{block 18}

**colorforth jul31 chuck moore public domain 24 load 26 load 28 load 30 load  
dump 32 load ;  
icons 34 load ;  
print 38 load ;  
file 44 load ;  
north 46 load ;  
colors 56 load ; mark empty**

{block 19}

{block 20}

{block 21}

{block 24}

**macro  
swap 168B 2, C28B0689 , ;  
0 ?dup C031 2, ;  
if 74 2, here ;  
-if 79 2, here ;  
a ?dup C28B 2, ;  
a! ?lit if BA 1, , ; then D08B 2, drop ;  
2\* E0D1 2, ;  
a, 2\* 2\* , ;  
@ ?lit if ?dup 58B 2, a, ; then 85048B 3, 0 , ;  
! ?lit if ?lit if 5C7 2, swap a, , ; then 589 2, a, drop ; then a! 950489 3, 0 , drop ;  
nip 4768D 3, ;  
+ ?lit if 5 1, , ; then 603 2, nip ;  
or 633  
binary ?lit if swap 2 + 1, , ; then 2, nip ;  
and 623 binary ;  
u+ ?lit if 681 2, , ; then 44601 3, drop ;  
? ?lit A9 1, , ;**

{block 25}

**pentium macros: 1, 2, 3, , compile 1-4 bytes  
drop lodsd, flags unchanged, why sp is in ESI  
over sp 4 + @  
swap sp xchg  
0 0 0 xor, macro 0 identical to number 0  
a 2 0 mov, never used?  
a! 0 2 mov, unoptimized  
@ EAX 4 \*, unoptimized  
! EDX 4 \*  
nop used to thwart look-back optimization  
- ones-complement  
2\*  
2/  
if jz, flags set, max 127 bytes, leave address  
-if jns, same  
then fix address - in kernel  
push EAX push  
pop EAX pop  
u+ add to 2nd number, literal or value  
? test bits, set flags, literal only!**

{block 26}

**macros  
over ?dup 4468B 3, ;  
push 50 1, drop ;  
pop ?dup 58 1, ;  
- D0F7 2, ;  
for push begin ;  
\*next swap  
next 75240CFF  
0next , here - + 1, 4C483 3, ;  
-next 79240CFF 0next ;  
i ?dup 24048B 3, ;  
\*end swap  
end EB 1, here - + 1, ;  
+! ?lit if ?lit if 581 2, swap a, , ; then 501 2, a, drop ; then a! 950401 3, 0 , drop ;  
nop 90 1, ;  
align here - 3 and drop if nop align ; then ;  
or! a! 950409 3, 0 , drop ;  
\* 6AF0F 3, nip ;  
\*/ C88B 2, drop F9F72EF7 , nip ;  
/mod swap 99 1, 16893EF7 , ;  
/ /mod nip ;  
mod /mod drop ;**

{block 27}

**for n push count onto return stack, falls into begin  
begin -a current code address - byte  
\*next aa-aa swap for and if addresses  
next a decrement count, jnz to for, pop return stack when done  
-next a same, jns - loop includes 0  
i -n copy loop index to data stack  
end a jmp to begin  
+! na add to memory, 2 literals optimized  
align next call to end on word boundary  
or! na inclusive-or to memory, unoptimized  
\* mm-p 32-bit product  
\*/ mnd-q 64-bit product, then quotient  
/mod nd-rq remainder and quotient  
/ nd-q quotient  
mod nd-r remainder  
time -n pentium cycle counter, calibrate to get actual clock rate**

{block 28}

**compiled macros  
2/ F8D1 2, ;  
time ?dup 310F 2, ; forth  
@ @ ;  
! ! ;  
+ + ;  
\*/ \*/ ;  
\* \* ;  
/ / ;  
2/ 2/ ;  
dup dup ; arithmetic  
negate - 1 + ;  
min less if drop ; then swap drop ;  
abs dup negate  
max less if swap then drop ;  
v+ vv-v push u+ pop + ;  
writes acn for write next drop drop ;  
reads acn for read next drop drop ;  
oadf qwerty  
save 0 dup nc @ writes stop ;**

{block 29}

**these macros may be white, others may not  
@ etc arithmetic  
negate n-n when you just cant use -  
min nn-n minimum  
abs n-u absolute value  
max nn-n maximum  
v+ vv-v add 2-vectors  
nc -a number of cylinders booted  
save write colorforth to bootable floppy  
oadf save as spelled by qwerty. for typing with blank screen**

{block 30}

**colors etc  
block 100 \* ;  
white FFFFFF color ;  
red FF0000 color ;  
green FF00 color ;  
blue FF color ;  
silver BFBFBF color ;  
black 0 color ;  
screen 0 dup at 1024 768 box ;  
5\* 5 for 2emit next ;  
cf 25 dup at red 1 3 C 3 A 5\* green 14 2 1 3 3E 5\* ;  
logo show black screen 800 710 blue box 600 50 at 1024 620 red box 200 100 at 700 500 green box text cf keyboard ;  
empty empt logo ;**

