Forth83 Benchmarks#

Below is a collection of some Benchmarks for Forth83 systems like VolksForth.

I found most of these benchmarks on comp.lang.forth, Hans Bzemers 4th and Marcel Hendrix benchmark collection

Table of Contents

Results#

Name	System	Forth	Benchmark	Time (sec/ round)	
Carsten Strotmann	Atari Portfolio 8088 4.92Mhz	VolksForth MS-DOS (ITC)	Integer Calc	4.96	
Carsten Strotmann	Amstrad NC100 Z80 4.606Mhz	VolksForth CP/M (ITC)	Integer Calc	6.23	
Martin Metz	Amstrad NC100 Z80 4.606Mhz	VolksForth CP/M (ITC)	GCD 1	38.1	
Andreas Böhm	Commodore C64 6510	Audiogenic Forth-64	Integer Calc	526	
Andreas Böhm	Commodore C64 6510	Audiogenic Forth-64	Count Bits	140.22	
Andreas Böhm	Commodore C64 6510	Audiogenic Forth-64	Sieve Bench	18.1	
Andreas Böhm	Commodore C64 6510	Audiogenic Forth-64	GCD 1	215.52	
Andreas Böhm	Commodore C64 6510	Audiogenic Forth-64	GCD 2	84.84	
H. Jakob	c't 86 8086 5Mhz	Laxen/Perry F83	Integer Calc	9	
Neil Franklin	HP 100LX 80186 7.9Mhz	VolksForth 3.81.41 MS-DOS	Integer Calc	2.8	
Carsten Strotmann	Atari 130XE 6502 1.79Mhz	VolksForth 3.81	Integer Calc	596	
Carsten Strotmann	Atari 130XE 6502 1.79Mhz noDMA	VolksForth 3.81	Integer Calc	438	
J. Kunz	DEC 3000-600 Alpha 21064 175Mhz	pForth	Integer Calc	0.091	
J. Kunz	DEC 3000-600 Alpha 21064 175Mhz	pForth	Fibonacci 1	0.0038	
J. Kunz	DEC 3000-600 Alpha 21064 175Mhz	pForth	Fibonacci 2	0.00001425	
J. Kunz	DEC 3000-600 Alpha 21064 175Mhz	pForth	Nesting 32Mil	22	
J. Kunz	DEC 3000-600 Alpha 21064 175Mhz	pForth	6502emu	18	
Ingo Soetebier	Nextstation 68040 33Mhz	pfe	Nesting 1Mil	340	
KC85 Team	KC85/4 U880 4Mhz	VolksForth CP/M	Nesting 1Mil	144	
Venty	Thinkpad T61, 2Ghz Core Duo	gforth-fast, Linux	Integer Calc	0.0013	
Venty	Thinkpad T61, 2Ghz Core Duo	gforth, Linux	Integer Calc	0.0019	
Venty		gforth-fast, Linux	Nesting 32Mil	3.9	

