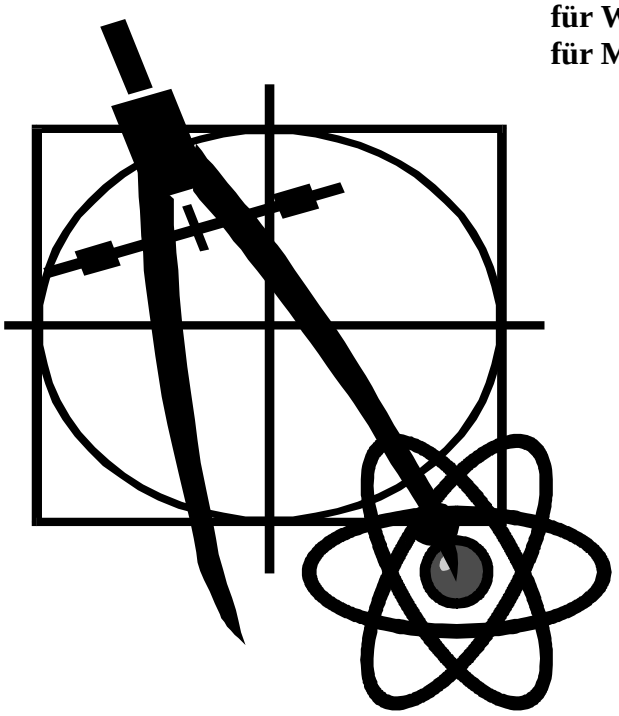
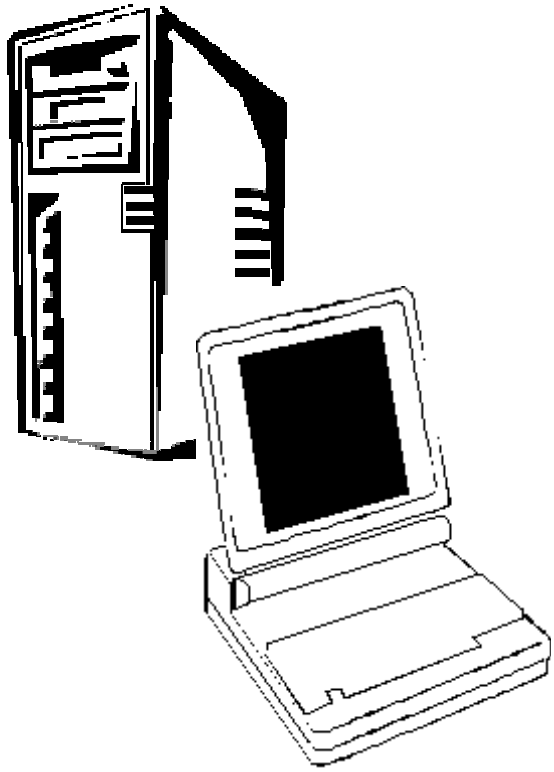
VIERTE DIMENSION

1 / 2003

Das FORTH-Magazin



für Wissenschaft und Technik, für kommerzielle EDV, für MSR-Technik, für den interessierten Hobbyisten.



In dieser Ausgabe:

19. Jahrgang - **Ausg. 1/2003** - 4,00 €

Organ der FORTH-Gesellschaft e.V.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

tematik GmbH
Technische Informatik

Feldstrasse 143
D-22880 Wedel
Fon 04103 – 808989 – 0
Fax 04103 – 808989 – 9
mail@tematik.de
www.tematik.de

Gegründet 1985 als Partnerinstitut der FH-Wedel beschäftigten wir uns in den letzten Jahren vorwiegend mit Industrieelektronik und Präzisionsmeßtechnik und bauen z.Z. eine eigene Produktpalette auf.

Know-How Schwerpunkte liegen in den Bereichen Industriewaagen SWA & SWW, Differential-Dosierwaagen, DMS-Messverstärker, 68000 und 68HC11 Prozessoren, Sigma-Delta A/D. Wir programmieren in Pascal, C und Forth auf SwiftX86k und seit kurzem mit Holon11 und MPE IRTC für Amtel AVR.

LEGO RCX-Verleih

Seit unserem Gewinn (VD 1/2001 S.30) verfügt unsere Schule über so ausreichend viele RCX Komponenten, dass ich meine privat eingebrachten Dinge nun Anderen, vorzugsweise Mitgliedern der Forthgesellschaft e.V., zur Verfügung stellen kann.

Angeboten wird: Ein komplettes LEGO-RCX-Set, so wie es für ca. 230,- € im Handel zu erwerben ist.

Inhalt:

1 RCX, 1 Sendeturm, 2 Motoren, 4 Sensoren und ca. 1.000 LEGO Steine.

Anfragen bitte an

Martin.Bitter@t-online.de

Letztlich enthält das Ganze auch nicht mehr als einen Mikrocontroller der Familie H8/300 von Hitachi, ein paar Treiber und etwas Peripherie. Zudem: dieses Teil ist ‘narrensicher‘ !

Dipl.-Ing. Arndt Klingelnberg

Tel.: ++32 +87 -63 09 89 (Fax: -63 09 88)
Waldring 23, B-4730 Hauset, Belgien
akg@aachen.kbbs.org

Computergestützte Meßtechnik und Qualitätskontrolle, Fuzzy, Datalogger, Elektroakustik (HiFi), MusiCassette HighSpeedDuplicating, Tonband, (engl.) Dokumentationen und Bedienungsanleitungen.

Forth Engineering
Dr. Wolf Wejgaard

Tel.: +41 41 377 3774 - Fax: +41 41 377 4774
Neuhöflirain 10
CH-6045 Meggen http://holonforth.com

Wir konzentrieren uns auf Forschung und Weiterentwicklung des Forth-Prinzips und offerieren HolonForth, ein interaktives Forth Cross-Entwicklungssystem mit ungewöhnlichen Eigenschaften. HolonForth ist erhältlich für 80x86, 68HC11 und 68300 Zielprozessoren.

KIMA Echtzeitsysteme GmbH

Tel.: 02461/690-380
Fax: 02461/690-387 oder -100
Karl-Heinz-Beckurtz-Str. 13
52428 Jülich

Automatisierungstechnik: Fortgeschrittene Steuerungen für die Verfahrenstechnik, Schaltanlagenbau, Projektierung, Sensorik, Maschinenüberwachungen. Echtzeitrechnersysteme: für Werkzeug- und Sondermaschinen, Fuzzy Logic.

FORTecH Software
Entwicklungsbüro Dr.-Ing. Egmont Woitzel

Budapester Straße 80 a D-18057 Rostock
Tel.: (0381) 46 13 99 10 Fax: (0381) 4 58 34 88

PC-basierte Forth-Entwicklungswerkzeuge, comFORTH für Windows und eingebettete und verteilte Systeme. Softwareentwicklung für Windows und Mikrocontroller mit Forth, C/C++, Delphi und Basic. Entwicklung von Gerätetreibern und Kommunikationssoftware für Windows 3.1, Windows95 und WindowsNT. Beratung zu Software-/Systementwurf. Mehr als 15 Jahre Erfahrung.

Ingenieurbüro
Dipl.-Ing. Wolfgang Allinger

Tel.: (+Fax) 0+212-66811
Brander Weg 6
D-42699 Solingen

Entwicklung von µC, HW+SW, Embedded Controller, Echtzeitsysteme 1-60 Computer, Forth+Assembler PC / 8031 / 80C166 / RTX 2000 / Z80 ... für extreme Einsatzbedingungen in Walzwerken, KKW, Medizin, Verkehr / >20 Jahre Erfahrung.

Ingenieurbüro
Klaus Kohl

Tel.: 08233-30 524 Fax: - 9971
Postfach 1173
D-86404 Mering

FORTH-Software (volksFORTH, KKFORTH und viele PD-Versionen). FORTH-Hardware (z.B. Super8) und Literaturservice. Professionelle Entwicklung für Steuerungs- und Meßtechnik.

Inhalt

Diese Ausgabe der VD wird vier bis sechs Wochen nach dem Erscheinen der Druckausgabe im Internet auf der Web-Seite der Forthgesellschaft e.V. veröffentlicht.

http://www.forth-ev.de

Eine PDF-Version dieser Ausgabe wird ab dem Zeitpunkt der Veröffentlichung im Internet ebenfalls zur Verfügung stehen. Bitte wenden Sie sich hierzu über die oben angegebene Adresse an den Webmaster der Forthgesellschaft e.V., oder an die Redaktion der „Vierte Dimension“.

fep

In der nächsten Ausgabe finden Sie voraussichtlich:

- Beiträge zur Jahrestagung in Lambrecht (verschiedene Autoren)
- SWAP in der Schule (Martin Bitter)
- Debugging mit einem MSO (Mixed Signal Oszilliskop) (Klaus Zobawa)
- Auswahl der „richtigen“ Programmiersprache (Tim Danieluk)
- ...und was immer SIE uns schreiben...

Editorial

IMPRESSUM

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth-Gesellschaft e.V.
Postfach 16 12 04
D-18025 Rostock
Tel.: 0381-400 78 28
E-Mail:
SECRETARY@FORTH-EV.DE
DIREKTORIUM@FORTH-EV.DE

Bankverbindung: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208

Redaktion & Layout

Friederich Prinz
Hombergerstraße 335
47443 Moers
Tel.: 02841-58 3 98
E-Mail:
VD@FORTH-EV.DE
FRIEDERICH.PRINZ@T-ONLINE.DE

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluß

März, Juni, September, Dezember
jeweils in der dritten Woche

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

4,00 € + Porto u. Verpackung

Manuskripte und Rechte

Berücksichtigt werden alle eingesandten Manuskripte. Leserbriefe können ohne Rücksprache gekürzt wiedergegeben werden. Für die mit dem Namen des Verfassers gekennzeichneten Beiträge übernimmt die Redaktion lediglich die presserechtliche Verantwortung. Die in diesem Magazin veröffentlichten Beiträge sind urheberrechtlich geschützt. Übersetzung, Vervielfältigung, Nachdruck sowie Speicherung auf beliebigen Medien ist auszugsweise nur mit genauer Quellenangabe erlaubt. Die eingereichten Beiträge müssen frei von Ansprüchen Dritter sein. Veröffentlichte Programme gehen - soweit nichts anderes vermerkt ist - in die Public Domain über. Für Fehler im Text, in Schaltbildern, Aufbauskizzen u.ä., die zum Nichtfunktionieren oder eventuellem Schadhaftwerden von Bauelementen oder Geräten führen, kann keine Haftung übernommen werden. Sämtliche Veröffentlichungen erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes. Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Liebe Leser,

die Suche nach einem Thema für das Editorial ist jedes Mal beinahe genau so schwierig wie das Zusammenstellen der Texte für eine ganze Ausgabe der VD. Mich kostet das mindestens einen Abend. Ein bißchen persönlich soll‘s nämlich schon sein, was der Editor an dieser Stelle schreibt. Es soll ein wenig von dem wiedergeben, was uns alle gemeinsam in diesen Tagen umtreibt, was uns beschäftigt. Aber da wäre viel Unangenehmes darunter, wie die Sorge um den Frieden, die wir in diesen Tagen wohl alle teilen. Und die ist, so wie die Politik wohl im Allgemeinen, nur ein weiterer Beweis dafür, daß Rousseau eben doch geirrt hat. Bildung und Ausbildung, ganz gleich auf welchem Niveau, garantieren für gar nichts; es sei denn, für Spaß am Forth.

Aber diesen Spaß teilen wir auch in weniger guten Zeiten gerne mit den beiden neuen Mitgliedern der Forthgesellschaft. Ich begrüße darum im Namen aller unserer Offiziellen die Freunde im Forth

Gebhard Herget aus Würzburg (geb. 1966)

Carsten Strotmann aus Bad Schönborn, (1969).

Ich würde mich sehr freuen, beide in Lambrecht zur Tagung der Forthgesellschaft persönlich begrüßen zu können. Das wäre sicher eine gute Gelegenheit, einem schon etwas älteren Forther zu erklären, was einen relativ jüngeren Menschen heute bewegt, Mitglied in unserem Verein zu werden. Meine Neugierde ist immens.

Geradezu gespannt bin ich auf die Reaktionen unserer Leser auf den Leserbrief von Gerard Baecker, auf der Seite 6. Da schreibt doch so ein Grünschnabel, daß er eigentlich alles mindestens genau so gut kann wie wir, die wir unsere elektronischen Handwerkskünste über viele Jahre hinweg lernen, üben und immer wieder verfeinern und neu an „Prüfsteinen messen“ mußten. Hat er Recht, der Herr Baecker? Na, Spaß macht er - aber lesen Sie selbst. Muß ich erwähnen, daß ich auch Herrn Baecker gerne in Lambrecht begrüßen würde?

Und weil ich den Bogen von der Sorge zum Spaß – an Forth – doch recht gut hinbekommen habe, möchte ich gleich noch hinzufügen, daß es mich besonders freut, daß auch diese Ausgabe unserer Zeitschrift wieder „proppevoll“ ist. Einige wirklich lesenswerte Zuschriften, unter anderem von längst „verloren geglaubten“ ehemaligen Mitglieder der FG und interessante Beiträge sowohl von Mitgliedern als auch wieder von einem „Externen“, werden Ihnen die Tage bis zur Tagung mit Kurzweil vertreiben.

Auf Wiedersehen in Lambrecht,

Ihr

Friederich Prinz

Haben Sie sich schon zur Tagung der Forthgesellschaft angemeldet? Anmeldungsformulare finden Sie in diesem Heft (Einlegeblätter) und/oder auf der Web-Seite der Forthgesellschaft:

Http://www.forth-ev.de

fep

Quelltext-Service

Die Quelltexte in der VD müssen Sie nicht abtippen. Sie können diese Texte auch direkt bei uns anfordern und sich zum Beispiel per E-Mail schicken lassen. Schreiben Sie dazu einfach eine E-Mail an die Redaktionsadresse. *fep*

Von: j.merkel@tbx.berlinet.de (Joachim Merkel)
Firma: BBS am Rande der Vernunft http://**www.chatnoir.de**

... als Nachtrag zu meiner Buchbesprechung von "Alan Turing - The enigma" in der VD 1/2000 einen Hinweis auf das sehr sehenswerte "Tony Sale's museum" mit Bildern und Beschreibungen der alten Dechiffriermaschinen und ihrer Bedienung aus der Zeit des 2. Weltkrieges, die das Knacken des Codes der sogenannten Marine-Enigma besorgten.
"http://www.codesandciphers.org.uk/bletchleypark/index.htm"

Salut, *Joachim*

...übrigens ist CHATNOIR.DE selbst eine sehenswerte Webadresse. Eine kleine Berliner Mailbox hat sich der modernen Technik und ihren Anforderungen angepaßt, ohne dabei ihre ursprüngliche Identität zu verlieren, oder auch nur zu verbiegen. „Dennoch bleibt die CN eine klassische Mailbox, deren User sich regelmäßig treffen. Für überraschende Entdeckungen hinter der Anonymität und zur Gewinnung unverzichtbarer Erkenntnisse, daß auf der anderen Seite auch ein Mensch liest und schreibt". Schauen Sie doch mal 'rein.

fep

Erinnern Sie sich noch an unseren Editor Rolf Kretzschmar? Rolf hat einige Jahre lang mit viel Fingerspitzengefühl – für die Vierte Dimension ebenso wie für deren Autoren – unsere Zeitschrift erarbeitet. Aus privaten und beruflichen Gründen mußte Rolf seine Tätigkeit als Editor aufgeben. Ein großer Anteil dieser beruflichen Gründe war sicher die Hinwendung zu seiner eigentlichen Berufung. Rolf Kretzschmar ist nämlich Bildhauer. Und was er in diesem Genre im schönsten Sinne des Wortes schafft, läßt sich zumindest teilweise im Netz bestaunen.

WWW.SCULPTUR.DE

ist eine weitere Adresse, die Sie unbedingt einmal besuchen sollten. Und wenn Sie von dort aus Kontakt mit unserem ehemaligen Editor aufnehmen, dann wird diesen das ganz sicher sehr freuen.

fep

Betreff: Noch´n Stackprozessor in FPGA
Von: Rafael Deliano <Rafael_Deliano@t-online.de>

Am1601 von Lyle Johnson gedacht für AMSAT-Projekte mit IPS als OS & Programmiersprache in Satelliten.
Das Handbuch und einen Simulator findet man via Google mit "Am1601 IPS".
Das Teil scheint unwesentlich üppiger als Novix zu sein: zwei interne 16 Bit Stacks je 16 Worte tief und einige Register.
Es scheint aber derzeit keine echte Implementierung in "strahlungsfestem" oder sonstigem FPGA zu geben. Im Entwurf ist angeblich vorgesehen auch die internen Register dreifach redundant auszuführen. Die externe Speicherschnittstelle (64k Byte) arbeitet laut Handbuch mit Fehlerkorrektur.

MfG *JRD*

Betreff: embedded 8
Von: Rafael Deliano <Rafael_Deliano@t-online.de>

http://**www.embeddedFORTH.de**
Inhalt: Magnetkarten, Fingerabdrucksensor,
Ansteuerung von HP-Tintenstrahldrucker,
FIR aus CCD, Zyklische Codes: Golay,
Signaturanalyse, EPROM-Testadapter,
D/A-Wandler mit Widerständen.

MfG *JRD*

Muß man mehr zu Rafael Delianos Hinweis sagen? Vielleicht darf ich der Hoffnung Ausdruck geben, daß sich jemand findet, der auch diese und folgende Ausgaben rezensiert. Jedenfalls ist auch die achte „Embedded" wieder eine technisch hochinteressante und gelungene Zeitschrift. Und diese Ausgabe steht, wie jetzt auch alle ihre Vorgängerinnen, unter der oben angegebenen Adresse zum Download bereit.

fep

Betreff: Minforth in Win32 console
Von: Andreas Kochenburger <kochenburger@gmx.de>

See http://www.minforth.net.ms

Have fun. *Andreas*

Hinter dieser kurzen Mitteilung in comp/lang(forth/de verbirgt sich das Angebot, das MinForth von Andreas Kochenburger herunter zu laden. Minforth ist ein „minimalistisches aber vollständiges Forth-System, geschrieben in C". MinForth steht für DOS, Windows und Linux zur Verfügung. Andreas Kochenburger stellt die C und Forth Quellen seines Systems zum Download zur Verfügung und benennt gleich passende C-Compiler (z.B. Turbo C 2.0 – Freeware) und deren Download-Adressen, mit denen ein erstes Forth compiliert werden kann. Leider reicht die Zeit nicht mehr, um dieser Ausgabe der VD einen ersten Einblick und Erfahrungsbericht beizugeben. Aber ich hoffe, daß unsere Leser sich auf MinForth stürzen und der Redaktion ihre Erfahrungen möglichst ausführlich berichten.

fep

Betreff: Re: www.forth-ev.de (war: Gruesse)
Von: Ulrich.Hoffmann@heidelberg.com

...wenn wir nun ab und an ein paar Kurzmeldungen auf der Titelseite einspielen, dann ist das ganze sogar auch noch belebt :-)
Dafür brauche ich einfach nur einen ISO8859-1-Text der Meldung.... Der Rahmen wird ja einfach generiert. Vielleicht hat der Ein oder Andere Interessantes, Forthiges zu vermelden? Ute, Bernd, Fred? **Vielleicht mal ein Aufruf in der Vierten Dimension?**
Anfang Februar werde ich mal an das Ablaufen der Rabatt-Frist für die Tagungsanmeldung erinnern. Vielleicht kann ich sie dann selbst einhalten. :-)

Viele Grüße, *Ulli*

Leserbriefe

Der E-Mail auf der vorherigen Seite war eine kurze Korrespondenz bezüglich der neuen Gestaltung der Webseiten der Forthgesellschaft vorausgegangen. Ulrich Hoffmann hat sich hierzu wirklich Mühe gegeben und eine ebenso ansprechende Seite gestaltet wie vor ihm Egmont Woitzel. Aber auch diese Arbeit lebt vom Mittun Anderer. Und darum bittet unser 'Webmaster' um Informationen jeglicher Art, die er als aktuelle Kurzmeldungen auf der Homepage der FG veröffentlichen kann.
Gleichzeitig soll daran erinnert werden, daß im April die Tagung der Forthgesellschaft stattfindet, daß NOCH Zeit ist, sich zur Teilnahme zu entschließen und daß Anmeldeformulare auch über die Webseite der FG herunterladbar sind:

www.forth-ev.de/tagung/Forthtagung-2003-Anmeldung.pdf

fep

Betreff: Leserbrief
Von: Rolf @ Sculptur.de <rolf@sculptur.de>

Lieber Fritz, liebe Forth-Freunde,

Michael Kalus möchte von mir wissen, was "für mich an Forth wichtig [war]" und wann das war....(ich fühlte mich angesprochen, als ich eher beiläufig (websurfend) feststellte, dass es die VD ja noch gibt). Nun denn, er soll nicht vergeblich gefragt haben.
Forth war für einen längeren Zeitraum Teil meines privaten und beruflichen Lebens. Von 1978 (FORTH auf AIM65) bis 1995 (Paradigmenwechsel ->Midlife-Krise mit Bewältigung) habe ich geFORTHelt. Als ich FORTH -nach Assembler, Fortran, Pascal und Basic- kennen lernte, war ganz klar, dass ich nur noch so programmieren wollte. Und seit dieser Zeit habe ich mich um keine andere Programiersprache mehr bemüht. Warum sollte ich. Als man mir in der Schule (Berufskolleg) durch die Blume zu verstehen gab, dass meine FORTH-Kurse nicht mehr state-of-art seien, hab ich eben nicht mehr Programmieren unterrichtet. Entweder FORTH oder NIX.
Soviel zum beruflichen Teil. Privat habe ich durch FORTH und die FORTH-Gesellschaft Freunde gewonnen und nette Bekanntschaften gemacht. Selten traf man "gewöhnliche" Leute im Dunstkreis von FORTH. Und ich fühlte mich immer wohler unter Menschen, die besondere Ausprägungen meiner eigenen Charaktereigenschaften zeigten: Chaoten, Träumer, Verspielte, Künstler, Improvisierer... Die Tatsache, dass immer noch in FORTH programmiert und über FORTH geschrieben wird, lässt mich hoffen, dass sich an dem Wurzelwerk nicht viel geändert hat.
Ob ich mit FORTH Spuren hinterlassen habe? Ich erinnere mich an kein Programm, von dem ich erwarten könnte, dass es heute noch irgend etwas leistet. Von meinen verkauften Sculpturen hoffe ich immerhin, dass sie noch und in Zukunft Freude bereiten.
Gerade repariert Klaus-Peter Schleisieck mein 1x1m-Lampenfeld, damit ich es als Lichtsculptur wiederverwenden kann. Und schwupps... bin ich wieder über FORTH gestolpert. Mal sehen, vielleicht wird es ja noch zu einem Werkzeug meiner Bildhauertätigkeit! Es ist ja nicht so, dass ich mich vom Computer hätte lösen können: meine Homepage (sculptur.de) ist mein Schaufenster und macht genug Arbeit, da ich von den Fotos bis zum Design alles selber mache; aber eben FORTH-Programmierung is´ nich´ mehr.
Ach ja, und dann ist da noch mein neues Taschenkalender-Windoofs-CE-Gerät (Toshiba E330). Hab ich mir zugelegt, weil es mich per Gong an wichtige Termine erinnern kann. Für dieses Ding hab ich doch wahrhaftig auch ein FORTH gefunden (dsFORTH/2). Während einer längeren Bus- oder Bahnfahrt könnte ich jetzt wieder FORTH üben, wenn ich nur wüsste, wie man dieses spezielle System bedient. Und da sind wir wieder beim typischen Problem: keine vernünftige Bedienungsanleitung oder Beschreibung.....
So, Schluss jetzt! Der Michael hat seine Antwort und Du, lieber Fritz, hast einen weiteren Leserbrief unterzubringen. Gefällt mir übrigens, die VD aus Moers!
Viele Grüße an die Redaktion sagt

Rolf Kretzschmar, Bildhauer.

