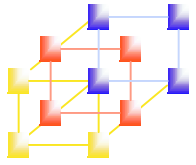


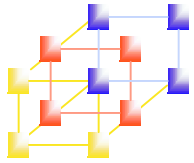
Chapter 2 Assemblers

-- 2.4 Assembler Design Options



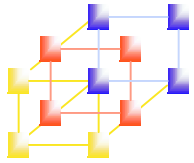
Outline

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



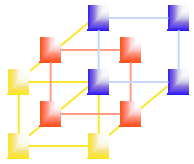
Load-and-Go Assembler

- Load-and-go assembler generates their object code in memory for immediate execution.
- No object program is written out, no loader is needed.
- It is useful in a system with frequent program development and testing
 - 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

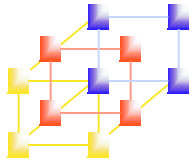
- Scenario for one-pass assemblers
 - Generate their object code in memory for immediate execution – *load-and-go* assembler
 - External storage for the intermediate file between two passes is slow or is inconvenient to use
- Main problem - Forward references
 - Data items
 - Labels on instructions
- 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

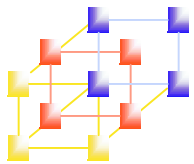
Figure 2.18, pp. 94

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 <u>RDREC</u>	48203D
20	2015	LDA LENGTH	00100C
25	2018	COMP ZERO	281006
30	201B	JEQ <u>ENDFIL</u>	302024
35	201E	JSUB <u>WRREC</u>	482062
40	2021	J CLOOP	302012
45	2024	<u>ENDFIL</u> LDA EOF	001000
50	2027	STA BUFFER	0C100F
55	202A	LDA THREE	001003
60	202D	STA LENGTH	0C100C
65	2030	JSUB <u>WRREC</u>	482062
70	2033	LDL RETADR	081009
75	2036	RSUB	4C0000
110			



Forward Reference in One-pass Assembler

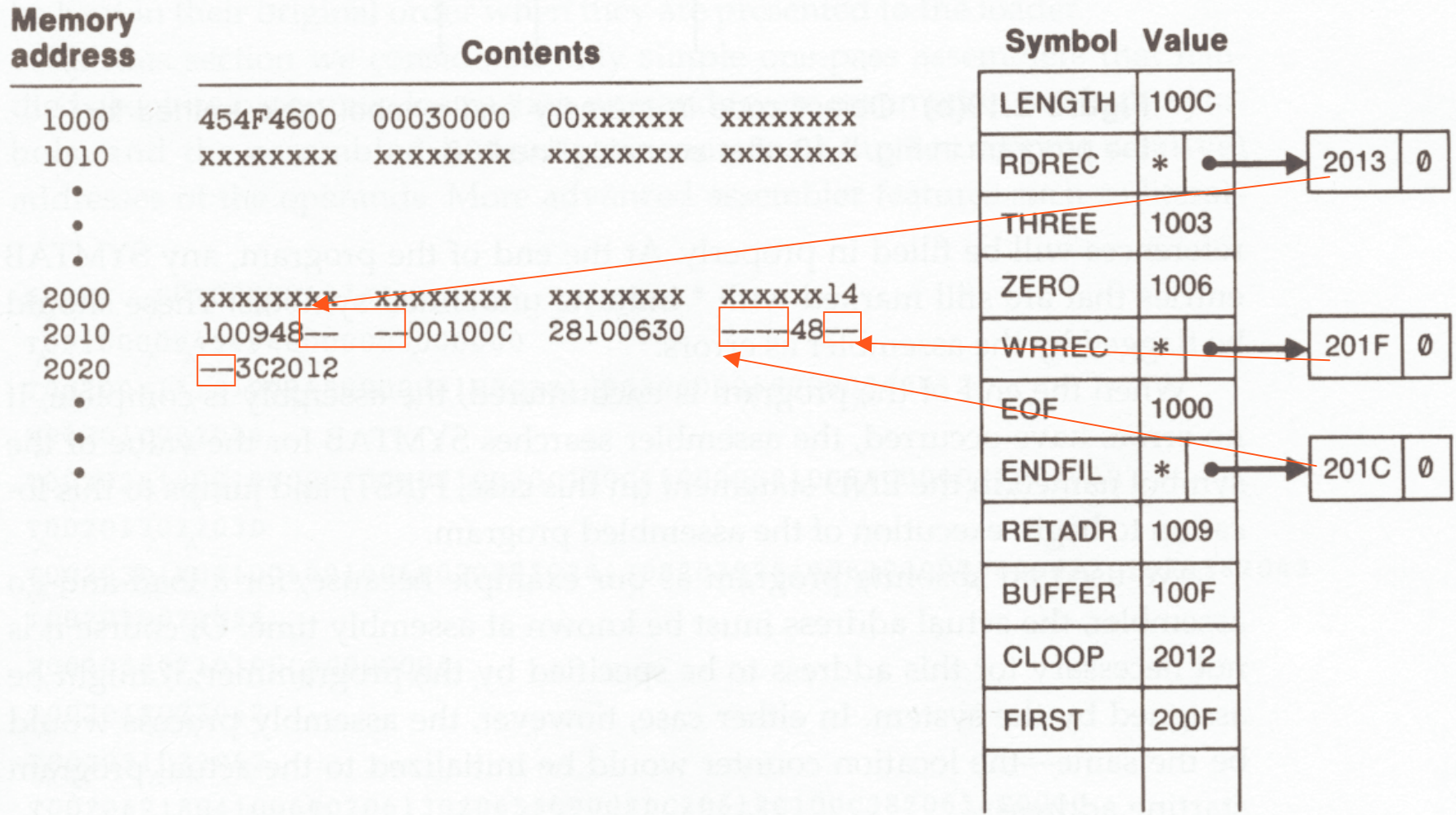
- Omits the operand address if the symbol has not yet been defined
- Enters this undefined symbol into SYMTAB and indicates that it is undefined
- Adds the address of this operand address to a list of forward references associated with the SYMTAB entry
- When the definition for the symbol is encountered, 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

