Fixing MC for ARM v7-A

Just a few corner cases…

how hard can it be?

MC Hammer?

- Framework for exhaustive testing of MC layer implementation for ARM
- First introduced at Euro LLVM 2012

- 35 issues found, fixes open sourced
- Regression tests added
- Currently **2** issues remains to be fixed in upstream LLVM

- Not an entirely painless effort
- Exposed some infrastructure issues in MC

Parsing isn't easy…

- … but shouldn't be harder than it has to be

- ARM v7-A has some features that are difficult to work with:

  1. many instruction "aliases"
  2. optional modifiers to instructions
  3. Several encodings dependent on size of an immediate, but also mechanisms to override
  4. Several types of representations for immediates requiring custom parsing and range checking

The parser that isn't a parser

- MC's "parser" is not a parser…
- String matcher with occasional quirky behaviour

1. Dependent on evaluation order of instruction templates
2. Order controlled by tablegen, can vary as side effect of commits
3. Essentially arbitrary. C++ code required to accept/reject operands
4. ARM asm parser heavily hardcoded; makes life hard
5. ARM-specific behaviour in tablegen

Decoder

- Tablegen syntax insufficient to completely declare how to decode instructions
- Some command words decode to different instructions depending on combinations of registers, not only opcode!
- Some instructions undefined for specific combinations of operands
- Decoding of some command words dependent on subtarget features – unavailable in custom decoders

In short…

- Assembly parsing
  - MC needs a true parser
- Decoding
  - Improve tablegen to reduce the need for custom decoders
  - New decoder logic to better disambiguate instruction forms
  - Needs API to expose architectural features to decoder
- Hardcoding used to work around these issues