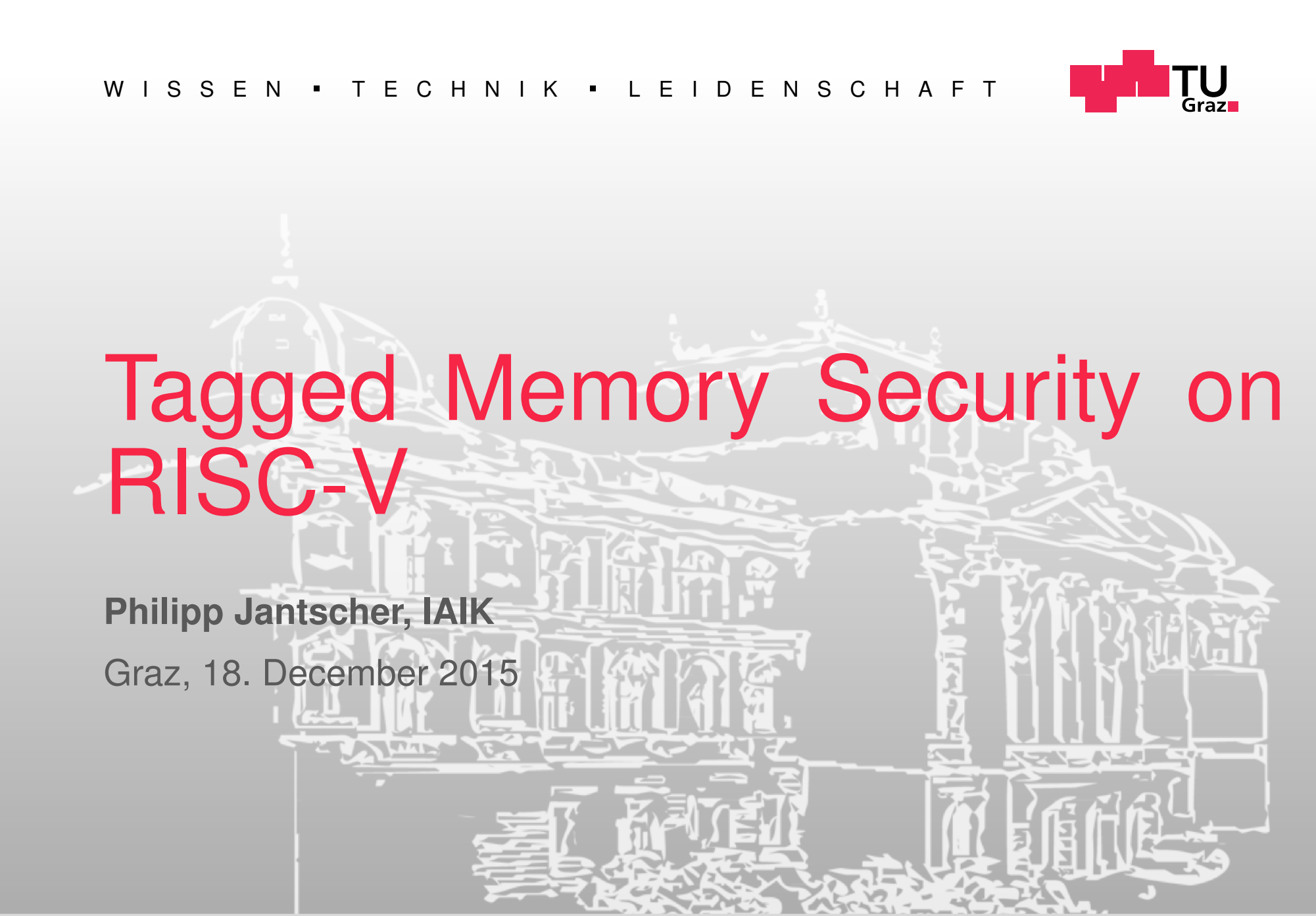


Tagged Memory Security on RISC-V

Philipp Jantscher, IAIK

Graz, 18. December 2015

Outline

- Motivation
- Vulnerable buffer overflow attacks
- RISC-V
- Hardware and Toolchain
- Tagged Memory
- Untethered RISC-V
- Tag Security Policies
- Project Time-line

Motivation

Current Status

- Stack buffer overflow still a huge impact on security
- Getting access to the computer by injection of code
- No solution yet which protects this asset

