IBM REXX COMPILER

BERT MOSER
IBM

IBM REXX Compiler

Bert Moser

IBM Vienna Software Development Lab
Wien 1, Cobdengasse 2

c/o IBM Austria
Obere Donaustrasse 95
A-1020 Austria
EUROPE

MOSER@VABVM1.IINUS1.IBM.COM

(+431) 21145-4476

May/91

Past

REXX and Interpreters

Present

CMS REXX Compiler COMPLEMENTs SPI

Future

REXX Compiler Improvements and Requirements

'79 Mike Cowlishaw becomes father of REXX

'83 Command Language for IBMs VM/CMS
'87 SAA Procedures Language
'89 REXX supported on MVS
'90 REXX supported on OS/2 and OS/400

2/89 IBM announces the CMS REXX Compiler

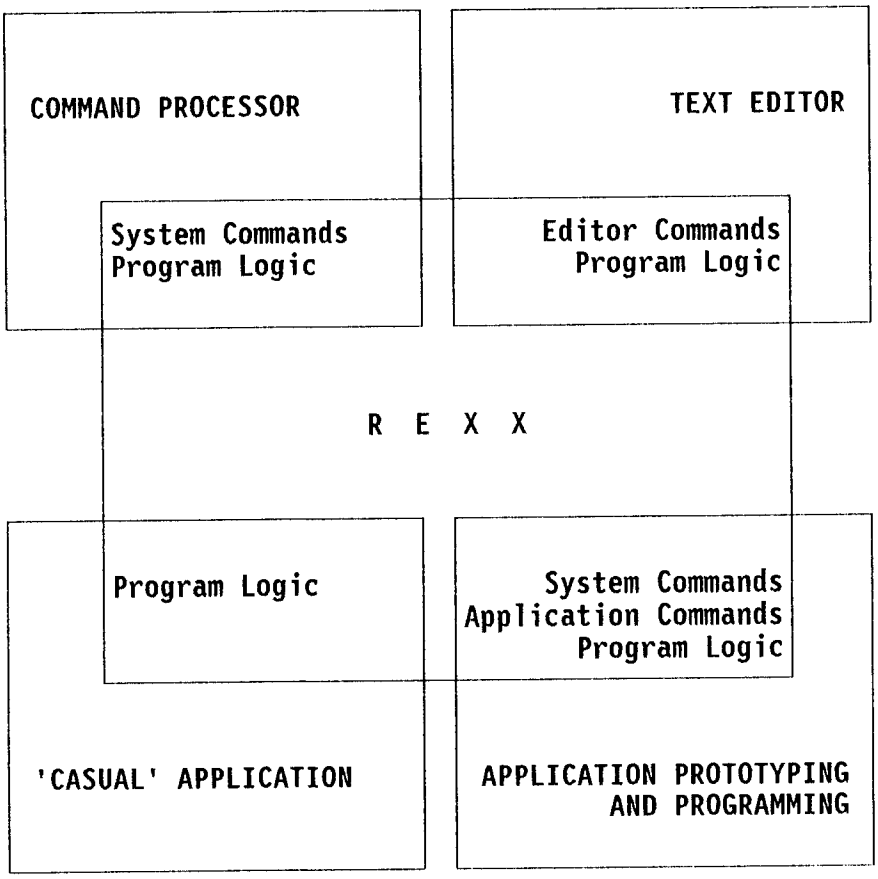
Available since 7/89

Developed by IBM Vienna Software Development Lab
Based on IBM Israel Scientific Center's feasibility study