Betreff: Leserbrief
From: Gerard Baecker <baecker@informatik.hu-berlin.de>

Liebe Forth-Gemeinschaft,

mit grossem Interesse habe ich die WWW-Ausgaben Eures Magazins gelesen.
< Es ist sehr schön, zu lesen, daß diese Ausgaben tatsächlich gelesen werden und die damit verbundene Arbeit nicht vergebens getan ist ! -fep >
Ich selbst bin kein Fortherianer und eigentlich schreibe ich nur, weil mir eine gewisse Tendenz nicht gefaellt. Bei einigen Eurer Artikel bekommt man den Eindruck, dass gerade die juengere Generation, die sich mit Rechnern beschaeftigt, keine Ahnung davon hat, wie man effektiv programmiert und stattdessen lieber Sprachen benutzt, die einen grossen Teil an Rechenleistung verschwendet.
Ich selbst bin Informatikstudent und moechte zur Verteidigung meiner Zunft einige Dinge sagen:
Natuerlich lernt man immer noch wie ein Prozessor funktioniert (Es gehoert zum Grundstudium zu verstehen, wie Busse, Speicher, Logische Bauelemente, Logische Baugruppen usw. arbeiten.). Man lernt auch mit minimalsten Rechen- und Speicherressourcen Algorithmen umzusetzen (Turingmaschinen, Assembler usw.)
Jeder Informatikstudent weiss, was UPN ist und wie Stapelprogrammierung funktioniert. Dass trotzdem wenig/kaum in Forth programmiert wird, liegt weniger an Ignoranz, sondern an dem was man nach dem Informatikstudium kennen muss. Ein Informatiker ist kein Ingenieur. D.h., man lernt beim Studium nicht eine oder mehrere spezielle Programmiersprachen, sondern -Paradigmen. Dazu gehoeren neben einfachen auf Kellerautomaten basierende auch die prozedurale, funktionale, objektorientierte und logische Programmierung. Es ist sogar so, dass beim Informatikstudium gar nicht so sehr wert

auf die praktische Programmierung gelegt wird (zum Programmieren muss man nicht studieren). Es geht auch darum faehig zu sein, zu entscheiden, welches Problem wie am besten und vor allem, ob es ueberhaupt rechnergestuetzt geloest werden kann.
Ich finde nicht, dass Forth DIE beste Programmiersprache ist, sie ist eine fuer eine bestimmte Art von Problemen eher geeignete als andere, mehr nicht.
Forth zu verstehen, kann sicherlich gut dazu beitragen, zu verstehen wie Rechner funktionieren (und selbst das gilt nur fuer die aktuelle verbreitete Form von Rechnern), aber ich denke, dass die zunehmende Komplexitaet von zu loesenden Problemen eine immer staerkere Abstraktion erfordert.
Meine ersten Programmiererfahrungen habe ich mit BASIC und Assembler auf einem kleinen 8-Bit-Computer gemacht. Spaeter lernte ich prozedurale Sprachen wie Pascal und C. Dann kam die funktionale wie in LISP, die objektorientierte wie in Java, Smalltalk usw. und die logische wie in Prolog oder Produktionssystemen. Inzwischen benutze ich eine Forth-aehnliche Sprache fuer meinen HP-Taschenrechner (die dort Reverse Polish Lisp genannt wird, meiner Meinung nach aber eher HP-Forth oder so heissen sollte), ich kaeme aber nie auf die Idee auf meinem PC eine solche Low-Level-Sprache einzusetzen.
Ja, ich verschwende die Ressourcen meines Rechners (und das gerade bei meinem Hang zu Sprachen wie Lisp, Python, Prolog sogar massiv), aber ich spare dabei meine wertvollste Ressource, Zeit. Wenn ich schnell mal eben einen Algorithmus ausprobieren moechte, dann mach ich mir kaum Gedanken darum, wie ich das Problem auf einem Microcontroller mit 128 Byte Speicher umsetzen wuerde (Das soll nicht heissen, dass ich das nicht koennte). Die Leistungsfaehigkeit eines Programms zeichnet sich auch nicht unbedingt dadurch aus, dass es besonders schnell oder klein ist. Als Leistungskriterium kann man auch die schnelle Lesbarkeit und Erfassbarkeit durch Dritte aufgefasst werden (demnach sind z.B. Pythonprogramme wesentlich leistungsfaehiger als Perlprogramme).
Ich wuerde mich also freuen, wenn die Forth-Gemeinschaft weniger dazu tendiert, das Desinteresse der Gruenschnaebel an Forth als Unfaehigkeit zu effektivem Programmieren zu missinterpretieren. Niemand behauptet, Forth waere eine tote Sprache (es gibt andere, die es wirklich verdient haetten und trotzdem erschreckenderweise extrem weit verbreitet sind, z.B. MS-VisualBasic). Forth ist eine winzige (leider zu wenig bekannte) aber trotzdem extrem reizvolle Facette des Computeruniversums. In diesem Sinne,

G. Baecker

P.S.
In alten Computerzeitschriften wurde oft nach der elegantesten Loesung (dem kuerzesten oder schnellsten Programm) zur Loesung eines Problems gesucht, an der sich dann verschiedene Leser versuchen konnten (so etwas wurde bis vor kurzem z.B. bei den sog. Mini-Challenges auf HPCalc.org gemacht). Vielleicht koenntet Ihr in Eurem Magazin sowas auch machen. Ist aber nur ein Vorschlag.

Lieber G. Baecker,
von solchen Leserbriefen träumt sicher Jeder, der versucht, immer wieder eine neue Ausgabe einer Zeitschrift mit interessanten Beiträgen voll zu kommen. Ich danke Ihnen sehr.
Und ich hoffe, daß Ihr Leserbrief zu einer regen und intensiven Diskussion in der FG führen wird – sowohl auf unserer anstehenden Tagung im April, als auch in der VD! Die Mitglieder unserer Gesellschaft sollen sich an dieser Stelle ausdrücklich aufgefordert sehen, den hingeworfenen Handschuh aufzunehmen.
Den Vorschlag, interessante Programmieraufgaben zu stellen, würde die VD nur zu gerne aufnehmen. Allein – es fehlt an Solchen, die uns diese Aufgaben geben, Lösungen vorbereiten und eingehende Lösungen prüfen. Ich bin ganz sicher, daß wir in der FG noch mehr als genug helle Köpfe haben, die uns mit entsprechenden Problemen fesseln können. Nur her damit... *fep*

Laptops zu verschenken? – Leserbrief, Martin Bitter

Im März dieses Jahres rief mich Fritz Prinz an. Ob ich Interesse an zwei 'alten' Laptops für 'die Schule' hätte, war seine Frage. Ich hatte!
< Natürlich meint Martin Bitter den März 2002. Daß sein Kurzbericht erst heute erscheint, hängt einzig damit zusammen, daß die VD in den letzten Ausgaben immer gut gefüllt war mit Beiträgen von besonderem Interesse. Aber gerade die hier vorgetragene Geschichte darf keinesfalls unveröffentlicht bleiben – fep >
Fritz teilte mir mit, dass sich telefonisch bei ihm jemand gemeldet hätte, der zwei Laptops abzugeben habe. Ich bekam eine Telefonnummer - Essener Vorwahl (Essen liegt nicht allzuweit von Mehrhoog entfernt)- und das war‘s.
Mit gemischten Gefühlen rief ich Herrn H. an. Immerhin besteht die Sammlung an Rechnern in 'meinem' Klassenraum inzwischen aus gut einsetzbaren Pentium PC (der langsamste hat 100 MHZ), aber es war mühsam aus den geschnorrten und geschenkten Rechnern diese Sammlung zusammenzubasteln. Manchmal möchte der 'edle' Spender sich die Mühe oder die Kosten des Verschrottens sparen und spendet halt lieber. Dennoch: Laptops fehlten 'uns' noch.
Schon am Telefon schien es so, als sei bei Herrn H. das Gegenteil der Fall. Wegen eines Umzuges gäbe er die Laptops her, sie seien gut erhalten und komplett.
Schnell war ein Termin für den nächsten Sonntag abgemacht. Der Besuch bei Herrn H. war eine Überraschung. Die Laptops waren nicht nur gut erhalten - sie sahen aus 'wie neu'. Elfenbeinweiß und sehr gediegen. Sogar die Originalverpackungen mitsamt allem Zubehör und Rechnungen waren vorhanden. Riesengroß die Kartons mit den Dockenstationen.
Herr H. berichtete mir, er habe sie einmal für ein Netzwerkprojekt angeschafft. Herr H. verdient sein Geld mit dem Programmieren. Angefangen hat er auf einem ATARI mit FORTH. Zur Zeit benutzt er POLYFORTH um in der Hauptsache DOS-Anwendungen zu schreiben. Er zieht um nach Heidelberg (Stand März 2002), um sich dort als selbständiger

Forthprogrammierer niederzulassen. Er ist nicht Mitglied der Gesellschaft, aber er kennt sie und schaut gelegentlich auf der Website vorbei. Dort fand er auch Fritzens Adresse.
Als Dreingabe zu den Laptops bekam ich noch einen Stapel interessanter Bücher. Insgesamt eine sehr umfangreiche Ladung – ich war froh, dass ich einen Kombi fahre.
Zu Hause angekommen ging das Staunen erst richtig los. Die 'alten' Laptos waren von 1995, verfügten aber über 486/DX CPUs bei 50 MHZ mit 4 MB Hauptspeicher und 250 MB Festplatte. Win3.1 und DOS waren installiert. Und, oh Wunder, alles lief problemlos. Ein Highlight sind die Dockingstationen: zusätzlicher Platz für Laufwerke und eine herausgeführte SCSI-Schnitstelle, Netzwerkanschluß usw.
In der Schule werden die Laptops nicht nur in 'meiner' Klasse gelegentlich eingesetzt. Sie ergänzen die unbeweglichen Hauptrechner. Haupteinsatzgebiet ist das Erstellen von Texten. Und sie sehen einfach toll aus --> die Schüler sind stolz darauf.
Bei Compaq gibt es sogar noch Unterstützung für diese Laptops. Ich könnte aufrüsten auf Win95 und 24 MB Hauptspeicher – aber 'never change a running system!' Und ihren Zweck erfüllen sie, so wie sie sind, gut.
Leider habe ich Herrn H.s neue Adresse nicht. Es mag sein, dass ihn auf Umwegen dieser Bericht erreicht und dass er dann erahnt, welche Freude er mir und einigen Schülern gemacht hat.
Ich wünsche ihm alles Gute und bedanke mich noch einmal herzlich.

Martin Bitter

Betreff: Bemerkungen zu QSORT mit LOCALS
Von: fsala@t-online.de (filippo sala)

Errata zu QSORT mit LOCALS in der VD 4/2002

1. Der Erfinder von Quicksort heisst C.A.R. Hoare.
2. In 2 Zeilen soll RECURSIVE und nicht RECURSE stehen.
3. Die Version mit LOCALS ist in der Druckmaschine steckengeblieben.

Bemerkung zu QSORT mit LOCALS

Die Version von Quicksort mit LOCALS sollte zeigen, wie die Forth-Worte hierdurch verständlicher werden. Aber jetzt habe ich festgestellt, dass bei LOCALS mit Recursion Vorsicht geboten ist. Der Test mit Win32Forth brachte den Returnstack ganz schön ins Schwitzen und schon bei 7000 Elementen gab es einige Abstürze. Die 1024 Bytes vom Returnstack waren verbraucht! Beim Test mit Gforth lief vorher alles richtig, weil erstens der Returnstack wesentlich grösser ist und zweitens die lokalen Parameter bei Gforth nicht auf dem Returnstack landen. Herzliche Grüsse

Filippo Sala

Qsort Listing, siehe Seite 9.

Grüße!
Nach einer sechswöchigen Pause war die SVFIG wieder "zurück-in-der-Schule". Ich glaube, ich habe schon zuvor erwähnt, daß das Cogswell College (www.cogswell.edu), eines der ältesten polytechnischen Colleges im Gebiet von San Francisco, einen angenehmen und bequemen Platz für unsere Treffen bietet. Es scheint mir, daß es eines der wenigen verbleibenden Beispiele für "Small is beautiful" ist, ungeachtet, natürlich, des reichlichen Parkplatzbereiches rund ums Gebäude. Es ist eine Tatsache, daß es heutzutage eine Ausnahme ist, einen freien Parkplatz auf unseren College- oder Universitätsgeländen zu finden - sogar an den Wochenenden.
Forthers sind nicht anders als 'normale' Leute, wenn es darum geht pünktlich zu erscheinen (auch wenn ein Parkplatz direkt vor der Tür ist). Und ziemlich oft sind sogar die geplanten Redner um 10 Uhr nicht hier; in diesem Fall springt Dr. Ting ein. Dieses Mal war Dr. Ting abwesend und demzufolge auch seine Kaffekanne. Aber Henrik Thurfjell stand bereit die Entwicklung eines Produktes zu beschreiben, welches er und John Peters den ganzen Weg, von der Rückseite einer Serviette bis zu einem vorzeigbaren Arbeitsmodell gebracht hatten. Es ist ein Werkzeug eines Elektrikers zur Verfolgung von Leitungen in Gebäuden. Die Eingangssignale einiger Sensoren, die den Strom in den Leitungen des Netzes erfassen, werden durch ein 8051-Board, welches AMR Forth spricht, verarbeitet. Sprachmeldungen als Ausgabe werden per Funk zum Techniker im Gebäude übertragen, während er durch das Gebäude geht und die Stromaufnahme in den unterschiedlichen Stromkreisen variiert, indem er Leuchten oder Geräte anschaltet oder Lampen in Steckdosen steckt. Damit kann eine Arbeit, die üblicherweise von zwei Personen ausgeführt wird, in kürzerer Zeit von einem allein erledigt werden.
Henrik's Vortrag regte eine umfangreichere Diskussion an und der Ideenaustausch mit den Zuhörern dauerte zwei Stunden bis zum Mittag.
Am Nachmittag beschrieb Tim Duncan, der Direktor der Abteilung für Digitale Audio Techniken am Cogsweel College, ein Kursprogramm, das er für das Curriculum vorgeschlagen hat. Dieses würde Forth gegenüber C und LISP unter den Sprachen, die von Musikern, die mit digitalem Audio arbeiten, bevorzugt werden, begünstigen. Ich glaube, daß wir Cogswell's Gastfreundschaft nur Dank Tim Duncan's Neigung zu Forth jetzt schon über sechs Jahre genießen dürfen.
Um den Tag abzuschließen, fuhren die meisten von uns gegen 3 p.m. zu einer geplanten Tour zum Computer History Museum, welches in der Nähe des bekannten Flugzeughangars im nahegelegenen Moffett Field/NASA Ames Research Park gelegen ist. Zuerst ein Wort über den Hangar (und ihr könnt mehr darüber im Web finden, wenn ihr nach Moffett+Field+Hangar sucht):
Er ist so alt wie ich, gebaut 1933 um ein gewaltiges Navy Luftschiff zu beherbergen. Und, 345 Meter lang, 94 m breit und 60 m hoch, ist er ein beeindruckendes Objekt, welches das Interesse von jedem weckt, der das erste mal zum San Francisco Airport fliegt.
Das Computer History Museum wurde hier 1996 eingerichtet und übernahm das meiste der Sammlung, die vorher in einem

Nachtrag zu F. Salas Beitrag in der VD 04 2002, siehe Seite 8.

```
\ --------------------------------- QSORT-Funktionsbeschreibung
\ v=Vergleichswert, p=adr1, q=adr2      Initialisierung
\ p -->                      <-- q      p+ q- wiederholen bis
\        p               q              p@ >=v  &  v >=q@
\        p               q              Inhalte vertauschen
\         p             q               p+ q-
\              q p                      bis q<p
\ wenn adr1 < q  Rekursion mit den Parametern  adr1 q
\ wenn p < adr2  Rekursion mit den Parametern  p adr2

\ ------------------------------------------- QSORT mit LOCALS
: (qsort) ( adr1 adr2 -- )
   0 0 0
   LOCALS| v p q adr2 adr1 |
   adr2 @ to v  adr1 to p  adr2 to q
   BEGIN
      BEGIN  p @ v u< WHILE  p 1 cells + to p  REPEAT
      BEGIN  v q @ u< WHILE  q 1 cells - to q  REPEAT
      q p u<
      IF
         true
      ELSE
         p @ ( n )  q @ p !  ( n ) q !
         q 1 cells - to q
         p 1 cells + to p
         false
      THEN
   UNTIL
   adr1 q < IF  adr1 q  recurse  THEN
   p adr2 < IF  p adr2  recurse  THEN ;

: qsort ( adr u -- ) 1- cells over + (qsort) ;

\ ------------------------------------ Test - Pseudozufallsbytes
DECIMAL

17 VALUE startwert

: random ( -- x2 )
  startwert 25173 *  13849 +  dup to startwert ;

: randombytes ( adr u -- )
   ( u ) 4/ 0
   ?DO
      ( adr ) random over !
      ( adr ) cell+
   LOOP
   drop ;

\ --------------------------------- Kontrolle mit 32-Bit-Zahlen
16 VALUE u        \ Anzahl der Elementen
u array zahlen[]  \ 32-Bit Datenfeld

: .daten ( -- )
   base @ >r decimal
   u 0
   DO
      i zahlen[] @
      cr 12 u.r
   LOOP
   r> base ! ;

: ntest ( -- )
   0 zahlen[]  u cells  randombytes
   cr .daten ( unsortiert )
   0 zahlen[] u qsort
   cr .daten ( sortiert ) ;
```

Museum in Boston untergebracht war. Momentan sind nur ca. 10% der über 3500 Ausstellungsstücke für die Besucher zugänglich, aber die Erweiterung zu einem größeren Gebäude ist in der Planung. Unser Curator stellte sich als alter Forther heraus, LaFarr Stuart (cf. Forth Dimensions, May/June 1980. p.2), was unseren Besuch besonders interessant und nostalgisch machte. Für Euch, meine Freunde, empfehle ich einen Besuch via http://www.computerhistory.org/exhibits/, und während ihr zu den Glanzlichtern kommt, könnt ihr euren Favoriten unter dem MITS Altair 8800, der Cray 1A, der Enigma der Wehrmacht, der Xerox PARC Alto von 1972, dem Apple-1, oder einer Anzahl anderer auswählen. Noch besser, besucht uns bei der SVFIG und wir nehmen Euch zum Museum mit.
Die abschließenden Neuigkeiten für heute, die ich auf dem Treffen hörte: Chuck Moore und seine Frau haben ihr Haus im Silicon Valley verkauft und sind nach Sierra City gezogen, über der Schneegrenze in den kalifornischen Vorbergen.
Frohes Wandern und Ski fahren, Chuck!
Mit den besten Grüßen,

Henry

Hallo Freunde,
dies wird ein kurzer Bericht, so wie auch unser Oktober-Treffen. Ich habe Euch schon zuvor gesagt, daß die SVFIG-Maschine ohne Dr. Ting und George Perry nicht auf allen Zylindern läuft - oder wie wir in Yankee-Slang sagen: "Es ließ einiges vermissen." Ting war verreist und George, der unter dem Wetter litt, bat darum, von der Leitung des Treffens befreit zu werden.
Sobald er erfuhr, daß Jay McKnight, am längsten in unserer Mitte, das Treffen am Laufen halten würde, verließ er uns.
(Now, how about all this California slang for our friendly translators? - *Mit der Übersetzung zufrieden, Henry? Ansonst brauche ich ein wenig Nachhilfeunterricht T.B....* :-)).
Wie auch immer, John Peters blieb bei einigen wenigen von uns den Morgen über und zeigte uns, wie er schnell mit all den Anhängern von Win32Forth kommunizieren konnte. Falls es da draußen noch irgendwelche Forther geben sollte, die noch nicht beim Polieren der Forth Fenster mitmachen, sie sollten auf alle Fälle mit John in Kontakt treten.
Die Nachmittagsgruppe wuchs bis auf fast 20 Personen, um Al Mitchell zuzuhören, der Windows zwei Jahre zuvor aufgegeben hat. Er war mit dem physischen Beweis der Forth Briefmarke, die er entwickelt, anwesend. Ich habe Euch darüber schon vorher geschrieben. Ich will hier nur an einige Stichworte erinnern:
PIC, C8051F017 und C8051F300 von Cygnal, Linux, GForth, GUI in TCL, amrBASIC, Fuzzy Logic. Al's Forth Briefmarke ist bis zu 500 mal schneller als Parallax's Basic Briefmarke. Al's eigenes AMRForth Rev. 6 wird in einem Monat auf seiner Webseite zu finden sein, und Ihr solltet **www.amresearch.com** besuchen, wenn Ihr mehr darüber wissen wollt, wie man BASIC auf Forth zum Laufen bringt.
Wir beendeten das Treffen zeitig, um zur Vintage Computer Messe auf dem nahegelegenen Moffet Field zu gehen. Dort waren vielleicht ein paar Dutzend Aussteller mit allen Arten von frühen Maschinen. Mein erster Blick fiel auf eine APL Tastatur und einen KIM-1, auf dem ein Schachprogramm lief. Ich traf Hans Franke nahe einem Schild, welches die Aufmerksamkeit auf **www.gfhr.de** (Gesellschaft fuer historische Rechenanlagen) lenkte, und er erzählte mir alles über die guten Computerteile, die man in der Schillerstraße in München finden kann. Ich muß Dr. Behringer bitten, mir darüber zu berichten.
Das war es für diesen Monat. Tschuess!

