Copying Bytes

M. Anton Ertl, TU Wien

Myths

- Copying bytes efficiently is simple
- `Cmove` is faster than `move`
- Implementing `cmove` efficiently is simple
- Implementing `move` efficiently is more complex

Cycles for 50-byte non-overlapping copy

Skylake		Zen 3			
sf	gforth	vfx32	vfx64		
95	36	34	24	232	move
100	87	32	21	27	cmove
83	90	33	21	224	cmove>

byte loop memmove() cell loop rep movsb

Words and C functions

Forth	C	
`move`	`memmove()`	to-range contains original from-range contents
`cmove`		propagates patterns if to $\in$ [from, from + u)
`cmove>`		propagates patterns if from $\in$ [to, to + u)
	`memcpy()`	undefined behaviour on overlap
`move<`		don't call if to $\in$ [from, from + u)
`move>`		don't call if from $\in$ [to, to + u)

Efficient implementations

```
: move ( from to u -- )
  over 3 pick - 2 pick u< if \ to in [from,from+u)
    move>
  else
    move<
  then ;

: cmove ( afrom ato u -- )
    dup 0= if exit then
    begin ( afrom1 ato1 u1 )
        over 3 pick - 2>r
        2dup 2r@ umin move<
        2r@ 1 rot within while
            2r> /string repeat
    2r> 2drop 2drop ;
```

Extend 2-byte pattern to 1000 bytes with `cmove`

Zen 3 cycles/cmove

VFX64		VFX32	
rep movsb		cell loop	
orig	new	orig	new
3360	965	4273	386

Conclusion

- ~~Moving bytes efficiently is simple~~
- `Cmove` is faster than `move`? Sometimes
- ~~Implementing `cmove` efficiently is simple~~
- ~~Implementing `move` efficiently is more complex~~