VFX Forth for ARM 64 CPUs

Trials and Tribulations

Stephen Pelc, stephen@vfxforth.com, Wodni & Pelc GmbH, Sept 20225

In the beginning …
When ARM first appeared

- First ARM32 chips in 1985, DEC’s StrongARM in 1996
- Phillips LPC2000 marked start of ARM embedded explosion in 2004
- Many, many variants since
- ARM has often “upgraded” and “improved” instruction sets
- Primary 32 bit ISAs (Instruction Set Architectures)
  ARM32 - the original
  Thumb2 - 16/32 encoding of ARM32 ISA
- Leading up to ARM64

About ARM64

Beginning again

- ARM32 and T2 ISAs supported for legacy reasons
- ARM64 has 32 regs with new instruction set - 32 bit opcodes
- Most of the ARM64 books are not useful
- Most useful documentation is the
  “Compiler Writer’s Guide to the Power PC”
- Some instructions appear to be bizarre
- The cache is a PITA

ARM64 ISA - 1

Immediate data

- Arithmetic - 16 bit, 12 + shift, 12, 8, 7, 6
  large literals require several instructions
- Logical - N:immr6:imms6 is 13 bit mask
  about 5000 options used by **`AND`** **`ORR`** and **`EOR`**
- Branches - 26 bit, 19 bit, 14 bit
- Memory offset - s19, u12, s9, s7, 0
- Calls have +/-128 Mb range, conditionals +/-1 Mb range
  big improvement over many CPUs

ARM64 ISA - 2

Some PowerPC inheritances

- Conditional instructions

```
cmp         tos, x17       \ (n1-n2)-(n3-n2)
csinv tos, xzr, xzr, .ls \ cy -> -1, ncy -> 0
```

Conditional select invert

Destination reg = first or second source register processed by condition

Builds **0= 0<** and friends with no branches

CSEL CSET CSETM CSINC CSINV CSNEG

Caches

The pain begins

- Nothing like ARM32
- Separate level 1 Instruction and data caches
- Minimum cache line size is 64 byte
  size can be read at execution level 0 (user)
- Will require changes to the VFX64 kernel in the long term

Tools - 1

What we need

- Cross compiler - x64 -> ARM64
  Can use Rosetta under macOS and Linux
  Rosetta has bugs which can destroy integrity of the target image
  First bytes of new 4k page can be corrupt

- Back to plan B
  cross compile on x64 box
  deliver to ARM64 box
  debug on ARM64 Linux
  macOS later

Tools - 2

Both on x64 and ARM64 - in cross compiler and in target

- Assembler - prefix notation mostly follows ARM notation
- Disassembler - needed to debug code generator
- Code generator and optimiser
  1 and a bit passes with heuristics and backtracking
  data stack is modelled at compile time
- Required code - best done in assembler
- Operating system interface
  Externals and callbacks

Example 1

Store with Cache Invalidation - source

```
code l!c(t)  \ l addr --
\ *G Store and Flush instruction cache line containing *\i{addr}.
\ ** Use in stores of code.
  ldr x0, [ psp ]!, # 8   \ get l
  str w0, [ tos, # 0 ]    \ save l
  ic  ivau tos       \ flush
  dsb   SY        \ ensure completion of the invalidation
  isb   SY        \ ensure instruction fetch path sees
              \ new I cache state
  ldr   tos, [ psp ]!, # 8  \ restore TOS
  ret x30
end-code
```

•

Example 1
Disassembly

```
dis l!c(t)
L!C(T)
( 0041:1850 A08740F8 )              LDR       X0, [ XPSP  ]!, # $8
( 0041:1854 400300B9 )              STR       W0, [ XTOS, # $0 ]
( 0041:1858 3A750BD5 )              SYS       $1BA9 XTOS
( 0041:185C 9F3F03D5 )              DSB       # $0F
( 0041:1860 DF3F03D5 )              ISB       # $0F
( 0041:1864 BA8740F8 )              LDR       XTOS, [ XPSP  ]!, # $8
( 0041:1868 C0035FD6 )              RET       XLR  ( NEXT/EXIT )
28 bytes, 7 instructions.
•
```

Example 2

High level Forth - source

```
: InOvl?      \ addr1 -- addr2|0
\ *G Returns the overlay address (addr2) if the address (addr1)
\ ** is within an overlay, otherwise returns 0.
  ovl-link @
     begin                        \ -- addr *ovl
    dup
  while                           \ -- addr *ovl
    2dup OVI.end 2@ within if     \ -- addr *ovl
      nip  exit
    then
    ovi.link @
  repeat
  nip                             \ remove addr
;
```

Example 2

High level Forth - disassembly

```
dis inovl?
INOVL?
( 0042:9F88 00FF9AD2 )              MOVZ      X0, # $D7F8
( 0042:9F8C 2008A0F2 )              MOVK      X0, # $41 LSL # $10
( 0042:9F90 110040F8 )              LDUR      X17, [ X0, # $0 ]
( 0042:9F94 BA8F1FF8 )              STR       XTOS, [ XPSP, # $-8 ]!
( 0042:9F98 FA0311AA )              MOV       XTOS, X17
( 0042:9F9C 9E8F1FF8 )              STR       XLR, [ XRSP, # $-8 ]!
( 0042:9FA0 5F031FEB )              CMP       XTOS, XZR  LSL# $00
( 0042:9FA4 40020054 )              B .EQ     # $429FEC
( 0042:9FA8 50BF41A9 )              LDP    X16, X15, [ XTOS, # $18 ]
( 0042:9FAC BD6300D1 )              SUB       XPSP, XPSP, # $18
( 0042:9FB0 AF0300F9 )              STR       X15, [ XPSP, # $0 ]
( 0042:9FB4 A10F40F9 )              LDR       X1, [ XPSP, # $18 ]
( 0042:9FB8 A10700F9 )              STR       X1, [ XPSP, # $8 ]
( 0042:9FBC BA0B00F9 )              STR       XTOS, [ XPSP, # $10 ]
( 0042:9FC0 FA0310AA )              MOV       XTOS, X16
( 0042:9FC4 53A3FF97 )              BL        # $412D10     WITHIN
( 0042:9FC8 5F031FEB )              CMP       XTOS, XZR  LSL# $00
( 0042:9FCC BA8740F8 )              LDR       XTOS, [ XPSP  ]!, # $8
( 0042:9FD0 80000054 )              B .EQ     # $429FE0
( 0042:9FD4 BD230091 )              ADD       XPSP, XPSP, # $08
( 0042:9FD8 9E8740F8 )              LDR       XLR, [ XRSP  ]!, # $8
( 0042:9FDC C0035FD6 )              RET       XLR  ( NEXT/EXIT )
( 0042:9FE0 510340F8 )              LDUR      X17, [ XTOS, # $0 ]
( 0042:9FE4 FA0311AA )              MOV       XTOS, X17
( 0042:9FE8 CEFDFF54 )              B         # $429FA0
( 0042:9FEC BD230091 )              ADD       XPSP, XPSP, # $08
( 0042:9FF0 9E8740F8 )              LDR       XLR, [ XRSP  ]!, # $8
( 0042:9FF4 C0035FD6 )              RET       XLR  ( NEXT/EXIT )
112 bytes, 28 instructions.
 ok
```

Current status

Optimism at last

- VFX64 kernel runs (sort of) with x64 tools on macOS and Linux VM
- We are migrating to separate x64 and ARM64 development tools
- Production release needs
  - Compiler on ARM64
  - Enhanced code generator
  - Finish port of OS dependencies
  - First target ARM64 Linux

Thanks

- Questions?