Henry

Hallo, Forth Freunde!
"Die zweite jährliche FIG Tagung war ein großer Erfolg mit 250 FORTH Nutzern, Händlern und Enthusiasten, die einen vollen Tag mit Sitzungen über FORTH und mit FORTH zusammenhängende Themen verbrachten. Das Villa Hotel in San Mateo, CA, führte das Treffen in diesem Jahr."
Ich zitiere aus dem Volume II No. 5 der Forth Dimensions, der Ausgabe vom Januar/Februar 1981. Zweiundzwanzig Jahre später gibt es das Villa Hotel immer noch, und die Tradition Forth-Tage im November zu Thanksgiving abzuhalten, wird immer noch von der Silicon Valley Forth Interest Group getragen. Und obwohl die Gruppe der Teilnehmer auf ein Zehntel geschrumpft ist, erfreut sie Charles Moore immer noch mit Gesprächen über seines Geistes Kind, und wenn ich so sagen darf, Geistes Enkel, wie z.B. ColorForth.
Ich bedauere, daß ich nur an der Morgensitzung des SVFIG Forth-Treffens am 16. November teilnehmen konnte und Dr. Ting's traditionelles Mittags-Barbequeue verpassen mußte, ebenso wie Chuck's Gespräche am Kamin zum Ende des Tages. Ich hörte Jeff Fox's Präsentation über seine und Sören Tiedemann's Forth GUIs und Betriebssysteme, die beide das meiste der gebräuchlichen Windows Eigenschaften mit nur 400 bis 600 Worten Forth-Code duplizieren. Ich zitiere Jeff: "Es ist schneller, deinen eigenen Code zu schreiben, als zu Windows zu gehen und zu versuchen, die benötigte Funktion zu finden. Wenn man alles in Forth selbst macht, hat man die Kontrolle über alle Abstraktionen. Benutze nicht mehr Abstraktionen als Du benötigst."
Ich nehme an, daß der Leser mehr Informationen über Jeff's Aha und Sören's Allegra Compiler auf ihren entsprechenden Webseiten finden kann.
Dr. Ting folgte Jeff mit einer Demonstration von F# (F-Sharp, wie in der Musik), welches in Chinesisch auf einer Windows XP Maschine lief. Die Plattform wurde von den Taiwanesen, mit denen Dr. Ting seit einiger Zeit zusammengearbeitet hat, ausgewählt, aber F# entstand aus seinem eigenen eForth, nachdem er Win32Forth ausprobiert und als zu kompliziert verworfen hatte. Das Projekt beinhaltet die Erzeugung chinesischer Schriftzeichen und die Entwicklung eines Forth, welches nicht englisch sprechende Chinesen, insbesondere junge Kinder, als Einführung in das Programmieren und in Forth selbst nutzen können. Mittels einer Redefinition mit "Aliases" hat Ting allen benötigten Forth-Worten chinesische Namen gegeben, und konstruieren können.
Ein großartiges Werkzeug für alle die, die durch mangelnde Kenntnis des Englischen behindert sind! Um Dr. Ting zu zi-

tieren: " Forth ist die beste Sprache für jede ausländische Zunge."
Ich wünschte, ich könnte mit einem Zitat aus dem Gespräch mit Chuck enden, aber da ich es verpaßt habe, laßt mich wiederholen, was in der Forth Dimension von 1981 geschrieben stand: "Nach einer Diskussionsrunde auf der FORML Konferenz in Asilomar, schloß Charles Moore von Forth, Inc. die Morgensitzung mit einer Erinnerung daran, daß es die extreme Flexibilität und Beweglichkeit von FORTH ist, die zunehmend mehr Probleme verursachen wird, um so mehr Leute damit in Berührung kommen." Ich hoffe, daß dies einige Kommentare unserer Leser hervorrufen wird.

Euer *Henry*

Lieber Friederich, Fred und Chris!
Ich habe wirklich einen kleinen Bericht über die SVFIG, aber das wird erst am Ende dieses Briefes sein, der nicht warten kann, bis er in Euren Magazinen gedruckt wird:
Ich möchte Euch allen eine gute Weihnachtszeit und ein frohes und gesundes Jahr 2003 wünschen. Bitte leitet meine Grüße an die anderen Forth'er weiter, die so nett waren, mit mir zu korrespondieren. Grüßt auch die, die an den Berichten 'über den großen Teich' arbeiten bzw. diese lesen, insbesondere Thomas Beierlein und Martin Bitter, da ich mir nicht über die aktuellen Email-Adressen aller Beteiligten sicher bin. Ich möchte an Fred mehr über seine Vorschläge für künftige Zeitschriftsartikel schreiben, aber das wird wahrscheinlich mehr Zeit benötigen, als ich heute habe.
< Die Grüße wurden, so hoffe ich, allesamt rechtzeitig übermittelt. Falls Jemand vergessen wurde, bitte ich um Entschuldigung – fep >

Das Dezember-Treffen der SVFIG überraschte mich dadurch, daß es sehr zeitig stattfand -- am zweiten Samstag des Monats. Ich schaffte es daher nur, die erste Hälfte des Treffens zu besuchen, zu dem ein kleineres Publikum als üblich Dr. Ting's Berichten über seine aktuellen Projekte in Taiwan zuhörte. Es stellte sich heraus, daß die taiwanesische Forth Interest Group ungefähr 20 Mitglieder hat und daß es dort ein Gebiet gibt, welches, etwa so wie das Silicon Valley vor zwanzig Jahren, fähigen Programmierern und Hardware Spezialisten mit Unmengen von Ideen und Enthusiasmus reichhaltig Gelegenheiten gibt. Ting arbeitet mit einer Firma, die sich selbst eForth Technlogy Inc. nennt und eine ihrer letzten Entwicklungen findet sich im Web, sowohl in chinesisch als auch in englisch - es ist der virtuelle Campus der Forth Academy. Wir konnten die Webseiten vom Computer aus dem benachbarten Raum aus besuchen und ich muß sagen, daß sie interessant sind. Der volle Satz an chinesischen Schriftzeichen ist nur unter WindowsXP verfügbar, allerdings zeigen andere Windows-Plattformen immer noch genug, um zu illustrieren, wie man in chinesisch Forth programmieren kann, ohne zuvor englisch zu lernen. Die Webadresse ist: "www.eforth.com.tw". Wenn man "/academy" hinzufügt und auf "kid's classroom" klickt, sieht man, was ich meine. Für die Heranwachsenden gibt es dort eine Menge von Informationen von Dr. Ting's Veröffentlichungen, Forth Lektionen, Manuals und eine Menge über Forth an einem Platz.
Das einzige Forth, daß man hier bisher nicht findet ist Win32Forth, welches nach Ting's Beschreibung "zu kompliziert" ist, und ich stimme dem mit meiner unqualifizierten Meinung zu. Das war auch der Grund, weshalb ich nicht zur Nachmittagssitzung blieb, zu der eine Anzahl Freiwilliger weiter an Win32Forth polieren wollten. "Chacun a son gout", wie sie in Frankreich sagen.

Mit besten Wuenschen, *Henry*

Im vergangenen Jahr sind mehrere Berichte von Henry 'aufgelaufen'. Zeitliche Überschneidungen zwischen dem Eintreffen in Moers und der bereits erfolgten Montage der nächsten Ausgabe der VD, der Wechsel im Editoriat und volle Ausgaben unserer Zeitschrift haben leider einige Briefe von Henry 'auf der Halde' gelassen. Das soll hiermit abgearbeitet sein.

fep

Viruswarnung ?!?

Kennen Sie das? Da „hängt sich das System auf" (Win98SE), ohne erkennbaren Grund. Mitten in der Arbeit – natürlich an der nächsten 'Vierte Dimension' – geht mit einem Mal gar nichts mehr. Selbst der Mauszeiger reagiert nur noch unwillig auf die Bewegung des Gerätes. Da muß alle angefangene Arbeit unerledigt bleiben. Der Fehler muß gefunden werden, und der erste Verdacht geht intuitiv in Richtung Virus.
Ich lasse auf meinem Arbeitsrechner das kleine Sharewaretool **Taskinfo 2002** in der Version 4,0,0,34 Beta immer im Hintergrund arbeiten (Igor M. Arsenin; **www.iarsn.com**). Taskinfo macht nur durch ein kleines Icon in der Systemleiste auf sich aufmerksam und stört den Betrieb des Rechners nicht. Bei Bedarf rückt Taskinfo aber mit einer Vielzahl an Informationen über den Rechner heraus. Unter anderem läßt sich damit auch auf Systemen mit Win9x – viel dezidierter als mit NTs Taskmanager – kontrollieren, welcher Prozeß gerade welche Ressourcen des Systems auffrißt.
Gleich der erste Versuch, mit Hilfe von Taskinfo dem Übel auf den Leib zu rücken, war erfolgreich. Da lief doch tatsächlich ein Prozeß mit einer **KC9XCR32.EXE**. So können doch nur Viren heißen, oder Trojaner und andere ungebetene Gäste. Den zugehörigen Prozeß zu stoppen, die EXE Datei auf der Platte zu suchen und zu löschen, zugehörige DLL ausfindig zu machen und ebenfalls zu löschen, war kein Problem. Allerdings lies sich nach einem Neustart des Systems der neue Drucker der Forthgesellschaft auch nicht mehr ansprechen. Mir schwante, daß der Virus bereits üblen Schaden angerichtet haben mußte. Mein Virenscanner konnte aber keine weiteren Spuren von ihm feststellen. Nach der Neuinstallation des Druckertreibers (aufwendige Geschichte) traf mich aber fast der Schlag. Der „Virus" war wieder da. KC9XCR32.EXE ist der Druckertreiber für den **KYOCERA mita FS 1900**. Peinlich, diese Intuition.
Die Systemhänger scheinen übrigens vom Memory zu kommen, aber da gehe ich GANZ LANGSAM heran ;-) *fep*

Threaded Code
Varianten und Optimierungen
(Kurzfassung)

M. Anton Ertl
Technische Universität Wien

Ziele des Gforth-Projekts:

Konkurrenzfähige Geschwindigkeit, Portabilität

Neue Herausforderungen:

- die Konkurrenz wurde schneller durch Compiler, die Maschinencode (statt threaded code) erzeugen.
- Direct threaded code für IA-64 (Itanium) macht Schwierigkeiten.

Unsere Antwort:

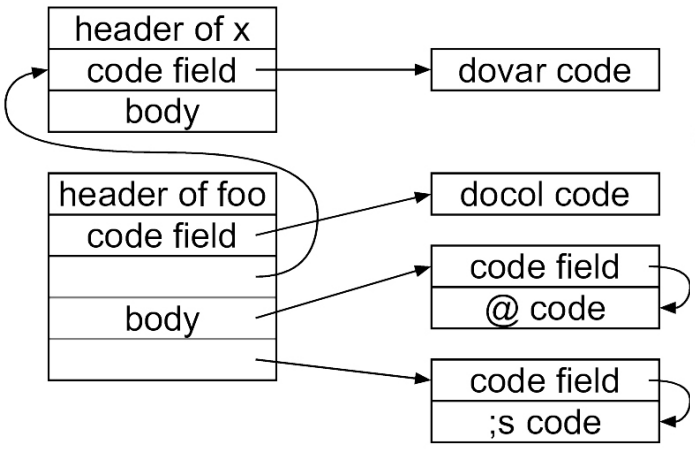
Wir bleiben bei threaded code (aus Portabilitätsgründen), und optimieren ihn.
U.a. kombinieren wir Sequenzen von Primitives in einen Superbefehl (superinstruction); z.B. die Sequenz "+ @" in den Superbefehl "+_@" [Schütz92].

Allerdings lassen sich in traditionellem threaded code nicht-Primitives (z.B. Variablen und :-Definitionen) nicht in Superbefehle einbauen. Im folgenden verfolgen wir den Weg vom klassischen indirect threaded code zu Superbefehlen, mit ein paar Abstechern zu auf dem Weg liegenden Varianten.

Wir betrachten folgendes Programm als laufendes Beispiel:

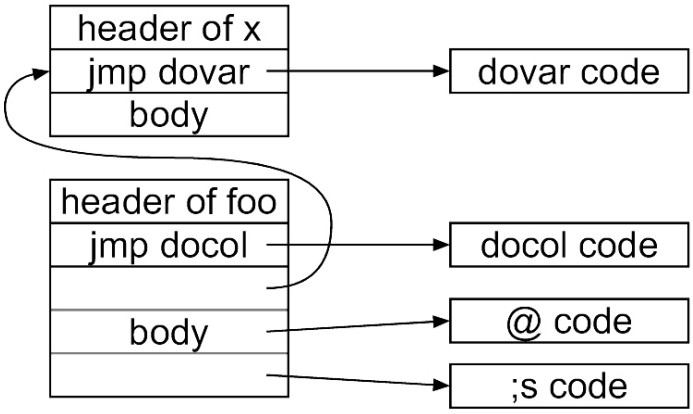
```
variable x
: foo x @ ;
```

Klassischer indirect threaded code:



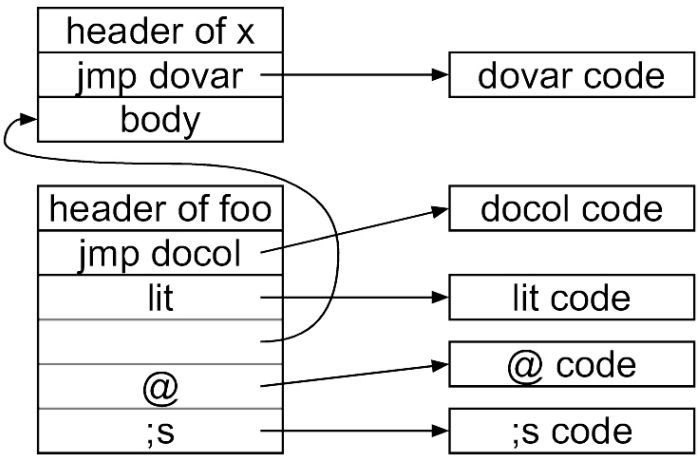
Hier zeigt eine threaded-code-Zelle (CFA) im Body einer **:**-Definition auf ein code field, und dort steht die Code-Adresse (CA), die auf den eigentlichen Code zeigt. Der Prozessor führt das nächste Wort aus, indem er die CFA aus dem Body lädt, dann die Code-Adresse aus dem Code-field, und schließlich springt er zu der Code-Adresse.
Warum speichert man nicht gleich die Code-Adresse im Body und erspart sich so die zusätzliche Indirektion? Damit man den Body von Nicht-Primitives wie der Variablen x finden kann.
Hier kann man die Variable x nicht in einen Superbefehl einbinden.

Direct threaded code (im klassischen Stil):



Hier zeigt eine threaded-Code-Zelle direkt auf Code. Um den Body eines Nicht-Primitives zu finden, zeigt die threaded-code-Zelle auf das code field; weil das code-field angesprungen wird, enthält einen Maschinencode-Sprung zur eigentlichen Code-Addresse. Es werden also Ladebefehle vor dem Ausführen aller Wörter durch Maschinencode-Sprünge bei Ausführung eines Nicht-Primitives ersetzt; da ca. 3/4 aller ausgeführten Wörter Primitives sind, zahlt sich das auf den meisten Prozessoren aus.
Auch bei dieser Variante kann man x nicht in einen Superbefehl einbinden.

Direct Threaded Code (primitive-basiert):



Holländisch ist gar nicht so schwer. Es ähnelt sehr den norddeutschen Sprachgepflogenheiten. Und außerdem ist Forth sowieso international. Neugierig ? Werden Sie Förderer der

HCC-Forth-gebruikersgroep.

Für 10 € pro Jahr schicken wir Ihnen 5 oder 6 Hefte unserer Vereinszeitschrift 'Het Vijgeblaadje' zu. Dort können Sie sich über die Aktivitäten unserer Mitglieder, über neue Hard- und Softwareprojekte, über Produkte zu günstigen Bezugspreisen, über Literatur aus unserer Forth-Bibliothek und vieles mehr aus erster Hand unterrichten. Auskünfte erteilt:

Willem Ouwerkerk
Boulevard Heuvelink 126
NL-6828 KW Arnhem
E-Mail: ***w.ouwerkerk@kader.hobby.nl***

Oder überweisen Sie einfach 10 € auf das Konto 525 35 72 der HCC-Forth-gebruikersgroep bei der Postbank Amsterdam. Noch einfacher ist es wahrscheinlich, sich deshalb direkt an unseren Vorsitzenden, Willem Ouwerkerk zu wenden.

Hier wird das Nicht-Primitive x übersetzt in das Primitive lit mit der Body-Adresse von x als immediate-Parameter.
In dieser Variante können wir schon Superbefehle einsetzen und z.B. die Sequenz "lit @ ;s" zu lit_@_;s kombinieren. Wir übersetzen Wörter in folgender Weise:

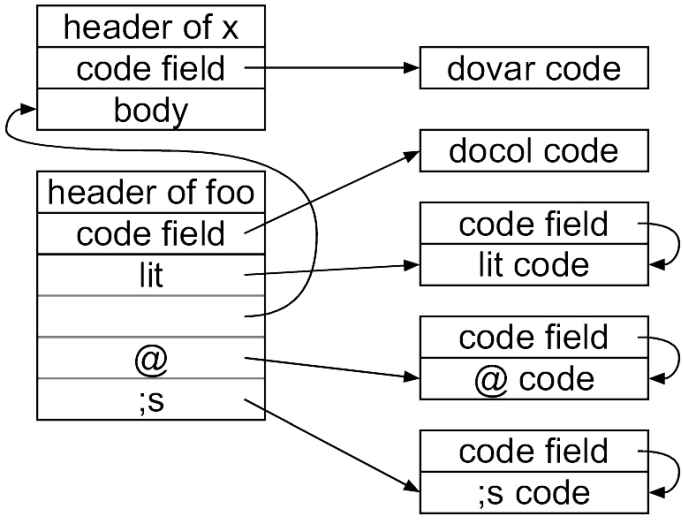
:-Definition	call	<body>
Konstante	lit	<value>
Value	lit	<body> @
Variable	lit	<body>
User-Variable	useraddr	<offset>
Defered	lit	<body> @ execute
Field	lit	<offset> +
Does-definiert	lit	<body> call <does-code>
;code-definiert	lit	<cfa> execute

Wir übersetzen teilweise in mehrere Primitives, weil sie später ohnehin zu Superbefehlen zusammengefasst werden.

Das Code field wird in dieser Variante immer noch benötigt, und zwar für EXECUTE; dem muss man ja eine Zelle übergeben, kein Gespann aus Primitive und Parameter.

Bei näherer Betrachtung fällt auf, dass das Code field nur für EXECUTE verwendet wird, und der threaded code im Body nur für die normale Ausführung (NEXT). Die beiden Arten der Ausführung sind also unabhängig voneinander und können unterschiedliche Threading-Methoden verwenden. Dabei erhält man hybride Methoden. Eine besonders nützliche Variante ist der

Hybride direct/indirect threaded code:



FIGUK
(Englische Forth-Gesellschaft)

Treten Sie unserer Forth-Gruppe bei.
Verschaffen Sie sich Zugang zu unserer umfangreichen Bibliothek.
Sichern Sie sich alle zwei Monate
ein Heft unserer Vereinszeitschrift.
(Auch ältere Hefte erhältlich)
Suchen Sie unsere Webseite auf:
www.users.zetnet.co.uk/aborigine/Forth.htm
Lassen Sie sich unser Neuzugangs-Gratis-Paket geben.
Der Mitgliedsbeitrag beträgt 12 engl. Pfund.
Hierfür bekommen Sie 6 Hefte unserer
Vereinszeitschrift Forthwrite.
Beschleunigte Zustellung (Air Mail)
ins Ausland kostet 20 Pfund.
Körperschaften zahlen 36 Pfund,
erhalten dafür aber viel Werbung.

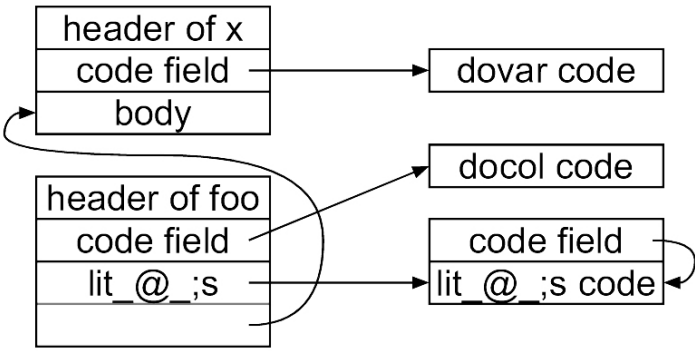
Wenden Sie sich an:

Dr. Douglas Neale
58 Woodland Way
Morden Surrey
SM4 4DS
Tel.: (44) 181-542-2747
E-Mail: dneale@w58wmorden.demon.co.uk

Hier verwendet NEXT direct threaded code und EXECUTE indirect threaded code. Ein Vorteil dieser Methode ist, dass kein Maschinencode-Sprung erzeugt werden muss (leichtere Portabilität).

Superinstructions:

Und jetzt können wir endlich Superbefehle verwenden:



Cache-Konsistenz:

Implementationen der 386-Architektur mit getrenntem Daten- und Befehls-Cache müssen dafür sorgen, dass die Inhalte dieser Caches konsistent bleiben. Das führt zu Geschwindigkeitseinbussen, wenn Daten und Code nahe beeinander liegen (je nach Prozessor ist dabei die Blockgröße 32-1024 bytes (zukünftig eventuell 4096 Bytes), und die Probleme tauchen bei Schreib- oder bei allen Datenzugriffen in der Nähe von ausgeführtem Code auf).

Durch diesen Effekt habe ich bei Gforth mit direct threaded code in klassischem Stil (Maschinencode-Sprünge im code field gleich neben Daten im Body) auf einem Athlon Verlangsamungen um bis zum Faktor 31 bei Mikrobenchmarks und bis zum Faktor 3 in Anwendungsbenchmarks gemessen.

Diese Probleme kann man normalerweise durch Trennung von Code und Daten vermeiden, allerdings ist das bei direct threaded code nicht möglich, denn da steht der Code bei den Daten, damit man die Daten findet.

Allerdings gibt es trotzdem Abhilfe: primitive-basiertes direct threading reduziert die Anzahl der Ausführungen von code fields (und die entsprechenden Verlangsamungen) von ca. 25% auf 1%-1.6% aller ausgeführten Wörter. Und hybrides direct/indirect-threading eliminiert auch noch den Rest.

