Why did the original code for FIND.COM use lop as a label instead of loop?

devblogs.microsoft.com/oldnewthing/20150416-00

April 16, 2015

Raymond Chen

A few years ago, I left you with an exercise: Given the code

```
        mov     dx,st_length            ;length of the string arg.
        dec     dx                      ;adjust for later use
        mov     di, line_buffer
lop:
        inc     dx
        mov     si,offset st_buffer     ;pointer to beg. of string argument

comp_next_char:
        lodsb
        cmp     al,byte ptr [di]
        jnz     no_match

        dec     dx
        jz      a_matchk                ; no chars left: a match!
        call    next_char               ; updates di
        jc      no_match                ; end of line reached
        jmp     comp_next_char          ; loop if chars left in arg.
```

why is the loop label called `lop` instead of `loop`?

The answer is that calling it `loop` would create ambiguity with the 8086 instruction `loop`.

Now, you might say (if your name is Worf), that there is no ambiguity. "Every line consists of up to four things (all optional). A label, an instruction/pseudo-instruction, operands, and comments. The label is optionally followed by a colon. If there is no label, then the line must start with whitespace."

If those were the rules, then there would indeed be no ambiguity.

But those aren't the rules. Leading whitespace is not mandatory. If you are so inclined, you can choose to begin your instructions all in column zero.

```
mov dx,st_length
dec dx
mov di, line_buffer
lop:
inc dx
mov si,offset st_buffer
comp_next_char:
lodsb
cmp al,byte ptr [di]
jnz no_match
dec dx
jz a_matchk
call next_char
jc no_match
jmp comp_next_char
```

It's not recommended, but it's legal. (I have been known to do this when hard-coding breakpoints for debugging purposes. That way, a search for `/^int 3/` will find all of my breakpoints.)

Since you can put the opcode in column zero, a line like this would be ambiguous:

```
loop ret
```

This could be parsed as "Label this line `loop` and execute a `ret` instruction." Or it could be parsed as "This is an unlabeled line, consisting of a `loop` instruction that jumps to the label `ret`."

Label	Opcode	Operand
`loop`	`ret`	
	– or –	
	`loop`	`ret`

Disallowing instruction names as labels or macros or equates is the simplest way out of this predicament. Besides, you probably shouldn't be doing it anyway. Imagine the havoc if you did

```
or equ and
```

Raymond Chen

Follow