Microcore Status Report

Klaus Schleisiek
SEND Off-Shore Electronics GmbH
Hamburg
ks@send.de

Overview

- Debugger in Gforth
- Objects & Methods
- VHDL code amalgamation
- Instruction overview
- Un-interruptible sequences
- Optimization
- uCore 2

Debugger in Gforth

- Cross compiler
- VHDL backend
- Interactive debugger with 2 wire umbilical
- Program memory change without processor state change
- Breakpoints
- interactive tracing

Objects & Methods

Class
inherit
New (Object)
:: (Attribute)
[ ] (Reference), is
Array

Class Point Point definitions
Cell :: X
Cell :: Y Point seal
M: @ (point -- x y) dup Self X @ Self Y @ ;
Forth definitions

VHDL code amalgamation

alu.vhd

dstack.vhd

rstack.vhd

sequencer.vhd

uBus.vhd

uCore.vhd => uCore.vhd

BRANCH group

```
00 nop          -0 branch       +0 dup          -0 drop
00 rot          ?0 ?dup-branch  ?0 ?dup         -0 pack
00 -rot         -0 s-branch     +0 tuck         -+ call
00 0<           -0 ns-branch    +0 under        -0 times
00 less?        -0 z-branch     +0 unpack       -? z-exit
0- rdrop        -0 nz-branch    +0 ovfl?        -? nz-exit
0- ldivs        -0 no-branch    +0 carry?       -? next
0- exit         -0 nc-branch    +0 I            -- iret
```

ALU group

00 mults	-0 +	+0 2dup +	00 invert
00 divs	-0 +c	+0 2dup +c	00 2*
	-0 -	+0 2dup -	00 2/
00 crcs	-0 swap-	+0 2dup swap-	00 u2/
00 drol	-0 and	+0 2dup and	00 ror
00 dror	-0 or	+0 2dup or	00 rol
00 +st (2 cyc)	-0 xor	+0 2dup xor	00 0=
00 swap	-0 nip	+0 2dup over	00 time?

MEM group

```
-0 status!      -0 ST          +0 LD          +0 status@
-0 st_set       -0 1 + ST      +0 1 + LD      +0 r@
-+ >r           -0 2 + ST      +0 2 + LD      +- r>
-0 lst          -0 3 + ST      +0 3 + LD      +0 lld
-0 rsp!         -0 4 - ST      +0 4 - LD      +0 rsp@
-0 dsp!         -0 3 - ST      +0 3 - LD      +0 dsp@
-0 tst          -0 2 - ST      +0 2 - LD      +0 tld
                -0 1 - ST      +0 1 - LD
```

USER group

```
-+ 0divs        -+ -divs
++ int
0+ exc
0? ?ovfl
0+ break
```

un-interruptible sequences

```
+st ( n addr -- addr )
: +!   +st drop ;
```

only 1st cycle is new and its Forth equivalent is

```
: (+st  ( n addr -- n+ addr )
  >r   r@ +  r> ;
```

2nd cycle executes a store instruction

An un-interruptible sequence !

bit banging the status register

```
set_st ( mask -- )

1 Constant #carry
#carry set_st
```

sets carry

```
#carry invert set_st
```

resets carry

instruction efficiency

```
: <              - less? ;
: >         swap - less? ;
: u>        swap - drop carry? 0= ;
: u<             - drop carry? 0= ;
```

Optimizations

```
<word> CALL ;   =>   <word> BRANCH
>r ;            =>   BRANCH
?dup IF         =>   ?dup_BRANCH
0= IF           =>   0<>BRANCH
0< IF           =>   NS-BRANCH
0< 0= IF        =>   S-BRANCH
carry? IF       =>   NC-BRANCH
ovfl? IF        =>   NO-BRANCH
```

Optimizations

```
over over +     =>  2dup_+
swap -          =>  swap-
swap over       =>  tuck
over swap       =>  under
<n> + LD        =>  +n_LD
```

uCore 2

Bit#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALL	1	15-bit abs. Address														
LIT	0	1	14-bit Lit													
BRANCH	0	0	1	1	12-bit signed branch offset											
?BRANCH	0	0	1	0	1	11-bit signed branch offset										
NEXT	0	0	1	0	0	11-bit signed branch offset										
MEM	0	0	0	1	1	pop	push	9-bit signed post-increment								
ZPAGE	0	0	0	1	0	pop	push	1	8-bit absolute address							
[RSP]	0	0	0	1	0	pop	push	0	1	7-bit offset						
[TASK]	0	0	0	1	0	pop	push	0	0	7-bit offset						
ALU	0	0	0	0	1	pop	push	exit	256 Operations							
REG	0	0	0	0	0	pop	push	exit	8-bit signed address							