Geschwindigkeit:

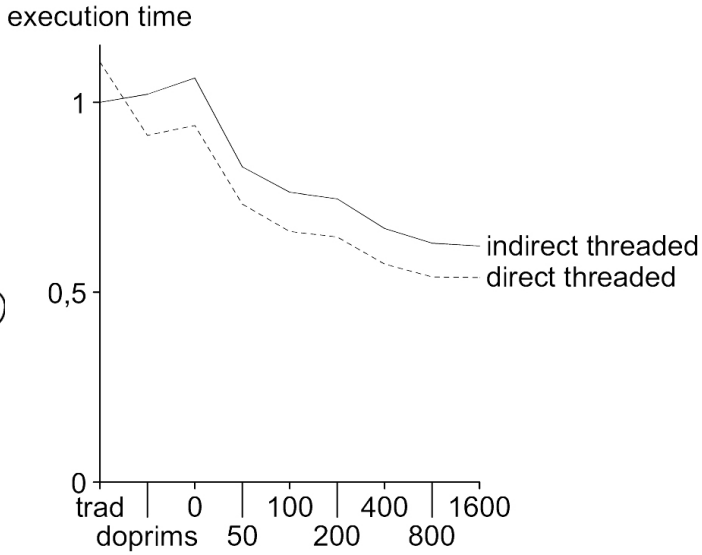
Ich vergleiche folgende Varianten (sowohl direct als auch indirect threaded):

trad: Klassischer threaded code

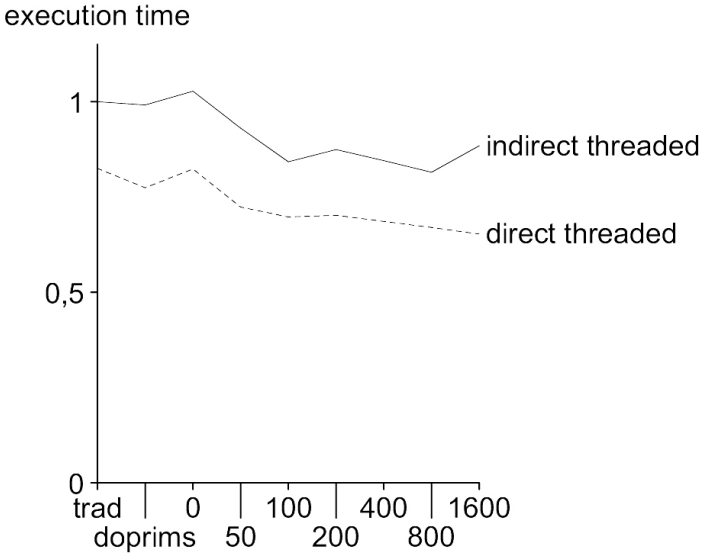
doprims: primitive-basiert, wobei jedes Nicht-Primitive auf ein Primitive übersetzt wird (nicht mehrere).

0: primitive-basiert mit Übersetzung von Nicht-Primitives in mehrere Primitives.

50-1600: wie "0", aber Sequenzen von primitives werden zu Superbefehlen zusammengefasst (aus 50-1600 vorgegebenen Superbefehlen).



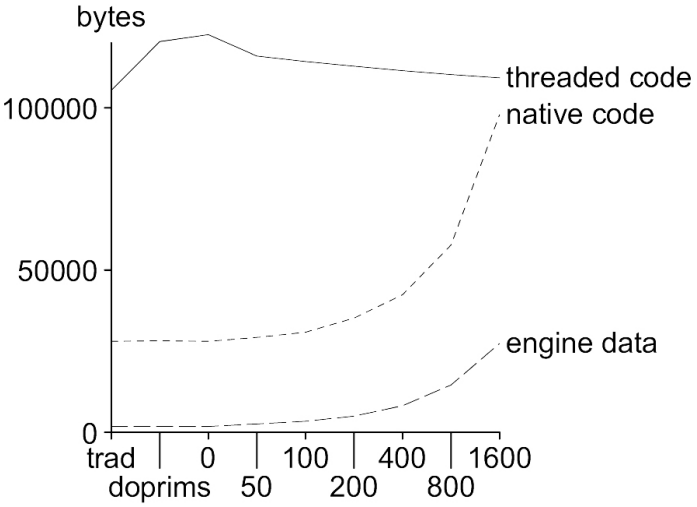
Hier die Zeiten für den Athlon (Benchmark: benchgc). Man sieht bei direct threaded und trad den Verlangsamungseffekt durch Cache-Konsistenz, der sich durch den Übergang zu primitive-basiertem threaded code stark verringert, sodass direct threaded code dann schneller ist als indirect threaded code. Abgesehen davon ist doprims ca. gleich schnell wie trad. "0" ist etwas langsamer, und durch das Einführen von Superbefehlen wird Gforth auf dem Athlon beträchtlich schneller (vor allem durch bessere Sprungvorhersage des branch target buffer (BTB)).



Hier sind die Zeiten für den Alpha-Prozessor 21164a mit dem gleichen Benchmark. Dieser Prozessor zeigt keine Verlangsamung durch Cache-Konsistenz-Hardware (für Cache-

Konsistenz ist auf der Alpha die Software zuständig). Die Beschleunigung durch Superbefehle ist auf dem 21164a geringer als für den Athlon, weil der 21164a keinen BTB hat.

Speicherverbrauch:



Primitive-basierter threaded code ist größer als klassischer threaded code; diese Expansion wird durch Superbefehle zum Teil wieder ausgeglichen. Allerdings kosten große Mengen an Superbefehlen wiederum Platz für zusätzlichen Maschinencode und Verwaltungsdaten.

Erfahrungen:

Primitive-basierter Code und Superbefehle sind seit Anfang 2001 in Gforth eingebaut, hybrider direct/indirect threaded code seit Anfang 2002.

Für den hybriden direct/indirect threaded code war eine Änderung im Image-Format nötig (und damit einiges an Arbeit).

Die Umstellung auf Primitive-basierten code machte (nichtstandard) Forth-Code wie den folgenden unbrauchbar:

create xt-table] word1 word2 [

Das Compilieren eines Wortes ist nun nicht mehr unbedingt äquivalent zu "' wort ,". Wir hatten ein oder zwei solche Stellen in unserem Code.

Umgekehrt kann man durch hybriden direct/indirect threaded code nicht mehr "," schreiben, wenn man "compile," meint. Wir hatten allerdings keine Probleme in der Richtung.

Schließlich ist es jetzt unmöglich, ein Wort wie REPLACE-WORD zu schreiben, das ein beliebiges Wort durch ein anderes ersetzt, indem es das erste Wort durch ein deferred word ersetzt (bei Konstanten, die zu literals auscompiliert werden, hätte es z.B. keine Wirkung).

Eine weitere, etwas überraschende Erfahrung ist, dass hybrid direct/indirect threaded code 3%-13% schneller ist als normaler (primitive-basierter) direct threaded code.

Zusammenfassung:

Übersetzung in Primitives erlaubt

- Superbefehle (Effizienz)
- Vermeidung von Cache-Konsistenz-Problemen auf 386 (Effizienz)
- Hybrider direct/indirect threaded code (Portabilität)

Nachteil: mehr Speicher für threaded code

Superbefehle:

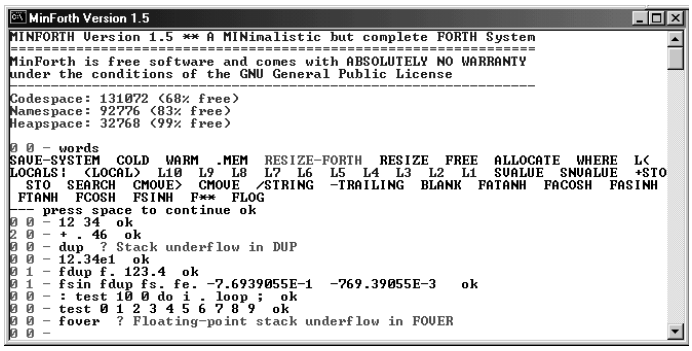
- schneller
- weniger Speicher für threaded code
- mehr Speicher für Primitives

Eine längere, englische Version dieses Artikels ist <**http://www.complang.tuwien.ac.at/papers/ertl02.ps.gz**>.

Literatur:

[Schütz92] Udo Schütz, Optimierung von Fadencode, Forth-Tagung 1992.

MinForth



MinForth baut auf eine sehr einfache, virtuelle Maschine auf, die nur aus wenigen Low-Level Worten besteht. Die virtuelle Maschine ist in einfachem C geschrieben. Alles Andere ist High-Level-Forth, de facto ANS X 3.125.
MinForth läuft auf PCs. Die Quellen können ohne jegliche Modifikation für DOS, Windows oder Unix kompiliert werden. Images von MinForth können unter beliebigen Betriebssystemen arbeiten.
MinForth ist „Crash-getestet". Alle Low-Level-Worte sind mit maximalen Laufzeit-Prüfungen definiert. Jede „gefährliche" Bedingung führt zu einer High-Level-Ausnahme.
Sind Sie neugierig geworden? Holen Sie sich MinForth – und berichten Sie über Ihre Erfahrungen in den nächsten Ausgaben der Vierte Dimension.

fep

Gehaltvolles

zusammengestellt und übertragen
von
Fred Behringer

VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

Nr. 31, April 2002

Im Protokoll einer der FG-Hauptversammlungen der letzten Jahre wurde festgehalten, man müsse die Beziehungen zu den holländischen Forth-Freunden verbessern. Das war ein Mißverständnis oder eine unglückliche Formulierung. Zu verbessern brauchten wir die Beziehungen nicht, sie waren und sind gut, lediglich zu intensivieren. Jeder, der auf der Tagung 2002 in Garmisch-Partenkirchen war, wird zugeben, daß uns das gelungen ist. Daß die Beziehungen gut waren und sind, wird auch vom Redakteur des Vijgeblaadjes, Albert Nijhof, bekräftigt, dessen Worte ich im folgenden in Übersetzung in voller Länge wiedergebe:

Albert Nijhof: Seit Jahren schon haben wir (E-Mail-)Kontakt mit der deutschen Forth-Gesellschaft. In ihrer Zeitschrift "Vierte Dimension" besprechen sie regelmäßig den Inhalt anderer Forth-Zeitschriften, worunter sich auch unser Vijgeblaadje befindet. In ihrer Zeitschrift erscheint (jedesmal) eine Werbeanzeige von uns. Als Gegenleistung steht diesmal wieder eine deutsche Werbeanzeige in unserem Vijgeblaadje. Bei uns geschieht das etwas spärlicher, da wir weniger Platz haben.

In Kürze werden die Kontakte noch stärker werden. Die Deutschen haben uns (Willem Ouwerkerk und Albert Nijhof) eingeladen, auf ihrer Jahresversammlung, diesmal Mitte April in Garmisch-Partenkirchen, etwas vorzutragen. Wir werden darüber berichten. Siehe http://www.forth-ev.de .

Van de redaktie
Albert Nijhof

Die "Funktionäre" der Forth-gebruikersgroep sind für eine bestimmte Zeit gewählt. Albert Nijhof, der Redakteur, bleibt bis 2003 im Amt, Willem Ouwerkerk, der Vorsitzende, bis 2004.

De belofte voor 2001, VI
Willem Ouwerkerk

Die verschiedenen Sensoren des Roboters Ushi müssen aufeinander abgestimmt werden. Dazu hat Willem ein paar Test- und Abstimmprogramme geschrieben.

Hier erklärt er RING.FRT. Er geht von drei Ringsensoren an der Unterseite des Roboters aus, mit deren Hilfe die Tischkante oder das Äußere eines Sumorings erkannt werden kann. Das Abstimmen geschieht über Leuchtdioden, die verlöschen, sobald ein kleines Stückchen schwarzer Untergrund auftaucht.

Ons wonderkind
Ernst Kouwe (Mitglied des Ushi-Teams)

Ein humorvoller Bericht mit vielen Andeutungen über Ushi, den Roboter, der zunächst im Baukasten schlummert, bevor er mit Hilfe eines Lötkolbens zur Welt gebracht wird. Genau wie ein Tamagotchi will er versorgt, umsorgt, gehegt und gepflegt werden. Prädestiniert ist er fürs Erlernen der Weltsprache Forth, aber das nachträgliche Erlernen anderer, weniger bedeutender Sprachen wird nicht ausgeschlossen.

Nr. 32, Juni 2002

Round Robin multitasking
Willem Ouwerkerk

Ein interessanter Artikel über das von F83 her bekannte cooperative Multitasking. Willem erklärt die Dinge anhand des im holländischen Roboter-Projekt eingesetzten AVR-ByteForth. Er gibt ein vollständiges Multitasking-System an. Um über gewollte oder ungewollte Ähnlichkeiten mit Bekanntem gar nicht erst räsonieren zu müssen (?), verwendet er für die Forth-Namen holländische Bezeichnungen (PAUZE, TAAK#, ACTIVEER, STACK-RUIMTE etc).

DOERS
Albert Nijhof

Ein ebenfalls sehr interessanter Artikel über die Wirkung und die Vor- und Nachteile von DOES>. Albert sagt: "DOES> zu erklären, fällt nicht leicht. DOES> ist eine künstliche Verflechtung (Optimierung?) verschiedener Aktionen, die gar nicht nötig ist und beim unerfahrenen Programmierer zwar Sicherheit gegen fehlerhaften Einsatz, aber Unsicherheit in der Erkennung der eigentlichen Möglichkeiten der Anwendung hervorruft. Für das Laufzeitverhalten bringt die vermeintliche Optimierung nichts. Frei verfügbare, mit Namen versehene DOER sind die Lösung." Es wird nicht klar herausgearbeitet, in welchem Forth die Erklärungsbeispiele geschrieben sind.

Nr. 33, August 2002

High level interrupts (in ByteForth)
Willem Ouwerkerk

Willem beschreibt die in die neue Version 2.00 von ByteForth eingebaute Interrupt-Organisation. Version 2.00 liegt zur Zeit

noch in einer Beta-Version vor. "Offiziell" läuft noch Version 1.70. Der AT89C2051 hat 5 Interrupt-Quellen, die alle vom IRQ-Typ sind. Es gibt auch ein Interrupt-Prioritätsregister. Forth-Worte zur Rettung des Stacks. Ein Timer-Beispiel.

Debouncing
Willem Ouwerkerk

Willem macht ganz kurz auf Möglichkeiten aufmerksam, Schalter zu entprellen. Zwei Forth-Worte als Beispiel.

Unsere holländischen Forth-Nachbarn

treffen sich regelmäßig in:

Katholieke Vrouwengilde Nederland
Bisonspoor 1204 te Maarssenbroek
Näheres unter: **http://www.forth.hccnet.nl/nieuws**

Nr. 34, Oktober 2002

ciforth
Albert van der Horst

Der Autor gibt eine erste, aber schon ausführliche Beschreibung seines Systems (ci = "common Intel" oder "computer intelligence"). Ein Forth, das sich selbst begreifen und modifizieren kann. ciforth ist indirekt gefädelt. Ein ganz kleiner Kern in Assembler, alle anderen Worte werden hinzugeladen. Geschwindigkeit war nicht das oberste Ziel. Nur für Intel-Prozessoren. Läuft aber unter Windows wie unter Linux. Für 16-Bit- und 32-Bit-Systeme. Automatische Dokumentation aus dem Quelltext heraus. Erzeugt "echte Programme", bei denen der Nur-Anwender nichts davon merkt, daß sie in Forth geschrieben sind. ciforth arbeitet gut mit den Programmen von Windows und Linux zusammen. Es hat eine Anzahl von Eigenschaften einer IDE (integrated development environment).

Nr. 35, Dezember 2002

De belofte, deel 7
Willem Ouwerkerk

Der siebente Teil des "Gelöbnisses", des Versprechens, des Vorhabens 2002. Es geht um das IR-RC-Protokoll für Ushi, das Roboter-Projekt. Das RC5-Protokoll ist zu langsam (9 Kommandos pro Sekunde). Aus dem TSOP1836 lässt sich mehr herausholen. RCU (Remote Control Ushi) schafft 200 Kommandos pro Sekunde. Für den RCU-Sender wird ein Forth-Programm angegeben.

ATS535A-bord; Introductiedag
Ben Koehorst

Auf der HCC-Veranstaltung hat Art van Wijk eine neue ATS535A-Platine vorgestellt. 60 Euro für die komplett aufgebaute Platine. ROM 10 Euro extra. Das 8052-ANS-Forth-Buch kann als PDF-Datei von www.forth.hccnet.nl heruntergeladen werden.

ciforth (slot)
Albert van der Horst

Schluss der Vorstellung von ciforth. ciforth besteht aus genau zwei Dateien, dem Programm und der Bibliothek. Programmierumgebung Windows oder Linux.

Bericht uit Duitsland
Albert Nijhof

Eine Kurzfassung (in deutscher Sprache) unserer Einladung zur Forth-Tagung 2003.

FORTHWRITE der FIG UK, Großbritannien

Nr. 116, April 2002

Die Innenseite des Hefteinbandes enthält (ganzseitig) die Anzeige unserer Forth-Gesellschaft zur Mitgliederwerbung.

1 Editorial
Chris Jakeman <cjakeman@bigfoot.com>

Keith Matthews, der langjährige Schatzmeister der FIG UK, ist nach kurzer Krankheit verstorben. Zwei neue Mitglieder, von Exeter und aus den USA, werden begrüßt. Die Zahl der Downloads von der FIGUK-Website steigt und beträgt jetzt etwa 1000 pro Monat.

2 Forth News

Processor-Core IGNITE von PTSC für Entwickler jetzt frei; 160 GB Download von Ultra Technology; Forth-Editor für Windows (ED4W) von http://www.getsoft.com; Mac OS X Forth; Win32Forth; PFE jetzt wahlweise indirekte Fädelung oder Unterprogrammaufrufe; Schwierigkeiten im ANS Technical Committee; Stringpakete .

4 Special Features of kForth
Krishna Myneni <krishnamyneni@compuserve.com> and David P. Wallace

Der erstgenannte Autor ist Mitglied der FIG UK, lebt aber in den USA. Dies ist der erste von zwei Teilen eines Artikels, der für die JFAR, das Journal of Forth Applications and Research, geschrieben wurde. Zusammenfassung der Autoren: Wir be-

sprechen zwei spezielle und ungewöhnliche Eigenschaften des kForth-Interpreters. (kForth ist weitestgehend ANS-Forth-kompatibel. Nicht vorhanden sind die üblichen Worte "HERE", "C," und "," , damit ein dynamischer Datenbereich (siehe Teil 2) eingerichtet werden kann.) Zunächst erklären wir, wie man mit einer rudimentären Form von Datentypisierung und Typenüberprüfung eine beträchtliche Anzahl von Forth-Programmierfehlern auffangen kann, ohne am üblichen Forth-Code viel ändern zu müssen. (Parallel zu Daten- und Returnstack werden Typenstacks mitgeführt - der Rezensent.) Sodann diskutieren wir den Nutzen eines dynamisch wachsenden Dictionarys und die Einschränkungen, die das mit sich bringt. Die beiden neuen Eigenschaften unseres Forth-Systems erfordern die Einführung zweier neuer Worte: A@ und ?ALLOT .

11 From the 'Net

Chris sammelt ein paar Stimmen von comp.lang.forth, "die wert sind, wiedergegeben zu werden". Mike Losh, der Begründer des internationalen Projektes "WebForth" (Forth als Java-Applet) würde das Thema heute ganz anders angehen. Vielleicht unter Linux, vielleicht als Daemon-Prozeß in Native-Forth oder C, der an einem TCP/IP-Port lauscht. "Als Java-Applet taugt Forth höchstens zu Lernzwecken." *(Ich, der Rezensent, dachte, es war sowieso nur als solches gedacht.)*

14 F11-UK
Jeremy Fowell <jeremy.fowell@btinternet.com>

Motorola-HC11-PygmyForth-Compiler unter DOS vom PC aus. Multitasking und Assembler. Alle Quelltexte. 47 engl. Pfund + Porto und Verpackung (4 Pfund für Deutschland) + 25 Dollar Registrierung für PygmyForth.

15 Flickwriter Project
Jenny Brien <webmaster@figuk.plus.com>

Jenny arbeitet ehrenamtlich mit körperlich Behinderten. Vor einiger Zeit kam die Idee auf, ein PC-Eingabegerät zu entwikkeln, bei welchem nie mehr als nur eine Taste, ein Knopf oder ein Hebel gleichzeitig gedrückt zu werden braucht, um eine bestimmte Reaktion auszulösen. (Man denke an Caps-Lock, mit dem im Einfingersystem in Ruhe beispielsweise H erzeugt werden kann.) Jenny berichtet über ihre Ideen und den Entwicklungsstand des von ihr initiierten Projektes.

17 euroFORTH 2001 - The Report
Howerd Oakford

Sechzehn Berichte über die gehaltenen Vorträge, in jeweils etwa fünf Zeilen gut kommentiert. Fünf Vorträge stammen von Mitgliedern der FIG UK, sechs Vorträge kommen aus dem deutschsprachigen Raum. Tagungsort war Schloß Dagstuhl/Germany. Konferenzsprache war Englisch.

21 Presenting the FIG UK Awards of 2001

Den diesjährigen Forth-Preis bekam Chris Hainsworth für 20 Jahre enthusiastischen Einsatz für die FIG UK, in den letzten Jahren als Vorsitzender. Den Forthwrite-Preis bekam Dave Pochin für seine zahlreichen Artikel (11 bisher), zuletzt über den Einsatz von Windows von Win32Forth aus. Der Rezensent darf auch im Namen der Forth-Gesellschaft den beiden Preisträgern gratulieren.

22 The FIG-UK Library
Graeme Dunbar

Die Forth-Bibliothek (Literaturverleih) der FIG UK hat ein neues Zuhause: Greame, Professor für Elektrotechnik, hat sie von Sylvia Hainsworth übernommen und konnte sie in den Räumlichkeiten der Robert Gordon University in Aberdeen unterbringen.

24 Seven Times Five Equals Eleven
Graham Telfer <gtelfer@po.synapse.ne.jp>

Graham beschreibt den von ihm durchgemachten Entwicklungsprozeß bei der Aufstellung einiger Routinen, die Modul-Arithmetik verwenden. Bei "Sieben mal fünf gleich elf" dachte ich, der Rezensent, eigentlich an die Zahlenbasis 34d. Nein, es geht um die Uhr: Einmal rum, zweimal rum ... , fünfmal rum, modulo 12. Graham meint, daß der Datenstack (als Behälter für das Anlegen lokaler Variablen) den Entwickler leicht von seiner eigentlichen Aufgabe ablenken kann. Er verwendet benannte (globale) Variablen.

32 Across the Big Teich
Henry Vinerts <volvovid@aol.com>

Zwei von Henrys Berichten in Originalfassung. Wir kennen sie in der Übersetzung von Thomas Beierlein. Im Vorspann steht: Dieses Material wurde von Henry Vinerts für die Vierte Dimension vorbereitet und erscheint hier mit Genehmigung der Forth-Gesellschaft (German FIG).

