Chapter 2 Assemblers

-- 2.4 Assembler Design Options

Outline

- One-pass assemblers
- Multi-pass assemblers
- Two-pass assembler with overlay structure

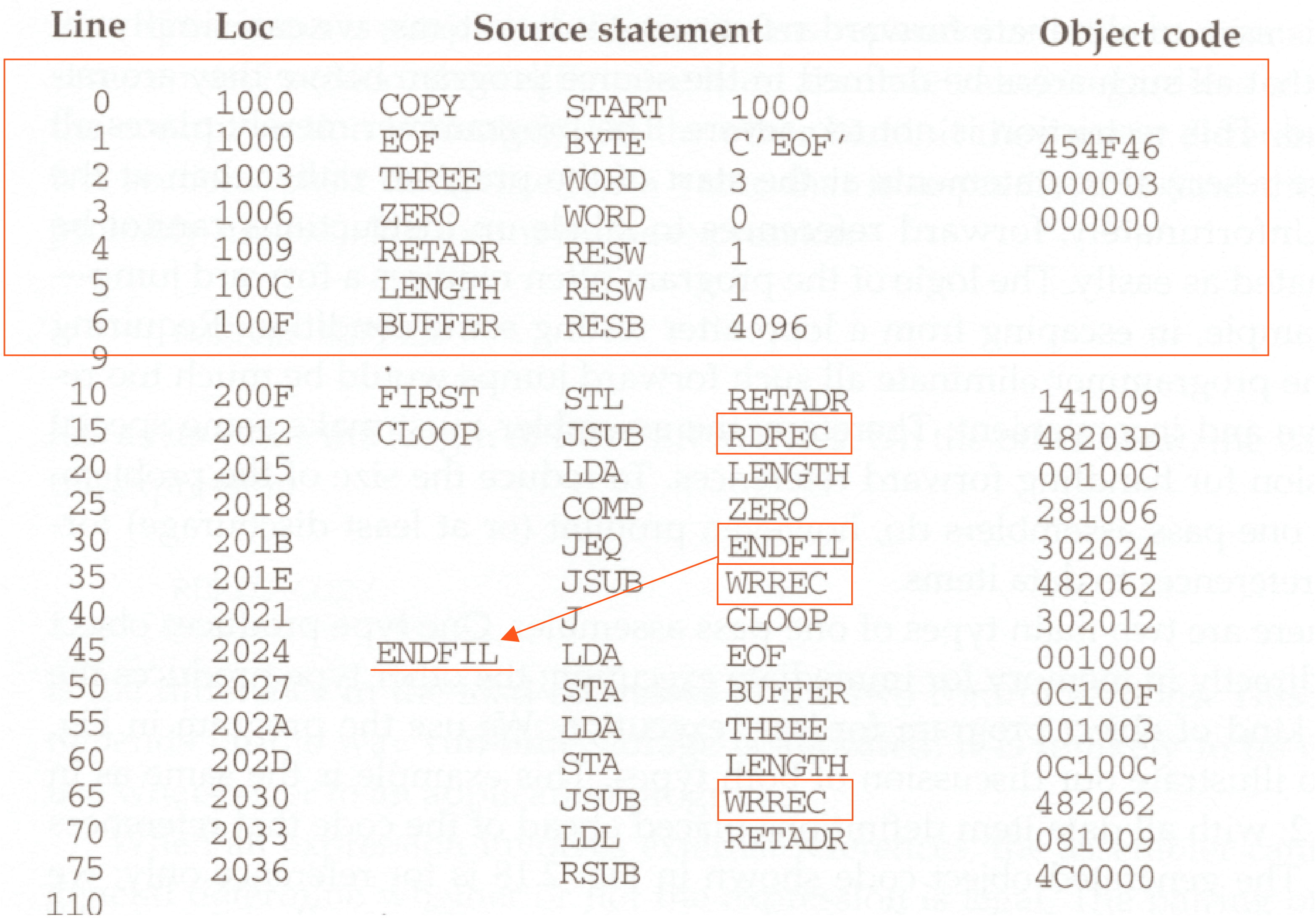
Load-and-Go Assembler

- Load-and-go assembler generates their object code in <u>memory</u> for immediate execution.
- No object program is written out, no loader is needed.
- It is useful in a system oriented toward program development and testing such that the efficiency of the assembly process is an important consideration.
- Programs are re-assembled nearly every time they are run, efficiency of the assembly process is an important consideration.

