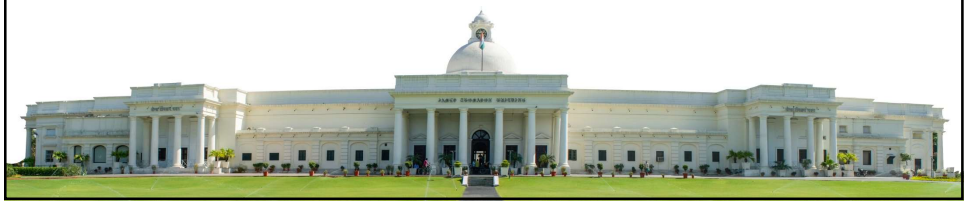
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

# System Software
# CSN-252
# Multi-pass Assembler
# Loaders and Linkers

Chapter 2 & 3 L. L. Beck



## Multi-Pass Assemblers

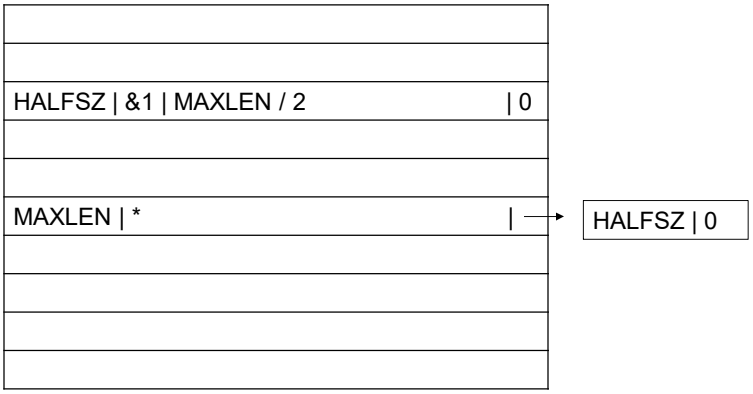
- Restriction on EQU and ORG
  - no forward reference, since symbols' value can't be defined during the first pass

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

- Such forward references create difficulty in reading the program also
- Solution – Multi-pass Assembler
  - Store the symbol definitions that involve forward references in the symbol table
  - Use link list to keep track of whose value depend on an undefined symbol

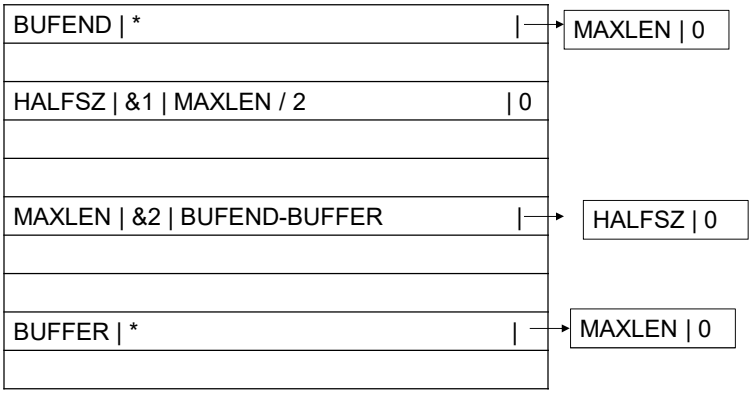
```
HALFSZ      EQU   MAXLEN/2
MAXLEN      EQU   BUFEND-BUFFER
PREVBT      EQU   BUFFER-1

BUFFER      RESB  4096
BUFEND      EQU   *
```



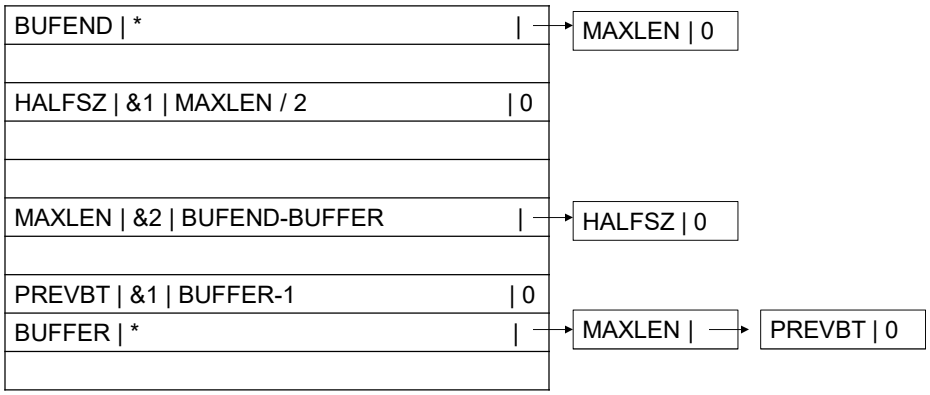

```
HALFSZ      EQU   MAXLEN/2
MAXLEN      EQU   BUFEND-BUFFER
PREVBT      EQU   BUFFER-1

BUFFER      RESB  4096
BUFEND      EQU   *
```