- Compilability of REXX
- Appropriate run-time performance improvements

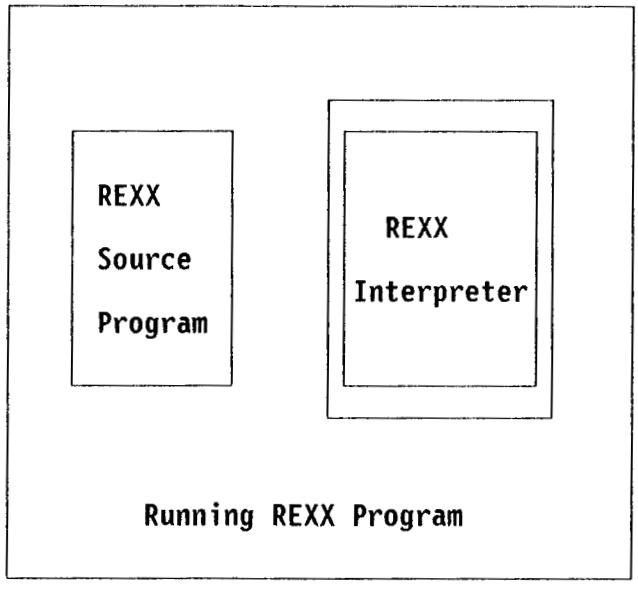
11/89 Library becomes a separate product

REXX Positioning



- REXX initially implemented by Interpreters
  - Excellent debugging features
  - Very short and appealing edit/run cycle
- HOWEVER
  - Better performance desirable

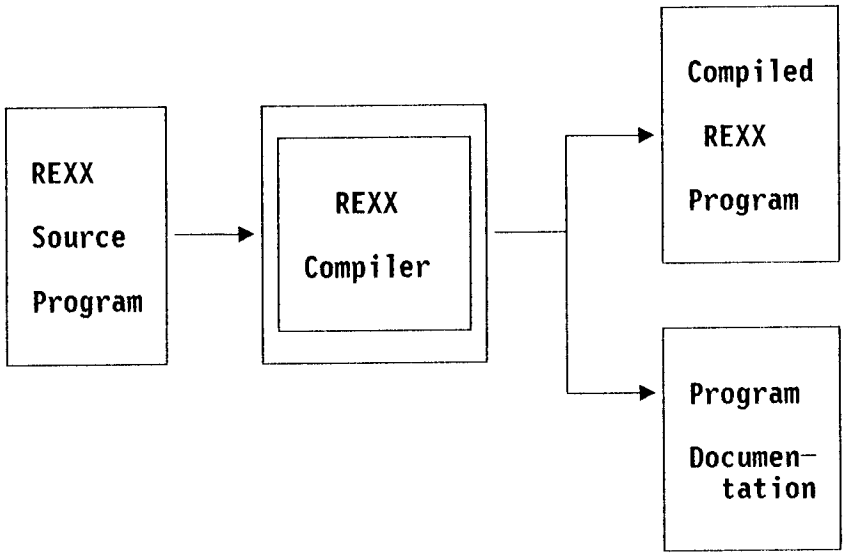
Single Step Approach



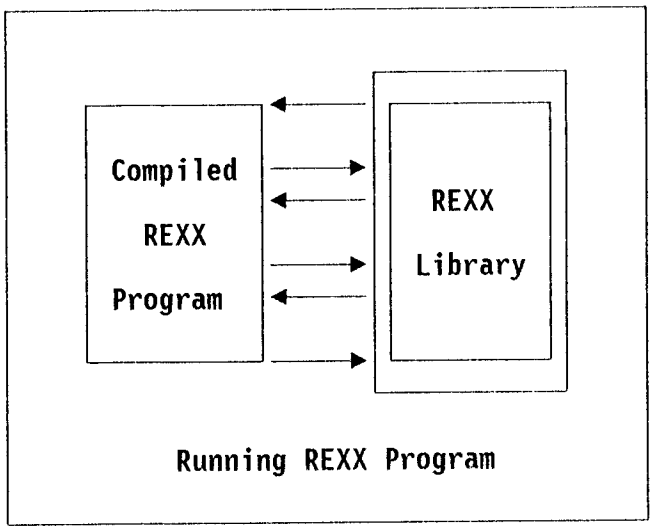
- Everything needs to be done at run-time
- On every REXX program invocation
- REXX source must be made available to every user

