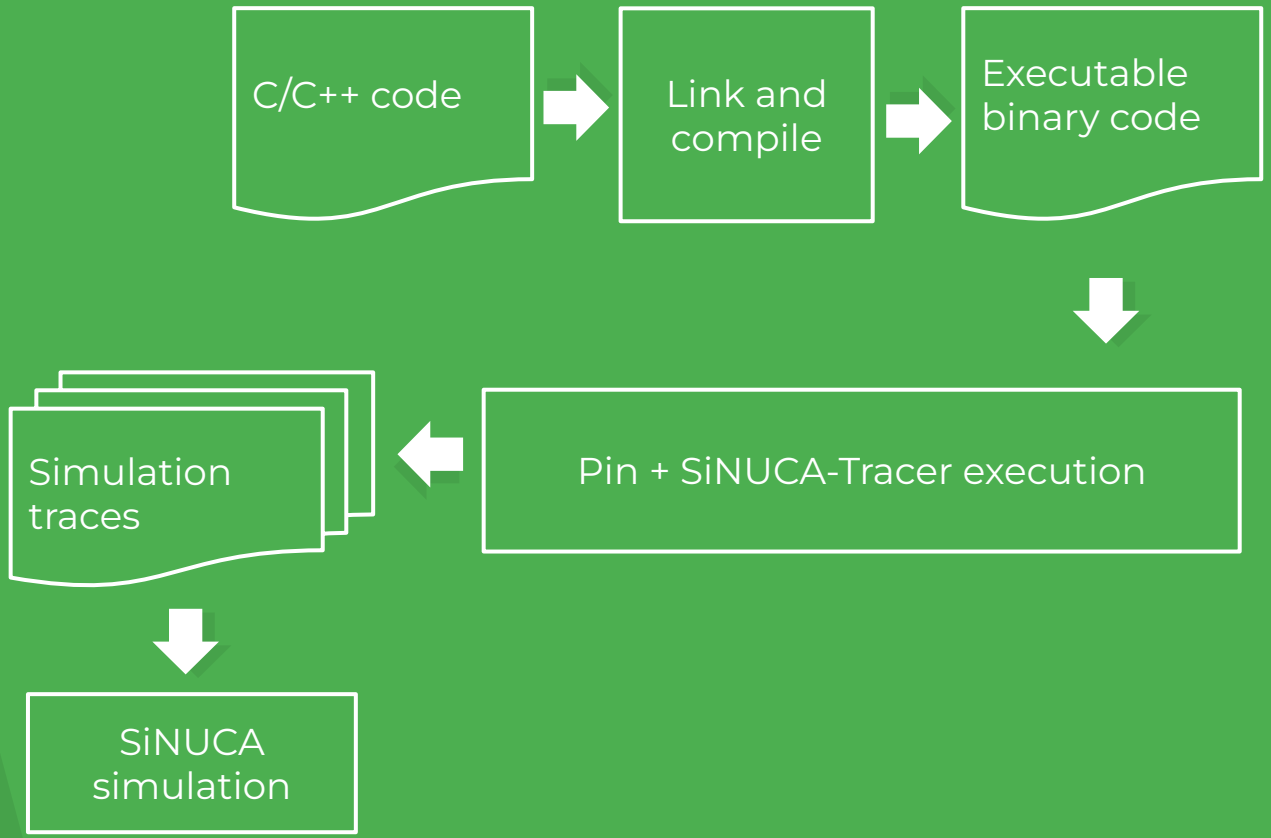
Requires input traces

Summary

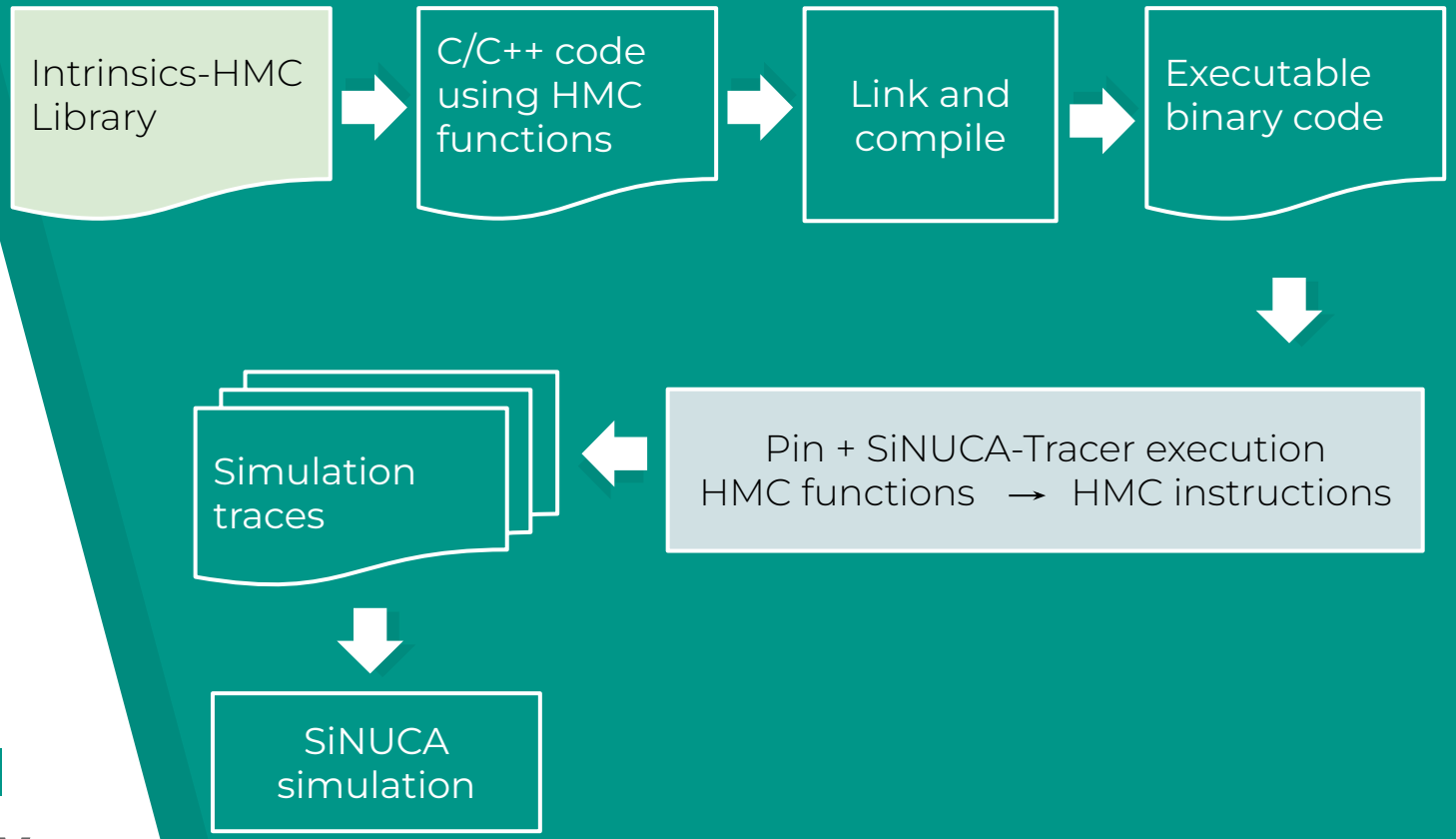
- All simulators need binary ou trace simulation;
- SiNUCA has a trace generator tool;
  - Is used by the group;
  - More convenient;

