The Forth Enlightenment

Hardware for the Forth programming language

Charles Edward Pax

December 11, 2024

Self-publishing

Contents

1 Introduction

In This Chapter

you will learn

- First topic
- Second topic
- Third topic

you will be able to

- Task one
- Task two
- Task three

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

1.1 Purpose

The purpose of this book is to address the necessity of trustless computing or verifyable computing.

- Don’t trust, verify
- trustless computing
- trusting the compiler

Design princples: simplicity is a virtue.

Don't trust, verify

In this section talk about blah blah blah.

Trustless computing

Discuss trustless computing here.

Trusting the compilers

In this subsection there will be a discussion of trusting the compiler.

Application: Bitcoin

These topics are relevant to Bitcoin because it relates to money. Don't get hacket. Artificial Intelligence is coming for your money.

1.2 Hardware Overview

No microcode. Implement logic in hardware. Complex optimization is bad. conceptual simplicity. Simplicity is a cirtue.

visual verification view with your own eyes
simplicity keep it simple to understand
repitation repeat blocks

Stack computers

- Why stack computers?
- Parts of a stack computer
- Types of stack computers

Do we cover the instruction cycle here? Instruction timing?

1.3 Instruction set Architecture

The instruction set architecture should be kept simple. Each bit encodes specific information.

The software should:

- be simple
- reuse code blocks
- not require a compiler
- be hand-writable

Threaded code. No compiler. Forth.

2 Hardware

In This Chapter

you will learn

- First topic
- Second topic
- Third topic

you will be able to

- Task one
- Task two
- Task three

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2.1 Stack Computers

We will discuss stack computers[1].

