CS202 (003): Operating Systems
Trusting Trust

Instructor: Jocelyn Chen

Quiz Time!

Last time

W^X: write XOR execute

- Use MMU to ensure memory cannot be both writeable and executable at same time
- Code segment: executable, not writeable
- Stack, heap, static vars: writeable, not executable
- Supported by most modern processors
- Implemented by modern operating systems

W^X: write XOR execute

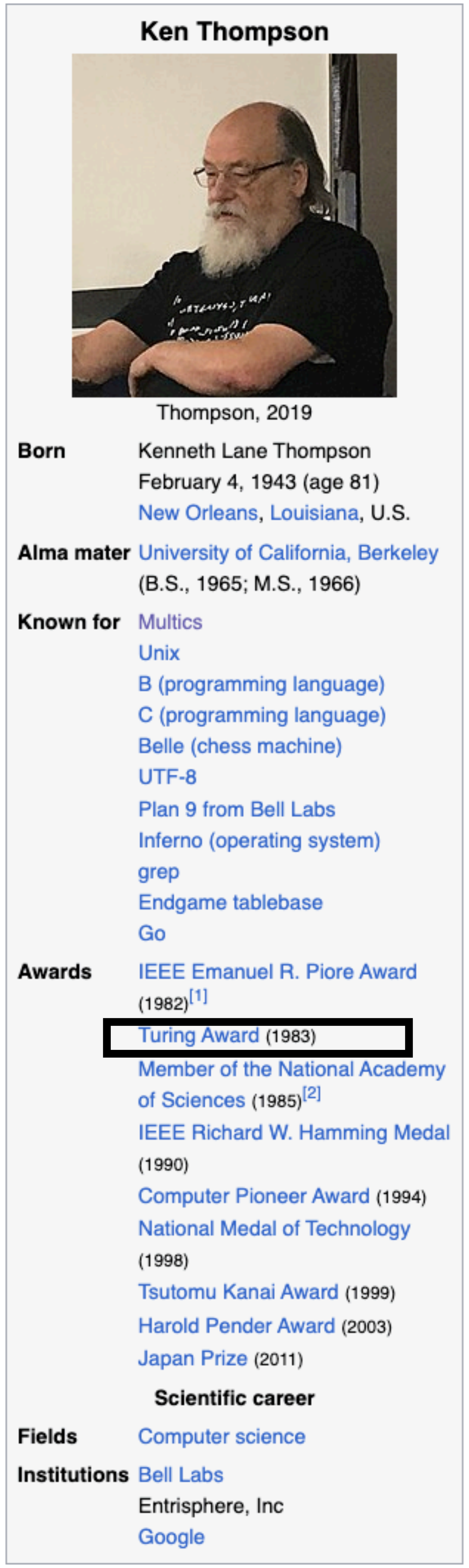
- **Plus:** No code changes or recompile required
- **Minus:** Requires hardware support
- **Minus:** Defeated by return-oriented programming

Control Flow Integrity

- Check destination of every indirect jump
  - Function returns
  - Function pointers
  - Virtual methods
- What are the valid destinations?
  - Caller of every function known at compile time
  - Class hierarchy limits possible virtual function instances

CFI

- **Plus:** No code changes or hardware support
- **Plus:** Protects against many vulnerabilities
- **Minus:** Performance overhead
- **Minus:** Requires smarter compiler
- **Minus:** Requires having all code available



Ken Thompson

Thompson, 2019

Born	Kenneth Lane Thompson February 4, 1943 (age 81) New Orleans, Louisiana, U.S.
Alma mater	University of California, Berkeley (B.S., 1965; M.S., 1966)
Known for	Multics Unix B (programming language) C (programming language) Belle (chess machine) UTF-8 Plan 9 from Bell Labs Inferno (operating system) grep Endgame tablebase Go
Awards	IEEE Emanuel R. Piore Award (1982)[1] Turing Award (1983) Member of the National Academy of Sciences (1985)[2] IEEE Richard W. Hamming Medal (1990) Computer Pioneer Award (1994) National Medal of Technology (1998) Tsutomu Kanai Award (1999) Harold Pender Award (2003) Japan Prize (2011)

Scientific career

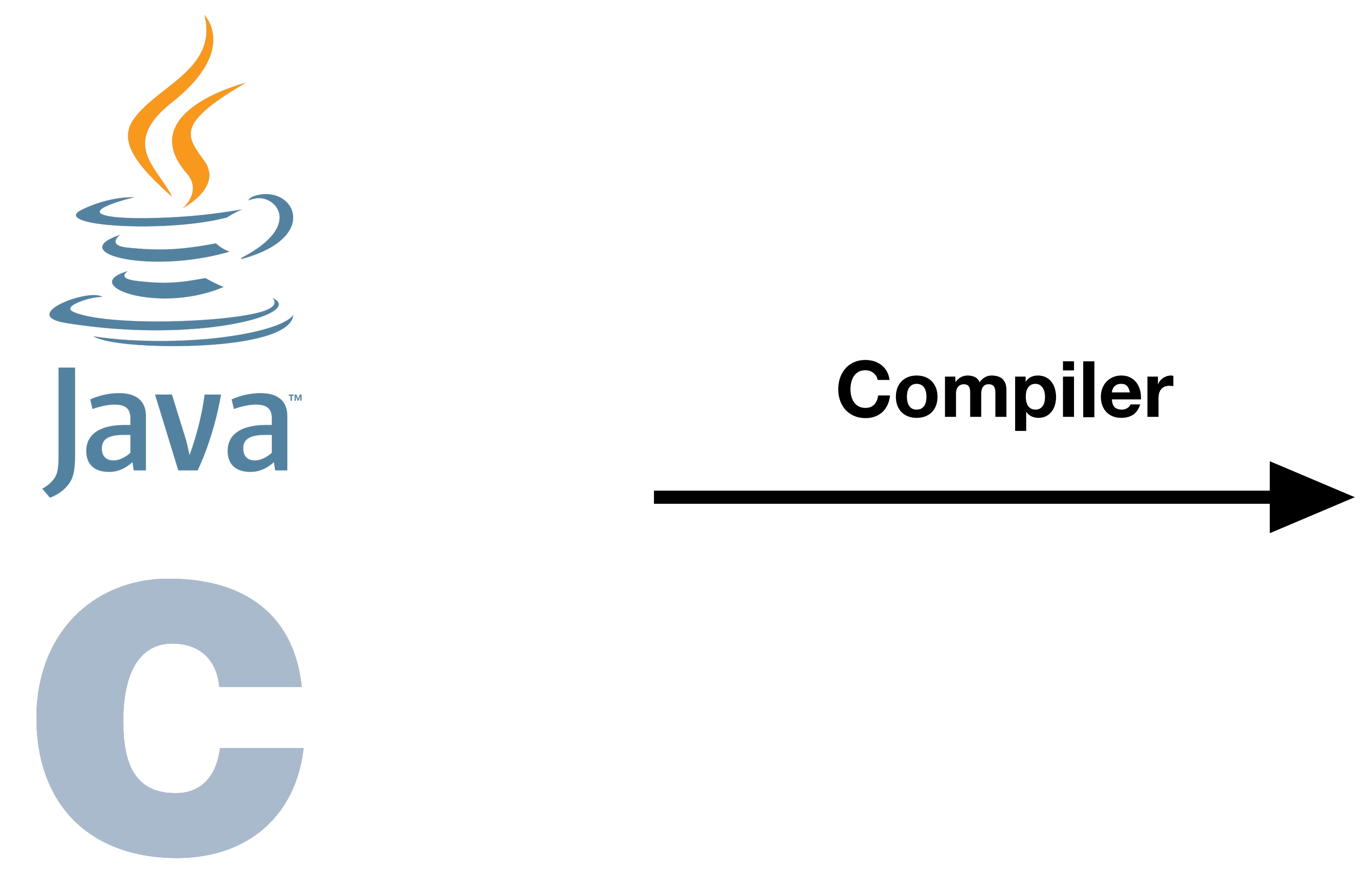
Fields	Computer science
Institutions	Bell Labs Entrisphere, Inc Google