Main Goals

- Implement strong Security policies using Tagged Memory
- Efficient and Secure Solution
- Effortless protection of existing programs
- Development on a open source processor

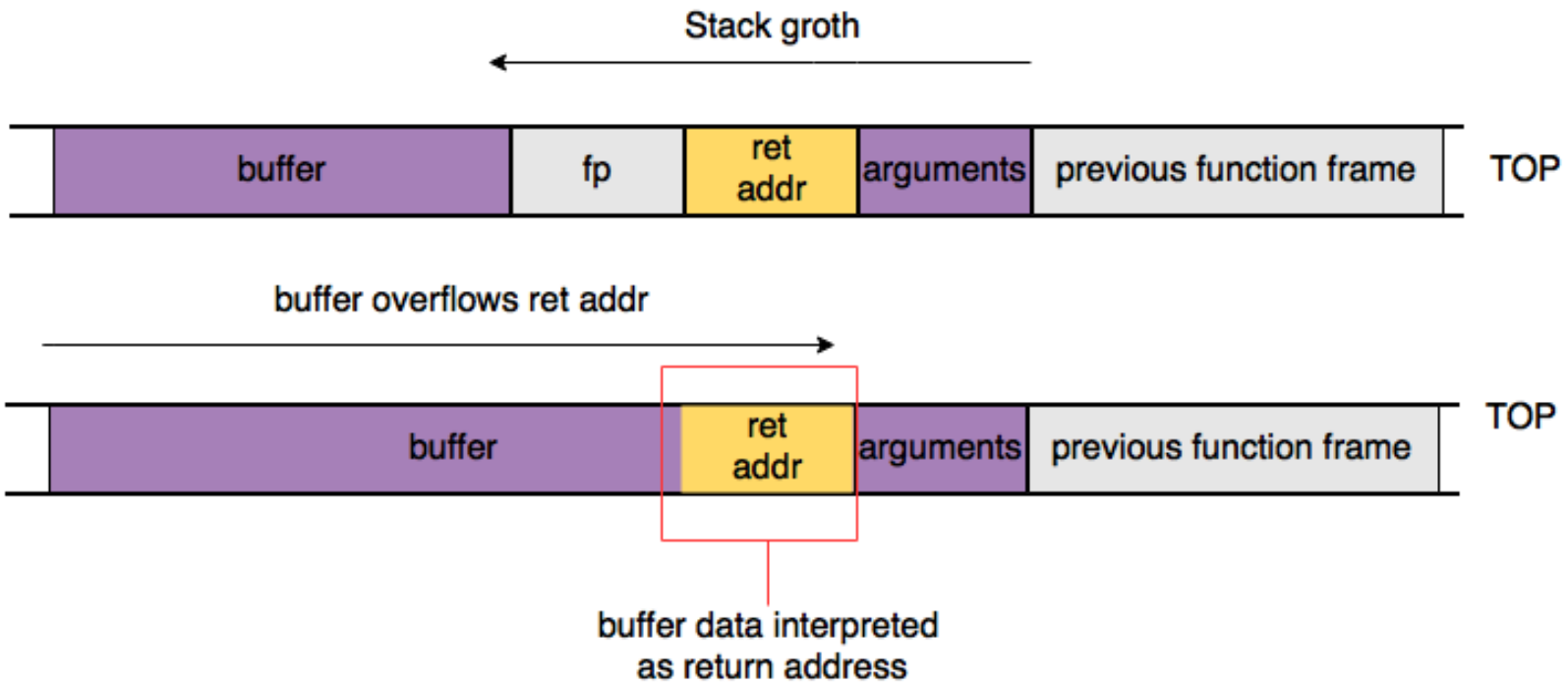
Buffer Overflows

Stack and Heap often target of attacks due to programming errors (range checks)

- Leaks easy to fix, also easy to oversee
- Attack can take control of machine
- Change of control flow
- Execution of arbitrary attacker code