SiNUCA was used in this work

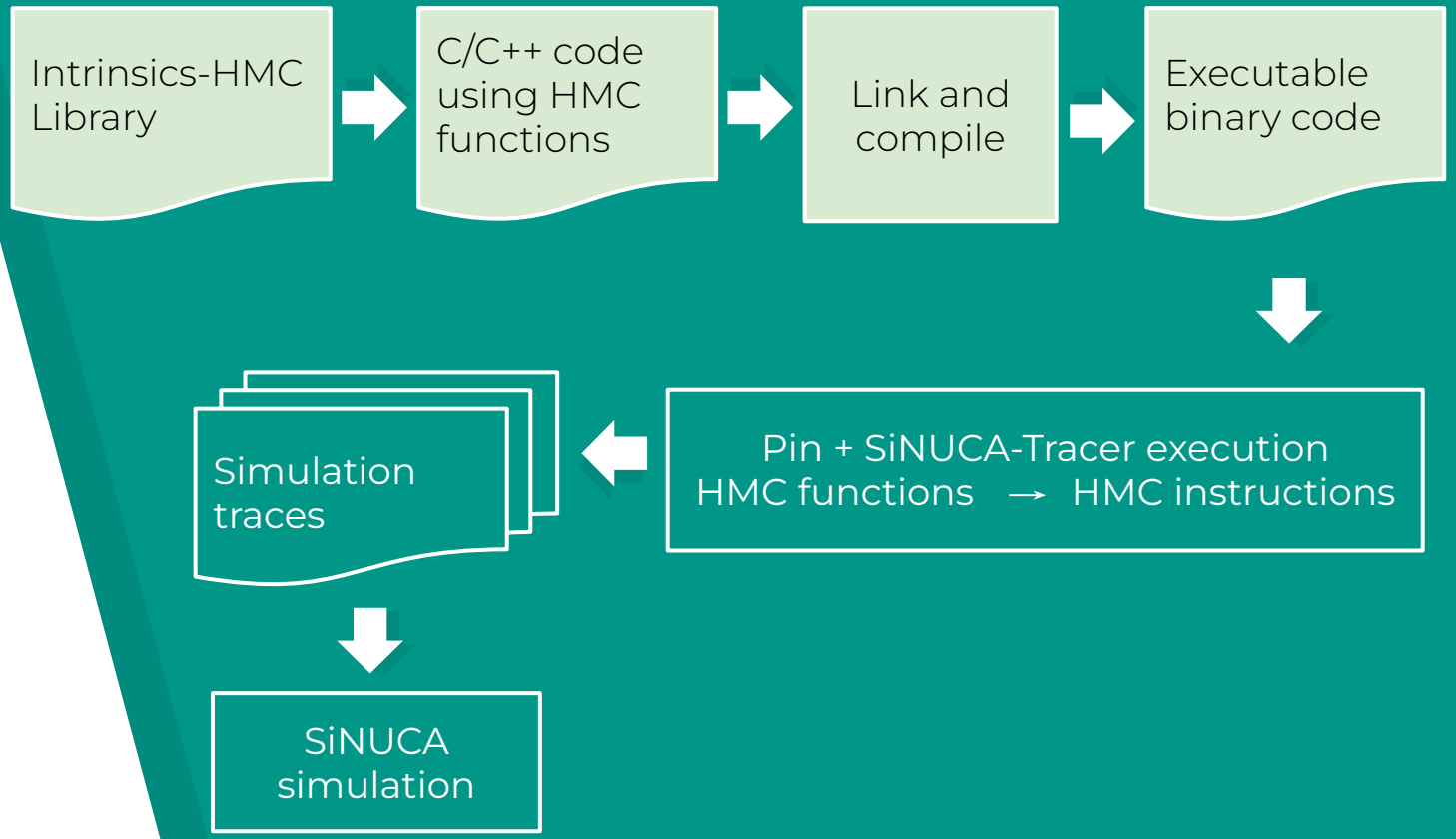
SiNUCA before modification



Proposal Overview



First Step


Intrinsics-HMC Library
C/C++ code using HMC functions
Link and compile
Executable binary code
Pin + SiNUCA-Tracer execution
HMC functions → HMC instructions
Simulation traces
SiNUCA simulation

Intel Intrinsics

- High level code (C/C++);
- Arithmetic, bitwise, math functions, etc;
- Optimization:
  - Vector Instructions (SIMD);
  - SSE, AVX, AVX2

Specification of HMC operations

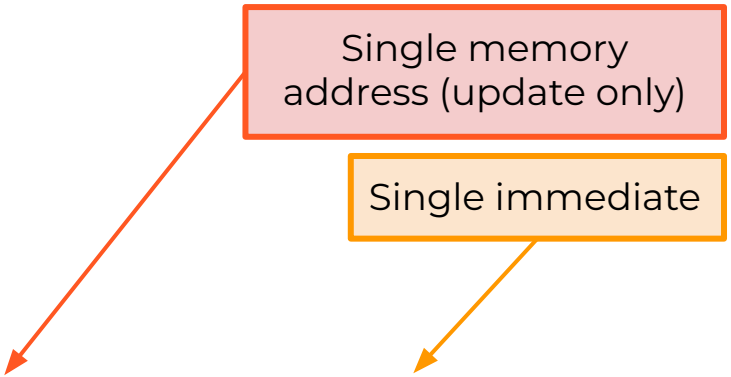
- Traditional Read/Write operations

- Arithmetic;
- Bitwise;
- Boolean;
- Comparison.

Specification of HMC operations

- Traditional Read/Write operations

- Arithmetic;
- Bitwise;
- Boolean;
- Comparison.



Eg.:

nor (memory_address) $immediate

Intrinsics-HMC

- Written in C++ language;
- Reproduce the HMC instructions;
- Data type standardized;
- Inspired in Intel Intrinsics;