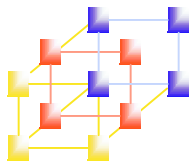


Object Code in Memory and SYMTAB

Figure 2.19(a), pp.95

After scanning line 40 of the program in Fig. 2.18

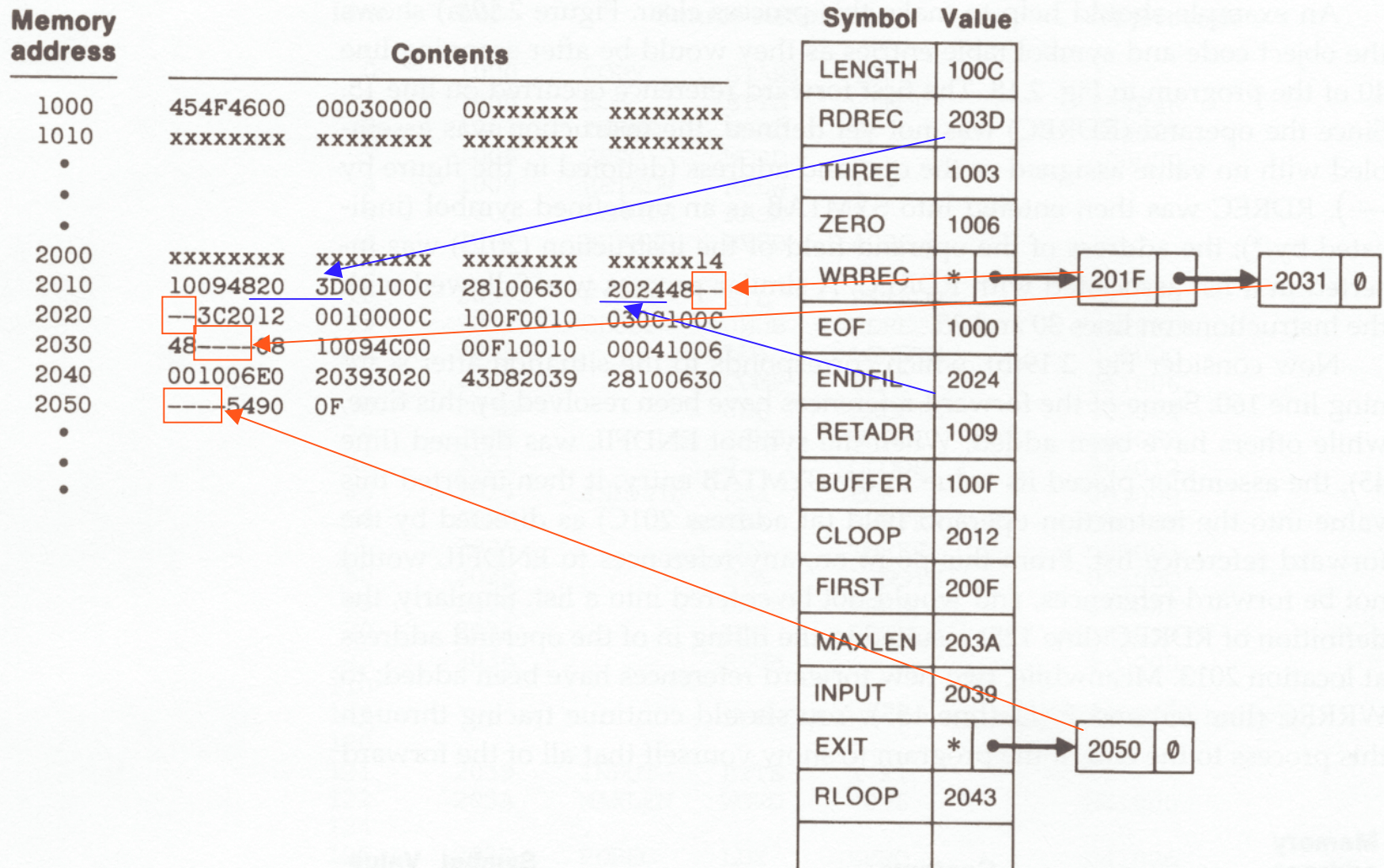


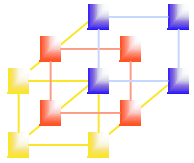


Object Code in Memory and SYMTAB

Figure 2.19(b), pp.96

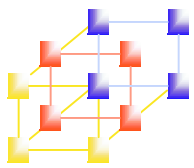
After scanning line 160 of the program in Fig. 2.18





If One-Pass Assemblers Need to Produce Object Codes

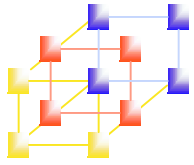
- If the operand contains an undefined symbol, use 0 as the address and write the Text record to the object program.
- Forward references are entered into lists as in the load-and-go assembler.
- When the definition of a symbol is encountered, the assembler generates another Text record with the correct 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

```
HCOPY  ^00100000^107A
T0010000^9454F4600000^3000000
T00200F1514100948000000^100C2810063000000^4800000^3C2012
T00201C022024
T002024190010000C100F0010030C100C4800000^810094C0000F1001000
T00201302203D
T00203D1E041006001006E02039302043D820392810063000000^54900F2C203A382043
T00205002205B
T00205B0710100C4C0000005
T00201F022062
T002031022062
T00206218041006E0206130206550900FDC20612C100C3820654C0000
E00200F
```

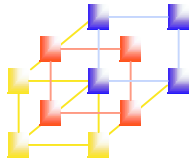


Multi-Pass Assemblers

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

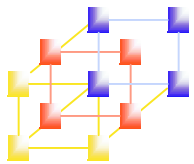
```
ALPHA      EQU      BETA
BETA       EQU      DELTA
DELTA      RESW    1
```

- 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.

