# Moving Forth
# Part 1 – 8

## A Minimal TTL Processor

## B.Y.O. Assembler

**Selected Articles**

**by Brad Rodriguez**

**Shortened Version – just Pictures**

**The current Forth Bookshelf can be found at**
https://www.amazon.co.uk/Juergen-Pintaske/e/B00N8HVEZM

**1 Charles Moore -** Forth - The Early Years: Background information about the beginnings of this Computer Language

**2 Charles Moore -** Programming A Problem Oriented Language: Forth - how the internals work

**3 Leo Brodie -** Starting Forth

**4 Leo Wong – Juergen Pintaske – Stephen Pelc** FORTH LITE TUTORIAL: Code tested with free MPE VFX Forth, SwiftForth and Gforth or else

**5 Stephen Pelc -** Programming Forth: Version July 2016

**6 Tim Hentlass -** Real Time Forth

**7 Chen-Hanson Ting -** Footsteps In An Empty Valley issue 3

**8 Chen-Hanson Ting -** Zen and the Forth Language: EFORTH for the MSP430G2552 from Texas Instruments

**9 Chen-Hanson Ting -** eForth and Zen - 3rd Edition 2017: with 32-bit 86eForth v5.2 for Visual Studio 2015

**10 Chen-Hanson Ting -** eForth Overview

**11 Chen-Hanson Ting -** FIG-Forth Manual Document /Test in 1802 IP

**12 Chen-Hanson Ting -** EP32 RISC Processor IP: Description and Implementation into FPGA – ASIC tested by NASA

**13 Chen-Hanson Ting –** Irriducible Complexity

**14 Burkhard Kainka -** Learning Programming with MyCo: Learning Programming easily - independent of a PC (Forth code to follow soon)

**15 Burkhard Kainka -** BBC Micro:bit: Tests Tricks Secrets Code, Additional MicroBit information when running the Mecrisp Package

**16 Juergen Pintaske – A START WITH FORTH -** Bits to Bites Collection – 12 Words to start, then 35 Words, Javascript Forth on the Web, more

**17 Burkhard Kainka – Thomas Baum** – Web Programming ATYTINY13

**18 Brad Rodriguez -** Moving Forth / TTL CPU / B.Y.O. Assembler v14

# Introduction

In 2018 we celebrate ***50 Years of Forth***. And when I looked for a new eBook project, I realized that one area is not very well covered:

*How do the Forth internals work?*

*How can you build a Minimal Processor executing Forth directly?*

*How do you write an Assembler in Forth?*

When I looked around for some documentation, I remembered Brad's excellent series of articles again. They seemed to fit very well together.

I contacted Brad and asked for permission to publish them and add them to the **Forth Bookshelf** I had started 5 years ago.

He liked the idea, so I started editing and formatting. I did not change any of the original material. The only part I added was an appendix, where I re-did some of the pictures in Excel, so I could understand them better.

And as in many cases, additional material will be made available on the www.Forth-eV.de  Wiki, we will start with the appendix added there to download and edit locally, and add more that we might come up with.

There are many references and links as part of the articles – a good source to search for additional information for interested parties.

This is not new material – actually exactly 25 years old (half the Forth age) - but is still the best material I could find for this project to understand the Forth Internals better - covering Software and Hardware. And more importantly: the Forth structure is stable and has not changed that much. Probably one reason why no new material has been done.

I have to thank Brad Rodriguez for the copyright to publish this documentation.

Enjoy reading and any feedback please send to epldfpga@aol.com.

There is another eBook which covers similar aspects: Chen-Hanson Ting's eForth Overview at https://www.amazon.co.uk/eForth-Overview-C-H-Ting-ebook/dp/B01LR47JME/ref=asap_bc?ie=UTF8

