Absolute loader:

begin

read Header record

verify program name and length

read first Text record

while record type ≠ '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

**Object1.txt**

H COPY 001000 00107A

T 001000 0C 141033 482039 001036 001036

T 00101E 0C 0C1036 482061 081033 001036

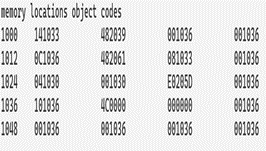
T 001047 0C 041030 001030 E0205D 001036

T 001077 0C 101036 4C0000 000000 001036

Output.txt

Create this text doc and keep.

Terminal O/P



Relocation Loader Algorithm

Begin

Get PROGADDR from OS

While not end of input do

{read next record

while record type !='E' do

{read next input record

while record type =' T' do

{ move object code from record to location

ADDR+ specified address }

while record type = 'M'

add PROGADDR at the location PROGADOR +specified address }}

end

**Object1.txt**

H COPY 001000 00107A

T 001000 0C 141033 482039 001036 001036

T 00101E 0C 0C1036 482061 081033 001036

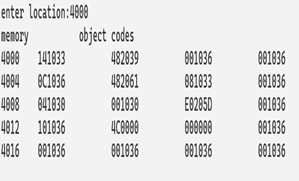
T 001047 0C 041030 001030 E0205D 001036

T 001077 0C 101036 4C0000 000000 001036

Create the output2.txt txt doc -

Terminal O/P

Enter the desired location to relocate the object code.



Algorithm for Pass 1 of a linking loader.

Begin / get PROGADDR from operating system

set CSADDR to PROGADDR (for first control section)

while not end of input do

begin / read next input record (Header record for control section)

set CSLTH to control section length

search ESTAB for control section name

if found then

set error flag {duplicate external symbol}

else / enter control section name into ESTAB with value CSADDR

while record type ≠ 'E' do

begin /read next input record

if record type = 'D' then

for each symbol in the record do

begin / search ESTAB for symbol name

if found then

set error flag (duplicate external symbol)

else /enter symbol into ESTAB with value

(CSADDR+ indicated address)

end {for} / end (while ≠ 'E'}

add CSLTH to CSADDR (starting address for next control section)

end {while not EOF} / end {Pass 1}

**linkinput.txt:**

H PROGA 001000 000063

D LISTA 000054 ENDA 000054

R LISTB ENDB LISTC ENDC

T 000020 0A 03201D 77100004 050014

T 000054 0F 100014 000008 004051 000004 100000

M 000024 05 +LISTB

M 000054 06 +LISTC

M 000060 06 +LISTB

M 000060 06 -LISTA

E 000020 / H PROGB 001000 00007F

D LISTB 000060 ENDB 000070

R LISTA LISTC ENDY

T 000036 0B 03100000 772027 05100000

T 000070 0F 100000 000008 004051 000004 100060

M 000037 05 +LISTA

M 00003E 05 -LISTA

M 000070 06 -LISTA

M 000070 06 +LISTC

M 00007C 06 +PROGB

M 00007C 06 -LISTA / E 000000

H PROGC 001000 0000051D LISTC 000030 ENDC 000042

R LISTA LISTB ENDB

T 000018 0C 03100000 77100004 05100000

T 000042 0F 100030 000008 004051 000004 100000

M 00001D 05 +LISTB

M 000021 05 -LISTA

M 000042 06 -LISTA

M 000042 06 +PROGC

M 00004E 06 +LISTB

M 00004E 06 -LISTA / E / END

Pass 2 of a linking loader.

begin

set CSADDR to PROGADDR

set EXECADDR to PROGADDR

while not end of input do

begin

read next input record {Header record}

set CSLTH to control section length

while record type !='E' do

begin

read next input record

if record type = 'T' then

begin

{if object code is in character form, convert into internal representation}

move object code from record to location {CSADDR + specified address}

end {if 'T'}

else if record type = 'M' then

begin

search ESTAB for modifying symbol name

if found then

add or subtract symbol value at location

(CSADDR + specified address)

else

set error flag (undefined external symbol)

end (if 'M'}

end (while ≠ 'E'}

if an address is specified (in End record) then

set EXECADDR to (CSADDR + specified address)