Modify Return Address

Unchecked length of input string - overwrites return address

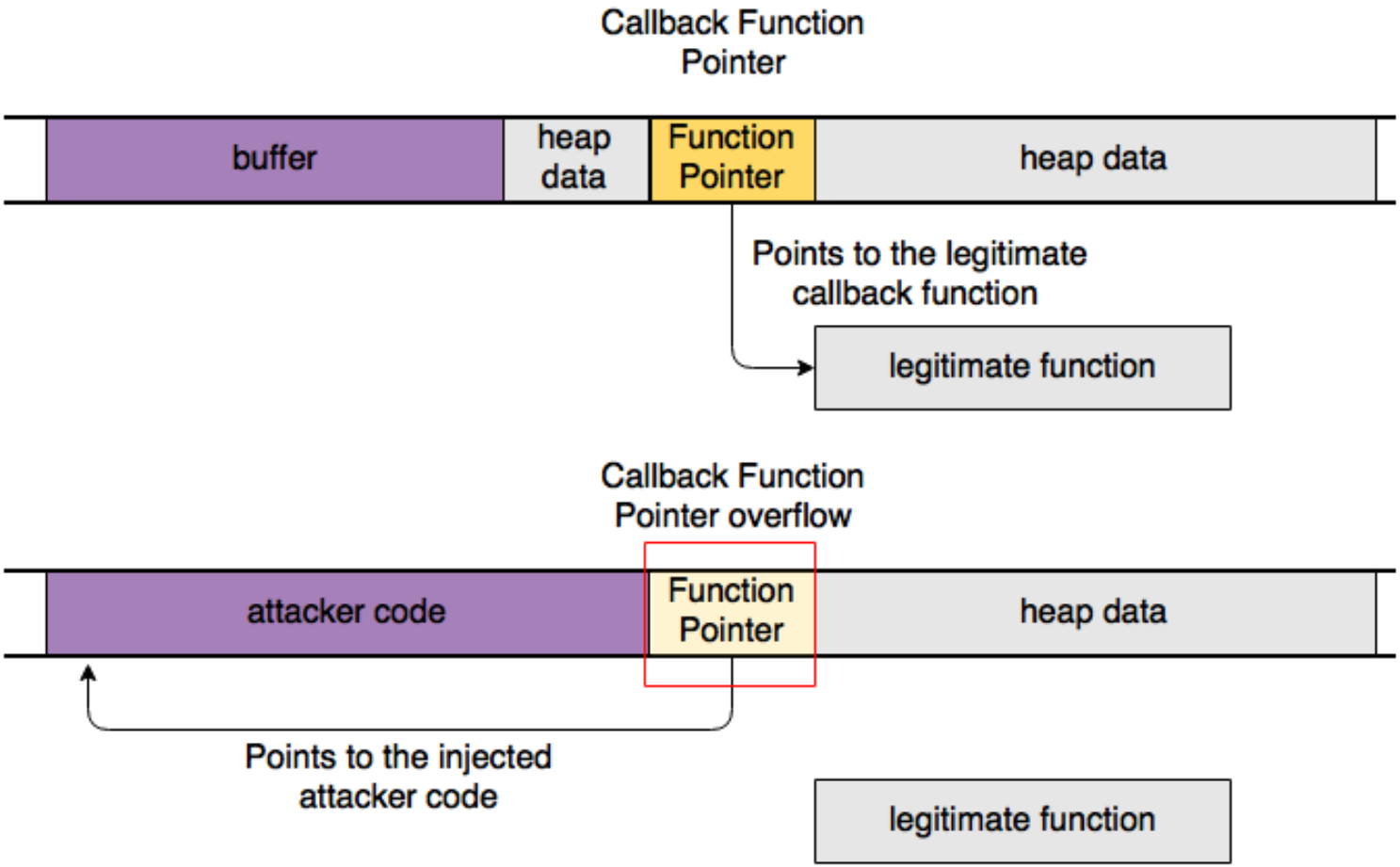


Modify Pointer To Callback Function

Similar attack but operates also on heap ...

- callback function pointer somewhere in heap
- inject attack code by heap buffer overflow
- change the function pointer to point to attack code

