identity after assembly; each such control section can be loaded and relocated independently of the others. Different control sections are most often used for subroutines or other logical subdivisions of a program. The programmer can assemble, load, and manipulate each of these control sections separately. The resulting flexibility is a major benefit of using control sections. We consider examples of this when we discuss linkage editors in Chapter 3.

When control sections form logically related parts of a program, it is necessary to provide some means for *linking* them together. For example, instructions in one control section might need to refer to instructions or data located in another section. Because control sections are independently loaded and relocated, the assembler is unable to process these references in the usual way. The assembler has no idea where any other control section will be located at execution time. Such references between control sections are called *external references.* The assembler generates information for each external reference that will allow the loader to perform the required linking. In this section we describe how external references are handled by our assembler. Chapter 3 discusses in detail how the actual linking is performed.

Figure 2.15 shows our example program as it might be written using multiple control sections. In this case there are three control sections: one for the main program and one for each subroutine. The START statement identifies the beginning of the assembly and gives a name (COPY) to the first control section. The first section continues until the CSECT statement on line 109. This assembler directive signals the start of a new control section named RDREC. Similarly, the CSECT statement on line 193 begins the control section named WRREC. The assembler establishes a separate location counter (beginning at 0) for each control section, just as it does for program blocks.

Control sections differ from program blocks in that they are handled separately by the assembler. (It is not even necessary for all control sections in a program to be assembled at the same time.) Symbols that are defined in one control section may not be used directly by another control section; they must be identified as external references for the loader to handle. Figure 2.15 shows the use of two assembler directives to identify such references: EXTDEF (external definition) and EXTREF (external reference). The EXTDEF statement in a control section names symbols, called *external symbols,* that are defined in this control section and may be used by other sections. Control section names (in this case COPY, RDREC, and WRREC) do not need to be named in an EXTDEF statement because they are automatically considered to be external symbols. The EXTREF statement names symbols that are used in this control section and are defined elsewhere. For example, the symbols BUFFER, BUFEND, and LENGTH are defined in the control section named COPY and made available to the other sections by the EXTDEF statement on line 6. The third control section (WRREC) uses two of these symbols, as specified in its EXTREF statement

Source statement

```
COPY     START   0            COPY FILE FROM INPUT TO OU
         EXTDEF  BUFFER,BUFEND,LENGTH
         EXTREF  RDREC,WRREC
FIRST    STL     RETADR       SAVE RETURN ADDRESS
CLOOP    +JSUB   RDREC        READ INPUT RECORD
         LDA     LENGTH       TEST FOR EOF (LENGTH = 0)
         COMP    #0
         JEQ     ENDFIL       EXIT IF EOF FOUND
         +JSUB   WRREC        WRITE OUTPUT RECORD
         J       CLOOP        LOOP
ENDFIL   LDA     =C'EOF'      INSERT END OF FILE MARKER
         STA     BUFFER
         LDA     #3           SET LENGTH = 3
         STA     LENGTH
         +JSUB   WRREC        WRITE EOF
         J       @RETADR      RETURN TO CALLER
RETADR   RESW    1
LENGTH   RESW    1            LENGTH OF RECORD
         LTORG
BUFFER   RESB    4096         4096-BYTE BUFFER AREA
BUFEND   EQU     *
MAXLEN   EQU     BUFEND-BUFFER

RDREC    CSECT
.
.        SUBROUTINE TO READ RECORD INTO BUFFER
.
         EXTREF  BUFFER,LENGTH,BUFEND
         CLEAR   X            CLEAR LOOP COUNTER
         CLEAR   A            CLEAR A TO ZERO
         CLEAR   S            CLEAR S TO ZERO
         LDT     MAXLEN
RLOOP    TD      INPUT        TEST INPUT DEVICE
         JEQ     RLOOP        LOOP UNTIL READY
         RD      INPUT        READ CHARACTER INTO REGIS
         COMPR   A,S          TEST FOR END OF RECORD (X
         JEQ     EXIT         EXIT LOOP IF EOR
         +STCH   BUFFER,X     STORE CHARACTER IN BUFFER
         TIXR    T            LOOP UNLESS MAX LENGTH
         JLT     RLOOP          HAS BEEN REACHED
EXIT     +STX    LENGTH       SAVE RECORD LENGTH
         RSUB                 RETURN TO CALLER
INPUT    BYTE    X'F1'        CODE FOR INPUT DEVICE
MAXLEN   WORD    BUFEND-BUFFER

WRREC    CSECT
.
.        SUBROUTINE TO WRITE RECORD FROM BUFFER
.
         EXTREF  LENGTH,BUFFER
         CLEAR   X            CLEAR LOOP COUNTER
         +LDT    LENGTH
WLOOP    TD      =X'05'       TEST OUTPUT DEVICE
         JEQ     WLOOP        LOOP UNTIL READY
         +LDCH   BUFFER,X     GET CHARACTER FROM BUFFER
         WD      =X'05'       WRITE CHARACTER
         TIXR    T            LOOP UNTIL ALL CHARACTERS
         JLT     WLOOP          HAVE BEEN WRITTEN
         RSUB                 RETURN TO CALLER
         END     FIRST
```

