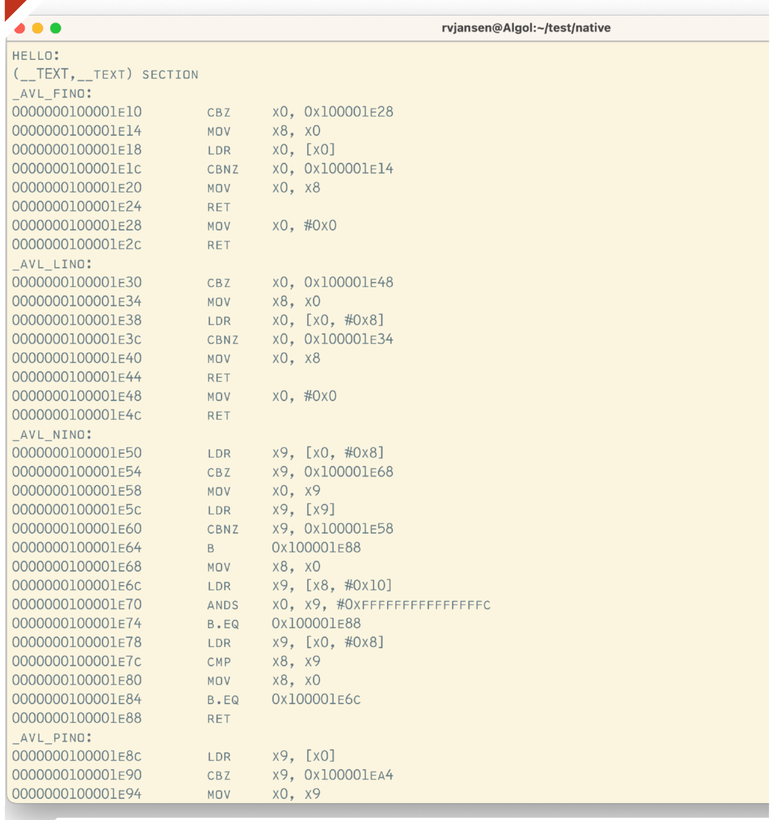
CREXX - Building a native executable

René Vincent Jansen, 34th International Rexx Language Symposium
Almere, 20230517

Why do this?



- With a native executable, it becomes possible to build a program that runs on a machine without any Rexx language tools installed
  - Needs to have the same ISA and OS (later more)
- Speed: at runtime we need to do less
- (When you do not want to hand over source)

What is native

- Native with respect to the
  - ISA
  - Executable File Format

Instruction set architecture (ISA)

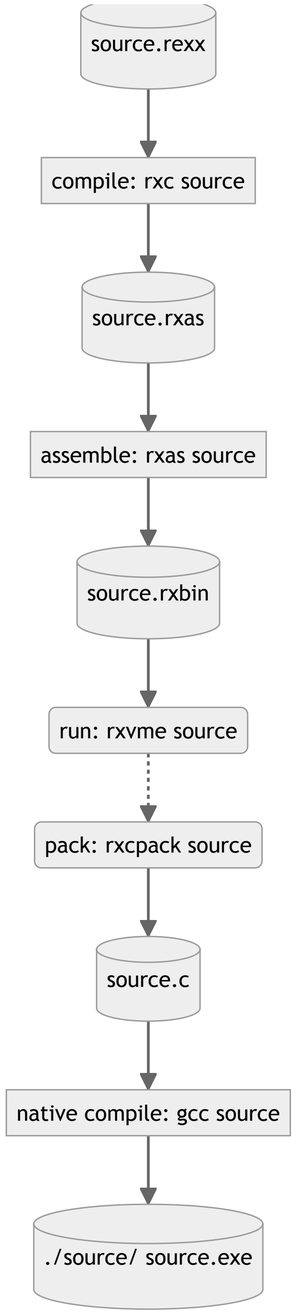
- Some important ones
  - IA86_64
  - AARCH64 (ARM64)
  - RISC/V
  - Z/ARCH (the mainframe)

  - GPU's and other vector processors

Tiered Compilation

- There is an interpreter that executes the RXVM instruction set
  - Which is already compiled when we built the interpreter
  - The Rexx byte code just jumps to these precompiled instructions
- The RXAS assembler source is compiled into RXVM code in .rxbin files

Build Chain


source.rexx
compile: rxc source
source.rxas
assemble: rxas source
source.rxbin
run: rxvme source
pack: rxcpack source
source.c
native compile: gcc source
./source/ source.exe

```
File Edit Options Buffers Tools Index Oorexx Help
/* REXX */
OPTIONS LEVELB
IMPORT RXFNSB
SAY 'HELLO cREXX WORLD!'
SAY 'TODAY IT''S' DATE('W')
SAY '2*21 IS:' 2*21
```