Modify Pointer To Callback Function

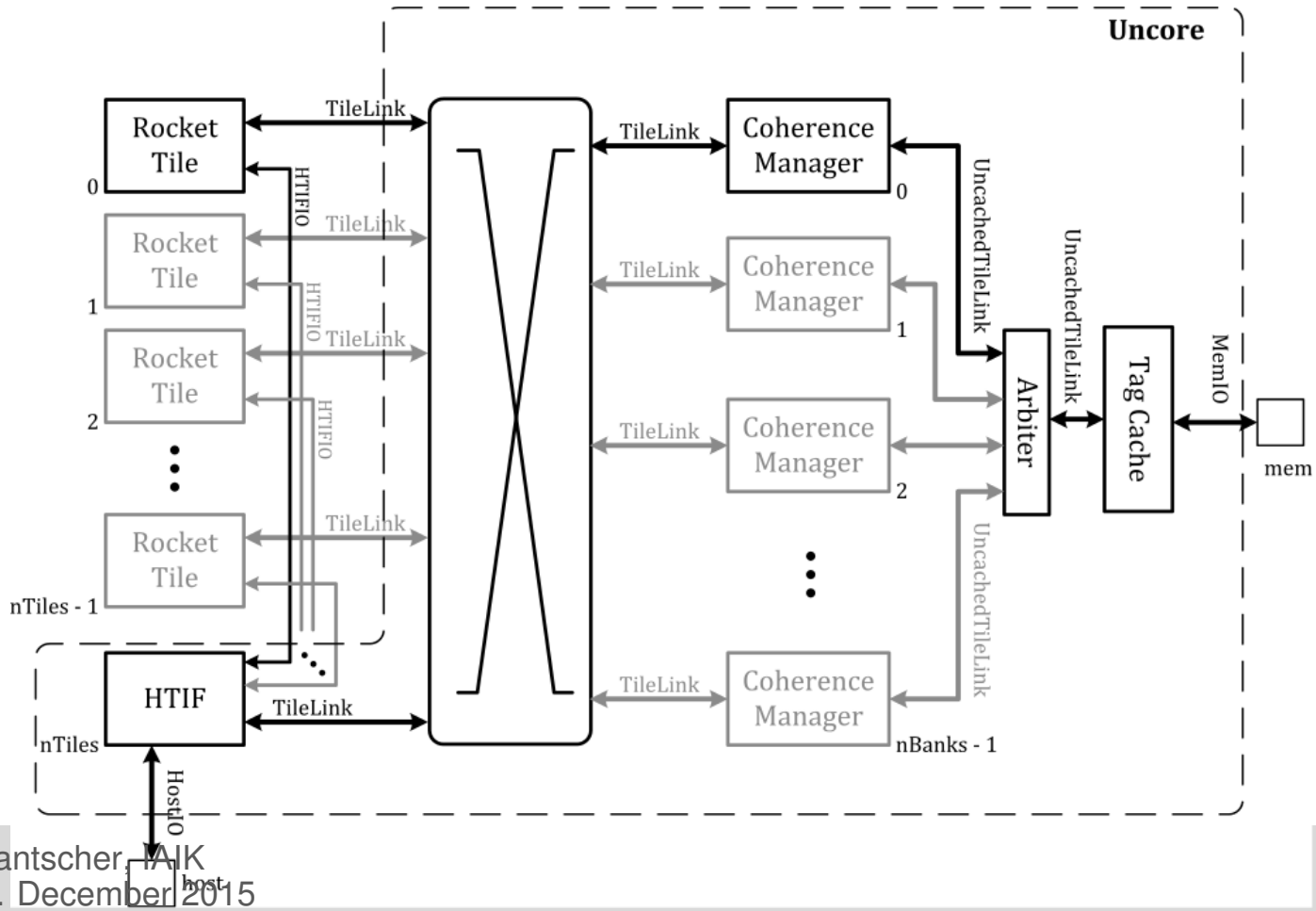


RISC-V

- Open-Source processor, University of Berkley
- Optimized Instruction set for research topics
- Developed with Chisel
- PROCESSOR DATA
- Currently tethered to ARM-Processor (boot-strapping)

RISC-V Processor Structure

ToDo: Other picture



The lowRISC project

Sub organisation

- Goal to make a fully open SoC
- Raspberry Pi for grown ups
- Community project to support hardware and application designers

Hardware

Original designed for Zynq boards. This thesis will be done on Kintex-7

- Kintex 7 information

Chisel

Developed by University of Berkley

- New high level hardware design language
- Based on Scala
- Full control over resulting RTL structures
- Powerful generator (e.g add and remove modules per define)
- Outputs: Cycle accurate C++ Emulator, Verilog
- Use Vectors, Maps, Objects within Scala

Toolchain

Several software tools for development

- Fully adapted GNU Toolchain
- RISC-V bare-metal compiler
- RISC-V Linux compiler
- Simulation with C++ Emulator (Waveform output included)
- FPGA-implemenation using Verilog output and Xilinx Vivado

