A free toolchain for 0.01 € - computers

The free toolchain for the Padauk 8-bit microcontrollers

Philipp Klaus Krause

February 2, 2020

Table of Contents

Table of Contents

Padauk µC

- Taiwanese manufacturer (rebrand reseller: Puolop)
- Cheap (down to 0,01 €)
- Low power
- Accumulator-based architecture
- Relatively nice architecture (similar to MCS-51)
- 60 B to 256 B RAM
- 0.5 KW to 4 KW program memory (PROM or Flash)
- Few peripherals (only timer, comparator, ADC, PWM, watchdog)
- 1 to 8 “Cores” (hardware threads)

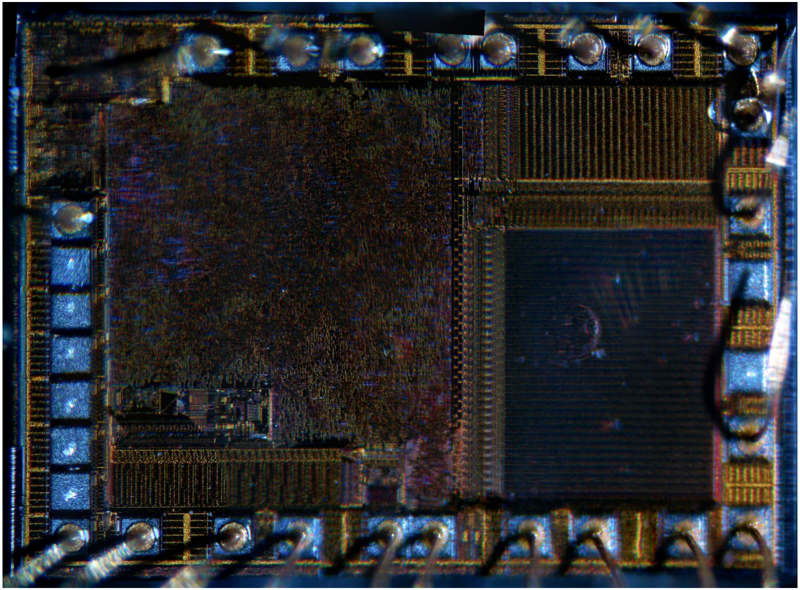
Subarchitectures

subarchitecture	pdk13	pdk14	pdk15	pdk16
internal name	SYM_84B	SYM_85A	SYM_86B	SYM_83A
prog. mem. width	13	14	15	16
prog. addr. bits	10	11	12	13
data addr. bits	6	7	8	9
I/O addr. bits	5	6	7	6
hardware threads	1	1 or 2	1	2, 4 or 8

Hardware Threads

- Barrel processor
- Per-thread state: accumulator, stack pointer, program counter, flag register
- “Core”, “processing unit”, “FPP”, “FPPA”
- Lack of instruction support

PMC234 Die



Non-free Tools

- Mini-C: IDE integrated with compiler/assembler/software for writer/emulator.
- Assembly with a bit of C-like syntactic sugar.
- Program writer
- In-circuit emulator

Free Tools

- Small Device C compiler (SDCC) with assembler, linker, simulator
- Easy PDK programmer with firm- and software
- development boards

Table of Contents

Easy PDK programmer

- Simple hardware usable with various OSes
- Fully cupports 6 Padauk-µC so far (PMS15A, PMS150C, PMS154B, PMS154C, PFS154, PFS173)
- For 12 few more, there already is read-only support
- https://github.com/free-pdk

Development Boards

- Minimal boards
- 4 LED
- Power supply via programmer, pins oder USB
- https://github.com/free-pdk/f-eval-boards

Table of Contents

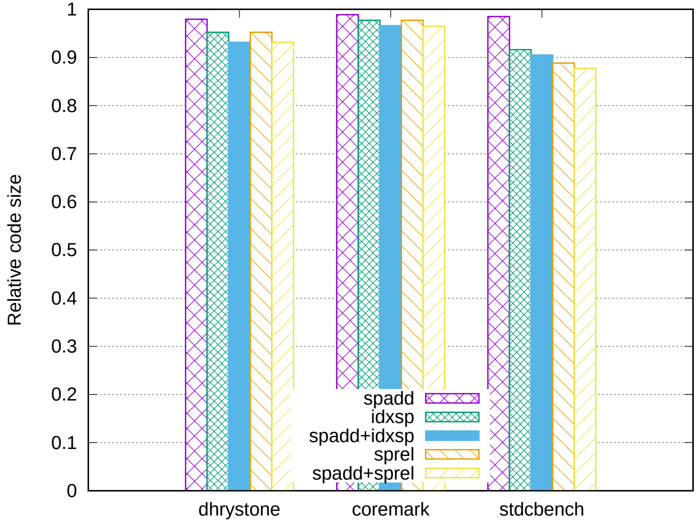
What is SDCC?

