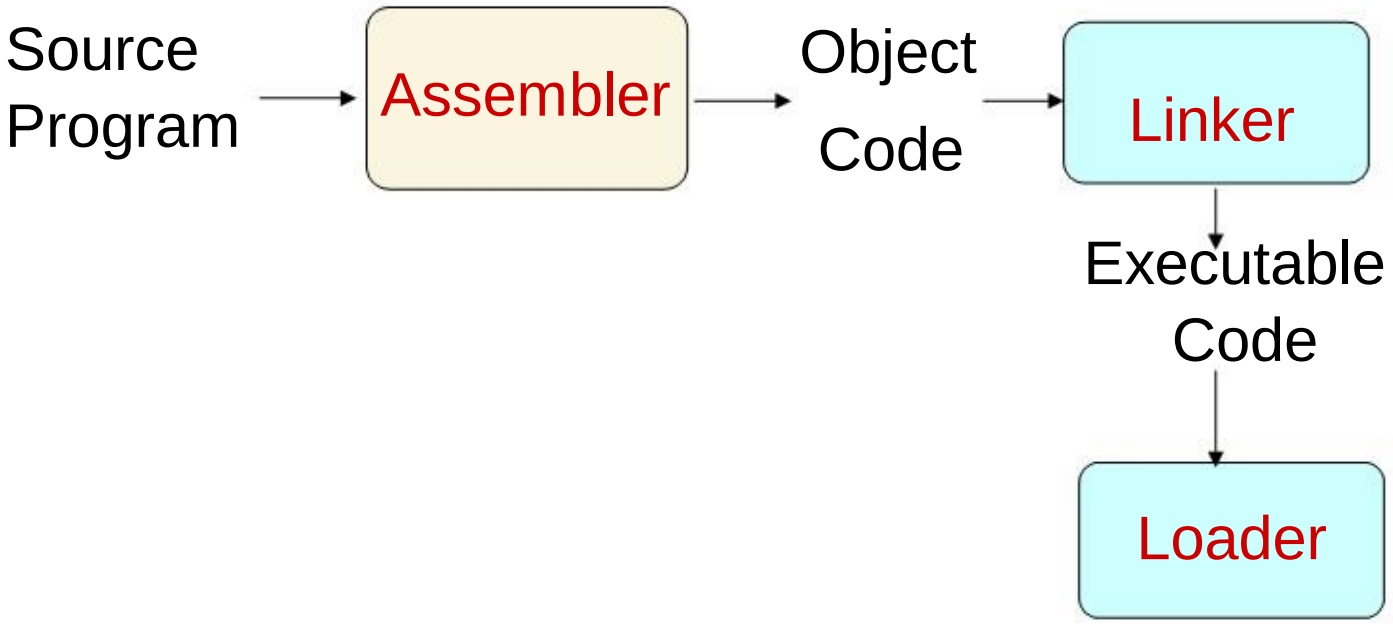
SWE4001 – System Programming
Module 4: Loader and Linkers
Lesson 1 of 6: Basic Loader Functions

Module 4
Loaders and Linkers



4.1 Basic Loader Functions

- In Module 3, we discussed
  - *Loading:* brings the OP into memory for execution
  - *Relocating:* modifies the OP so that it can be loaded at an address different form the location originally specified.
  - *Linking:* combines two or more separate OPs (sec. 2.3.5)
- In Module 4, we will discuss
  - A *loader* brings an object program into memory and starting its execution.
  - A *linker* performs the linking operations and a separate loader to handle relocation and loading.

4.1 Basic Loader Functions
Design of Absolute Loader

- Loader does not perform functions as linking and program location.
- Operation is very simple.
- All functions are accomplished in a single pass.
- Header record
  - Check the Header record for program name, starting address, and length (available memory)
- Text record
  - Bring the object program contained in the Text record to the indicated address
- End record
  - Transfer control to the address specified in the End record