Figure 2.15 Illustration of control sections and program linking.

Now we are ready to look at how external references are handled by the assembler. Figure 2.16 shows the generated object code for each statement in the program. Consider first the instruction

```
15      0003    CLOOP   +JSUB   RDREC           4B100000
```

The operand (RDREC) is named in the EXTREF statement for the control section, so this is an external reference. The assembler has no idea where the control section containing RDREC will be loaded, so it cannot assemble the address for this instruction. Instead the assembler inserts an address of zero and passes information to the loader, which will cause the proper address to be inserted at load time. The address of RDREC will have no predictable relationship to anything in this control section; therefore relative addressing is not possible. Thus an extended format instruction must be used to provide room for the actual address to be inserted. This is true of any instruction whose operand involves an external reference.

Similarly, the instruction

```
160     0017            +STCH   BUFFER,X          57900000
```

makes an external reference to BUFFER. The instruction is assembled using extended format with an address of zero. The *x* bit is set to 1 to indicate indexed addressing, as specified by the instruction. The statement

```
190     0028    MAXLEN  WORD    BUFEND-BUFFER     000000
```

is only slightly different. Here the value of the data word to be generated is specified by an expression involving two external references: BUFEND and BUFFER. As before, the assembler stores this value as zero. When the program is loaded, the loader will add to this data area the address of BUFEND and subtract from it the address of BUFFER, which results in the desired value.

Note the difference between the handling of the expression on line 190 and the similar expression on line 107. The symbols BUFEND and BUFFER are defined in the same control section with the EQU statement on line 107. Thus the value of the expression can be calculated immediately by the assembler. This could not be done for line 190; BUFEND and BUFFER are defined in another control section, so their values are unknown at assembly time.

As we can see from the above discussion, the assembler must remember (via entries in SYMTAB) in which control section a symbol is defined. Any attempt to refer to a symbol in another control section must be flagged as an error unless the symbol is identified (using EXTREF) as an external reference. The assembler must also allow the same symbol to be used in different control