[1]: Philip J. Koopman (1989), *Stack Computers the New Wave*

2.2 Architecture Block Diagram

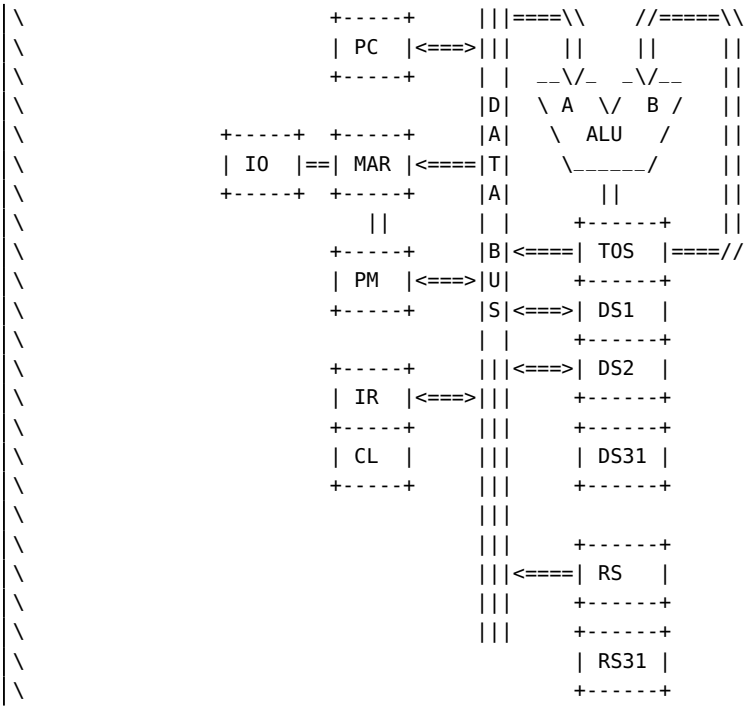

Figure 2.1: Simplified architecture block diagram

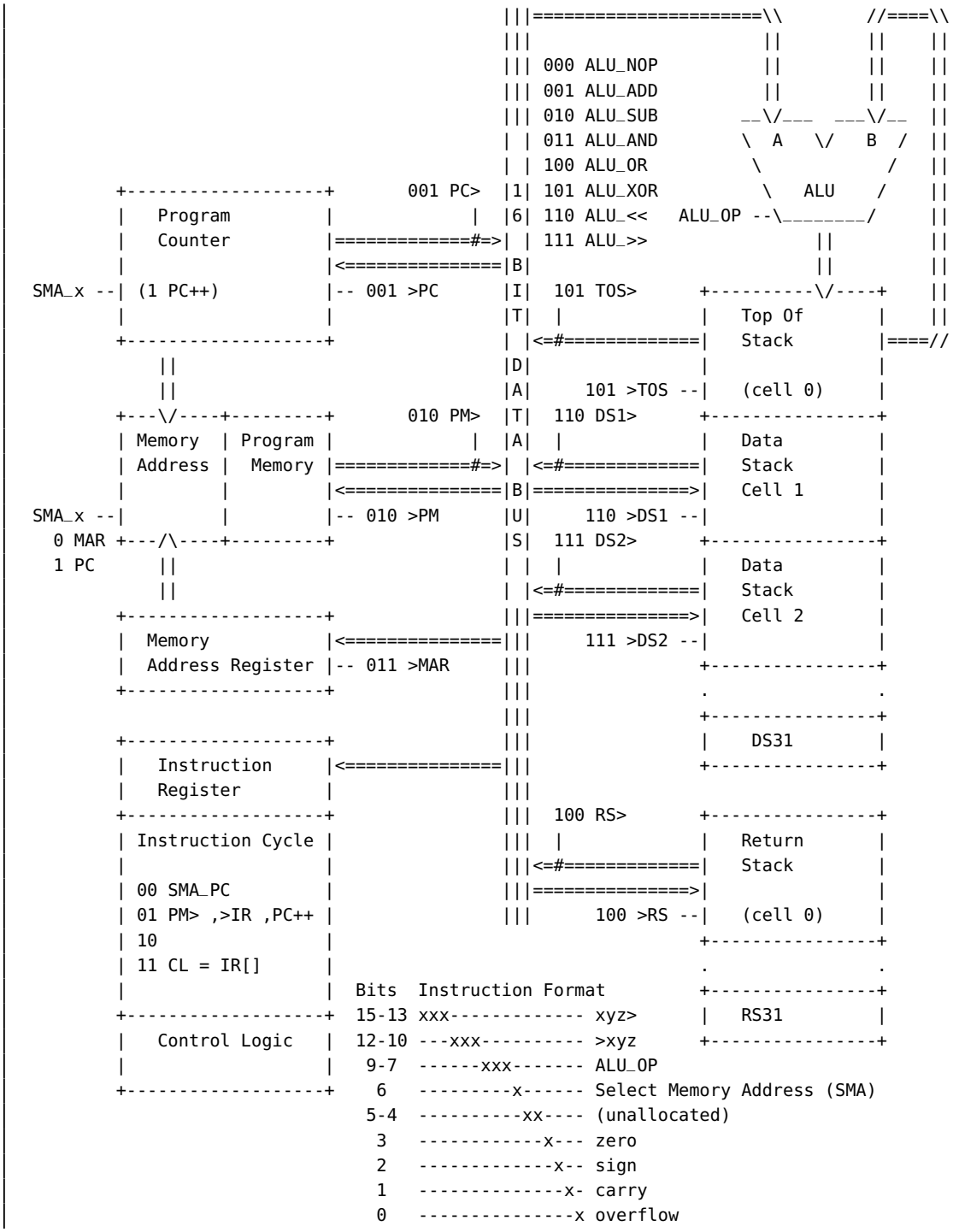

Figure 2.2: Complete architecture block diagram

Table 2.1: Instruction Set Architecture (ISA)

