Safety in process CPU execution state

Ben Dooks
Codethink Ltd

Dr Jens Petersohn
Elektrobit Automotive GmbH

Pre Release

About Ben

- Ben is senior engineer and long time Linux kernel contributor at Codethink.
- Codethink is an ethical, independent, and versatile software services company, expert in the use of Open Source technologies for systems software engineering.
  - More info at https://www.codethink.co.uk/

About Jens

- Jens Petersohn has been active in a variety of industries, the last 12 years in the automotive industry. He has been at Elektrobit for two years and prior to that at Continental AG for 10 years. In the past Jens has been employed at Silicon Graphics, Inc. in their Cray Supercomputer Division and has helped port Linux to the Intel IA-64 processor family.
- At Elektrobit Jens is responsible for ADAS and HAD products and has supported the development of EB corbos Linux for automotive applications for the last year.
- Elektrobit (EB) is an award-winning and visionary global supplier of embedded and connected software products and services for the automotive industry.
- A leader in automotive software with over 30 years serving the industry, EB’s software powers over one billion devices in more than 100 million vehicles and offers flexible, innovative solutions for car infrastructure software, connectivity & security, automated driving and related tools, and user experience.
- EB is a wholly owned subsidiary of Continental AG.

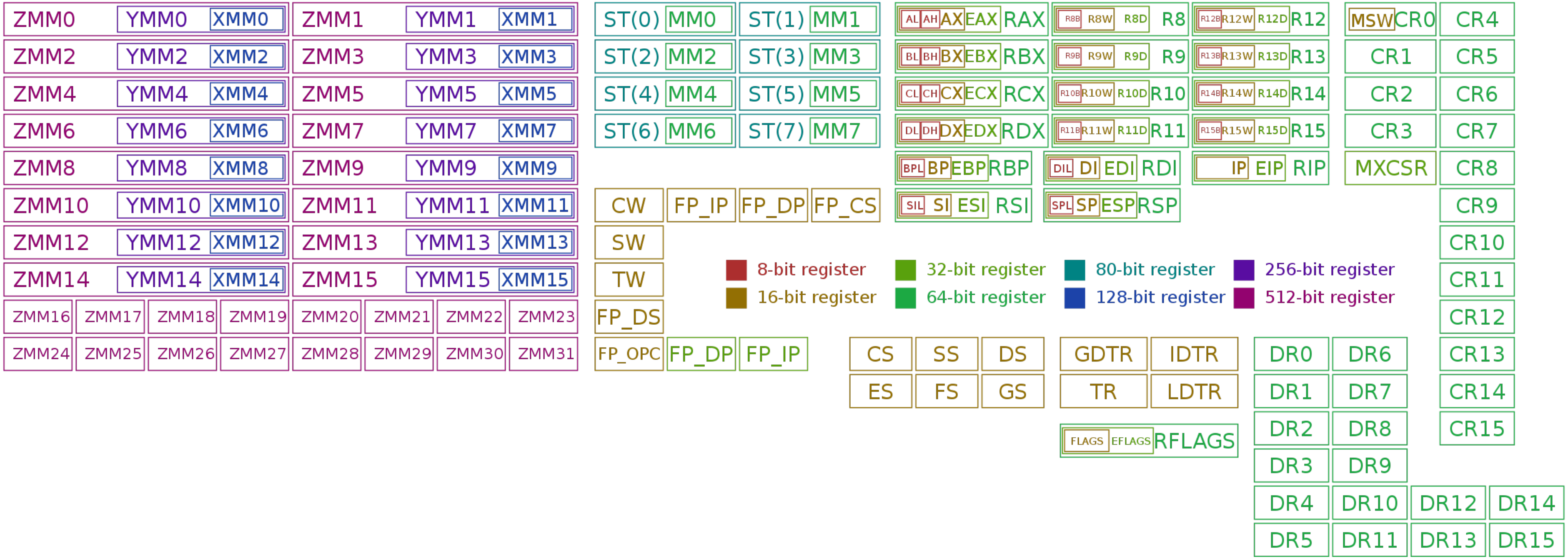
Introduction

- What are we protecting and why
- The system
  - Linux with IEC61508 SIL-2 mixed criticality
- The code flow
- Possible faults and mitigations
- A review of our mitigation