Tagged Memory

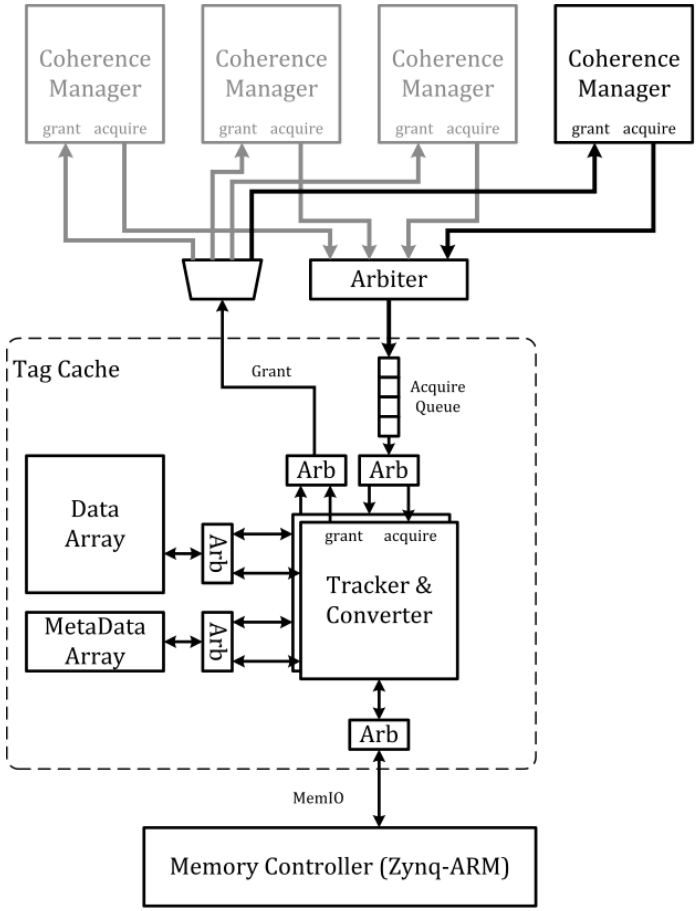
Extension to RISC-V by lowRISC team.

- Ability to tag every 64 bits in memory with one or more tags
- So far only Framework (load and store tag)
- Additional Tag-Cache to improve performance

Changes to RISC-V

- Add L1 data cache support for tagged memory type
- TileLink size 512 to 544 bits to transfer data + tags
- Adding of Tag-Cache