**Juergen Pintaske, ExMark, 4 April  2018**

# Contents:

**eBook brad_rodriguez-Forth_v14**

# Pictures in this eBook:

## Chapter 1

## Chapter 3

## Chapter 6

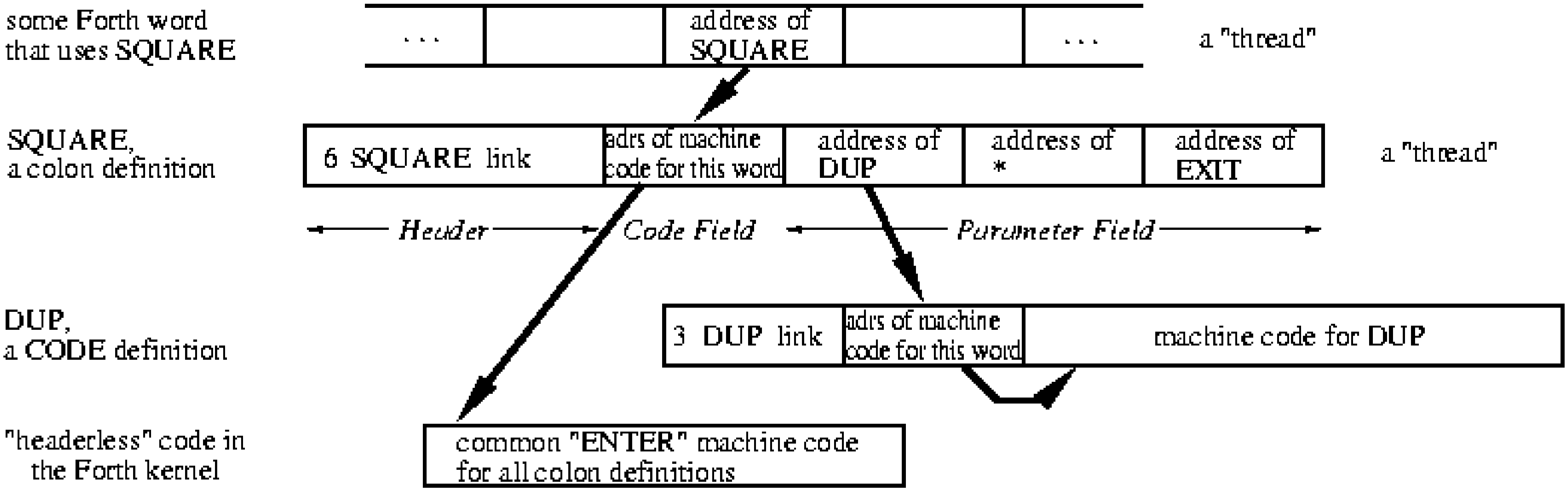
## Chapter 7

## A Minimal TTL Processor

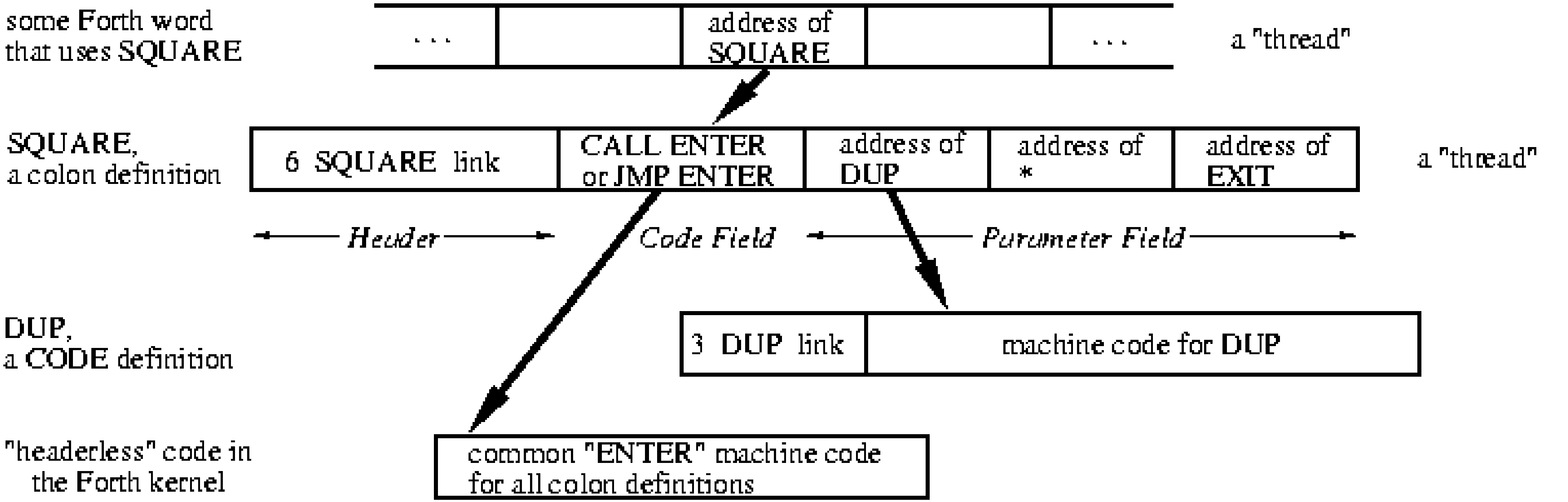
# MOVING FORTH by Brad Rodriguez

## Part 1: Design Decisions in the Forth Kernel

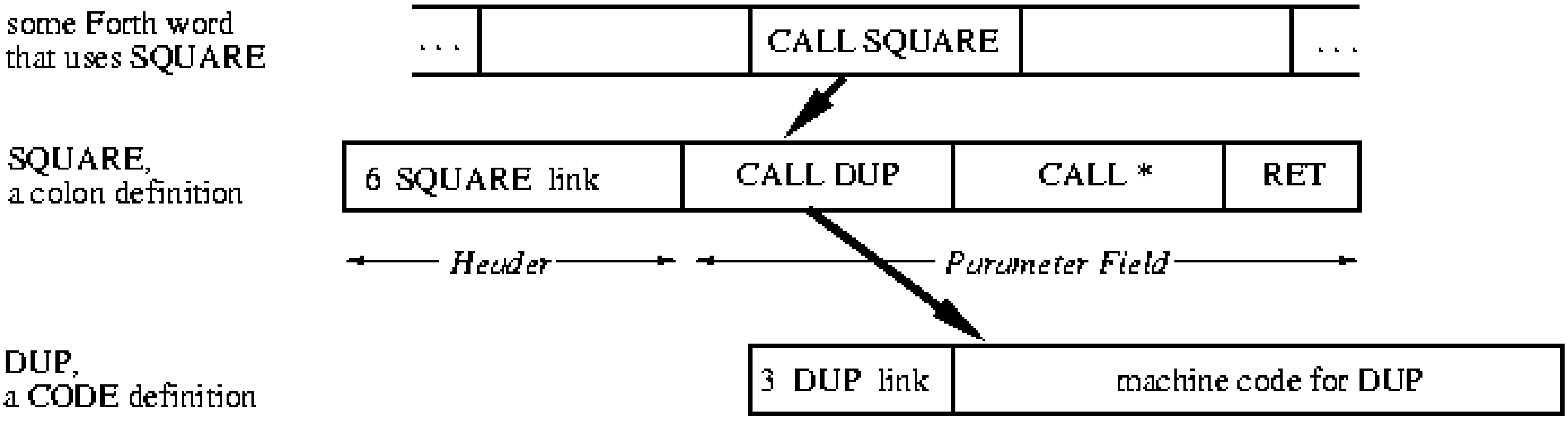
# FIGURE 1. INDIRECT THREADED CODE



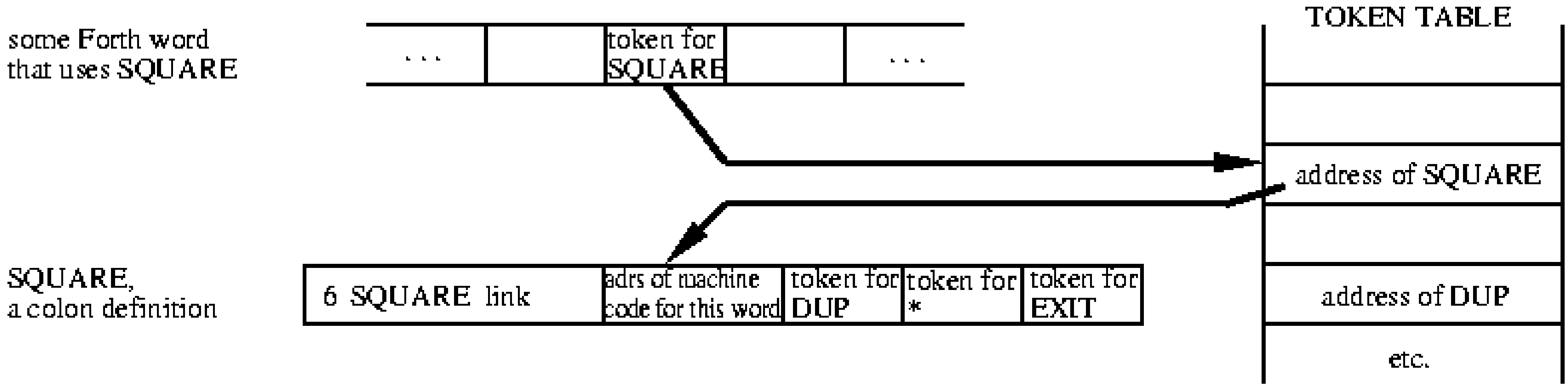
# FIGURE 2. DIRECT THREADED CODE



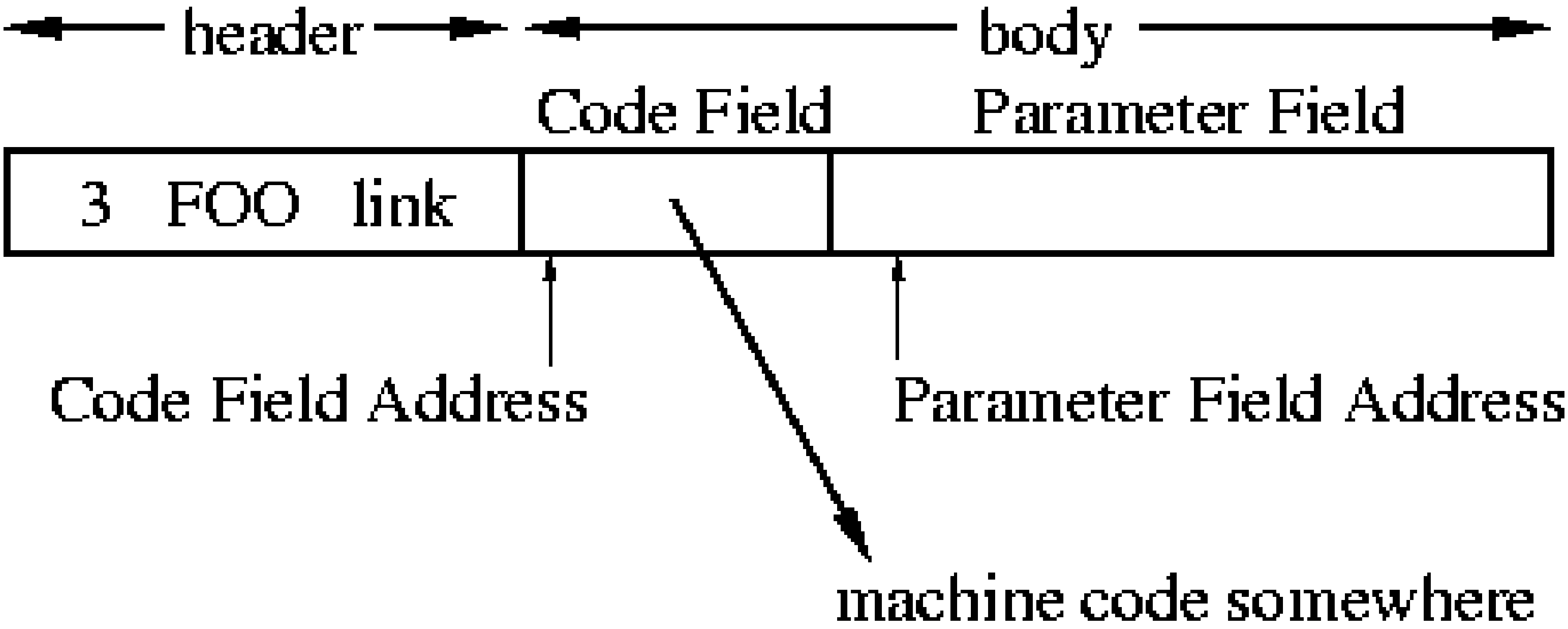
## FIGURE 3. SUBROUTINE THREADED CODE



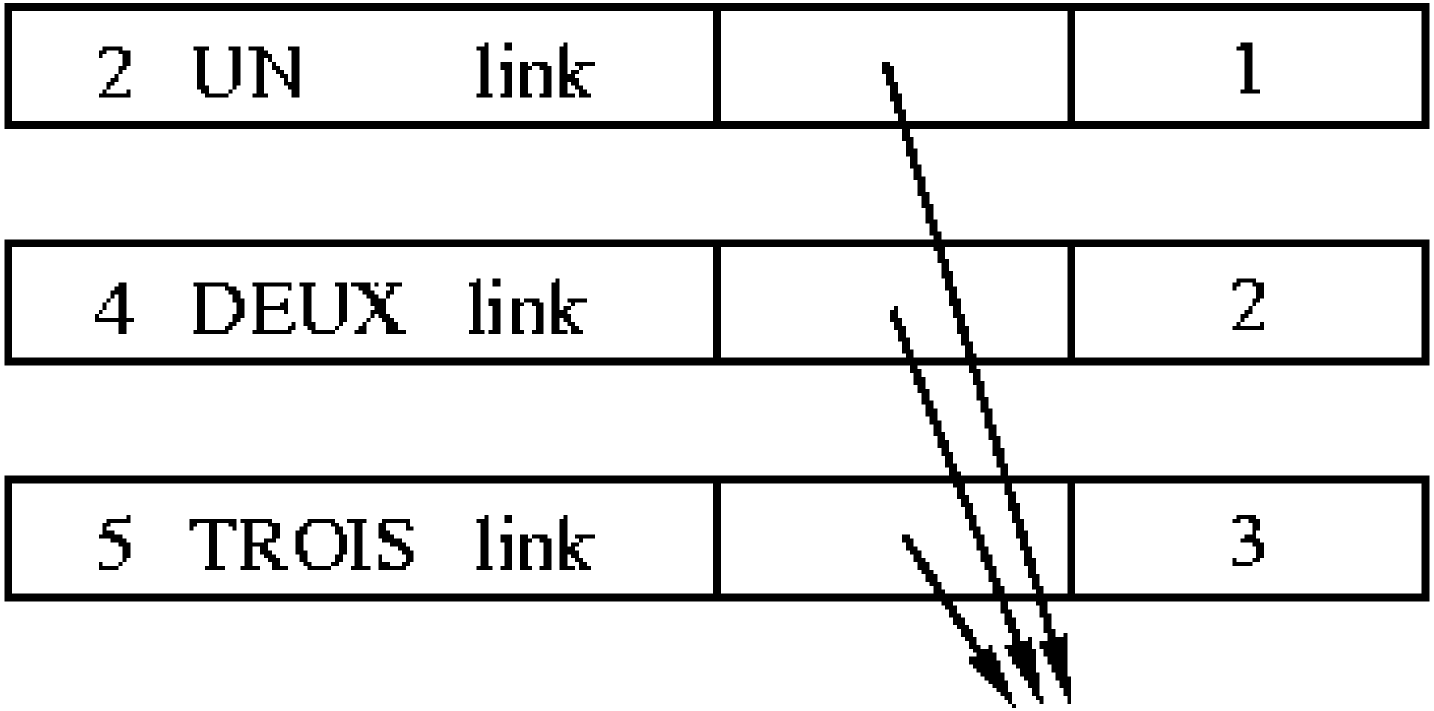
# FIGURE 4. TOKEN THREADED CODE



# FIGURE 1. AN ITC FORTH WORD



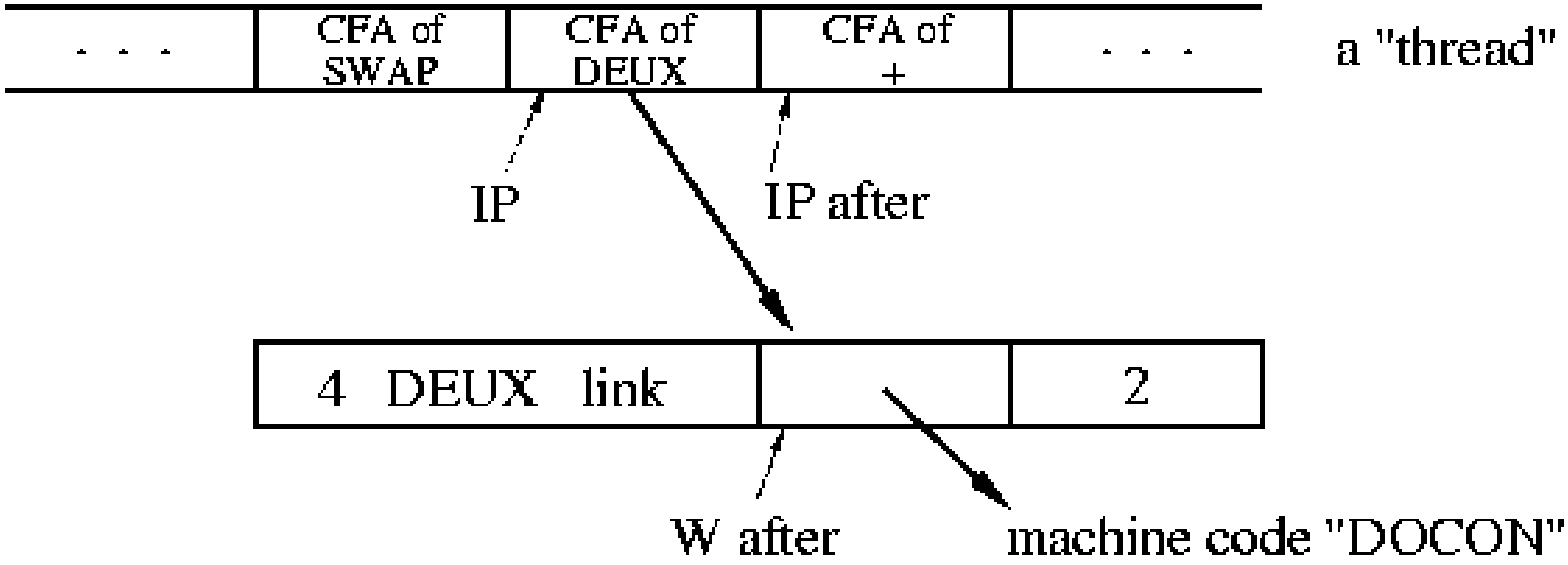
# FIGURE 2. THREE CONSTANTS



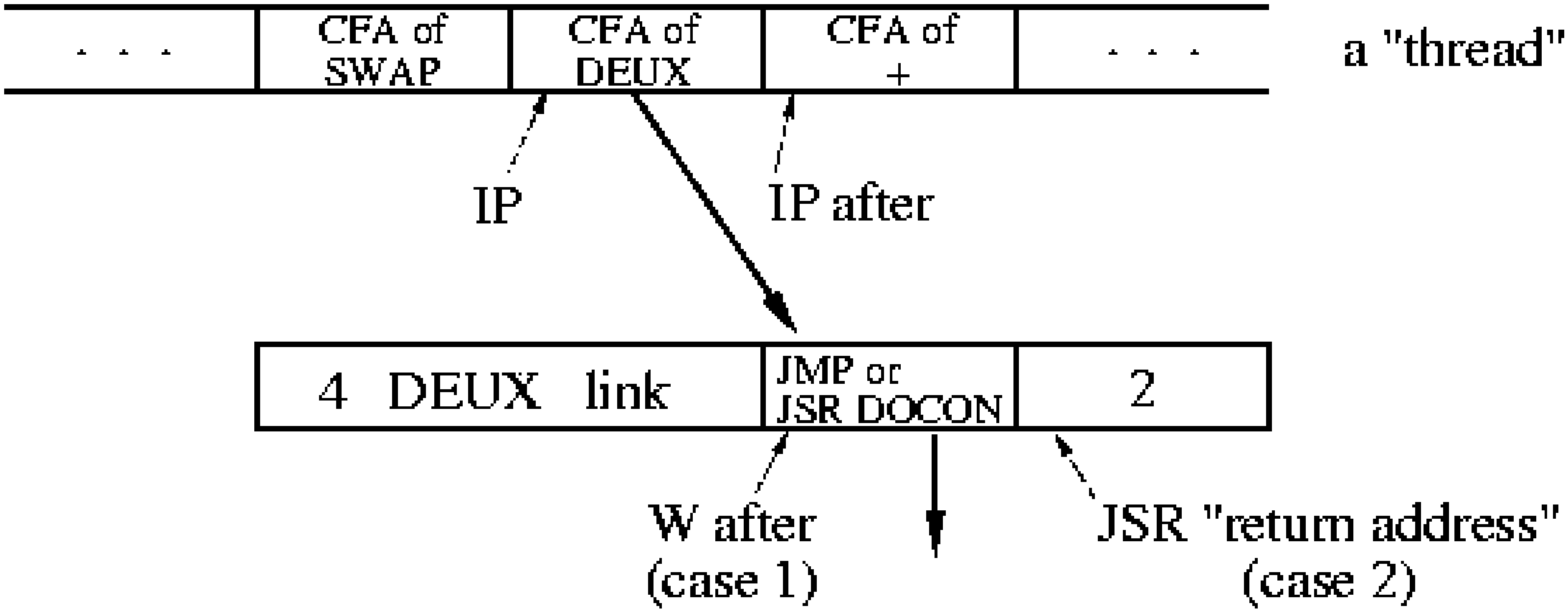
| | | |
|---|---|---|
| 2 UN link | | 1 |
| 4 DEUX link | | 2 |
| 5 TROIS link | | 3 |