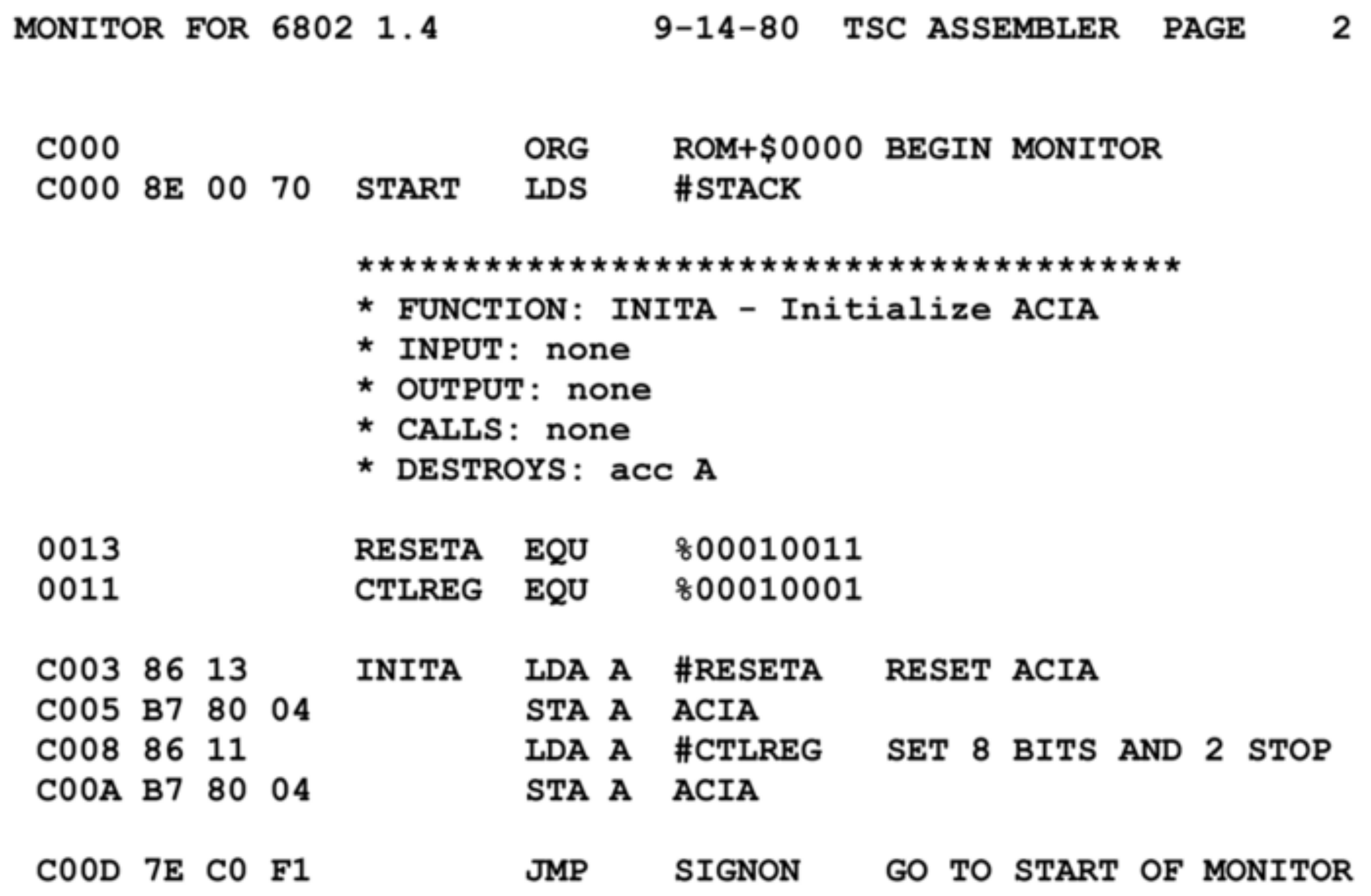
Did you do the reading?

To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.

https://en.wikipedia.org/wiki/Ken_Thompson

Forget about what you read for a sec…





```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2

 C000                   ORG    ROM+$0000 BEGIN MONITOR
 C000 8E 00 70  START   LDS    #STACK

                ***************************************
                * FUNCTION: INITA - Initialize ACIA
                * INPUT: none
                * OUTPUT: none
                * CALLS: none
                * DESTROYS: acc A

 0013           RESETA  EQU    %00010011
 0011           CTLREG  EQU    %00010001

 C003 86 13     INITA   LDA A  #RESETA   RESET ACIA
 C005 B7 80 04          STA A  ACIA
 C008 86 11             LDA A  #CTLREG   SET 8 BITS AND 2 STOP
 C00A B7 80 04          STA A  ACIA

 C00D 7E C0 F1          JMP    SIGNON    GO TO START OF MONITOR
```