	Nokia N900 ARM A8 600Mhz				
Venty	Nokia N900 ARM A8 600Mhz	gforth-fast, Linux	Sieve Bench	0.015	
Venty	Nokia N900 ARM A8 600Mhz	gforth-fast, Linux	6502emu	1	
Venty	Nokia N900 ARM A8 600Mhz	gforth-dtc, Linux	Nesting 32Mil	5.5	
Venty	Nokia N900 ARM A8 600Mhz	gforth-dtc, Linux	Sieve Bench	0.025	
Venty	Nokia N900 ARM A8 600Mhz	gforth-dtc, Linux	6502emu	1.8	
Venty	Nokia N900 ARM A8 600Mhz	gforth-itc, Linux	Nesting 32Mil	6.9	
Venty	Nokia N900 ARM A8 600Mhz	gforth-itc, Linux	Sieve Bench	0.028	
Venty	Nokia N900 ARM A8 600Mhz	gforth-itc, Linux	6502emu	2.2	
Thorsten Kuphaldt	Amiga 3000 68030 25Mhz	jforth	Integer Bench	0.24	
Thorsten Kuphaldt	Amiga 3000 68030 25Mhz	jforth	Nesting 1Mil	1.32	
Thorsten Kuphaldt	Amiga 3000 68030 25Mhz	jforth	Memory Move	0.67	
Thorsten Kuphaldt	Amiga 3000 68030 25Mhz	jforth	Sieve Bench	0.148	
Thorsten Kuphaldt	Amiga 3000 68030 25Mhz	jforth	GCD 1	0.64	
Stefan Herold	Amstrad 6128+ Z80A 4Mhz	Uniforth	Integer Calc	17	
Stefan Herold	Amstrad 6128+ Z80A 4Mhz	Uniforth	Fibonacci 2	0.23	
Stefan Herold	Amstrad 6128+ Z80A 4Mhz	Uniforth	Nesting 1Mil	206	
Stefan Herold	Amstrad 6128+ Z80A 4Mhz	Uniforth	Sieve Bench	12	
Ingo Soetebier	iBook PPC 750lx (G3) 600Mhz	OpenFirmware	Integer Calc	0.03	
Ingo Soetebier	iBook PPC 750lx (G3) 600Mhz	OpenFirmware	Fibonacci 1	0.0026	
Ingo Soetebier		OpenFirmware	Fibonacci 2	0.0027	

	iBook PPC 750lx (G3) 600Mhz				
Ingo Soetebier	iBook PPC 750lx (G3) 600Mhz	OpenFirmware	Nesting 1Mil	1	
Ingo Soetebier	iBook PPC 750lx (G3) 600Mhz	OpenFirmware	Sieve Bench	0.031	
Ingo Soetebier	iBook PPC 750lx (G3) 600Mhz	OpenFirmware	GCD 1	0.024	
Michael Kalus	Rockwell R1200-14, 2Mhz 65F12	RSC-Forth	Fibonacci 1	16.09	
Michael Kalus	Rockwell R1200-14, 2Mhz 65F12	RSC-Forth	Fibonacci 2	0.05	
Michael Kalus	Rockwell R1200-14, 2Mhz 65F12	RSC-Forth	Nesting 1Mil	149	
Michael Kalus	Rockwell R1200-14, 2Mhz 65F12	RSC-Forth	Integer Calc	31	
Matthias Trute	Atmega16 8MHz	amForth 4.4	Integer Calc	1.56	
Matthias Trute	Atmega16 8MHz	amForth 4.4	Fibonacci 1	1.46	
Matthias Trute	Atmega16 8MHz	amForth 4.4	Fibonacci 2	0.0047	
Matthias Trute	Atmega16 8MHz	amForth 4.4	Nesting 1Mil	15.4	
Matthias Trute	Atmega16 8MHz	amForth 4.4	Nesting 32Mil	489	
Matthias Trute	Atmega16 8MHz	amForth 4.4	GCD 1	7.12	
Matthias Trute	Atmega16 8MHz	amForth 4.4	GCD 2	10.5	
Matthias Trute	Atmega16 8MHz	amForth 4.4	Takeuchi	0.7	
Michael Kalus	MSP430FR5739, 8Mhz DCO intern MSP-EXP430FR5739 Experimenter Board	CamelForth	Integer Calc 100x	02'45':10	
Michael Kalus	MSP430FR5739, 8Mhz DCO intern MSP-EXP430FR5739 Experimenter Board	CamelForth	FIB1 100x	00'46':39	
Michael Kalus		CamelForth	FIB2 10000x	00'16':91	

