Assembler

- Assembler is system software which is used to convert an assembly language program to its equivalent object code.
- The input to the assembler is a source code written in assembly language (using mnemonics) and the output is the object code.
- The design of an assembler depends upon the machine architecture as the language used is mnemonic language.

Design of Assembler

- Convert mnemonic operation codes to their machine language equivalents
- Convert symbolic operands to their equivalent machine addresses
- Decide the proper instruction format
- Convert the data constants to internal machine representations
- Write the object program and the assembly listing
- The assembler design can be done:
  - Single pass assembler
  - Multi-pass assembler

Single Pass Assembler

- The whole process of scanning, parsing, and object code conversion is done in single pass.
- The only problem with this method is resolving forward reference.
- See example 2.1 later.

Multi Pass Assembler

- Resolves the forward references and then converts into the object code.
- Pass-1
  - Assign addresses to all the statements
  - Save the addresses assigned to all labels to be used in Pass-2
  - Perform some processing of assembler directives such as RESW, RESB to find the length of data areas for assigning the address values.
  - Defines the symbols in the symbol table(generate the symbol table)
- Pass-2
  - Assemble the instructions(translating operation codes and looking up addresses)
  - Generate data values defined by BYTE, WORD etc.
  - Perform the processing of the assembler directives not done during pass-1
  - Write the object program and assembler listing

Assembler Design

- The most important things which need to be concentrated is the generation of Symbol table and resolving forward references.

  Symbol Table:

  This is created during pass 1

  All the labels of the instructions are symbols

  Table has entry for symbol name, address value.

  Forward reference:

  Symbols that are defined in the later part of the program are called forward referencing.

  There will not be any address value for such symbols in the symbol table in pass 1.

Main Functions

- Translate mnemonic operation codes to their machine language equivalents
- Assign machine addresses to symbolic labels used by the programmers

Depend heavily on the source language it translates and the machine language it produces.

E.g., the instruction format and addressing modes

Example 2.1

Line numbers are not part of the program. They are for reference only.

```
  5     COPY      START   1000        COPY FILE FROM INPUT TO OUTPUT
 10     FIRST     STL     RETADR      SAVE RETURN ADDRESS
 15     CLOOP     JSUB    RDREC       READ INPUT RECORD
 20               LDA     LENGTH      TEST FOR EOF (LENGTH = 0)
 25               COMP    ZERO
 30               JEQ     ENDFIL      EXIT IF EOF FOUND
 35               JSUB    WRREC       WRITE OUTPUT RECORD
 40               J       CLOOP       LOOP
 45     ENDFIL    LDA     EOF         INSERT END OF FILE MARKER
 50               STA     BUFFER
 55               LDA     THREE       SET LENGTH = 3
 60               STA     LENGTH
 65               JSUB    WRREC       WRITE EOF
 70               LDL     RETADR      GET RETURN ADDRESS
 75               RSUB                RETURN TO CALLER
 80     EOF       BYTE    C'EOF'
 85     THREE     WORD    3
 90     ZERO      WORD    0
 95     RETADR    RESW    1
100     LENGTH    RESW    1           LENGTH OF RECORD
105     BUFFER    RESB    4096        4096-BYTE BUFFER AREA
```

```
110     .
115     .       SUBROUTINE TO READ RECORD INTO BUFFER
120     .
125     RDREC   LDX     ZERO        CLEAR LOOP COUNTER
130             LDA     ZERO        CLEAR A TO ZERO
135     RLOOP   TD      INPUT       TEST INPUT DEVICE
140             JEQ     RLOOP       LOOP UNTIL READY
145             RD      INPUT       READ CHARACTER INTO REGISTER A
150             COMP    ZERO        TEST FOR END OF RECORD (X'00')
155             JEQ     EXIT        EXIT LOOP IF EOR
160             STCH    BUFFER,X    STORE CHARACTER IN BUFFER
165             TIX     MAXLEN      LOOP UNLESS MAX LENGTH
170             JLT     RLOOP         HAS BEEN REACHED
175     EXIT    STX     LENGTH      SAVE RECORD LENGTH
180             RSUB                RETURN TO CALLER
185     INPUT   BYTE    X'F1'       CODE FOR INPUT DEVICE
190     MAXLEN  WORD    4096
195     .
```

```
195         .
200         .       SUBROUTINE TO WRITE RECORD FROM BUFFER
205         .
210         WRREC   LDX     ZERO            CLEAR LOOP COUNTER
215         WLOOP   TD      OUTPUT          TEST OUTPUT DEVICE
220                 JEQ     WLOOP           LOOP UNTIL READY
225                 LDCH    BUFFER,X        GET CHARACTER FROM BUFFER
230                 WD      OUTPUT          WRITE CHARACTER
235                 TIX     LENGTH          LOOP UNTIL ALL CHARACTERS
240                 JLT     WLOOP             HAVE BEEN WRITTEN
245                 RSUB                    RETURN TO CALLER
250         OUTPUT  BYTE    X'05'           CODE FOR OUTPUT DEVICE
255                 END     FIRST
```

Example 2.1

Line numbers are not part of the program. They are for reference only.