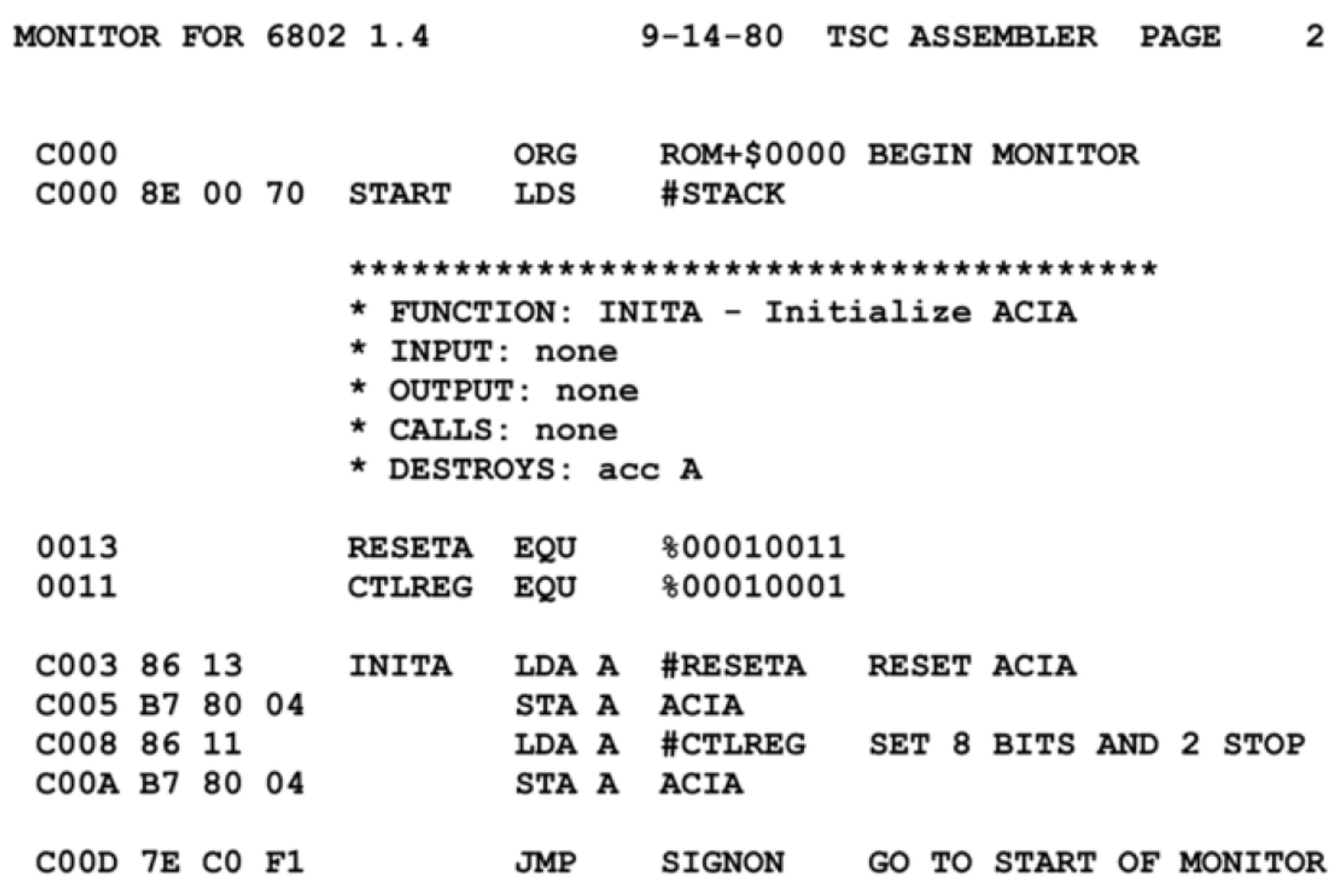
Compiler is a program.
So what does this program written in?

Forget about what you read for a sec…

Java

Compiler written in

C



```
MONITOR FOR 6802 1.4          9-14-80   TSC ASSEMBLER   PAGE    2

 C000                ORG    ROM+$0000 BEGIN MONITOR
 C000 8E 00 70  START   LDS    #STACK

                *****************************************
                * FUNCTION: INITA - Initialize ACIA
                * INPUT: none
                * OUTPUT: none
                * CALLS: none
                * DESTROYS: acc A

 0013           RESETA  EQU    %00010011
 0011           CTLREG  EQU    %00010001

 C003 86 13     INITA   LDA A  #RESETA   RESET ACIA
 C005 B7 80 04          STA A  ACIA
 C008 86 11             LDA A  #CTLREG   SET 8 BITS AND 2 STOP
 C00A B7 80 04          STA A  ACIA

 C00D 7E C0 F1          JMP    SIGNON    GO TO START OF MONITOR
```

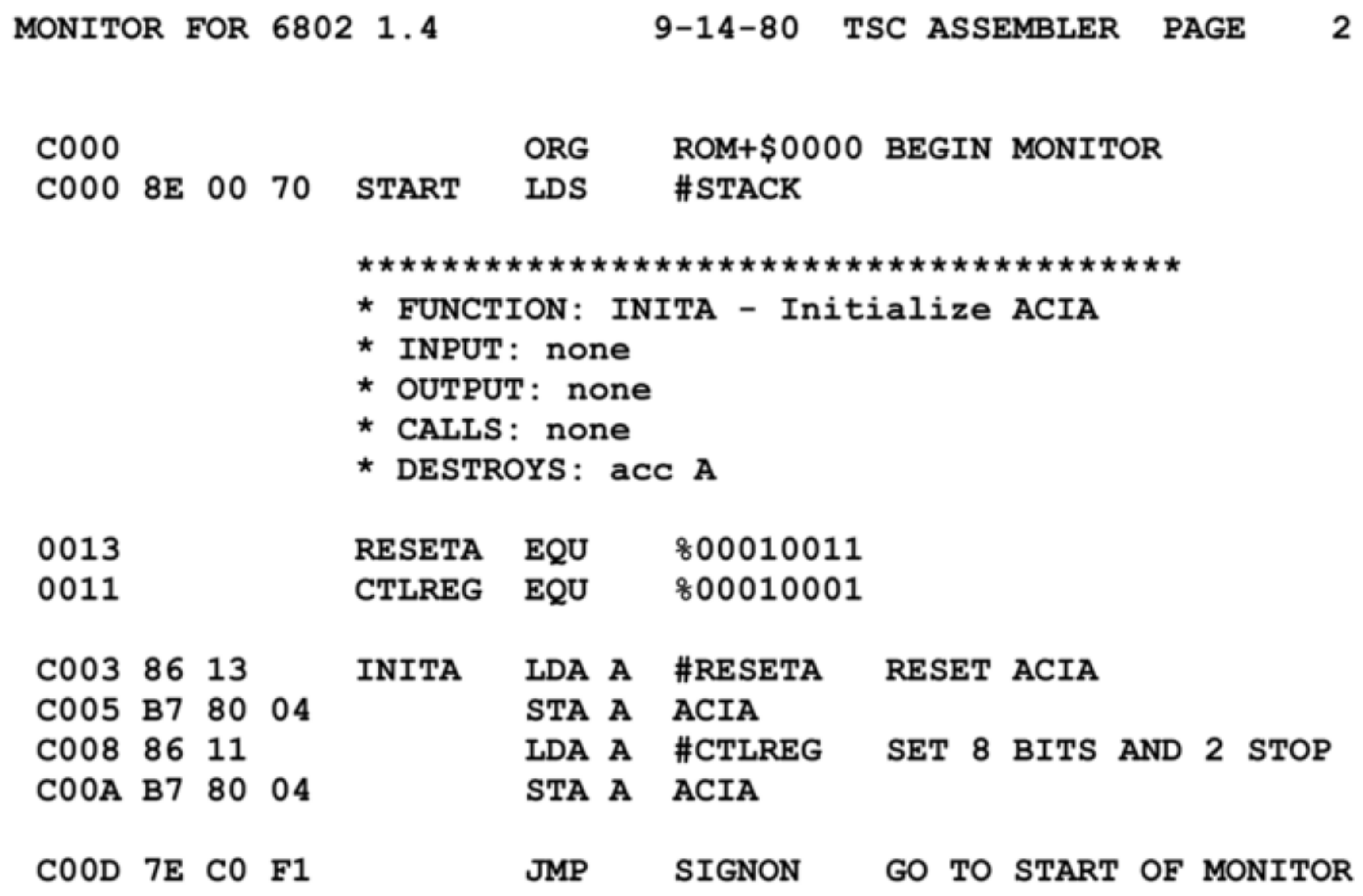
How does compiler know how to translate different types of language features (conditionals, loops, classes) into another language?