Two Step Approach

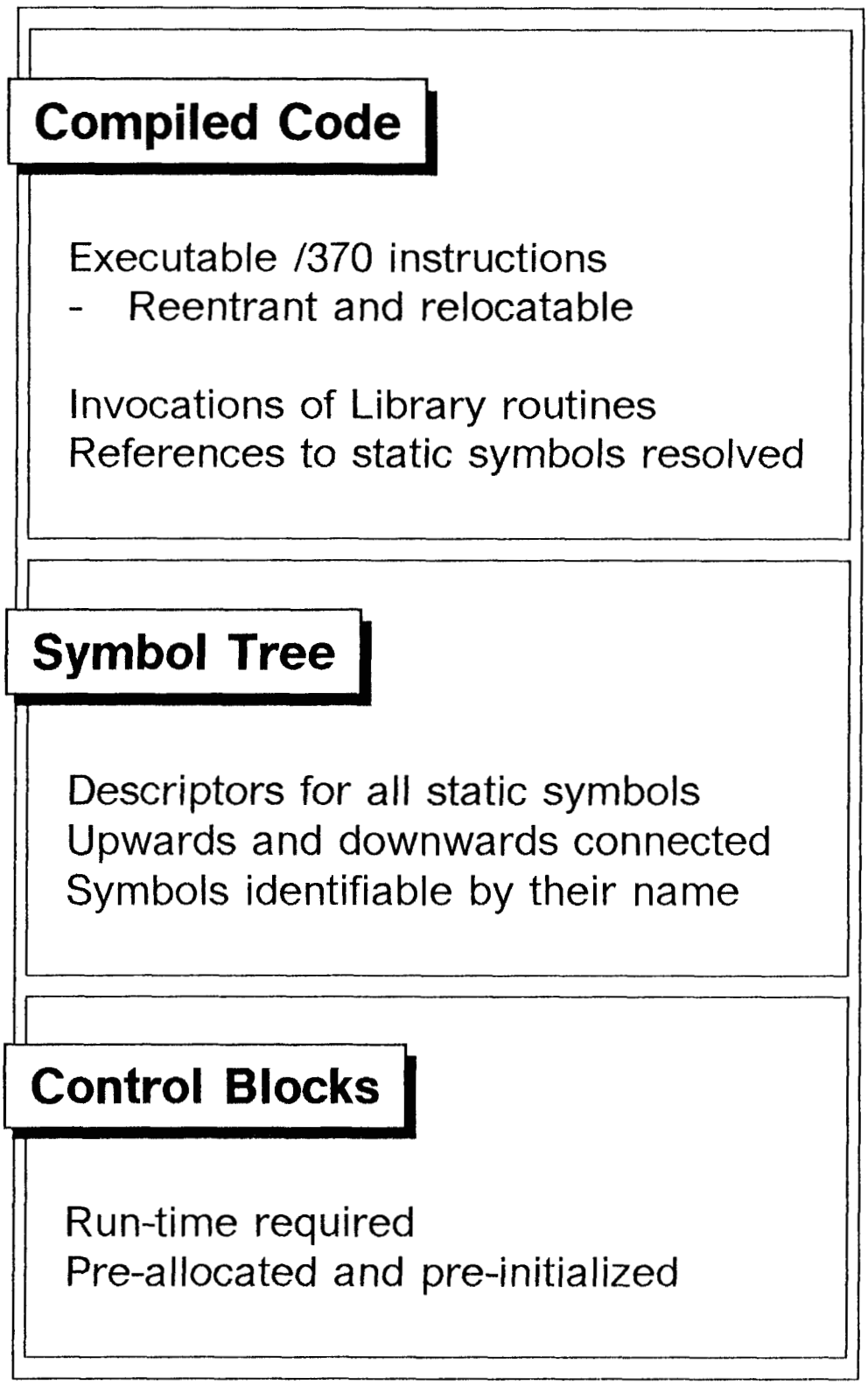
Compile



Run



Compiled REXX Program



EXEC-type

Same behavior as interpreted - "transparently" replace

- Same way of invocation and search sequence
- EXECLOADable
- Shared segment capability

Module-type

Other HLL compilers' object format (ESD, RLD, TXT,..)

- Linkable to other object programs
- Need to be LOADed - can be GEN'd into a module

- Search order is different
- CMS restriction: SVC-type arg/parmlist (PLIST)

Compiler

Set of phases performing all compilation tasks

- Compiled with IBM SAA C/370 Compiler
- Prerequisite when compiling:
  IBM C/370 Library V1 (5668-039) or later

Run-time Library

Routines invoked from compiled REXX programs

- Common to every compiled program

  Initialization, Termination, ...

