f8

An 8-Bit architecture based on lessons learned from SDCC and the architectures it supports

Philipp Klaus Krause

July 8, 2022

Lessons learned - big picture

- An efficient stackpointer-relative addressing is essentiakl for reentrant functions
- A unified address space is essential for efficient pointer access
- Registers help
- Hardware multithreading can replace peripheral hardware, but it needs good support for atomics, and thread-local storage
- Irregular architectures can be very efficient with tree-decomposition-based register allocation
- A good mixture of 8-bit and 16-bit operations helps
- Pointers should be 16 bits

Lessons learned - details

- Zero-page, etc addressing isn't useful if we have efficient stackpointer-relative addressing
- A index-pointer-relative read instruction for both 8 and 16 bits is important
- Prefix bytes can be a good way to allow more operands (e.g. registers)
- Hardware $8 \times 8 \rightarrow 16$ multiplication helps
- Division is rare
- Multiply-and-add helps speeds up wider multiplications
- BCD support provides cheap printf without need for hardware division
- Good shift and rotate support helps

Where do we get

- Irregular CISC architecture
- The core becomes bigger than for RISC, but we save so much on code memory that it is worth it
- Approx. 30% lower code size than stm8 - in current very early experimental SDCC backend