36 Vierte Dimension 4/01
Alan Wenham

Fast drei Seiten Rezension unserer VD mit dem Hinweis darauf, daß man über Alan Artikel oder/und Übersetzungen anfordern kann.

39 Did you Know? Forth Helps Nobel Prize Winners
Chris Jakeman

Wiedergabe einer Meldung von Elizabeth Rather über Forth und die Nobelpreisträger Penzias und Wilson.

40 Letters

Zwei Briefe: Wiedergabe eines im PCW Magazine veröffent-

lichten Briefes von Chris Jakeman an dessen Herausgeber über Forth und den Jupiter-Ace-Computer und ein Brief von Thierry Charlier de Chily aus Frankreich, der die F11-Platine der FIG UK erworben hat und über seine Erfahrungen berichtet.

Nr. 117, Juli 2002

1 Editorial
Chris Jakeman <cjakeman@bigfoot.com>

Chris Jakeman berichtet über die "joint chat session" am ersten Samstag im Mai dieses Jahres mit #Forth in irc.openprojects.net (40 Teilnehmer, darunter Chuck Moore) und #FIGUK (10 Teilnehmer). Außerdem begrüßt Chris drei neue Mitglieder (aus England, Holland und Frankreich).

2 Forth News

iForth 2.0; 4-Bit-Forth-MC MARC4 von Atmel; 4th von Hans Bezemer; Dokumentation für kForth 1.0.11; H8/300-Assembler für den RCX unter pbForth von Ralph Hempel; ISO bis 2007; ANS-Forth-Revision 2005; Schach-Programm von Jos Ven für Win32For 4.2; kForth-Beispiel-Sammlung; TCP/IP zur Verbindung von Mikroprozessoren; dynamisches Stringpaket von David Williams.

5 From the 'Net

Ein Interview (8 Seiten) mit Chuck Moore im Internet-Chat-Room unter Beteiligung mehrerer Fragesteller. Schlagworte, Paradigmen, Erfahrungen, gute Ratschläge, Argumente, bloße Meinungen - von jedem etwas, in bunter Reihenfolge: Howdy. Good crowd. Es gibt Mega-Forths, die alles erschlagen wollen. Einfachheit ist die Devise. Keep it simple. Zum Bücherschreiben fehlt mir die Geduld. FOR NEXT ist viel einfacher als DO LOOP. CATCH und THROW mag ich nicht. Fehler sollten gar nicht erst auftreten können. Oder sofort repariert werden. Es gibt Situationen, wo man DEFER nicht vermeiden kann. Worte sollten aber definiert sein, bevor sie verwendet werden. Stabile, zuverlässige Programme sind immer einfach. Der Datenstack dient der Speicherung namenloser Zwischenwerte. Bei der Erfindung von Namen läßt uns immer unsere Vorstellungskraft im Stich. Man denke an die endlosen Bindestrich-Zusammensetzungen in C. In meinem Chipentwurf sind die meisten Signale unbenannt. Anders als in VHDL. ... ich mache das alles ganz anders ... Sollte IF den Stack wirklich POPpen? Portierbarkeit ist nicht wichtig. Wirkliche Portierbarkeit ist überhaupt nicht möglich. Echte Anwendungen sind immer mit der Hardware verknüpft. Man ändere die Plattform und das gesamte Programm wird hinfällig. Quelltextfreier Code ist gut. Ich habe solchen bei OKAD jahrelang verwendet. FML (Forth Markup Language) ist wie ColorForth. Kompakt. Macht Java überflüssig. Ideen sind wie Meme (Verhaltensmuster, Gedankeneinheiten). Sie kommen, unvorhergesehen, und vergehen. Fortran, Algol, PLI, Pascal, C - alles dasselbe. Forth ist besser. Es ahmt die natürlichen Sprachen nach.

Ein Vermögen habe ich mit meinen Arbeiten nicht verdient, aber so leidlich leben konnte ich schon davon. Ich erwarte nicht, daß man mich akzeptiert. Ich forsche und probiere einfach weiter. Man gebe mir 100 Millionen Dollar und ich nehme es mit Bill Gates auf.

Ideen sollte man nicht patentieren lassen können. Je mehr Leute daran arbeiten, desto besser die Ergebnisse. Computer, die sich selbst programmieren? Solange ich lebe, wohl kaum. Forth auf einem Forth-Chip ist unschlagbar. Schade, daß die Computer dermaßen schnell geworden sind, daß C auch schon ausreicht. OKAD war selbstmodifizierend. Bis jetzt ist kein FML in Aussicht. Man bräuchte Leute, die die Idee aufgreifen und es entwickeln. Das lila Wort P (in ColorForth) könnte in FML anstelle des HTML-Tags <p> stehen, und so fort. Ich bin mit LISP und Prolog vertraut. Mit den anderen Sprachen nicht. Als ich Forth entwickelte, kannte ich sie alle. Seit Forth beachte ich sie nicht mehr. Man zeige mir eine Idee (beispielsweise Prolog) und ich baue sie in Forth ein. LISP hat viel zu Forth beigetragen. Die Idee, daß man etwas berechnen kann, ohne etwas abspeichern zu müssen. An den neuen Sprachen kann ich keine neuen Ideen erkennen. Für den Chip-Entwurf hatte ich keine spezielle Ausbildung, nur die Entschlossenheit, es zu tun. Ich würde mich selbst nicht als Vorbild empfehlen. Wie ich zu den 27 Primitives in einem optimierten Minimalsystem komme? Sitzfleisch, probieren und kombinieren. OK to all.

12 From the 'Net

Forth wird von Außenstehenden immer wieder angegriffen. Forth hat viele andere Sprach-Kandidaten, wie beispielsweise Modula 2, überlebt. Chris gibt Pro-Argumente wieder, von: Elizabeth Rather, Stephen Pelc, Anton Ertl, Neal Bridges.

14 F11-UK
Jeremy Fowell <jeremy.fowell@btinternet.com>

Das PygmyForth wird über die serielle Standard-Schnittstelle des PCs in den FLASH-Speicher (oder das RAM) des HC11 (8 MHz) geladen.

15 euroFORTH 2002

Diesmal in Wien, ein paar Schritte von der Innenstadt entfernt, 6.-8. September, bei Erscheinen der vorliegenden VD also schon Vergangenheit.

16 Expanding the Use of the Stack
Graham Telfer <gtelfer@po.synapse.ne.jp>

Der Autor ist den Lesern der Forthwrite nicht unbekannt. Es geht diesmal um Stackkommentare und um eine Nachrichten-

kette kürzlich in der Forth-Newsgroup. Er möchte neben den Stackkommentaren für vor und nach der Ausführung eines Wortes in dessen Definition auch die beteiligten formalen Parameter festhalten. In seinem Beispiel ist es die Konstante Pi, die vor Ausführung seines Beispielswortes nicht auf dem Stack liegt, sondern erst während der Ausführung in das Wort hineingetragen wird. Der Rezensent: Interessant! Theoretisch kann man ja bei der Definition eines jeden Wortes, zumindest bei Colon-Definitionen, ganz auf den Stack verzichten, indem man alle Eingaben aus Variablen holt und die Ausgaben in Variablen ablegt. Teilweises Beispiel: MOV aus dem Assembler nimmt vier und mehr Werte vom Stack auf [1 # 1000 #) MOV] und legt nichts auf den Stack zurück. Es tut nichts weiter als übersetzen, aber alles Übersetzte landet nicht mehr auf dem Stack, sondern verschwindet im Dictionary. Man traut sich nicht, in die Definition von MOV einen Stackkommentar hineinzuschreiben, da er sowieso nichts aussagt. Graham ruft zur Diskussion auf.

18 Book Review "Write Your Own Programming Language Using C++"
Boris Fennema <fennema@gofree.indigo.ie>

Der Reviewer bespricht das bekannte Buch, auf dessen Titel immer wieder und überall hingewiesen wurde, das man aber nie in einer Buchhandlung oder einer Bibliothek zu Gesicht bekommen konnte (108 Seiten, 1 Diskette, erschienen 1996, vordergründig über C++, aber in Wirklichkeit über Forth). Er hat es von Amazone (Internet) bezogen. Die Besprechung ist ungewöhnlich ausgedehnt (8 Seiten). Der Reviewer beschreibt einige Dinge, die er mit Hilfe dieses Buches angestellt hat.

26 Special Features of kForth
Krishna Myneni <krishnamyneni@compuserve.com> and David P. Wallace

Der erstgenannte Autor ist Mitglied der FIG UK, lebt aber in den USA. Dies ist der zweite von zwei Teilen eines Artikels, der für die JFAR, das Journal of Forth Applications and Research, angefertigt wurde.

Es geht um die dynamische Dictionary-Zuordnung in kForth. ALLOT von ANS-Forth reicht nicht. ALLOCATE muß her und ?ALLOT muß angeglichen werden. Komma (,) und C-Komma (C,) werden geopfert.

Der Forthwrite-Redakteur: Krishna Myneni ist Physiker. Das Programmieren hat er sich selbst beigebracht. Er beschäftigt sich gern mit neuen Experimenten, die er Stück für Stück zusammensetzt und über Software dirigiert. Forth verwendet und propagiert er seit Urzeiten, sehr zum Vergnügen seiner Kollegen.

(Der Rezensent: Man werde Pensionär. Als solcher hat man die Freiheit, im Kreise ehemaliger Kollegen auch über die Beschäftigung mit Forth zu sprechen. Schade nur, daß man mit den Erklärungen jedesmal ganz von vorn anfangen muß.)

31 Across the Big Teich
Henry Vinerts <volvovid@aol.com>

Drei Berichte (3!) von Henry über SVFIG-Aktivitäten in Originalfassung (März, April, Mai 2002). Wir kennen sie in der Übersetzung von Thomas Beierlein. Im Vorspann steht: Dieses Material wurde von Henry Vinerts für die Vierte Dimension vorbereitet und erscheint hier mit Genehmigung der Forth-Gesellschaft. Henry frug kürzlich, ob er weniger Meinung und mehr Fakten bringen solle. "Fakten sind gut", sagte Chris, "Meinungen sind besser". Ich (der Rezensent) schloß mich im Namen der FG an und schlug Henry vor, alles so zu machen wie bisher.

36 Vierte Dimension 1/02
Alan Wenham

Das wird wohl die letzte von Alans Rezensionen sein. Außergewöhnliche familiäre Belastungen hindern ihn daran, sein "Amt" fortzuführen. Es wird schwer werden, unter den Forthwrite-Lesern einen Nachfolger zu finden. Weiß jemand aus der FG Rat? Wir sollten diese Dinge nicht vernachlässigen. Eine regelmäßige Rezension der VD in der Forthwrite ist ein gutes Aushängeschild für uns.

39 German FIG Annual Conference 2002
Fred Behringer

Eine spontane, enthusiastische Meldung des Rezensenten über den Erfolg der Tagung in Garmisch. Wir danken Chris Jakeman für die freundliche Wiedergabe.

40 Dutch Forth Users Group

Diesmal wieder die Anzeige zur Mitgliederwerbung der holländischen Forthfreunde. Die Angaben noch in Gulden. Die Anfreundung mit dem Euro braucht noch ihre Zeit. Das nächste Mal sind wir wieder dran.

41 Letters

Drei Briefe: Dave Pochin bedankt sich für den ihm verliehenen Forthwrite-Preis, Chris Hainsworth schreibt von seinem Rentnersitz in Spanien aus und bedankt sich für den ihm verliehenen FIGUK-Preis, Bill Young, ein Senior Scientist, schreibt, daß er Kopien von fast allen Circuit-Cellar-Artikeln (Steve Garcia) hat. Der Rezensent erinnert sich gut an solche Garcia-Artikel in der BYTE, als er noch mit FIG-Forth auf dem VC 20 experimentierte.

Nr. 118, September 2002

1 Editorial
Chris Jakeman <cjakeman@bigfoot.com>

Chris gratuliert Bernd Paysan ("of German FIG") für seine

Teilnahme am Wettbewerb (siehe unten) und begrüßt wieder eine Firma, die als Mitglied in die FIG UK eingetreten ist.

2 Forth News

Forth Code Index; FIG UK erweitert seine Website; Dave Pochin: Getting Started; TSCP CHESS; URL-Monitor; FIR und elektronische Filter; ANS-Forth-Schriften; Delta Forth .NET; PicForth für 16F87x; globales Standortsystem.

5 From the 'Net - Choosing Forth

Sechs Punkte von Brad Rodriguez aus dem Internet, die für Forth sprechen.

7 Competitive Programming with Forth

Bernd Paysan vertrat Forth auf einem Roboter-Wettbewerb im Rahmen der diesjährigen ICFP-Konferenz (International Conference on Functional Programming). 21 Programmiersprachen: Java 28 Vertreter, C++ 23, Perl 13 ... Forth 1.

8 Linear Interpolation
Chris Jakeman

Chris zeigt, wie leicht es ist, in High-Level-Forth ein Interpolationsverfahren aufzubauen. Als Beispiel verwendet er die Sinusfunktion. 9 Werte aus einer Tafel, linear interpoliert, die einen ganzen rechten Winkel umspannen. Er gewinnt in einem zweiten Entwurf den Faktor 10 an Zeit gegenüber seinem ersten Entwurf, indem er /MOD durch RSHIFT ersetzt.

12 Rätsel
Michael Gassanenko

: X? 2DUP > TUCK INVERT AND >R AND R> OR ; Wie lautet der tatsächliche Name dieser Funktion? Ihr Rezensent hat eine Lösung nebst vollständiger mathematischer Analyse eingeschickt, die im Forthwrite-Heft 119 stehen wird.

13 F11-UK
Jeremy Fowell <jeremy.fowell@btinternet.com>

Prozessor: Motorola HC11, Version E1 - 8 MHz. E/A: 20 Anschlüsse. 2 Interrupts (IRQ & XIRQ). Analog-Eingänge: Bis zu 8 Leitungen über eingebaute 8-Bit-A/D-Wandler.

14 Book Review "The Practice of Programming"
Boris Fennema <fennema@gofree.indigo.ie>

"Ein brauchbares Buch, das die 4 Prinzipien, Klarheit, Allgemeinheit, Einfachheit und Automatisierung, gut bespricht. Beispiele hauptsächlich aus C/C++."

16 FIG UK - AGM

Die Jahresversammlung war am 19. Oktober. Diese Mitteilung enthält den Kassenbericht mit Abschluß per 31.3.2002.

18 Forth - The Early Years

Ein kurzer Hinweis auf einen Internet-Bericht von Chuck Moore. Die volle Übersetzung eben dieses Berichtes wird bei uns im vorliegenden VD-Heft zu lesen sein.

19 Iteration with Many:
Leo Wong und Chris Jakeman

Es geht um eine Nachrichtenkette in comp.lang.forth, in welcher die Frage diskutiert wird, wie man elegant eine fortlaufende Reihe von Konstanten eingeben könne, ohne inhaltlich überflüssige Worte mitschleppen zu müssen. Leo Wong verwendet MANY, das er zu MANY: korrigiert, um auch über die Zeilenenden hinaus operieren zu können.

23 Source Code Index
Chris Jakeman

Auf der Website der FIG UK befindet sich jetzt eine Auflistung von Forth-Quelltexten, die das Ziel hat, möglichst viel Material zu sammeln. (http://www.fig-uk.org/codeindex/). Die Liste enthält bis jetzt 243 Eintragungen, aufgegliedert nach 34 Sachgruppen.

25 Across the Big Teich
Henry Vinerts <volvovid@aol.com>

Drei Berichte von Henry über SVFIG-Aktivitäten in Originalfassung (Juni, Juli, August 2002). Wir kennen sie in der Übersetzung von Thomas Beierlein.

30 Vierte Dimension
Alan Wenham

Alan bedauert, aus Krankheitsgründen in der Familie die regelmäßige Rezension der Vierten Dimension nicht mehr weiterführen zu können. Chris Jakeman dankt ihm für über 4 Jahre wertvoller Arbeit und teilt den Lesern mit, daß sich Joe Anderson ab jetzt um die Rezensionen kümmern wird. Joe hat eine ganze Reihe von Jahren in führender Industrieposition in München gelebt.

31 Deutsche Forth-Gesellschaft

Unsere Anzeige zur Mitgliederwerbung.

32 Letters

Drei Briefe: Thomas Worthington mit Neuigkeiten von Aztec; Phillip Eaton, ein neuer F11-UK-Benutzer, über sein Hobby, ältere Arcade-Spiele zu restaurieren; James Power, ein Akademiker aus Irland, über seine und seiner Kollegen Untersuchungen über die Java Virtual Machine und mehr Stacksicherheit in Forth.

INDEXSORT

ein neuer Sortieralgorithmus

Filippo Sala
fsala@t-online.de

1. Allgemeines

In diesem Bericht wird ein neuer Algorithmus beschrieben, der sehr schnell, insbesondere bei grossen Zahlenmengen, arbeitet. Die Methode nützt eine verblüffende Eigenschaft der 'Inverse Funktion'. Sortiert werden nicht die Zahlen selbst sondern die Indizes. Es wird ein Feld index[i] verarbeitet, so dass am Ende die Werte des Feldes die Zahlen des Datenfeldes in aufsteigender Reihenfolge indizieren. Die Laufzeit des Algorithmus beträgt O(N).

2. Sortierung von 8-Bit-Felder

Gegeben sei ein vollständig belegtes Datenfeld mit genau 256 verschiedenen Bytes.

xi = data[i] (xi = 0 ... 255, i = 0 ... 255)

Die Bildung der Inverse-Funktion, (Daten mit Index (vertauscht) liefert das Feld invers[ ].

invers[xi] = invers[data[i]] = i

Die Werte der einzelnen Feldelemente stellen die Indizes der geordneten Elementen des Datenfeldes dar. In der Tat:

invers[0] = Index von data[i]=0
invers[1] = Index von data[i]=1
... .
invers[255] = Index von data[i]=255

Die Werte von data[ ] stellen die Indizes von invers[ ] dar. Hat das Datenfeld weniger als 256 Elemente, so wird während der Inversion das Inversfeld an manchen Stellen nicht verändert. Im Inversfeld verbleibt dort der Wert, mit dem das Feld initialisiert wurde. Die Eliminierung dieser sogenannten 'leeren Elemente' erzeugt das Indexfeld, das die Daten in aufsteigender Reihenfolge indiziert.
Ein Beispiel möge das Ganze erläutern.

2.1 Beispiel

Sortierung eines Bytefeldes mit 6 Elementen

i	0	1	3	4	5	6
xi	89	22	87	10	25	45

Die Feldinversion liefert das Inversfeld invers[ ].

i	...	10 ...	22 ...	25 ...	45 ...	87 ...	89 ...
invers[i]		4	1	5	6	3	0

Die Indizes der nicht vorhanden Elemente sind mit ... bezeichnet. Der zugehörige Wert ist gleich dem Initialiserungswert des Feldes. Wird hierfür 0 verwendet, so darf Index 0 im Datenfeld nicht vorkommen. Im Allgemein ist es nicht der Fall und daher wird in INDEXSORT das Feld invers[ ] mit -1 initialisiert.
Die Eliminierung der leeren Elemente liefert das Indexfeld index[ ].

i	0	1	2	3	5	5
index[i]	4	1	5	6	3	0

index[i] indiziert das Bytefeld in aufsteigender Reihenfolge.

10 22 25 45 87 89

2.2 Wertwiederholungen

Treten im Bytefeld Wertwiederholungen auf, (mit Sicherheit wenn u > 256), so werden nicht alle Indizes erfasst. Bei jeder Wertwiederholung wird der in invers[ ] gespeicherte Index überschrieben, und somit verbleibt dort nur der zuletzt erfasste Index. Um ausnahmslos alle Indizes zu erfassen, ist ein zusätzliches Feld notwendig. In diesem Feld (multi[ ] genannt) werden die Indizes der Wertwiederholungen in Forth-Art (d.h. als verlinkte Kette) gespeichert.
Der im Feld invers[ ] zuletzt verbliebene Index zeigt auf den nächsten Index in multi[ ] und dieser auf den Nächsten usw. bis -1 das Ende der Kette anzeigt.

2.3 Erweiterungen

Die naheliegende Erweiterung auf einen Wortfeld ist ohne weiteres möglich, aber nachteilig. Das Inversfeld hat 65536 Elemente und die Eliminierung der 'leeren' Elemente kostet viel Zeit. Die Erweiterung auf ein Longfeld ist schlichtweg unmöglich.

3. Funktionsweise von INDEXSORT

INDEXSORT benutzt die beschriebene Sortierung eines 8-Bit-Feldes mehrfach.
Vom LSB ausgehend und index[i]=i werden die Indizes 'geordnet' und verwendet für den nächsten Lauf. Allgemein gilt:

2 Läufe für 16-Bit-Zahlen
4 Läufe für 32-Bit-Zahlen
u Läufe für Zeichenketten der Länge u

Die Schleife für die Inversion des Datenfeldes läuft rückwärts, d. h., von u-1 nach 0, weil die gelinkte Kette der Indizes in multi[ ] vom LIFO-Typ ist.

3.1 Sortierung von relativen Zahlen

Im Allgemeinen liegen die relativen Zahlen in 2s'comp vor und daher sind die Bytewerte des MSB nicht linear aufsteigend. INDEXSORT wandelt die Zahlendarstellung in Offset-Format um. Dadurch laufen die Werte des MSB von 0 (kleinster Wert) nach $ff (grösster Wert). Die nachfolgende Tabelle erläutert den Zusammenhang für 32-Bit-Zahlen.

	2s'comp-Format	Offset-Format
max	$7fffffff	$ffffffff
....		
1	$00000001	$80000001
0	0	$80000000
-1	$ffffffff	$7fffffff
....		
min	$80000000	0

Die Umwandlung geschieht bei der letzte Iteration indem man zum Bytewert $80 addiert und den Übertrag unterdruckt.

3.2 Sortierprogramm

Das Sortierprogramm INDEXSORT arbeitet ohne globale Variablen und mit allozierten Feldern. Auf Grund der vielen Parameter ist die Verwendung von LOCALS zwingend notwendig.

3.3 Sortiergeschwindigkeit

Die Laufzeit von INDEXSORT beträgt O(N). Eine Vergleichmessung mit dem Sortierverfahren Quicksort ergab, wie erwartet, eine Überlegenheit von INDEXSORT bei zunehmender Anzahl der Elemente. Die Laufzeit von Quicksort ist bekanntlich O(N*logN).

Nachfolgende Messungen in Millisekunden wurden unter Win32Forth auf einem PC/100 MHz durchgeführt.