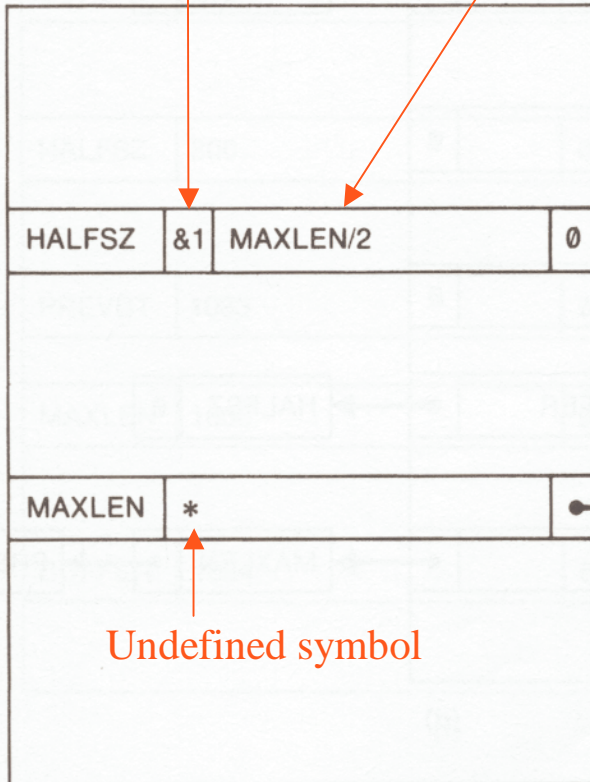


Multi-pass assembler example

Figure 2.21, pp. 99-101

of undefined symbols in the defining expression

The defining expression

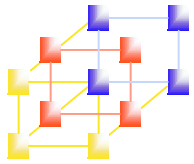


Undefined symbol

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

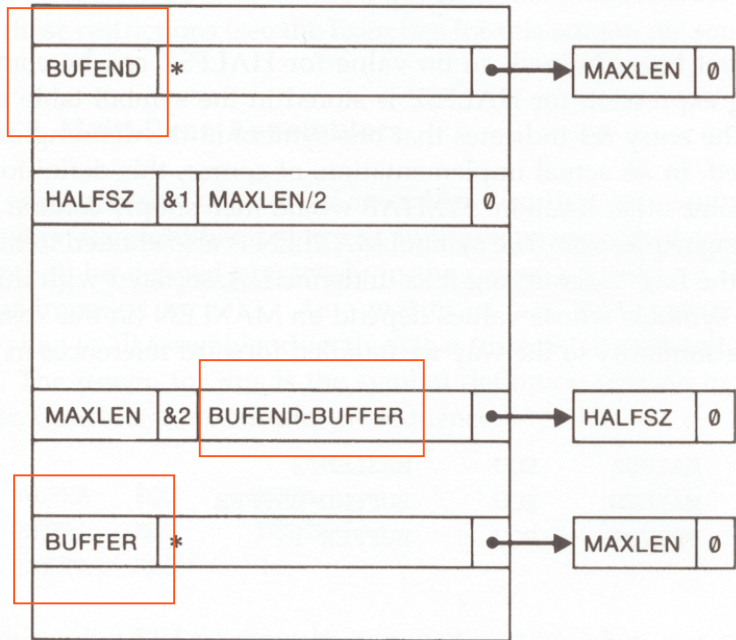
Depending list



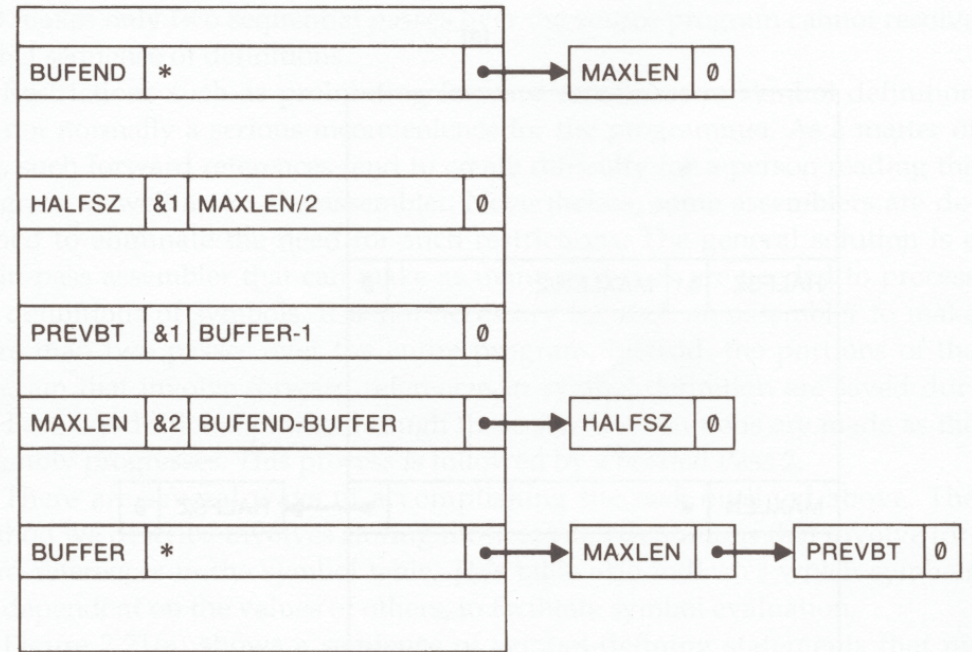