One-Pass Assemblers

- Scenario for one-pass assemblers
  - Generate their object code in memory for immediate execution – *load-and-go* assembler
  - External storage for the intermediate file between two passes is slow or is inconvenient to use
- Main problem - Forward references
  - Data items
  - Labels on instructions
- Solution
  - Require that all areas be defined before they are referenced.
  - It is possible, although inconvenient, to do so for data items.
  - Forward jump to instruction items cannot be easily eliminated.
    - Insert (label, *address_to_be_modified*) to SYMTAB
    - Usually, *address_to_be_modified* is stored in a linked-list

Sample program for a one-pass assembler Figure 2.18, pp. 94



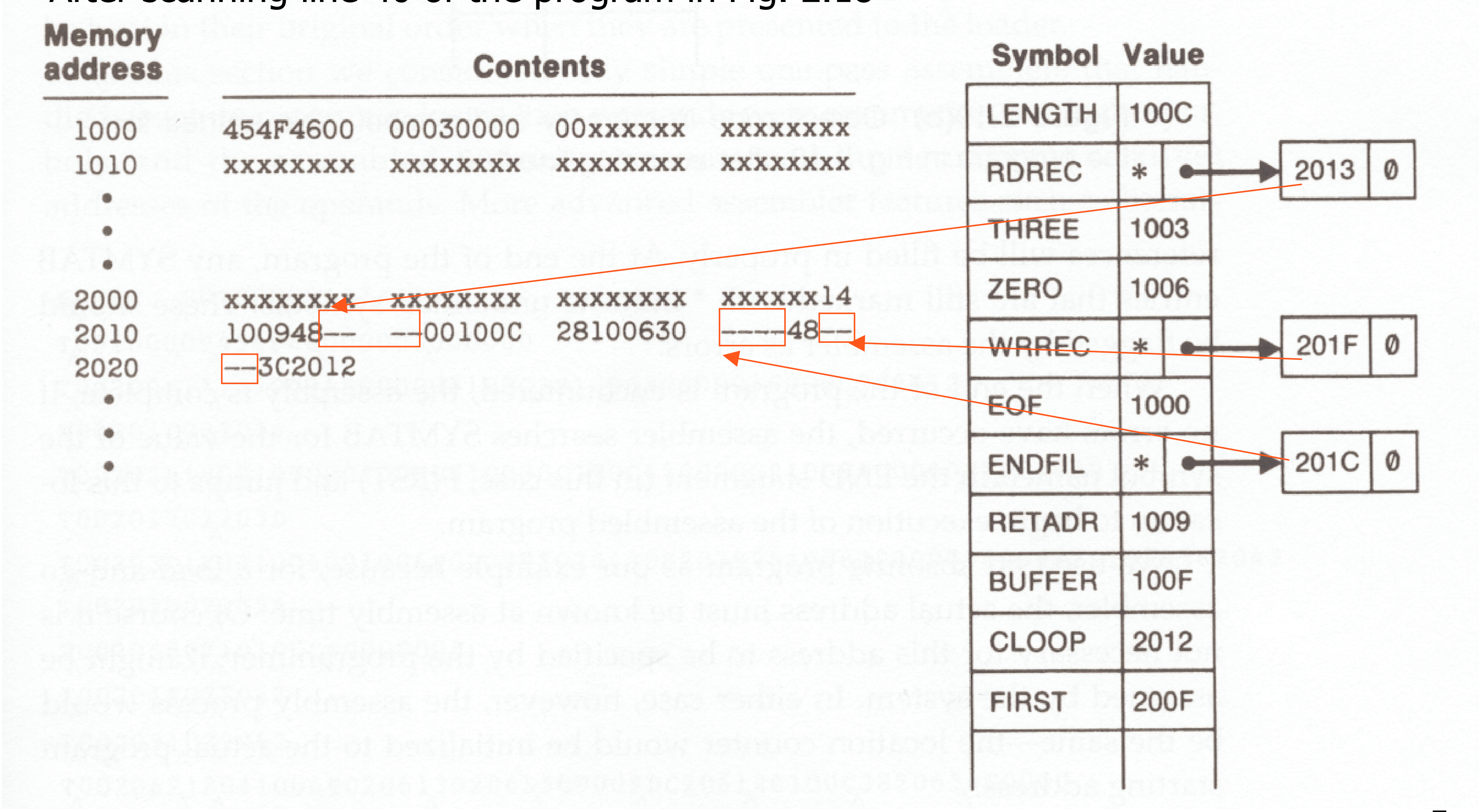
Line	Loc	Source statement			Object code
0	1000	COPY	START	1000	
1	1000	EOF	BYTE	C'EOF'	454F46
2	1003	THREE	WORD	3	000003
3	1006	ZERO	WORD	0	000000
4	1009	RETADR	RESW	1	
5	100C	LENGTH	RESW	1	
6	100F	BUFFER	RESB	4096	
9		.			
10	200F	FIRST	STL	RETADR	141009
15	2012	CLOOP	JSUB	RDREC	48203D
20	2015		LDA	LENGTH	00100C
25	2018		COMP	ZERO	281006
30	201B		JEQ	ENDFIL	302024
35	201E		JSUB	WRREC	482062
40	2021		J	CLOOP	302012
45	2024	ENDFIL	LDA	EOF	001000
50	2027		STA	BUFFER	0C100F
55	202A		LDA	THREE	001003
60	202D		STA	LENGTH	0C100C
65	2030		JSUB	WRREC	482062
70	2033		LDL	RETADR	081009
75	2036		RSUB		4C0000
110		.			