code to push the *contents* of the Parameter Field onto the stack

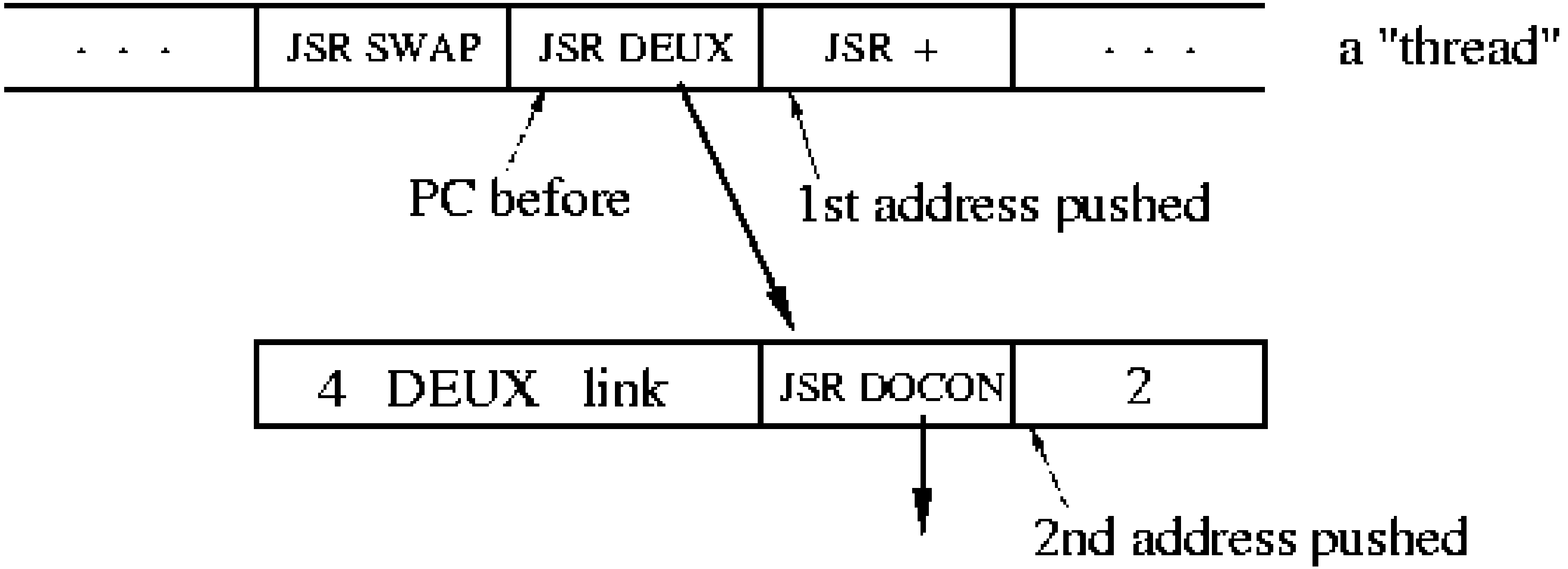
# FIGURE 3. ITC BEFORE AND AFTER "NEXT"



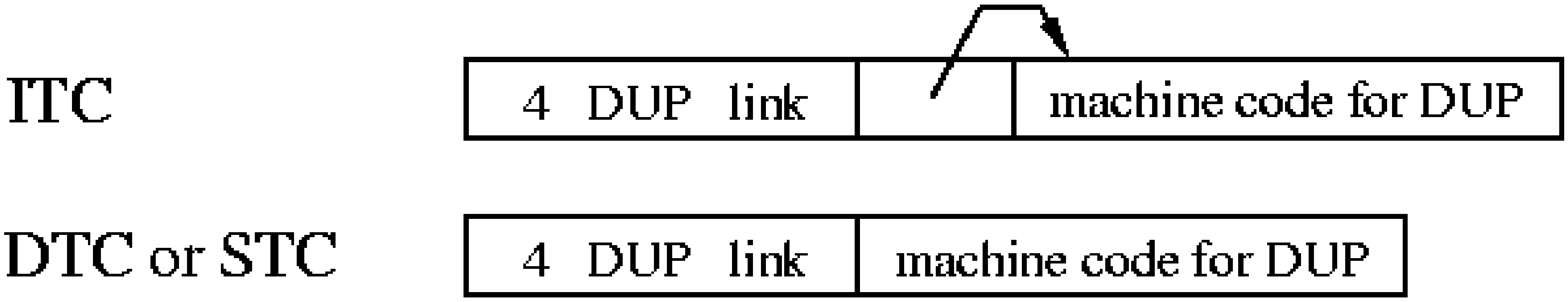
# FIGURE 4. DTC BEFORE AND AFTER "NEXT"