Word	Binary OP code	OP code description
NOP	`000 000 000 0 00 0000`	No Operation
RESET	`111 111 111 1 11 1111`	Reset system
NO>	`000 ... ... . .. ....`	No input source. High *Z*
PC>	`001 ... ... . .. ....`	Program Counter out
PM>	`010 ... ... . .. ....`	Program Memory out
MAR>	`011 ... ... . .. ....`	Memory Address Register out
RS>	`100 ... ... . .. ....`	Return stack out
TOS>	`101 ... ... . .. ....`	POP Data Stack cell 0
DS1>	`110 ... ... . .. ....`	POP Data Stack cell 1
DS2>	`111 ... ... . .. ....`	POP Data Stack cell 2
>NO	`... 000 ... . .. ....`	No output destination.
>PC	`... 001 ... . .. ....`	Input to Program Counter
>PM	`... 010 ... . .. ....`	Input to Program Memory
>MAR	`... 011 ... . .. ....`	Input to Memory Address Register
>RS	`... 100 ... . .. ....`	PUSH to Return stack
>TOS	`... 101 ... . .. ....`	PUSH to Data Stack cell 0
>DS1	`... 110 ... . .. ....`	PUSH to Data Stack cell 1
>DS2	`... 111 ... . .. ....`	PUSH to Data Stack cell 2
ALU_NOP	`... ... 000 . .. ....`	No ALU operation selected
ALU_ADD	`... ... 001 . .. ....`	ALU addition
ALU_SUB	`... ... 010 . .. ....`	ALU subtraction
ALU_AND	`... ... 011 . .. ....`	ALU bitwise AND
ALU_OR	`... ... 100 . .. ....`	ALU bitwise OR
ALU_XOR	`... ... 101 . .. ....`	ALU bitwise XOR
ALU_«	`... ... 110 . .. ....`	ALU bitshift left
ALU_»	`... ... 111 . .. ....`	ALU bitchift right
SMA_MAR	`... ... ... 0 .. ....`	Select Memory Address from MAR
SMA_PC	`... ... ... 1 .. ....`	Select Memory Address from PC
	`... ... ... . xx ....`	(unallocated)
if=0	`... ... ... . .. 1...`	Zero
if<0	`... ... ... . .. .1..`	Sign
if_C	`... ... ... . .. ..1.`	Carry
if_OF	`... ... ... . .. ...1`	Overflow

3 Instruction Set Architecture (ISA)

In This Chapter

you will learn

- First topic
- Second topic
- Third topic

you will be able to

- Task one
- Task two
- Task three

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3.1 ISA Explained

Explain what an ISA is.

3.2 Instruction cycle

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not

at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3.3 Instruction Format

3.4 Instruction Timing

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

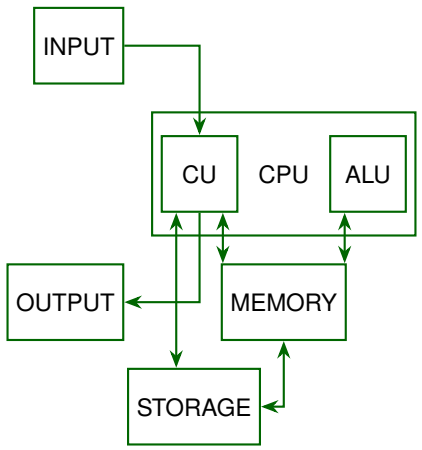

Figure 3.1: This is an example timing diagram that will be replaced by the real thing.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of

the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

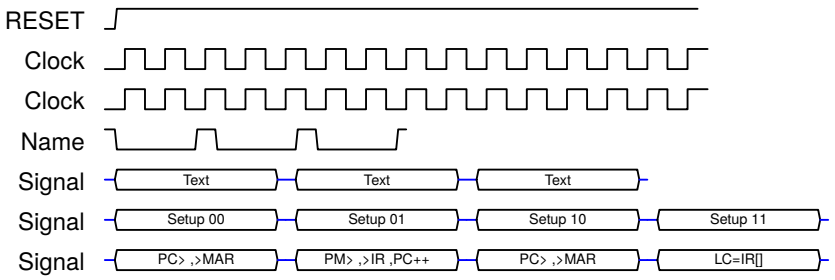

Figure 3.2: This is an example timing diagram that will be replaced by the real thing.

```
1
2  RESET _/
3
4
5   CLK  ____/       \_______/       \_______/       \_______/       \_______/
6        ... | Execute 00    | Execute 01    | Execute 10    | Execute 11   |
7            |  PC> ,>MAR    | PM> ,>IR ,PC++| PC> ,>MAR     | CL=IR[]      | ...
8        ____          _______         _______         _______         _______
9  ~CLK      \_______/       \_______/       \_______/       \_______/       \____
10   ... Setup 00    | Setup 01      | Setup 10      | Setup 11      | ...
11       PC> ,>MAR   | PM> ,>IR ,PC++| PC> ,>MAR     | CL=IR[]       |
```

Figure 3.3: Instructions are executed. The contyrol logic is configured in between each step. Timing chart.