Forget about what you read for a sec…

Java

Compiler written in

C



```
MONITOR FOR 6802 1.4          9-14-80   TSC ASSEMBLER   PAGE    2

 C000                   ORG     ROM+$0000 BEGIN MONITOR
 C000 8E 00 70   START  LDS     #STACK

                 ***************************************
                 * FUNCTION: INITA - Initialize ACIA
                 * INPUT: none
                 * OUTPUT: none
                 * CALLS: none
                 * DESTROYS: acc A

 0013            RESETA EQU     %00010011
 0011            CTLREG EQU     %00010001

 C003 86 13      INITA  LDA A   #RESETA    RESET ACIA
 C005 B7 80 04          STA A   ACIA
 C008 86 11             LDA A   #CTLREG    SET 8 BITS AND 2 STOP
 C00A B7 80 04          STA A   ACIA

 C00D 7E C0 F1          JMP     SIGNON     GO TO START OF MONITOR
```

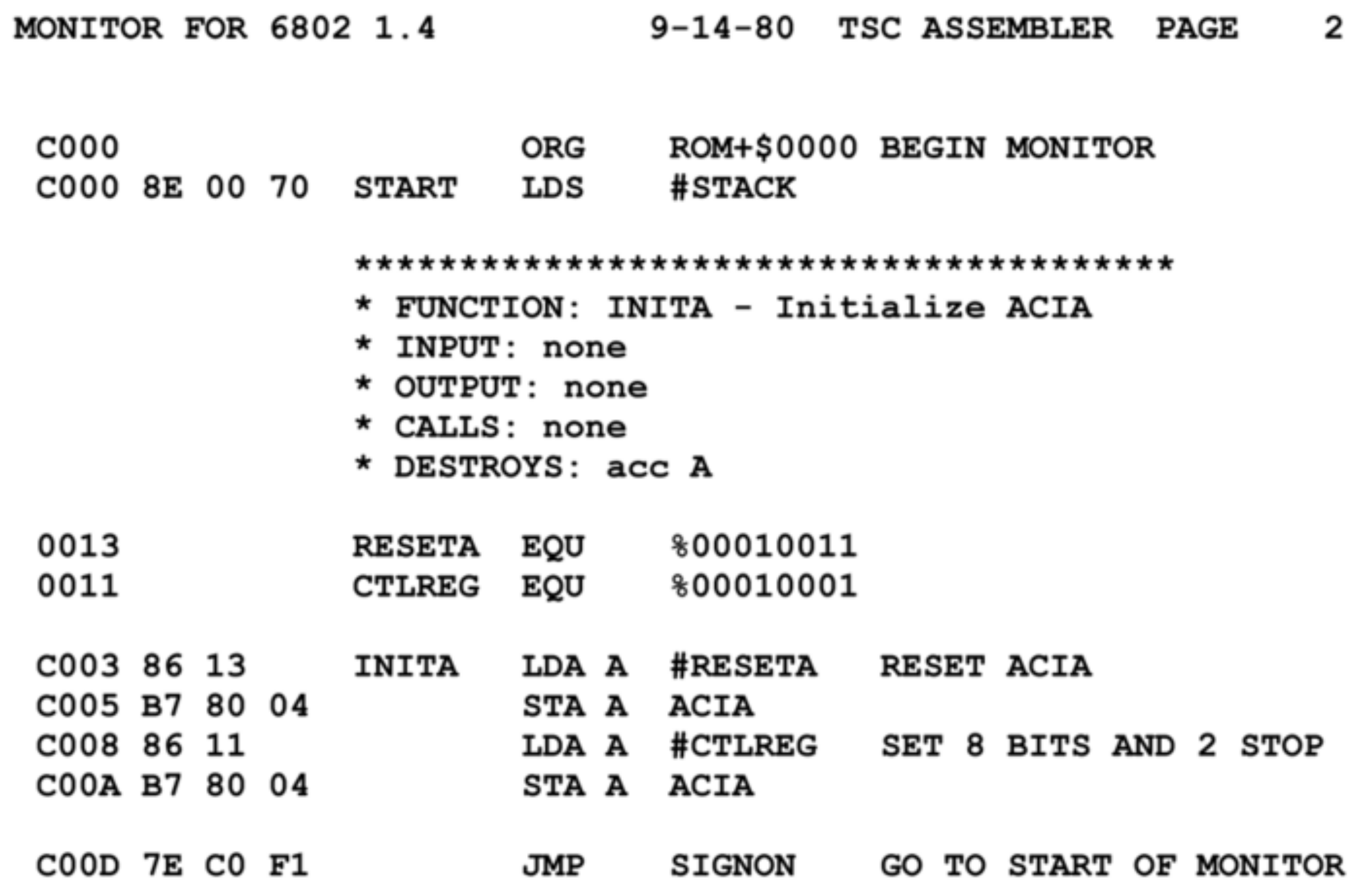
How can we add new language features to Java?