Forward reference

```
 5     COPY      START   1000          COPY FILE FROM INPUT TO OUTPUT
10     FIRST     STL     RETADR        SAVE RETURN ADDRESS
15     CLOOP     JSUB    RDREC         READ INPUT RECORD
20               LDA     LENGTH        TEST FOR EOF (LENGTH = 0)
25               COMP    ZERO
30               JEQ     ENDFIL        EXIT IF EOF FOUND
35               JSUB    WRREC         WRITE OUTPUT RECORD
40               J       CLOOP         LOOP
45     ENDFIL    LDA     EOF           INSERT END OF FILE MARKER
50               STA     BUFFER
55               LDA     THREE         SET LENGTH = 3
60               STA     LENGTH
65               JSUB    WRREC         WRITE EOF
70               LDL     RETADR        GET RETURN ADDRESS
75               RSUB                  RETURN TO CALLER
80     EOF       BYTE    C'EOF'
85     THREE     WORD    3
90     ZERO      WORD    0
95     RETADR    RESW    1
100    LENGTH    RESW    1             LENGTH OF RECORD
105    BUFFER    RESB    4096          4096-BYTE BUFFER AREA
```

Call subroutine

code

Comment line

```
110         .
115         .       SUBROUTINE TO READ RECORD INTO BUFFER
120         .
125     RDREC   LDX     ZERO        CLEAR LOOP COUNTER
130             LDA     ZERO        CLEAR A TO ZERO
135     RLOOP   TD      INPUT       TEST INPUT DEVICE
140             JEQ     RLOOP       LOOP UNTIL READY
145             RD      INPUT       READ CHARACTER INTO REGISTER A
150             COMP    ZERO        TEST FOR END OF RECORD (X'00')
155             JEQ     EXIT        EXIT LOOP IF EOR
160             STCH    BUFFER,X    STORE CHARACTER IN BUFFER
165             TIX     MAXLEN      LOOP UNLESS MAX LENGTH
170             JLT     RLOOP         HAS BEEN REACHED
175     EXIT    STX     LENGTH      SAVE RECORD LENGTH
180             RSUB                RETURN TO CALLER
185     INPUT   BYTE    X'F1'       CODE FOR INPUT DEVICE
190     MAXLEN  WORD    4096
195         .
```

Indexing mode

Hexadecimal number

Subroutine entry point

```
195      .
200      .        SUBROUTINE TO WRITE RECORD FROM BUFFER
205      .
210      WRREC    LDX     ZERO          CLEAR LOOP COUNTER
215      WLOOP    TD      OUTPUT        TEST OUTPUT DEVICE
220               JEQ     WLOOP         LOOP UNTIL READY
225               LDCH    BUFFER,X      GET CHARACTER FROM BUFFER
230               WD      OUTPUT        WRITE CHARACTER
235               TIX     LENGTH        LOOP UNTIL ALL CHARACTERS
240               JLT     WLOOP           HAVE BEEN WRITTEN
245               RSUB                  RETURN TO CALLER
250      OUTPUT   BYTE    X'05'         CODE FOR OUTPUT DEVICE
255               END     FIRST
```

Subroutine return point

Purpose of Example 2.1 (COPY)

- It is a copy function that reads some records from a specified input device and then copies them to a specified output device
  - Reads a record from the input device (code F1)
  - Copies the record to the output device (code 05)
  - Repeats the above steps until encountering EOF.
  - Then writes EOF to the output device
  - Then call RSUB to return to the caller

RDREC and WRREC

- Data transfer
  - A record is a stream of bytes with a null character (00_{16}) at the end.
  - If a record is longer than 4096 bytes, only the first 4096 bytes are copied.
  - EOF is indicated by a zero-length record. i.e., a byte stream with only a null character.
  - Because the speed of the input and output devices may be different, a buffer is used to temporarily store the record
- Subroutine call and return
  - On line 10, “STL RETADDR” is called to save the return address that is already stored in register L.
  - Otherwise, after calling RD or WR, this COPY cannot return back to its caller.

Assembler Directives

- Assembler directives are pseudo instructions
  - They will not be translated into machine instructions.
  - They only provide instruction/direction/information to the assembler.
  - No memory location, no object code.
- Basic assembler directives :
  - START :
    - Specify name and starting address for the program
  - END :
    - Indicate the end of the source program, and (optionally) the first executable instruction in the program.