4 The Forth Programming Language

In This Chapter

you will learn

- First topic
- Second topic
- Third topic

you will be able to

- Task one
- Task two
- Task three

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.1 Dictionary Entry Format

Each word in Forth is defined by a dictionary entry immediate word.

The dictionary entry format is dependent upon the Forth implmentation, but a typical 16-bit Forth system will implement the structure described in Figure 4.1.

Add headings for these three columns: Address, Contents, Description

where
NFA is the address pointing to the first cell in the word’s name field,
LFA is the address pointing to the word’s link pointer field,
CFA is the address pointing to the word’s code pointer field,
PFA is the address pointing to the first cell in the word’s parameter field,

how are these items sorted? Alphabetical? Order of appearance?

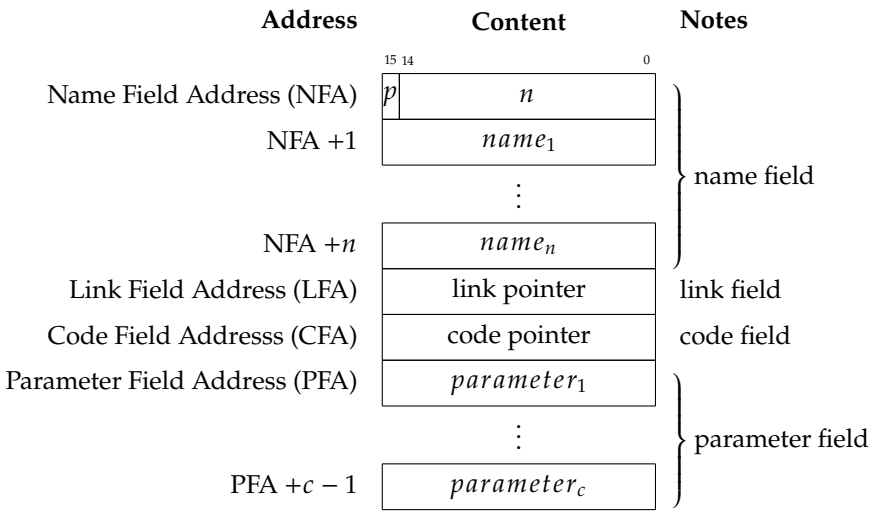

Figure 4.1: Dictionary entry format.

p is the precedence bit (0,1),
n is the number of characters in the word's name (e.g. $\text{DUP}_n = 3$),
$name_n$ is the n^{th} character in the name,
c is the number of cells in the parameter field, and
$parameter_c$ is the c^{th} cell in the parameter field.

This text uses the term parameter field. Other resources may use the term data field or other term.

Each dictionary entry has a link pointer field, which holds an memory address pointing to the preceeding dictionary entry. By design, when the system searches for a word it starts at the end of the dictionary and works its way to the beginning by following these pointers. We call these addresses link pointers because they link the words together and to distinguish them from pointers with other purposes.

Each dictionary entry in the dictionary will share the same format for the name field, link field, and code field. The parameter field format depends on the word type. Constants, variables, colon-defined words, and any other type of defining word can have its own uniquie parameter field format.

The link pointer of a dictionary entry points to the name field of the previous word. The value of a link pointer is equal to the NFA of the pervious dictionary entry. The link pointer is an address pointing to the first cell of the previous dictionary entry's name field. The LFA of a dictionary entry contains the NFA of the previous dictionary entry.i

Code pointer contains an address. This address is the "address of the instruction first executed when this type of word is executed." Starting Forth p. 222

Variable words. The code pushes the variable address onto the stack.

Constant words. The code pushes the contents of the constant onto the stack.

Colon defined words. The code executes the rest of the words in the colon definition.

In all the cases the code pointed to is called "tun-time code."