Forward Reference in One-pass Assembler

- Omits the operand address if the symbol has not yet been defined
- Enters this undefined symbol into SYMTAB and indicates that it is undefined
- Adds the address of this operand address to a list of forward references associated with the SYMTAB entry
- When the definition for the symbol is encountered, scans the reference list and inserts the address.
- At the end of the program, reports the error if there are still SYMTAB entries indicated undefined symbols.
- For Load-and-Go assembler
  - Search SYMTAB for the symbol named in the END statement and jumps to this location to begin execution if there is no error

Object Code in Memory and SYMTAB Figure 2.19(a), pp.95

After scanning line 40 of the program in Fig. 2.18



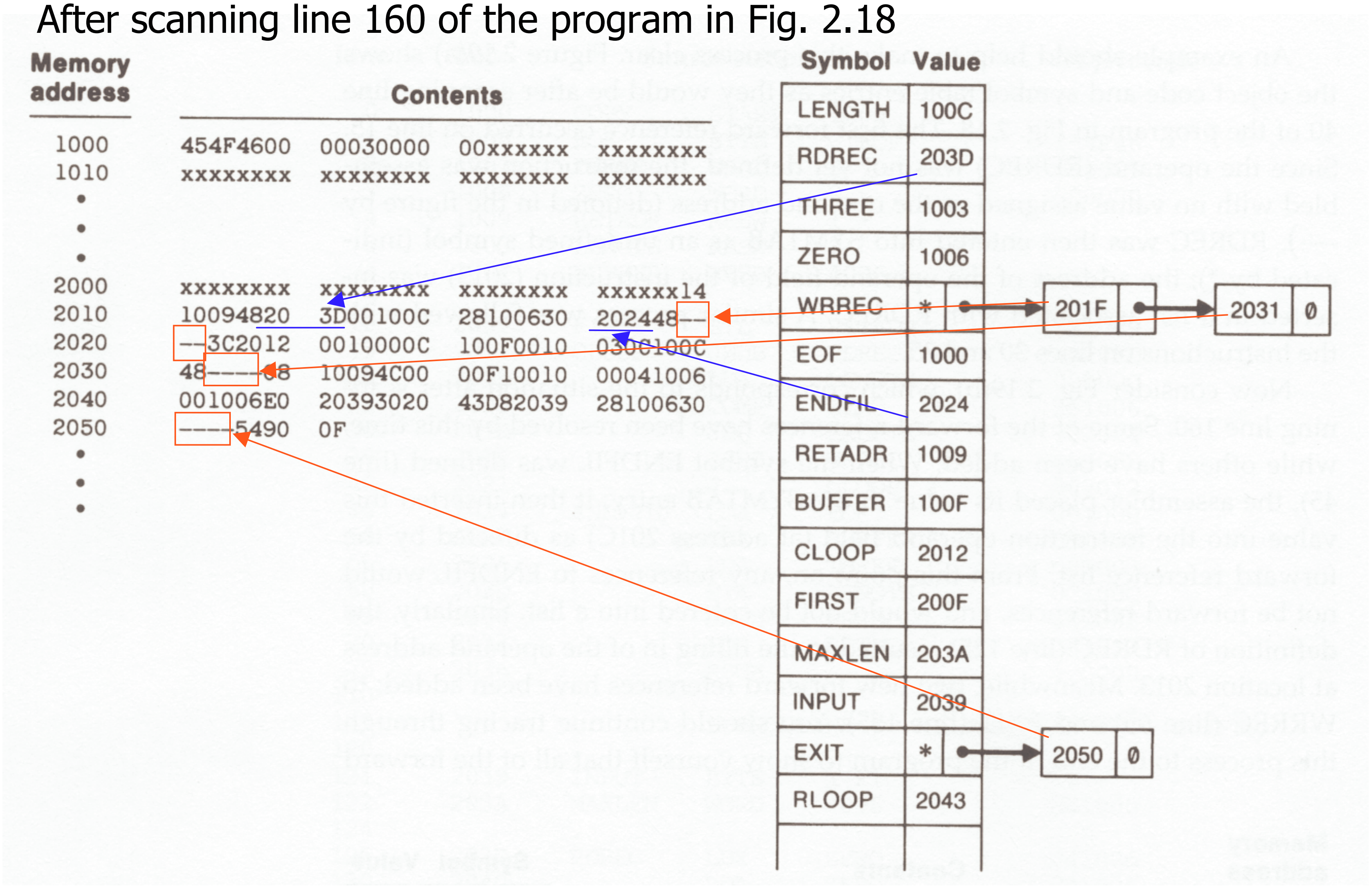
Memory address | Contents

```
1000   454F4600  00030000  00xxxxxx  xxxxxxxx
1010   xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxxxx
  •
  •
  •
2000   xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxx14
2010   100948--  --00100C  28100630  ------48--
2020   --3C2012
  •
  •
  •
```