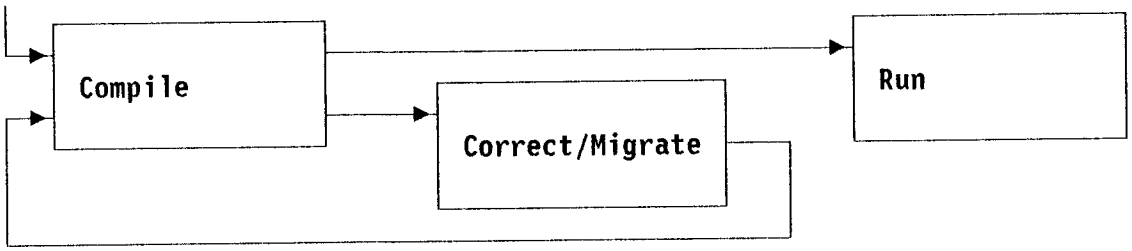
- Too bulky as to be copied to every program

  String Arithmetic, Conversions, Comparisons, Built-in Functions, Compound Variable Access/Handling, ...

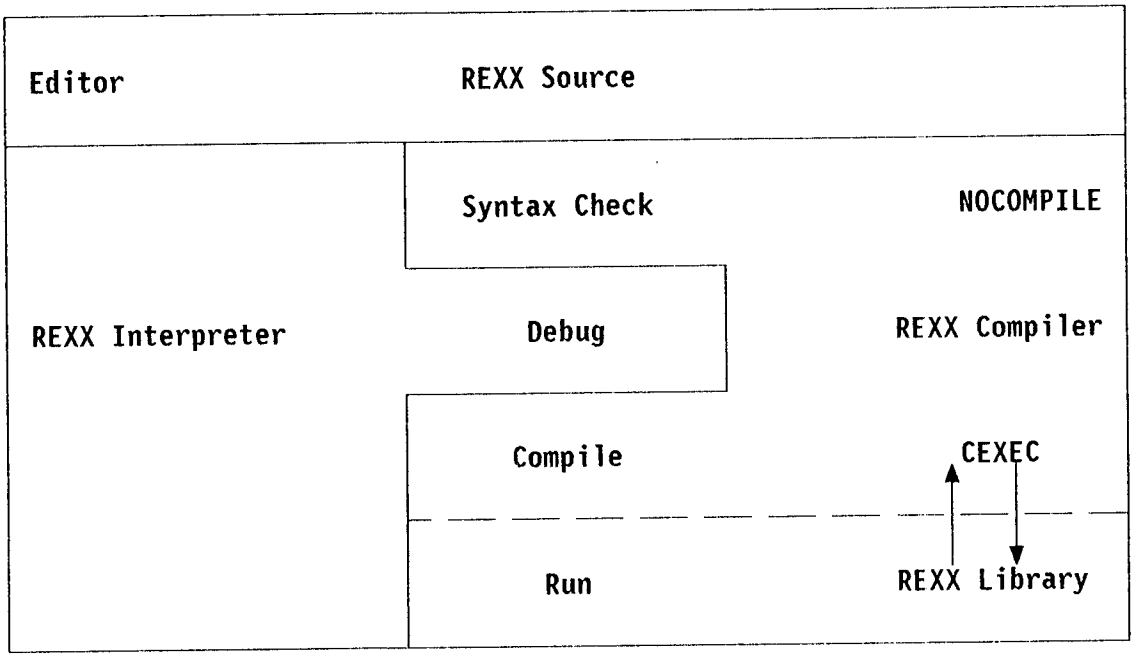
- Extremely time critical --> written in Assembler

Compiler/Interpreter COMPLEMENT Each Other

Existing REXX Programs



Newly Developed REXX Programs



Significantly Faster than Interpreted

"Plug-compatible" with Interpreted

Language Equivalent to Interpreter

"Unreadable" REXX Programs

Comprehensive Program Documentation

Comprehensive REXX Program Documentation

Source and Cross-Reference Listings
Syntax check whole program
More accurate messages
Begin debugging with more-correct programs

- Improve program quality
- Increase developer productivity

"Unreadable" Compiled REXX

Executable /370 code

- Provide program integrity
- Improved maintainability
- Protect REXX-coded assets