Tag Cache

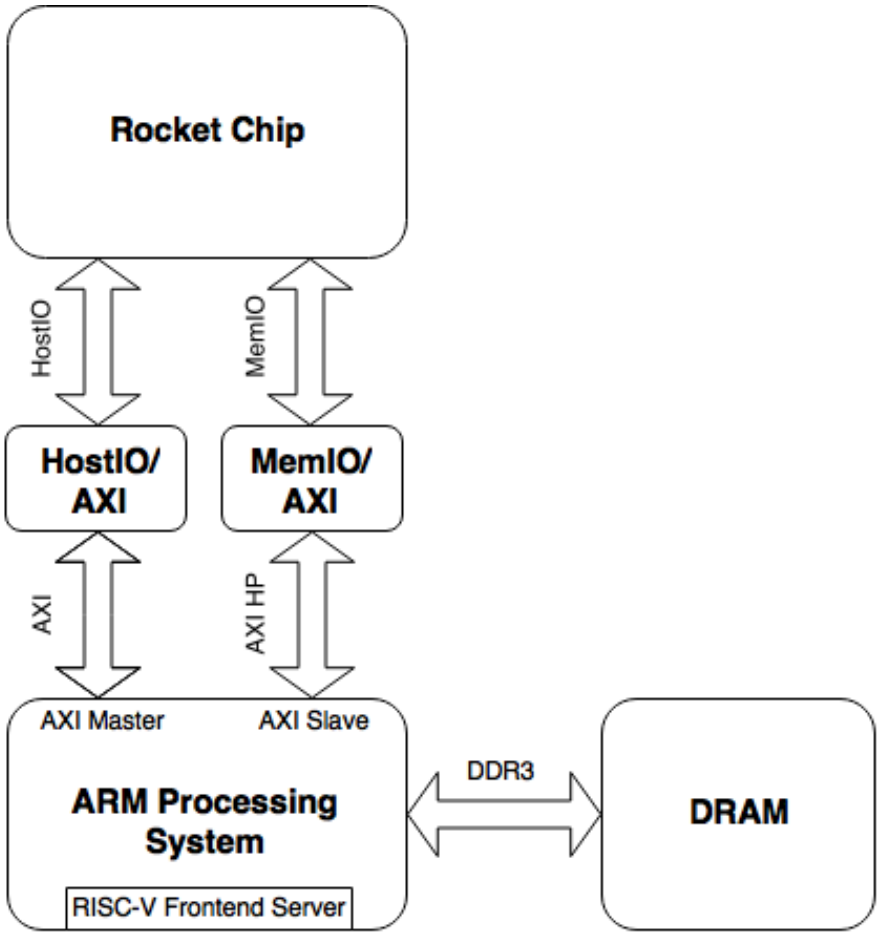

Coherence Manager
grant acquire
Coherence Manager
grant acquire
Coherence Manager
grant acquire
Coherence Manager
grant acquire
Arbiter
Tag Cache
Grant
Acquire Queue
Arb
Arb
Data Array
Arb
grant acquire
Tracker & Converter
MetaData Array
Arb
Arb
MemIO
Memory Controller (Zynq-ARM)

Current RISC-V

Currently RISC-V is thehered to ARM-Processing System

- HostIO: AXI Lite interface for Input/Output handled by ARM
- MemIO: AXI HP for DDR3 Memory access through ARM
- Boot strapping
- RISC-V Frontend Server handles console in/out

Current Structure

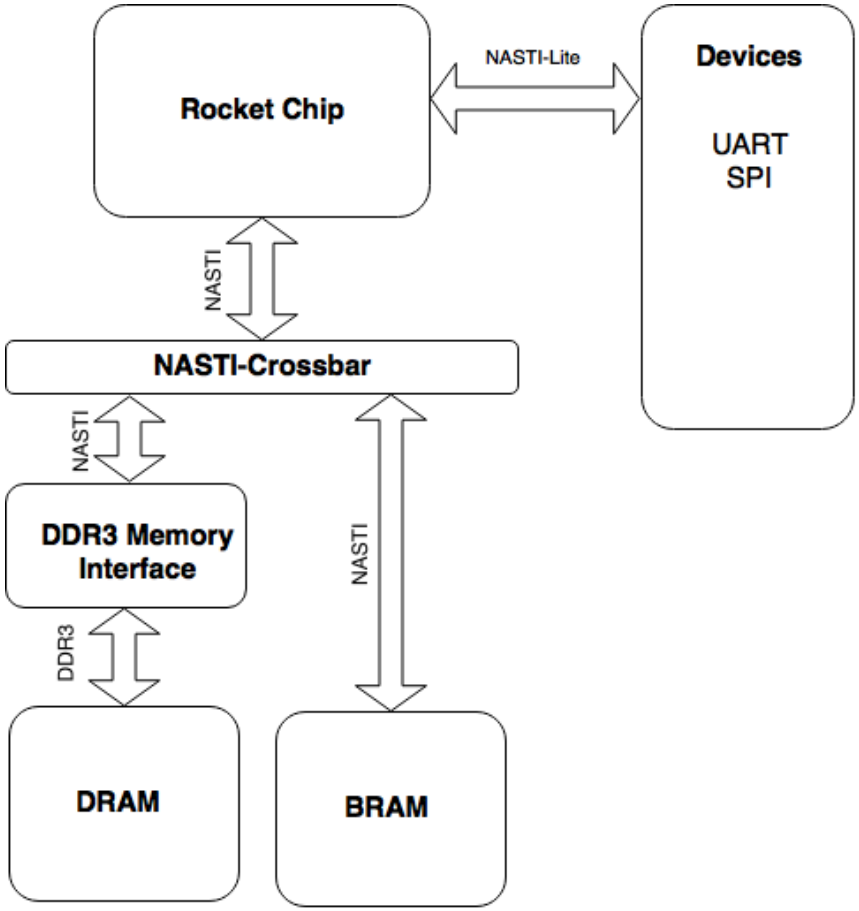


Changes for Untethering

lowRISC team implemented untethered version on Kintex-7 board (upcoming release)

- MemIO exchanged with NASTI-Bus (Berkley AXI-4 implementation)
- HostIO for debug purposes
- Adding IO devices like UART and SPI
- IO interface is NASTI-Lite
- Implementation of IO memory map
- internal and external Bus width reduced to 128 bit (4 bursts for one cache line)
- Tagged Memory not ported yet.