	MSP430FR5739, 8Mhz DCO intern MSP-EXP430FR5739 Experimenter Board				
Michael Kalus	MSP430FR5739, 8Mhz DCO intern MSP-EXP430FR5739 Experimenter Board	CamelForth	Nesting 32Mil	02'31':23	
Carsten Strotmann	IBM L40S (386SX)	mina (Fig-Forth)	Fib2 (1000)	8s	
Carsten Strotmann	IBM L40X (386SX)	F83 (Laxen & Perry)	Fib2 (1000)	8s	
Carsten Strotmann	IBM L40X (386SX)	GNU Forth 0.5.0 ec8086	Fib2 (1000)	24s	
Carsten Strotmann	IBM L40X (386SX)	VolksForth MS-DOS	Fibonacci 1	0.36s	
Thorsten Schoeler	Sinclair Spectrum+	Aber Forth (FIG-Forth)	Integer	25s	
Thorsten Schoeler	"	"	prime	11s	
Thorsten Schoeler	"	"	Nesting 1m	3m17s	
Thorsten Schoeler	"	"	GCD1	2m14s	
Thorsten Schoeler	"	"	Fib2 (1000)	1m46s	
Thorsten Schoeler	HX-20	Epson Forth E1.0	Fib2 (1000)	3m16s	
Thorsten Schoeler	HX-20	"	Nesting 32mil	2h43m49s	
Thorsten Schoeler	HX-20	"	Nesting 1mil	5m08s	
Thorsten Schoeler	HX-20	"	Integer 32tsd	1m03s	
Thorsten Schoeler	HX-20 6301 614khz	"	Prime	23s	
Wolfgang Stief	SUN SparcStation 10 TI TMS390255	OpenFirmware	Integer	0,14s	
Wolfgang Stief	SUN SparcStation 10 TI TMS390255	OpenFirmware	Fib1	0,005s	
Wolfgang Stief	SUN SparcStation 10 TI TMS390255	OpenFirmware	Fib2	0,2s	
Wolfgang Stief	SUN SparcStation 10 TI TMS390255	OpenFirmware	Memory Move	143s	
Wolfgang Stief		OpenFirmware	Prime	0,11s	

	SUN SparcStation 10 TI TMS390255				
Wolfgang Stief	SUN SparcStation 10 TI TMS390255	OpenFirmware	GCD1	0,51s	
Wolfgang Stief	SUN SparcStation 10 TI TMS390255	OpenFirmware	GCD2	0,65s	
Wolfgang Stief	SUN SparcStation 10 TI TMS390255	OpenFirmware	Takeuchi	0,06s	
Wolfgang Stief	SUN Ultra 1 200 Mhz UltraSprac	OpenBoot 3.25	Integer	0,33s	
Wolfgang Stief	SUN Ultra 1 200 Mhz UltraSprac	OpenBoot 3.25	Fib1	0,014s	
Wolfgang Stief	SUN Ultra 1 200 Mhz UltraSprac	OpenBoot 3.25	Fib2	0,06s	
Wolfgang Stief	SUN Ultra 1 200 Mhz UltraSprac	OpenBoot 3.25	Nesting 32mil	9s	
Wolfgang Stief	SUN Ultra 1 200 Mhz UltraSprac	OpenBoot 3.25	Mempry Move	0,014s	
Wolfgang Stief	SUN Ultra 1 200 Mhz UltraSprac	OpenBoot 3.25	Prime	0,03s	
Wolfgang Stief	SUN Ultra 1 200 Mhz UltraSprac	OpenBoot 3.25	GCD1	0,08s	
Wolfgang Stief	SUN Ultra 1 200 Mhz UltraSprac	OpenBoot 3.25	GCD2	0,11s	
Wolfgang Stief	SUN Ultra 1 200 Mhz UltraSprac	OpenBoot 3.25	Takeuchi	0,009s	
Thorsten Schoeler	Fignition (ATMEL)	Fignition Forth	fib2 (1000)	13s	
Stefan Niestegge	Atari Falcom 68060 100mhz	f68kans	Integer	0,022s	
Stefan Niestegge	Atari Falcon 68060 100mhz	f68kans	Fib2	0,0012s	
Stefan Niestegge	Atari Falcon 68060	f68kans	Countbits	0,05s	
Stefan Niestegge	Atari Falcon 68060	f68kans	GCD1	0,063s	
Stefan Niestegge	Atari Falcon 68060	f68kans	GCD2	0,067s	
Stefan Niestegge	Atari Falcon 68060	f68kans	Nesting 32mil	7,4s	
Thorsten Kuphaldt	C64 (normal)	Forth64	Nesting 1mill	6m20	
"	C64 (Turbo FPGA 6502)	Forth64	Nesting 1mill	25s	
"	C64 (normal)	Forth64	Fib2 (1000)	3m50s	
"	C64 (Turbo FPGA 6502)	Forth64	Fib2 (1000)	16s	
Martin Neitzel		FreeBSD 9 FICL Bootloader	Integer	0,00075s	

