SiFive

GDB II : Debugging RISC-V Linux MMU (Software View)

December 08, 2023 • Taiwan

Wayling

Outline

- Why GDB again?
  - Scripting and Extending GDB
- RISC-V Linux MMU(Memory Management Unit) basic (software view)
  - RISC-V MMU basic
  - Linux MM(Memory Management) basic
- Demo (Power up your debugging skills)

Confidential ©2023 SiFive

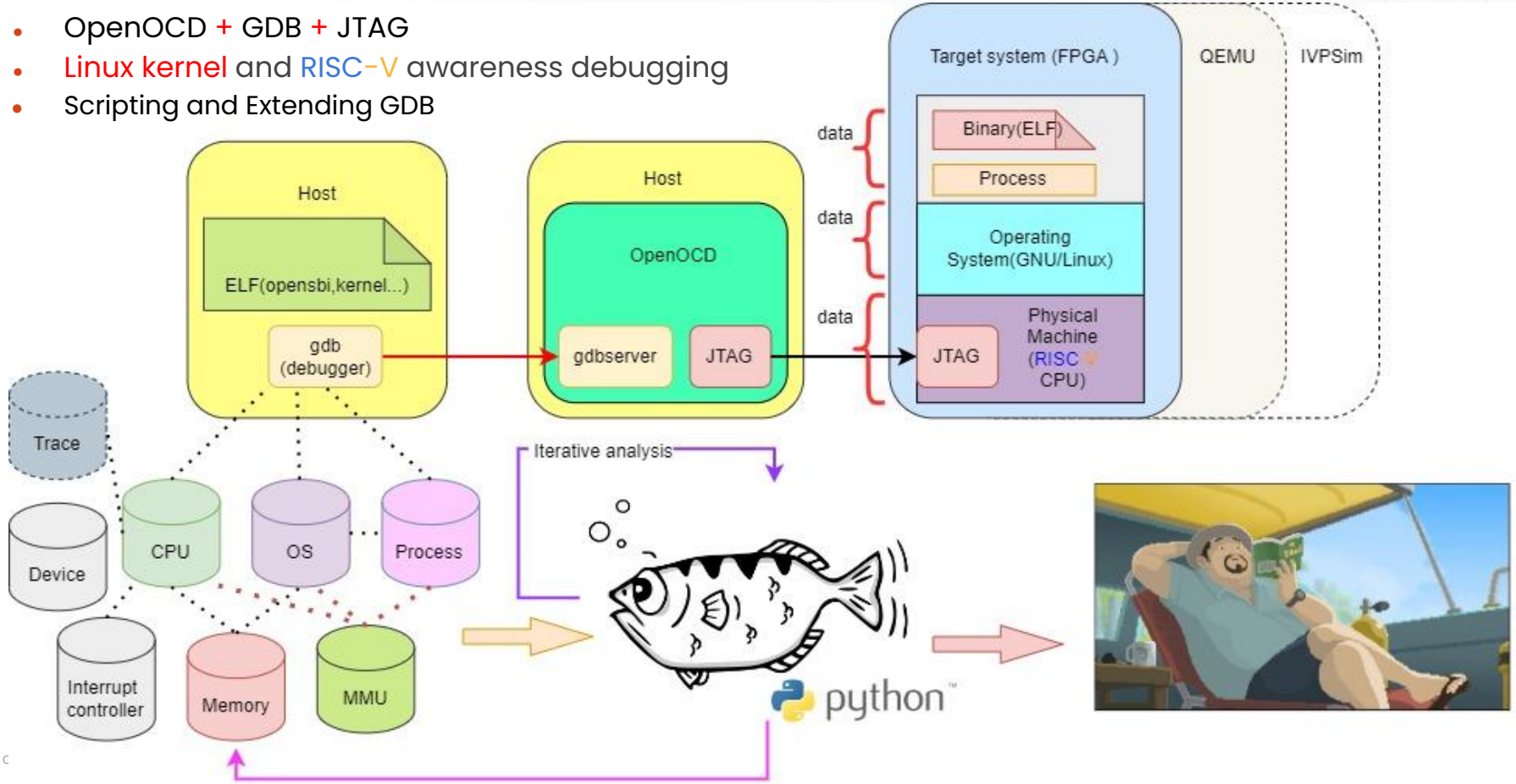
Why GDB again ? What is a debugger?

- It’s not a tool to remove bugs!
- Tools like GDB have the ability to …
  - Load ELF or Binary (opensbi + initramfs + Linux)
  - Access the program state
  - Read and write memory cells ,MMIO and CPU registers
  - In the language’s type system
  - Control the application execution
  - …
- Everything done through collaboration between
  - the OS, the compiler, the CPU … and hackers tricks!