Multi-pass assembler example

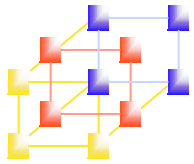
Figure 2.21, pp. 99-101



2 MAXLEN EQU BUFEND-BUFFER

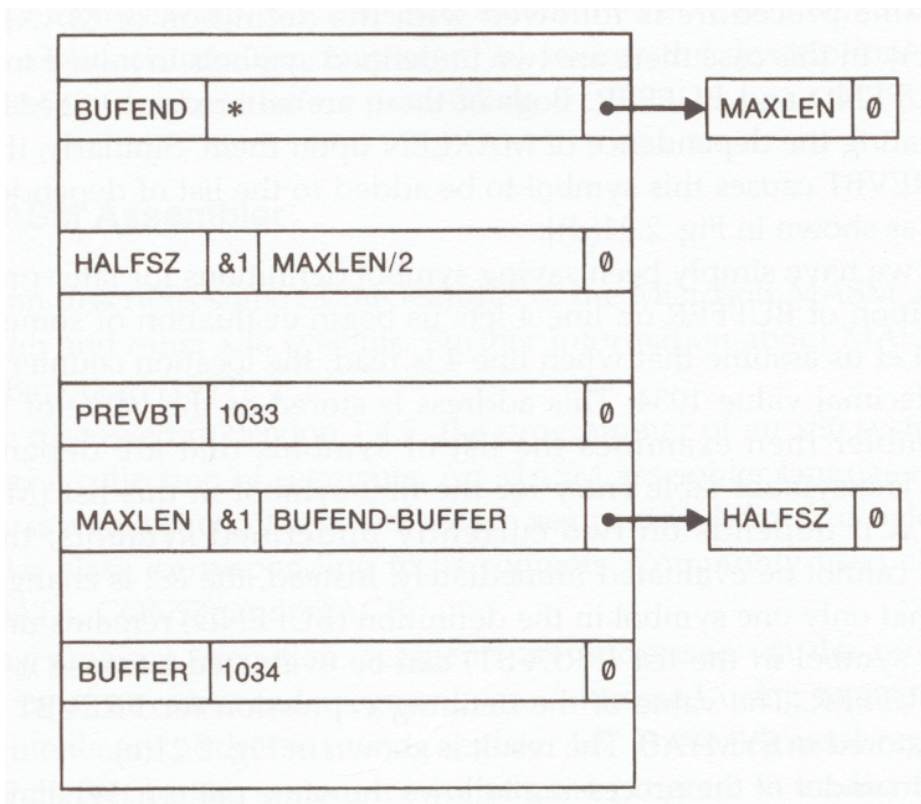


3 PREVBT EQU BUFFER-1

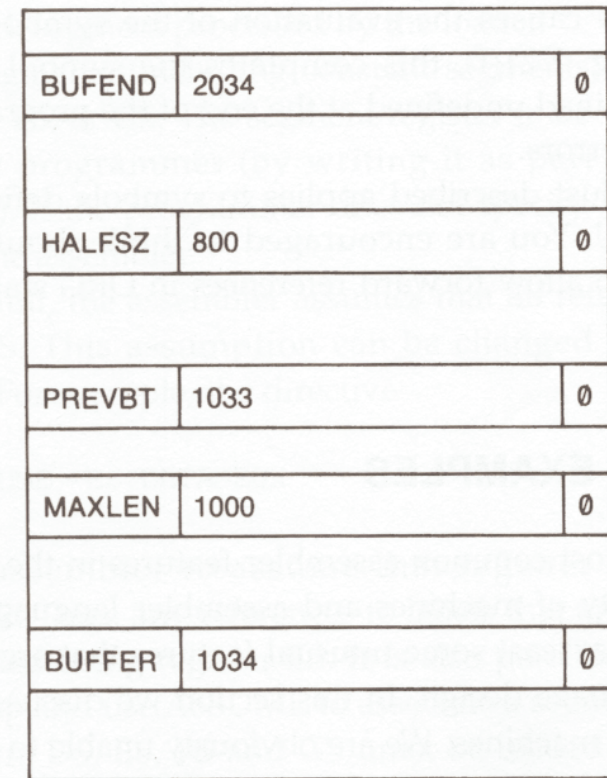


Multi-pass assembler example

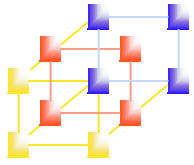
Figure 2.21, pp. 99-101



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

