USER’S GUIDE:

**🡪Table Of Contents:**

***Content: Page Number***

**-Overview:**

Introduction to assembler and contents of Users Guide **2**

**-Prerequisites:**

Information to gather before running Assembler Program **3**

**-Input and Output:**

Guide to proper I/O of Assembler **3**

1. ***Object Output File*  4**

**-Relocatabilty**

**2.) *Executable Input File*  7**

**3.) *Output Error Log File* 8**

**-Running the Program:**

Guide to installation and usage of the Assembler Program **8**

**1.) *Contents of Software* 8**

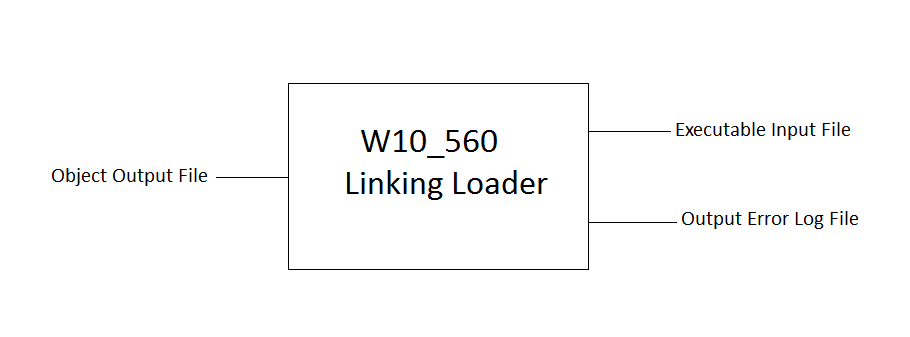
**2.) *Installation and Execution* 9**

**-Troubleshooting:**

References to error handling **10**

**🡪Overview:**

The following guide explains how to install, understand and properly use the contents and functions of the W10\_560 Machines linking loader program.(Note: To have success with this linking loader please first reference the **Prerequisites** section for some suggested base knowledge.) The main purpose of this linking loader is to convert the given executable input file containing external symbols into an absolute executable input file for the W10\_560 machine simulator. A brief preview of the process can be seen below.



For a description of syntax and contents of these I/O files refer to the **Input and Output** section

For information on how to install and run the assembler, please reference the **Running the Program** section of this guide. Additional support information can be found in the proceeding **Troubleshooting** section. Specifically, the Troubleshooting section shows potential runtime errors found in the listing output and provides guidance to fixing the problem.

**🡪Prerequisites:**

Usage of the software contained in c560ab03\_lab4 requires knowledge of certain basic concepts. The following is a suggested knowledge base to obtain before running the program.

- Understanding of ASCII table and mathematic Conversions, (Hexadecimal to Decimal,

binary arithmetic etc.).

- Understanding of navigation through “Unix-esque” operating system.

- Assembler language programming experience

-List of operations and usage can be found in attached ***PSEUDO\_OPS.doc****x* and

***MACHINE\_GUIDE.pdf***  located in docs directory

- Understanding of basic assembler function and operation.

(For additional technical W10\_560 Machine information, feel free to reference MACHINE\_GUIDE.pdf located in the docs directory)

**🡪Input and Output:**

The user accessible inputs and outputs for the linker loader are as follows:

- *Object Output File*

- *Executable Input File*

In an effort to lessen user debug time, the following table will highlight expected syntax, and show examples of both input and output files. These are to be used as reference and will hopefully help to expand and further complex program usage in the future.

1.) *Object Output File*

The *Object Output File* contains the machine code produced by the W10\_560 machine assembler. This machine code serves as the input to the linker/loader in order to generate the input for the W10\_560 machine simulator. Multiple *Object Output Files* will be needed if there are external symbols involved. A correct *Object Output File* must have the following syntax:

**A Header Record**

**record position 1: H**

**record positions 2-3: a 2 Hex character address at which execution is to begin**

**record positions 4-9: a 6 character segment name**

**record positions 10-11: a 2 Hex character value denoting the segment load address**

**record positions 12-13: a 2 Hex character value denoting the length of the segment**

**Text Records**

**record position 1: T**

**record positions 2-3: a 2 Hex character address at which the information is to be stored**

**record positions 4-8: Initial value at that address, as a 5 Hex character value**

**record position 9 : Either empty, “X”, or “M”**

**“M”**

**-An “M” in position 9 means the line is relocatable (see Relocatability section for reference)**

**“X**

**-An “X” in position 9 means the line uses an external symbol see External Symbol section for reference)**

**-record position 10-13: Name of external symbol**

Here is an example of multiple files to be handled by the linking loader

File Program.asm

----------------

Main ORI

EXT Displ,V

ENT Start

EXT X

;

Start BRS 3,Displ Display 6..0

LD 1,V r1 <- M[V]

ST 1,X M[X] <- r1

BRS 3,Displ Display 2..0

BR 0,0 ; halt

END

File Subr.asm

-------------

Mesg ORI

ENT Displ,X

NOUT EQU 2

;

Txt NMD 42336 also represents "\nV"

CCD al

CCD ue

CCD =

X NMD 6

SavR RES 1

Pairs RES 1

;

Displ ST 3,SavR M[SavR] <- r3

LD 0,X r0 <- M[X]

BRN 0,Done if (r0 < 0) goto Done

Loop LDI 1,4 Display text "\nValue= " (4 pairs of chars)

TLP BRZ 1,Next Terminate Text LooP

LDI 3,4 Calc. offset in r3

ST 1,Pairs prepare for subtraction

SUB 3,Pairs offset = 4 - r1

LD 2,Txt(3) r2 <- M[Txt + r3]

PTC 2 output first char. of pair

SHL 2,8 get second char. ready

PTC 2 output second char.