Assembler Directives (cont'd)

- BYTE :
  - Generate character or hexadecimal constant, occupying as many bytes as needed to represent the constant.
- WORD :
  - Generate one-word integer constant
  - Reserve a word of memory and store constant in memory.
- RESB :
  - Reserve the indicated number of bytes for a data area.
  - First convert given value into hex
- RESW :
  - Reserve the indicated number of words for a data area
  - Do not store constant in memory

An Assembler's Job

- Convert mnemonic operation codes to their machine language codes
  - e.g. translate STL to 14 , JSUB to 48 etc
- Convert symbolic (e.g., jump labels, variable names) operands to their machine addresses
  - e.g. translate RETADR to 1033 , RDREC to 2039 etc
- Use proper addressing modes and formats to build efficient machine instructions
- Translate data constants into internal machine representations
  - e.g. translate EOF to 454F46
- Output the object program and provide other information (e.g., for linker and loader)

Object Program

- Object Program will be loaded in memory for execution.
- Three types of records:
  - Header: program name, starting address, length
  - Text: Starting address, length, object code
  - End: address of first executable code
- The symbol ^ is used to separate fields.

Object Program Format

- Header

Col. 1	H
Col. 2~7	Program name
Col. 8~13	Starting address of object program (hex)
Col. 14-19	Length of object program in bytes (hex)

- Text

Col.1	T
Col.2~7	Starting address for object code in this record (hex)
Col. 8~9	Length of object code in this record in bytes (hex)
Col. 10~69	Object code, represented in hexa (2 col. per byte)

- End

Col.1	E
Col.2~7	Address of first executable instruction in object program (hex)

Line	Loc	Source statement			Object code
5	1000	COPY	START	1000	
10	1000	FIRST	STL	RETADR	141033
15	1003	CLOOP	JSUB	RDREC	482039
20	1006		LDA	LENGTH	001036
25	1009		COMP	ZERO	281030
30	100C		JEQ	ENDFIL	301015
35	100F		JSUB	WRREC	482061
40	1012		J	CLOOP	3C1003
45	1015	ENDFIL	LDA	EOF	00102A
50	1018		STA	BUFFER	0C1039
55	101B		LDA	THREE	00102D
60	101E		STA	LENGTH	0C1036
65	1021		JSUB	WRREC	482061
70	1024		LDL	RETADR	081033
75	1027		RSUB		4C0000
80	102A	EOF	BYTE	C'EOF'	454F46
85	102D	THREE	WORD	3	000003
90	1030	ZERO	WORD	0	000000
95	1033	RETADR	RESW	1	
100	1036	LENGTH	RESW	1	
105	1039	BUFFER	RESB	4096	
110		.			

```
110                 .
115                 .     SUBROUTINE TO READ RECORD INTO BUFFER
120                 .
125     2039    RDREC     LDX     ZERO            041030
130     203C              LDA     ZERO            001030
135     203F    RLOOP     TD      INPUT           E0205D
140     2042              JEQ     RLOOP           30203F
145     2045              RD      INPUT           D8205D
150     2048              COMP    ZERO            281030
155     204B              JEQ     EXIT            302057
160     204E              STCH    BUFFER,X        549039
165     2051              TIX     MAXLEN          2C205E
170     2054              JLT     RLOOP           38203F
175     2057    EXIT      STX     LENGTH          101036
180     205A              RSUB                    4C0000
185     205D    INPUT     BYTE    X'F1'           F1
190     205E    MAXLEN    WORD    4096            001000
195
```

```
195                 .
200                 .    SUBROUTINE TO WRITE RECORD FROM BUFFER
205                 .
210     2061    WRREC    LDX     ZERO          041030
215     2064    WLOOP    TD      OUTPUT        E02079
220     2067             JEQ     WLOOP         302064
225     206A             LDCH    BUFFER,X      509039
230     206D             WD      OUTPUT        DC2079
235     2070             TIX     LENGTH        2C1036
240     2073             JLT     WLOOP         382064
245     2076             RSUB                  4C0000
250     2079    OUTPUT   BYTE    X'05'         05
255                      END     FIRST
```