	Asus EeePC 1000h (Atom N270 1.6Ghz)				
Martin Neitzel	Asus EeePC 1000h (Atom N270 1.6Ghz)	FreeBSD 9 FICL Bootloader	Fib2	66s	
Martin Neitzel	Asus EeePC 1000h (Atom N270 1.6Ghz)	FreeBSD 9 FICL Bootloader	Nesting 1mil	0.66s	
Martin Neitzel	Asus EeePC 1000h (Atom N270 1.6Ghz)	FreeBSD 9 FICL Bootloader	Nesting 32mil	21s	
Martin Neitzel	Asus EeePC 1000h (Atom N270 1.6Ghz)	FreeBSD 9 FICL Bootloader	GCD2	0.57s	
Sabine "Atari Frosch" Engelhardt	Atari Portfolio	VolksForth 3.81	Fib2	35s	
Sabine "Atari Frosch" Engelhardt	Atari Portfolio	VolksForth 3.81	Prime	6s	
Sabine "Atari Frosch" Engelhardt	Atari Portfolio	VolksForth 3.81	Takeuchi	17s	
Herbert Lange	Compaq Deskpro P166	pForth V27	Integer Calc	0,052s	
Herbert Lange	Compaq Deskpro P166	pForth V27	Fib1	0,061s	
Herbert Lange	Compaq Deskpro P166	pForth V27	Fib2	0,001s	
Herbert Lange	Compaq Deskpro P166	pForth V27	Nesting 32mil	15,42s	
Herbert Lange	Compaq Deskpro P166	pForth V27	Memory Move	0,124s	
Herbert Lange	Compaq Deskpro P166	pForth V27	Prime	0,007s	
Herbert Lange	Compaq Deskpro P166	pForth V27	GCD1	0,002s	
Herbert Lange	Apple iMac G3 400Mhz	pForth V27	Integer Calc	0,013s	
Herbert Lange	Apple iMac G3 400Mhz	pForth V27	Fib1	0,015s	
Herbert Lange	Apple iMac G3 400Mhz	pForth V27	Fib2	0,001s	
Herbert Lange	Apple iMac G3 400Mhz	pForth V27	Nesting 32mil	4,335s	
Herbert Lange	Apple iMac G3 400Mhz	pForth V27	Memory Move	0,028s	
Herbert Lange	Apple iMac G3 400Mhz	pForth V27	Prime	0,017s	
Herbert Lange	Apple iMac G3 400Mhz	pForth V27	GCD1	0,063s	
Herbert Lange	DEC 3000 400s	pForth V27	Integer	0,123s	