Source statement				Objec
COPY	START	0		
	EXTDEF	BUFFER,BUFEND,LENGTH		
	EXTREF	RDREC,WRREC		
FIRST	STL	RETADR		17202
CLOOP	+JSUB	RDREC		4B100
	LDA	LENGTH		03202
	COMP	#0		29000
	JEQ	ENDFIL		33200
	+JSUB	WRREC		4B100
	J	CLOOP		3F2FE
ENDFIL	LDA	=C'EOF'		03201
	STA	BUFFER		0F201
	LDA	#3		01000
	STA	LENGTH		0F200
	+JSUB	WRREC		4B100
	J	@RETADR		3E200
RETADR	RESW	1		
LENGTH	RESW	1		
	LTORG			
*	=C'EOF'			
BUFFER	RESB	4096		454F46
BUFEND	EQU	*		
MAXLEN	EQU	BUFEND-BUFFER		
RDREC	CSECT			
	SUBROUTINE TO READ RECORD INTO B			
	EXTREF	BUFFER,LENGTH,BUFEND		
	CLEAR	X		B410
	CLEAR	A		B400
	CLEAR	S		B440
RLOOP	LDT	MAXLEN		77201F
	TD	INPUT		E3201B
	JEQ	RLOOP		332FFA
	RD	INPUT		DB2015
	COMPR	A,S		A004
	JEQ	EXIT		332009
	+STCH	BUFFER,X		579000
	TIXR	T		B850
	JLT	RLOOP		3B2FE9
EXIT	+STX	LENGTH		131000
	RSUB			4F0000
INPUT	BYTE	X'F1'		F1
MAXLEN	WORD	BUFEND-BUFFER		000000
WRREC	CSECT			
	SUBROUTINE TO WRITE RECORD FROM B			
	EXTREF	LENGTH,BUFFER		
	CLEAR	X		B410
	+LDT	LENGTH		7710000
WLOOP	TD	=X'05'		E32012
	JEQ	WLOOP		332FFA
	+LDCH	BUFFER,X		5390000
	WD	=X'05'		DF2008
	TIXR	T		B850
	JLT	WLOOP		3B2FEE
	RSUB			4F0000
*	END	FIRST		
	=X'05'			05

Figure 2.16 Program from Fig. 2.15 with object code.

sections. For example, the conflicting definitions of MAXLEN on lines 107 and 190 should cause no problem. A reference to MAXLEN in the control section COPY would use the definition on line 107, whereas a reference to MAXLEN in RDREC would use the definition on line 190.

So far we have seen how the assembler leaves room in the object code for the values of external symbols. The assembler must also include information in the object program that will cause the loader to insert the proper values where they are required. We need two new record types in the object program and a change in a previously defined record type. As before, the exact format of these records is arbitrary; however, the same information must be passed to the loader in some form.

The two new record types are Define and Refer. A Define record gives information about external symbols that are defined in this control section—that is, symbols named by EXTDEF. A Refer record lists symbols that are used as external references by the control section—that is, symbols named by EXTREF. The formats of these records are as follows.

Define record:

Col. 1	D
Col. 2–7	Name of external symbol defined in this control section
Col. 8–13	Relative address of symbol within this control section (hexadecimal)
Col. 14–73	Repeat information in Col. 2–13 for other external symbols

Refer record:

Col. 1	R
Col. 2–7	Name of external symbol referred to in this control section
Col. 8–73	Names of other external reference symbols

The other information needed for program linking is added to the Modification record type. The new format is as follows.

Modification record (revised):

Col. 1	M
Col. 2–7	Starting address of the field to be modified, relative to the beginning of the control section (hexadecimal)
Col. 8–9	Length of the field to be modified, in half-bytes (hexadecimal)

Col. 10	Modification flag (+ or –)
Col. 11–16	External symbol whose value is to be added to c tracted from the indicated field

The first three items in this record are the same as previously discusse two new items specify the modification to be performed: adding or su ing the value of some external symbol. The symbol used for modificatic be defined either in this control section or in another one.