https://github.com/AlineS/intrinsics-hmc

Intrinsics-HMC Function call

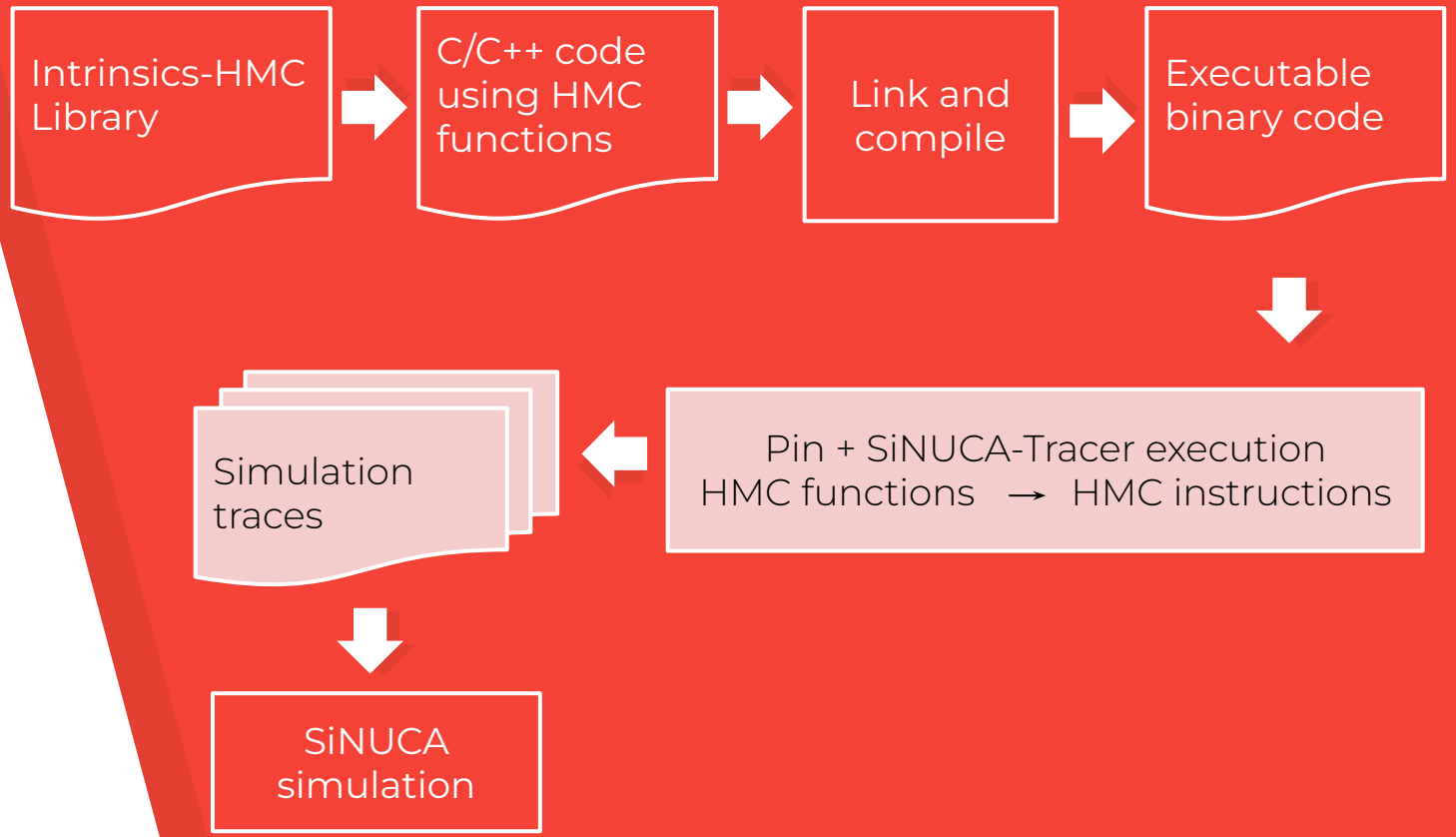
```
#include “../hmc.hpp”

int main(int argc, char *argv[]){
    uint128_t mem_ret;
    mem_ret = _hmc128_nor_s(&mem_op, imm_op);
}
```

Intrinsics-HMC Source code

```
__h128ll1 _hmc128_nor_s
  (__h128ll1 *mem_op, __h128ll1 imm_op){
    __h128ll1 r = *mem_op;
    *mem_op = ~(mem_op | imm_op);
    return r;
}
```

Second Step


Intrinsics-HMC Library
C/C++ code using HMC functions
Link and compile
Executable binary code
Pin + SiNUCA-Tracer execution
HMC functions → HMC instructions
Simulation traces
SiNUCA simulation

5.

SiNUCA-Tracer using Pin (intel)

Trace generator &
Instrumentation tool

Pin

- Intel Instrumentation & analysis tool;
- Provide analysis routines:
  - Pin tools -> **SiNUCA-Tracer**
  - C/C++;
  - Uses binary code.

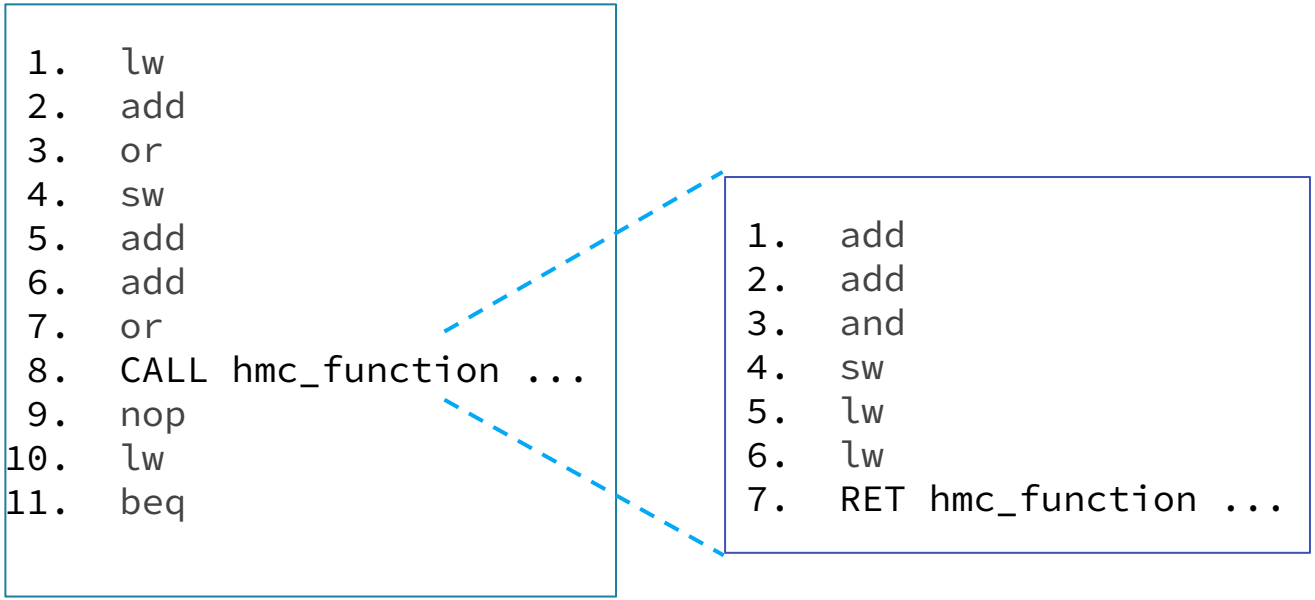
Pin Instrumentation

- Per image;
- Per section;
- Per routine;
- Per basic block;
- Per instructions.

SiNUCA-Tracer

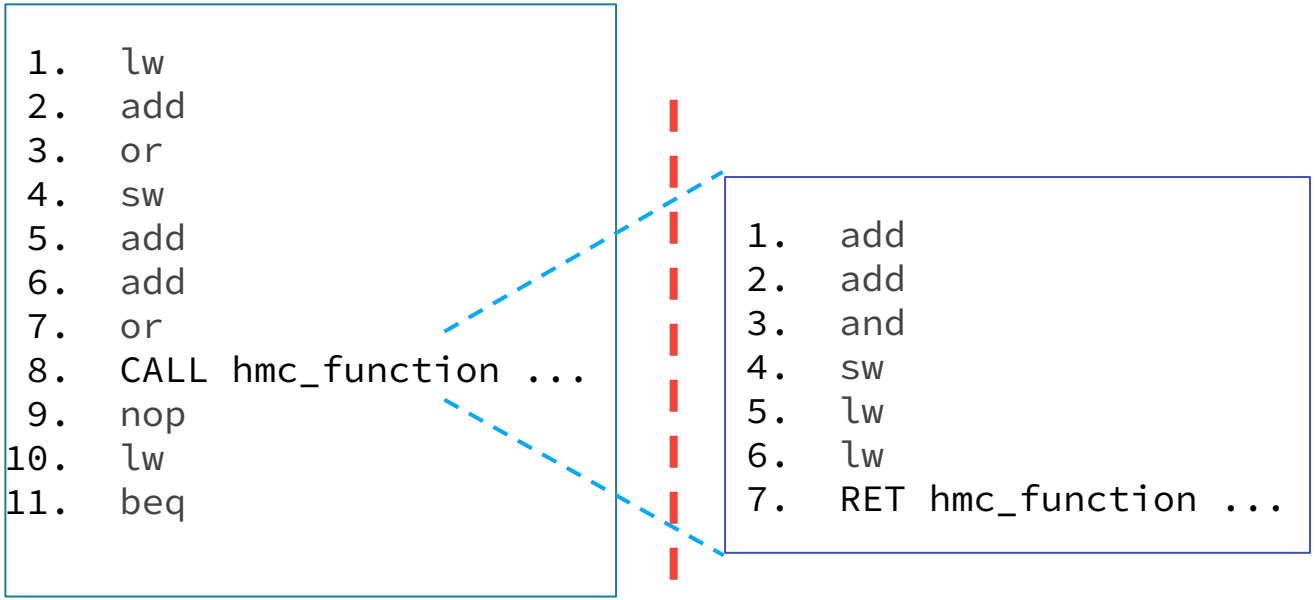
- Generate SiNUCA traces
- Converts x86 ISA ➔ SiNUCA ISA.