Herbert Lange	DEC 3000 400s	pForth V27	Fib1	0,098s	
Herbert Lange	DEC 3000 400s	pForth V27	Fib21	0,001s	
Herbert Lange	DEC 3000 400s	pForth V27	Nesting 32mil	30,694s	
Herbert Lange	DEC 3000 400s	pForth V27	Memory Move	0,207s	
Herbert Lange	DEC 3000 400s	pForth V27	Prime	0,117s	
Herbert Lange	DEC 3000 400s	pForth V27	GCD1	0,483	
Herbert Lange	SUN Ultra 1 Creator 3D	pForth V27	Integer	0,049s	
Herbert Lange	SUN Ultra 1 Creator 3D	pForth V27	Fib1	0,052s	
Herbert Lange	SUN Ultra 1 Creator 3D	pForth V27	Fib2	0,001s	
Herbert Lange	SUN Ultra 1 Creator 3D	pForth V27	Nesting 32mil	15,631s	
Herbert Lange	SUN Ultra 1 Creator 3D	pForth V27	Memory Move	0,093s	
Herbert Lange	SUN Ultra 1 Creator 3D	pForth V27	Prime	0,060s	
Herbert Lange	SUN Ultra 1 Creator 3D	pForth V27	GCD1	0,022s	
Ralf Neumann	mc-CP/M Z80 4Mhz	FIG-Forth 1.1	Fib2	1m19s	
Ralf Neumann	Prof80 CP/M Z80 6Mhz	FIG-Forth 1.1	Fib2	53s	

Benchme Helper#

```
: benchme ( xt n -- ) \ executes the word with the execution token 'xt' n-times
  dup >r              \ save number of iterations
  0 do dup execute loop \ execute word. word must have a neutral stack effect
  cr r> . ." Iterations." cr \ emit message
;
```

Integer Calculations

```
32000 constant intMax

variable intResult

: DoInt
  1 dup intResult dup >r !
  begin
    dup intMax <
  while
    dup negate r@ +! 1+
    dup r@ +! 1+
    r@ @ over * r@ ! 1+
    r@ @ over / r@ ! 1+
  repeat
  r> drop drop
;
```

Fibonacci 1#

This version uses a recursive call. Recursive calls are not standardized in early Forth systems. The word to call the current definition can have different names in your forth (recurse, self, ...). Please check you system documentation (if available) or the wordlist (using WORDS or VLIST).

```
: fib1 ( n1 -- n2 )
    dup 2 < if drop 1 exit then
    dup  1- recursive
    swap 2- recursive  + ;

: fib1-bench 1000 0 do i fib1 drop loop ;
```

Fibonacci 2#

```
: fib2 ( n1 -- n2 )
   0 1 rot 0 do
      over + swap loop
   drop ;

: fib2-bench 1000 0 do i fib2 drop loop ;
```

Forth Nesting Benchmark#

```
 \ Forth nesting (NEXT) Benchmark                    cas20101204
 : bottom ;
 : 1st bottom bottom ;  : 2nd 1st 1st ;      : 3rd 2nd 2nd ;
 : 4th 3rd 3rd ;        : 5th 4th 4th ;      : 6th 5th 5th ;
 : 7th 6th 6th ;        : 8th 7th 7th ;      : 9th 8th 8th ;
 : 10th 9th 9th ;       : 11th 10th 10th ;   : 12th 11th 11th ;
 : 13th 12th 12th ;     : 14th 13th 13th ;   : 15th 14th 14th ;
 : 16th 15th 15th ;     : 17th 16th 16th ;   : 18th 17th 17th ;
 : 19th 18th 18th ;     : 20th 19th 19th ;   : 21th 20th 20th ;
 : 22th 21th 21th ;     : 23th 22th 22th ;   : 24th 23th 23th ;
 : 25th 24th 24th ;

 : 32million   CR ." 32 million nest/unnest operations" 25th ;
 :  1million   CR ."  1 million nest/unnest operations" 20th ;

 CR .( enter 1million or 32million )
```