3.1 Basic Loader Functions
3.1.1 Design of an Absolute Loader

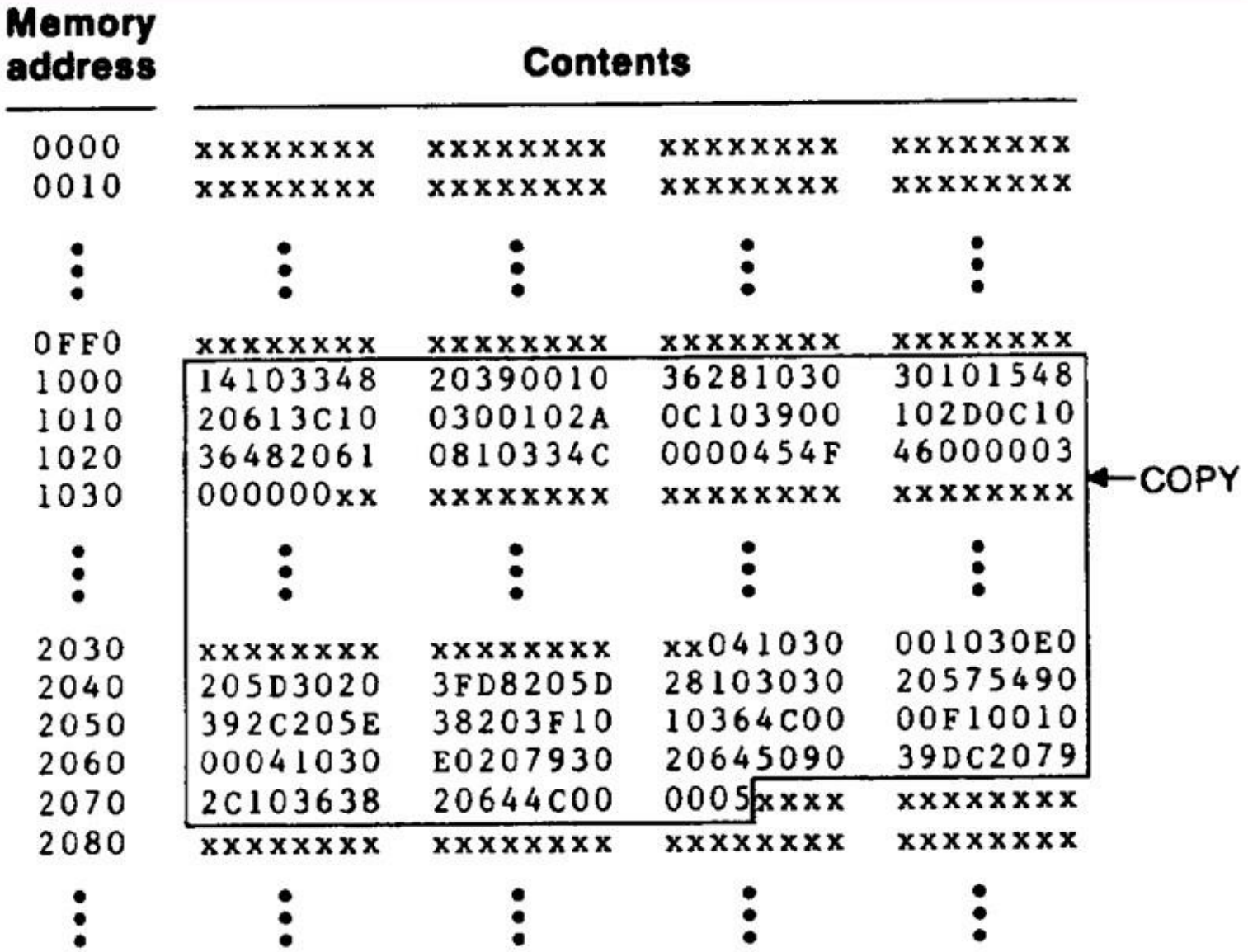
- Absolute loader (for SIC), in Figures 3.1 and 3.2.
  - Does not perform linking and program relocation.
  - The contents of memory locations for which there is no Text record are shown as xxxx.
  - Each byte of assembled code is given using its Hex representation in character form.

```
HCOPY  001000001007A
T001000 1E 141033 482039 001036 281030 301015 482061 3C1003 00102A 0C1039 00102D
T00101E 15 0C1036 482061 081033 4C0000 454F46 000003 000000
T002039 1E 041030 001030 E0205D 30203F D8205D 281030 302057 549039 2C205E 38203F
T002057 1C 101036 4C0000 F10010 000041 030E02 079302 064509 039DC2 079 2C1036
T002073 07 382064 4C0000 05
E001000
```

(a) Object program

3.1.1 Design of an Absolute Loader

- Absolute loader, in Figure 3.1 and 3.2.
  - STL instruction, *pair of characters* 14, when these are read by loader, they will occupy *two* bytes of memory.
  - 14 (Hex 31 34) ----> 00010100 (one byte)
  - For execution, the operation code must be store in a single byte with hexadecimal value 14.
  - Each pair of bytes must be packed together into one byte.
  - Each printed character represents one half-byte.