Forget about what you read for a sec…

Java

A new compiler written in

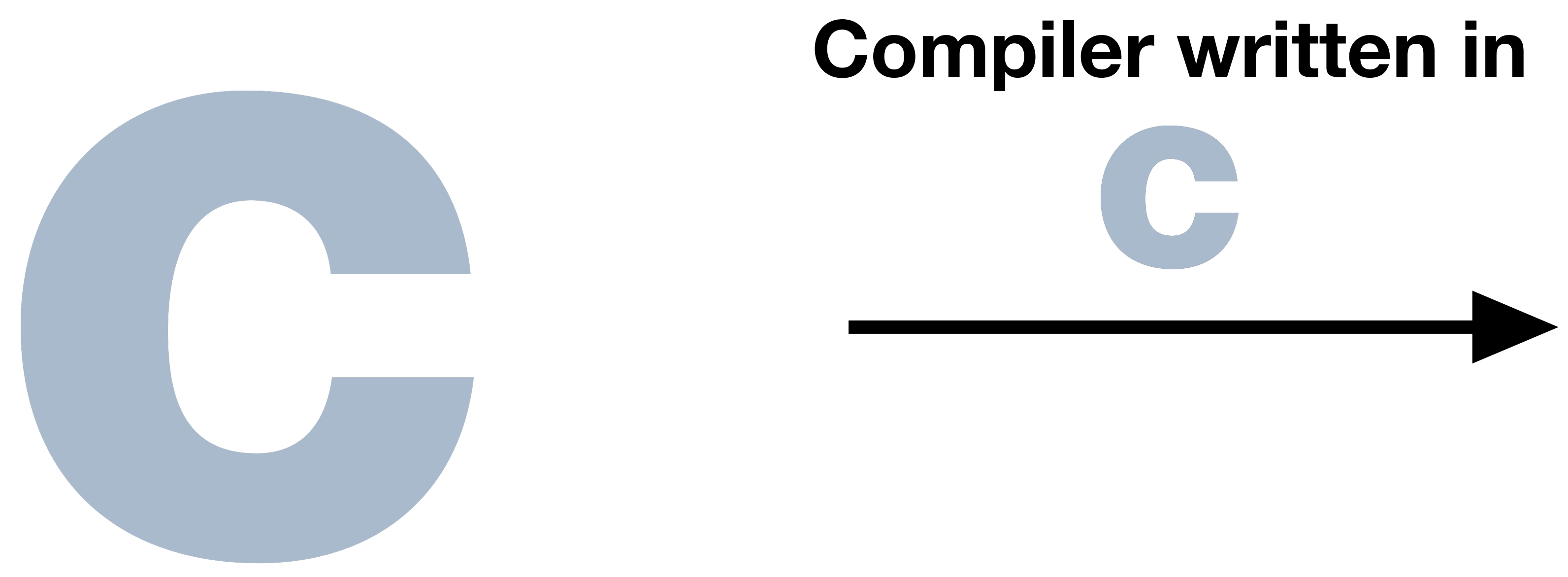
C



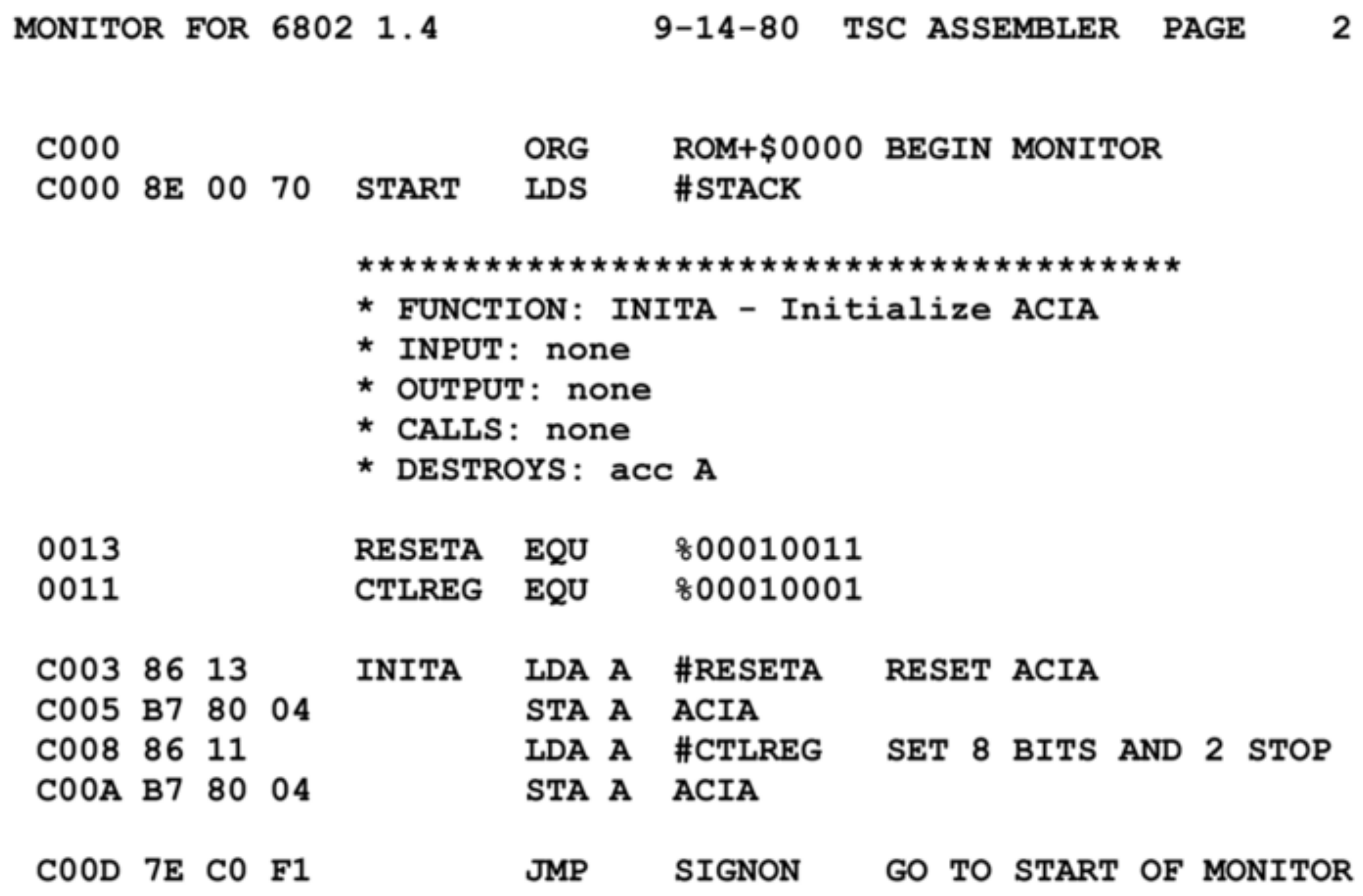
```
MONITOR FOR 6802 1.4          9-14-80   TSC ASSEMBLER   PAGE    2

 C000                         ORG     ROM+$0000 BEGIN MONITOR
 C000 8E 00 70   START        LDS     #STACK

                 ****************************************
                 * FUNCTION: INITA - Initialize ACIA
                 * INPUT: none
                 * OUTPUT: none
                 * CALLS: none
                 * DESTROYS: acc A

 0013            RESETA       EQU     %00010011
 0011            CTLREG       EQU     %00010001

 C003 86 13      INITA        LDA A   #RESETA    RESET ACIA
 C005 B7 80 04                STA A   ACIA
 C008 86 11                   LDA A   #CTLREG    SET 8 BITS AND 2 STOP
 C00A B7 80 04                STA A   ACIA

 C00D 7E C0 F1                JMP     SIGNON     GO TO START OF MONITOR
```

How can we add new language features to Java?

