CHAPTER 2. THE VIRTUAL FORTH MACHINE

2.1. SEGMENT AND REGISTER ASSIGMENT

This version of LaForth is implemented for IMB-PC/XT/AT and the clones, which uses MS-DOS as the underline operating system. The Virtual Forth Machine is the mechanism which turns the 8088 CPU under DOS into a computer which execute Forth instructions. Forth must reside in the memory of the CPU somewhere, and it must use some of the registers in the CPU for its specific purposes.

LaForth needs a 64K byte segment of memory to store the dictionary, the data stack, and working space. This segment will be assigned by DOS when LaForth is loaded under DOS. The actual location of this code-data-stack segment is returns by the LaForth word DSADDR. This segment address is stored in the CS, DS, and SS segment registers.

LaForth uses an extra segment to store text from source files and the return stack. This segment is immediately above the code-data-stack segment, and its address is loaded into the ES segment register. Thus the most important register assignments of LaForth are:

IP	CS:SI	Instruction pointer
SP	SS:SP	Data stack pointer
RP	ES:DI	Return stack pointer
W	DS:AX	Current word pointer, can be used for scratch.
Scratch Registers	BX,CX,DX,BP	These registers do not have to be restored before NEXT.

2.2. THE INNER INTERPRETERS

The inner interpreters in Forth are the machine routines which executes one Forth word after another. The two most important inner interpreters are the NEXT routine which jumps from one code word to the next code word, and the NEST/UNNEST pair of routines which scan a list of addresses in a colon definition. These inner interpreters are defined as macros in LaForth:

```
NEXT      MACRO
          LODSW                       ;; Pick up next address
          JMP       AX                ;; Jump to it
ENDM
;
```

LODSW loads the address of the next Forth word, pointed to by CS:SI, into the AX register. The SI register is automatically incremented by 2 and then points to the next word to be executed. JMP AX jumps to the beginning of the Code Field of the word an executes the machine code stored in it. The macro NEXT assembles these two machine instructions at the end of every code word in LaForth. Because of the auto-incrementing feature in the SI register, 8088 is quite efficient in supporting a Virtual Forth Machine.

```
NEST      MACRO
          DB        0E9h              ;; Long Jump instruction
          DW        DOLIST-$-2        ;; Offset
ENDM
;
DOLIST:   ADD       AX,3
          XCHG      AX,SI
          STOSW
XNEXT:    NEXT
```

;

NEST is the instruction executed by a high level colon word in LaForth. It makes a long jump to the colon word inner interpreter DOLIST. DOLIST adds 3 to AX, which then points to the beginning of the word list in the colon definition. This list pointer is copied into the IP (CS:SI) register, while the contents of IP are pushed on the return stack, ES:DI, by the next instruction STOSW. Now, executing the NEXT macro will cause the first Forth word in the colon list to be executed, while the address of the unfinished colon word is saved on the return stack. The last word in a colon list must be UNNEST, which undoes what NEST did and resumes the execution of the unfinished colon word.

RET
Return from a level of nesting. Returns control to word which called the present one. Generated by semi-colon. Return address must be on top or R-stack, hence cannot be used inside of #[ ]# or if >R values are on R-stack. It can be used within conditionals to abort.

```
            HEADER      TER,R
UNNEST:     SUB         DI,2
            MOV         SI,ES:[DI]
            NEXT

Back-Tab    A synonym of RET
            DB          0,15                        ; Will become "back-tab"
            CHAIN       H
RETRN:      JMP         UNNEST
```

2.3. EXECUTE FORTH WORD DIRECTLY

High level Forth words can be executed directly by placing the execution address of a word on the data stack and invoking EXECUTE. EXCUTE pops the execution address into AX and jumps to this address. As LaForth is based on the Direct Threaded Code and the execution address points to executable code, the control is passed directly to the word we want to execute.

EXECUTE (addr --)
The IP word on top of stack. Execute the word whose address is on top.

```
            HEADER      ETUCEXE,E
EXEC:       POP         AX
            JMP         AX
```

To call a machine language subroutine, we have to use GO.

GO (addr --)
Call a machine language subroutine. A Jump-to-Subroutine is executed going to the address in top. The ;JSR instruction leaves its return address on top, unless ;preserved in the subroutine the RTS will remove it.

```
            HEADER      OG,G
GO:         POP         BX
            CALL        [BX]
            NEXT
```

2.4. STARTING THE FORTH MACHINE

At the beginning of LaForth object code, there is a short machine code routine which initialize the 8088 under DOS and reconfigure the CPU so that it starts to execute Forth code. The preparation word is done by the routine at ORIGN. ORIGN first saves the interrupt vectors used by DOS and replaces them by the ones required by LaForth, and then initializes the IP, SP, and RP registers. The IP register is pointing to the cold start word COLD, which eventually invokes the LaForth text interpreter to interact with the user.

```
ORIGN:  MOV     AX,CS                   ; One-time code.
        MOV     DS,AX
        MOV     AX,351Bh
        INT     21h                     ; Get Int. Vect. for 1B (KB Break)
        MOV     BBKIV,BX                ; Save IV in BIOS Break Int Vect.
        MOV     BBKIV+2,ES
        MOV     DX,OFFSET BIOSBK
        MOV     AX,251Bh
        INT     21h                     ; Set new BIOS Bk Handler
        MOV     DX,OFFSET DOSBK
        MOV     AX,2523h
        INT     21h                     ; Set new DOS Bk Handler
        MOV     DX,1
        MOV     AX,3301h
        INT     21h                     ; Set Break-On Feature.
        MOV     AX,3500h
        INT     21h                     ; Save Divide by zero vector.
        MOV     DB0IV,BX
        MOV     DB0IV+2,ES
        MOV     DX,OFFSET DIVBY0
        MOV     AX,2500h
        INT     21h                     ; Set Division By Zero trap
        LJMP    CINIT                   ; Cold start
WINIT:  MOV     SI,OFFSET WARM+3        ; Warm start
        JMP     SHORT CINIT3
CINIT:  MOV     SI,OFFSET COLD+3
        MOV     AX,CS
        MOV     SS,AX                   ; Set up Stack Segment
        MOV     DS,AX                   ; and Data Segment
        ADD     AX,1000h
        MOV     ES,AX                   ; and Extra Segment
        MOV     BOTB+2,AX
        MOV     LBUF+2,AX
        MOV     AX,BOTB
        MOV     TPTR,AX
        MOV     LBUF,AX
CINIT3: MOV     SP,TOES                 ; Just control stack
        JMP     RCLR
```