Symbol	Value
LENGTH	100C
RDREC	* → 2013 ∅
THREE	1003
ZERO	1006
WRREC	* → 201F ∅
EOF	1000
ENDFIL	* → 201C ∅
RETADR	1009
BUFFER	100F
CLOOP	2012
FIRST	200F

Object Code in Memory and SYMTAB Figure 2.19(b), pp.96

After scanning line 160 of the program in Fig. 2.18



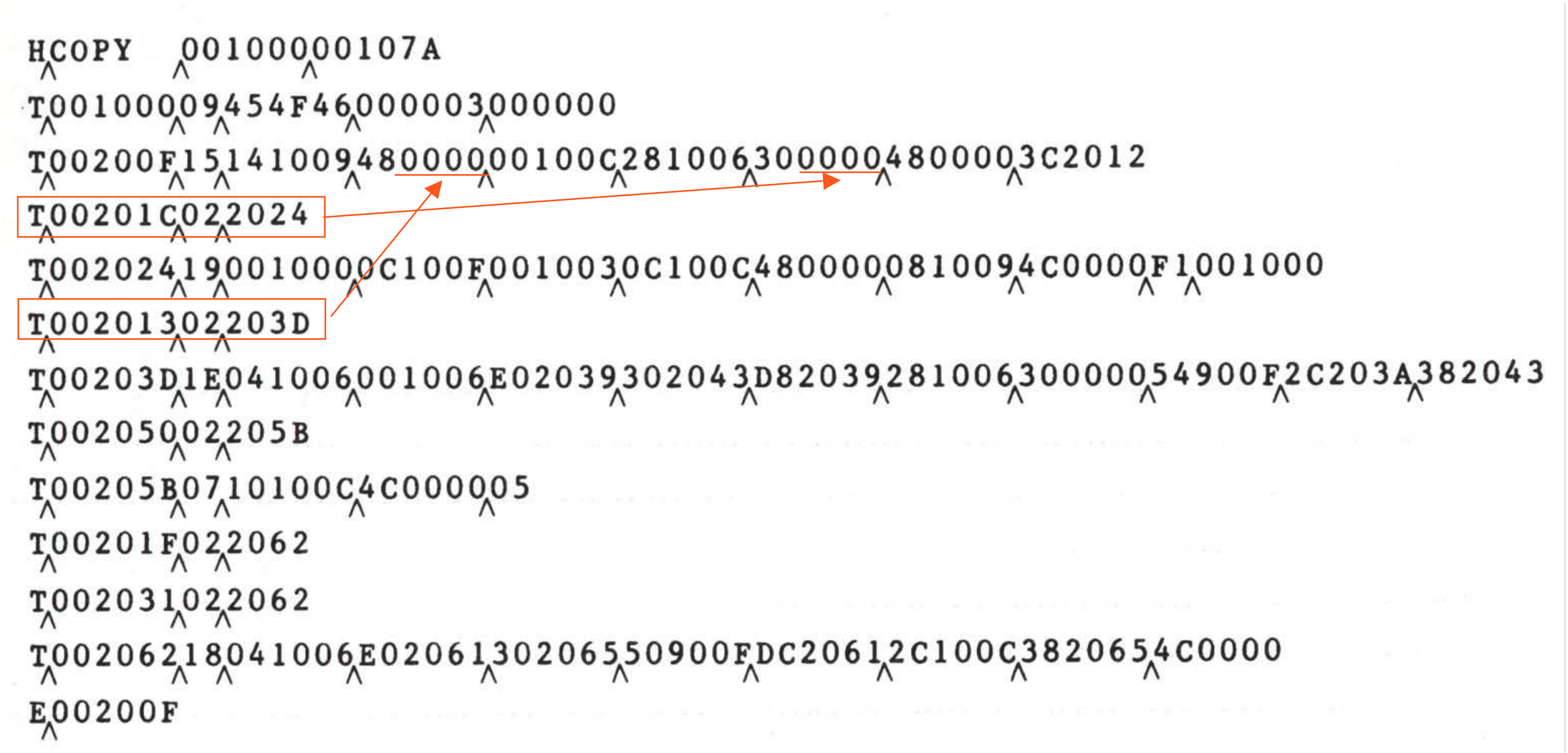
```
Memory
address        Contents

1000     454F4600  00030000  00xxxxxx  xxxxxxxx
1010     xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxxxx
  •
  •
  •
2000     xxxxxxxx  xxxxxxxx  xxxxxxxx  xxxxxx14
2010     10094820  3D00100C  28100630  202448--
2020     --3C2012  0010000C  100F0010  030C100C
2030     48------08  10094C00  00F10010  00041006
2040     001006E0  20393020  43D82039  28100630
2050     ----5490  0F
  •
  •
  •
```

