Disassembleren

het belang van automatismen

Experiment: fietsen met gekruiste armen

het belang van automatismen

Automatische akties, waarbij je dus niet hoeft na te denken, zijn super belangrijk in het dagelijks leven.

het belang van automatismen

Welke automatismen spelen een rol bij het lezen en begrijpen van forth code?

forth code ontwerp

```
12 5 * .
```

Wat zien we hier?

forth code ontwerp

Forth gebruikt een radicaal simpele notatie waarbij acties gebeuren in de volgorde waarin je ze typt: create `X 0 , 1 X !`

forth code ontwerp

Het ontwerp van Forth is tot op het bot vereenvoudigd:

- Witruimte
- Getallen
- Woorden

forth code ontwerp

uitvoering van acties is altijd van links naar rechts en van boven naar beneden

assembler code

Een voorbeeld:

MOV R5,R6

Wat doet de processor met deze instructie?

assembler code

of deze: ADD R2,R4

Wat doet de processor met deze instructie?

assembler code

Helaas, het kan van alles zijn. De data van R5 wordt gekopieerd naar R6. Of juist het omgekeerde!

R2 wordt bij R4 opgeteld. Of andersom.

assembler code

Maar: we maken zelf assemblers en disassemblers voor diverse soorten computers

assembler code

Bij een disassembler is het hoofddoel het begrijpen van de code.

We hoeven geen code in te typen dus we hoeven niet beknopt te zijn in wat we laten zien.

noforth assembler en disassembler zijn forth-friendly

De noforth assembler en disassembler hebben een aantal goed uitgedachte ergonomische eigenschappen

noforth assembler en disassembler zijn forth-friendly

forth volgorde: eerst operands, daarna de instructie

dus R1 R2 MOV in plaats van MOV R1,R2

noforth assembler en disassembler zijn forth-friendly

directe disassembly van forth woorden:

dus DAS <woordnaam> bv DAS dup of DAS +

noforth assembler en disassembler zijn forth-friendly

betere register mnemonics dan R1 ... R13

noforth assembler en disassembler zijn forth-friendly

ARM:	r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	sp	lr	pc
Forth:	ip	sp	w	tos	hop	day	sun	moon	vv	ww	xx	yy	zz	rp	lr	pc

Maar er is nog verdere verbetering mogelijk

In de rp2040 ROM staat op adres 0x3A dit:

```
003A  H 48A7  r0 pc 29C x) ldr  Lit: 40004000
adrs ch code  operands  instr   value
```

Maar er is nog verdere verbetering mogelijk

De forth friendly versie is dit:

```
003A  H 48A7  pc 29C + @ =40004000 to r0
adr  ch code  instructie
```

werkt vooral voor eenvoudige instructies

Er zijn heel specifieke instructies die je niet met deze aanpak kan verbeteren, bijvoorbeeld:

LDM, REVSH, WFI, SVC etc.

Die laten we zoals ze zijn, en kan je in de documentatie opzoeken...

Samenvatting

automatismen zijn fantastisch behulpzame helpers op allerlei terreinen

Samenvatting

forth code heeft een strakke van links naar rechts volgorde

Samenvatting

forth experts hebben automatismen geleerd die begrip van de code ondersteunen gebaseerd op die volgorde

Samenvatting

assembler codes hebben geen vaste volgorde

Samenvatting

we maken onze eigen assemblers/disassemblers en die zijn 'forth friendly'

Samenvatting

een groot deel van assembler instructies verplaatst data van A naar B

Samenvatting

disassemblers kunnen die verplaatsingen van A naar B laten zien in forth notatie

Samenvatting

dat helpt het begrip, verbetert informationele ergonomie en vermindert cognitieve belasting

Nog vragen?

? +

Demo