{block 31}

**block n-a block number to word address  
colors specified as rgb: 888  
screen fills screen with current color  
at xy set current screen position  
box xy lower-right of colored rectangle  
dump compiles memory display  
print compiles screen print  
icon compiles icon editor  
logo displays colorforth logo  
show background task executes following code repeatedly  
keyboard displays keypad and stack**

{block 32}

**dump x 200000 y 201200  
one dup @ h. space dup h. cr ;  
lines for one -1 + next drop ;  
dump x !  
r show black screen x @ 15 + 16 text lines keyboard ;  
it @ + @ dup h. space ;  
lines for white i x it i y it or drop if red then i . cr -next ;  
cmp show blue screen text 19 lines red x @ h. space y @ h. keyboard ;  
u 16  
+xy dup x +! y +! ;  
d -16 +xy ;  
ati F4100000 ff7fc000 or  
byte 4 / dump ;  
fix for 0 over ! 1 + next ; dump**

{block 33}

**does not say empty, compiles on top of application  
x -a current address  
one a-a line of display  
lines an  
dump a background task continually displays memory  
u increment address  
d decrement  
ati address of AGP graphic registers  
byte a byte address dump  
fix an-a test word**

{block 34}

**icons empty macro  
@w 8B66 3, ;  
!w a! 28966 3, drop ;  
\*byte C486 2, ; forth ic 0 cu 15F  
sq xy @ 10000 /mod 16 + swap 16 + box 17 0 +at ;  
loc ic @ 16 24 8 \*/ \* 12 block 4 \* + ;  
0/1 8000 ? if green sq ; then blue sq ;  
row dup @w \*byte 16 for 0/1 2\* next drop -17 16 \* 17 +at ;  
ikon loc 24 for row 2 + next drop ;  
adj 17 \* swap ;  
cursor cu @ 16 /mod adj adj over over at red 52 u+ 52 + box ;  
ok show black screen cursor 18 dup at ikon text ic @ . keyboard ; 36 load ok h**

{block 35}

**draw big-bits icon  
@w a-n fetch 16-bit word from byte address  
!w na store same  
\*byte n-n swap bytes  
ic -a current icon  
cu -a cursor  
sq draw small square  
xy -a current screen position, set by at  
loc -a location of current icons bit-map  
0/1 n-n color square depending on bit 15  
row a-a draw row of icon  
+at nn relative change to screen position  
ikon draw big-bits icon  
adj nn-nn magnify cursor position  
cursor draw red box for cursor  
ok background task to continually draw icon, icon number at top 4210752 4210752 4210752**

{block 36}

**edit  
+ic 1 ic +! ;  
-ic ic @ -1 + 0 max ic ! ;  
bit cu @ 2/ 2/ 2/ 2/ 2\* loc + 10000 cu @ F and 1 + for 2/ next \*byte ;  
toggle bit over @w or swap !w ;  
td toggle  
d 16  
wrap cu @ + 16 24 \* dup u+ /mod drop cu ! ;  
tu toggle  
u -16 wrap ;  
tr toggle  
r 1 wrap ;  
tl toggle  
l -1 wrap ;  
nul ;  
h pad nul nul accept nul tl tu td tr l u d r -ic nul nul +ic nul nul nul nul nul nul nul nul nul nul nul nul 2500 , 110160C dup , , 2B000023 , 0 , 0 , 0 ,**

{block 37}

**edit icon**

{block 38}

**PNG empty w 36 h 20 d 4  
frame 1E80000 ; file 42 load 40 load  
-crc a here over negate + crc . ;  
crc -crc ;  
wd -a here 3 and drop if 0 1, wd ; then here 2 2/s ;  
bys n-a . here swap , ;  
plte 45544C50 48 bys 0 3, FF0000 3, FF00 3, FFFF00 3, FF 3, FF00FF 3, FFFF 3, FFFFFF 3, 0 3, C00000 3, C000 3, C0C000 3, C0 3, C000C0 3, C0C0 3, C0C0C0 3, crc ;  
png awh d @ / h ! d @ / w ! wd swap 474E5089 , A1A0A0D , ihdr 52444849 13 bys w @ . h @ . 304 , 0 1, crc plte idat 54414449 0 bys swap deflate crc iend 444E4549 0 bys crc wd over negate + ;  
at 1024 \* + 2\* frame + ;  
full 4 d ! 0 dup at 1024 768 png ;  
pad 1 d ! 46 -9 + 22 \* nop 25 -4 + 30 \* at 9 22 \* nop 4 30 \* png ;**