add CSLTH to CSADDR

end {while not EOF}

jump to location given by EXECADDR {to start execution of loaded program }

end {Pass 2}

**input.txt:** H PROGA 000000 000063

D LISTA 000054 ENDA 000064

R LISTB ENDB LISTC ENDC

T 000020 0A 03201D 77100004 050014

T 000054 0F 100014 000008 004051 000004 100000

M 000024 05 +LISTA

M 000054 06 +LISTC

M 000060 06 +LISTB

M 000060 06 -LISTA

E 000020

H PROGB 000000 00007F

D LISTB 000060 ENDB 000070

R LISTA LISTC ENDY

T 000036 0B 03100000 772027 05100000

T 000070 0F 100000 000008 004051 000004 100060

M 000037 05 +LISTA

M 00003E 05 -LISTA

M 000070 06 -LISTA

M 000070 06 +LISTC

M 00007C 06 +PROGB

M 00007C 06 -LISTA

E 000000

H PROGC 000000 0000051

D LISTC 000030 ENDC 000042

R LISTA LISTB ENDB

T 000018 0C 03100000 77100004 05100000

T 000042 0F 100030 000008 004051 000004 100000

M 00001D 05 +LISTB

M 000021 05 -LISTA

M 000042 06 -LISTA

M 000042 06 +PROGC

M 00004E 06 +LISTB

M 00004E 06 -LISTA

E

END

Assembler Pass 1:

begin

read first input line

if OPCODE='START' then

begin

save #[OPERAND] as starting address

initialize LOCCTR to starting address

write line to intermediate file

read next input line

end (if START}

else

initialize LOCCTR to 0

while OCODE != 'END' do

begin

if this is not a comment line then

begin

if there is a symbol in the LABEL field then begin

search SYMTAB for LABEL

if found then

set error flag (duplicate symbol) else

insert (LABEL,LOCCTR) into SYMTAB

end {if symbol}

search OPTAB for OPCODE

if found then

add 3 (instruction length) to LOCCTR

else if OPCODE='WORD' then

add 3 to LOCCTR

else if OPCODE = 'RESW' then

add 3\* #[OPERAND] to LOCCTR

else if OPCODE = 'RESB' then

add #[OPERAND] to LOCCTR

else if OPCODE='BYTE' then

begin

find length of constant in bytes

add length to LOCCTR end {if BYTE)

else

set error flag (invalid operation code)

end if not a comment)

write line to intermediate file

read next input line

end (while not END}

write last line to intermediate file

save (LOCCTR starting address) as program length

end {Pass 1}

**Input.txt-**

\*\* START 2000

\*\* LDA FIVE

\*\* STA ALPHA

\*\* ORG 2050

\*\* LDCH CHARZ

\*\* STCH C1

\*\* ORG 3000

A EQU 2000

FIVE WORD 5

\*\* ORG 8000

B EQU 90

C1 RESB 1

\*\* END \*\*

Optab.txt

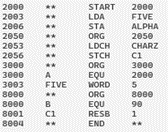
LDA 03

STA 0f

LDCH 53

STCH 57

END \*



Assembler Pass2:

begin

read first input line (from intermediate file)

if OPCODE='START' then

beginwrite listing line

read next input line

end {if START}

write Header record to object program

initialize first Text record

while OPCODE != 'END' do

begin

if this is not a comment line then

begin

search OPTAB for OPCODE

if found then

begin

if there is a symbol in OPERAND field then begin

search SYMTAB for OPERAND

if found then.

store symbol value as operand address else

begin

store 0 as operand address

set error flag (undefined symbol) end

end (if symbol}

else

store 0 as operand address

assemble the object code instruction

end {if opcode found)

else if OPCODE='BYTE' or 'WORD' then convert constant to object code

if object code will not fit into the current Text record then begin

write Text record to object program

initialize new Text record

end

add object code to Text record

end if not comment}

write listing line

read next input line

end(while not END)

write last Text record to object program write End record to object program

write last listing line

end{Pass 2}

Intermediate.txt-

\*\* START 2000

2000 \*\* LDA FIVE

2003 \*\* STA ALPHA

2006 \*\* LDCH CHARZ

2009 \*\* STCH C1

2012 ALPHA RESW 1

2015 FIVE WORD 5

2018 CHARZ BYTE C'EOF'

2019 C1 RESB 1

2020 \*\* END \*\*

Length.txt-

25

Optab.txt

LDA 00

STA 0C

LDCH 50

STCH 54

END \*

Symtab.txt

ALPHA 2012

FIVE 2015

CHARZ 2018

C1 2019