All words of a given type have the same code pointer.

code address is the address of the machine instruction the interpreter will load for execution, and
data is any data that is required for execution. todo[noline]Decide if the name field characters are 16-bit or 8-bit in the final implementation.

The code address field is also known as the code pointer field. The data field is also known as the parameter field.

Precedence bit. 0 - address of the word gets compiled normally. 1 - The word is executed during compilation.

Are the characters in the name field limited to ASCII? Is there any technical reason they could not be any arbitrary 16-bit value? Maybe null would be a problem.

Leo Brodie's Starting Forth[2] is good reading for this.

[2]: Brodie (1981), *Starting FORTH*

4.2 The Forth Inner Interpreter

1: Use sidenotes if necessary

This section gives a general explaination[1] of the chapter topic[3].

[3]: Loeliger (1981), *Threaded Interpretive Languages*

The word `:` causes the innter interpreter to enter compile mode. The name of the word is next, followed by the definition. The word `;` causes the inner interpreter to exit compilation and reenter interpretation mode.

Example:

Listing 4.1: Definition of `dup` in Forth.

```
1 : square ( x -- x )
2     dup
3     *
4     ;
```

This, of course, can be more conveniently written on a single line

```
: square ( x -- x )
    dup * ;
```

4.3 Return Stack

The top of the return stack holds the return address. When a function call is made, the return address must be stored. If the call command is executed at address x, the return address x+1 is pushed to the top of the return stack.

4.4 Data Stack

The data styack is where temporary data is stored.

If a system has only one hardware stack, it shoulld be used for the data stack. Data manipulations are more common than function calls and the hardware stack is faster than memory access. The call stack will need to be stored in memory.

For "colon words" (words compiled with the : word), the data field is a series of cells, each cell containing an address pointing to a word in the Forth dictionary.

The address interpreter (inner interpreter) copies the address stored in the address field into the interpreter pointer then executes the code at that address.

The interpreter does not keep track of or know about the type of word it is working with. The code linked to by the CFA is responsible for behaving appropriately.

4.5 The Forth Outter Interpreter

This an intermediate section. There will be several. each such section will cover a discrete topic within this chapter.

Margin notes can be used to explain detail or add reminders that would otherwise break the flow of the document. I think sidenotes are numbered while margin notes are not.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.6 Forth Primitives

4.7 Metacompiler

see BYO Assembler, Moving Forth

4.8 Notes

Check out the YouTube channel learnmeabitcoin. The creator has a great video explaining SHA256 and another video explaining the Bitcoin Script stack machine.

Table 4.1: Example interpretation of a Forth word. This table would be paired with a code listing for the word. A column can be added for the return stack or other information. Exercises can be made in the style of fill-in-the-missing-item.

`: times14` (n_1 — n_2), $n_2 = 14 * n_1$

Step	Data Stack	Interpret	Description
1	n_1	2	2 → TOS
2	n_1 2	3	3 → TOS
3	n_1 2 3	4	4 → TOS
4	n_1 2 3 4	+	(x_1 x_2 — x_3), $x_3 = x_1 + x_2$
5	n_1 2 7	*	(x_1 x_2 — x_3), $x_3 = x_1 * x_2$
6	n_1 14	*	(x_1 x_2 — x_3), $x_3 = x_1 * x_2$
7	n_2	;	Return to interpreter

```
1     : literal  ( n --  )
2         (Usage: : <name> ... [ n ] literal ... ; )
3         (Description: Compiles LIT and TOS number (n). When <name>
      is executed, n is placed on TOS.)
4         state @ if compile lit , then ; immediate
5
6     :lit  (  -- n  )
7         PM> ,>TOS ,PM-PC
```

5 The Forth System

In This Chapter

you will learn

- what is the first topic
- how to identify the second topic
- best practices for the third topic

you will be able to

- perform the steps required to accomplish task one
- create a program that performs task two

This introduction gives the reader any context necessary to understand the proceeding sections in this chapter.

The contents of the 'In This Chapter' box should concrete. This should coorespond to a 'Chapter Review' section at the end of the chapter.

Illuminate any common obstacles for the reader. A few words of encouragement are also welcomed.

