GoTalks - 28. 11. 2023. - Zagreb

RISC-V & Golang @ Linux & FreeBSD

sartura

History

- In 1982, David Patterson and colleagues at the University of California, Berkeley had built two working RISC microprocessors:
    - RISC-I
    - RISC-II
- By then, they had published papers showing that the performance of these designs exceeded that of some leading Complex Instruction Set Computer (CISC) designs, including the VAX 11/780 minicomputer

History, cont.

- The work at Berkeley had been supported by The Defense Advanced Research Projects Agency (DARPA) as part of their Very Large Scale Integration (VLSI) program
- This support included funding work to build design tools and support a VLSI fabrication service, MOSIS
- DARPA also funded work on RISC designs at Stanford University
- John Hennessy and his graduate students started work in early 1981 on a design known as 'MIPS', an acronym for 'Microprocessor without Interlocked Pipeline Stages'

MIPS

- MIPS shared a lot of features with Berkeley RISC, but there were two important differences:
  - **Register windows**: The MIPS design didn't make use of register windows in the way that the Berkeley RISC designs had. Thus, registers typically needed to be saved to memory when a subroutine was called
  - **Pipeline Hazards**: The design didn't have hardware to deal with pipeline hazards, for example, where pipelining means that the results of an instruction aren't yet available for a later instruction that needs that result. Instead, the compiler was expected to deal with the hazard, an assumption that simplified the design of the MIPS processors

History, cont.

- As had been the case for the Berkeley RISC project, the team at Stanford published numerous papers
- These again showed that RISC could outperform the leading CISC microprocessors of the day, such as the Motorola 68000
- The Berkeley and Stanford papers had now put the ideas behind RISC out in the open, and had exposed the performance that RISC designs could achieve
- Naturally, interest in RISC started to grow rapidly
- Just as important as performance was the fact that they had each been built by small teams of graduate students, over a timescale of a little more than a year, and with very few bugs

Historical overview of architecture and derivatives

- Soon it would seem that almost everyone with an interest in semiconductor manufacturing had their own RISC design
- Here is a (non-exhaustive) list of significant RISC (or RISC derived) architectures that emerged over the second half of the 1970s and then in the 1980s:
    - 1975 (Working design in 1978) - IBM 801
    - 1977 (Revealed publicly in 1984, first appearance in a product in 1986) - IBM ROMP
    - 1980 (Working design in 1981) - Berkeley RISC-I / RISC-II
    - 1981 (Working design in 1982) - Stanford MIPS
    - 1982 (Introduced in products in 1986) - HP PA-RISC (PA for Precision Architecture)

Historical overview of architecture and derivatives, cont.

- 1982-1984 - DEC - Titan, SAFE (Streamlined Architecture for Fast Execution), HR-32 (Hudson, RISC, 32-bit)
- 1985 - DEC - PRISM (Parallel Reduced Instruction Set Machine)
- 1984 (First designs available in 1985) - MIPS
- 1984 (First working designs in 1985) - i960
- 1984 (Releases in 1988) - AMD Am29000
- 1986 (Commercial release 1987) - SPARC (Scalable Processor Architecture)
- 1986 (First commercial release in 1990) - IBM Power
- 1986 (Commercial introduction) - Fairchild/Intergraph Clipper
- 1987 (Commercial release 1988) - Motorola 88000
- 1988 (Commercial release 1989) - Intel i860 (code-named N10 ("N-Ten") - Windows NT named after it)
- 1988 - Apollo PRISM (Parallel Reduced Instruction Set Microprocessor)

The Death of CISC?

- Perhaps just as notable as all the new RISC designs is the fact that new CISC architectures would become rare
- The late 1970s had seen Intel's 8086, the Motorola 68000, National Semiconductor 32016 and efforts from Texas Instruments and several minicomputer makers looking to shrink their designs onto VLSI
- The 1980s though were notable for their absence of new CISC designs
- Why would anyone commit to the bigger expense of building a new CISC architecture, when developing a, most likely faster, RISC architecture would be so much cheaper?