Simple Test

mind the options

mind the import

simple say

function

expression

```
File Edit Options Buffers Tools Asm Help
/*
 * cREXX COMPILER VERSION : cREXX F0045WIP1
 * SOURCE                 : HELLO.REXX
 * BUILT                  : 2023-05-17 08:24:02
 */

.SRCFILE="HELLO.REXX"
.GLOBALS=0

MAIN() .LOCALS=8
   .META "HELLO.REXX.MAIN"="B" ".VOID" MAIN() "" ""
   .SRC 4:1="SAY 'HELLO cREXX WORLD!'"
   SAY "HELLO cREXX WORLD!"
   .SRC 5:1="SAY 'TODAY IT''S' DATE('W')"
   LOAD R1,5
   LOAD R2,"W"
   SETTP R2,3
   SETTP R3,2
   SETTP R4,2
   SETTP R5,2
   SETTP R6,2
   CALL R7,DATE(),R1
   SCONCAT R6,"TODAY IT\'S",R7
   SAY R6
   .SRC 6:1="SAY '2*21 IS:' 2*21"
   SAY "2*21 IS: 42"
   .SRC 6:20=""
   RET

/* IMPORTED DECLARATION FROM FILE: HELLO.REXX */
DATE() .EXPOSE=RXFNSB.DATE
   .META "RXFNSB.DATE"="B" ".STRING" DATE() "?OFORMAT=.STRING,?IDATE=.STRING,?IFORMAT=.STRING,?OSEP=.STRING,?ISEP=.STRING"
```

The RXAS

look at how the imported declaration is done

look how the expression is pre-compiled

The C Packer

- This creates a C-program of a very peculiar nature
- Demonstration

Using GCC

- using the standard C/C++ compiler from GNU,
- gcc (which is an alias for clang on the mac) this large C program is compiled and linked into a native executable,
- and very possibly optimised even more.

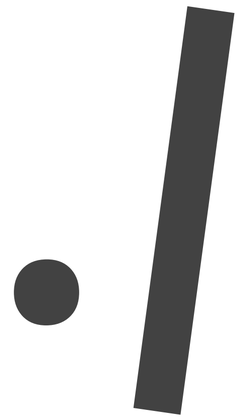
```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
THISDIR=$(dirname $0)
FILENAME=$(basename "$1")
PATH=$(dirname "$1")
EXTENSION="${FILENAME##*.}"
FILENAME="${FILENAME%.*}"

IF [ "$EXTENSION" = "$FILENAME" ]; THEN
  # If the filename has no extension, check for a file with .rexx extension
  FILENAME_REXX="${FILENAME}.rexx"
  IF [ ! -f "$FILENAME_REXX" ]; THEN
    ECHO "Error: $FILENAME_REXX does not exist."
    EXIT 1
  FI
  # Set extension to "rexx" if the file exists
  EXTENSION="rexx"
FI

CASE "$EXTENSION" IN
  rexx) TRUE
        ;;
  *)    ECHO "File extension is not correct. Expected .rexx file."
        EXIT 1
        ;;
ESAC

RXC  ${FILENAME}
RXAS ${FILENAME}
RXCPACK ${FILENAME} $THISDIR/../LIB/RXFNS/LIBRARY
GCC -O ${FILENAME} \
    -LRXVML -LMACHINE -LAVL_TREE -LPLATFORM -LM  \
    -L$THISDIR/../INTERPRETER \
    -L$THISDIR/../MACHINE \
    -L$THISDIR/../AVL_TREE \
    -L$THISDIR/../PLATFORM \
    ${FILENAME}.c

-UU-:----F1  crxc          All L37    (Shell-script[bash]) ----------------------------------------------------------------
End of buffer
```

This is the current build chain for native

in bash: blèèech!

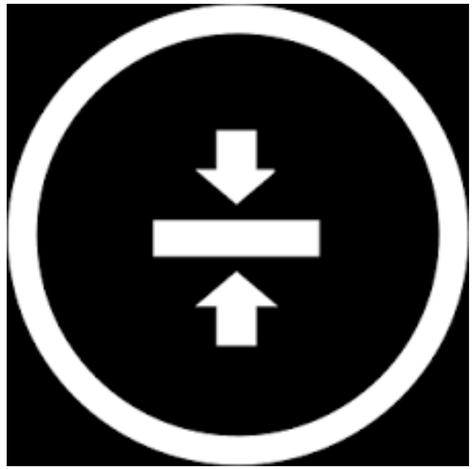
Running the executable

- can be on the PATH environment variable
- or started from current dir with ./



Size considerations

- Currently the executable file is quite big (±600-700K)
- This can be compressed by several utilities
  - but because of the dependency, we will look into that ourselves
    - but not with great haste



- depending on 3rd party products like UPX turned out not to be a great idea

Interesting Diversions

- LLVM
- FPGA
- Vectorizing
- Actually Portable Executables (APE)

The LLVM future

- LLVM is a construction set for generation of native code by compiler backends
- Extensive suite with lots of optimizations for different hardware
- Able to target most current hardware, including Z/Arch (the mainframe)

- The plan is to build an **rxas** --> **llvm** translator

RXAS FPGA

- Rexx in hardware!
  - Why not?

- This will need significant free time from someone(s)

Vectorizing hardware

- Some vector instructions will be already used on current IA_64 and ARM architectures
- Sometimes the compiler still needs a bit of help

- NVIDIA and other GPU-like processors
- Would also need time and love of motivated individuals

APE: Actually Portable Executable

- This is possible and demonstrated: search for Cosmopolitan
- Would be an interesting distribution format

Please be in touch
(and thanks for your attention!)

https://github.com/adesutherland/CREXX