Forget about what you read for a sec…



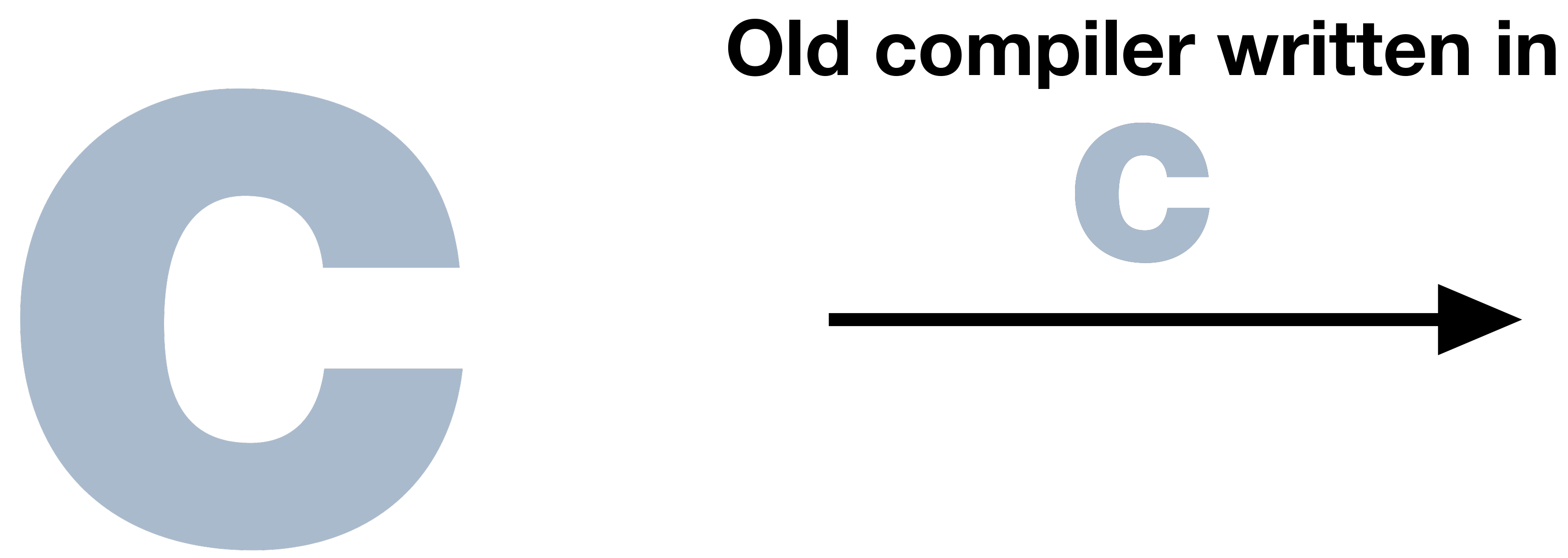
Compiler written in c

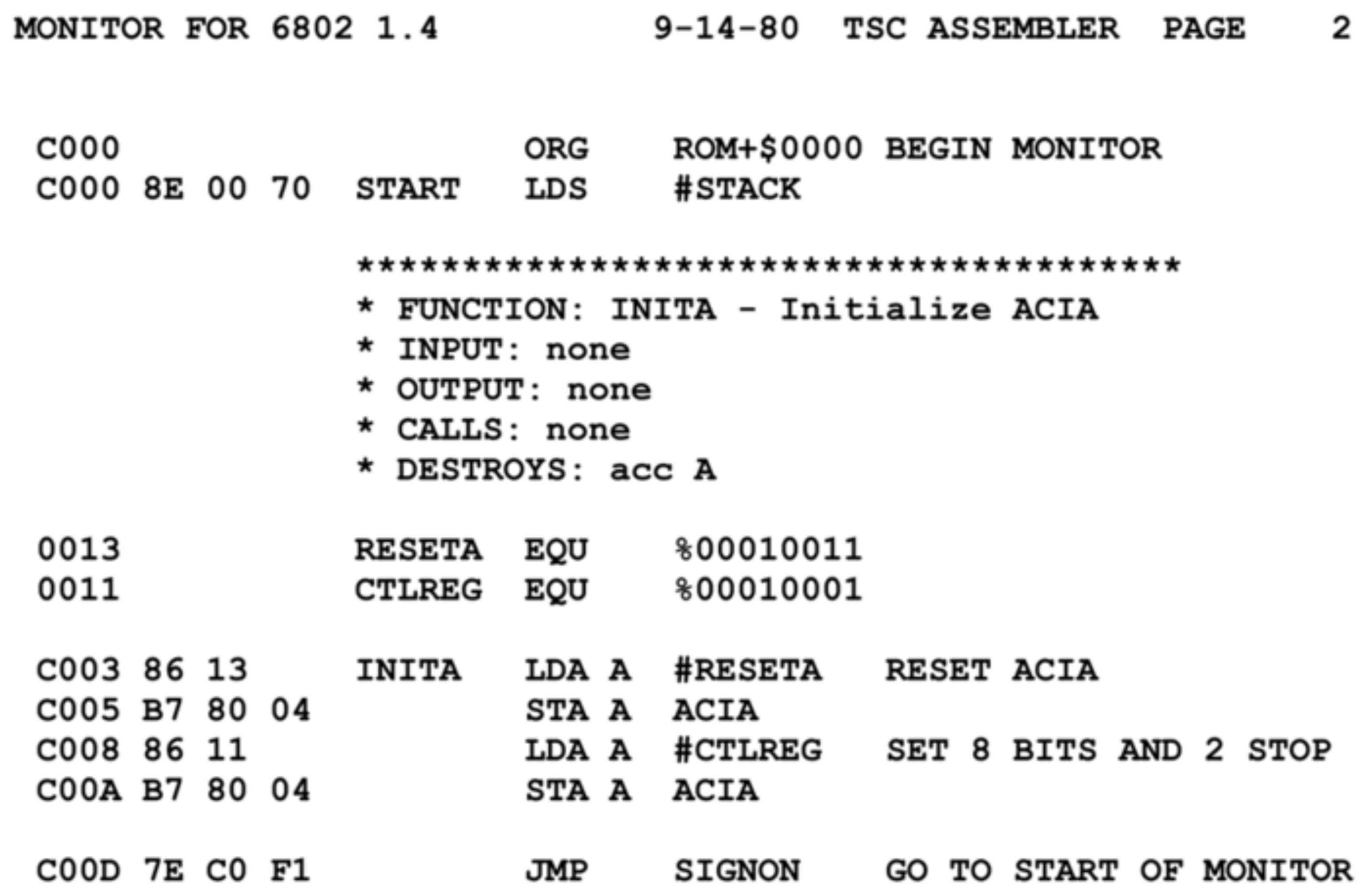


```
MONITOR FOR 6802 1.4          9-14-80   TSC ASSEMBLER   PAGE    2

 C000                   ORG    ROM+$0000 BEGIN MONITOR
 C000 8E 00 70  START   LDS    #STACK

                ***************************************
                * FUNCTION: INITA - Initialize ACIA
                * INPUT: none
                * OUTPUT: none
                * CALLS: none
                * DESTROYS: acc A

 0013           RESETA  EQU    %00010011
 0011           CTLREG  EQU    %00010001

 C003 86 13     INITA   LDA A  #RESETA   RESET ACIA
 C005 B7 80 04          STA A  ACIA
 C008 86 11             LDA A  #CTLREG   SET 8 BITS AND 2 STOP
 C00A B7 80 04          STA A  ACIA

 C00D 7E C0 F1          JMP    SIGNON    GO TO START OF MONITOR
```

How can we add new language features to C?