Confidential ©2023 SiFive

Debugging tool on FPGA

- OpenOCD + GDB + JTAG
- Linux kernel and RISC-V awareness debugging
- Scripting and Extending GDB



Quick demo

- GDB script : hello and myps (list process)

Scripting and Extending GDB

- Provides the debugger with additional knowledge of the underlying operating system and system information to enable a better debugging experience. e.g. Where is the task list in memory? kernel log buffer? CPU ? Interrupt controller (AIA.m)?
- You can extend GDB using python. See link
- What is exposed via python interface has been steadily improving in GDB
  - Add custom commands
  - Implement pretty printers
  - frame filters, frame decorators and much more.
- Using on SiFive
  - How to use GDB in Freedom-Tools with Python script
  - Build our GDB + python
  - syssw public repo : Linux-gdb-python
- GDB dashboard
- Kernel built in script

Confidential ©2023 SiFive

Python API

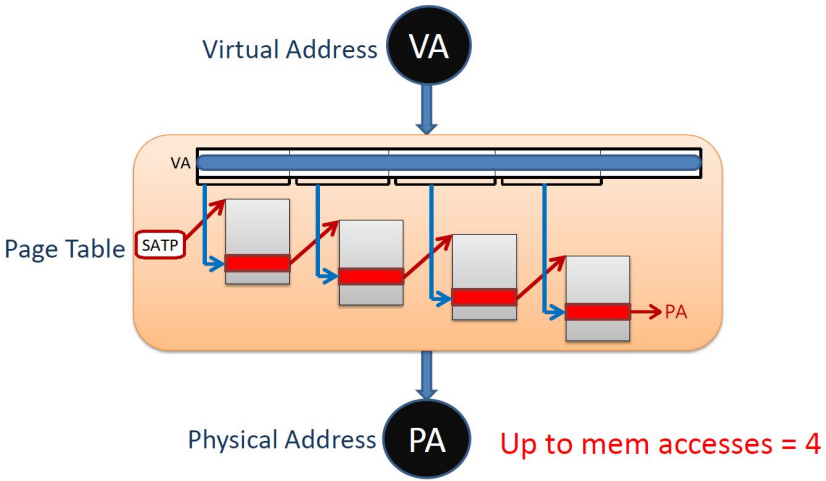
- GDB provides numerous Python interfaces. Among them, the most commonly used ones are `gdb.execute` and `gdb.parse_and_eval.`
- `gdb.parse_and_eval` takes a string as an expression and returns the result of evaluating the expression in the form of a gdb.Value.

```
Ex1: gdb.execute("run")
Ex2:
struct point p;
struct point *ptr;

point = gdb.parse_and_eval('p')
point['x'] # point.x
point['y'] # point.y
point.referenced_value() # &point

pointptr = gdb.parse_and_eval('ptr')
point2 = pointptr.dereference() # *pointptr
point2['x'] # (*pointptr ).x or pointptr ->x
```

Confidential ©2023 SiFive

RISC-V MMU basic

- The Memory Management Unit (MMU) is responsible for translating VA(virtual addresses) to PA(physical addresses) by walking through the page tables which reside in memory.
- The MMU has a hardware state machine — Page Table Walker, PTW, which issues memory requests to fetch Page Table Entries (PTEs) until a leaf PTE has been found or a page fault condition is encountered.

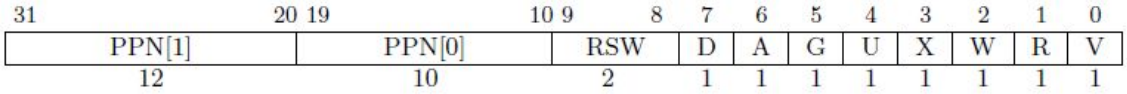


- **Sv39: 9 + 9 + 9 + 12**
- **Sv48: 9 + 9 + 9 + 9 + 12**
- **Sv57: 9 + 9 + 9 + 9 + 9 + 12**

- **leaf PTE: access attribute**
- **Non-leaf PTE: like a pointer**
- **satp: mode and pointer**

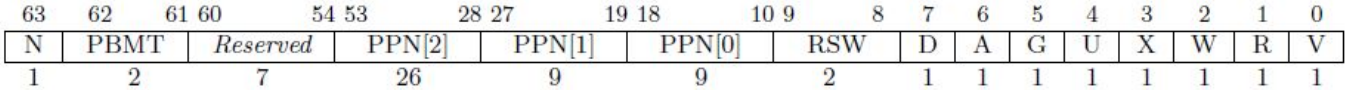
RISC-V MMU - Page Table and Page Table Entry

- Page Table: array of page table entry(PTE)
- Special encoding of PTE R/W/X
  - R/W/X = 0/0/0: non-leaf PTE
  - R/W/X = 0/1/x: reserved (writable but not readable is weird)
  - Others: legal setting
- PTE_U: 1 for U-mode, 0 for S-mode
- PTE_A: access mark
- PTE_D: dirty mark
- Magic number 22 & 44



31 – 20	19 – 10	9 – 8	7	6	5	4	3	2	1	0
PPN[1]	PPN[0]	RSW	D	A	G	U	X	W	R	V
12	10	2	1	1	1	1	1	1	1	1

Figure 4.18: Sv32 page table entry.



