Subject: Rapsberry PI - cpu dilemma
Posted by mirek on Fri, 08 May 2020 14:36:37 GMT
View Forum Message <> Reply to Message

There is an interesting issue. Original PI had Armv6 CPU, since PI 2, it is Armv7.

However, all binaries are still compiled for PI 1.

There two major differences between those 2 CPUs:

- Armv7 allows unaligned memory access, which is quite useful to optimize some things (e.g. Stream::Get32 method)
- Armv7 supports thumb 2, which results in significantly reduced code size

It is possible to activate compilation for Armv7 with clang (and GCC) using compiler flags and the code is really quite better, on platform where every bit of performance matters.

So it makes me think we should react to this somehow. I can definitely detect that I am running on RPI with armv7 and add those flags to default build methods. But that would make resulting binaries not work on original PI and on PI zero....

So I can:

- ignore armv7, leave it to user
- ignore PI 1 / 0 and the fact that resulting binary will not work on them, simply add damned flags
- create CLANG method for armv7 and CLANG_armv6 for backward compatibility with PI 1 / 0 (*)
- create CLANG method for armv6 and optimized CLANG_armv7

(*) is my currently the prefered plan...

Any ideas?

(also posting it here for future documentation reference).

Subject: Re: Rapsberry PI - cpu dilemma
Posted by Novo on Sat, 09 May 2020 04:11:45 GMT
View Forum Message <> Reply to Message

IMHO,
1) create CLANG method for armv7 and CLANG_armv6 for backward compatibility (maximum performance)
2) cross-compilation for Pi (nobody really wants to compile on Pi itself)
3) optimize code for small platform the way game developers do that (no exceptions, no RTTI). There is plenty of other stuff which can be optimized.

Basically, game developers have been doing this for decades. Just follow their steps ...

Subject: Re: Rapsberry PI - cpu dilemma
Posted by amrein on Sat, 09 May 2020 04:29:21 GMT
View Forum Message <> Reply to Message

It's just an opinion: completely useless complexity. If someone want to optimize their final binaries, they will add the correct flags themselves according to their c++ compiler.

We don't do this for amd, amd rizen of intel, so why bother?

Note: Raspberry Pi 2 Model B v1.2 and several other Raspberry Pi are ARM8 and you can boot some of them with the 64 kernel (available in the default Rasbian distribution). So, do we need to enable 64 bit support too?

https://en.wikipedia.org/wiki/Raspberry_Pi

To much complexity for small return on investment.

Subject: Re: Rapsberry PI - cpu dilemma
Posted by Novo on Sat, 09 May 2020 05:08:09 GMT
View Forum Message <> Reply to Message

amrein wrote on Sat, 09 May 2020 00:29
We don't do this for amd, amd rizen of intel, so why bother?

Do you mean why somebody wants to compile code without exception support?
For the same reason why Sony removed exception support from Clang for their PlayStation 4, which is based on amd rizen.
Exception support is a complicated thing, especially on 32-bit platforms.
World of small devices is very different from regular PCs. People do care about instruction cache misses there.

Subject: Re: Rapsberry PI - cpu dilemma
Posted by amrein on Sat, 09 May 2020 05:32:38 GMT
View Forum Message <> Reply to Message

What I mean is: we don't need to bother because if a developer needs to (say, he wants to gain 1 ms on his embedded application for x reasons), he will find his way himself.

We don't need to do anything.

Subject: Re: Rapsberry PI - cpu dilemma
Posted by mirek on Sat, 09 May 2020 08:21:22 GMT
View Forum Message <> Reply to Message

Novo wrote on Sat, 09 May 2020 06:11IMHO,
2) (nobody really wants to compile on Pi itself)

Actually, it is not as bad. Yes, it is about 5-10 times slower, but it reached my usability limit.

Mirek

Subject: Re: Rapsberry PI - cpu dilemma
Posted by mirek on Sat, 09 May 2020 08:23:25 GMT
View Forum Message <> Reply to Message

Unfortunately, I have just found that CLANG crashes when compiling theide for armv7, so the whole thing is sort of pointless now...

Subject: Re: Rapsberry PI - cpu dilemma
Posted by Novo on Sat, 09 May 2020 15:16:32 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 09 May 2020 04:23Unfortunately, I have just found that CLANG crashes when compiling theide for armv7, so the whole thing is sort of pointless now...
Clang crashes when compiling BenchmarkTess on Intel. This is not a reason to stop supporting Intel/AMD.
Eventually, Clang will be fixed and ARM will become a major platform.
It is good to be ready for that, IMHO.

Subject: Re: Rapsberry PI - cpu dilemma
Posted by Novo on Sat, 09 May 2020 15:25:11 GMT
View Forum Message <> Reply to Message

amrein wrote on Sat, 09 May 2020 01:32What I mean is: we don't need to bother because if a developer needs to (say, he wants to gain 1 ms on his embedded application for x reasons), he will find his way himself.

We don't need to do anything.
It is not just "1 ms". All these optimizations are about running code several times faster and using several times less memory.
Compiler is not a magic wand. -O3 enables almost all optimizations, but you still need to tell your compiler in the code that a particular data structure is moveable ...
The same story is with aliasing ...

Subject: Re: Rapsberry PI - cpu dilemma
Posted by Novo on Sat, 09 May 2020 16:05:35 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 09 May 2020 04:23Unfortunately, I have just found that CLANG crashes when compiling theide for armv7, so the whole thing is sort of pointless now...
Most likely, cross-compiler won't crash

Subject: Re: Rapsberry PI - cpu dilemma
Posted by mirek on Sat, 09 May 2020 18:55:51 GMT
View Forum Message <> Reply to Message

Novo wrote on Sat, 09 May 2020 17:25amrein wrote on Sat, 09 May 2020 01:32What I mean is: we don't need to bother because if a developer needs to (say, he wants to gain 1 ms on his embedded application for x reasons), he will find his way himself.

We don't need to do anything.
It is not just "1 ms". All these optimizations are about running code several times faster and using several times less memory.

Several times is gross exaggeration. So far I have seen about 20% speedup in single particular benchmark (basically Stream::Get32 loop).

That said, I think this is definitely worth investigating, for various reasons.

Mirek