The Death of CISC?, cont.

- David Patterson himself had a hand in sealing the fate of perhaps the most complex design of the era, Intel's iAPX432 'Micromainframe' that we discussed in 'Intel iAPX432 : Gordon Moore, Risk and Intel's Super-CISC Failure'
- In May 1982, Patterson and others published a paper that showed the iAPX432 performing poorly against not only the VAX 11/780 but also against the first generation of 16-bit microprocessors, such as the 8086
- So had RISC already won the battle by the mid-1980s?
- Not entirely

The Death of CISC?, cont.

- CISC architectures that had a market foothold continued to be updated
- Intel’s 8086 was succeeded by the 80286 and then the 80386, each offering a step change in performance and capabilities, as well as, crucially, backwards software compatibility
- With Intel’s manufacturing expertise and the stranglehold that IBM compatible designs had on the business Personal Computer market, the future of the x86 architecture was assured
- Likewise, Motorola kept updating the 68000 architecture and these designs would be used in Apple’s Mac, Steve Job’s Next workstations as well as other workstations and home computers from Atari and Commodore

The Death of CISC?, cont.

- This was swimming against the RISC tide
- Many CISC designs were replaced by RISC newcomers, for example in Sun workstations where the 68000 series was replaced by Sun’s own SPARC RISC processors

The RISC Wars

- But if building a new RISC chip was a low-cost and low-risk proposition in the early 1980s, then updating and ensuring that it had appropriate support would be more expensive
- It was always unlikely that the market would be able to support so many RISC designs, none of which had any software compatibility with each other
- These simple RISC designs started to add more features and become more complex
- They needed floating point co-processors, cache memory, more complex pipelines and so on. Plus, they had to be fabricated on increasingly expensive processes
- All of this made them more expensive for firms to develop and support

Who would triumph in the RISC market?

- As the principles behind RISC were relatively simple, and the designs often had a lot in common, there was little to pick between the architectures
- It was even the case now that adding new features, in the form of complex instructions, could be seen as going against the whole idea of RISC
- So it would come down to other factors
- Electronics magazine in 1988 , under the headline 'RISC Slugfest' asked 'Is Marketing Muscle getting more important than chip performance?'
- The article went on to say that Intel and Motorola, with their established presence in the microprocessor market, would be likely to be winners in the RISC market too

Who was the winner, eventually?

- And now, the conclusion
- Intel, with the x86, was by far the most successful microprocessor designer of the era even though the x86 architecture was, for most of this period at least, demonstrably inferior to its RISC competitors
- Even thought i860 and i960 were fiascos, the ability to run IBM PC compatible software was simply the most important factor in most desktop computer purchasing decisions and the volume that the PC market gave Intel enabled it to continue to invest both in the x86 architecture and in maintaining its manufacturing lead
- **So in the end, the architecture probably mattered very little.**

What happens when you combine two of the most influential technology movements of the last few decades, RISC and Open Standards?

- **You get RISC-V**
- RISC-V (pronounced risk-five) is a relatively new Instruction Set Architecture. It's supporters make ambitious claims for the architecture
- By any standards, RISC-V is starting to make an impact
- RISC-V International also says that as a the end of 2022, over 10 billion RISC-V cores had been shipped
- RISC-V International corporate members include Intel, Google, Qualcomm, AMD, Nvidia and hundreds more
- So why is RISC-V attracting so much attention?

RISC-V, history

- The story of RISC-V takes us back to the UC, Berkeley in February 2005
- A group of researchers, including David Patterson and Krste Asanović, started meeting to discuss issues around parallelism in computing system
- This in turn led to the formation of a 'Parallel Computing Laboratory' also known as 'Par Lab' at Berkeley
- This in turn led to the formation of a research program to investigate the issues around building systems with large numbers (from 64 to over a thousand) of processor cores
- The program became known as 'Research Accelerator for Multiple Processors' or RAMP

RISC-V, history, cont.

