CSI 402 – Lecture 8
(Linkers and Loaders)

Linkers and Loaders

Ref: Chapter 3 of [Beck].

Tasks Carried out by Linkers & Loaders:

- **Linking:** Combining two or more object programs into a single unit ("load module").
- **Loading:** Bringing an object program (load module) into memory for execution.
- **Relocation:** Modifying an object program so that it can be loaded from a location different from the one originally specified.

Discussion on Loaders

Functions of a Loader:

- Brings an object program into memory.
- Starts its execution.

Two Forms Loaders:

- **Absolute loader**: A loader which does not perform relocation. (Needed for programs associated with the operating system.)
- **Relocating** (or **Relative**) **loader**: Carries out relocation. (Needed for users' programs.)

Discussion on Loaders (continued)

Absolute Loader:

- Easy to design.
- Need to know the load module format.

Load module format – Example:

```
Header record
Text record
Text record
    .
    .
    .
Text record
End  record
```

Discussion on Loaders (continued)

Absolute Loader (continued):

- The Header record contains Program Name, Start Address and Program Length.
- Each Text record contains Start Address, Length, Object Code and Checksum.
- The End record contains only the Start address.

Note: The start addresses in the Header Record and End Record may be different. (Why?)

Absolute Loader Outline: Handout 8.1.

Checksum

- A simple error detection scheme.
- Each text record contains a checksum value (e.g. sum of all the bytes in that record).
- Loader computes checksum as the bytes in a text record are copied into memory.
- If the computed checksum does not match the stored value, the loader retries the loading process for the text record.
- If the checksums match, we can't be sure that the loading process is error-free.
- Using a checksum for each text record reduces the amount of code to be reloaded due to an error.
- Checksum-like schemes commonly used in hardware and software (e.g. parity bits in memory and checksums in network packets).

Program Relocation (review)

- Assembler sets start address at zero.
- Fills in relative addresses while assembling instructions.
- Produces a Modifier record for each instruction which needs relocation.
- Each modifier record contains starting byte address and the length of the address field to be modified.
- Modifier records follow the text records in the load module format.
- For SIC, all instructions except `RSUB` need modifier records.
- For SIC/XE, only 4-byte instructions need modifier records.

Relocating Loader Outline: Handout 8.2.

Relocation Bits

- Alternative to modifier records.
- Useful for machines such as SIC where the number of modifier records may be large.
- Use one bit for each word of object code; bit value = 1 if relocation is needed and 0 otherwise.
- Include the relocation bits in the text record so that each text record contains starting address, length, relocation bit mask and the object code.

Example: To be presented in class.

- With modification bits (instead of modifier records), the size of object code will be significantly smaller.
- The outline for relocating loader must be revised.

Exercise: Revise the outline for the relocating loader to allow relocation bits instead of modifier records.

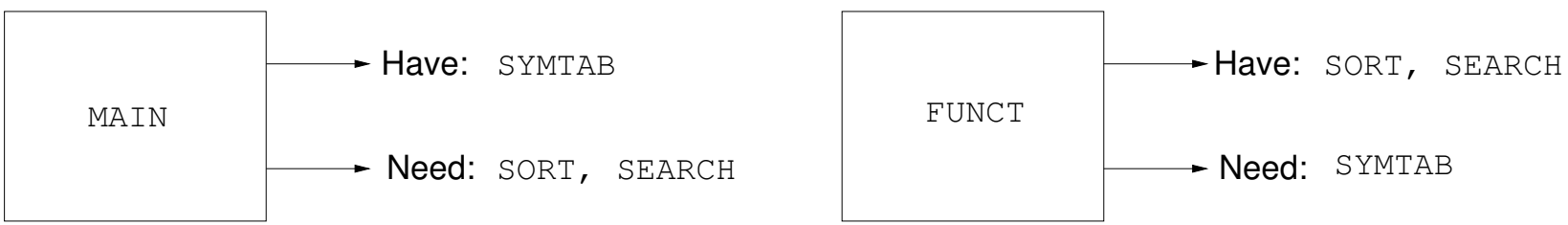
Separately Assembled Modules (Review)

- Assembler directives `EXTDEF` and `EXTREF`.
- All instructions referencing external symbols use 4-byte format.
- Assembler produces External Definition Table (EDT) and External Reference Table (ERT) for each module.
- Each entry of EDT contains a symbol and its (relative) address.
- Each entry of ERT contains a symbol and the (relative) addresses where the address of the symbol is needed.

A Simplified View of Linking

- The EDT for a module gives the external symbols that are provided by the module.
- The ERT for a module gives the external symbols that are needed by the module.
- Linker must ensure that each external symbol needed by a module is provided by another.

Example: Handout 8.3.



Steps Involved in Linking

1. Obtain load point for each module.
2. Combine the separate EDTs into a single EDT (called "Combined EDT").
3. Obtain the "External Symbol Table" from the Combined EDT by adding to each relative address, the load point of the corresponding module.
4. Now, the External Symbol Table can be used to resolve each external reference.

Example: To be presented in class using Handout 8.3.

Suggested Exercises

1. Make sure that you understand the need for and the differences between relocation, linking and loading.
2. Understand the need for checksum.
3. Understand the use of relocation bits; modify the outline of relocating loader to accommodate relocation bits instead of modifier records (used for relocation).
4. Study the example in Handout 8.3 to understand all the steps involved in linking.