Assembly code with Intrinsics-HMC



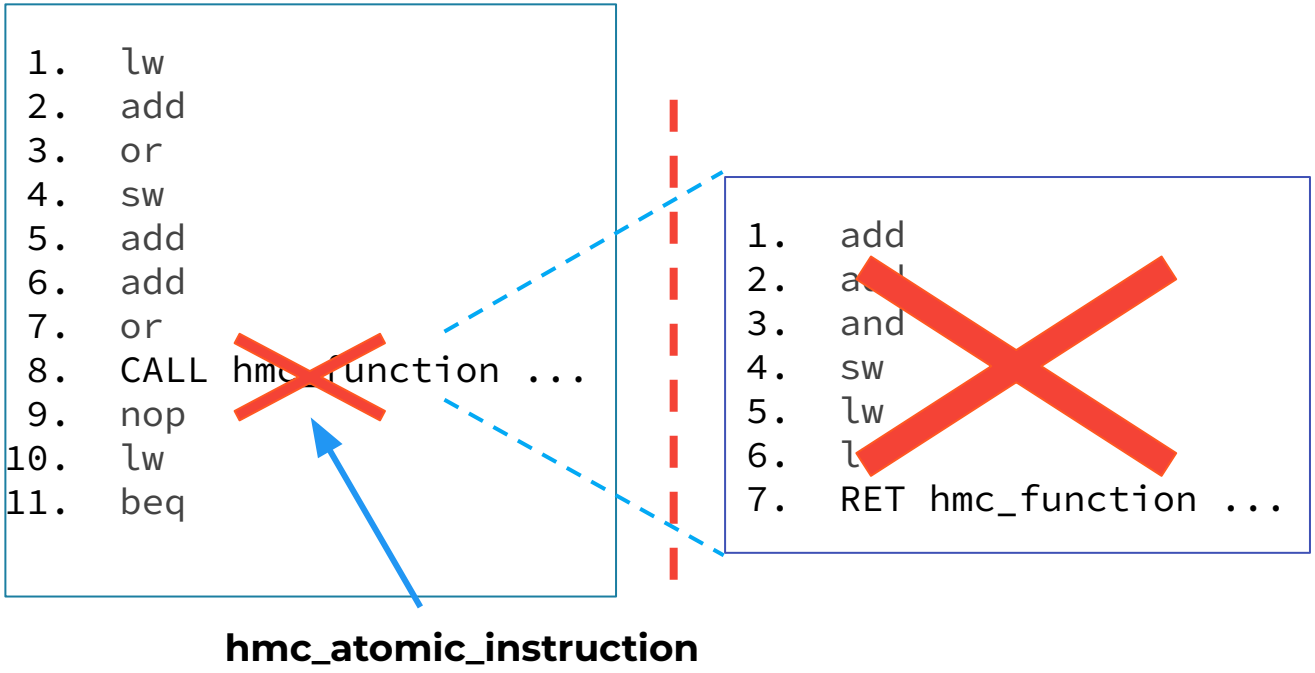
Function call: A block of x86 instructions