Forth Memory Move Benchmark#

```
\ Forth Memory Move Benchmark                       cas 20101204
  8192 CONSTANT bufsize
 VARIABLE buf1 HERE bufsize 1+ allot BUF1 !
 VARIABLE buf2 HERE bufsize 1+ allot BUF2 !

 : test-CMOVE 49 0 DO BUF1 @ BUF2 @ bufsize CMOVE LOOP ;

 : test-CMOVE> 49 0 DO BUF2 @ BUF1 @ bufsize CMOVE> LOOP ;

 : test-MOVE> 49 0 DO BUF1 @ BUF2 @ bufsize MOVE LOOP ;

 : test-<MOVE 49 0 DO BUF2 @ BUF1 @ bufsize MOVE LOOP ;
```

```
: move-bench test-CMOVE test-CMOVE> test-MOVE> test-<MOVE ;
```

count bits in byte#

"BOUNDS" can be defined with:

```
( Convert str len to range for DO-loop )
: bounds ( str len -- str+len str )
  over + swap ;
```

"OFF" can be defined with:

```
( stores zero into address )
: OFF ( addr -- )
  0 SWAP ! ;
```

```
 \ Forth Benchmark - count bits in byte              cas 20101204

 VARIABLE cnt

 HEX
 : countbits ( uu -- #bits )
   cnt off
   8 0 DO dup 01010101  and cnt +!
          2/
   LOOP drop
   0 cnt 4 bounds DO i C@ + LOOP ;

 DECIMAL

 : bench5
   8192 DO I countbits . LOOP ;
```

Sieve Benchmark#

FIG-Forth derived systems or Forth-79 Systems require the initial value of a variable on the stack.

So instead of "VARIABLE FLAGS 0 FLAGS !" use "0 VARIABLE FLAGS".

```
\ Sieve Benchmark -- the classic Forth benchmark    cas 20101204

 8192 CONSTANT SIZE
 VARIABLE FLAGS
 0 FLAGS !
 SIZE ALLOT

 : DO-PRIME
   FLAGS SIZE 1 FILL  ( set array )
   0 ( 0 COUNT ) SIZE 0
   DO FLAGS I + C@
     IF I DUP + 3 + DUP I +
        BEGIN DUP SIZE <
        WHILE 0   OVER FLAGS +  C!  OVER +  REPEAT
        DROP DROP 1+
     THEN
 LOOP
 . ." Primes" CR ;
```

Greatest Common Divisor#

```
\ gcd - greatest common divisor                      cas 20101204

 : gcd ( a b -- gcd )
   OVER IF
     BEGIN
       DUP WHILE
          2DUP U> IF SWAP THEN OVER -
     REPEAT DROP ELSE
     DUP IF NIP ELSE 2DROP 1 THEN
   THEN ;

: gcd1-bench 100 0 DO
      100 0 DO j i gcd drop loop
      loop ;
```

"D0=" compares a double integer against zero. It can be defined as:

```
: D0= ( d d -- f )
  + 0= ;

\ another gcd O(2) runtime speed                     cas 20101204

 : gcd2 ( a b -- gcd )
   2DUP        D0= IF  2DROP 1 EXIT   THEN
   DUP          0= IF   DROP   EXIT   THEN
   SWAP DUP     0= IF   DROP   EXIT   THEN
   BEGIN  2DUP -
   WHILE  2DUP < IF OVER -
                 ELSE SWAP OVER - SWAP
                 THEN
   REPEAT NIP ;

: gcd2-bench 100 0 DO
      100 0 DO j i gcd2 drop loop
      loop ;
```

Takeuchi#

```
( takeuchi benchmark in volksForth Forth-83 )
( see http://en.wikipedia.org/wiki/Tak_(function) )

DECIMAL

 : 3dup 2 pick 2 pick 2 pick ;

 : tak ( x y z -- t )
   over 3 pick < NEGATE IF nip nip exit then
   3dup rot 1- -rot recursive >r
   3dup swap 1- -rot swap recursive >r
             1- -rot recursive
   r> swap r> -rot recursive ;

 : takbench ( -- )
   0 1000 0 DO DROP 18 12 6 tak LOOP ;
```