Anzahl	Indexsort	Quicksort
1000	50	50
10000	350	500
100000	3550	5720
1000000	36500	66740

Listing

```
\ INDEXSORT
\ Sortieren von Zahlen und Strings gleicher Länge
\ Filippo Sala - München
\
\ Aufruf:   dindex data u size dtype ISORT
\ dindex  Datenindexfeld
\ data    Datenfeld
\ u       Anzahl der Elemente
\ size    Elementgrösse in Bytes
\ dtype   Datentyp (0 strings, 1 signed, 2 unsigned)

ONLY FORTH ALSO DEFINITIONS

DECIMAL

\ Hilfsworte

: [] ( adr i -- adr[i] ) cells + ;

: BYTEFELD_INVERTIEREN ( noffset u invers multi dindex data size byte# -- )
   \ Daten mit Indices vertauschen.
   \ invers[data[i]] = i
   \ Indizes der Wertwiederholungen
   \ verkettet in multi.
   0 0
   LOCALS| x1 x2 byte# size data dindex multi invers |
   \
   ( nofs u ) 0 swap 1-                \ --> 0 weil multi ein LIFO ist
   DO
     dindex i [] @ to x1               \ x1 = Datenindex
     x1 size * data + byte# + c@       \ x2 = data[x1] in Offset-Format
     ( nofs x2 ) over + 255 and to x2
     invers x2 [] @  multi x1 []  !    \ Zelle im Inversfeld freimachen
     x1 invers x2 [] !                 \ invers[x2] = x1
     -1
   +LOOP
   ( nofs ) drop ;
```

```
: INDICES_ORDNEN ( invers multi dindex -- )
   \ invers und multi --> dindex
   0 0 0
   LOCALS| n x1 x2 dindex multi invers |
   0 to n                          \ Laufindex
   256 0
   DO
     invers i [] @ dup to x1       \ x1 = Datenindex aus invers
     -1 <>
     IF                            \ wenn vorhanden
       x1  dindex n []  !          \ x1 --> dindex[n]
       n 1+ to n
       BEGIN
         multi x1 [] @ dup to x2
         -1 <>                     \ x2 = Datenindex aus multi
       WHILE                       \ wenn vorhanden
         x2  dindex n []  !        \ x2 --> dindex[n]
         n 1+ to n
         x2 to x1                  \ nächster gelinkter Index
       REPEAT
     THEN
   LOOP ;

\ Anwendungswort

: ISORT ( dindex data u size dtype -- )
   0 0 0
   LOCALS| invers multi noffset dtype size u data dindex |
   256 cells ALLOCATE throw to invers
   u   cells ALLOCATE throw to multi
   0 to noffset
   u 0 DO  i dindex i [] !  LOOP  \ Indexfeld initialisieren
   \ Indexordnung bytewise bilden
   \ von LSB --> MSB
   dtype
   IF size 0 ELSE 0 size 1- THEN  \ f□r 680X0-CPU immer 0 size 1-
   DO
     invers 256 cells 255 FILL    \ Inversfeld mit -1 initialisieren
     dtype 1 =  i 3 =  and        \ wenn Zahlen mit Vorzeichen
     IF  128 to noffset  THEN     \ beim MSB 2s'comp --> Offset-Format
     \
     noffset u invers multi dindex data size i BYTEFELD_INVERTIEREN
     invers multi dindex INDICES_ORDNEN
     \
     dtype IF  1  ELSE  -1  THEN  \ f□r 680X0-CPU immer -1
   +LOOP
   multi  FREE throw
   invers FREE throw ;


\ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
\ Tests für Win32Forth
\ Pseudozufallsbytes

DECIMAL

17 VALUE startwert

: random ( -- x )
  startwert 25173 *  13849 +  dup to startwert ;


: randombytes ( adr u -- )
   ( u ) 4 / 0
   ?DO
     ( adr ) random over !
     ( adr ) cell+
   LOOP
   drop ;
```

```
\ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
\ Kontrolltest mit 32-Bit-Zahlen
\ Aufruf: NTEST

16 VALUE u      \ Anzahl der Elemente
1  VALUE dtype  \ relative Zahlen

CREATE data   u cells allot  \ Datenfeld
CREATE dindex u cells allot  \ Indexfeld

: stop? ( -- flag )
   key? dup
   IF   key 27 <>
        IF  drop key 27 =  THEN
   THEN ;

: .daten ( dindex adr u -- )
   \ Anzeige
   LOCALS| u adr dindex |
   base @ >r decimal
   u 0
   DO
     stop? IF  abort  THEN
     dindex i [] @
     adr swap [] @ cr 12 .r
   LOOP
   r> base ! ;

: ntest ( -- )
   \ Kontrolltest
   data u cells randombytes
   u 0 DO  i  dindex i []  !  LOOP
   cr cr ." Unsortiert" dindex data u .daten
   dindex  data  u  1 cells  dtype ISORT
   cr cr ." Sortiert" dindex data u .daten ;

\ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
\ Laufzeittest mit 32-Bit-Zahlen
\ Aufruf: anzahl LTEST

: ltest ( anzahl  -- )
   \ Laufzeittest
   depth 1 < abort"  Parameter?"
   0 0
   LOCALS| dindex data u |
   u cells ALLOCATE throw to dindex    \ Indexfeld
   u cells ALLOCATE throw to data      \ Datenfeld
   data u cells randombytes            \ füllen
   time-reset                          \ Wort von Win32Forth
   dindex  data  u  1 cells  1  ISORT
   cr .elapsed                         \ Wort von Win32Forth
   data   FREE throw
   dindex FREE throw
   cr ." Kontrolle? (y/n)"
   key [char] y = IF  cr dindex data u .daten  THEN ;

\ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
\ Kontrolltest mit Zeichenketten
\ Aufruf: ZTEST

10 VALUE zlaenge  \ Stringlaenge

CREATE zkette u zlaenge * allot  \ Zeichenkettenfeld ( u wie oben )

: randomchars ( adr u -- )       \ Pseudozufallszeichen
   ( u ) 0
   DO
      65 random 26 mod +
      ( adr c ) over i chars + c!
   LOOP
   ( adr ) drop ;
```

```
: .zkette ( -- )
   \ Anzeige
   u 0
   DO
      cr 2 spaces
      zkette dindex i [] @ zlaenge * +
      zlaenge 0
      DO  i over + c@ emit  LOOP
      drop
   LOOP ;

: ztest ( -- )
   \ Kontrolltest
   zkette u zlaenge * randomchars
   u 0 DO  i  dindex i []  !  LOOP
   cr cr ." Unsortiert " .zkette
   dindex zkette u zlaenge 0 ISORT
   cr cr ." Sortiert" .zkette ;
```

b16 --- Ein Forth Processor im FPGA

Bernd Paysan

Zusammenfassung

Dieser Artikel präsentiert Architektur und Implementierung des b16 Stack-Prozessors. Dieser Prozessor ist von Chuck Moores neusten Forth-Prozessoren inspiriert. Das minimalistische Design paßt in kleine FPGAs und ASICs und ist ideal geeignet für Applikationen, die sowohl Steuerung als auch Berechnungen benötigen. Die synthetisierbare Implementierung erfolgt in Verilog.

Einleitung

Minimalistische CPUs können in vielen verschiedenen Designs benutzt werden. Eine State-Maschine ist oft zu kompliziert und zu aufwendig zu entwickeln, wenn es mehr als ein paar wenige States gibt. Ein Programm mit Subroutinen kann viel komplexere Aufgaben erledigen, und ist dabei noch einfacher zu entwickeln. Auch belegen ROM- und RAM-Blöcke viel weniger Platz auf dem Silizium als „Random Logic". Das gilt auch für FPGAs, bei denen „Block RAM" im Gegensatz zu Logik-Elementen reichlich vorhanden ist.

Die Architektur lehnt sich an den c18 von Chuck Moore [1] an. Der exakte Befehlsmix ist etwas anders, ich habe zugunsten von Divisionsstep und Forth-üblicher Logikbefehle auf 2* und 2/ verzichtet; diese Befehle lassen sich aber als kurzes Makro implementieren. Außerdem ist diese Architektur byteadressiert.

Das ursprüngliche Konzept (das auch schon synthetisierbar war, und ein kleines Beispielprogramm ausführen konnte) war an einem Nachmittag geschrieben. Die aktuelle Fassung ist etwas beschleunigt, und läuft auch tatsächlich in einem Altera Flex10K30E auf einem FPGA-Board von Hans Eckes.

Die Größe und Geschwindigkeit des Prozessors kann man damit auch abschätzen.

Flex10K30E Etwa 600 LCs, die Einheit für Logik-Zellen im Altera. Die Logik zur Ansteuerung des Eval-Boards braucht nochmal 100 LCs. Im langsamsten Modell könnte man etwas mehr als 25 MHz erreichen.
Altera: Eine Logik-Zelle kann eine Logik-Funktion mit vier Inputs und einem Output berechnen, oder einen Voll-Addierer, und enthält darüber hinaus noch ein Flip-Flop.

Xfab 0.6µ ~ 1 mm^2mit 8 Stack-Elementen, das ist eine Technologie mit nur 2 Metall-Lagen.

TSMC 0.5µ < 0.4 mm^2 mit 8 Stack-Elementen, diese Technologie hat 3 Metall-Lagen. Mit einer etwas optimierten ALU kommt man mit der 5V-Library auf 100MHz.

Die ganze Entwicklung (bis auf das Board-Layout und Testsynthese für ASIC-Prozesse) ist mit freien oder kostenlosen Tools geschehen. Icarus Verilog ist in der aktuellen Version für Projekte dieser Größenordnung ganz brauchbar, und Quartus II Web Edition ist zwar ein großer Brocken zum Downloaden, kostet aber sonst nichts (Pferdefuß: Windows NT, die Versionen für richtige Betriebssysteme kosten richtig Geld).

Ein paar Sätze zu Verilog: Verilog ist eine C-ähnliche Sprache, die allerdings auf den Zweck zugeschnitten ist, Logik zu simulieren, und synthetisierbaren Code zu geben. So sind die Variablen Bits und Bitvektoren, und die Zuweisungen sind typischerweise non-blocking, d.h. bei Zuweisungen werden zunächst erst einmal alle rechten Seiten berechnet, und die linken Seiten erst anschließend verändert. Auch gibt es in Verilog Ereignisse, wie das Ändern von Werten oder Taktflanken, auf die man einen Block warten lassen kann.

1. Übersicht über die Architectur

Die Kernkomponenten sind:

- ► Eine ALU
- ► Ein Datenstack mit top und next of stack (T und N) als Inputs für die ALU
- ► Ein Returnstack, bei dem der top of return stack (R) als Adresse genutzt werden kann
- ► Ein Instruction Pointer P
- ► Ein Adreßregister A
- ► Ein Adreßlatch addr, um externen Speicher zu adressieren
- ► Ein Befehlslatch I.

Ein Blockdiagram zeigt Abbildung 1.

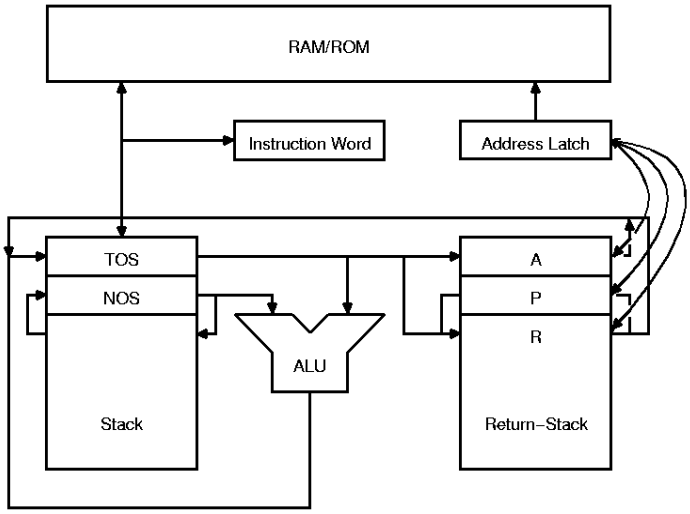


Abbildung 1

1.1 Register

Neben den für den Benutzer sichtbaren Latches gibt es auch noch Steuerlatches für das externe RAM (r und w) und Stackpointer (sp und rp), Carry c und den Wert incby, um den addr erhöht wird.

```
<register declarations>≡
   reg    r;
   reg [1:0] w;
   reg [sdep-1:0] sp;
   reg [rdep-1:0] rp;

   reg `L T, N, I, P, A, addr;

   reg [2:0] state;
   reg c;
   reg incby;
```

2. Befehlssatz

Es gibt insgesamt 32 verschiedene Befehle. Da in ein 16-Bit-Wort mehrere Befehle 'reinpassen', nennen wir die einzelnen Plätze für einen Befehlwortes einen „Slot", und das Befehlswort selbst "Bundle". Die Aufteilung hier ist 1,5,5,5, d.h. der erste Slot ist nur ein Bit groß (die höherwertigen Bits werden mit 0 aufgefüllt), die anderen alle 5 Bit.

Die Befehle in einem Befehls-Wort werden der Reihe nach ausgeführt. Jeder Befehl braucht dabei einen Takt, Speicherzugriffe (auch das Befehlsholen) brauchen nochmal einen Takt. Welcher Befehl gerade an der Reihe ist, wird in der Variablen state gespeichert.

Der Befehlssatz teilt sich in vier Gruppen: Sprünge, ALU, Memory und Stack. Tabelle [instructions] zeigt eine Übersicht über die Befehle.

Sprünge verwenden den Rest des Befehlswort als Zieladresse (außer ret natürlich). Dabei werden nur die untersten Bits des Instruction Pointers P ersetzt, es wird nichts addiert. Für Befehle im letzten Slot bleibt da natürlich nichts mehr übrig, die nehmen dann T (TOS) als Ziel.

	0	1	2	3	4	5	6	7	Comment
0	nop	call exec	jmp goto	ret ret	jz gz	jnz gnz	jc gc	jnc gnc	for slot 3
8	xor	com	and	or	+	+c	*+	/--	
10	A!+ A!	A@+ A@	R@+ R@	lit lit	Ac!+ Ac!	Ac@+ Ac@	Rc@+ Rc@	litc litc	for slot 1
18	nip	drop	over	dup	>r	>a	r>	a	

Tabelle 1: Instruction set

Name	Function
T	Top of Stack
N	Next of Stack
I	Instruction Bundle
P	Program Counter
A	Address Register
addr	Address Latch
state	Processor State
sp	Stack Pointer
rp	Return Stack Pointer
c	Carry Flag
incby	Increment Address by byte/word

```
<instruction selection>≡
   // instruction and branch target selection
   reg [4:0] inst;
   reg `L jmp;

   always @(state or I)
      case(state[1:0])
         2'b00: inst <= { 4'b0000, I[15] };
         2'b01: inst <=   I[14:10];
         2'b10: inst <=   I[9:5];
         2'b11: inst <=   I[4:0];
       endcase // casez(state)

   always @(state or I or P or T)
      case(state[1:0])
     2'b00: jmp <= { I[14:0], 1'b0 };
     2'b01: jmp <= { P[15:11], I[9:0], 1'b0 };
     2'b10: jmp <= { P[15:6], I[4:0], 1'b0 };
     2'b11: jmp <= { T[15:1], 1'b0 };
      endcase // casez(state)
```

Die eigentlichen Befehle werden dann abhängig von inst ausgeführt:

```
<instructions>≡
   casez(inst)
      <control flow>
      <ALU operations>
      <load/store>
      <stack operations>
   endcase // case(inst)
```

2.1 Sprünge

Im Einzelnen werden die Sprünge wie folgt ausgeführt: Die Sprungadresse wird nicht im P-Register gespeichert, sondern im Adreßlatch addr, das für die Adressierung des Speichers genutzt wird. Das Register P wird dann nach dem Befehlsholen mit dem inkrementierten Wert von addr gesetzt. Neben call, jmp und ret gibt's auch bedingte Sprünge, die auf 0 oder Carry testen. Das unterste Bit auf dem Returnstack wird genutzt, um das Carryflag zu sichern. Unterprogramme lassen also das Carryflag in Ruhe. Bei den bedingten Sprüngen muß man als Forther berücksichtigen, daß die den getesteten Wert nicht vom Stack nehmen.
Der Einfachheit beschreibe ich den Effekt eines jeden Befehls noch in einer Pseudo-Sprache:

nop (---)

call (--- r : P) P ← jmp; c ← 0

jmp (---) P ← jmp

ret (r : a ---) P ← a ^ $FFFE; c ← a ^ 1

jz (n --- n) if(n = 0) P ← jmp

jnz (n --- n) if(n ≠ 0) P ← jmp

jc (---) if(c) P ← jmp

jnc (---) if(c = 0) P ← jmp

```
<control flow>≡
   5'b00001: begin
      rp <= rpdec;
      addr <= jmp;
      c <= 1'b0;
      if(state == 3'b011) `DROP;
   end // case: 5'b00001
   5'b00010: begin
      addr <= jmp;
      if(state == 3'b011) `DROP;
   end
   5'b00011: begin
      { c, addr } <= { R[0], R[l-1:1], 1'b0 };
      rp <= rpinc;
   end // case: 5'b01111
   5'b001??: begin
      if((inst[1] ? c : zero) ^ inst[0])
       addr <= jmp;
       if(state == 3'b011) `DROP;
   end
```

2.2 ALU-Operationen

Die ALU-Befehle nutzen die ALU, die aus T und N ein Ergebnis res und das Carry-Bit ausrechnet. Ausnahme ist der Befehl com, der einfach nur T invertiert --- dazu braucht man keine ALU.

Die beiden Befehle *+ (Multiplikationsschritt) und /- (Divisionsschritt) schieben das Ergebnis noch über das A- Register und das Carry-Bit. *+ addiert N zum T, wenn das Carry gesetzt ist, und schiebt das Ergebnis eins nach rechts.

/- addiert auch N zum T, prüft aber, ob es dabei einen Überlauf gegeben hat, oder ob das alte Carry gesetzt war. Dabei schiebt es das Ergebnis eins nach links.

Normale ALU-Befehle nehmen einfach das Resultat der ALU in T und c, und laden N nach.

xor (a b --- r) r ← a ⊖ b

com (a --- r) r ← a ⊖ $FFFF, c ← 1

and (a b --- r) r ← a ^ b

or (a b --- r) r ← a v b

+ (a b --- r) c,r ← a + b

+c (a b --- r) c,r ← a + b + c

***+** (a b --- a r) **if**(c) c_n,r ← a + b **else** c_n,r ← 0,b;
r,A,c ← c_n,r,A

/-- (a b --- a r) c_n,r_n ← a + b + 1; **if**(c v c_n) r ← r_n;
c,r,A ← r,A,c v c_n

```
<ALU operations>≡
   5'b01001: { c, T } <= { 1'b1, ~T };
   5'b01110: { T, A, c } <=
     { c ? { carry, res } : { 1'b0, T }, A };
   5'b01111: { c, T, A } <=
     { (c | carry) ? res : T, A, (c | carry) };
   5'b01???: begin
      c <= carry;
      { sp, T, N } <= { spinc, res, toN };
   end // case: 5'b01???
```

2.3 Speicher-Befehle

Chuck Moore benutzt nicht mehr den TOS als Adresse, sondern hat ein A-Register eingeführt. Wenn man Speicherbereiche kopieren will, braucht man noch ein zweites Adreßregister; dafür nimmt er den Top-of-Returnstack R. Da man den P nach jedem Zugriff erhöhen muß (auf den nächsten Befehl), ist in der Adressierungslogik schon ein Autoinkrement enthalten. Das wird dann auch für andere Zugriffe verwendet.

Speicher-Befehle, die im ersten Slot stehen, und nicht über P indizieren, inkrementieren den Pointer nicht; damit sind Read-Modify-Write-Befehle wie +! einfach zu realisieren. Speichern kann man nur über A, die beiden anderen Pointer sind nur zum Lesen gedacht.

A!+ (n ---) mem[A] ← n; A ← A + 2

A@+ (--- n) n ← mem[A]; A ← A + 2

R@+ (--- n) n ← mem[R]; R ← R + 2

lit (--- n) n ← mem[P]; P ← P + 2

Ac!+ (c ---) mem.b[A] ← c; A ← A + 1

Ac@+ (--- c) c ← mem.b[A]; A ← A + 1

Rc@+ (--- c) c ← mem.b[R]; R ← R + 1

litc (--- c) c ← mem.b[P]; P ← P + 1

```
<address handling>≡
   wire `L toaddr, incaddr, toR, R;
   wire tos2r;

   assign toaddr = inst[1] ? (inst[0] ? P :
                                        R) : A;
   assign incaddr =
        { addr[l-1:1]  + (incby | addr[0]),
                       ~(incby | addr[0]) };
   assign tos2r = inst == 5'b11100;
   assign toR = state[2] ? incaddr :
              (tos2r ? T : { P[15:1], c });
```

Der Zugriff kann nicht nur wortweise, sondern auch byteweise erfolgen. Dazu gibt es zwei Write-Leitungen. Für byteweises Speichern wird das untere Byte in T ins obere kopiert.

```
<load/store>≡
   5'b10000: begin
      addr <= toaddr;
      w <= 2'b11;
   end
   5'b10100: begin
      addr <= toaddr;
      w <= { ~toaddr[0], toaddr[0] };
      T <= { T[7:0], T[7:0] };
   end
   5'b10???: begin
      addr <= toaddr;
      r <= 1'b1;
   end
```

Speicherzugriffe benötigen einen Extra-Takt. Dabei wird das Ergebnis des Speicherzugriffs verarbeitet.

```
<load-store>≡
  <debug>
  state <= nextstate;
  <pointer increment>
  r <= 1'b0;
  w <= 2'b0;
  if(|state[1:0]) begin
     <store afterwork>
  end else begin
     <ifetch>
  end
  <next>
```

Eine kleine Besonderheit gibt's beim angesetzten Instruction-Fetch (dem NEXT der Maschine) noch: Wenn der aktuelle Speicherbefehl ein Literal ist, müssen wir incaddr statt P nehmen.

```
<next>≡
 if(nextstate == 3'b100) begin
     { addr, r } <= { &inst[1:0] ?
                      incaddr : P, 1'b1 };
 end // if (nextstate == 3'b100)
```

```
<debug>≡
   $write("%b[%b] T=%b%x:%x[%x], ",
       inst, state, c, T, N, sp);
   $write("P=%x, I=%x, A=%x, R=%x[%x],
       res=%b%x\n"
       P, I, A, R, rp, carry, res);
```

Ist der Zugriff beendet, muß das Resultat abgearbeitet werden --- bei Load-Zugriffen der Wert auf den Stack oder ins Instruction-Register geladen, bei Store-Zugriffen der TOS gedropt werden.

```
<store afterwork>≡
 if(r)
    if(incby)
       { sp, T, N } <= { spdec, data, T };
    else
       { sp, T, N } <= { spdec, 8'h00,
       addr[0] ? data[7:0] : data[1-1:8], T };
 if(|w)
    `DROP;
 incby <= 1'b1;
```

Außerdem muß bei Bedarf die inkrementierte Adresse zurück in den entsprechenden Pointer geladen werden.

```
<pointer increment>≡
   casez({ state[1:0], inst[1:0] })
       4'b00??: P <= incaddr;
       4'b1?0?: A <= incaddr;
     // 4'b1?10: R <= incaddr;
       4'b??11: P <= incaddr;
   endcase // casez({ state[1:0], inst[1:0] })
```

Damit der erste Befehl (nur nop oder call) keine unnötige Zeit verbraucht, wird ein nop hier einfach übersprungen. Das ist der zweite Teil des NEXTs.

```
<ifetch>≡
   I <= data;
   if(!data[15]) state[1:0] <= 2'b01;
```

2.4 Stack-Befehle

Die Stack-Befehle ändern den Stackpointer und schieben entsprechend die Werte in und aus den Latches. Bei den 8 benutzten Stack-Effekten fällt auf, daß swap fehlt. Stattdessen gibt es nip. Der Grund ist eine damit mögliche Implementierungs-Option: Man kann das separate Latch N einfach weglassen, und diesen Wert direkt aus dem Stack-RAM holen. Das dauert zwar länger, spart aber Platz.