Source Listing Example

```
                                        SAMPLE   EXEC     G1

CMS REXX COMPILER 1.1.0      TIME: 11:35:02              DATE: 30 May 1989

 IF  DO SEL  LINE C ----+----1----+----2----+----3----+----4----+----5----

                1   /* SAMPLE incorrect REXX program */
                2
                3   Parse Arg Tmp
                4   val. = TRANSLATE(tmp)
                5   line.2 = LEFT(val.,2,'40')
                                              |
+++EAGGA00771S Invalid or missing argument(s) on built-in function
                6   $ = EXTFUNC(line.2)
                7   Call INTFUNC 2
                8   Exit
                9
               10   INTFUNK: Procedure Expose x. i
               11      Signal on NOVALUE NAME my_value
                                                 |
+++EAGGA00072S Label not found
               12      Do x.i
     1         13         If x.i//2 /= 0 then
  1  1         14            say "Odd: " x.i
     1         15         End
               16      Return
               17
               18   my_valu: Say "NOVALUE raised at: " sigl
               19      Return
               20   /* end of program SAMPLE
                    |
+++EAGGA00654S Unmatched "/*"
```

```
ITEM                            ATTRIBUTE LINE REFERENCES

------ LABELS, BUILT-IN FUNCTIONS, EXTERNAL RTNS ------

EXTFUNC                         EXT RTN   6
INTFUNC                         EXT RTN   7
INTFUNK                         LABEL     10(d)
LEFT                            BUILT-IN  5
MY_VALU                         LABEL     18(d)
MY_VALUE                        LABEL+++  11(u)
TRANSLATE                       BUILT-IN  4

------ CONSTANTS ------

'NOVALUE raised at: '           LIT STR   18
'Odd: '                         LIT STR   14
0                               NUMBER    13
2                               CONST SYM 5 6
2                               NUMBER    5 7 13
'40'                            LIT STR   5

------ SIMPLE VARIABLES ------

$                               SIMP VAR  6(s)
I                               SIMP VAR  10 12 13 14
SIGL                            SIMP VAR  18
TMP                             SIMP VAR  3(s) 4

------ STEMS AND COMPOUND VARIABLES ------

LINE.2                          COMP VAR  5(s) 6
VAL.                            STEM      4(s) 5
X.I                             COMP VAR  10 12 13 14
```

Language Equivalence with REXX Interpreter

NO compiler-specific language features !

- Minimize migration effort
- Almost all REXX programs run unchanged
  - except those with INTERPRET instructions

Flag Non-SAA Items - optional

Support SAA Procedures Language level 1.1

- Ease programming for multiple SAA environments

"Plug-Compatibility" with Interpreted Programs

Identical external interfaces - invocation and use

- "Transparently" replace interpreted
- No restriction on mutual invocation

31-Bit Capability

Compiler, Library, and Compiled Code **run** and **use storage** above the 16 Mega-byte line

- Make room for others below the line

No Conventional Block Structure

PROCEDURE is an executable instruction
- Not a syntactic boundary

Variables' life-time is dynamic
- Depends on calling sequence
- "Exposure" among procedures

No denotation of the END of a procedure
- Logical end is an executed RETURN

SIGNAL

Control can be transferred to everywhere
- Even into "procedure" and loop bodies

Computed GOTO - SIGNAL VALUE

No data types

All data is "character string"

Sometimes contents must be "numeric",
- Whole number", or "Boolean"

No declarations

Variables come and go - EXECCOMM/DROP

Can be shared with external programs

Names of variables can be computed
- Tails of compound variables

Value of variables only limited by storage
- Storage for values must be allocated dynamically

Arithmetic precision can be set dynamically
- NUMERIC DIGITS

Run-Time Performance Improvements

Performance gains depend on program mix

Programs with a lot of ...	TIMES faster than Interpreter	Performance Category
Default-precision Arithmetic	6 - 10+	VERY HIGH
String Arithmetic	4 - 25	
Assignments	6 - 10	
Changes to Variables' Values	4 - 6	HIGH
Constants and Simple Variables	4 - 6	
Reuse of Compound Variables	2 - 4	MEDIUM
Host Commands	1 -	LOW