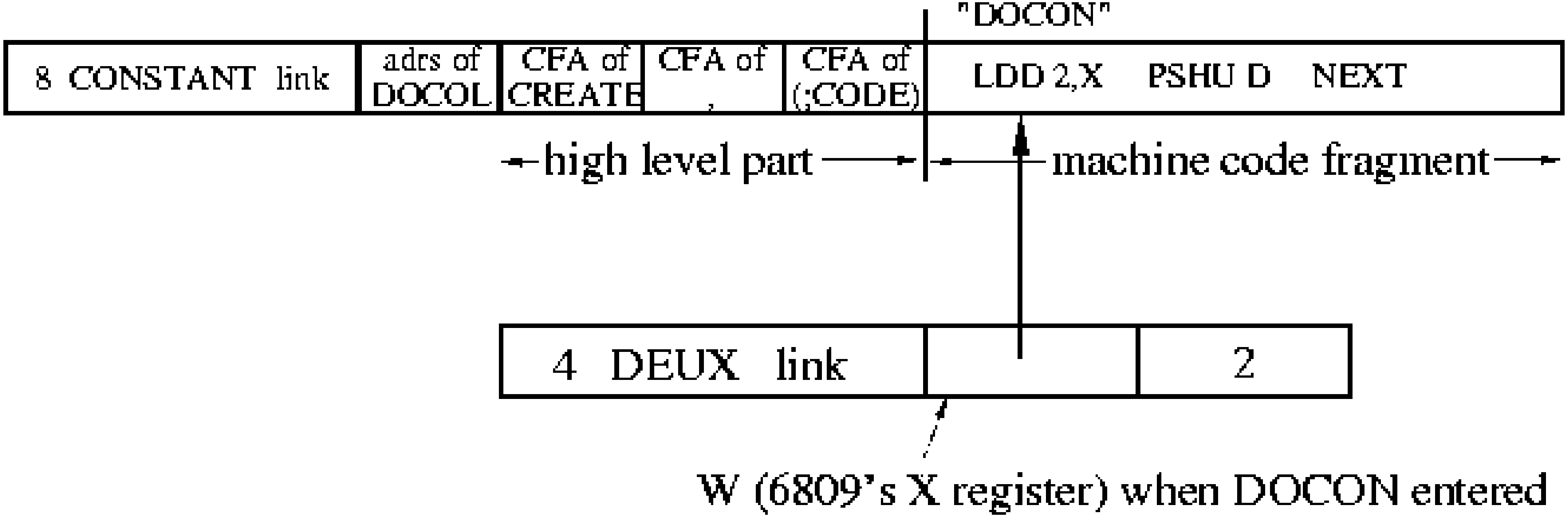
# FIGURE 5. SUBROUTINE THREADED CODE



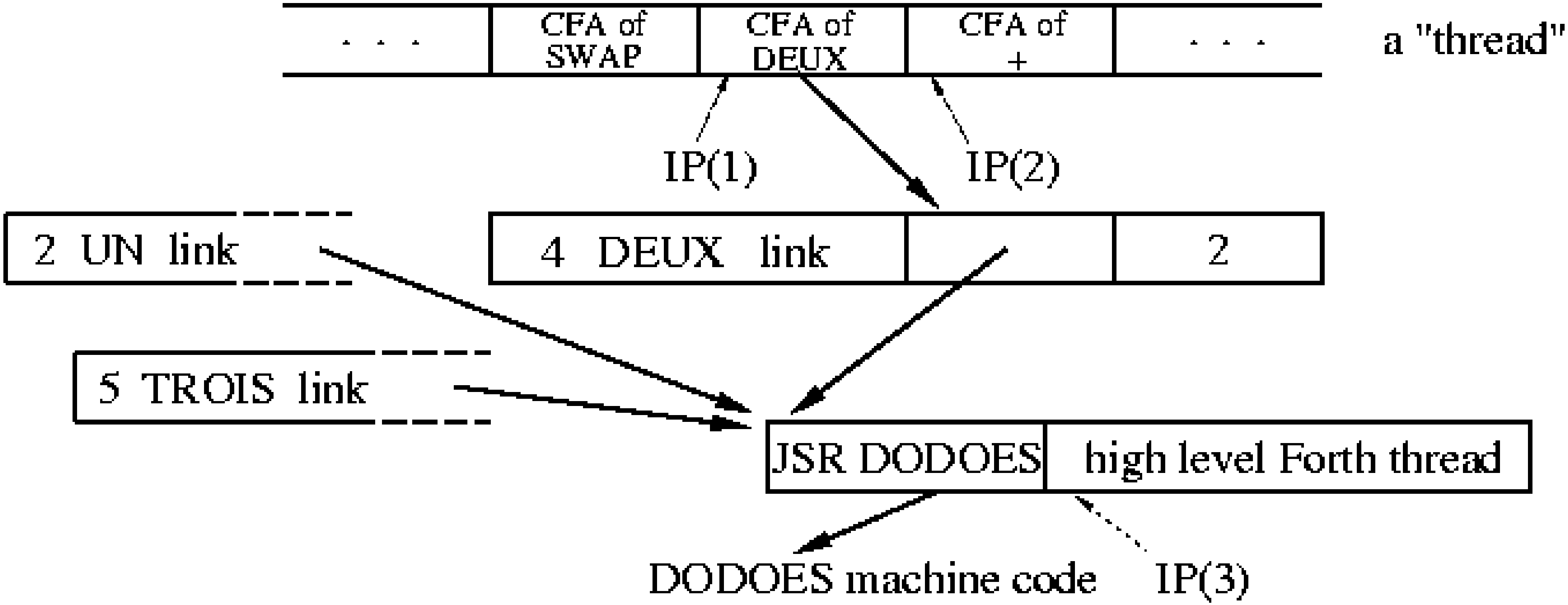
# FIGURE 6. CODE WORDS



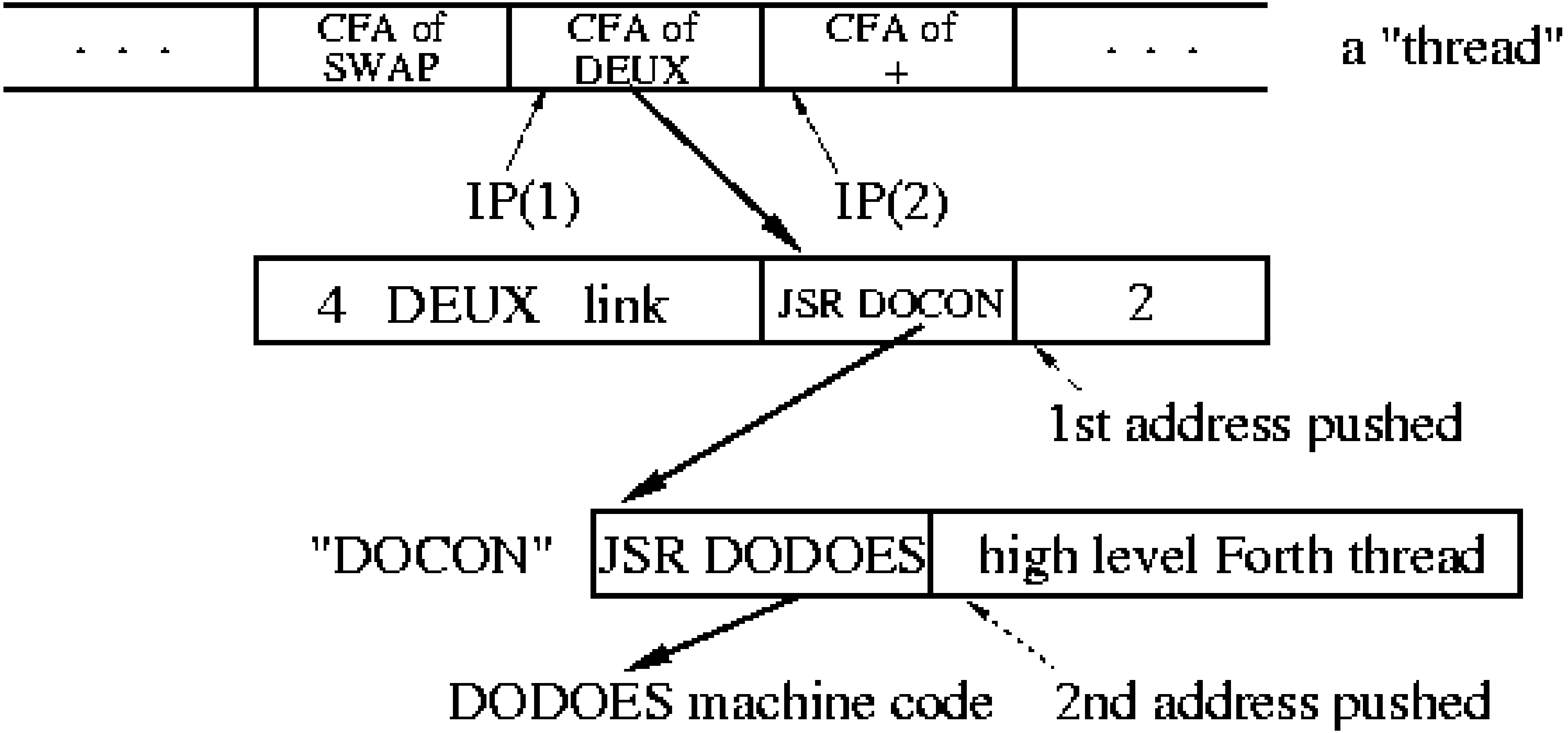
# FIGURE 7. ITC ;CODE



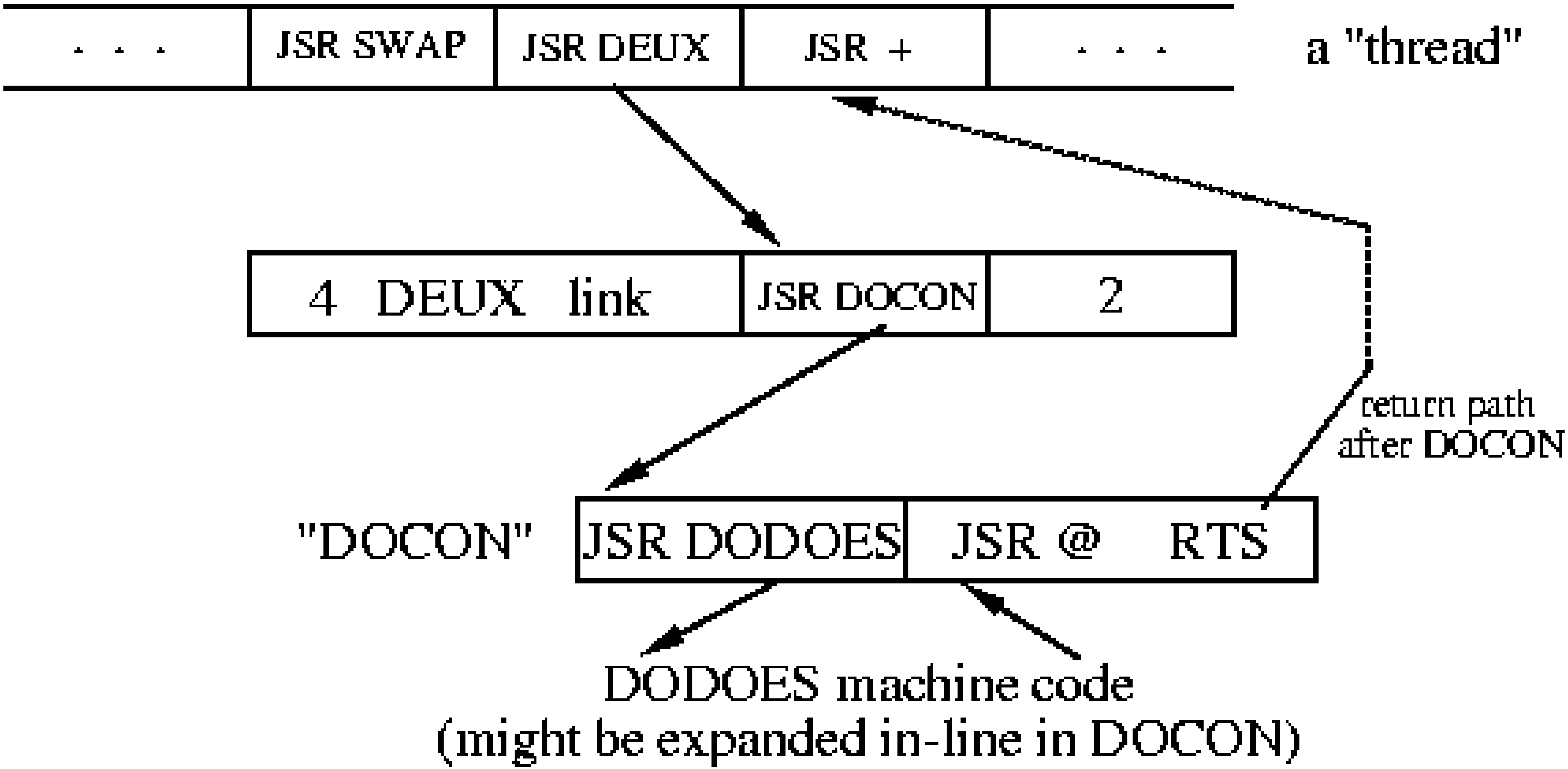
# FIGURE 8. ITC DODOES



# FIGURE 9. DTC DODOES



# FIGURE 10. STC DODOES



. . .

JSR SWAP

JSR DEUX

JSR +

. . .

a "thread"

4 DEUX link

JSR DOCON

2

return path after DOCON

"DOCON"

JSR DODOES

JSR @ RTS

DODOES machine code (might be expanded in-line in DOCON)