5.1 Things Forth Does

5.1.1 System Operations

Note: the text interpreter handles input from terminal and files. It also handles compilation

Initialize system

Copy values into the correct location in memory. Set register values. Example: Read from memory the designated address for the stack pointer and load that into a particular register.

Interpret input stream

The outer interpreter (AKA text interpreter). The input stream could be a keyboard buffer, serial buffer, or other source.

Execution

'Innter interpreter.' 'Address interpreter.' Track which word is being executed and which to execute next. Jumping to machine code. Controls the flow of execution.

Error detection and handling

Usually by clearing the stack and returning control to the outer interpreter.

5.2 Forth System Parameters

Data Stack Pointer (DS), Return Stack Pointer (RS), Interpreter Pointer(IP), Word Pointer (W).

Data Stack Pointer. Holds the address of the top cell of the data stack. SP can be in memory or in a register. Register is better. This paragraph is true for systems that have the stack in RAM.

The data stack can be held in memory or in hardware stack provided by the CPU. The same is true of the return stack.

When using a hardware stack provided by the CPU, the Top of Stack (ToS) element is typically the only element that can be accessed by the data bus. Some microprocessors provide access to multiple cells. Three cells are a good number (Stack Computers?).

Return Stack Pointer (RS). Similar to the data stack. The data stack pointer and the data stack itself can beach be stored in memory. RS can be a hardware stack, but doing so does not provide the same performence gain as doind so with the data stack (source?).

During execution, a word may use the Return Stack for temkporary storage. Be careful!

Interpretative Pointer (IP). 'instruction pointer?'. Points to the code field (?) of the next word to be executed. Does thi depend on the system implementation? Can be stored in memory or a register.

Word Pointer (WP). Points to the word currently being executed. (Name field? Code field?).

5.3 Chapter Review

Chapter Review

you have learned

- what is the first topic
- how to identify the second topic
- best practices for the third topic

you are able to

- perform the steps required to accomplish task one
- create a program that performs task two

6 Template Chapter

In This Chapter

you will learn

- what is the first topic
- how to identify the second topic
- best practices for the third topic

you will be able to

- perform the steps required to accomplish task one
- create a program that performs task two

This introduction gives the reader any context necessary to understand the proceeding sections in this chapter.

The contents of the 'In This Chapter' box should concrete. This should coorespond to a 'Chapter Review' section at the end of the chapter.

Illuminate any common obstacles for the reader. A few words of encouragement are also welcomed.

6.1 Heaps of Examples

This section is fairly dense with examples. You can delete everything from here until the end of the page.[1]

1: After you have saved this document as a new file.

This section serves as an easy way to demonstrate* multiple graphical elements that can be used in this book.

Add a list of elements

Keep chapters, sections, and sub sections capitalized.

6.1.1 Margin Stuff

Generally, any text that would be written between parentheses would do well as a margin note. Keep the flow going.

Remember

a kaobox can be used in the margins for special reminders)

* Like this footnote

Sections in Depth

Notice that `\subsubsection` does not cause numbering.

Sections within a chapter can be `\section`, `\subsection`, or `\subsubsection`.

6.1.2 Figures and Tables

Listing 6.1: This Forth code does NOT generate Table 6.1

```
: square  ( x -- x )
( Forth code that squares a number
   dup * ;
```

This is an example of a table contained withing the main column of text.

Table 6.1: A useless table.

col1	col2	col3	col4
Multiple row	cell2	cell3	cell4
	cell5	cell6	cell7
	cell8	cell9	cell10
Multiple row	cell2	cell3	cell4
	cell5	cell6	cell7
	cell8	cell9	cell10

We can also have full-width tables.

Table 6.2: A wide table with invented data about three people living in the UK. Note that wide figures and tables are centered and their caption also extends into the margin.

Name	Surname	Job	Salary	Age	Height	Country
Alice	Red	Writer	4.000 £	34	167 cm	England
Bob	White	Bartender	2.000 £	24	180 cm	Scotland
Drake	Green	Scientist	4.000 £	26	175 cm	Wales

And full-width images.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Figure 6.1: A wide seaside, and a wide caption. Credits: By Bushra Feroz, CC BY-SA 4.0, `https://commons.wikimedia.org/w/index.php?curid=68724647`

6.1.3 Citations

Cite an author in the margin[4].

