Today’s Topic

• Section 2.3 of Beck’s “System Software” book -- Machine Independent Assembler Features
  – Literals
  – Symbol Defining Statements
  – Expressions
  – Program Blocks
  – Control Sections and Program Linking
Literals

• Design idea
  – Let programmers to be able to write the value of a constant operand as a part of the instruction that uses it.
  – This avoids having to define the constant elsewhere in the program and make up a label for it.

• Example (Fig. 2.10)

```
45  001A  ENDFIL  LDA =C'EOF'  032010
93  LTORG

002D  * =C'EOF'  454F46
215  1062  WLOOP  TD =X'05'  E32011
```

![Fig. 2.10](image-url)
Literals vs. Immediate Operands

• Immediate Operands
  – The operand value is assembled as part of the machine instruction
  – e.g. 55 0020 LDA #3 010003

• Literals
  – The assembler generates the specified value as a constant at some other memory location
  – e.g. 45 001A ENDFIL LDA =C’EOF’ 032010

• Compare (Fig. 2.6)
  – e.g. 45 001A ENDFIL LDA EOF 032010
  80 002D EOF BYTE C’EOF’ 454F46

Literal - Implementation (1/3)

• Literal pools
  – Normally literals are placed into a pool at the end of the program
    • See Fig. 2.10 (END statement)
      255 END FIRST
      1076 * =X’05’ 05
  – In some cases, it is desirable to place literals into a pool at some other location in the object program
    • Assembler directive LTORG
    • Reason: keep the literal operand close to the instruction
Literal - Implementation (2/3)

• Duplicate literals
  – e.g. 215 1062 WLOOP TD =X’05’
  – e.g. 230 106B WD =X’05’
  – The assemblers should recognize duplicate literals and store only one copy of the specified data value
    • Comparison of the defining expression
      – Same literal name with different value, e.g. LOCCTR=* 
    • Comparison of the generated data value
      – The benefits of using generate data value are usually not great enough to justify the additional complexity in the assembler

Literal - Implementation (3/3)

• LITTAB
  – Literal name, the operand value and length, the address assigned to the operand

• Pass 1
  – Build LITTAB with literal name, operand value and length, leaving the address unassigned
  – When LTORG statement is encountered, assign an address to each literal not yet assigned an address

• Pass 2
  – Search LITTAB for each literal operand encountered
  – Generate data values using BYTE or WORD statements
  – Generate modification record for literals that represent an address in the program
Symbol-Defining Statements (1/2)

- Labels on instructions or data areas
  - The value of such a label is the address assigned to the statement
- Defining symbols
  - Symbol EQU Value
  - Value can be: constant, other symbol, expression
  - Making the source program easier to understand
  - No forward reference

Symbol-Defining Statements (2/2)

- Example 1
  - MAXLEN EQU 4096
  - LDT #MAXLEN
- Example 2
  - BASE EQU R1
  - COUNT EQU R2
  - INDEX EQU R3
- Example 3
  - MAXLEN EQU BUFEND-BUFFER
ORG (origin)