Symbol	Value		
LENGTH	100C		
RDREC	203D		
THREE	1003		
ZERO	1006		
WRREC	*	201F	2031 ∅
EOF	1000		
ENDFIL	2024		
RETADR	1009		
BUFFER	100F		
CLOOP	2012		
FIRST	200F		
MAXLEN	203A		
INPUT	2039		
EXIT	*	2050 ∅	
RLOOP	2043		

If One-Pass Assemblers Need to Produce Object Codes

- If the operand contains an undefined symbol, use 0 as the address and write the Text record to the object program.
- Forward references are entered into lists as in the load-and-go assembler.
- When the definition of a symbol is encountered, the assembler generates another Text record with the correct operand address of each entry in the reference list.
- When loaded, the incorrect address 0 will be updated by the latter Text record containing the symbol definition.

Object code generated by one-pass assembler Figure 2.18, pp.97



```
H^COPY  ^001000^00107A
T^001000^09^454F46^000003^000000
T^00200F^15^141009^480000^00100C^281006^300000^480000^3C2012
T^00201C^02^2024
T^002024^19^001000^0C100F^001003^0C100C^480000^081009^4C0000^F1^001000
T^002013^02^203D
T^00203D^1E^041006^001006^E02039^302043^D82039^281006^300000^54900F^2C203A^382043
T^002050^02^205B
T^00205B^07^10100C^4C0000^05
T^00201F^02^2062
T^002031^02^2062
T^002062^18^041006^E02061^302065^50900F^DC2061^2C100C^382065^4C0000
E^00200F
```

Multi-Pass Assemblers

- For a two pass assembler, forward references in symbol definition are not allowed:

```
ALPHA     EQU      BETA
BETA      EQU      DELTA
DELTA     RESW   1
```

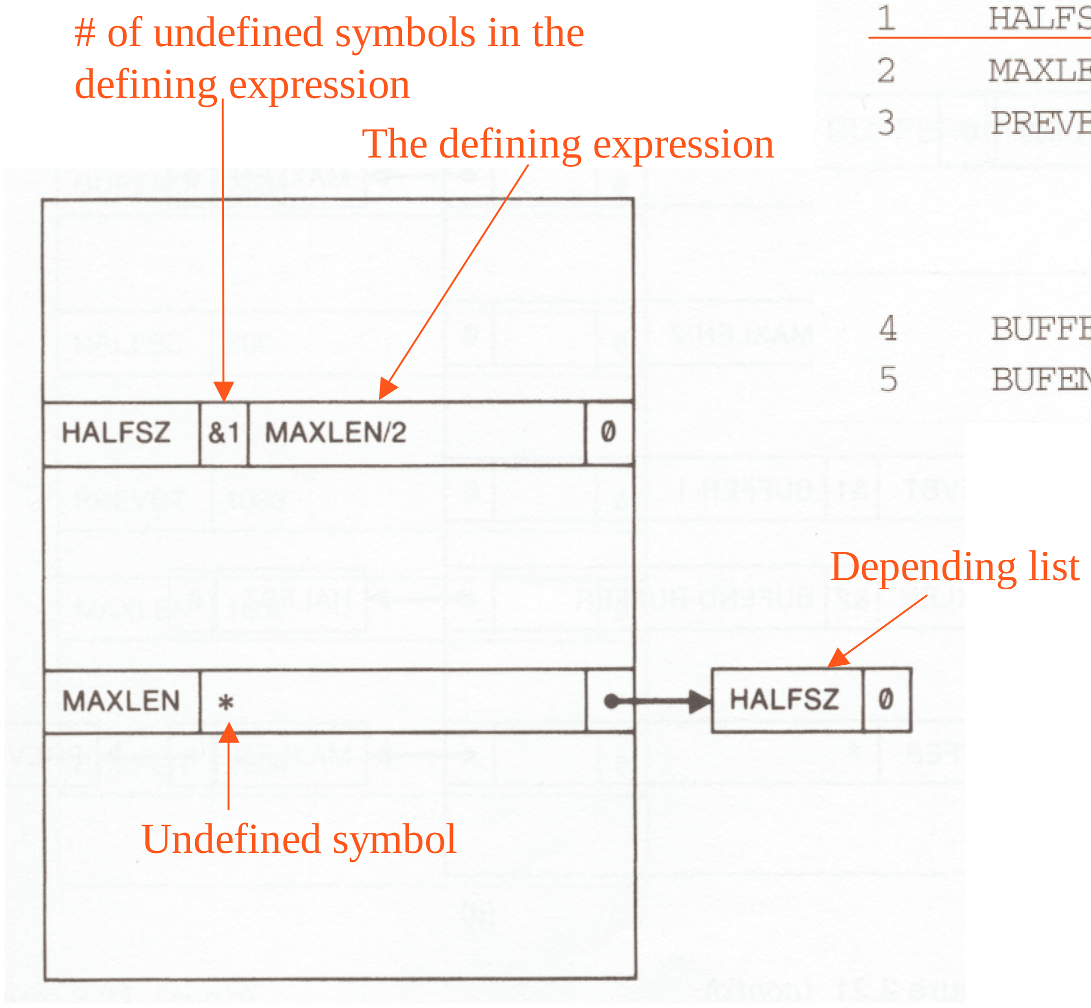
  - Symbol definition must be completed in pass 1.
- Prohibiting forward references in symbol definition is not a serious inconvenience.
  - Forward references tend to create difficulty for a person reading the program.