Außerdem behauptet Chuck Moore, daß man swap gar nicht so nötig braucht --- wenn es nicht verfügbar ist, behilft man sich mit den anderen Stack-Operationen, und wenn es gar nicht anders geht, gibt's ja immer noch >a >r a r>.

nip (a b --- b)

drop (a ---)

over (a b --- a b a)

dup (a --- a a)

>r (a --- r : a)

>a (a ---) A ← a

r> (r : a --- a)

a (--- a) a ← A

```
<stack operations>≡
   5'b11000: { sp, N } <= { spinc, toN };
   5'b11001: `DROP;
   5'b11010: { sp, T, N } <= { spdec, N, T };
   5'b11011: { sp, N } <= { spdec, T };
   5'b11100: begin
      rp <= rpdec; `DROP;
   end // case: 5'b11100
   5'b11101: begin
      A <= T; `DROP;
   end // case: 5'b11101
   5'b11110: begin
       { sp, T, N } <= { spdec, R, T };
       rp <= rpinc;
   end // case: 5'b11110
   5'b11111:  { sp, T, N } <= { spdec, A, T };
```

Wer auf swap nicht verzichten möchte, kann einfach die Implementierung des nips in der ersten Zeile ersetzen:

```
<swap>≡
   5'b11000: { T, N } <= { N, T };
```

3. Beispiele

Ein paar Beispiele sollen zeigen, wie man den Prozessor programmiert. Die Multiplikation funktioniert – wie gesagt – über das A-Register. Es ist ein Extra-Schritt nötig, weil ja jedes Bit zunächst einmal ins Carry geschoben werden muß. Da call das Carry-Flag löscht, brauchen wir uns darum nicht zu kümmern.

```
<mul>≡
   : mul ( u1 u2 -- ud )
       >A 0 #
       *+ *+ *+  *+ *+ *+  *+ *+ *+
       *+ *+ *+  *+ *+ *+  *+ *+
       >r drop a r> ;
```

Auch bei der Division muß ein Extra-Schritt eingelegt werden. Eigentlich bräuchten wir hier wirklich ein swap, da wir

aber keines haben, nehmen wir zunächst over und nehmen in Kauf, daß wir ein Stackelement mehr brauchen als im anderen Fall. Anders als bei mul müssen wir hier nach dem com das Carry wieder löschen. Und am Schluß müssen wir noch den Rest durch zwei teilen, und den Carry nachschieben.

```
<div>≡
   : div ( ud udiv -- uqout umod )
      com >r >r >a r> r> over 0 # +
      /- /- /-  /- /- /-  /- /- /-
      /- /- /-  /- /- /-  /- /-
      nip nip a >r -cIF *+ r> ;
      THEN 0 # + *+ $8000 # + r> ;
```

Das nächste Beispiel ist etwas komplizierter, weil ich hier eine serielle Schnittstelle emuliere. Bei 10MHz muß jedes Bit 87 Takte brauchen, damit die Schnittstelle 115200 Baud schnell ist. Erst hinter dem zweiten Stop-Bit haben wir Ruhe, die Gegenseite wird sich schon wieder synchronisieren, wenn das nächste Bit kommt.

```
<serial line>≡
 : send-rest ( c -- c' ) *+
 : wait-bit
  1 # $FFF9 # BEGIN  over +  cUNTIL  drop
  drop ;
 : send-bit ( c -- c' ) nop \ delay at start
 : send-bit-fast ( c -- c' )
  $FFFE # >a dup 1 # and
  IF    drop $0001 # a@ or  a!+ send-rest ;
  THEN  drop $FFFE # a@ and a!+ send-rest ;
 : emit ( c -- ) \ 8N1, 115200 baud
  >r 06 # send-bit r>
  send-bit-fast send-bit send-bit send-bit
  send-bit send-bit send-bit send-bit
  drop send-bit-fast send-bit drop ;
```

Der ; hat hier wie bei ColorForth die Funktion des EXITs, so wie der : nur ein Label einleitet. Steht vor dem ; ein Call, so wird der in einen Sprung umgewandelt. Das spart Returnstack-Einträge, Zeit und Platz im Code.

4. Der Rest der Implementierung

Zunächst einmal den Rumpf der Datei.

```
<b16.v>≡
/*
 * b16 core: 16 bits,
 * inspired by c18 core from Chuck Moore
 *
<inst-comment>
 */

`define L [l-1:0]
`define DROP { sp, T, N } <=
 { spinc, N, toN }
<ALU>
<Stack>
<cpu>
```

```
<inst-comment>≡
* Instruction set:
* 1, 5, 5, 5 bits
*      0     1     2     3     4     5     6     7
*  0: nop   call  jmp   ret   jz    jnz   jc    jnc
*  /3       exec  goto  ret   gz    gnz   gc    gnc
*  8: xor   com   and   or    +     +c    *+    /-
* 10: A!+   A@+   R@+   lit   Ac!+  Ac@+  Rc@+  litc
*  /1 A!    A@    R@    lit   Ac!   Ac@   Rc@   litc
* 18: nip   drop  over  dup   >r    >a    r>    a
```

4.1 Toplevel

Die CPU selbst besteht aus verschiedenen Teilen, die aber alle im selben Verilog-Modul implementiert werden.

```
<cpu>≡
   module cpu(clk, reset, addr, r, w,
   data, T);

      <port declarations>
      <register declarations>
      <instruction selection>
      <ALU instantiation>
      <address handling>
      <stack pushs>
      <stack instantiation>
      <state changes>
      always @(posedge clk or negedge reset)
      <register updates>

   endmodule // cpu
```

Zunächst braucht Verilog erst mal Port Declarations, damit es weiß, was Input und Output ist. Die Parameter dienen dazu, auch andere Wortbreiten oder Stacktiefen einfach einzustellen.

```
<port declarations>≡
   parameter l=16, sdep=3, rdep=3;
   input clk, reset;
   output `L addr;
   output r;
   output [1:0] w;
   input  `L data;
   output `L T;
```

Die ALU wird mit der entsprechenden Breite instanziiert, und die nötigen Leitungen werden deklariert.

```
<ALU instantiation>≡
   wire `L res, toN;
   wire carry, zero;
   alu #(l) alu16(res, carry, zero,
              T, N, c, inst[2:0]);
```

Da die Stacks nebenher arbeiten, müssen wir noch ausrechnen, wann ein Wert auf den Stack gepusht wird (also nur wenn etwas gespeichert wird).

```
<stack pushs>≡
      reg dpush, rpush;
      always @(clk or state or inst or r)
          begin
            dpush <= 1'b0;        rpush <= 1'b0;
            if(state[2]) begin
              dpush <= |state[1:0] & r;
              rpush <= state[1] & (inst[1:0]==2'b10);
            end else
               casez(inst)
                  5'b00001: rpush <= 1'b1;
                  5'b11100: rpush <= 1'b1;
                  5'b11?1?: dpush <= 1'b1;
               endcase // case(inst)
      end
```

Zu den Stacks gehören nicht nur die beiden Stack-Module, sondern auch noch inkrementierter und dekrementierter Stackpointer. Beim Returnstack kommt erschwerend dazu, daß der Top of Returnstack manchmal geschrieben wird, ohne daß sich die Returnstacktiefe ändert.

```
<stack instantiation>≡
   wire [sdep-1:0] spdec, spinc;
   wire [rdep-1:0] rpdec, rpinc;

   stack #(sdep,1) dstack(clk, sp, spdec,
                          dpush, N, toN);
   stack #(rdep,1) rstack(clk, rp, rpdec,
                          rpush, toR, R);

   assign spdec = sp-{{(sdep-1){1'b0}}, 1'b1};
   assign spinc = sp+{{(sdep-1){1'b0}}, 1'b1};
   assign rpdec = rp+{(rdep){(~state[2] |
                                    tos2r)}};

   assign rpinc = rp+{{(rdep-1){1'b0}}, 1'b1};
```

Der eigentliche Kern ist das voll synchrone Update der Register. Die brauchen einen Reset-Wert, und für die verschiedenen Zustände müssen die entsprechenden Zuweisungen codiert werden. Das meiste haben wir weiter oben schon gesehen, nur das Befehlsholen und die Zuweisung des nächsten States und des Wertes von incby bleibt noch zu erledigen.

```
<register updates>≡
    if(!reset) begin
        <resets>
    end else if(state[2]) begin
       <load-store>
    end else begin // if (state[2])
        <debug>
        if(nextstate == 3'b100)
            { addr, r } <= { P, 1'b1 };
        state <= nextstate;
        incby <= (inst[4:2] != 3'b101);
        <instructions>
    end // else: !if(reset)
```

Als Reset-Wert stellen wir die CPU so ein, daß sie sich gerade den nächsten Befehl holen will, und zwar von der Adresse 0. Die Stacks sind alle leer, die Register enthalten alle 0.

```
<resets>≡
   state <= 3'b100;
   incby <= 1'b1;
   P <= 16'h0000;
   addr <= 16'h0000;
   A <= 16'h0000;
   T <= 16'h0000;
   I <= 16'h0000;
   c <= 16'h0000;
   r <= 1'b1;
   w <= 2'b00;
   sp <= 0;
   rp <= 0;
```

Der Übergang zum nächsten State (das NEXT innerhalb eines Bundles) wird getrennt erledigt. Das ist nötig, weil die Zuweisungen der anderen Variablen zum Teil nicht nur abhängig vom aktuellen State sind, sondern auch vom nächsten (z.B. wann das nächste Befehlswort geholt werden soll).

```
<state changes>≡
   reg [2:0] nextstate;
   always @(inst or state or w or r)
    if(state[2]) begin
        <rw-nextstate>
    end else begin
        casez(inst)
             <inst-nextstate>
        endcase // casez(inst[0:2])
    end // else: !if(state[2]) end

<rw-nextstate>≡
   nextstate <= state[1:0] + { 2'b0, |
                              state[1:0] };
<inst-nextstate>≡
   5'b00000: nextstate <= state[1:0] + 3'b001;
   5'b00???: nextstate <= 3'b100;
   5'b10???: nextstate <= { 1'b1,
                     state[1:0] };
   5'b?????: nextstate <= state[1:0] + 3'b001;
```

4.2 ALU

Die ALU berechnet einfach die Summe mit den verschiedenen möglichen Carry-ins, die logischen Operationen, und ein Zero-Flag. Zwar können hier gemeinsame Ressourcen verwendet werden (die XORs des Volladdierers können auch die XOR-Operation machen, und die Carry-Propagation könnte OR und AND berechnen), dieses Quetschen von Logik überlassen wir aber dem Synthese-Tool.

```
<ALU>≡
 module alu(res, carry, zero, T, N, c, inst);
    <ALU ports>
        wire        `L sum, logic;
    wire        cout;
        assign { cout, sum } =
            T + N +  ((c | andor) & selr);
    assign logic = andor ?
          (selr ? (T | N) : (T & N)) : T ^ N;
    assign { carry, res } =
         prop ? { cout, sum } : { c, logic };
    assign zero = ~|T;
    endmodule // alu
```

Die ALU hat die Ports T und N, carry in und die untersten 3 Bits des Befehls als Input, ein Ergebnis, carry out und der Test auf 0 als Output.

```
<ALU ports>≡
   parameter l=16;
   input `L T, N;
   input c;
   input [2:0] inst;
   output `L res;
   output carry, zero;
   wire prop, andor, selr;

   assign #1 { prop, andor, selr } = inst;
```

4.3 Stacks

Die Stacks werden im FPGA als Block-RAM implementiert. Dazu sollten sie am besten nur einen Port haben, denn solche Block-RAMs gibt's auch in kleinen FPGAs. Im ASIC wird diese Art von Stack mit Latches implementiert. Dabei könnte man auch Read- und Write-Port trennen (oder für FPGAs, die dual-ported RAM können), und sich den Multiplexer für spset sparen.

```
<Stack>≡
 module stack(clk, sp, spdec, push, in, out);
     parameter dep=3, l=16;
     input clk, push;
     input [dep-1:0] sp, spdec;
     input `L in;
     output `L out;
     reg `L stackmem[0:(1@<<dep)-1];
     wire [dep-1:0]  spset;
     always @(clk or push or spset or in)
        if(push & ~clk) stackmem[spset]
            <= #1 in;
     assign spset = push ? spdec : sp;
     assign #1 out = stackmem[spset];
   endmodule // stack
```

4.4 Weitere mögliche Optimierungen

Eigentlich könnte man Speicherzugriffe und Berechnungen auf den Stacks überlappend ausführen. Durch die separaten Pointer-Register ist das möglich. Die Verständlichkeit des Prozessors würde darunter aber sicher leiden, und der kritische Pfad würde wohl auch länger. Angesichts einer garantierten Beschleunigung um 25% (der Zyklus zum Befehlsholen fällt weg) und einer maximalen Beschleunigung um 100% (bei speicher-intensiven Anwendungen) könnte es aber wert sein --- wenn der Platz da ist.

Falls Platz knapp ist, kann man fast alle Register als Latches auslegen. Nur T muß ein echtes Flip-Flop bleiben. Für FPGAs ist das keine Option, Flip-Flips sind dort allemal günstiger.

4.5 Skalierbarkeit

Zwei mögliche Ansatzpunkte gibt es, den b16 schnell an eigene Wünche anzupassen: Die Wortbreite und die Stacktiefe. Die Stacktiefe ist dabei der einfachste Punkt. Die gewählte Tiefe 8 ist für den Bootloader ausreichend, kann aber für komplexere Applikationen Schwierigkeiten bereiten. Einfachere Applikationen sollten dagegen mit einem kleineren Stack zurechtkommen.

Die Wortbreite kann auch der Applikation angepaßt werden. So wird eine auf 12 Bit abgemagerte Version in einem Projekt bei meinem Arbeitgeber Mikron AG eingesetzt. Dabei muß man natürlich noch das Decodieren der einzelnen Befehle im Slot ändern, und die Logik zum Überspringen des ersten nops anpassen.
Außerdem kann man natürlich einzelne Befehle auswechseln. So wird bei der 12-Bit-Version kein Byte-Zugriff auf den Speicher benötigt, aber relativ viele Bit-Zugriffe. Entsprechend werden die Byte-Zugriffe dann durch Bit-Operationen auf den obersten Teil des Speichers ersetzt, und das Register incby weg-optimiert (nur noch Wort-Zugriffe).

5. Entwicklungsumgebung

Hier könnte ich noch ein etwas längeres Listing präsentieren, diesmal in Forth. Ich will mich aber auf eine Funktions-Beschreibung beschränken. Alle drei Programme sind in einer einzigen Datei vereint, und erlauben damit eine interaktive Benutzung des Simulators und des Targets.

5.1 Assembler

Der Assembler ist leicht an Chuck Moore's ColorForth angelehnt. Es gibt aber keine Farben, sondern nur normale Interpunktion, wie in Forth üblich. Der Assembler ist schließlich in Forth geschrieben, und erwartet damit Forth-Tokens.

Labels definiert man mit : und |. Erstere geben automatisch einen Call, können aber mit ' auf den Stack gelegt werden. Letztere entsprechen etwa einem interaktiven Create. Labels können nur rückwärts aufgelöst werden. Literals muß man explizit mit # oder #c vom Stack nehmen. Die Zuordnung in Slots nimmt der Assembler selber in die Hand. Ein ret compiliert man normal mit einem ;, der vorangestellte Calls in einen jmp konvertiert. Man kann Makros definieren (macro: ... endmacro).

Auch die aus Forth bekannten Kontrollstrukturen können (bzw. müssen --- für Vorwärtssprünge) verwendet werden. IF wird zu einem jz, jnz erreicht man mit -IF. cIF und -cIF entsprechen jnc und jc. Entsprechende Prefixes gibt es auch für WHILE und UNTIL.

5.2 Downloader

Im FPGA ist ein Stück Block-RAM mit einem Programm vorbelegt, dem Boot-Loader. Dieses kleine Programm läßt ein Lauflicht laufen, und wartet auf Kommandos über die serielle Schnittstelle (115.2kBaud, 8N1, kein Handshake). Es gibt drei Kommandos, die mit ASCII-Zeichen eingeleitet werden:

0 addr, len, <len*data>: Programmiere den Speicherbereich ab addr mit len Datenbytes

1 addr, len: Lese len Bytes vom Speicherbereich ab addr zurück

2 addr: Führe das Wort an addr aus.
Diese drei Befehle reichen aus, um den b16 interaktiv zu bedienen. Auch auf der Host-Seite reichen ein paar Befehle aus:

comp Kompiliert bis zum Ende der Zeile, und schickt das Ergebnis an das Eval-Board

eval Kompiliert bis zum Ende der Zeile, schickt das Ergebnis ans Eval-Board, führt den Code aus, und setzt den RAM-Pointer des Assemblers zurück an den Ausgangspunkt

sim Wie eval, nur wird das Kompilat nicht vom FPGA ausgeführt, sondern vom Emulator

check (addr u ---) Liest den entsprechenden Speicherbereich vom Eval-Board, und zeigt ihn mit dump an

6. Ausblick

Mehr Material gibt's auf meiner Homepage. Alle Quellen sind unter GPL verfügbar. Wer ein bestücktes Board haben will, wendet sich am besten an Hans Eckes. Und wer den b16 kommerziell verwenden will, an mich.

References

[1] c18 ColorForth Compiler, Chuck Moore, 17th EuroForth Conference Proceedings, 2001

[2] b16 Processor, Bernd Paysan, Internet Homepage, http://www.jwdt.com/~paysan/b16.html

...in letzter Minute...

Ein Hinweis aus de.comp.lang.forth, von Jörg Plewe:

http://www.immersive.com

Virtuelle Welten in Forth programmiert. Schauen Sie mal ‘rein...

Neues von "Avise"

Wolfgang Schemmert
Jan. 2003

Avise steht für "AVR Virtual Stack Engine". Die erste Version wurde beschrieben in der "Vierten Dimension", Heft Nr 4/2000, S.21. Vor etwa einem Jahr wurde die weiterentwikkelte Version 2 ins Internet gestellt. Im Unterschied zur ersten Version wurde dort der Stack auf 16 Bit Datenworte aufgebohrt und frei benennbare Variable eingeführt.

Soeben ist die Version 3 fertig geworden. Sie bringt einige interessante programmiersprachliche Erweiterungen mit sich, die hier vorgestellt werden sollen.

Bereits im oben erwähnten VD-Artikel wurde der Wunsch formuliert, eine Basic-ähnliche Syntax mit Forth Eigenschaften zu haben.

Ich habe verschiedene in Hochsprache programmierbare Interpreter ausprobiert, z.B. den "Little C" Interpreter von H. Schildt [1]. Dieses und vergleichbare Produkte leiden unter dem Nachteil, dass sie dazu zwingen, ein "Programm" ähnlich wie bei Compilerprogrammierung zu entwickeln. D.h., sie zwingen mehr oder weniger zu einem "top down" Programmierverfahren: "Unterprogramme" werden von einem "Hauptprogramm" aufgerufen.

Für eine interaktive oder auch spielerisch-austestende Programmiermethode eignet sich aber besser das bei Forth angewandte "bottom up" Verfahren, d.h., das letztliche "Programm" ist das Resultat einer Folge sich nach und nach entwickelnder und jeweils für sich einsetzbarer Teilfunktionen - "Worte".

Beim Studium der verschiedenen Lehrbücher über Interpreter-Programmierung - mir haben vor allem die Bücher von R. Mak [2] und N.Wirth [3] weitergeholfen - stellt sich heraus, dass die meisten Basic- C- oder Pascal- Interpreter ohnehin für die Laufzeit-Ausführung den "infix" Sourcecode in einen Forth-ähnlichen "postfix" Tokencode umsetzen. Die hierfür eingesetzte Standardmethode (bei einfachen Interpretern) ist der "rekursive Abstieg". Ohne das hier vertiefen zu können:

Die Methode des "rekursiven Abstiegs" besteht darin, die verschiedenen Operatoren entsprechend ihrer Priorität in gegenüber dem Sourcecode veränderter Reihenfolge auszuführen bzw. zu kompilieren.

Genauer formuliert besteht also die Zielsetzung darin, den Forth-Kern um ein Modul zur Ausführung des rekursiven Abstiegs zu erweitern. Während ein Standard Forth Interpreter die Worte des Sourcecodes strikt nacheinander abarbeitet, muss ein für den rekursiven Abstieg geschriebener Interpreter als entscheidende Erweiterung stets ein Wort im Voraus interpretieren und sich Operationen von geringer Priorität zunächst für die spätere Ausführung merken.

Schon seit langem habe ich abgeheftet den Artikel von Bernd Paysan "Infix nach Postfix" aus der VD 4/1991 [4], um das irgendwann mal umzusetzen. Mehr oder weniger durch "anfangen und weitermachen" bin ich zu einem etwas anderen Konzept gelangt.

Das Problem des "Merkens von Operatoren" habe ich nicht mit einem separaten Operatorenstack gelöst, sondern durch den vorgezogenen Abstieg in Subroutinen höherer Priorität. Die Priorität jedes Operators ist durch ein zusätzliches Byte in der Symboltabelle (Forth-Slang: im Dictionary) eingetragen. Zu merkende Details der Symboltabelle werden ebenfalls mit einem "push" auf dem Returnstack zwischengelagert und wieder ge"pop"t, wenn diese Subroutine ihrer Priorität entsprechend zu Ende geführt wird. Der Operatoren-Merkspeicher wird also auf den ohnehin vorhandenen CPU-Returnstack verlagert. Anders als im o.g. Artikel von Bernd Paysan ist "Avise" strikt in möglichst kompaktem Assemblercode programmiert, daher kann meine Umsetzung hier nicht in Form von "Screens" dokumentiert werden.