The CPU state

- Concentrating on per-core state
- Directly accessible registers
  - Integer
  - Floating point
  - Accelerators (MMX, SSE, AVX, etc)
- Indirect
  - Core control registers (debug, interrupt, etc)

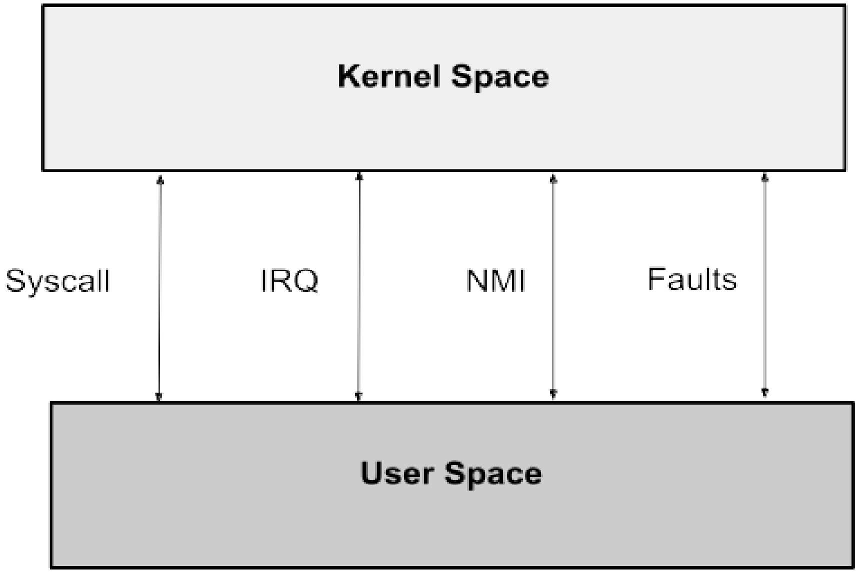
X86 64-bit core registers



https://en.wikipedia.org/wiki/X86#/media/File:Table_of_x86_Registers_svg.svg

How the code flows

- CPU executes instructions
- Intentional diversions
  - System calls
- External events
  - Interrupts
  - Exceptions (sync or async)
  - Signals (software events)
  - Architecture specific events



Faults and errors

- Not a complete list
- Mitigations
- Avoidance
- Useful Linux Kernel features

Hardware faults

Failure	Mitigations
Multiple or No entry	Verify actions post call Sequence numbers
Partial entry	KPTI SMAP, SMEP Memory permissions watchdog

Software faults

- Data corruption
  - Whole other topic
- Incorrect task switching
  - Kernel saves essential state on entry
  - Only swaps everything on re-schedule
- Bad kernel code
  - Non-integer use requires notification to kernel

Mitigation strategies

- Task isolation
  - Kernel threads still run
  - Interrupts and other events cannot be blocked
  - TL;DR – you can reduce but not stop
- Kernel checking
  - Kernel sanitisers
  - Rewrite in safe language

Codethink mitigation

- Kernel code to detect errors
  - Using shadow state
  - ~2000 lines of C
- Wraps syscall and other entry points
  - Save state on entry
  - Compare on exit
- Detection not correction

Our mitigation issues

- Significant overhead to kernel access
  - 170% slower for integer
  - 460% slower for fp/mmx/sse
  - Tested with getpid() call
- Does not cover 100% of the kernel code
  - The entry_64.s not covered
- Upstream acceptability

Testing issues

- Time
  - Kernel oops requires reboot
  - Number of test combinations
- Virtual vs Real
  - qemu issues with things like segment registers
  - And sometimes it just crashes with little explanation
- How to induce actual CPU hardware/microcode faults?

Conclusions

- Mitigations can impact performance
- Difficult/impossible covering 100% of core failures
- Testing can be time consuming
- Going forward:
  - More user-space mitigations?
  - Partial task isolation?

- Any other suggestions

Presentation copyright

- Creative Commons Attribution 4.0 International License
  - https://creativecommons.org/licenses/by/4.0/
-

Elektrobit Automotive GmbH
Am Wolfsmantel 46
D-91058 Erlangen / Germany

www.elektrobit.com

Codethink Ltd
3rd Floor Dale House
35 Dale Street
Manchester, M1 2HF, UK

www.codethink.co.uk