{block 39}

{block 40}

**lz77 macro  
@w 8B66 3, ;  
\*byte C486 2, ;  
!b a! 289 2, drop ; forth  
\*bys dup 16 2/s \*byte swap FFFF and \*byte 10000 \* + ;  
. \*bys , ;  
+or over - and or ;  
0/1 10 ? if 1E and 1E or drop if 7 ; then F ; then 0 and ;  
4b dup 0/1 9 and over 6 2/s 0/1 A and +or swap 11 2/s 0/1 C and +or 8 or ;  
pix dup @w d @ 2\* u+ 4b ;  
row 1, dup w @ 2/ dup 1 + dup 2, - 2, 0 dup 1, +adl for pix 16 \* push pix pop or dup 1, +adl next drop +mod d @ 1024 2 \* \* + ;  
deflate 178 2, 1 0 adl! h @ -1 + for 0 row next 1 row drop ad2 @ \*byte 2, ad1 @ \*byte 2, here over 4 + negate + \*bys over -4 + !b ;**

{block 41}

{block 42}

**crc macro  
2/s ?lit E8C1 2, 1, ;  
1@ 8A 2, ; forth ad1 BDA2 ad2 BDD8  
array -a pop 2 2/s ;  
bit n-n 1 ? if 1 2/s EDB88320 or ; then 1 2/s ;  
fill nn for dup 8 for bit next , 1 + next drop ;  
table -a align array 0 256 fill  
crc an-n -1 swap for over 1@ over or FF and table + @ swap 8 2/s or 1 u+ next - nip ;  
+adl n FF and ad1 @ + dup ad2 @ +  
adl! ad2 ! ad1 ! ;  
+mod ad1 @ 65521 mod ad2 @ 65521 mod adl! ;**

{block 43}

{block 44}

**DOS file  
blks 256 \* ;  
w/c 18 blks ;  
buffer 604 block ;  
size -a buffer 0 1 reads buffer 98F + ;  
set n ! buffer 0 1 writes ;  
cyls n-nn 1 swap w/c -1 + + w/c / ;  
put an dup 2\* 2\* size set cyls writes stop ;  
get a size @ 3 + 2/ 2/ cyls reads stop ;  
.com 0 63 blocks put ;**

{block 45}

**blks n-n size in blocks to words  
w/c -n words per cylinder  
buffer -a 1 cylinder required for floppy dma  
size -a locate size of 2nd file. floppy has first FILLER then FILE allocated. FILLER is 2048 bytes, to fill out cylinder 0. names at most 8 letters, all caps. directory starts at buffer 980 +  
set n size. FILE must be larger than your file.  
cyls n-nn starting cylinder 1 and number of cylinders  
put an write file from address  
get a read file to address**

{block 46}

**north bridge empty macro  
4@ dup ED 1, ;  
4! EF 1, drop ; forth dev 3B00  
nb 0 dev ! ;  
sb 3800 dev ! ;  
agp 800 dev ! ;  
ess 6800 dev ! ;  
ric 7800 dev ! ;  
win 8000 dev ! ;  
ati 10000 dev ! ;  
add CF8 a! 4! CFC a! ;  
q 80000000 + add 4@ ;  
en 8004 q -4 and or 4! ;  
dv dup 800 \* q swap 1 + ;  
regs dev @ 19 4 \* + 20 for dup q h. space dup h. cr -4 + next drop ;  
devs 0 33 for dup q dup 1 + drop if dup h. space drop dup 8 + q dup h. space over h. cr then drop 800 + next drop ;  
ok show black screen text regs keyboard ;  
u 40 dev +! ;  
d -64 dev +! ;  
test FF00 + a! 4@ ; ok**

{block 47}

{block 48}

**ASCII macro  
1@ 8A 2, ; forth  
string pop ;  
cf-ii string 6F747200 , 696E6165 , 79636D73 , 7766676C , 62707664 , 71757868 , 336A7A6B , 37363534 , 2D313938 , 2F322E30 , 2B213A3B , 3F2C2A40 ,  
ch 7FFFFF0 and unpack cf-ii + 1@ FF and ;  
ii-cf string 2A00 , 0 , 2B2D0000 , 2725232E , 1B262224 , 1F1E1D1C , 28292120 , 2F000000 , 3A43355C , 3D3E3440 , 484A3744 , 3336393C , 38314742 , 3F414632 , 493B45 , 0 , A13052C , D0E0410 , 181A0714 , 306090C , 8011712 , F111602 , 190B15 ,  
chc 7FFFFE0 + ii-cf + 1@ FF and ;**