- Up to 30% CPU load reduction reported - "... better than last CPU upgrade"
- On average 10% - 15%
- Savings example

```
Interpreted program runs     60     times a day (12 sec's)
                  using      12     min's CPU
   assume improvement of      6     times

            runs compiled      2     min's
```

INTERPRET Instruction not Supported

Rarely used

- Compiler diagnoses - no code generated

- Try to avoid

  Interpret target'='expr

  Call SETVAR target,expr

  RXSETVAR sample Assembler program
  User's Guide & Reference SH19-8120

- Restructure the program

  Isolate interpretative part
  Make it a separate program, and
  Let the Interpreter handle it

TRACE Instruction and Function not Supported

Does not change the semantics of a REXX program

- No need to change REXX program

  TRACE instruction - NOP instruction
  TRACE built-in function - "O"
  Interpreter default - "N"

- Diagnosed with an informational message

Save CPU Time & Reduce System Load

Improve Program Quality

Increase Developer Productivity

Protect REXX-Coded Assets

Allow to Keep Applications in REXX

Save Expensive Rewrites to Other HLL's

Attract to Write Even More REXX Applications

Reduce Storage Needed at Compile-Time

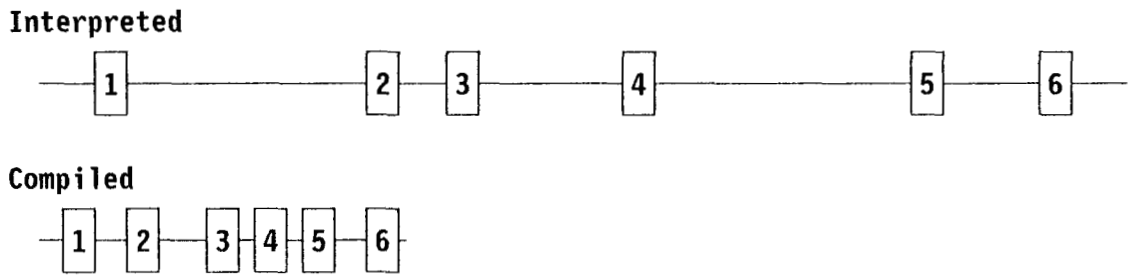
Improve Compiler's Performance

Improve Access/Handling of Compound Vars

- Binary tails
- Faster algorithms

Improve Built-in Functions

Compiled REXX Increases Paging



Running compiled on an I/O bound systems makes it WORSE

Compiled REXX Scatters Storage

Sorry for this one - was a bug
4 Bytes of a control block left over - sometimes

Compiled REXX Needs More Storage when Run

NO
Both implementations show similar storage consumption

Reduce Disk Space Needed by Compiled REXX

Remember: Code + Symbol Tree + Control Blocks

- Improve Assignment

  Special casing by Compiler = lot of code

  - Trade-off between performance and storage
  - Move case distinction to Library

- Compress Compiled Output

  Compiler option

  - Reduce med/large to size of source
  - Automatically de-compress
  - Reduce expensive I/O

Static Binding

- Allow to link external subroutines and functions

  For Module-type output only

  Compiler option
  Dualism - resolved address/dynamic invocation

  Tie together REXX-written application
  Function-package capability

Tailorable Cross Reference Listing

- Make Xref of CONSTANTs and LITERALs optional
- Compiler option XREF(S)

Library as "Test" Shared Segment

- Test new Library in parallel with "production" version

Relax Restriction on INTERPRET

- SEVERE ERROR - NO code generated

- Diagnose as ERROR - produce code

  Code should raise ERROR when executed
  - Allow to program around
  - Parse Version & REXXC370

  - Support development of multi-environmental programs

Implement INTERPRET

Long Range Consideration

Provide an MVS REXX Compiler

Accepted

MVS Compiler - Requirements

- Same language level as TSO/E Interpreter

- Same external interfaces - invocation and run

- Similar behavior and benefits as CMS REXX Compiler

- Cross compile ?

  - REXX code could run unchanged in VM and MVS
  - No need to re-compile

- Support MVS Parameter List Conventions

  - EFPL for external functions
  - CPPL for TSO/E commands
  - MVS JCL parameters for batch programs
  - CALL command parameters for foreground programs