- C compiler (ANSI C89, ISO C99, ISO C11, ISO C2X)
- Freestanding implementation or part of a hosted implementation
- Supporting tools (assembler, linker, simulator, ...)
- Works on many host systems (GNU/Linux, Windows, macOS, Hurd, OpenBSD, FreeBSD, ...)
- Targets various 8-bit architectures (MCS-51, DS80C390, Z80, Z180, eZ80 in Z80 mode, Rabbit 2000, Rabbit 3000A, LR35902, TLCS-90, HC08, S08, STM8, pdk14, pdk15, pdk13, PIC14, PIC16)
- Has some unusual optimizations that make sense for these targets (in particular in register allocation)
- http://sdcc.sourceforge.net

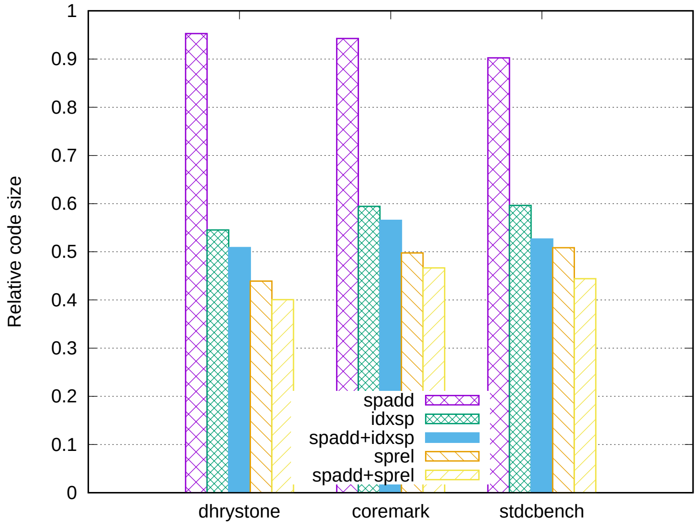
SDCC for Padauk

- Supports pdk13, pdk14, pdk15
- Functions are non-reentrant by default (local variables at fixed locations instead of on stack)
- Via `__reentrant` individual functions can be made reentrant, via `--stack-auto` whole translation units can be compiled as reentrant; this comes at a significant code size and runtime cost (e.g. 16-bit addition: 34 inst / 40 cycles vs. 6 inst / 6 cycles)
- Access to I/O-registers via `__sfr` and `__sfr16`

Code size benefits of pdk15 improvements



Code size benefits of pdk15 improvements - all reentrant



Optimal Register Allocation in Polynomial Time

- Register allocator based on graph-structure theory
- Optimal register allocation in polynomial time
- Flexible through use of cost function
- Provides substantial improvements in code quality
- Slow compilation for targets with many registers
- Compilation speed / code quality trade-off:
  –max-allocs-per-node

Regression testing

- Regression testing of nightly snapshots
- ≈ 12000 tests compiled and executed on simulators
- Tests mostly from fixed bugs and from GCC
- Targets architectures: MCS-51, DS390, Z80, Z180, eZ80 in Z80 mode, Rabbit 2000, Rabbit 3000A, LR35902, TLCS-90, HC08, S08, STM8, pdk14, pdk15
- Host OS: GNU/Linux, macOS, “Windows” (cross-compiled on GNU/Linux, tested via wine)
- Host architectures: x86, amd64, ppc, arm

LLVM+SDCC

- Uses LLVM C front- and backend to produce C code to be compiled with SDCC
- Code compiled with LLVM+SDCC can be mixed with C code compiled with SDCC
- Allows languages other than C
- Enables high-level optimizations
- Experimental, many issues remaining

Table of Contents

TODO

- SDCC needs developers
- Fix SDCC bugs
- Improve SDCC further in standard compliance, optimizations, debug info, etc
- Make LLVM+SDCC useable
- Improve IDE integration
- Add support for more devices in Easy PDK programmer