Implementation

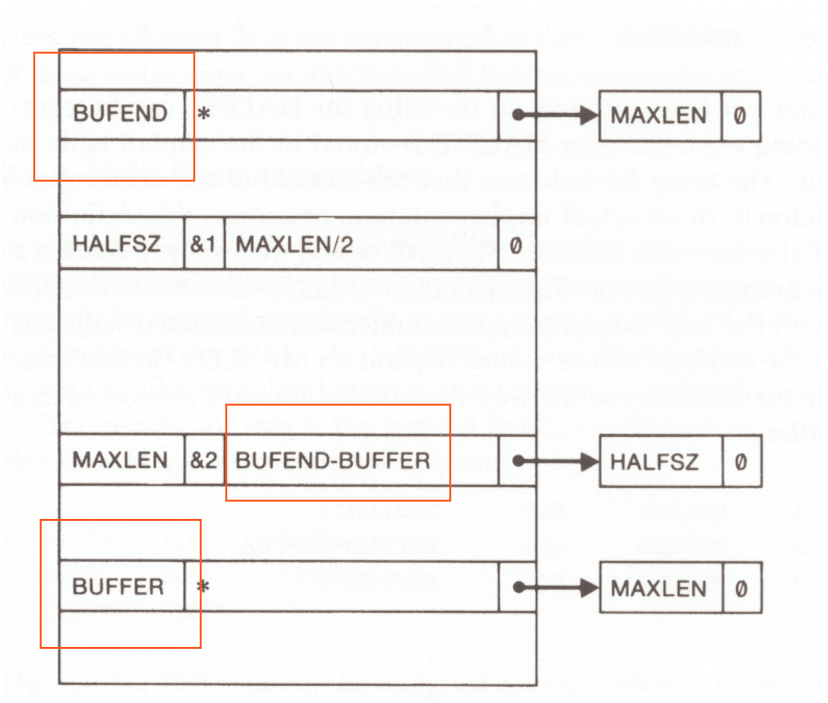
- For a forward reference in symbol definition, we store in the SYMTAB:
  - The symbol name
  - The defining expression
  - The number of undefined symbols in the defining expression
- The undefined symbol (marked with a flag *) associated with a list of symbols depend on this undefined symbol.
- When a symbol is defined, we can recursively evaluate the symbol expressions depending on the newly defined symbol.

