1. About
2. Products
3. For Teams

1. Stack Internal Implement a knowledge platform layer to power your enterprise and AI tools.
2. Stack Data Licensing Get access to top-class technical expertise with trusted & attributed content.
3. Stack Ads Connect your brand to the world's most trusted technologist communities.
4. Releases Keep up-to-date on features we add to Stack Overflow and Stack Internal.
5. About the company Visit the blog

Loading…

current community

Stack Overflow

help chat

- Meta Stack Overflow

your communities

Sign up or log in to customize your list.

more stack exchange communities

company blog

3. Log in
4. Sign up

AI Assist is now on Stack Overflow. Start a chat to get instant answers from across the network. Sign up to save and share your chats.

Home
Questions
AI Assist
Tags
Challenges
7. Chat
Articles
Users
Companies
Collectives

Communities for your favorite technologies. Explore all Collectives

Stack Internal

Stack Overflow for Teams is now called **Stack Internal**. Bring the best of human thought and AI automation together at your work.

Try for free Learn more

Stack Internal

Bring the best of human thought and AI automation together at your work. Learn more

Collectives™ on Stack Overflow

Find centralized, trusted content and collaborate around the technologies you use most.

Learn more about Collectives

Stack Internal

Knowledge at work

Bring the best of human thought and AI automation together at your work.

Explore Stack Internal

Why am I getting a VFP register arguments while adding a custom target to rust (arm-unknown-linux-uclibceabihf)?

Asked 5 years, 4 months ago

Modified 5 years, 1 month ago

Viewed 1k times

0

I am following the guide from the last rust newsletter: https://rust-embedded.github.io/embedonomicon/custom-target.html and encounter an issue related to soft/hard float configuration.

I created my new target by merging the mipsel-unknown-linux-uclibc.json with the arm-unknown-linux-gnueabihf.json to obtain the following result:

```
{
 "arch": "arm",
 "data-layout": "e-m:e-p:32:32-Fi8-i64:64-v128:64:128-a:0:32-n32-S64",
 "dynamic-linking": true,
 "env": "uclibc",
 "executables": true,
 "features": "+strict-align,+v6,+vfp2",
 "has-elf-tls": true,
 "has-rpath": true,
 "is-builtin": false,
 "linker-flavor": "gcc",
 "linker-is-gnu": true,
 "linker": "/home/ykoehler/work/tools/arm-5.3-uclibc-1.0.12/usr/bin/arm-buildroot-linux-uclibcgnueabihf-gcc",
 "llvm-target": "arm-unknown-linux-uclibceabihf",
 "max-atomic-width": 64,
 "os": "linux",
 "position-independent-executables": true,
 "pre-link-args": {
   "gcc": [
     "-Wl,--as-needed",
     "-Wl,-z,noexecstack"
   ]
 },
 "relro-level": "full",
 "target-c-int-width": "32",
 "target-endian": "little",
 "target-family": "unix",
 "target-mcount": "_mcount",
 "target-pointer-width": "32",
 "vendor": "unknown"
}
```

When I compile a no-std app with cargo using the new custom target:

```
cargo build -Z build-std=core --target arm-unknown-linux-uclibceabihf.json
```

I am getting the following error:

```
   Compiling app v0.1.0 (/home/ykoehler/work/app)
error: linking with `/home/ykoehler/work/tools/arm-5.3-uclibc-1.0.12/usr/bin/arm-buildroot-linux-uclibcgnueabihf-gcc` failed:
 |
 = note: "/home/ykoehler/work/tools/arm-5.3-uclibc-1.0.12/usr/bin/arm-buildroot-linux-uclibcgnueabihf-gcc" "-Wl,--as-needed" "
 = note: /home/ykoehler/work/tools/arm-5.3-uclibc-1.0.12/usr/bin/../lib/gcc/arm-buildroot-linux-uclibcgnueabihf/5.3.0/../../..
         /home/ykoehler/work/tools/arm-5.3-uclibc-1.0.12/usr/bin/../lib/gcc/arm-buildroot-linux-uclibcgnueabihf/5.3.0/../../..
         /home/ykoehler/work/tools/arm-5.3-uclibc-1.0.12/usr/arm-buildroot-linux-uclibcgnueabihf/sysroot/usr/lib/Scrt1.o: In f
         /var/tmp/tool-chains/buildroot-2016.02-5.3-arm/src/buildroot-2016.02/output/build/uclibc-1.0.12/libc/sysdeps/linux/ar
         /var/tmp/tool-chains/buildroot-2016.02-5.3-arm/src/buildroot-2016.02/output/build/uclibc-1.0.12/libc/sysdeps/linux/ar
         /var/tmp/tool-chains/buildroot-2016.02-5.3-arm/src/buildroot-2016.02/output/build/uclibc-1.0.12/libc/sysdeps/linux/ar
         collect2: error: ld returned 1 exit status


error: aborting due to previous error

error: could not compile `app`.
```

```
To learn more, run the command again with --verbose.
```

This error indicate that some code is compiled with soft float usage when other part is using hard float, from reading over the net about the error message. Yet, both files specified by the error appears to have been generated by rust, and I am unclear why rust would generate any soft float code based on the information I provided inside the custom target definition file.

Any help appreciated.