Intercept x86 code generation

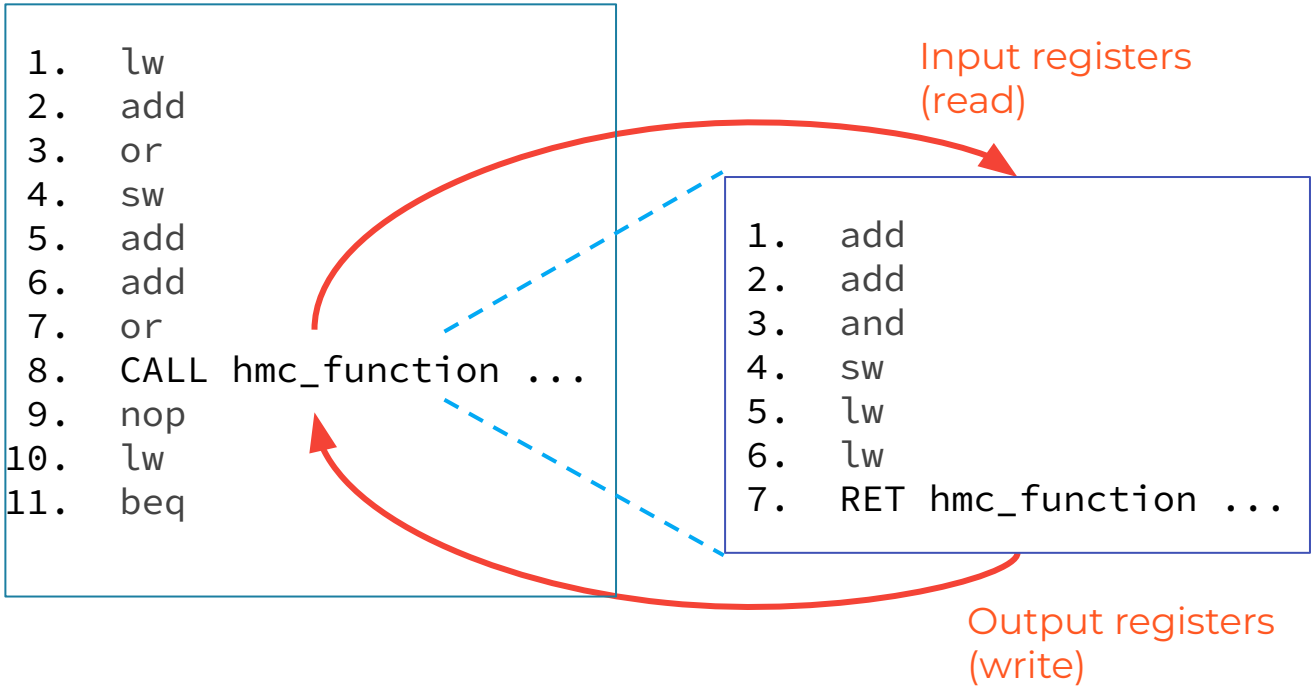


Restrain SiNUCA-Tracer for generating x86 code during HMC function call

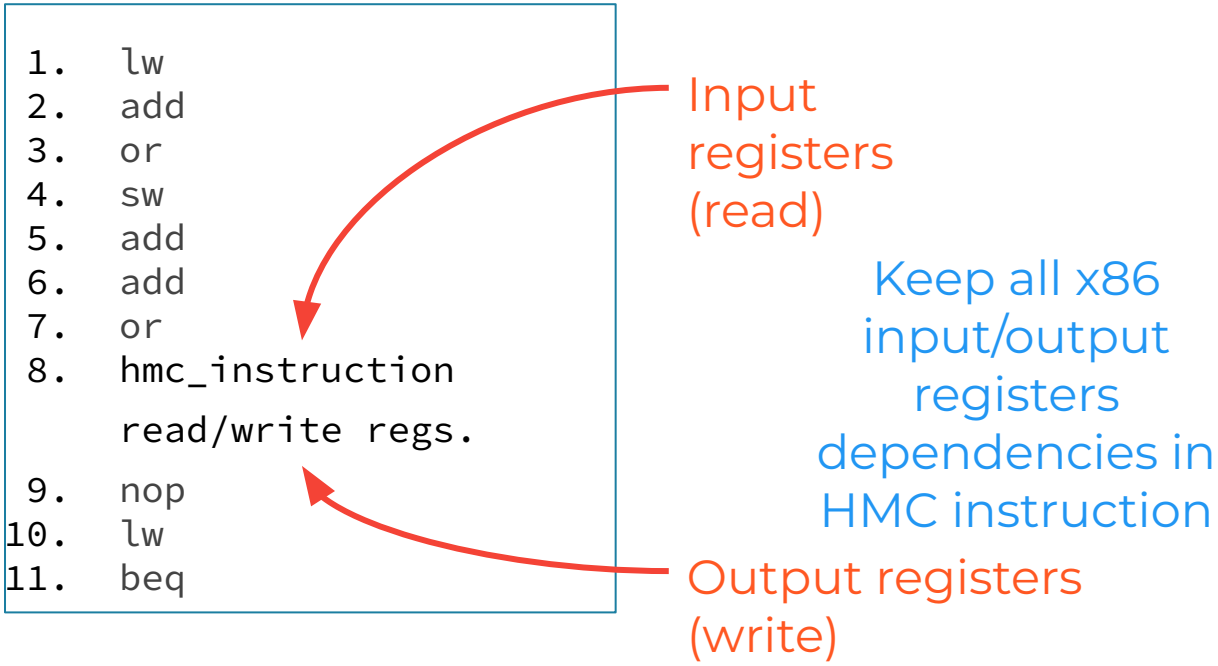
Insert HMC assembly code



x86 Register dependencies



x86 Register dependencies



6.

Results

Benchmarks applications

- Database algorithms:
  - Join;
  - Select Scan;

Data streaming behavior

Suitable to exploit PIM

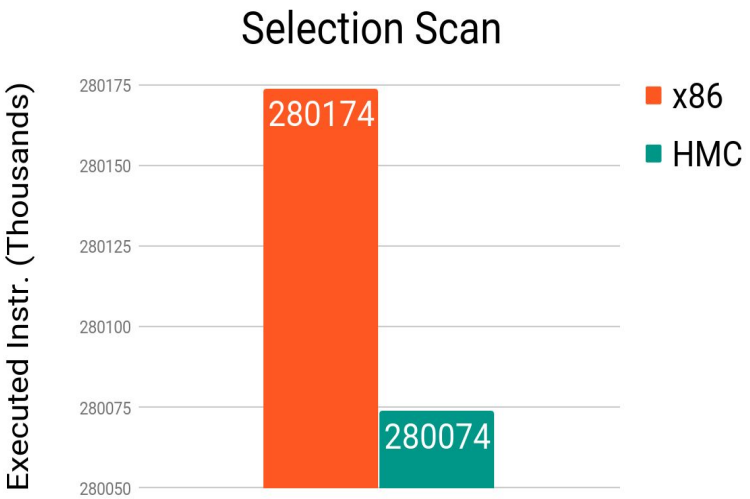
Join - Nested Loop

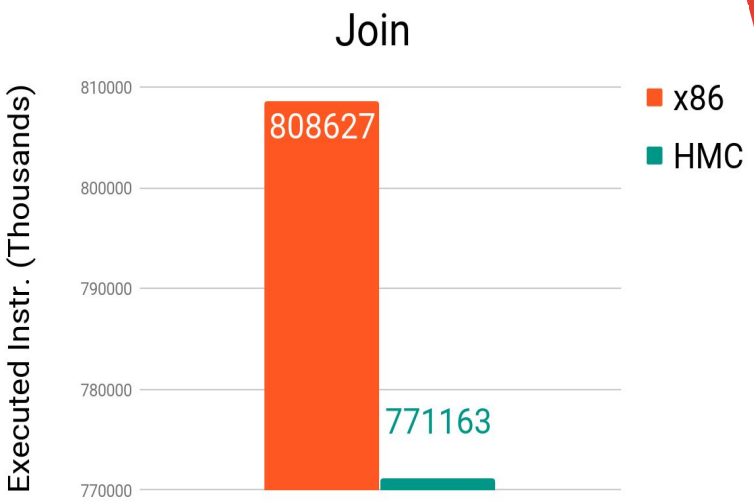
- Cross two tables looking for equal values;
- Uses mainly 2 nested loops.

Select Scan - Array search

- Selects all values in a table that match with an specific input;
- Go through an array

Results

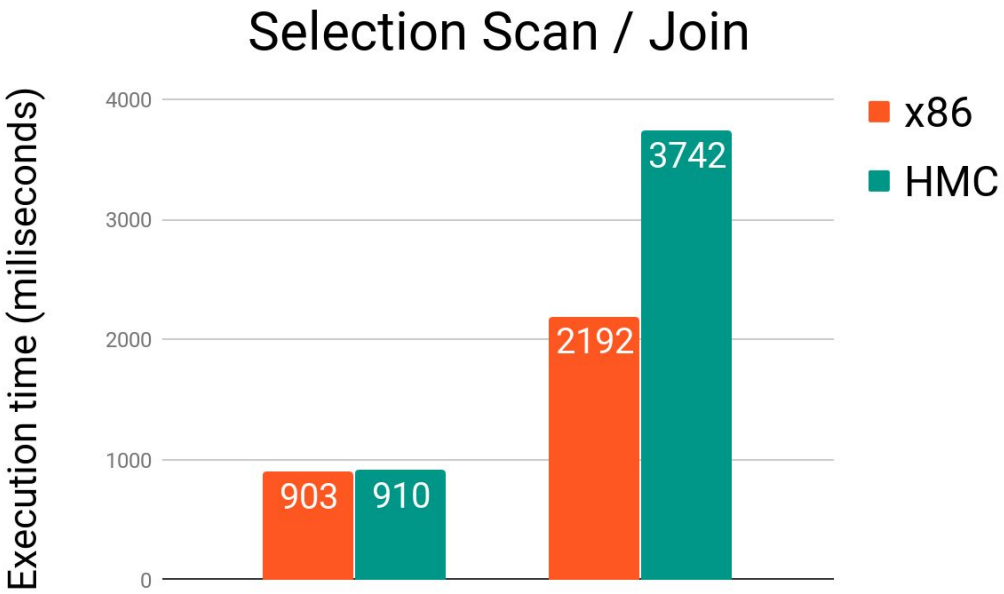




Expected reduction of instruction

1 function call:
1 instruction

Results



Execution time increased

HMC has an limited ISA

7.

Conclusion

Conclusion

- We can generate automatic HMC simulation traces;
- Easily extensible intrinsics library to support new ISA's;

Conclusion

- Enabling fast evaluations of new PIM architectures;
- Now we can evaluate complex apps with PIM.