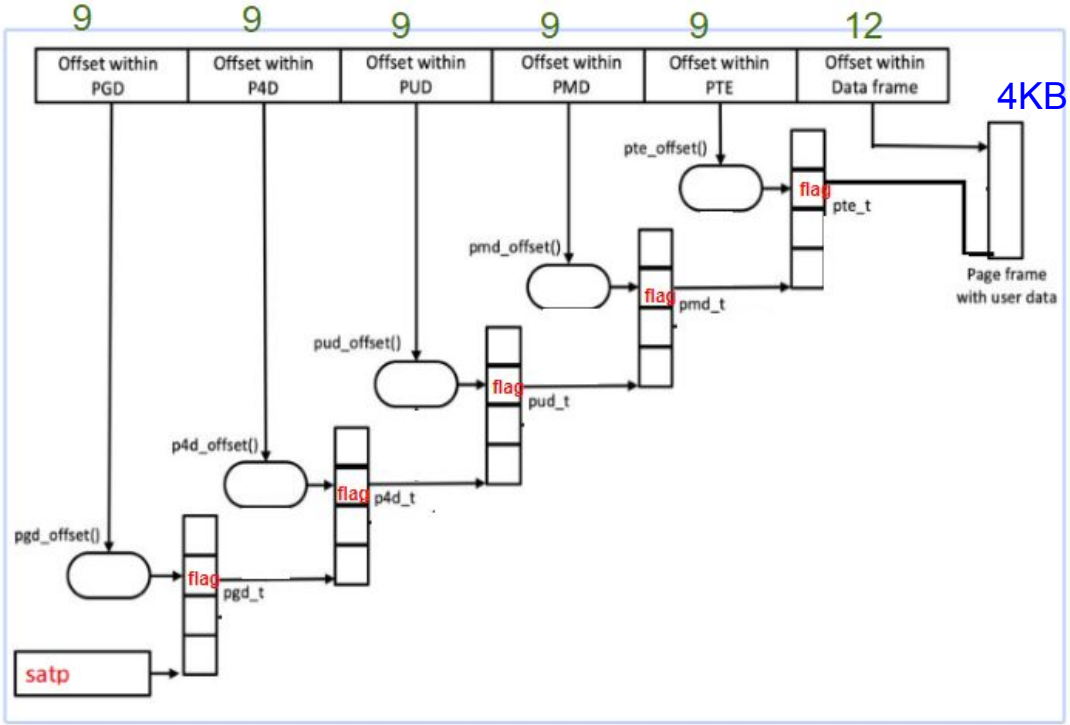
63	62 – 61	60 – 54	53 – 28	27 – 19	18 – 10	9 – 8	7	6	5	4	3	2	1	0
N	PBMT	*Reserved*	PPN[2]	PPN[1]	PPN[0]	RSW	D	A	G	U	X	W	R	V
1	2	7	26	9	9	2	1	1	1	1	1	1	1	1

Figure 4.21: Sv39 page table entry.

Confider

Linux MM basic - Linux 5 level

- A page table, of course, maps a virtual memory address to the physical address where the data is actually stored. It is conceptually a linear array indexed by the virtual address (or, at least, by the page-frame-number portion of that address) and yielding the page-frame number of the associated physical page.