- Indirectly assign values to symbols
- Reset the location counter to the specified value
  - ORG value
- Value can be: constant, other symbol, expression
- No forward reference
- Example
  - SYMBOL: 6bytes
  - VALUE: 1word
  - FLAGS: 2bytes
  - LDA VALUE, X

```
<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>VALUE</th>
<th>FLAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(100 entries)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

ORG Example

- Using EQU statements
  - STAB RESB 1100
    SYMBOL EQU STAB
    VALUE EQU STAB+6
    FLAG EQU STAB+9

- Using ORG statements
  - STAB RESB 1100
    ORG STAB
    SYMBOL RESB 6
    VALUE RESW 1
    FLAGS RESB 2
    ORG STAB+1100
Expressions

- Expressions can be classified as absolute expressions or relative expressions
  - MAXLEN EQU BUFEND-BUFFER
  - BUFEND and BUFFER both are relative terms, representing addresses within the program
  - However the expression BUFEND-BUFFER represents an absolute value
- When relative terms are paired with opposite signs, the dependency on the program starting address is canceled out; the result is an absolute value

SYMTAB

- None of the relative terms may enter into a multiplication or division operation
- Errors:
  - BUFEND+BUFFER
  - 100-BUFFER
  - 3*BUFFER
- The type of an expression
  - keep track of the types of all symbols defined in the program

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETADR</td>
<td>R</td>
<td>30</td>
</tr>
<tr>
<td>BUFFER</td>
<td>R</td>
<td>36</td>
</tr>
<tr>
<td>BUFEND</td>
<td>R</td>
<td>1036</td>
</tr>
<tr>
<td>MAXLEN</td>
<td>A</td>
<td>1000</td>
</tr>
</tbody>
</table>
## Example 2.9

<table>
<thead>
<tr>
<th>SYMTAB</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FIRST</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CLOOP</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>ENDFIL</td>
<td>1A</td>
<td></td>
</tr>
<tr>
<td>RETADR</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>LENGTH</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>BUFFER</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>BUFEND</td>
<td>1036</td>
<td></td>
</tr>
<tr>
<td>MAXLEN</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>RDREC</td>
<td>1036</td>
<td></td>
</tr>
<tr>
<td>RLOOP</td>
<td>1040</td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td>1056</td>
<td></td>
</tr>
<tr>
<td>INPUT</td>
<td>105C</td>
<td></td>
</tr>
<tr>
<td>WREC</td>
<td>105D</td>
<td></td>
</tr>
<tr>
<td>WLOOP</td>
<td>1062</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LITTAB</th>
<th>C'EOF'</th>
<th>454F46</th>
<th>3</th>
<th>002D</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'05'</td>
<td>05</td>
<td>1</td>
<td></td>
<td>1076</td>
</tr>
</tbody>
</table>

## Program Blocks

- **Program blocks**
  - Refer to segments of code that are rearranged within a single object program unit
  - **USE** `[blockname]`
  - At the beginning, statements are assumed to be part of the unnamed (default) block
  - If no USE statements are included, the entire program belongs to this single block
  - Example: Figure 2.11
  - Each program block may actually contain several separate segments of the source program
Figure 2.11

Program Blocks - Implementation

- **Pass 1**
  - Each program block has a separate location counter
  - Each label is assigned an address that is relative to the start of the block that contains it
  - At the end of Pass 1, the latest value of the location counter for each block indicates the length of that block
  - The assembler can then assign to each block a starting address in the object program

- **Pass 2**
  - The address of each symbol can be computed by adding the assigned block starting address and the relative address of the symbol to that block
Figure 2.12

- Each source line is given a relative address assigned and a block number

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block number</th>
<th>Address</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>(default)</td>
<td>0</td>
<td>0000</td>
<td>0066</td>
</tr>
<tr>
<td>CDATA</td>
<td>1</td>
<td>0066</td>
<td>000B</td>
</tr>
<tr>
<td>CBLKS</td>
<td>2</td>
<td>0071</td>
<td>1000</td>
</tr>
</tbody>
</table>

- For absolute symbol, there is no block number
  - line 107
- Example
  - 20 0006 0 LDA LENGTH 032060
  - LENGTH = (Block 1) + 0003 = 0066 + 0003 = 0069
  - LOCCTR = (Block 0) + 0009 = 0009

Program Readability

- Program readability
  - No extended format instructions on lines 15, 35, 65
  - No needs for base relative addressing (line 13, 14)
  - LTORG is used to make sure the literals are placed ahead of any large data areas (line 253)
- Object code
  - It is not necessary to physically rearrange the generated code in the object program
  - See Fig. 2.13, Fig. 2.14
Figure 2.13 Object program corresponding to Fig. 2.11.

Figure 2.14 Program blocks from Fig. 2.11 traced through the assembly and loading processes.
Control Sections and Program Linking

- **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
  - Instruction in one control section may need to refer to instructions or data located in another section
  - Because of this, there should be some means for linking control sections together
  - Fig. 2.15, 2.16

---

**Figure 2.15**

Illustration of control sections and program linking.
External Definition and References

- **External definition**
  - `EXTDEF name [, name]`
  - EXTDEF names symbols that are defined in this control section and may be used by other sections

- **External reference**
  - `EXTREF name [,name]`
  - EXTREF names symbols that are used in this control section and are defined elsewhere

- **Example**
  - 15 0003 CLOOP +JSUB RDREC 4B100000
  - 160 0017 +STCH BUFFER,X 57900000
  - 190 0028 MAXLEN WORD BUFEND-BUFFER 000000
Implementation

- The assembler must include information in the object program that will cause the loader to insert proper values where they are required

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

- Refer record
  - Col. 1  D
  - Col. 2-7  Name of external symbol referred to in this control section
  - Col. 8-73  Name of other external reference symbols

Modification Record

- Modification record
  - Col. 1  M
  - Col. 2-7  Starting address of the field to be modified (hexadecimal)
  - Col. 8-9  Length of the field to be modified, in half-bytes (hexadecimal)
  - Col.11-16  External symbol whose value is to be added to or subtracted from the indicated field
  - Note: control section name is automatically an external symbol, i.e. it is available for use in Modification records.

- Example
  - Figure 2.17
  - M00000405+RDREC
  - M00000705+COPY
External References in Expression

- Earlier definitions
  - Required all of the relative terms be paired in an expression (an absolute expression), or that all except one be paired (a relative expression)

- New restriction
  - Both terms in each pair must be relative within the same control section
    - Ex: BUFEND-BUFFER
    - Ex: RDREC-COPY

- In general, the assembler cannot determine whether or not the expression is legal at assembly time. This work will be handled by a linking loader.