(b) Program loaded in memory

Figure 3.1 Loading of an absolute program.

```
begin
    read Header record
    verify program name and length
    read first Text record
    while record type ≠ 'E' do
      begin
        {if object code is in character form, convert into
           internal representation}
        move object code to specified location in memory
        read next object program record
      end
    jump to address specified in End record
end
```

Figure 3.2 Algorithm for an absolute loader.

3.1.2 A Simple Bootstrap Loader

- A bootstrap loader, Figure 3.3.
  - Loads the first program to be run by the computer---usually an operating system.
  - The bootstrap itself begins at address 0 in the memory.
  - It loads the OS or some other program starting at address 80.

3.1.2 A Simple Bootstrap Loader

- A bootstrap loader, Figure 3.3.
  - Each byte of object code to be loaded is represented on device F1 as two Hex digits (by GETC subroutines).
  - The ASCII code for the character 0 (Hex 30) is converted to the numeric value 0.
  - The object code from device F1 is always loaded into consecutive bytes of memory, starting at address 80.

```
BOOT     START     0          BOOTSTRAP LOADER FOR SIC/XE
.
. THIS BOOTSTRAP READS OBJECT CODE FROM DEVICE F1 AND ENTERS IT
. INTO MEMORY STARTING AT ADDRESS 80 (HEXADECIMAL). AFTER ALL OF
. THE CODE FROM DEVF1 HAS BEEN SEEN ENTERED INTO MEMORY, THE
. BOOTSTRAP EXECUTES A JUMP TO ADDRESS 80 TO BEGIN EXECUTION OF
. THE PROGRAM JUST LOADED.  REGISTER X CONTAINS THE NEXT ADDRESS
. TO BE LOADED.
.
         CLEAR     A          CLEAR REGISTER A TO ZERO
         LDX       #128       INITIALIZE REGISTER X TO HEX 80
LOOP     JSUB      GETC       READ HEX DIGIT FROM PROGRAM BEING LOADED
         RMO       A,S        SAVE IN REGISTER S
         SHIFTL    S,4        MOVE TO HIGH-ORDER 4 BITS OF BYTE
         JSUB      GETC       GET NEXT HEX DIGIT
         ADDR      S,A        COMBINE DIGITS TO FORM ONE BYTE
         STCH      0,X        STORE AT ADDRESS IN REGISTER X
         TIXR      X,X        ADD 1 TO MEMORY ADDRESS BEING LOADED
         J         LOOP       LOOP UNTIL END OF INPUT IS REACHED
```

```
.
. SUBROUTINE TO READ ONE CHARACTER FROM INPUT DEVICE AND
. CONVERT IT FROM ASCII CODE TO HEXADECIMAL DIGIT VALUE. THE
. CONVERTED DIGIT VALUE IS RETURNED IN REGISTER A. WHEN AN
. END-OF-FILE IS READ, CONTROL IS TRANSFERRED TO THE STARTING
. ADDRESS (HEX 80).
.
GETC    TD      INPUT   TEST INPUT DEVICE
        JEQ     GETC    LOOP UNTIL READY
        RD      INPUT   READ CHARACTER
        COMP    #4      IF CHARACTER IS HEX 04 (END OF FILE),
        JEQ     80          JUMP TO START OF PROGRAM JUST LOADED
        COMP    #48     COMPARE TO HEX 30 (CHARACTER '0')
        JLT     GETC    SKIP CHARACTERS LESS THAN '0'
        SUB     #48     SUBTRACT HEX 30 FROM ASCII CODE
        COMP    #10     IF RESULT IS LESS THAN 10, CONVERSION IS
        JLT     RETURN      COMPLETE. OTHERWISE, SUBTRACT 7 MORE
        SUB     #7          (FOR HEX DIGITS 'A' THROUGH 'F')
RETURN  RSUB            RETURN TO CALLER
INPUT   BYTE    X'F1'   CODE FOR INPUT DEVICE
        END     LOOP
```

Figure 3.3 Bootstrap loader for SIC/XE.