Der Infix-Interpreter sollte in die Forth-Umgebung integriert sein, statt dass - wie bei den bekannten Basic etc. Interpretern üblich - der Forth-ähnlich konstruierte Laufzeitkern unsichtbar im Untergrund arbeitet. Daraus ergibt sich ein Vorteil im Sinne spontaner Arbeitsweise gegenüber anderen Programmiersprachen: die Möglichkeit, "Worte" linear hintereinander aufzurufen. Diese Möglichkeit bleibt beim hier vorgestellten Konzept erhalten, auch wenn sie nicht zwingend genutzt werden muss.

Um zu einer "flüssigen" Ausdrucksweise zu gelangen, musste leider ein Forth-Heiliger vom Sockel gekippt werden:

Variable (deklariert mit VAR) legen beim Aufruf grundsätzlich ihren WERT auf den Stack.

Wenn man die Adresse braucht, muss man AT davorstellen. Das AT ist technisch äquivalent mit dem LET oder SET verschiedener Programmiersprachen, scheint mir von der sprachlichen Formulierung her aber auf alle hier notwendigen Anwendungen am besten zu passen.

Der "Untergrund" funktioniert wiederum Forth-klassisch: sei KARL eine vorher deklarierte Variable.

Der simple Aufruf "KARL" erzeugt folgendes Kompilat:
DOVAR <Parameteradresse von KARL> RD

Der Aufruf vom "AT KARL" hingegen erzeugt
DOVAR <Parameteradresse von KARL>

DOVAR ist technisch praktisch identisch mit DOLIT, wird aber aus systematischen Gründen als eigenes Runtime-Primitive verwaltet. Aus mnemonischen und tipptechnischen Gründen ersetzt RD den Forth-Klammeraffen @.

Der Forth-Sprachumfang wird vor allem durch 5 neue Operatoren ("Worte") erweitert.

Wie man sieht, erinnert die schließlich real entstandene Syntax eher an ein extrem reduziertes Pascal als an Basic:

:= ist der Zuweisungsoperator (was man aus der Schule beim Formelschreiben als Gleichheitszeichen kennt). Wenn z.B. KARL eine vorher deklarierte Variable ist, so kann - auch eingebaut mitten in den Source-Code eines Forth-Wortes - ihr ein Wert zugewiesen werden in der Form:

AT KARL := <Folge von Zahlen und Operatoren> ;

; ist der Zuweisungs-Abschluss Operator. Das Semikolon wurde umgewidmet, um die sprachliche Ähnlichkeit zu anderen Programmiersprachen zu wahren. Es können mehrere Zuweisungen in eine Kommandozeile gereiht werden, was aber nicht besonders schön ist. Das Forth-"Semis"-Wort wurde dafür umbenannt in "RET"

Das Verfahren ist folgendes: **:=** entnimmt die zu diesem Zeitpunkt auf dem TOS liegende Parameteradresse von KARL, **;** schreibt als Abschluss des zuzuweisenden Ausdrucks den daraus erzeugten TOS dorthin. Abweichend von anderen Programmiersprachen darf eine Zuweisung auch mehr als ein Ergebnis erzeugen. Der NOS und alle weiteren Stackparameter können dann in Variable abgelegt werden durch nachfolgende Programmzeilen der Form

AT OTTO := ;
AT WILLI := ;

oder natürlich auch per Stack weiterverarbeitet werden. Das AT ist nur bei unmittelbar interpretierender Ausführung aus der Kommandozeile notwendig, beim Kompilieren kann es entfallen.

(und) sind die Vorrang-Operatoren,

D.h., geklammerte Sourcecode Segmente werden am engsten zusammengezogen und erhalten hohe Priorität - wie in der Schulmathematik. Kommentare werden stattdessen in geschweifte Klammern { ... } gesetzt. Zur Erleichterung des Tippens und Wahrnehmens haben die Klammer-Operatoren eingebautes "Delimiter"(=Wort-Trenner)-Verhalten, d.h. müssen nicht mit einer Leerstelle umgeben werden.

Schliesslich, das ist für Forthler wahrscheinlich am wichtigsten:

| ist der Postfixer.

Ein nachgestellter Postfixer verwandelt einen Infix Operator in einen Postfix Operator. Der Postfixer wirkt nur auf den unmittelbar davor stehenden Infix Operator, d.h., es gibt keine generelle Umschaltung zwischen "Infix" und "Postfix" Modus. Zur Erleichterung des Tippens und Wahrnehmens hat der

Postfixer ebenfalls eingebautes "Delimiter" Verhalten, d.h. muss nicht mit einer Leerstelle umgeben werden.

Anders ausgedrückt: der Postfixer verhindert den rekursiven Abstieg, d.h., veranlasst die sofortige Interpretation/ Kompilation des davor stehenden Operators. Man könnte auch formulieren: Der Postfixer verleiht dem vorangestellten Operator höchste Priorität. Ohne ihn haben alle Operatoren (soweit das einen Sinn macht) **als Source-Code das aus der "normalen" Mathematik bekannte "infix" Verhalten**. Der "rekursive Abstieg" organisiert jedoch eine Umsortierung der Operatoren entsprechend ihrer Priorität, **im Ergebnis ist die Laufzeitorganisation aller Operatoren "postfix"!** (Wie man mit "SEE" nachprüfen kann).

Wie das alles gemeint ist, geht hoffentlich aus weiter unten folgenden Beispielen hervor. Operatoren, die nur bei stackorientierter Programmierung einen Sinn machen, wie DUP, DROP; SWAP, OVER, ROT haben inherentes Postfix Verhalten, d.h., brauchen keinen Postfixer.

Das Komma ist ein syntaktisch ein der Leerstelle äquivalenter Delimiter.

Zusammen mit den Klammer-Operatoren können Funktionen (Worte) so mit Parameterlisten aufgerufen werden, wie es aus anderen Programmiersprachen bekannt ist. Um das zu veranschaulichen, 3 vom Ergebnis her völlig identische Sourcecode Fetzen:

2 + 3 oder **2 3 +|** oder **+(2,3)**

Alle 3 werden interpretiert/kompiliert als Forth-Fadencode .

DOLIT 2 DOLIT 3 +

Noch nicht perfekt gelöst ist die Parameterübergabe beim Aufruf von Secondaries. Nehmen wir an, wir haben ein Secondary "FUN" programmiert, das 2 Stackparameter X1 und X2 benötigt. Das lässt sich in folgenden äquivalenten Formen aufrufen:

FUN(X1,X2) oder gleichwertig: **X1 X2 FUN**

Diese Parameter können aber innerhalb des Secondary nicht mit eigenem Namen aufgerufen werden, wie das in z.B. in C oder Pascal üblich ist. Hier zeigt sich bereits einer der Gründe, warum bei "Avise" Variable ihren Wert statt der Adresse auf den Stack legen: statt X1 und X2 können ohne Probleme Namen von (globalen) Variablen eingesetzt werden, was bei Standard Forth zu unlösbaren Problemen führen würde.

Die **Prioritäten der verschiedenen Operatoren** entsprechen in etwa anderen Programmiersprachen:

Die höchste Priorität haben gleichberechtigt:
- --- Klammer, Zahl und Identifier
  (d.h., Instanzen von VAR und CONST)
- --- Forth spezifische Operatoren
  (DUP, DROP, SWAP, OVER, ROT etc)
- --- und mit dem Postfixer geschmückte Kernel-Operatoren
  (Primaries) der tieferen Prioritätsklassen

Danach folgen mit absteigender Priorität:

- --- Secondary
  (=vom Anwender programmiertes "Wort")
  und Operatoren, deren wesentliche Funktion darin besteht, Daten auf den Stack zu lesen, z.B. RD und AIN (=Analogeingang lesen).
- --- multiplikative Operation (* / und AND)
- --- additive Operation (+ - OR und XOR)
- --- Vergleichs-Operation (= , <> , >, < , etc.)

Schließlich gibt es noch eine nicht genau zuzuordnende Klasse von Operatoren, gegenüber denen allein Klammerausdrükke Vorrang haben (z.B. IF, UNTIL, EMIT).

Damit die Theorie nicht ganz grau bleibt, noch **einige Beispiele**:

Dass **3 * (4 + 2)** dasselbe bewirkt wie **3 4 2 +| *|** muss in Forth-Kreisen wohl nicht erwähnt werden.

zu IF .. ELSE .. ENDIF

```
VAR OTTO
VAR KARL
: XYZ
  IF (OTTO = 1)
     KARL := 7 ;
ELSE
  KARL := 3 ;
ENDIF
RET
```

ist gleichwertig mit bzw. kompiliert das Gleiche wie

```
VAR OTTO
VAR KARL
: XYZ
  OTTO 1 = IF
  7 AT KARL WR
          ELSE
  3 AT KARL WR
          ENDIF
RET
```

Aus mnemonischen und tipptechnischen Gründen ersetzt WR das Forth-Ausrufezeichen !.

Das syntaktisch überflüssige THEN habe ich mir erspart. Grundsätzlich ist in "Avise" auch die bei Pascal oder C übliche Blockbildung überflüssig, da Anfänge und Enden aller Blöcke durch eindeutige Operatoren gekennzeichnet sind.

Zu Schleifen:

Bereits seit der ersten Version wird bei "Avise" der Schleifenparameter in einer Variablen verwaltet. Um mehr Ähnlichkeit zu Pascal oder Basic herzustellen, wurde die Syntax der DO ... LOOP Schleife modifiziert. Man beachte, dass FOR die gleiche Funktionalität aufweist wie oben erwähntes AT, der Name wurde nur zur besseren Wiedererkennung gedoppelt. Ob die Schleife aufwärts oder abwärts zählt, wird automatisch erkannt durch signierten Vergleich des Start- mit dem Stopwert. Die Standard-Schrittweite ist +1 oder -1. Da der Schleifenparameter in einer Variablen gehalten wird, kann er innerhalb der Schleife explizit manipuliert werden. Zum Abbruch der Schleife wird ein signiertes >= bzw. <= getestet, d.h. der manipulierte Schleifenparameter muss nicht exakt den Stopwert treffen.

```
VAR KARL
  : XYZ
    FOR KARL(1,10) DO
    KARL .
    KARL := KARL + 3 ;
  NEXT
RET
```

ist gleichwertig mit

```
VAR KARL
: XYZ
1 10 AT KARL -ROT DO
      KARL .
    3 AT KARL +W
NEXT
RET
```

Bei der Postfix-Variante wurde absichtlich ein zusätzlicher Schnörkel eingebaut, durch den es von der Konstruktion her möglich ist, den Start- und Stopwert zur Laufzeit auf dem Stack zu übergeben, was bei der Infix-Variante bisher leider nur durch Einsatz zusätzlicher Variablen möglich ist.

Obwohl die hier vorgestellten sprachlichen Erweiterungen des Gerard Baecker an Vorkenntnisse aus Schulmathematik oder anderen Programmiersprachen anknüpfen lassen, bringen sie keinerlei Einschränkung mit sich, Programme auf Forth-spezifische Art zu entwickeln und auszuprobieren.

Eine detaillierte Beschreibung aller Operatoren und das Manual zu "Avise3" ist verfügbar auf unserer Webseite "www.cinetix.de/avise/"

Bereits im ersten Artikel in der VD 4/2000 wurde eine **einfache Hardware** für "Avise" beschrieben. Die Platine ist weiterhin "aktuell". Statt für den Auslauftyp AT90S4433 wird in der aktuellen Version ein Objektcode-Download für den pinkompatiblen ATmega8 angeboten. Weiterhin wird auf unserer Webseite der Aufbau einer Platine für den AT90S8535 beschrieben. Schliesslich ist eine Objektcode-Version für das Atmel-Evaluationsboard STK500 mit AT90S8515 verfügbar.

[1] Herbert Schildt, "C ent-packt", Bonn, mitp Verlag, 2001
[2] Ronald Mak, "Writing Compilers & Interpreters", New York, J.Wiley&Sons, 1991
[3] Nikolaus Wirth, "Grundlagen und Techniken des Compilerbaus", Bonn, Addison-Wesley, 1996
[4] Bernd Paysan, "Infix nach Postfix" VD 4/1991 S. 16ff.

Dies und Das

Das Neueste von Avise finden Sie im Netz unter **cinetix.de/avise/deutsch/index.htm**. Dort fehlt natürlich auch der Hinweis auf Forth nicht. Und unter **cinetix.de/avise/deutsch/gloindex.htm** sehen Sie dann im „Index nach Token Code“ ein forthige Liste bekannter Worte.
Schauen Sie einfach einmal dort vorbei. *fep*

Die tollen Tage stehen vor der Tür. Da ist der im Rheinland lebende Editor der VD froh, die vorliegende Ausgabe gerade noch rechtzeitig fertig bekommen zu haben. Bald spielen hier

nämlich sogar die SWAPs verrückt. Während Prinzessinnen ihre Frösche küssen, verkleiden sich Drachen als Königinnen. Und kaum ist dieses Fest vorüber, gehen die Drachen mit den Hasen ins Nest.

Helau, Alaaf und

fröhliche Ostern *fep*

Humor ist, wenn man trotzdem lacht.

Richtigstellung

In einer Ankündigung, die die Computerindustrie verblüffte, haben Ken Thompson, Dennis Ritchie und Brian Kernigham zugegeben, daß das von ihnen geschaffene Betriebssystem Unix und die Programmiersprache C ein raffinierter Aprilscherz sind, der sich über 20 Jahre am Leben erhalten hat. Bei einem Vortrag vor dem letzten UnixWorld-Software-Entwicklungsforum enthüllte Thompson:

"1969 hatte AT&T gerade die Arbeit am GE/Honeywell/AT&T-Multics-Projekt beendet. Brian und ich experimentierten zu dem Zeitpunkt mit einer frühen Pascal-Version von Professor Niklaus Wirth vom ETH-Laboratorium in der Schweiz und waren beeindruckt von seiner Einfachheit und Mächtigkeit. Dennis hatte gerade 'Der Herr der Klinge' gelesen, eine spöttische Parodie auf Tolkiens große Triologie 'Der Herr der Ringe'.

Im Übermut beschlossen wir, Parodien zur Multics-Umgebung und zu Pascal zu verfassen. Dennis und ich waren für die Betriebssystemumgebung verantwortlich.

Wir sahen uns Multics an und entwarfen ein neues System, das so komplex und kryptisch wie möglich sein sollte, um die Frustration der gelegentlichen Nutzer zu maximieren.

Wir nannten es Unix in Anspielung auf Multics und fanden es auch nicht gewagter als andere Verballhornungen. Danach entwickelten Dennis und Brian eine wirklich perverse Pascal-Version namens 'A'. Als wir bemerkten, daß einige Leute tatsächlich versuchen, in A zu programmieren, fügten wir schnell einige zusätzliche Fallstricke hinzu und nannten es B, BCPL und schließlich C. Wir hörten damit auf, als wir eine saubere Übersetzung der folgenden Konstruktion erhielten:

```
for(;P("\n"),R--;P("!"))for((e=C;e--;P("_"+(*u++/8)%2))
```

[*die zweite Zeile war leider nicht zu entziffern*]

Der Gedanke, daß moderne Programmierer eine Sprache benutzen würden, die solch eine Anweisung zuließ, lag jenseits unseres Vorstellungsvermögens. Wir dachten allerdings daran, alles den Sowjets zu verkaufen, um ihren Computerfortschritt 20 Jahre und mehr zu behindern. Unsere Überraschung war groß, als dann AT&T und andere US-Unternehmen tatsächlich begannen, Unix und C zu verwenden! Sie haben 20 weitere Jahre gebraucht, genügent Erfahrungen zu sammeln um einige bedeutungslose Programme in C zu entwickeln, und das mit einer Parodie auf die Technik der 60er Jahre! Dennoch sind wir beeindruckt von der Hartnäckigkeit (falls nicht doch Gemeinsinn) des gewöhnlichen Unix- und C-Anwenders. Jedenfalls haben Brian, Dennis und ich in den letzten Jahren nur in Pascal auf einem Apple Macintosh programmiert, und wir fühlen uns echt schuldig an dem Chaos, der Verwirrung und dem wirklich schlechten Programmierstil, der von unserem verrückten Einfall vor so langer Zeit ausging."

Namenhafte Unix- und C-Anbieter und Benutzer, einschließlich AT&T, Microsoft, Hewlett-Packard, GTE, NCR und DEC haben vorläufig jede Stellungnahme abgelehnt.

Borland International [...] meinte, sie hätten diesen Verdacht schon seit Jahren gehegt und würden nun dazu übergehen, ihre Pascal-Produkte zu verbessern, und weitere Bemühungen um die C-Entwicklung stoppen.

Ein IBM-Sprecher brach in unkontrolliertes Gelächter aus.

Wie Programmierer ihre Räder bauen

ADA-	Programmierer bauen ein viereckiges Rad und passen alle Straßen an.
ALGOL-	Programmierer weigern sich, Räder zu bauen, weil es sie irgendwohin bringen könnte.
APL-	Programmierer schweben in höheren Sphären, sie brauchen keine Räder.
ASSEMBLER-	Programmierer bauen tausende von Rädern, keine zwei passen auf eine Achse.
BASIC-	Programmierer bauen nur ein Rad, aber finden keine Achse dazu.
COBOL-	Programmierer bauen TAUSEND-RÄDRIGE-TRANSPORT-MODULE und verbieten das Gehen.
FUNKTIONAL-	Programmierer rufen eine Funktion HOLZ auf und hoffen, ein Rad zu bekommen.
FORTH-	Programmierer bauen Räder und vergessen, wo sie sie gestapelt haben.
FORTRAN-	Programmierer werden wahnsinnig bei der Suche nach Rädern, die mit I beginnen.
LOGO-	Programmierer bauen kleine rote Autos.
LISP-	Programmierer (bauen Räder mit (Rädern mit(Rädern mit (Rädern mit (dem was LISP-Programmierer bauen))))).
MICRO-	Programmierer wissen nicht, daß Räder existieren.
PASCAL-	Programmierer erklären das Gehen zur Tugend.
PL/I-	Programmierer setzen ein Team ein, um eine Räderfabrik zu entwerfen, die Räder der falschen Größe bauen wird.
RPG-	Programmierer haben ein Rad, schade daß es viereckig ist.
SYSTEM-	Analytiker sind viel zu beschäftigt, Räder zu suchen, um eins zu bauen.
TOS-	Benutzer bleiben wo sie sind, sie sind das Warten gewohnt.
UNIX	hat unter irgendeiner Schale sicher irgendwo ein Verzeichnis von Rädern.
C-	Programmierer laufen, weil keiner die Bauanleitungen für ihre Räder lesen kann

Möchten Sie mehr Spaß? Sind Sie gewappnet für extreme Angriffe auf Ihre Lachmuskeln? Halten Sie auch die Scherze unseres Forth-Freundes Wolfgang A. aus S. aus? Dann sehen Sie sich doch einfach einmal um bei **WWW.KissMyFloppy.com**. Bitte schauen Sie sich unbedingt die Animationen an. Und BITTE, schreiben Sie uns, wie Ihnen dieser Tip gefallen hat 8=()

fep

Forth-Gruppen regional

Moers

Friederich Prinz
Tel.: 02841-58398 (p) (Q)
(Bitte den Anrufbeantworter nutzen !)
(Besucher: Bitte anmelden !)
Treffen: 2. und 4. Samstag im Monat
14:00 Uhr, **MALZ, Donaustraße 1**
47441 Moers

Mannheim

Thomas Prinz
Tel.: 06271-2830 (p)
Ewald Rieger
Tel.: 06239-920 185 (p)
Treffen: jeden 1. Mittwoch im Monat
Vereinslokal Segelverein Mannheim e.V.
Flugplatz Mannheim-Neuostheim

München

Jens Wilke
Tel.: 089-89 76 890
Treffen: jeden 4. Mittwoch im Monat
Ristorante Pizzeria Gran Sasso
Ebenauer Str. 1
80637 München

µP-Controller Verleih

Thomas Prinz
Tel.: 06271-2830 (p)
micro@forth-ev.de

Gruppengründungen, Kontakte

Hier könnten SIE
sich zur Gründung einer
lokalen Gruppe zur Verfügung stellen !

Forth-Hilfe für Ratsuchende

Forth allgemein

Jörg Plewe
Tel.: 0208-49 70 68 (p)

Jörg Staben
Tel.: 02103-24 06 09 (p)

Karl Schroer
Tel.: 02845-2 89 51 (p)

Spezielle Fachgebiete

Arbeitsgruppe MARC4 — **Rafael Deliano**
Tel./Fax: 089-841 83 17 (p)

FORTHchips (FRP 1600, RTX, Novix) — **Klaus Schleisiek-Kern**
Tel.: 040-375 008 03 (g)

F-PC & TCOM, Asyst (Meßtechnik), embedded Controller (H8/5xx// TDS2020, TDS9092), Fuzzy — **Arndt Klingelnberg**, Consultants
akg@aachen.kbbs.org
Tel.: ++32 +87 -63 09 89 (pgQ)
(Fax -63 09 88)

KI, Object Oriented Forth, Sicherheitskritische Systeme — **Ulrich Hoffmann**
Tel.: 04351 -712 217 (p)
Fax: -712 216

Forth-Vertrieb
vlksFORTH
ultraFORTH
RTX / FG / Super8
KK-FORTH
Ingenieurbüro **Klaus Kohl**
Tel.: 08233-3 05 24 (p)
Fax : 08233-99 71
mailorder@forth-ev.de

Möchten Sie gerne in Ihrer Umgebung eine lokale Forthgruppe gründen, oder einfach nur regelmäßige Treffen initiieren ? Oder können Sie sich vorstellen, ratsuchenden Forthern zu Forth (oder anderen Themen) Hilfestellung zu leisten ? Möchten Sie gerne Kontakte knüpfen, die über die VD und das jährliche Mitgliedertreffen hinausgehen ?

Schreiben Sie einfach der VD - oder rufen Sie an - oder schicken Sie uns eine E-Mail !

Hinweise zu den Angaben nach den Telefonnummern:

Q = Anrufbeantworter
p = privat, außerhalb typischer Arbeitszeiten
g = geschäftlich

Die Adressen des Büros der Forthgesellschaft und der VD finden Sie im Impressum des Heftes.

1 / 2003

Forth-Tagung 2003

Lambrecht, Neustadt/Weinstraße

11. bis 13.April 2003

Jahrestagung der Forthgesellschaft e.V.

Haben Sie sich schon angemeldet?

Anmeldeformulare mit allen notwendigen Hinweisen (Organisation, Adressen, Kosten...) finden Sie als Einlegeblatt in dieser Ausgabe, oder im Internet unter:

http://www.forth-ev.de/tagung/forth2003.html

Sie können auch die Organisationsleitung direkt per Mail erreichen:

ewald.rieger@t-online.de