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

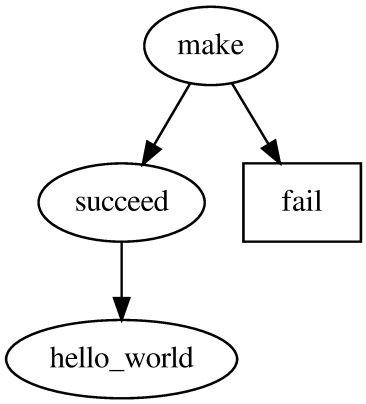

Figure 6.2: A diagram produced by the Dot program and saved as a PNG file.

[4]: McZampleface (1999), *The Great Book of Examples*

6.1.4 Glossaries and Indicies

6.1.5 Mathematics

The box below was created using the `definition` environment. Boxes are also provided by the `theorem`, `proposition`, `lemma`, `corollary`, `example`, `remark`, and `exercise` environments.

Definition 6.1.1 *Let y be some function of x such that $y = f(x)$*

where x and y are both variables in Definition 6.1.1.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font,

how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

6.2 Chapter Review

Chapter Review

you have learned

- what is the first topic
- how to identify the second topic
- best practices for the third topic

you are able to

- perform the steps required to accomplish task one
- create a program that performs task two

7 Code Listings

! (*n addr* —) Store *n* at *addr*.

```
1    : !  ( n addr -- )
2        TOS> ,>MAR
3        TOS> ,>PM
```

: (—) Colon definition. Create a new word. Usage: : <word> <...> ;

```
1    : :  ( -- )
```

; (—) Terminate a colon definition. Resume interpretation. Usage: : <word> <...> ;

```
1    : ;  ( -- )
```

>r (n —) Move *n* from data stack to return stack.

```
1    : >r  ( n -- )
2        TOS> ,>RS
```

@ (*addr* — *n*) Put on TOS the value *n* stored at address *addr*.

```
1    : @  ( addr -- n )
2        TOS> ,>MAR
3        PM> ,>TOS
```

(create)

```
1    : (create)  ( -- )
```

, Allot one cell of dictionary space. Store n at that cell's address.

```
1    : ,  (n -- )
```

blk User variable. Located at address `addr` is the number of the block being interpreted.

```
1    : blk  ( -- addr ) ( User variable. Located at address addr )
2                       ( is the number )
3                       ( of the block being interpreted )
```

here

```
1    : here  ( -- addr ) ( Put on TOS the address of the next free )
2                        ( cell in the dictionary )
```

interpret (—) Interpret the input string.

```
1    : interpret  ( -- )
```

next (—) Compile the machine instructions which simulate the Forth machine's `next` function. Must be used at the exit point of each code or ;code definition.

```
1    : next  ( -- )
```

r> (— *n*) Move *n* from return stack to data stack.

```
1    : r>  ( ( -- n )
2        RS> ,>TOS
```

A

Template Appendix

A.1 One section

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

A.2 Another Section

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Bibliography

Works are listed in order of first citation.

[1] Jr. Philip J. Koopman. *Stack Computers the New Wave*. Ellis Horwood Limited, 1989 (cited on page 3).

[2] Leo Brodie. *Starting FORTH*. Prentice-Hall, Inc, 1981 (cited on page 13).

[3] R. G. Loeliger. *Threaded Interpretive Languages*. BYTE Publications, 1981 (cited on page 13).

[4] Mr. Example McZampleface. *The Great Book of Examples*. Place Holder Publishing, 1999 (cited on page 23).