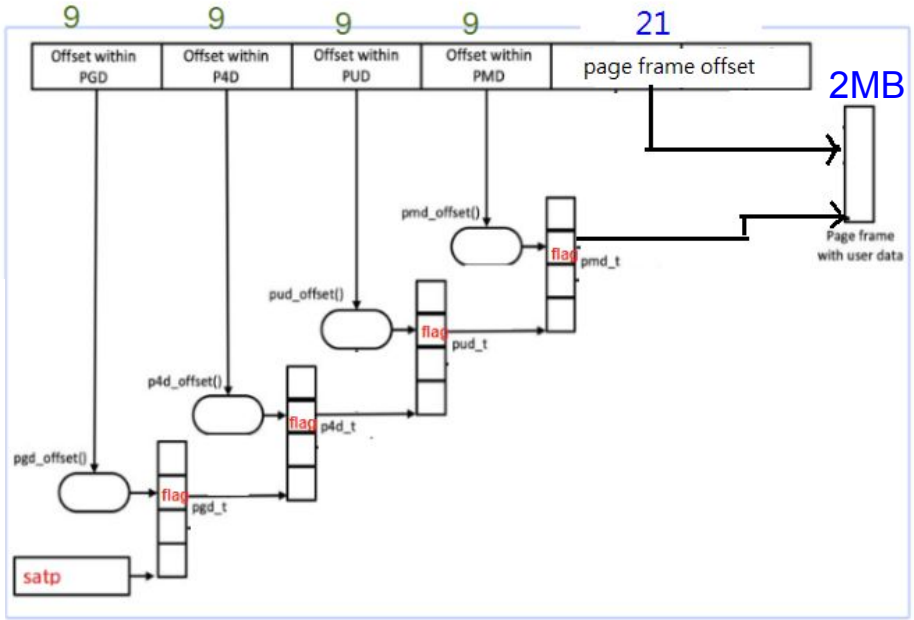
X	W	R	Meaning
0	0	0	Pointer to next level of page table.
0	0	1	Read-only page.
0	1	0	*Reserved for future use.*
0	1	1	Read-write page.
1	0	0	Execute-only page.
1	0	1	Read-execute page.
1	1	0	*Reserved for future use.*
1	1	1	Read-write-execute page.

Linux MM – Hugepage

- Huge pages are helpful in virtual memory management in the Linux system. As the name suggests, they help is managing huge size pages in memory in addition to standard 4KB page size. You can define as huge as 1GB page size using huge pages.

RISC-V - SV57

- 9 - 9 - 9 - 9 - 9(PTE) - 12 => 4KB page
- 9 - 9 - 9 - 9(PTE) - 21 => 2MB page
- 9 - 9 - 9(PTE) - 30 => 1GB page



Huge pages sizes

Architecture	huge page size
arm64	4K, 2M and 1G (or 64K and 512M if one builds their own kernel with CONFIG_ARM64_64K_PAGES=y)
i386	4K and 4M (2M in PAE mode)
ia64	4K, 8K, 64K, 256K, 1M, 4M, 16M, 256M
ppc64	4K and 16M

Confidential ©2023 SiFive

Linux MM - Virtual Memory Layout(SV57)

Virtual Memory Layout on RISC-V Linux

Start addr	Offset	End addr	Size	VM area description
0000000000000000	0	00ffffffffffffff	64 PB	user-space virtual memory, different per mm
0100000000000000	+64 PB	feffffffffffffff	~16K PB	... huge, almost 64 bits wide hole of non-canonical virtual memory addresses up to the -64 PB starting offset of kernel mappings.
				Kernel-space virtual memory, shared between all processes:
ff1bfffffea00000	-57 PB	ff1bfffffeffffff	6 MB	fixmap
ff1bffffff000000	-57 PB	ff1bffffffffffff	16 MB	PCI io
ff1c000000000000	-57 PB	ff1fffffffffffff	1 PB	vmemmap
ff20000000000000	-56 PB	ff5fffffffffffff	16 PB	vmalloc/ioremap space
ff60000000000000	-40 PB	ffdeffffffffffff	32 PB	direct mapping of all physical memory
ffdf000000000000	-8 PB	fffffffeffffffff	8 PB	kasan
				Identical layout to the 39-bit one from here on:
ffffffff00000000	-4 GB	ffffffff7fffffff	2 GB	modules, BPF
ffffffff80000000	-2 GB	ffffffffffffffff	2 GB	kernel

Kernel/linux/Documentation/riscv/vm-layout.rst

Confidential ©2023 SiFive

Demo

- RISC-V Page table walker (VA->PA)
- RISC-V Dump page table (Kernel/User)

Confide

Summary

- Ways to debug Linux using GDB and script.
- Does “Linux kernel and RISC-V awareness” mean.
- Linux MM basic (page & huge page).

Confidential ©2023 SiFive

Reference

- Hypervisor_light_talk
- Page table walker
- DAVE THE DIVER
- Linux GDB
- Python API

Confidential ©2023 SiFive

Thank you

SIFIVE.COM

©2023 SiFive, Inc. All rights reserved. All trademarks referenced herein belong to their respective companies.
This presentation is intended for informational purposes only and does not form any type of warranty.

Certain information in this presentation may outline SiFive's general product direction. The presentation shall not serve to amend or affect the rights or obligations of SiFive or its licensees under any license or service agreement or documentation relating to any SiFive product. The development, release, and timing of any products, features, and functionality remains at SiFive's sole discretion.