Laboratório de Sistemas Embarcados
Embedded Systems Laboratory

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

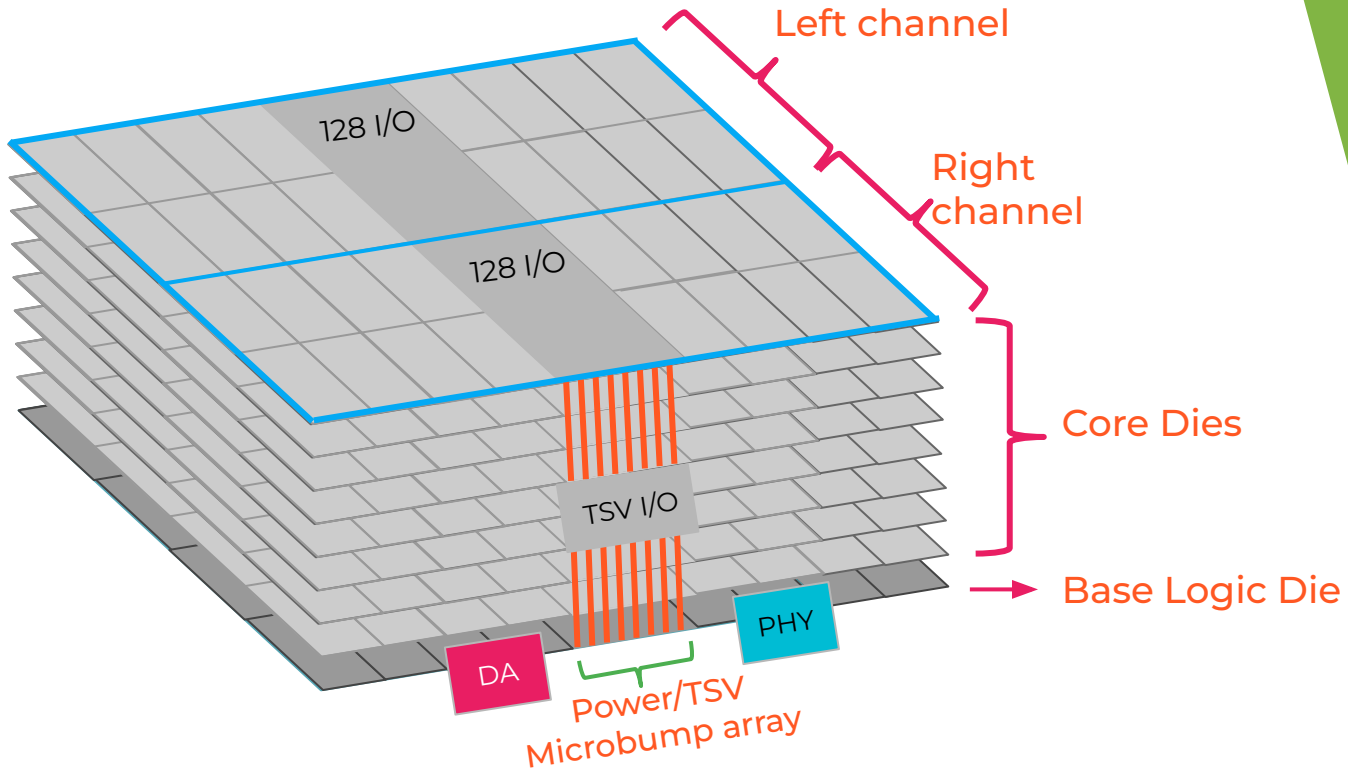
INTRINSICS-HMC

An automatic Trace-Generator for simulations of Processing-in-Memory Instructions

Thank you!

Aline S. Cordeiro - alinesantanacordeiro@gmail.com
Diego G. Tomé
Tiago R. Kepe
Eduardo C. Almeida
Marco A. Z. Alves

High Bandwidth Memory
Architecture



Tá certo?

4.

SiNUCA

Simulator of Non-Uniform Cache Architectures

State-of-the-art components

- Multi-core and multi-processor support;
- Out-of-order processor models;
- Branch predictors;
- Cache line prefetch;
- Memory controller and DRAM with main signals (DDR-3);
- Network-on-Chip.

Technologies modeled

- NoC with contention modeling
- NUCA cache (multi-banked LLC)
- NUMA systems
- Integrated memory controller model

Input traces format

Static traces:

```
1  #main
2  @1
3  MOV 1 0x95727 4 1 14 1 34 14 0 1 0 0 0 0 0
4  #main
5  @2
6  MOV 8 0x95717 3 1 14 1 65 14 0 1 0 0 0 0 0
7  ADD 1 0x95720 3 2 14 65 1 34 14 0 1 0 1 0 0 0
8  ADD 1 0x95723 4 1 14 1 34 14 0 1 0 1 0 0 0
9  CMP 1 0x95727 4 1 14 1 34 14 0 1 0 0 0 0 0
10 JBE 7 0x95731 2 2 35 34 1 35 0 0 0 0 0 1 0 0
```

Input traces format

Dynamic traces:

```
1  1
2  2
3  2
```

Memory traces:

```
1   R 4 0x1701448 1
2   #
3   R 4 0x1701448 2
4   R 4 0x1701452 2
5   W 4 0x1701452 2
6   R 4 0x1701448 2
7   W 4 0x1701448 2
8   R 4 0x1701448 2
9   #
10  R 4 0x1701448 2
11  R 4 0x1701452 2
12  W 4 0x1701452 2
13  R 4 0x1701448 2
14  W 4 0x1701448 2
15  R 4 0x1701448 2
```

Specification of HMC definitions

- Memory and immediate operands;
- Operands with 8/16 bytes length;
- Instructions with varied operands and length operands.

Specification of HMC definitions

- Memory and immediate operands;
- Operands with 8/16 bytes length;
- Instructions with varied operands and length operands.

Information standardized to fit in Intrisics-HMC library

Intrinsics-HMC data types

- __h16s1
- __h64l1
- __h64l2
- __h128ll1

- "__h" denotes HMC ISA;
- "16/64/128" indicates data length in bits;
- "s" refers to short unsigned;
- "l" refers to long unsigned;
- "ll" refers to long long unsigned;
- "1/2" indicates the number of variables allocated.

SiNUCA-Tracer

- Generate traces from x86 ISA to SiNUCA:
  - Static, dynamic and memory traces;

- Open image;
  - Open routines;
    - Open basic blocks;

Functionality

Binary code is “watched” and:

- Every routine and each basic block from it is recorded in static trace;

Functionality

Binary code is “watched” and:

- Every routine and each basic block from it is recorded in static trace;
- Basic blocks execution sequence are recorded in dynamic trace;

Functionality

Binary code is “watched” and:

- Every routine and each basic block from it is recorded in static trace;
- Basic blocks execution sequence are recorded in dynamic trace;
- Every memory instruction from the basic blocks, are recorded in memory trace.

Identifying HMC functions

- We are interested in intrinsics-HMC routines:
  - To translate them to HMC instructions;

Identifying HMC functions

- We are interested in intrinsics-HMC routines:
  - To translate them to HMC instructions;

- HMC routine has a set of x86 asm operations

Identifying HMC functions

- We are interested in intrinsics-HMC routines:
  - To translate them to HMC instructions;

- HMC routine has a set of x86 asm operations
- A HMC instructions is equivalent to and has only 3 HMC asm operation:
  - Load, HMC operation and store

Translating HMC functions

- Translate HMC functions to correct HMC instructions;

Translating HMC functions

- Translate HMC functions to correct HMC instructions;
  - Assuming the code was executed by a computer with support to HMC ISA.

Translating HMC functions

- Translate HMC functions to correct HMC instructions;
  - Assuming the code was executed by a computer with support to HMC ISA.

Compiler create register dependencies

Register dependencies

- Input and output dependencies;
- Each HMC instruction has a list of register dependencies of:
  - Input registers;
  - Output register; and
  - Index registers;

```
1  #_Z17_hmc128_saddimm_s
2  @3430
3  PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
4  MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
5  SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
6  MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
7  MOV 9 179 4 1 5 4 0 5 0 0 0 1 3 0 0 0
8  MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
9  MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
10 MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
11 XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
12 MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
13 MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
14 MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
15 MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
16 MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
17 CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
18 #_Z17_hmc128_saddimm_s
19 @3474
20 MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
21 MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
22 XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
23 JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
24 #_Z17_hmc128_saddimm_s
25 @3476
26 LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
27 RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0

28 #_hmc128_saddimm_s
29 @1
30 HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

HMC function:
add 16 bytes memory and immediate operands

How it works?