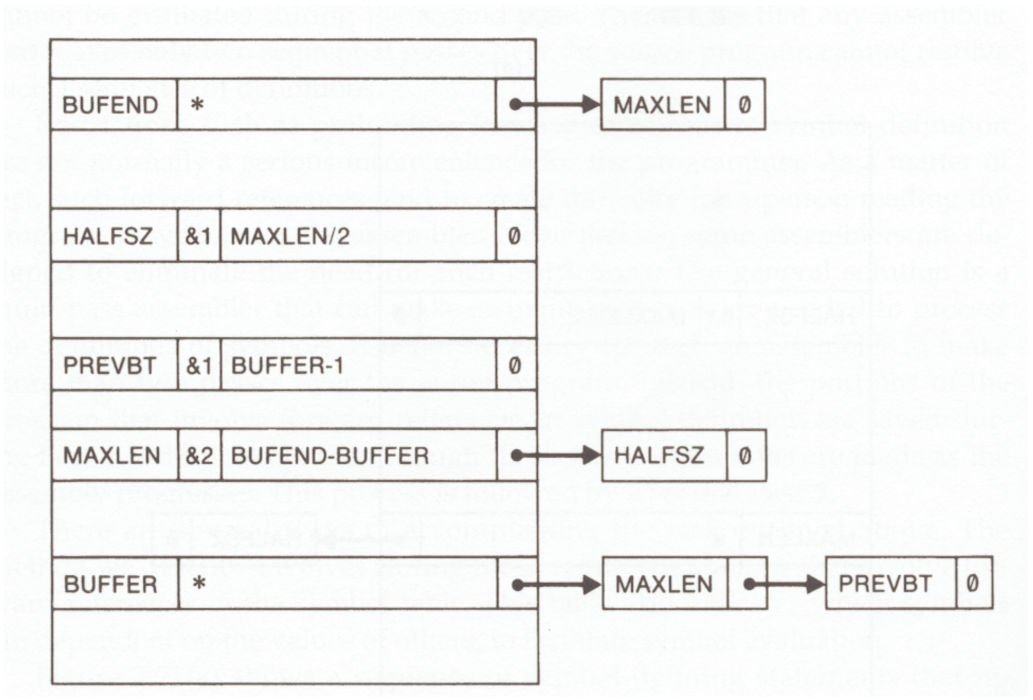
Multi-pass assembler example
Figure 2.21, pp. 99-101

```
1     HALFSZ    EQU     MAXLEN/2
2     MAXLEN    EQU     BUFEND-BUFFER
3     PREVBT    EQU     BUFFER-1
                        .
                        .
                        .
4     BUFFER    RESB    4096
5     BUFEND    EQU     *
```