{block 49}

{block 50}

**clock macro  
p@ EC 1, ;  
p! EE 1, drop ; forth  
ca 70 a! p! 71 a! ;  
c@ ca 0 p@ ;  
c! ca p! ;  
hi 10 c@ 80 and drop if ; then hi ;  
lo 0 p@ 80 and drop if lo ; then ;  
bcd c@ 16 /mod 10 \* + ;  
hms0 4 bcd 100 \* 2 bcd + 100 \* 0 bcd + ;  
hms hms0 2 ms dup hms0 or drop if drop hms ; then ;  
ymd 9 bcd 100 \* 8 bcd + 100 \* 7 bcd + ;  
day 6 c@ -1 + ;  
cal hi lo time - hi lo time + 748 ;**

{block 51}

{block 52}

**LAN empty 3F8 54 load init  
no block 4 \* 1024 ;  
send no for dup 1@ xmit 1 + next drop ;  
receive no for rcv over 1! 1 + next drop ;  
no 18 7 18 \* ;  
backup no for dup send 1 + next drop ;  
accept no for dup receive 1 + next drop ;**

{block 53}

**4210752 4210752 4210752**

{block 54}

**serial 3f8 2e8 1050 macro  
p@ a! dup EC 1, ;  
p! a! EE 1, drop ;  
1@ 8A 2, ;  
1! a! 288 2, drop ; forth  
r 0 + + ;  
9600 12 ;  
115200 1 ;  
b/s 83 3 r p! 9600 0 r p! 0 1 r p! 3 3 r p! ;  
init b/s 16550 1 2 r p! 0 4 r p! ;  
xmit n 5 r p@ 20 and drop if 0 r p! ; then pause xmit ;  
cts 6 r p@ 30 and 30 or drop if cts ; then xmit ;  
st 6 r p@  
xbits 30 and 10 / dup 1 and 2\* 2\* + 2/ ;  
st! 4 r p! ;  
?rcv 5 r p@ 1 and drop if 0 r p@ then ;  
rcv ?rcv if ; then pause rcv ;**

{block 55}

**p@ p-n fetch byte from port  
p! np store byte to port  
1@ a-n fetch byte from byte address  
1! na store byte to byte address  
r n-p convert relative to absolute port address. base port on stack at compile time. compiled as literal at yellow-green transition  
9600  
115200 baud-rate divisors. these are names, not numbers  
b/s set baud rate. edit to change  
init initialize uart  
xmit n wait for ready and transmit byte  
cts n wait for clear-to-send then xmit  
st -n fetch status byte  
xbits n-n exchange status bits  
st! n store control byte  
?rcv fetch byte if ready. set flag to be tested by if  
rcv -n wait for ready and fetch byte**

{block 56}

**hexagon empty col 0 del 202020  
lin dup 2/ 2/ dup 2\* line ;  
hex xy @ 7 and over 2/ for lin 7 + next over for lin next swap 2/ for -7 + lin next drop ;  
+del del @ nop  
petal and col @ + F8F8F8 and color 100 hex ;  
-del del @ F8F8F8 or 80808 + ;  
rose 0 +del -176 -200 +at F80000 -del petal 352 -200 +at F80000 +del -264 -349 +at F800 -del petal 176 -200 +at F8 +del -176 98 +at F8 -del petal 176 -200 +at F800 +del ;  
ok show black screen 512 282 at rose text col @ h. space del @ FF and . keyboard ; 58 load ok h**

{block 57}

**draws 7 hexagons. colors differ along red, green and blue axes.  
col color of center hexagon  
del color difference  
lin n draws 1 horizontal line of a hexagon  
hex n draws top, center and bottom. slope 7 x to 4 y is 1.750 compared to 1.732  
+del n increment color  
-del n  
petal n draw colored hexagon  
rose draw 7 hexagons  
ok describe screen. center color at top**

{block 58}

**pan  
in del @ 2\* 404040 min del ! ;  
out del @ 2/ 80808 max del ! ;  
r F80000  
+del del @  
+col and col @ + F8F8F8 and col ! ;  
g F800 +del ;  
b F8 +del ;  
-r F80000 -del +col ;  
-g F800 -del +col ;  
-b F8 -del +col ;  
nul ;  
h pad nul nul accept nul -r -g -b nul r g b nul out nul nul in nul nul nul nul nul nul nul nul nul nul nul nul 250000 , 130D01 dup , , 2B000023 , 0 , 0 , 0 ,**

{block 59}

**in increment color difference  
out decrement it  
r  
g  
b increment center color  
-r  
-g  
-b decrement it  
+del redefine with ;  
+col change center color  
nul ignore  
h describe keypad**

{block 62}

**timing empty macro  
out E1E6 2, ; forth  
tare time - 1000 for next time + ;  
tare+ time - push 1000 for dup next c pop time + ;  
test tare time + - 1000 for out next time + ; next 3 loop 5.7 /next 2 /swap 25 swap 7.2 macro  
c! C88B 2, drop here ;  
loop 49 1, 75 1, e2 here - + 1, ; forth  
try time - 1000 c! loop time + ;**

{block 63}