- Over the course of the next few years, they built a series of color-coded systems: RAMP Red, RAMP Blue, RAMP White and RAMP Gold
- These systems used a range of different processor core including PowerPC 405, Xilinx MicroBlaze cores running on Field Programmable Gate Arrays, or Sun's SPARC
- At around the same time Intel and Microsoft, who seeing the increasing importance of multicore and parallel computing had run a competition for university projects to do research in the area
- The competition and $10m of funding was jointly won by Berkeley's Par Lab

RISC-V, history, cont.

- By 2008, members of the RAMP team were getting frustrated with the cores they were using, commenting that “Most pre-existing IP cores are inappropriate without significant work.”
- As RAMP was coming to an end in 2010, Asanović, together with his graduate students Yunsup Lee and Andrew Waterman who had been working together on RAMP Gold, needed a processor core for use in what was intended to be a ‘three month’ research project
- Their first thoughts were to use the Arm ISA for the new processor

RISC-V, history, cont.

- Born in the UK, Asanović had owned a 6502-based Atom home computer from Arm developer Acorn
- He’d then studied in Cambridge at the time that the Arm processor was being developed
- He’d left the UK in 1989 to study under David Patterson at Berkeley, going on to gain a PhD at Berkeley in 1998, with a thesis on the subject of in vector microprocessors

RISC-V, history, cont.

- Asanović, Waterman and Lee soon realised, though, that using Arm **would mean buying a licence**
- Even worse, they would probably not be able to modify the core as they wanted and would then struggle to distribute their designs to researchers at other universities to study and build on
- They looked at existing ISAs with fewer restrictions, such as SPARC, OpenRISC and DLX, another ISA designed by David Patterson together with John Hennessy at Stanford in the 1990s
- In each case though the ISA didn't meet their needs.

RISC-V, history, cont.

- With no obvious alternatives available, in May 2010 they started work on the development their own ISA
- By August 2010 the new processor had a name 'RISC-V': with V chosen as 'V for Five, V for Vector, V for Variants' and was already taking shape
- The designation as the 'fifth RISC' was seen as appropriate as Patterson's later 1980s RISC projects at Berkeley, SOAR and SPUR, had been retrospectively designated as RISC-III and RISC-IV respectively
- The Berkeley team asked the question 'Are we crazy not to use a standard ISA?' before noting that existing standard ISAs (x86, ARM and GPUs) would probably be too complex for a university project anyway

RISC-V, history, cont.

- Asanović would later, credit Lee and Waterman for pushing the idea, reflecting that:
    - *One of the greatest powers in the universe is grad-student naivety. When you don't realise something is impossible, you try it anyway*
- By May 2011, they were ready to publish the first version of the 'RISC-V Instruction Set Manual'

RISC-V, history, cont.

- This short document (at just 32 pages) outlined several goals for the RISC-V ISA, including to:
  - Provide a realistic but open ISA that captures important details of commercial general-purpose ISA designs and that is suitable for direct hardware implementation
  - Provide a small but complete base ISA that avoids “over-architecting” for a particular microarchitecture style
  - Be simple to subset for educational purposes and to reduce the complexity of bringing up new implementations

RISC-V, history, cont.

- The new ISA was designed to support 32 and 64-bit variants (designated RV32 and RV64), multi-core implementations and floating point operations
- Crucially, it was also designed to allow processor designers to add their own extensions to the architecture
- **The minimum implementation of RV32 described in the first manual had just 47 instructions**
- The first RISC-V Instruction Set Manual acknowledges the influence of previous research architectures, including Torrent-0, The Scale (Software-Controlled Architecture for Low Energy) project at MIT and Maven (Malleable Array of Vector-thread ENgines)

Early RISC-V Hardware

- The first core was called Raven-1, developed in May 2011, which was built using a STMicroelectronics 28nm process
- Raven-1 was followed by Raven-2 in August 2012 and Raven-3 in September 2013
- By 2014 the Berkeley team were able to demonstrate significant progress with their new ISA
- There was more to RISC-V than some tools and few cores though
- There was a philosophy!