```
1  #_Z17_hmc128_saddimm_s
2  @3430
3  PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
4  MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
5  SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
6  MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
7  MOV 9 179 4 2 5 4 0 5 0 0 0 1 3 0 0 0
8  MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
9  MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
10 MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
11 XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
12 MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
13 MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
14 MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
15 MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
16 MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
17 CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
18 #_Z17_hmc128_saddimm_s
19 @3474
20 MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
21 MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
22 XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
23 JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 4 0 0 0
24 #_Z17_hmc128_saddimm_s
25 @3476
26 LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
27 RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0

28 #_hmc128_saddimm_s
29 @1
30 HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

x86 asm code

```
1  #_Z17_hmc128_saddimm_s
2  @3430
3  PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
4  MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
5  SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
6  MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
7  MOV 9 179 4 1 5 4 0 5 0 0 0 1 3 0 0 0
8  MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
9  MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
10 MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
11 XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
12 MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
13 MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
14 MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
15 MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
16 MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
17 CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
18 #_Z17_hmc128_saddimm_s
19 @3474
20 MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
21 MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
22 XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
23 JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
24 #_Z17_hmc128_saddimm_s
25 @3476
26 LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
27 RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0
```

x86 asm code

HMC asm equivalent code

```
28 #_hmc128_saddimm_s
29 @1
30 HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

```
1  #_Z17_hmc128_saddimm_s
2  @3430
3  PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
4  MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
5  SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
6  MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
7  MOV 9 179 4 2 5 4 0 5 0 0 0 1 3 0 0 0
8  MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
9  MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
10 MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
11 XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
12 MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
13 MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
14 MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
15 MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
16 MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
17 CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
18 #_Z17_hmc128_saddimm_s
19 @3474
20 MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
21 MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
22 XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
23 JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
24 #_Z17_hmc128_saddimm_s
25 @3476
26 LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
27 RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0
```

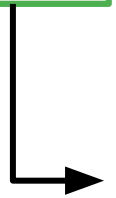
x86 asm code

HMC asm equivalent code

```
28 #_hmc128_saddimm_s
29 @1
30 HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

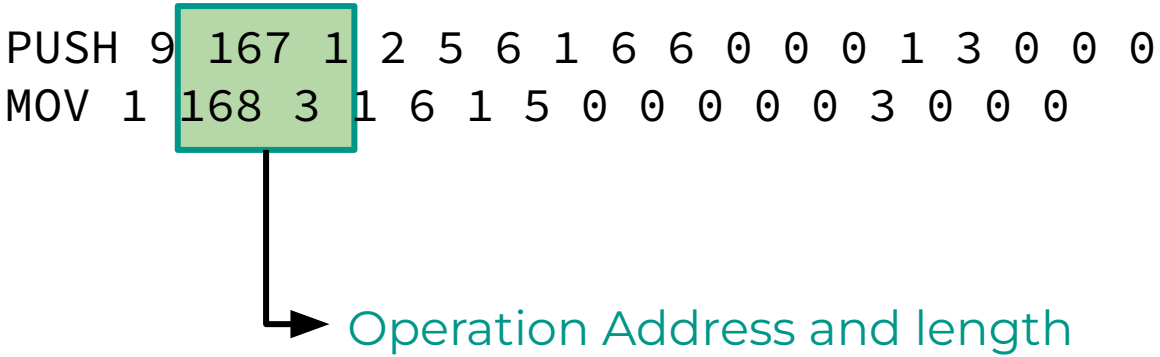
x86 Registers names

```
PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
```



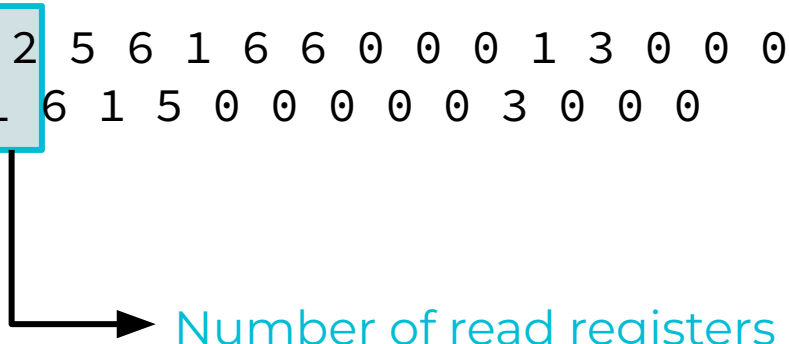
Assembly operation and opcode

x86 Registers names



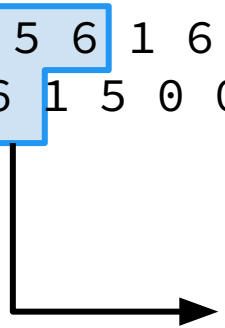
x86 Registers names

```
PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
```



x86 Registers names

```
PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
```



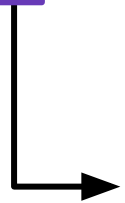
Read registers

x86 Registers names

```
PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
```

x86 Registers names

```
PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
```



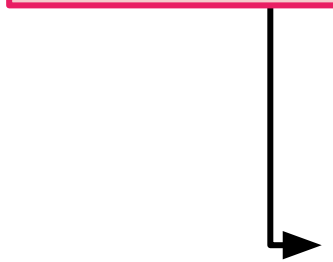
Write registers

x86 Registers names

```
PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
```

HMC Registers names

```
HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26
                  11 6 5 25 10 56 8 9 4 3 26 2
                  0 0 1 0 0 3 0 0 0
```



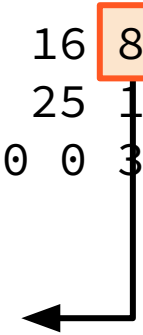
Assembly operation and opcode

HMC Registers names

```
HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26
               11 6 5 25 10 56 8 9 4 3 26 2
               0 0 1 0 0 3 0 0 0
```

HMC Registers names

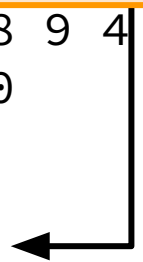
```
HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26
               11 6 5 25 10 56 8 9 4 3 26 2
               0 0 1 0 0 3 0 0 0
```



Number of read registers

HMC Registers names

```
HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26
               11 6 5 25 10 56 8 9 4 3 26 2
               0 0 1 0 0 3 0 0 0
```



Read registers

HMC Registers names

```
HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26
               11 6 5 25 10 56 8 9 4 3 26 2
               0 0 1 0 0 3 0 0 0
```

HMC Registers names

```
HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26
               11 6 5 25 10 56 8 9 4 3 26 2
               0 0 1 0 0 3 0 0 0
```

```
1  #_Z17_hmc128_saddimm_s
2  @3430
3  PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
4  MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
5  SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
6  MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
7  MOV 9 179 4 2 5 4 0 5 0 0 0 1 3 0 0 0
8  MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
9  MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
10 MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
11 XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
12 MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
13 MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
14 MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
15 MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
16 MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
17 CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
18 #_Z17_hmc128_saddimm_s
19 @3474
20 MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
21 MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
22 XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
23 JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
24 #_Z17_hmc128_saddimm_s
25 @3476
26 LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
27 RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0

28 #_hmc128_saddimm_s
29 @1
30 HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

Read registers
Write registers
Base and index registers