- [rust](#)
- [arm](#)
- [target](#)

1 Answer

1

I believe you just need to use `arm-unknown-linux-gnueabihf` for the `llvm-target`. The VFP error is related to the floating-point instruction set available (i.e. the hf in eabihf stands for Hard Float). I'm working on getting the `armv7-unknown-linux-uclibceabihf` target working in rust. I received the same VFP error while getting the `armv7-unknown-linux-uclibceabihf` target to work in rust and it turned out I just needed to use `armv7-unknown-linux-gnueabihf` for the `llvm-target`.

Also, here's my spec file for `armv7-unknown-linux-uclibceabihf` for comparison:

```
{
 "abi-blacklist": [
   "stdcall",
   "fastcall",
   "vectorcall",
   "thiscall",
   "win64",
   "sysv64"
 ],
 "arch": "arm",
 "data-layout": "e-m:e-p:32:32-Fi8-i64:64-v128:64:128-a:0:32-n32-S64",
 "dynamic-linking": true,
 "env": "uclibc",
 "executables": true,
 "features": "+v7,+vfp3,-d32,+thumb2,-neon",
 "has-elf-tls": true,
 "has-rpath": true,
 "linker-flavor": "gcc",
 "linker-is-gnu": true,
 "llvm-target": "armv7-unknown-linux-gnueabihf",
 "max-atomic-width": 64,
 "os": "linux",
 "position-independent-executables": true,
 "pre-link-args": {
   "gcc": [
     "-Wl,--as-needed",
     "-Wl,-z,noexecstack"
   ]
 },
 "relro-level": "full",
 "target-c-int-width": "32",
 "target-endian": "little",
 "target-family": "unix",
 "target-mcount": "\u0001__gnu_mcount_nc",
 "target-pointer-width": "32",
 "vendor": "unknown"
```

```
    }
```

Cheers

Improve this answer

Follow

edited Oct 31, 2020 at 13:48

answered Oct 30, 2020 at 21:35

Tudor Sidea

2633 bronze badges

Sign up to request clarification or add additional context in comments.

6 Comments

Add a comment

Yannick Koehler

Yannick Koehler Over a year ago

Hi, so llvm-target would be use to "compile" but then at link time I use the uclibc to link with? Is that the trick you are suggesting here?

2020-11-01T01:54:00.33Z+00:00

0

Reply

• Copy link

Tudor Sidea

Tudor Sidea Over a year ago

Yes, exactly. You are telling llvm that it needs to compile for hard float by using `gnueabihf` at the end. I'm not exactly sure why llvm doesn't compile for hard float if you put `uclibceabihf` at the end, but at least `gnueabihf` works for llvm. Then the linker will know how to link the files with uclibc since you need to install the linker with uclibc. You can take a look at the arm-unknown-linux-musleabihf target as well, it does the same thing.

2020-11-02T07:12:17.107Z+00:00

0

Reply

• Copy link

Yannick Koehler

Yannick Koehler Over a year ago

Were you able to submit your change to rust git repo as to get that target supported? using your recommendation I got a stage 1 compiler, trying stage2, and then will build a toolchain for rustup and test it out.

2020-11-21T05:44:21.627Z+00:00

0

Reply

• Copy link

Tudor Sidea

Tudor Sidea Over a year ago

I'm working on it. Here's a link to the topic I started regarding this target. You can get more info there and I also have a git repo that will build rust compiler for this target.

2020-11-22T10:06:04.903Z+00:00

0

Reply

• Copy link

Yannick Koehler

Yannick Koehler Over a year ago

ok, I have a successful rustc/core/libc/std toolchain right now and my binaries works great. I will see to submit a PR. I could not get to run ./x.py test library/std I am unclear what is required to get this working, I am getting an error about missing) parenthesis which I found on the net is related to incorrect arch binary run, meaning that somehow the test doesn't appear to kick in qemu and try to run the arm binary on x86 host (if I understood this well).

2020-11-24T14:44:28.833Z+00:00

0

Reply

• Copy link

Add a comment | Show 1 more comment

Your Answer

Thanks for contributing an answer to Stack Overflow!

- Please be sure to *answer the question*. Provide details and share your research!

But *avoid* …

- Asking for help, clarification, or responding to other answers.
- Making statements based on opinion; back them up with references or personal experience.

To learn more, see our [tips on writing great answers](#).

Sign up or [log in](#)

Sign up using Google

Sign up using Email and Password

Submit

Post as a guest

Name

Email

Required, but never shown

Post as a guest

Name

Email

Required, but never shown

Post Your Answer Discard

By clicking "Post Your Answer", you agree to our [terms of service](#) and acknowledge you have read our [privacy policy](#).

Start asking to get answers

Find the answer to your question by asking.

[Ask question](#)

Explore related questions

- [rust](#)
- [arm](#)
- [target](#)

See similar questions with these tags.

- The Overflow Blog
  [Introducing Stack Overflow AI Assist—a tool for the modern developer](#)

- Featured on Meta

Related

Hot Network Questions

Subscribe to RSS

To subscribe to this RSS feed, copy and paste this URL into your RSS reader.

Stack Overflow

- Questions
- Help
- Chat

Business

- Stack Internal
- Stack Data Licensing
- Stack Ads

Company

- About
- Press
- Work Here
- Legal
- Privacy Policy
- Terms of Service
- Contact Us
- Cookie Settings
- Cookie Policy

Stack Exchange Network

- Technology
- Culture & recreation
- Life & arts
- Science
- Professional
- Business
- API
- Data
- Blog
- Facebook
- Twitter
- LinkedIn
- Instagram