Untethered Structure



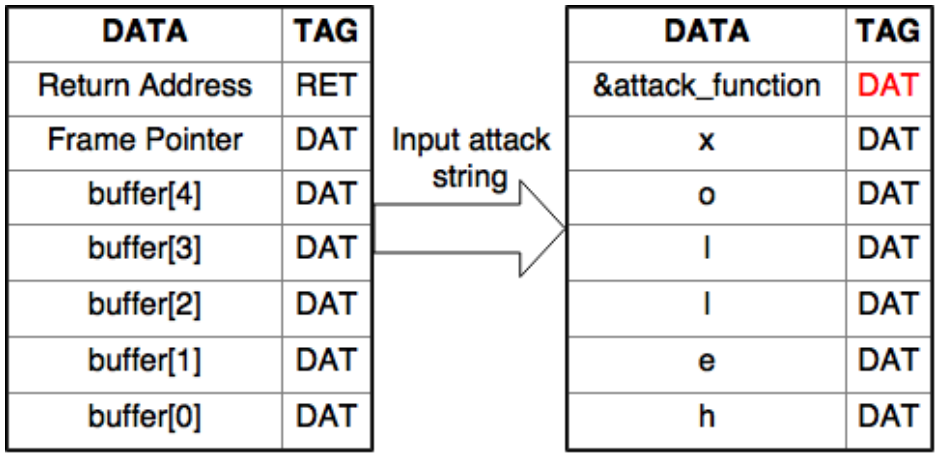
Tag Security Policies

Idea: Secure return address and function pointers from attackers

- Only minor to no change to existing Programs (Linux context switch)
- Use of 2 tag bits (Bit 1: DATA/INVALID, Bit 2: RETURN)
- Implementation in untethered version
- Tag Control Unit that checks tags and causes traps/reset
- No changes to compiler aimed for these policies

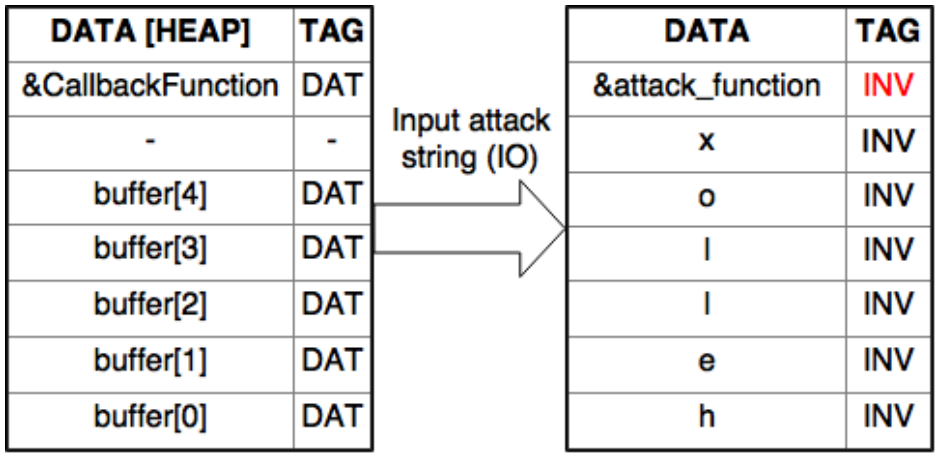
Return Address Protection

1. Function Call: JAL stores return address to stack. Tags memory as RET
2. Attacker overwrites return address. RET tag cleared.
3. JAL-R tries to jump to address in RA register without tag (trap performed).



Function Pointer Protection

1. Data and function pointers tagged as DATA
2. Attacker injects code and overwrites function pointer. (IO input causes INV tag)
3. JAL-R tries to jump to address with INV tag(trap performed).



Project Time-line

- Preparation and lowRISC tutorial. 1.12.2015, 180h [Done]
- Implementation of Tagged Memory. 31.01.2015, 150h
- Tag Security Policies. 31.04.2015, 200h
- Documentation. 31.05.2015, 200h