```
1  #_Z17_hmc128_saddimm_s
2  @3430
3  PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
4  MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
5  SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
6  MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
7  MOV 9 179 4 2 5 4 0 5 0 0 0 1 3 0 0 0
8  MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
9  MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
10 MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
11 XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
12 MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
13 MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
14 MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
15 MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
16 MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
17 CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
18 #_Z17_hmc128_saddimm_s
19 @3474
20 MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
21 MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
22 XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
23 JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
24 #_Z17_hmc128_saddimm_s
25 @3476
26 LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
27 RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0
28 #_hmc128_saddimm_s
29 @1
30 HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

Write registers

```
1  #_Z17_hmc128_saddimm_s
2  @3430
3  PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
4  MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
5  SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
6  MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
7  MOV 9 179 4 2 5 4 0 5 0 0 0 1 3 0 0 0
8  MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
9  MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
10 MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
11 XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
12 MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
13 MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
14 MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
15 MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
16 MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
17 CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
18 #_Z17_hmc128_saddimm_s
19 @3474
20 MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
21 MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
22 XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
23 JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
24 #_Z17_hmc128_saddimm_s
25 @3476
26 LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
27 RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0
```

```
28 #_hmc128_saddimm_s
29 @1
30 HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

Write registers

```
#_Z17_hmc128_saddimm_s
@3430
PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
MOV 9 179 4 2 5 4 0 5 0 0 0 1 3 0 0 0
MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
#_Z17_hmc128_saddimm_s
@3474
MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
#_Z17_hmc128_saddimm_s
@3476
LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0

#_hmc128_saddimm_s
@1
HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

Read registers
Base and index registers

```
1  #_Z17_hmc128_saddimm_s
2  @3430
3  PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
4  MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
5  SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
6  MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
7  MOV 9 179 4 1 5 4 0 5 0 0 0 1 3 0 0 0
8  MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
9  MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
10 MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
11 XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
12 MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
13 MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
14 MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
15 MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
16 MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
17 CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
18 #_Z17_hmc128_saddimm_s
19 @3474
20 MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
21 MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
22 XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
23 JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
24 #_Z17_hmc128_saddimm_s
25 @3476
26 LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
27 RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0
28 #_hmc128_saddimm_s
29 @1
30 HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

Read registers
Base and index registers

```
#_Z17_hmc128_saddimm_s
@3430
PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
MOV 9 179 4 2 5 4 0 5 0 0 0 1 3 0 0 0
MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
#_Z17_hmc128_saddimm_s
@3474
MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
#_Z17_hmc128_saddimm_s
@3476
LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0
#_hmc128_saddimm_s
@1
HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

Read registers
Base and index registers

```
#_Z17_hmc128_saddimm_s
@3430
PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
MOV 9 179 4 2 5 4 0 5 0 0 0 1 3 0 0 0
MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
#_Z17_hmc128_saddimm_s
@3474
MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
#_Z17_hmc128_saddimm_s
@3476
LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0
#_hmc128_saddimm_s
@1
HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

Read registers
Write registers

```
1  #_Z17_hmc128_saddimm_s
2  @3430
3  PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
4  MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
5  SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
6  MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
7  MOV 9 179 4 2 5 4 0 5 0 0 0 1 3 0 0 0
8  MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
9  MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
10 MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
11 XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
12 MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
13 MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
14 MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
15 MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
16 MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
17 CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
18 #_Z17_hmc128_saddimm_s
19 @3474
20 MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
21 MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
22 XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
23 JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
24 #_Z17_hmc128_saddimm_s
25 @3476
26 LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
27 RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0
28 #_hmc128_saddimm_s
29 @1
30 HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

Read registers
Write registers

```
1  #_Z17_hmc128_saddimm_s
2  @3430
3  PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
4  MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
5  SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
6  MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
7  MOV 9 179 4 1 5 4 0 5 0 0 0 1 3 0 0 0
8  MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
9  MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
10 MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
11 XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
12 MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
13 MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
14 MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
15 MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
16 MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
17 CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
18 #_Z17_hmc128_saddimm_s
19 @3474
20 MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
21 MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
22 XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
23 JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 4 0 0 0
24 #_Z17_hmc128_saddimm_s
25 @3476
26 LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
27 RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0

28 #_hmc128_saddimm_s
29 @1
30 HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

Write registers
Base and index registers

```
#_Z17_hmc128_saddimm_s
@3430
PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
MOV 9 179 4 2 5 4 0 5 0 0 0 1 3 0 0 0
MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
#_Z17_hmc128_saddimm_s
@3474
MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 0 4 0 0 0
#_Z17_hmc128_saddimm_s
@3476
LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0
```

```
#_hmc128_saddimm_s
@1
HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

Write registers

```
1  #_Z17_hmc128_saddimm_s
2  @3430
3  PUSH 9 167 1 2 5 6 1 6 6 0 0 0 1 3 0 0 0
4  MOV 1 168 3 1 6 1 5 0 0 0 0 0 3 0 0 0
5  SUB 1 171 4 1 6 2 6 25 0 0 0 0 0 3 0 0 0
6  MOV 9 175 4 2 5 3 0 5 0 0 0 1 3 0 0 0
7  MOV 9 179 4 2 5 4 0 5 0 0 0 1 3 0 0 0
8  MOV 9 183 4 2 5 8 0 5 0 0 0 1 3 0 0 0
9  MOV 8 187 9 1 23 1 10 0 0 1 0 0 3 0 0 0
10 MOV 9 196 4 2 5 10 0 5 0 0 0 1 3 0 0 0
11 XOR 1 200 2 2 56 56 2 56 25 0 0 0 0 0 3 0 0 0
12 MOV 8 202 4 1 5 1 10 5 0 1 0 0 3 0 0 0
13 MOV 8 206 4 1 5 1 8 5 0 1 0 0 3 0 0 0
14 MOV 8 210 4 1 5 1 9 5 0 1 0 0 3 0 0 0
15 MOV 1 214 3 1 9 1 4 0 0 0 0 0 3 0 0 0
16 MOV 1 217 3 1 10 1 3 0 0 0 0 0 3 0 0 0
17 CALL 7 220 5 2 26 6 2 26 6 6 0 0 0 1 1 0 0 0
18 #_Z17_hmc128_saddimm_s
19 @3474
20 MOV 8 225 4 1 5 1 10 5 0 1 0 0 3 0 0 0
21 MOV 8 229 4 1 5 1 9 5 0 1 0 0 3 0 0 0
22 XOR 1 233 9 2 9 23 2 9 25 0 0 1 0 0 3 0 0 0
23 JZ 7 242 2 2 26 25 1 26 0 0 0 0 0 4 0 0 0
24 #_Z17_hmc128_saddimm_s
25 @3476
26 LEAVE 8 249 1 3 5 5 6 2 5 6 5 0 1 0 0 3 0 0 0
27 RET 7 250 1 1 6 2 26 6 6 0 1 0 0 2 1 0 0

28 #_hmc128_saddimm_s
29 @1
30 HMC_ADD_S128OP 12 167 16 8 5 6 3 4 8 23 56 26 11 6 5 25 10 56 8 9 4 3 26 2 0 0 1 0 0 3 0 0 0
```

Read registers
Write registers
Base and index registers

Replacing functions by instructions

- New basic blocks in static trace for each translated HMC function;

Replacing functions by instructions

- New basic blocks in static trace for each translated HMC function;
- Dynamic trace references inserted basic blocks from HMC instructions;

Replacing functions by instructions

- New basic blocks in static trace for each translated HMC function;
- Dynamic trace references inserted basic blocks from HMC instructions;
- Memory trace keeps original parameter memory addresses.