# FIGURE 11. ITC DOES>

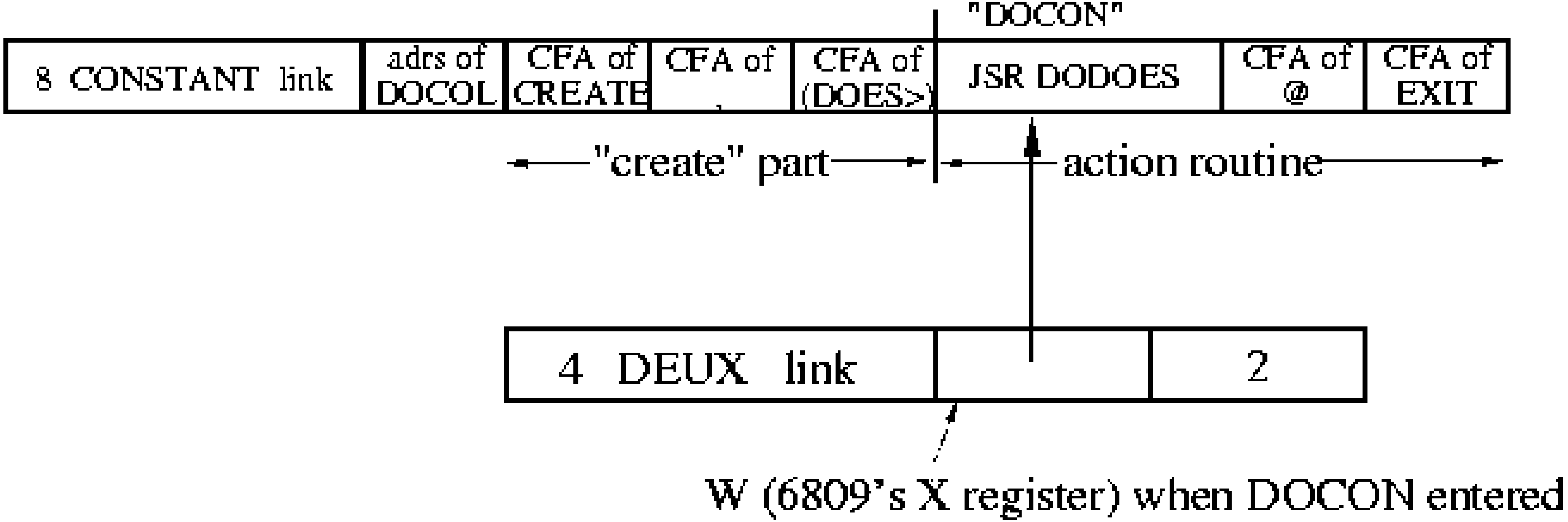
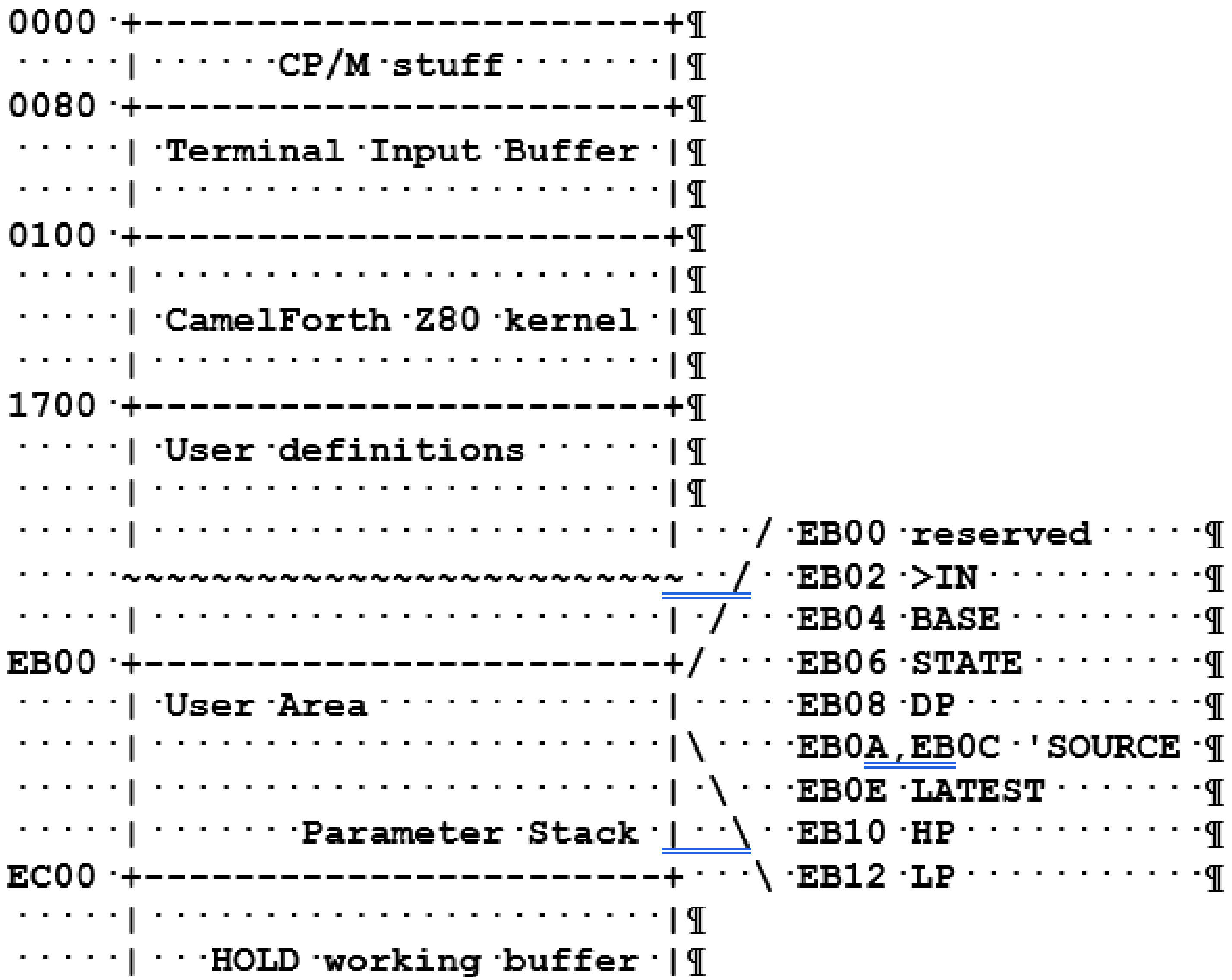
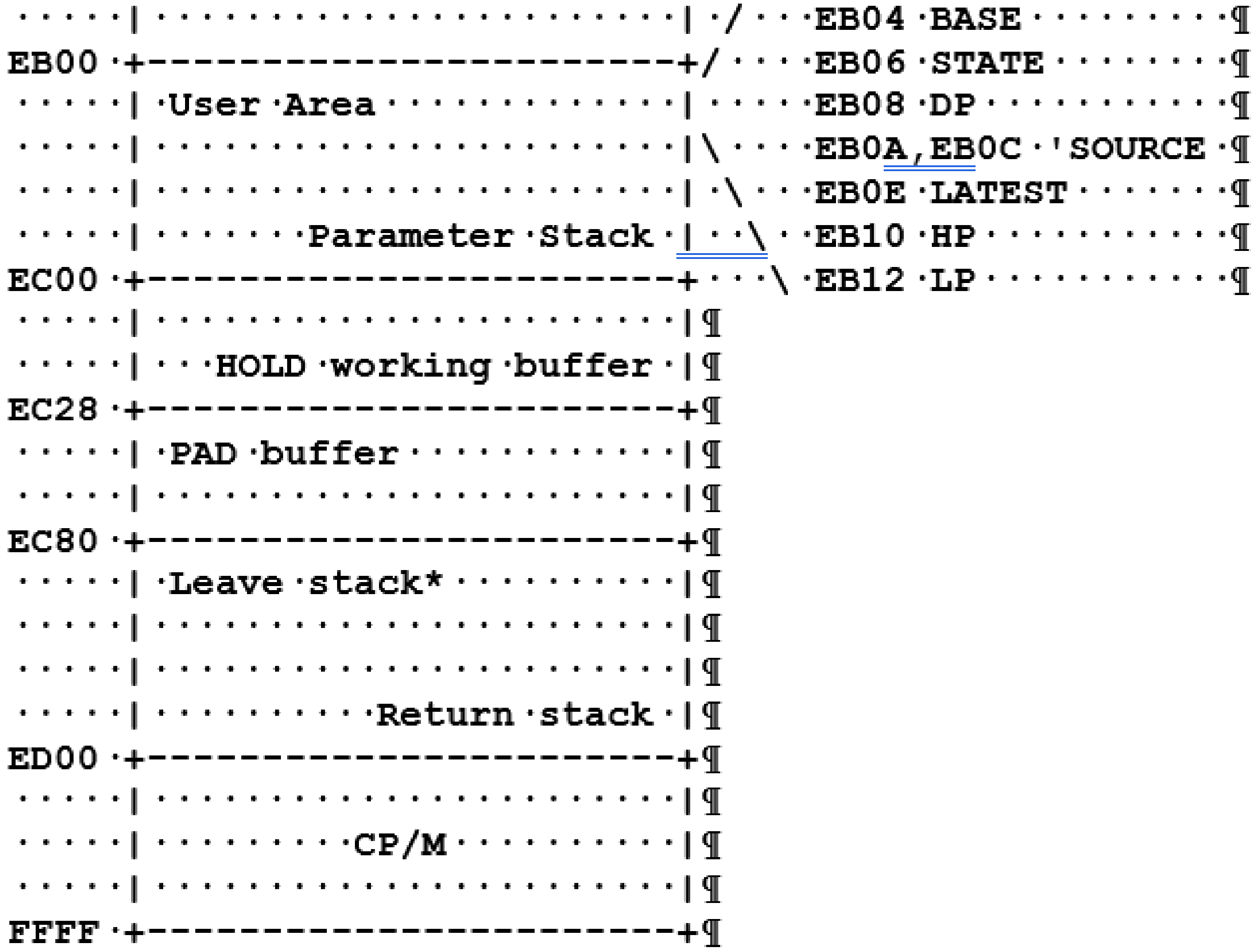


**FIGURE 1. Z80 CP/M CAMELFORTH MEMORY MAP**

assuming CP/M BDOS starts at ED00 hex.



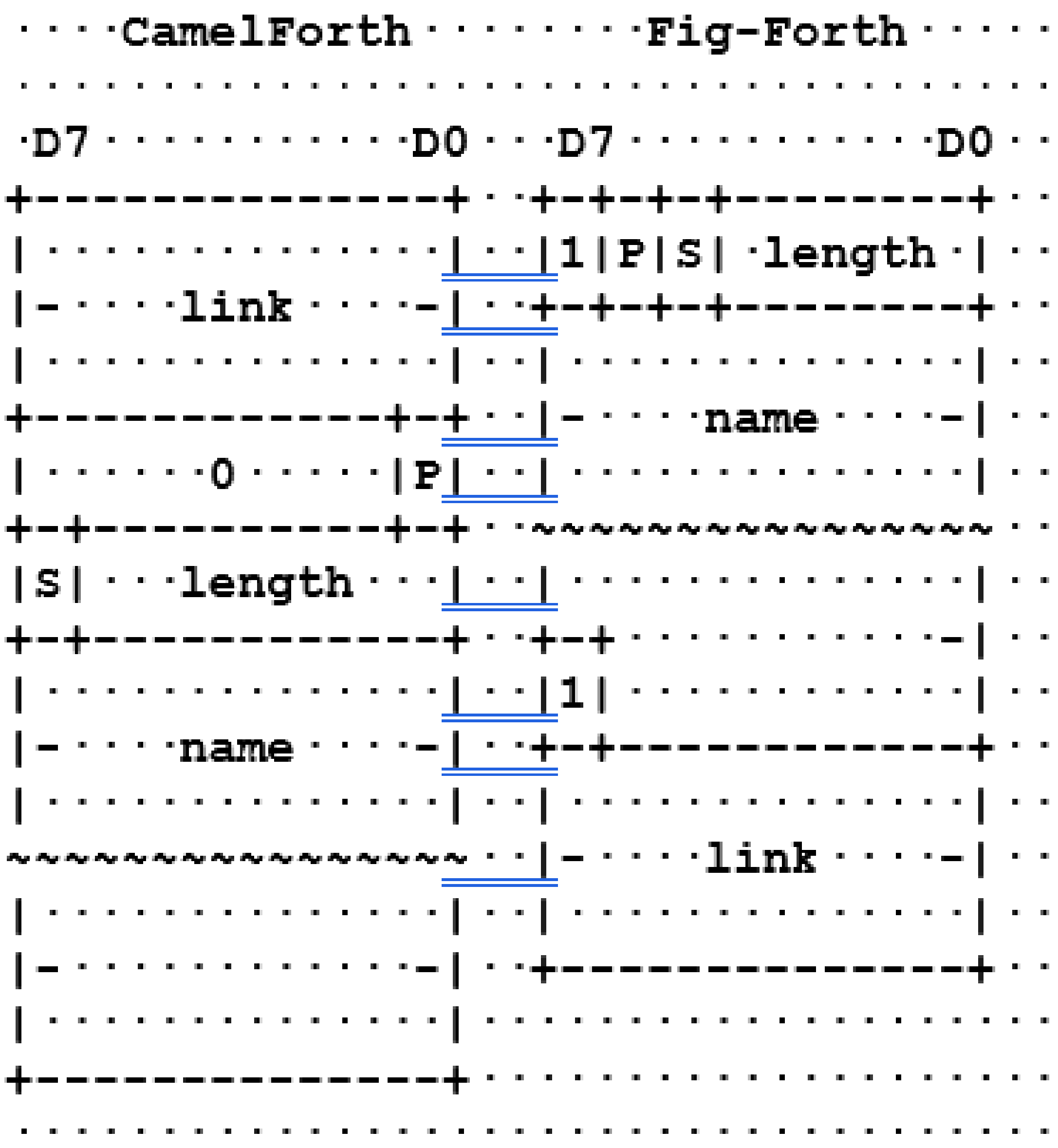
```
0000 +-----------------------+
     |      CP/M stuff       |
0080 +-----------------------+
     | Terminal Input Buffer |
     |                       |
0100 +-----------------------+
     |                       |
     | CamelForth Z80 kernel |
     |                       |
1700 +-----------------------+
     | User definitions      |
     |                       |
     |                       |   / EB00 reserved
     ~~~~~~~~~~~~~~~~~~~~~~~~~~  /  EB02 >IN
     |                       | /   EB04 BASE
EB00 +-----------------------+/    EB06 STATE
     | User Area             |     EB08 DP
     |                       |\    EB0A,EB0C 'SOURCE
     |                       | \   EB0E LATEST
     |       Parameter Stack |  \  EB10 HP
EC00 +-----------------------+   \ EB12 LP
     |                       |
     |   HOLD working buffer |
```



```
     |                       | /   EB04 BASE
EB00 +-----------------------+/    EB06 STATE
     | User Area             |     EB08 DP
     |                       |\    EB0A,EB0C  'SOURCE
     |                       | \   EB0E LATEST
     |       Parameter Stack |  \  EB10 HP
EC00 +-----------------------+   \ EB12 LP
     |                       |
     |   HOLD working buffer |
EC28 +-----------------------+
     | PAD buffer            |
     |                       |
EC80 +-----------------------+
     | Leave stack*          |
     |                       |
     |                       |
     |          Return stack |
ED00 +-----------------------+
     |                       |
     |         CP/M          |
     |                       |
FFFF +-----------------------+
```

* used during compilation of DO..LOOPs.