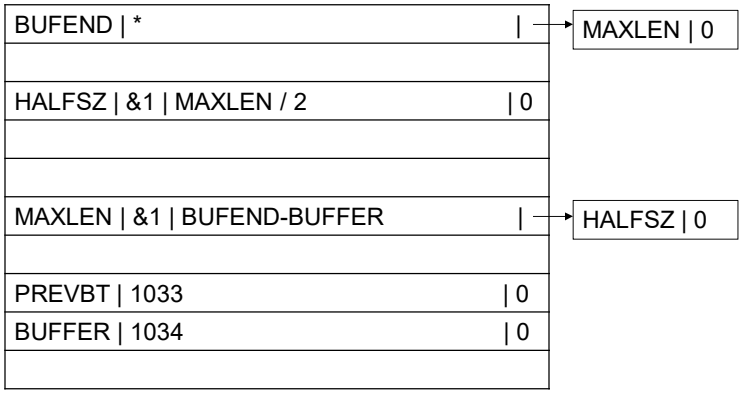
```
HALFSZ      EQU   MAXLEN/2
MAXLEN      EQU   BUFEND-BUFFER
PREVBT      EQU   BUFFER-1

BUFFER      RESB  4096
BUFEND      EQU   *
```




```
HALFSZ      EQU   MAXLEN/2
MAXLEN      EQU   BUFEND-BUFFER
PREVBT      EQU   BUFFER-1

BUFFER      RESB  4096
BUFEND      EQU   *
```



```
HALFSZ      EQU     MAXLEN/2
MAXLEN      EQU     BUFEND-BUFFER
PREVBT      EQU     BUFFER-1

BUFFER      RESB    4096
BUFEND      EQU     *
```

| | |
|---|---|
| BUFEND \| 2034 | \| 0 |
| | |
| HALFSZ \| 800 | \| 0 |
| | |
| | |
| MAXLEN \| 1000 | \| 0 |
| | |
| PREVBT \| 1033 | \| 0 |
| BUFFER \| 1034 | \| 0 |
| | |

## Loaders and Linkers

- Contents of object program
- To execute an object program, we need
  - **Relocation**, which modifies the object program so that it can be loaded at an address different from the location originally specified
  - **Linking**, which combines two or more separate object programs and supplies the information needed to allow references between them
  - **Loading and Allocation**, which allocates memory location and brings the object program into memory for execution

## Loaders

- Many loaders also support relocation and linking
- All program translators produce program in the same object format

  => Only one loader / linker

```
ld uses the general purpose BFD libraries to operate on
object files. This allows ld to read, combine, and write
object files in many different formats
```

- Common Object File Format (COFF): a format for executable, object code, and shared library computer files used on Unix systems before ELF
- Portable Executable (PE): Microsoft Windows



## Loaders

- Type of loaders
  - assemble-and-go loader
  - absolute loader (bootstrap loader)
  - relocating loader (relative loader)
  - linking (and relocating) loader
- Loader Design options
  - linkage editors
  - dynamic linking
  - bootstrap loaders

# Assemble-and-go Loader

- Characteristic
  - the object code is stored in memory after assembly
  - single JUMP instruction
- Advantage
  - simple, developing environment
- Disadvantage
  - whenever the program is to be executed, it has to be assembled again

| | | | | | |
|---|---|---|---|---|---|
| 1 | | copy | start | 1000 | |
| 2 | 1000 | eof | byte | c'eof' | 454f46 |
| 3 | 1003 | zero | word | 0 | 000000 |
| 4 | 1006 | retadr | resw | 1 | |
| 5 | 1009 | length | resw | 1 | |
| 6 | 100c | buffer | resw | 4096 | |
| 7 | 200c | first | stl | retadr | 141006 |
| 8 | 200f | cloop | jsub | rdrec | 48 |
| 9 | 2012 | | lda | length | 001009 |
| 10 | 2015 | | comp | zero | 281003 |
| 11 | 2018 | | jeq | endfil | 30 |
| 12 | 201b | | j | cloop | 30200f |
| 13 | 201e | endfil | ldl | retadr | 081006 |
| 14 | 2021 | | rsub | | 4c0000 |

1000 454f46 000000

200C 141006 48~~0000~~ 001009 281003 30~~0000~~ 30200f ……

202a 201e

# Design of an Absolute Loader

- Absolute Program
  - Advantage
    - Simple and efficient
    - Requires a single pass (?)
  - Disadvantage
    - the need for programmer to specify the actual address
    - difficult to use subroutine libraries
- Object Program format

## Algorithm for an absolute loader

```
Begin
read Header record
verify program name and length
read first Text record / next record
while record type is not 'E' do
   begin
   {if object code is in character form, convert into internal
   representation}
   move object code to specified location in memory
   read next object program record
   end
jump to address specified in End record
end
```

HCOPY__001000 000007
T001000 06 001006 4C0000 ascii code of 1 = $(49)_{10}$
T001006 01 05 ascii code of A = $(65)_{10}$
E001000

Memory Contents
000000
:
001000 0010064C000005

- each byte of assembled code is given using its hexadecimal representation in character form
- easy to read by human beings
- each byte of object code is stored as a half byte
- most machine store object programs in a binary form