Figure 2.17 shows the object program corresponding to the sou Fig. 2.16. Notice that there is a separate set of object program records

```
H COPY  000000001033
T BUFFER000033BUFEND001033LENGTH00002D
H RDREC WRREC
T 000001D172027 4B100000320232900003320074B100000 3F2FEC032016
T 00001D0D010003 0F200A4B100000 3E2000
T 000030034 54F46
M 000004 05+RDREC
M 000011 05+WRREC
M 000024 05+WRREC
E 000000

H RDREC 00000000002B
R BUFFER LENGTH BUFEND
T 000001DB410B400B44077201FE3201B332FFADB2015A004332009579000
T 00001D0E3B2FE9 131000004F0000F1000000
M 000018 05+BUFFER
M 000021 05+LENGTH
M 000028 06+BUFEND
M 000028 06-BUFFER
E

H WRREC 00000000001C
R LENGTH BUFFER
T 000000 1CB410 77100000E32012332FFA53900000DF2008B8503B2FEE4F0
M 000003 05+LENGTH
M 000007 05+BUFFER
E
```

Figure 2.17 Object program corresponding to Fig. 2.15

Header through End) for each control section. The records for each control section are exactly the same as they would be if the sections were assembled separately.

The Define and Refer records for each control section include the symbols named in the EXTDEF and EXTREF statements. In the case of Define, the record also indicates the relative address of each external symbol within the control section. For EXTREF symbols, no address information is available. These symbols are simply named in the Refer record.

Now let us examine the process involved in linking up external references, beginning with the source statements we discussed previously. The address field for the JSUB instruction on line 15 begins at relative address 0004. Its initial value in the object program is zero. The Modification record

```
M00000405+RDREC
```

in control section COPY specifies that the address of RDREC is to be added to this field, thus producing the correct machine instruction for execution. The other two Modification records in COPY perform similar functions for the instructions on lines 35 and 65. Likewise, the first Modification record in control section RDREC fills in the proper address for the external reference on line 160.

The handling of the data word generated by line 190 is only slightly different. The value of this word is to be BUFEND–BUFFER, where both BUFEND and BUFFER are defined in another control section. The assembler generates an initial value of zero for this word (located at relative address 0028 within control section RDREC). The last two Modification records in RDREC direct that the address of BUFEND be added to this field, and the address of BUFFER be subtracted from it. This computation, performed at load time, results in the desired value for the data word.

In Chapter 3 we discuss in detail how the required modifications are performed by the loader. At this time, however, you should be sure that you understand the concepts involved in the linking process. You should carefully examine the other Modification records in Fig. 2.17, and reconstruct for yourself how they were generated from the source program statements.

Note that the revised Modification record may still be used to perform program relocation. In the case of relocation, the modification required is adding the beginning address of the control section to certain fields in the object program. The symbol used as the name of the control section has as its value the required address. Since the control section name is automatically an external symbol, it is available for use in Modification records. Thus, for example, the Modification records from Fig. 2.8 are changed from


```
M00000705
M00001405
M00002705
```

to

```
M00000705+COPY
M00001405+COPY
M00002705+COPY
```

In this way, exactly the same mechanism can be used for program re and for program linking. There are more examples in the next chapter.

The existence of multiple control sections that can be relocated i dently of one another makes the handling of expressions slightly more cated. Our earlier definitions required that all of the relative term expression be paired (for an absolute expression), or that all except paired (for a relative expression). We must now extend this restriction ity that both terms in each pair must be relative within the same con tion. The reason is simple—if the two terms represent relative locatior same control section, their difference is an absolute value (regardless c the control section is located). On the other hand, if they are in differ trol sections, their difference has a value that is unpredictable (and t probably useless). For example, the expression

```
BUFEND-BUFFER
```

has as its value the length of BUFFER in bytes. On the other hand, the the expression

```
RDREC-COPY
```

is the difference in the load addresses of the two control sections. Tl depends on the way run-time storage is allocated; it is unlikely to b use whatsoever to an application program.

When an expression involves external references, the assembler c general determine whether or not the expression is legal. The pairing c terms to test legality cannot be done without knowing which of the ter in the same control sections, and this is unknown at assembly time. I case, the assembler evaluates all of the terms it can, and combines thes an initial expression value. It also generates Modification records so t can finish the evaluation. The loader can then check the expression for e discuss this further in Chapter 3 when we examine the design of a linkir