# FIGURE 2. HEADER STRUCTURES



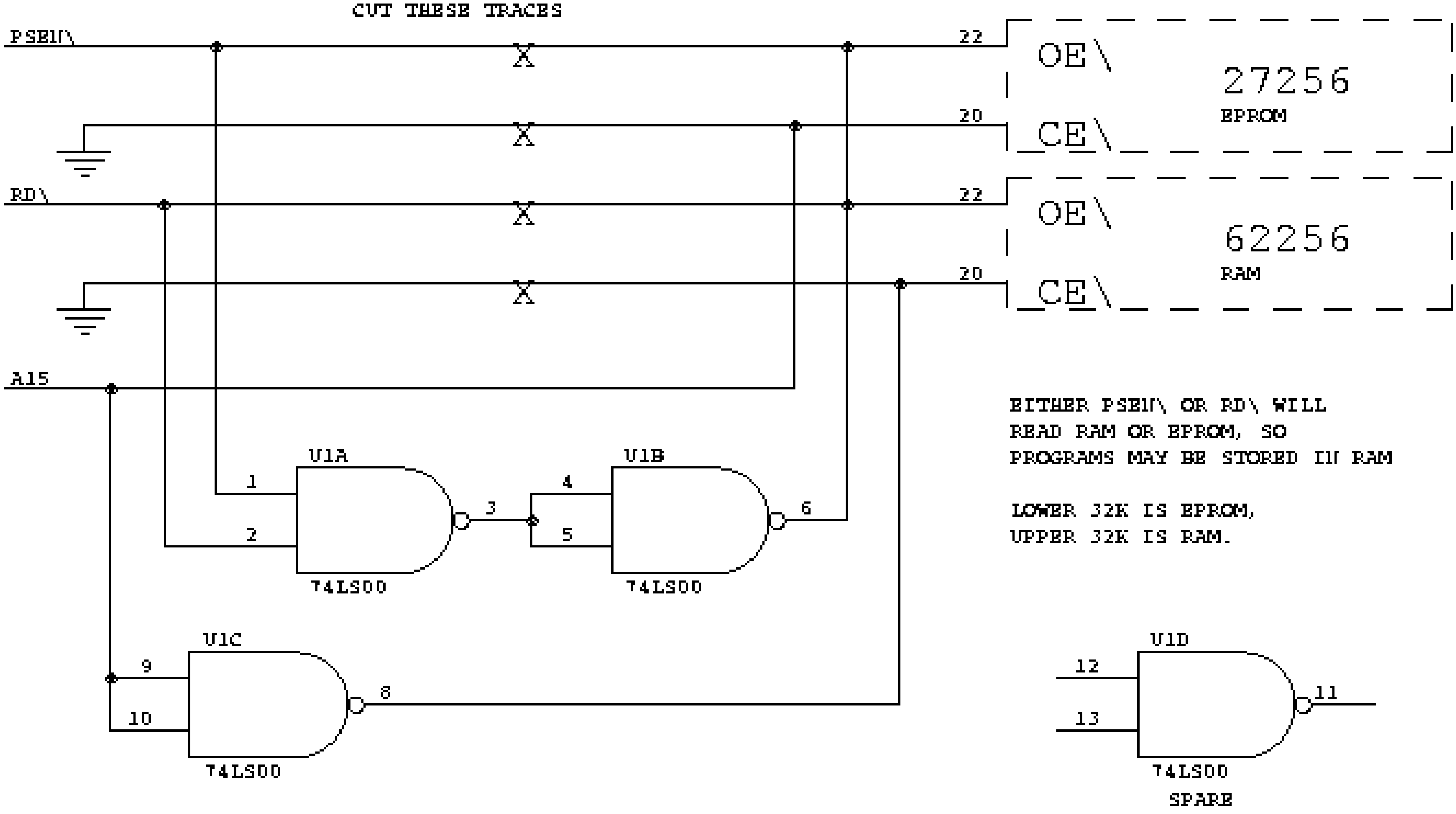
```
    CamelForth            Fig-Forth

 D7           D0     D7           D0
+---------------+  +-+-+-+---------+
|               |  |1|P|S| length  |
|-    link     -|  +-+-+-+---------+
|               |  |               |
+-------------+-+  |-    name     -|
|      0      |P|  |               |
+-+-----------+-+  ~~~~~~~~~~~~~~~~~
|S|   length    |  |               |
+-+-------------+  +-+            -|
|               |  |1|             |
|-    name     -|  +-+-------------+
|               |  |               |
~~~~~~~~~~~~~~~~~  |-    link     -|
|               |  |               |
|-             -|  +---------------+
|               |
+---------------+
```

```
   Pygmy Forth           F83

D7            D0  D7          D0
+--------------+  +-------------+
|              |  |             |
|-    view    -|  |-    view   -|
|              |  |             |
+--------------+  +-------------+
|              |  |             |
|-    link    -|  |-    lin    -|
|              |  |             |
+-+-+-+--------+  +-+-+-+-------+
|0|0|S| length |  |1|P|S| length|
+-+-+-+--------+  +-+-+-+-------+
|              |  |             |
|              |  |             |
|-    name    -|  |-    name   -|
|              |  |             |
~~~~~~~~~~~~~~~~  ~~~~~~~~~~~~~~~
|              |  |             |
|-            -|  +-+          -|
|              |  |1|           |
+--------------+  +-+-----------+
```

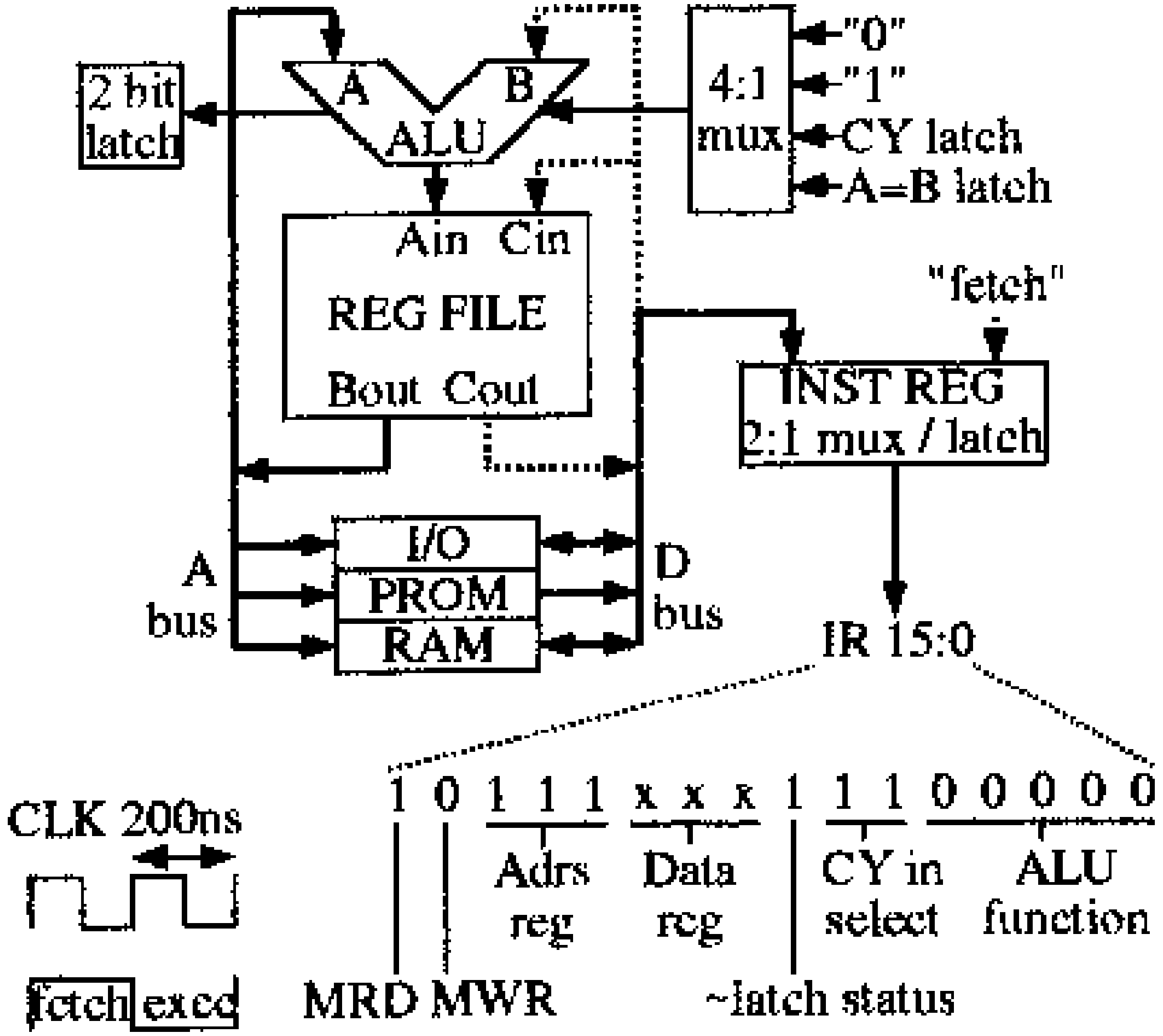
Figure 1



EITHER PSEN\ OR RD\ WILL
READ RAM OR EPROM, SO
PROGRAMS MAY BE STORED IN RAM

LOWER 32K IS EPROM,
UPPER 32K IS RAM.