simple 6502 emulator#

```
\ A simple 6502 emulattion benchmark                          cas
\ only 11 opcodes are implemented. The memory layout is:
\  2kB RAM at 0000-07FF, mirrored throughout 0800-7FFF
\ 16kB ROM at 8000-BFFF, mirrored at C000
decimal
create ram 2048 allot   : >ram $7FF  and ram + ;
create rom 16384 allot  : >rom $3FFF and rom + ;
\ 6502 registers
variable reg-a   variable reg-x  variable reg-y
variable reg-s   variable reg-pc  : reg-pc+ reg-pc +! ;
\ 6502 flags
variable flag-c  variable flag-n   variable cycle
variable flag-z  variable flag-v  : cycle+ cycle +! ;
hex
: w@ dup c@ swap 1+ c@ 100 * or ;
: cs@ c@ dup 80 and if 100 - then ;

: read-byte ( address -- )
  dup 8000 < if >ram c@ else >rom c@ then ;
: read-word ( address -- )
  dup 8000 < if >ram w@ else >rom w@ then ;
: dojmp ( JMP aaaa )
  reg-pc @ >rom w@ reg-pc ! 3 cycle+ ;
: dolda ( LDA aa )
  reg-pc @ >rom c@ ram + c@ dup dup reg-a !
  flag-z ! 80 and flag-n ! 1 reg-pc+ 3 cycle+ ;
: dosta ( STA aa )
  reg-a @ reg-pc @ >rom c@ ram + c! 1 reg-pc+ 3 cycle+ ;
: dobeq ( BEQ <aa )
  flag-z @ 0= if reg-pc @ >rom cs@ 1+ reg-pc+ else 1 reg-pc+ then 3 cycle+ ;
: doldai ( LDA #aa )
  reg-pc @ >rom c@ dup dup reg-a ! flag-z ! 80 and flag-n !
  1 reg-pc+ 2 cycle+ ;
: dodex ( DEX )
  reg-x @ 1- FF and dup dup reg-x ! flag-z ! 80 and flag-n !
  2 cycle+ ;
: dodey ( DEY )
  reg-y @ 1- ff and dup dup reg-y ! flag-z ! 80 and flag-n !
  2 cycle+ ;
: doinc ( INC aa )
  reg-pc @ >rom c@ ram + dup c@ 1+ FF and dup -rot swap c! dup
  flag-z ! 80 and flag-n !  1 reg-pc+ 3 cycle+ ;
: doldy ( LDY aa )
  reg-pc @ >rom c@ dup dup reg-y ! flag-z ! 80 and flag-n !
  1 reg-pc+ 2 cycle+ ;
: doldx ( LDX #aa )
  reg-pc @ >rom c@ dup dup reg-x ! flag-z ! 80 and flag-n !
  1 reg-pc+ 2 cycle+ ;
: dobne ( BNE <aa )
  flag-z @ if reg-pc @ >rom cs@ 1+ reg-pc+ else 1 reg-pc+ then
  3 cycle+ ;
: 6502emu ( cycles -- )
  begin cycle @ over  < while
    reg-pc @ >rom c@ 1 reg-pc+
    dup 4C = if dojmp then      dup A5 = if dolda then
    dup 85 = if dosta then      dup F0 = if dobeq then
    dup D0 = if dobne then      dup A9 = if doldai then
```

```
    dup CA = if dodex then      dup 88 = if dodey then
    dup E6 = if doinc then      dup A0 = if doldy then
        A2 = if doldx then      repeat drop ;

create testcode
  A9 c, 00 c,  \ start: LDA #0
  85 c, 08 c,  \        STA 08
  A2 c, 0A c,  \        LDX #10
  A0 c, 0A c,  \ loop1: LDY #10
  E6 c, 08 c,  \ loop2: INC 08
  88 c,        \        DEY
  D0 c, FB c,  \        BNE loop2
  CA c,        \        DEX
  D0 c, F6 c,  \        BNE loop1
  4C c, 00 c, 80 C, \   JMP start

: init-vm 13 0 do i testcode + c@ i rom + c! loop
          0 cycle ! 8000 reg-pc ! ;

: bench6502 100 0 do init-vm &6502 6502emu loop ;
```