The Object Code for COPY

H COPY 001000 00107A

T 001000 1E 141033 482039 001036 281030 301015 482061 3C1003
00102A 0C1039 00102D

T 00101E 15 0C1036 482061 081044 4C0000 454F46 000003 000000

T 002039 1E 041030 001030 E0205D 30203F D8205D 281030 302057
549039 2C205E 38203F

T 002057 1C 101036 4C0000 F1 001000 041030 E02079 302064 509039
DC2079 2C1036

T 002073 07 382064 4C0000 05

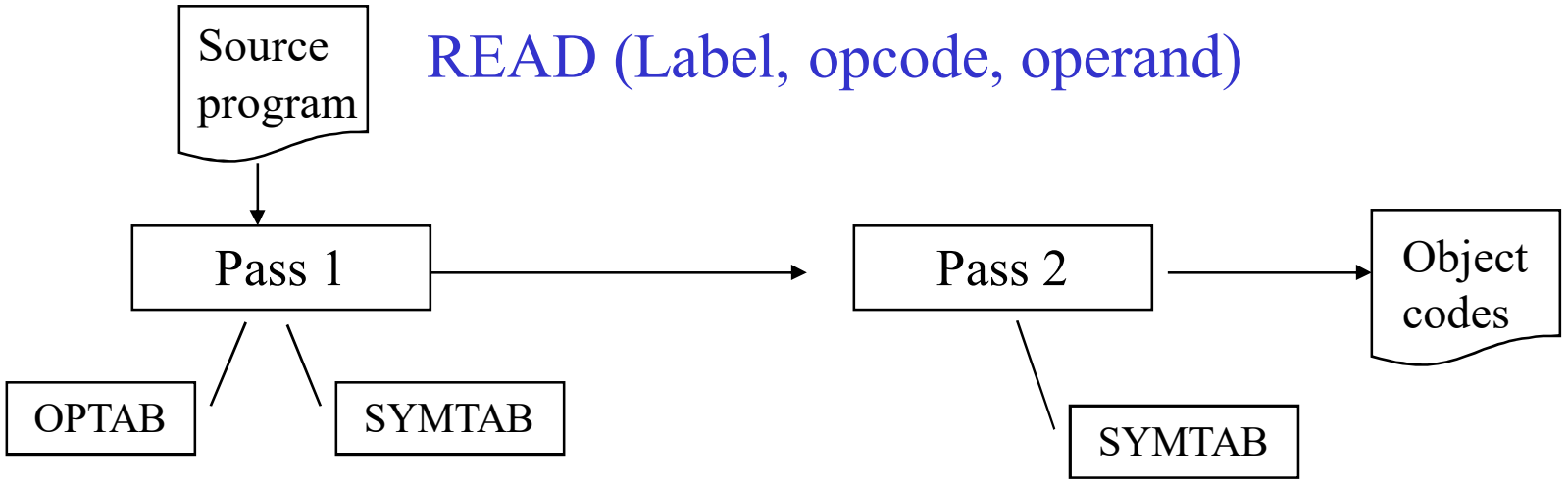
E 001000

There is no object code corresponding to addresses 1033-2038. This storage is simply reserved by the loader for use by the program during execution.

Two Pass Assembler

- Pass 1
  - Assign addresses to all statements in the program
  - Save the values (addresses) assigned to all labels (including label and variable names) for use in Pass 2 (deal with forward references)
  - Perform some processing of assembler directives (e.g., BYTE, RESW, these can affect address assignment)
- Pass 2
  - Assemble instructions (generate opcode and look up addresses)
  - Generate data values defined by BYTE, WORD
  - Perform processing of assembler directives not done in Pass 1
  - Write the object program and the assembly listing

A Simple Two Pass Assembler Implementation



Mnemonic and opcode mappings are referenced from here

Label and address mappings enter here

Label and address mappings are referenced from here

Three Main Data Structures

- Operation Code Table (OPTAB)
- Location Counter (LOCCTR)
- Symbol Table (SYMTAB)

OPTAB (operation code table)

- Content
  - The mapping between mnemonic and machine code. Also include the instruction format, available addressing modes, and length information.
- Characteristic
  - Static table. The content will never change.
- Implementation
  - Array or hash table. Because the content will never change, we can optimize its search speed.
- In pass 1, OPTAB is used to look up and validate mnemonics in the source program.
- In pass 2, OPTAB is used to translate mnemonics to machine instructions.

Location Counter (LOCCTR)

- This variable can help in the assignment of addresses.
- It is initialized to the beginning address specified in the START statement.
- After each source statement is processed, the length of the assembled instruction and data area to be generated is added to LOCCTR.
- Thus, when we reach a label in the source program, the current value of LOCCTR gives the address to be associated with that label.

Symbol Table (SYMTAB)

- Content
  - Include the label name and value (address) for each label in the source program.
  - Include type and length information (e.g., int64)
  - With flag to indicate errors (e.g., a symbol defined in two places)
- Characteristic
  - Dynamic table (I.e., symbols may be inserted, deleted, or searched in the table)
- Implementation
  - Hash table can be used to speed up search
  - Because variable names may be very similar (e.g., LOOP1, LOOP2), the selected hash function must perform well with such non-random keys.