Fig. 1 The Basic PISC-1a



# Fig. 2 Parallel Fetch and Execute

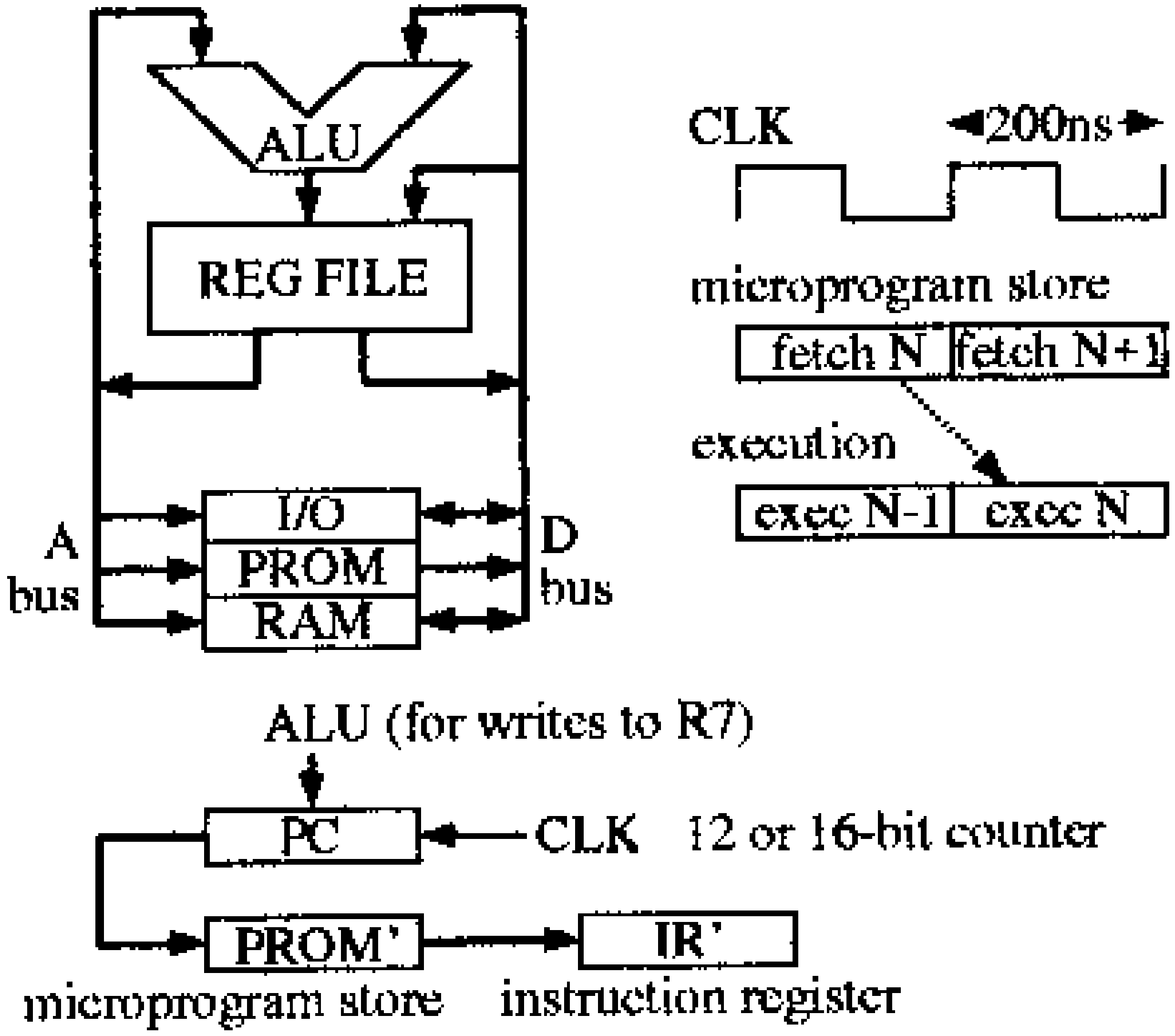
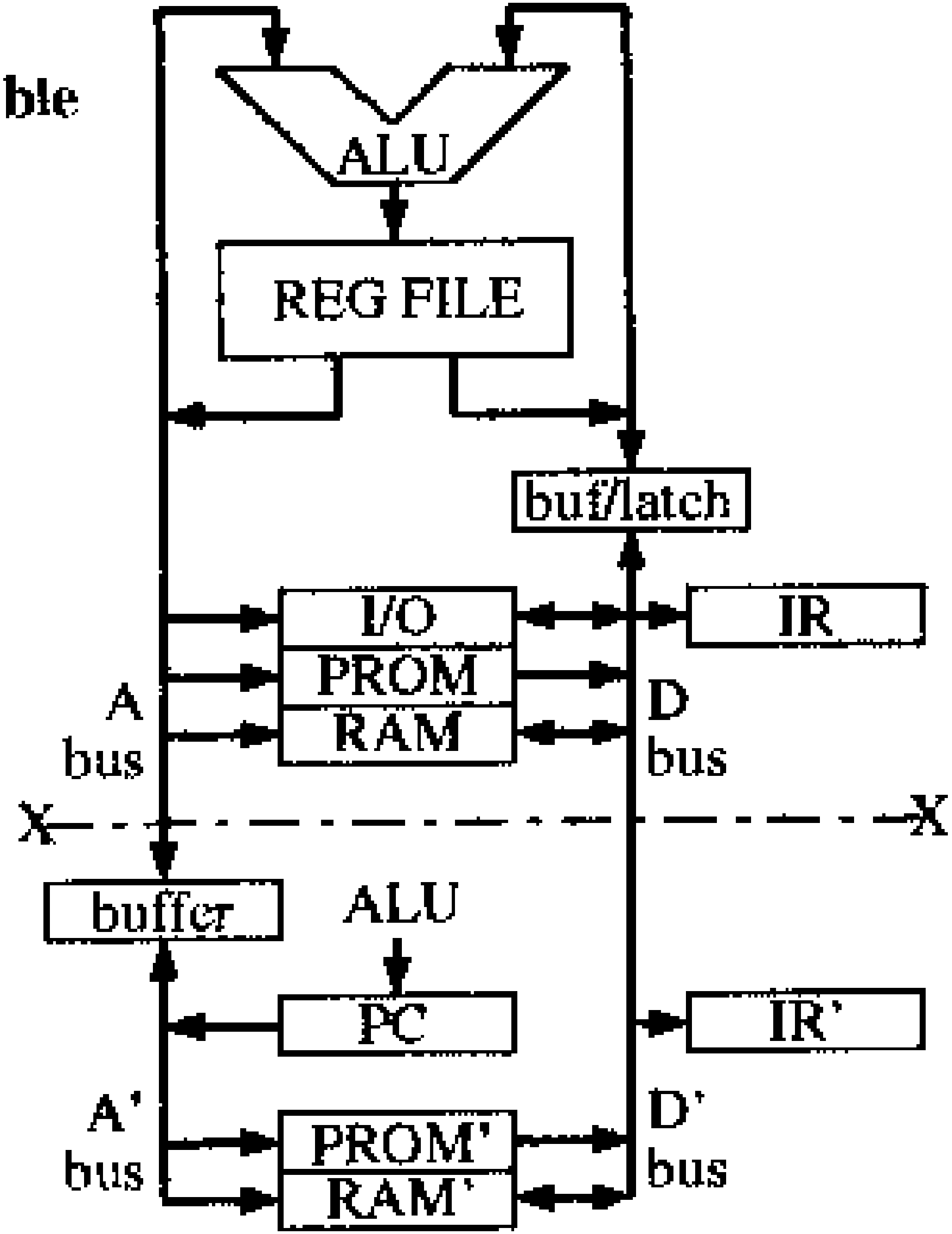
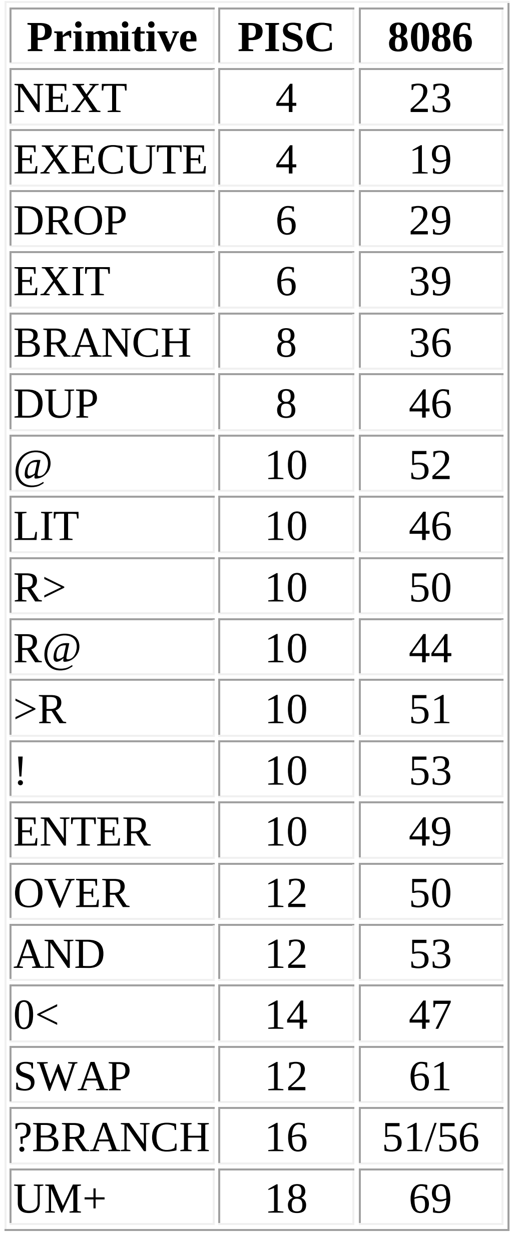


Fig. 3
The Mutable
PISC





| Primitive | PISC | 8086 |
|---|---|---|
| NEXT | 4 | 23 |
| EXECUTE | 4 | 19 |
| DROP | 6 | 29 |
| EXIT | 6 | 39 |
| BRANCH | 8 | 36 |
| DUP | 8 | 46 |
| @ | 10 | 52 |
| LIT | 10 | 46 |
| R> | 10 | 50 |
| R@ | 10 | 44 |
| >R | 10 | 51 |
| ! | 10 | 53 |
| ENTER | 10 | 49 |
| OVER | 12 | 50 |
| AND | 12 | 53 |
| 0< | 14 | 47 |
| SWAP | 12 | 61 |
| ?BRANCH | 16 | 51/56 |
| UM+ | 18 | 69 |

The schematic diagram of the PISC-1a is available if you have Adobe Acrobat or another .PDF reader. **Thanks to Derry Bryson** for converting the schematics to PDF files.

## B.Y.O. ASSEMBLER

| | -NON-INDIRECT-- | | ---INDIRECT---- | | |
|---|---|---|---|---|---|
| TYPE | MOTOROLA | FORTH | MOTOROLA | FORTH | POSTBYTE |
| no offset | ,r | r 0, | [,r] | r 0, [] | 1rri0100 |
| 5 bit offset | n,r | r n , | defaults to 8-bit | | 0rrnnnnn |
| 8 bit offset | n,r | r n , | [n,r] | r n , [] | 1rri1000 |
| 16 bit offset | n,r | r n , | [n,r] | r n , [] | 1rri1001 |
| A-reg offset | A,r | r A, | [A,r] | r A, [] | 1rri0110 |
| B-reg offset | B,r | r B, | [B,r] | r B, [] | 1rri0101 |
| D-reg offset | D,r | r D, | [D,r] | r D, [] | 1rri1011 |
| incr. by 1 | ,r+ | r ,+ | not allowed | | 1rr00000 |
| incr. by 2 | ,r++ | r ,++ | [,r++] | r ,++ [] | 1rri0001 |
| decr. by 1 | ,r- | r ,- | not allowed | | 1rr00010 |
| decr. by 2 | ,r-- | r ,-- | [,r--] | r ,-- [] | 1rri0011 |
| PC ofs. 8 bit | n,PCR | n ,PCR | [n,PCR] | n ,PCR [] | 1xxi1100 |
| PC ofs. 16 bit | n,PCR | n ,PCR | [n,PCR] | n ,PCR [] | 1xxi1101 |
| 16 bit address | not allowed | | [n] | n [] | 10011111 |

```
where n = a signed integer value,
      r = X (00), Y (01), U (10), or S (11)
      x = don't care
```

# INSTRUCTION SET

## Inherent addressing group

| MOTOROLA | FORTH | | MOTOROLA | FORTH |
|---|---|---|---|---|
| | | | | |
| ABX | ABX, | | MUL | MUL, |
| ASLA | ASLA, | | NEGA | NEGA, |
| ASLB | ASLB, | | NEGB | NEGB, |
| ASRA | ASRA, | | NOP | NOP, |
| ASRB | ASRB, | | ORA | ORA, |
| CLRA | CLRA, | | ORB | ORB, |
| CLRB | CLRB, | | ROLA | ROLA, |
| COMA | COMA, | | ROLB | ROLB, |
| COMB | COMB, | | RORA | RORA, |
| DAA | DAA, | | RORB | RORB, |
| DECA | DECA, | | RTI | RTI, |
| DECB | DECB, | | RTS | RTS, |
| INCA | INCA, | | SEX | SEX, |
| INCB | INCB, | | SWI | SWI, |
| LSLA | LSLA, | | SWI2 | SWI2, |
| LSLB | LSLB, | | SWI3 | SWI3, |
| LSRA | LSRA, | | SYNC | SYNC, |
| LSRB | LSRB, | | TSTA | TSTA, |
| | | | TSTB | TSTB, |

### Register-register group

| MOTOROLA | FORTH | | MOTOROLA | FORTH |
|---|---|---|---|---|
| | | | | |
| EXG s,d | s d EXG | | TFR s,d | s d TFR |

### Immediate-addressing-only group

| MOTOROLA | FORTH | | MOTOROLA | FORTH |
|---|---|---|---|---|
| | | | | |
| ANDCC #n | n # ANDCC, | | PSHS regs | n # PSHS, |
| CWAI #n | n # CWAI, | | PSHU regs | n # PSHU, |
| ORCC #n | n # ORCC, | | PULS regs | n # PULS, |
| | | | PULU regs | n # PULU, |

**Note:**
Motorola allows the PSH and PUL instructions to contain a register list.
The Forth assembler requires the programmer to compute the bit mask for this list and supply it as an immediate argument.