The philosophy

- Asanović considered a long list of historic, and mostly defunct, ISAs, asking:
  - Do we need all these different ISAs?
  - Must they be proprietary?
  - Must they keep disappearing?
  - What if there was one stable free and open ISA everyone could use for everything?
- **So RISC-V would be a free and open ISA**

The RISC-V Instruction Set : A Brief Introduction

- **Integer Registers**
- A RISC-V CPU typically has 32 general purpose registers designated x_0 to x_{31} (note that the RV32E (E for embedded) has only 16 registers)
- The register x0 is hard-coded to be zero; the program counter is distinct from these 'general purpose' registers
- Conditional branch instructions, specify the comparison to be undertaken to determine whether the branch is taken, rather than relying on a previously set 'flag' that is set by a previous arithmetic or logic instruction.
- The width of these registers depends on the 'Base ISA' used, with 32 and 64 bit versions - defining both the size of integer registers and the size of the address space - currently commonly being used
- There is also a 128-bit version, which started as a joke, but has now become part of the standard

The RISC-V Instruction Set : A Brief Introduction, cont.

- **Base ISA**
- Although it is convenient to speak of the RISC-V ISA, RISC-V is actually a family of related ISAs which build on what are known as ‘base’ ISAs
- There are currently four base ISAs
- The original integer ‘bases’ (known as I) provide basic arithmetic and logic and load / store instructions. The simplest 32-bit and 64-bit RISC-V implementations are known as RV32I and RV64I, respectively

The RISC-V Instruction Set : A Brief Introduction, cont.

- **Base Integer Instruction Set**
- The instructions in RV32I can be grouped as follows (with the assembler mnemonics for the instructions in each group):
  - 8 that load or store bytes, half-words or words to or from memory. [LB, LH, LW, LBU, LHU, SB, SH, SW]
  - 6 that perform program branches, including conditional branches, depending on the results of a specified comparison. [BEQ, BNE, BLT, BGE, BLTU, BGE]
  - 6 that shift registers by a specified number of bits. [SLL, SLLI, SRL, SRLI, SRA. SRAI]

The RISC-V Instruction Set : A Brief Introduction, cont.

- 3 arithmetic instructions. [ADD, ADDI, SUB]
- 6 logical instructions. [XOR, XORI, OR, ORI, AND, ANDI]
- 4 that load either 0 or 1 into a register, depending on the result of a comparison. [SLT, SLTI, SLTU, SLTUI]
- 2 jump and link instructions that store 'program counter+4' in a register and then jump to another address. [JAL, JALR]
- 2 that either load an immediate value into the upper 12 bits of a register or add an immediate value to the upper 12 bits of the program counter. [LUI, AUIPC]
- 2 that either transfer control either to the operating system or to a debugger. [SCALL, SBREAK]
- 1 relating to memory ordering. FENCE