Title: Implementing the Forth Inner Interpreter in High Level Forth Date: 2016-09 URL: http://www.euroforth.org/ef16/papers/ho
Abstract: This document defines a Forth threaded code (inner) interpreter written entirely in high level standard Forth. For this it defines a specific threaded code structure of colon definitions, a compiler from high level Forth to this threaded code and a corresponding inner interpreter to execute it. This inner interpreter can run in a stepwise way and so gives the surrounding environment control of its execution behavior. A real time environment thus might slice the execution of threaded code in small pieces and provide an interactive command shell while still meeting its real time requirements. References: [1] A synchronous FORTH framework for hard real-time control, U. Hoffmann and A. Read, euroForth 2016 [2] Threaded Interpretive Languages: Their Design and Implementation, R. G. Loeliger, McGraw Hill, 1981 Notes: - This paper describes a Forth inner interpreter that runs inside a Forth system. This allows the Forth system to control the execution of the 'virtual' interpreter. This is relevant to Real-Time Operating Systems (RTOS). - This paper is not relevant to the MacroProcessor Forth system

Title: Fig-Forth internals Date: 2021-09 URL: https://www.jimbrooks.org/archive/programming/forth/FIG-FORTH_internals.php Abstract: My notes from studying and modifying 8086 assembly code of FIG-FORTH. Describes how FIG-FORTH interpreter executes internally, how interpreting and compiling are virtually the same operation, ingenious solutions used to assemble FORTH interpreter. Also describes modifying FIG-FORTH to have direct-threaded code (DTC). Notes: - These notes are likely to be useful in understanding more low-level information. - This is not a walkthrough of the Forth system -"When FORTH starts, it begins executing machine-code of physical CPU. FORTH registers IP SP RP of FORTH virtual processor are initialized. System words COLD WARM ABORT are executed which completes initialization. ABORT executes QUIT" - "During execution of COLD which starts a FIG-FORTH system, QUIT creates command prompt. QUIT contains words QUERY INTERPRET. QUERY accepts commands, INTERPRET interprets or compiles them. QUIT emits OK message after INTERPRET returns."

Notation

The next list describes several symbols that will be later used within the body of the document.

c	Speed of light in a vacuum inertial frame
h	Planck constant
CFA	See Code Field Address
LFA	See Link Field Address
NFA	See Name Field Address
PFA	See Parameter Field Address

Glossary

code field
: code field description.

code pointer
: The address of the code. Stored in a word's code field. The Code Field Address is the addresss of the code field, not the address contained within the code field..

colon word
: Colon-defined words have parameter fields that contain pointers to code fields in other words. These pointers are Code Field Addresses..

compile word
: description.

CPU
: Central Processor Unit.

data field
: data see parameterField.

dictionary
: dictionary description.

dictionary entry
: is a set of data describing a Forth word.

immediate word
: A word that has the IMMEDIATE bit set in its dictionary header. Is executed during interpretation. Is executed during compilation rather than compiled into a definition. Is executed during compilation and interpretation.

link field
: link field description.

link field address
: link field address description.

Machine-code word
: A word that is directly executed by the CPU. Does not call other Forth words. May exit via a jump instruction to the address of NEXT or by including the contents of NEXT inline, thereby avoiding a jump instruction. Examples include the words NEXT EXECUTE BRANCH and EXIT ..

name field
: name field description.

name field address
: name field address description.

parameter field
: the address of a word's parameter field. The PFA points to the parameter field..

parameter field address
: parameter field address description.

threaded- word
: A word that contains other Forth words. May also contain machine-code words. Exits by calling the Forth word ; ..

Dictionary ==========

; "Semi-colon" Terminates a definition created by : .

1. Exits colon definition mode 2. Addes the definition to the dictionary 3. Enters interpretation mode

: ; (–) (definition goes here) ;

NEXT "Next" Terminates a code-word and jumps to the next instruction.

1. Fetch the Code Field Address (CFA), which is stored in a cell, which is pointed to by the Forth Instruction Pointer (IP) 2. Increment the Forth IP, which will then point to the proceeding cell, which contains the next Forth instruction 3. Jump to the fetched CFA

: NEXT (–) RS> ,>PC ,R–

Notes: - What is a cell? I think this is address of the code field of the proceeding Forth word. The IP always points to the next CFA to be executed. - Does the cell located at the CFA hold a machine instruction or is there a Forth dictionary word there? I think the contents of the cell pointed to by the CFA in the IP is always a machine instruction. I think the word INTERPRET handles the task of looking through the dictionary and determining the CFA of that word. The CFA should be the first cell of a word's code field, which can be one or more cells.

Alphabetical Index