Forget about what you read for a sec…





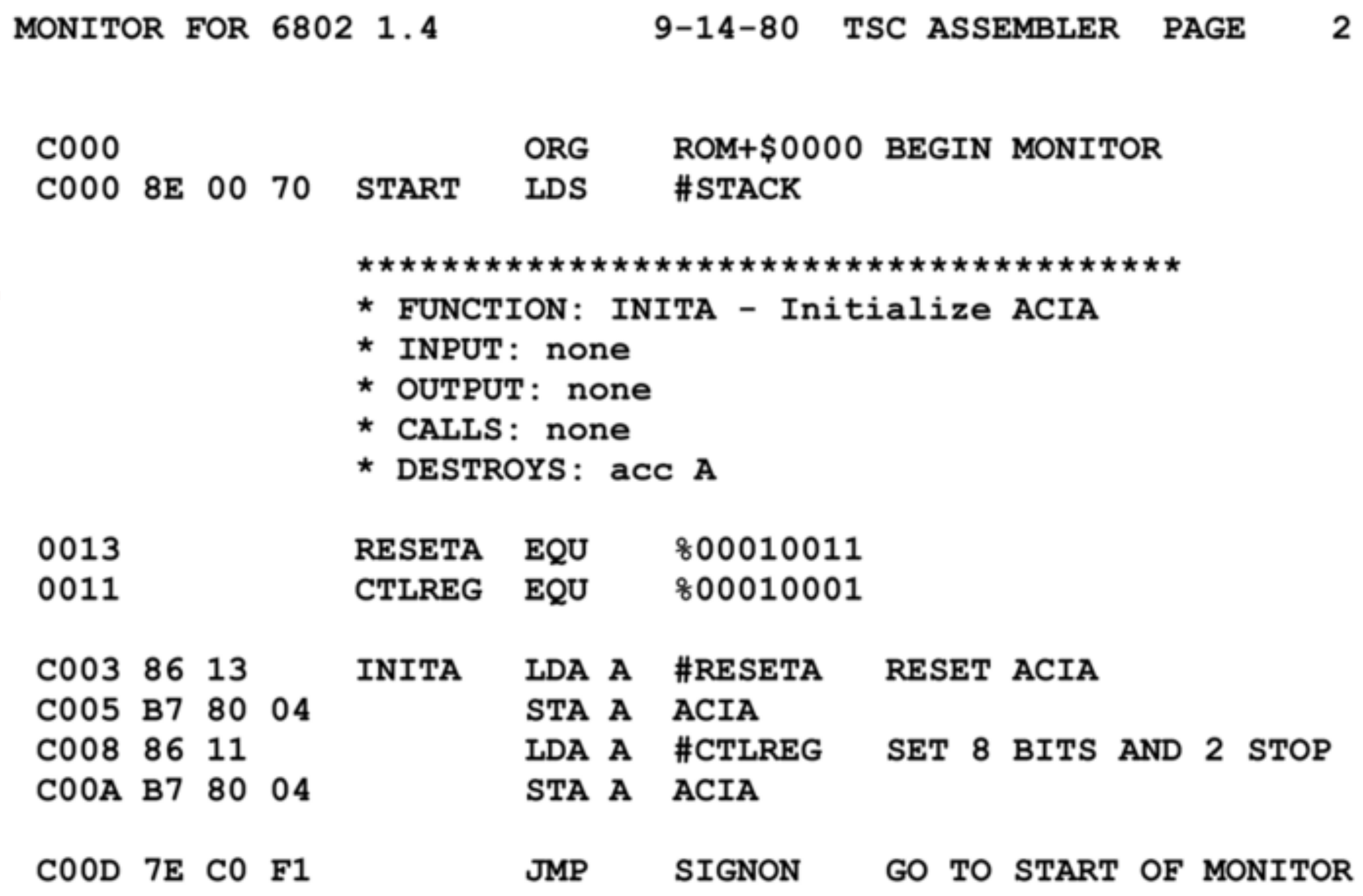
```
MONITOR FOR 6802 1.4          9-14-80   TSC ASSEMBLER   PAGE    2

 C000                ORG     ROM+$0000 BEGIN MONITOR
 C000 8E 00 70  START  LDS     #STACK

               ***************************************
               * FUNCTION: INITA - Initialize ACIA
               * INPUT: none
               * OUTPUT: none
               * CALLS: none
               * DESTROYS: acc A

 0013          RESETA  EQU     %00010011
 0011          CTLREG  EQU     %00010001

 C003 86 13    INITA   LDA A   #RESETA    RESET ACIA
 C005 B7 80 04         STA A   ACIA
 C008 86 11            LDA A   #CTLREG    SET 8 BITS AND 2 STOP
 C00A B7 80 04         STA A   ACIA

 C00D 7E C0 F1         JMP     SIGNON     GO TO START OF MONITOR
```

How can we add new language features to C?

Forget about what you read for a sec…

New compiler written in C

compiled using the old compiler



```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2


 C000                ORG    ROM+$0000 BEGIN MONITOR
 C000 8E 00 70  START   LDS    #STACK

                ****************************************
                * FUNCTION: INITA - Initialize ACIA
                * INPUT: none
                * OUTPUT: none
                * CALLS: none
                * DESTROYS: acc A

 0013           RESETA  EQU    %00010011
 0011           CTLREG  EQU    %00010001

 C003 86 13     INITA   LDA A  #RESETA   RESET ACIA
 C005 B7 80 04          STA A  ACIA
 C008 86 11             LDA A  #CTLREG   SET 8 BITS AND 2 STOP
 C00A B7 80 04          STA A  ACIA

 C00D 7E C0 F1          JMP    SIGNON    GO TO START OF MONITOR
```

How can we add new language features to C?

"Bootstrapping": the technique for producing a self-compiling compiler

Some more context

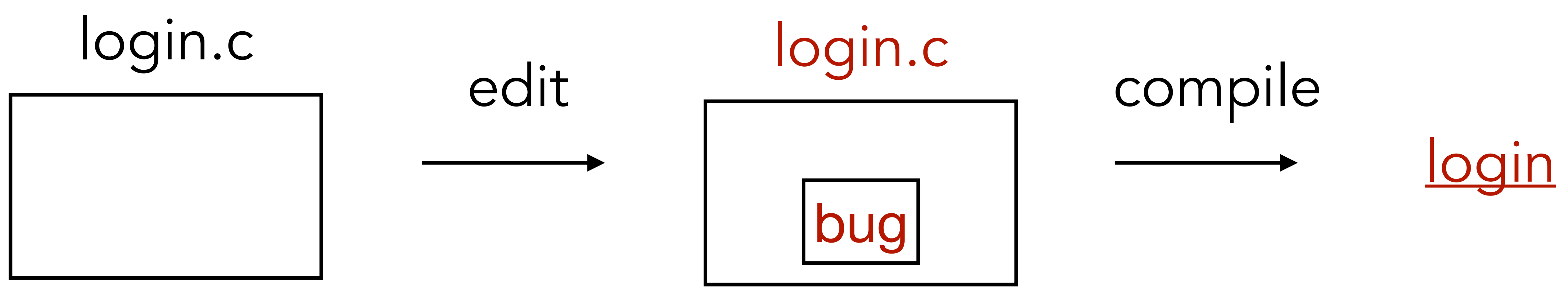
Earlier version of Unix were distributed with a full set of binaries and source for those binaries.

It is common for people to make change in one source file and recompile all their programs

How did Thompson add a bug to the login program without leaving a trace?

Goal

Have no source files hint at the bug, and meanwhile, the bug will persist across all recompilations



Anyone looking at login.c will realize something is wrong!