### Indexed-addressing-only group

(with example addressing modes)

| MOTOROLA | FORTH | | MOTOROLA | FORTH |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| LEAS D,U | U ,D LEAS, | | LEAX [,S++] | S ,++ [] LEAX, |
| LEAU -5,Y | Y -5 , LEAU, | | LEAY [1234] | 1234 [] LEAY, |

**General-addressing group**
(with example addressing modes)

| MOTOROLA | FORTH | | MOTOROLA | FORTH |
|---|---|---|---|---|
| ADCA #20 | 20 # ADCA, | | LDA #20 | 20 # LDA, |
| ADCB <30 | 30 <> ADCB, | | LDB <30 | 30 <> LDB, |
| ADDA 2000 | 2000 ADDA, | | LDD 2000 | 2000 LDD, |
| ADDB [1030] | 1030 [] ADDB, | | LDS [1030] | 1030 [] LDS, |
| ADDD ,S | S 0, ADDD, | | LDU ,X | X 0, LDU, |
| ANDA 23,U | U 23 , ANDA, | | LDX 23,Y | Y 23 , LDX, |
| ANDB A,X | X A, ANDB, | | LDY A,S | S A, LDY, |
| ASL  B,Y | Y B, ASL, | | LSL B,U | U B, LSL, |
| ASR  D,X | X D, ASR, | | LSR D,S | S D, LSR, |
| BITA ,S+ | S ,+ BITA, | | NEG ,X+ | X ,+ NEG, |
| BITB ,X++ | X ,++ BITB, | | ORA ,S++ | S ,++ ORA, |
| CLR  ,Y- | Y ,- CLR, | | ORB ,U- | U ,- ORB, |
| CMPA ,U-- | U ,-- CMPA, | | ROL ,Y-- | Y ,-- ROL, |
| CMPB -5,PCR | -5 ,PCR CMPB, | | ROR 12,PCR | 12 ,PCR ROR, |
| CMPD [,Y] | Y 0, [] CMPD, | | SBCA [,U] | U 0, [] SBCA, |
| CMPS [7,Y] | Y 7 , [] CMPS, | | SBCB [7,U] | U 7 , [] SBCB, |
| CMPU [A,S] | S A, [] CMPU, | | STA [A,X] | X A, [] STA, |

| | | | | |
|---|---|---|---|---|
| CMPX [B,U] | U B, [] CMPX, | | STB [B,Y] | Y B, [] STB, |
| CMPY [D,X] | X D, [] CMPY, | | STD [D,S] | S D, [] STD, |
| EORA [,Y+] | Y ,+ [] EORA, | | STS [,U+] | U ,+ [] STS, |
| EORB [,U++] | U ,++ [] EORB, | | STU [,Y++] | Y ,++ [] STU, |
| COM  [,S-] | S ,- [] COM, | | STX [,S-] | S ,- [] STX, |
| DEC [,X--] | X ,-- [] DEC, | | STY [,X--] | X ,-- [] STY, |
| INC [5,PCR] | 5 ,PCR [] INC, | | SUBA [3,PCR] | 3 ,PCR [] SUBA, |
| JMP [300] | 300 [] JMP, | | SUBB [300] | 300 [] SUBB, |
| JSR 1234 | 1234 JSR, | | SUBD 1234 | 1234 SUBD, |
| | | | TST #2 | 2 # TST, |

**Note that**, in the Forth assembler, #  signifies Immediate addressing, and <>  signifies Direct addressing.

Many instructions do not allow immediate addressing.
Refer to the Motorola data sheet.

**Branch instructions**

| MOTOROLA | FORTH | | MOTOROLA | FORTH |
|---|---|---|---|---|
| | | | | |
| BCC label | adrs BCC, | | BLT label | adrs BLT, |
| BCS label | adrs BCS, | | BMI label | adrs BMI, |
| BEQ label | adrs BEQ, | | BNE label | adrs BNE, |
| BGE label | adrs BGE, | | BPL label | adrs BPL, |
| BGT label | adrs BGT, | | BRA label | adrs BRA, |
| BHI label | adrs BHI, | | BRN label | adrs BRN, |

| | | | | |
|---|---|---|---|---|
| **BHS label** | **adrs BHS,** | | **BSR label** | **adrs BSR,** |
| **BLE label** | **adrs BLE,** | | **BVC label** | **adrs BVC,** |
| **BLO label** | **adrs BLO,** | | **BVS label** | **adrs BVS,** |
| **BLS label** | **adrs BLS,** | | | |

The branch instructions in the Forth assembler expect an absolute address.

The relative offset is computed, and the "long branch" form of the instruction is used if necessary.

# Some Pictures re-drawn

**A short note from Juergen Pintaske:**

In addition to just reformatting and publishing this material, I wanted to understand the pictures as much as possible, and as well how all of these addess lists hang together.

As it turned out, grasping it and to redraw the pictures will take more time than planned.

There was a decision that had to be taken:

Publish like the original material now
or
Delay the publication
and include the additional material.

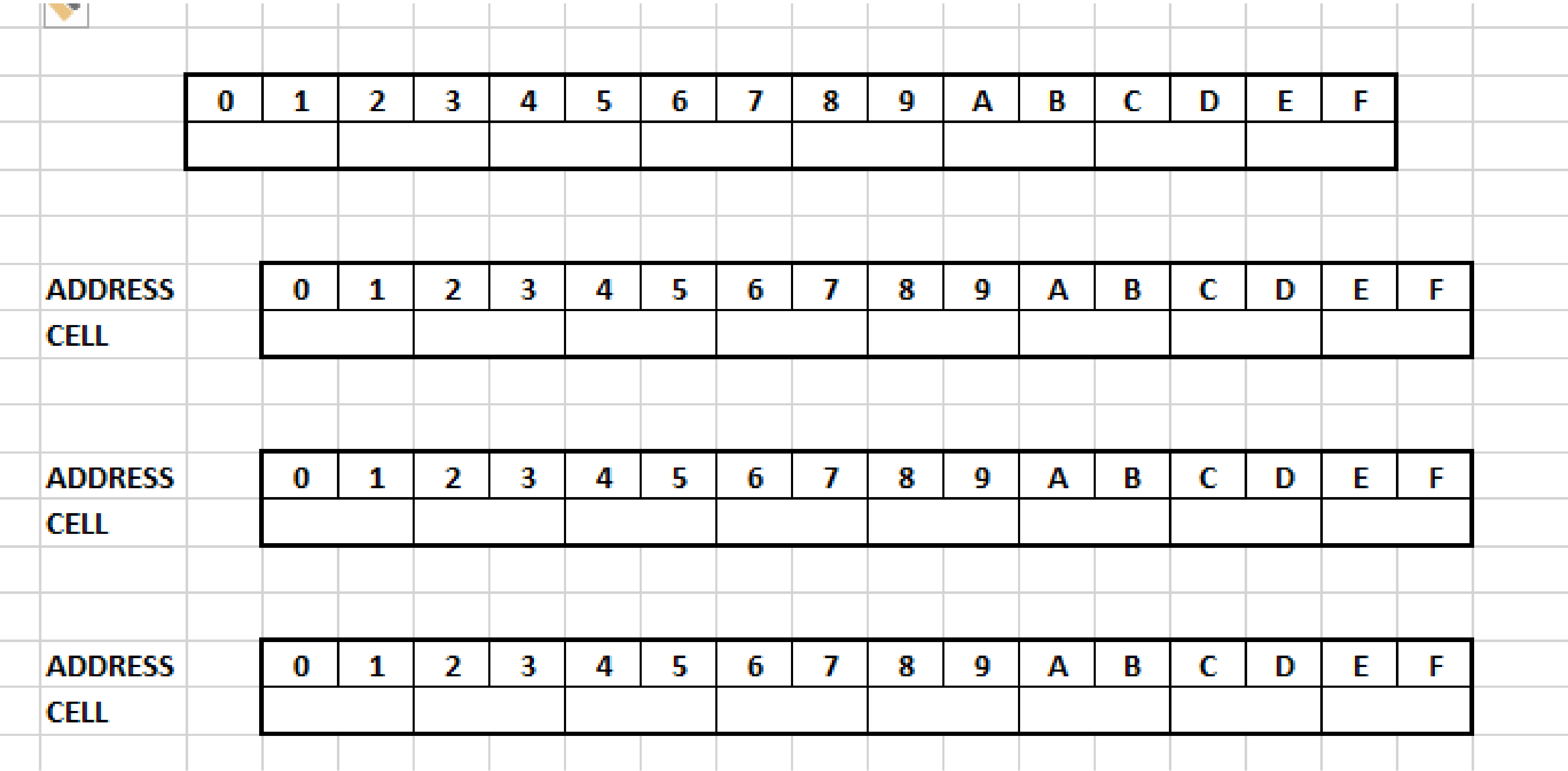
As this is eBook is planned for educational purposes,
and more material might be added anyway, I decided:

Time to get it out.

Material will be added as part of the update process.

And then with the link to a place where parts can be downloaded.

Here just one picture, showing address areas of the dictionary to be filled as you read the eBook. Here starting from boundaries, but could start anywhere and with variable length.



| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CELL | | | | | | | | | | | | | | | | |

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CELL | | | | | | | | | | | | | | | | |

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CELL | | | | | | | | | | | | | | | | |

These eBooks are a spare time activity - and job, family, dog come first.

**Juergen Pintaske – Exmark - April 2018**

# Bradford J. Rodriguez, Ph.D.

**Embedded and Distributed Control Systems** -- contract design and development of microprocessor hardware and software, for maximum performance on limited resources in HW and SW.

## SELECTED PROJECTS see it all at http://www.bradrodriguez.com/resume.htm

| | |
|---|---|
| **2013** | **ALN11 Controller:** Created a single-board ARM replacing discontinued 8051 board. New board uses Philips LPC2138 (ARM7 family); |
| **2013** | **Wireless Anemometer:** Wrote in GCC for a micropower 915 MHz wireless anemometer, MSP430G2553 MCU, and the TI CC1150 SPI transmitter, operates on accumulated wind energy. |
| **2011** | **DT12 Controller:** Designed a single-board computer as drop-in replacement for OEM board, for a low-power battery operated test instrument, from 6303 PolyForth to MC9S12X SwiftX Forth. |
| **2011- present** | **MiniPods:** Developed software for a family of 2.4 GHz devices to log work hours, travel distance, fuel dispensing, and operator inspections. Implemented in Forth using MC9S12, MC13202 transceiver. |
| **2009- 2011** | **USB OSBDM Linux drivers: S**oftware to control an Open Source BDM device from a Linux desktop via USB. OSBDM, debug for 9S08, 9S12, and 9S12X microcontrollers. |
| **2006- present** | **D6 ZPC:** Ongoing development and maintenance of the "ZPC" software on the "D6" hardware, in Forth and H8 assembler. New and improved features include expanded and multitrack MIDI recording. |
| **2008** | **MSP430 CamelForth:** Developed a Forth compiler/interpreter for the Texas Instruments MSP430 processor; less than 6K bytes of ROM, and offers "direct-to-Flash" compilation |
| **2006- 2007** | **SuperPod:** Converted the IsoPod operating software for this new board using the Freescale DSP56F8365 processor. The core was converted from C to assembler for threefold greater speed. |
| **1994** | **CamelForth:** Developed CamelForth, a portable ANS Forth compiler for Harvard and von Neumann processors, with implementations for Z80, 8051, and 6809. |
| **1993** | **68HC16 MPE Forth:** Wrote MPE Forth for the 68HC16: Kernel, multitasker, and documentation. |
| **1991** | **Z8 MPE Forth:** Wrote MPE Forth for Z8/Super8, including kernel, multitasker, and documentation. |
| **1987** | **VectorForth:** Developed software for PC-based array processing workstation. Wrote assembler and system interface for Vortex and Point-I array coprocessor boards, polyForth language support. |

| | |
|---|---|
| **1978- present** | Personal research:  Forth kernels, expert systems, assemblers, metacompiler and A "Minimal" Microprogrammed Forth Machine using Standard TTL |
| | |

# EDUCATION

| | |
|---|---|
| **1998** | Ph.D. in Electrical Engineering, McMaster University, |
| **1989** | M.S. in Computer Science, Bradley University. |
| **1980** | M.S. in Electrical Engineering, Bradley University. |
| **1977** | B.S. in Electrical Engineering, Bradley University. |

# ------------ MISCELLANEOUS ------------

**Languages:** Fluent in Forth, C, and assembler for 6502, 6809, 68HC11, 68HC12, 68HC16, 68000, 8051, 8080, 8086, ARM7, H8/S, LSI-11, MSP430, Super8, Z8, Z80. Proficient in Pascal, BASIC, FORTRAN, PHP, Tcl.

**Operating Systems:** Experienced with Unix, Linux, RT-11, CP/M, MS/DOS, polyFORTH, FreeRTOS.

**Exmark published v16 –shortened version 20 April 2018**