Just 40 instructions

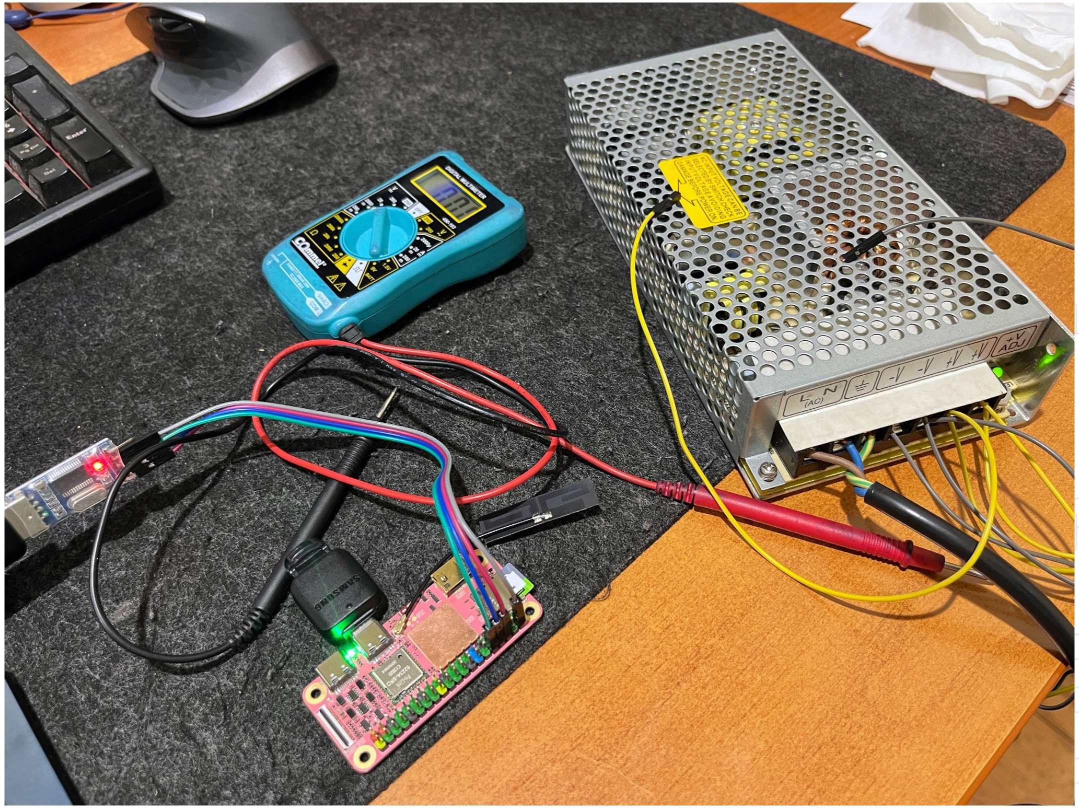
The RISC-V Instruction Set : A Brief Introduction, cont.

- **Instruction Encoding**
- The ISA is also designed to be straightforward for a processor to decode
- The base 32-bit integer ISA has just 6 instruction formats and these place things like the opcode and source and destination register in the same place in the 32-bit long instruction
- **Extensions**
- Many implementations of RISC-V ISA will add extensions that will materially increase the number of instructions - for example the vector extension adds almost two hundred new instructions - but the ISA is still significantly smaller than alternatives like x86 and Arm

Then why Go and RISC-V?

- This is why you probably came here in the first place
- At current state, RISC-V is not the most performant ISA implementation on the planet
- Still, embedded is starting getting filled
- More and more application on embedded rely on what you could traditionally call “high level user space applications”
- Ranging from IOT, network equipment and such, RISC-V is getting traction
- There are dedicated projects related to RISC-V and Go, namely *Project RP001 Accelerate the Go Runtime on RISC-V*

Our hardware showcase



MangoPi MQ-Pro

Manufacturer	MangoPi
Dimensions	65mm x 30mm (RBP Zero form factor)
Release Date	April 2022
SoC	Allwinner D1 @ 1.0Ghz (XuanTie C906 core)
DRAM	1GiB DDR3(L)
Power	DC 5V @ 2A (via OTG Type-C connector)
Video/Audio	HDMI (Type C - Mini), I2S
Network	Trolink TL8723DS Wi-Fi+BT, Ethernet via optional secondary board
Storage	µSD, optional SPI flash (bottom)
USB	1 USB Type-C OTG, 1 USB Type-C USB host

MangoPi MQ-Pro, Linux and Go

```
ubuntu@mangopi-mqpro:~$ uname -a
Linux mangopi-mqpro 6.5.0-10-generic #10.1-Ubuntu SMP Sat Oct 28 04:47:16 UTC 2023
riscv64 riscv64 riscv64 GNU/Linux
ubuntu@mangopi-mqpro:~$ cat /proc/cpuinfo
processor       : 0
hart            : 0
isa             : rv64imafdc_zicntr_zicsr_zifencei_zihpm
mmu             : sv39
uarch           : thead,c906
mvendorid       : 0x5b7
marchid         : 0x0
mimpid          : 0x0
ubuntu@mangopi-mqpro:~$ go version
go version go1.21.4 linux/riscv64
```

Hvala na pažnji!

Bruno Banelli – bruno.banelli@sartura.hr