Multi-pass assembler example Figure 2.21, pp. 99-101

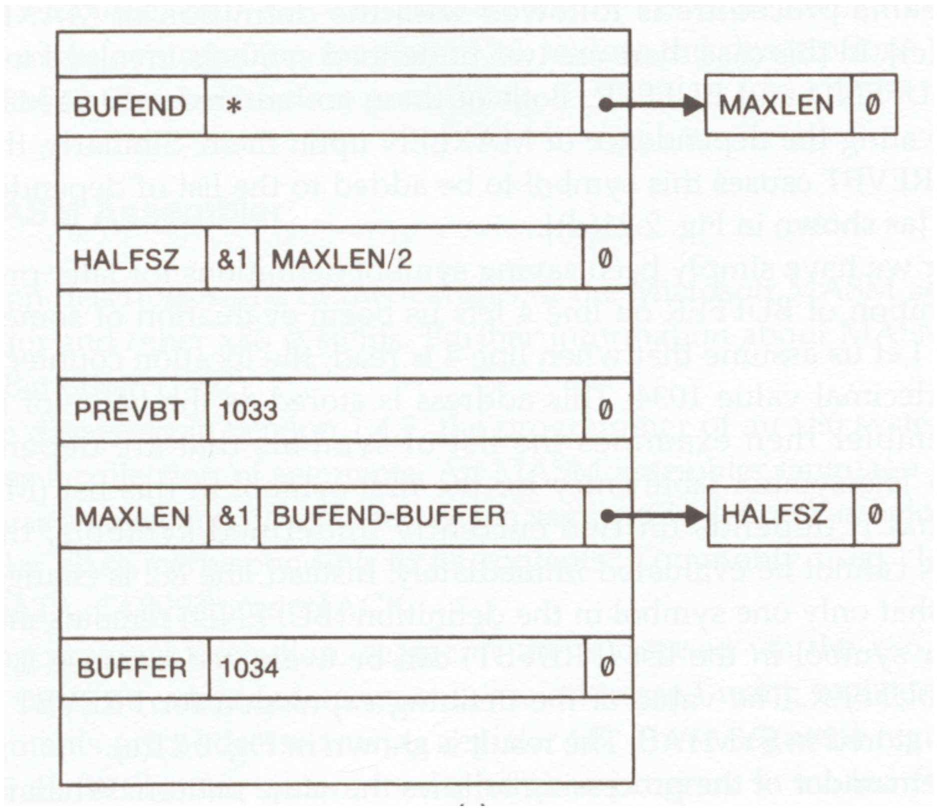




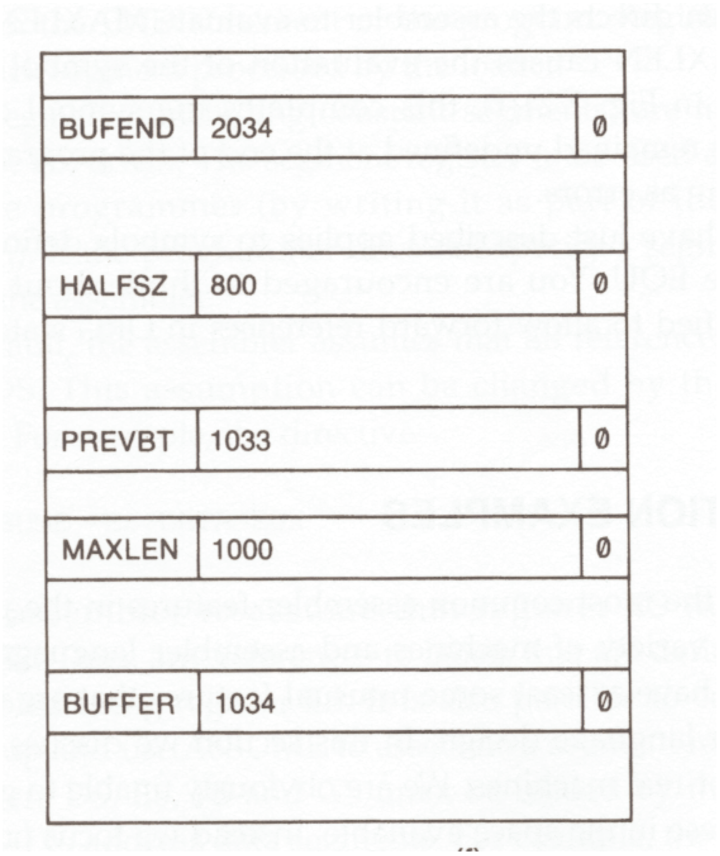
```
2   MAXLEN   EQU   BUFEND-BUFFER
```

```
3   PREVBT   EQU   BUFFER-1
```

Multi-pass assembler example Figure 2.21, pp. 99-101



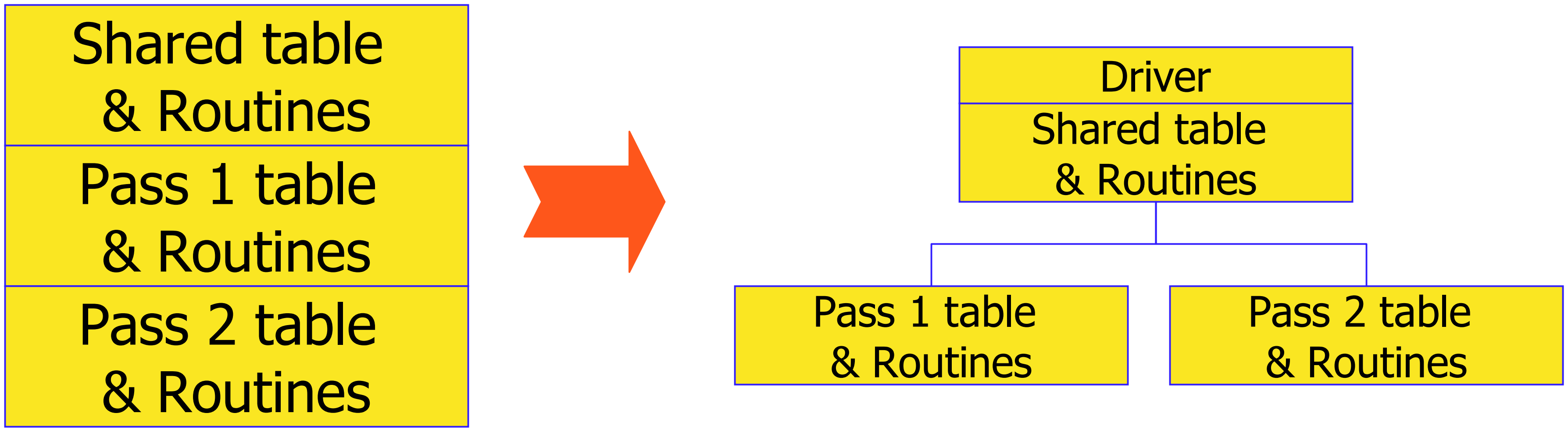
4 BUFFER RESB 4096



5 BUFEND EQU *

Two-pass assembler with overlay structure

- When memory is not enough
  - Pass 1 and pass 2 are never required at the same time
  - Three segments
  - Overlay program


Shared table & Routines
Pass 1 table & Routines
Pass 2 table & Routines
Driver
Shared table & Routines
Pass 1 table & Routines
Pass 2 table & Routines