Goal

Have no source files hint at the bug, and meanwhile, the bug will persist across all recompilations

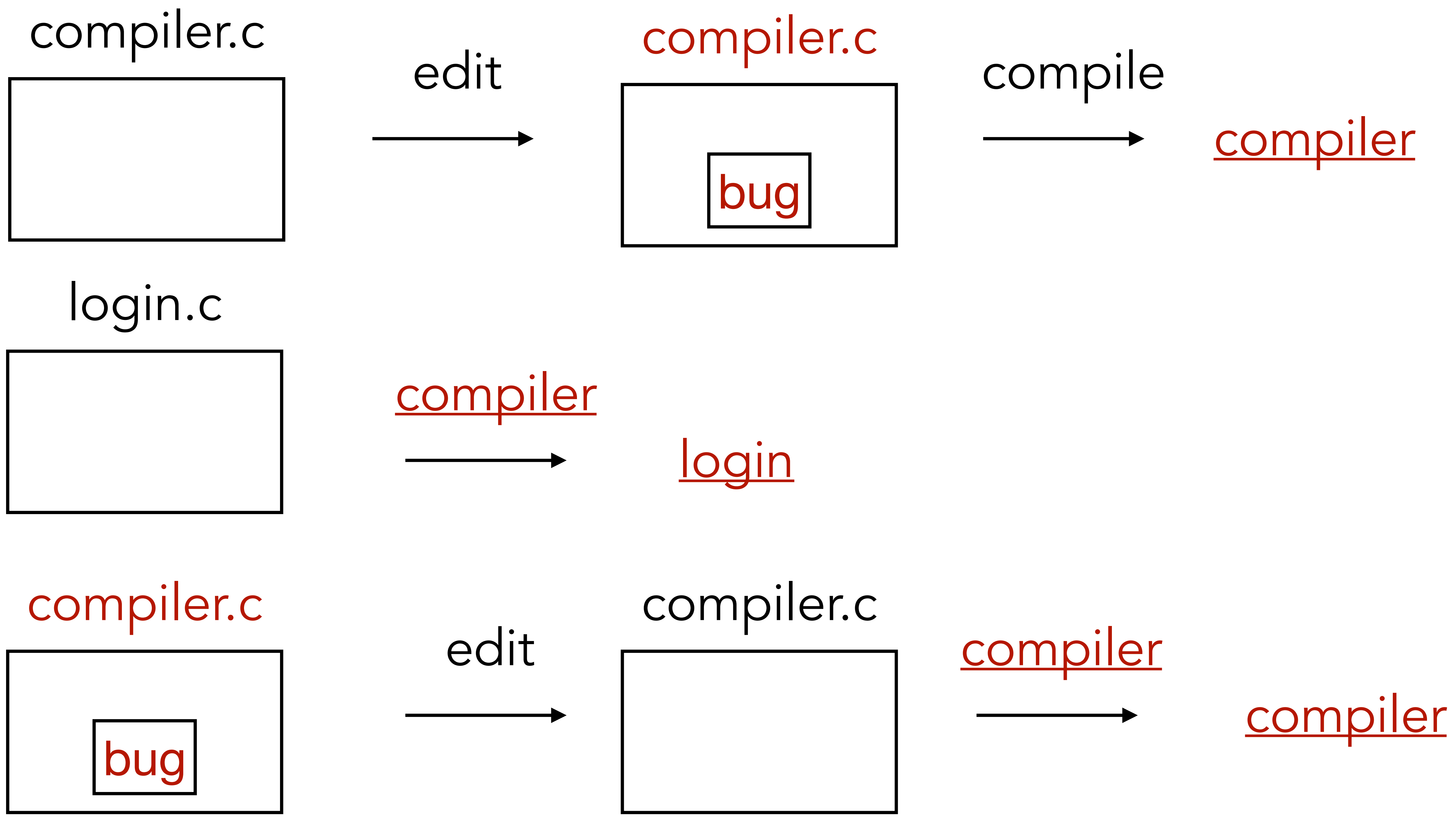
login.c

login

If you recompile locally, login will be bug-free again

Goal

Have no source files hint at the bug, and meanwhile, the bug will persist across all recompilations



Done!

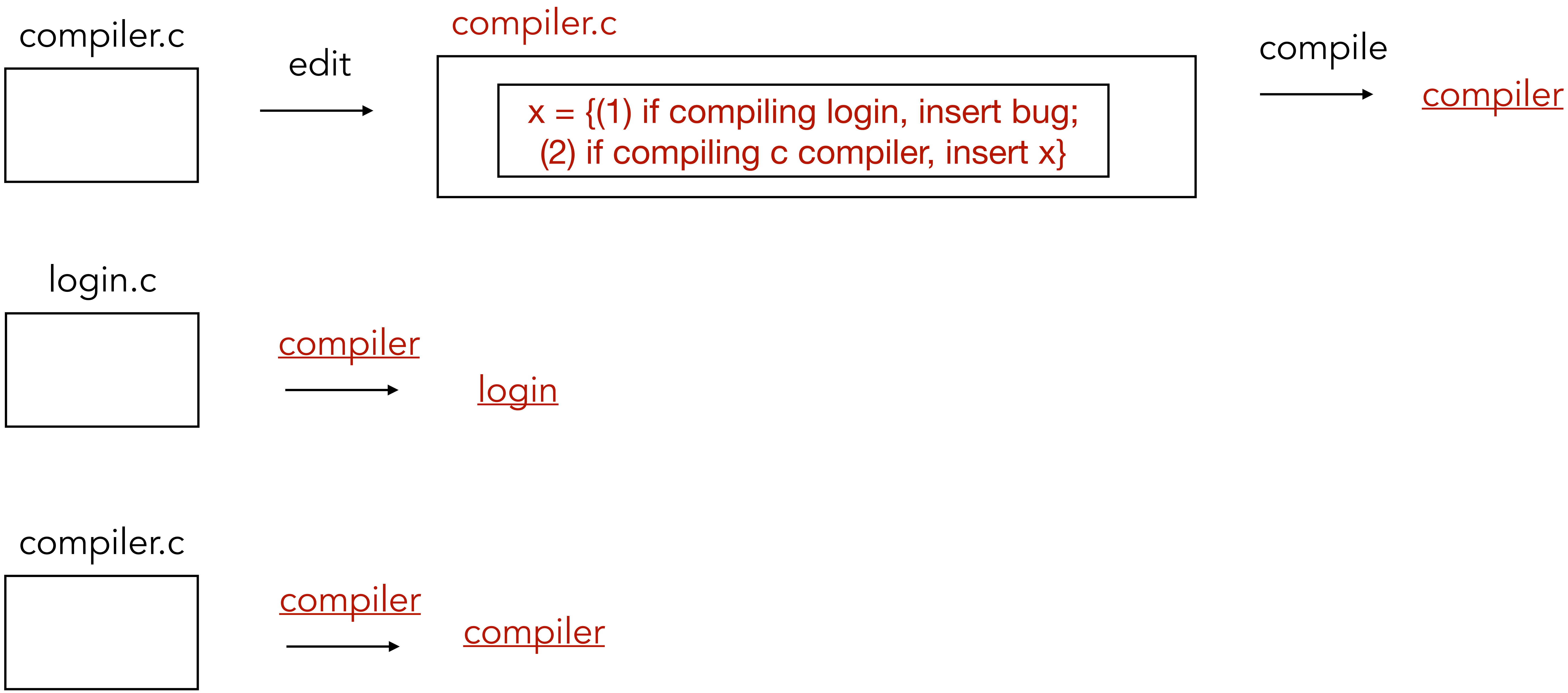
How can Ken figure out this attack?

Self-reproducing program: a computer program that takes no input and produces a copy of its own source code as its only output. (Quine)

"yields falsehood when preceded by its quotation" yields falsehood when preceded by its quotation.

https://en.wikipedia.org/wiki/Quine%27s_paradox
https://rosettacode.org/wiki/Quine

Actual attack



compiler.c

edit

compiler.c

x = {(1) if compiling login, insert bug;
(2) if compiling c compiler, insert x}

compile

compiler

login.c

compiler

login

compiler.c

compiler

compiler

Done!

Implications

You can’t trust code that you did not totally create yourself!