SUB 1,=1 r1--

BR 3,TLP goto TLP

Next IO NOUT,X Display value in M[X]

SUB 0,=1 r0--

ST 0,X M[X] <- r0

BRN 0,Done if (r0 < 0) goto Done

BR 3,Loop goto Loop

Done LD 3,SavR r3 <- M[SavR]

BR 3,0(3) return

END

File Val.asm

------------

Data ORI

EXT X

ENT V

V NMD 2

NOUT BR 0,0

Done BR 3,X

LD 1,=1

END Done

An example of the multiple *Object Output Files* is shown below. It is the corresponding output to the sample input shown above.

File Program.asm

----------------

H00Main 0005M

EStart 00

T00F3000XDispl

T0101000XV

T0221000XX

T03F3000XDispl

T04C0000

File Subr.asm

-------------

H07Mesg 001DM

EDispl 07

EX 04

T000A560

T01616C0

T0275650

T033D200

T0400006

T0723005M

T0800004M

T09E001AM

T0A11004

T0BD1015M

T0C13004

T0D21006M

T0E43006M

T0F02C00M

T10B3800

T1192008

T12B3800

T134101CM

T14C300BM

T15B2004M

T164001CM

T1720004M

T18E001AM

T19C300AM

T1A03005M

T1BC3C00

T1C00001

File Val.asm

------------

H02Data 0005M

EV 00

T0000002

T01C0000

T02C3000XX

T0301004M

T0400001

2.) *Executable Input File(s)*

The *Executable Input File* will contain the machine code to be used as the the input file for the w10\_560 machine simulator. This will be an absolute executable file that has processed the external symbols contained in the *Object Output File.* Below is the required syntax for the *Executable Input File*:

**A Header Record**

**record position 1: H**

**record positions 2-3: a 2 Hex character address at which execution is to begin**

**record positions 4-9: a 6 character segment name**

**record positions 10-11: a 2 Hex character value denoting the segment load address**

**record positions 12-13: a 2 Hex character value denoting the length of the segment**

**Text Records**

**record position 1: T**

**record positions 2-3: a 2 Hex character address at which the information is to be stored**

**record positions 4-8: Initial value at that address, as a 5 Hex character value**

An example of the *Executable Input File* is shown below for the given input above:

H28Main 2827

T28F3034

T290104A

T2A21031

T2BF3034

T2CC0000

T2D0A560

T2E616C0

T2F75650

T303D200

T3100006

T3423032

T3500031

T36E0047

T3711004

T38D1042

T3913004

T3A21033

T3B43033

T3C02C2D

T3DB3800

T3E92008

T3FB3800

T4041049

T41C3038

T42B2031

T4340049

T4420031

T45E0047

T46C3037

T4703032

T48C3C00

T4900001

T4A00002

T4BC0000

T4CC3031

T4D0104E

T4E00001

3.) *Output Error Log File*

This output file will contain all the run time errors that occur in the linking loader program. The file will have the following syntax:

“Error: ...”

“Error: …”

**.**

**.**

**.**

“Error: …”

A listing of all possible errors with descriptions and possible solutions are provided in the **Troubleshooting** section.

**🡪Running Linker Program**

**Contents of Software:**

The given folder, ***c560ab03\_lab4*** contains three subfolders: ***src***, ***docs***, ***test****.* The folder ***src*** contains all program coding; these files should not be manipulated by user without further knowledge of assembler program code. The ***docs*** folder contains all documentation for the assembler program. The final folder, ***test,*** will contain different testing cases and sample codes. Again, this information is simply for completeness and should not to be manipulated without further knowledge of coding and specifications or unless directed.

Note: For syntax examples shown below, the program is run in the UNIX environment. (Note:

Syntax will also apply for most \*nix environments)

**Installation and Execution:**

Note: The software is written in a combination of RESOLVE/C++, and C++. It is

necessary for installation that the Resolve/C++ catalog is included in the

desired installation directory.

**1.)** Given the assemblersoftware, installation first requires copying the ***c560ab03\_lab4***

folder to a desired directory. This can be done both inside or outside of the Unix

environment. A sample UNIX copy can be seen below using the sample directory,

*NEW\_USER,* as the desired the location.

% **cp *c560ab03\_lab4*** *NEW\_USER/*

**2.)** Once the ***c560ab03\_lab4***folder is located in the desired directory, copy the desired

*Object Output Files* to the ***src*** folder as shown below. We will call

this input file *objn*, with n being the number of the input file:

% **cp *objn*** *NEW\_USER/c560ab03\_lab4/src/*

If all files are located in the appropriate folders, the next step will be to run the program.

(Note: Remember list command ls to verify file locations.)

**3.)** The next step to take before running the program is to ensure you are located in the correct

directory. If following the commands above, changing directories as shown below:

***% cd*** *NEW\_USER/c560ab04\_lab4/src/*

Remember, despite ***c560ab03\_lab4***location, it is important to be in the ***src***

subfolder to run the command line assembler in the following step.

**4.)** Once you are located in the proper directory, the next step in execution is to **compile the**

**programs installed**. Using the Resolve/C++ rcpp-make command,(as

mentioned previously, RESOLVE/C++ components must be included), compile

linker.cpp. A sample call is shown below.

***% rcpp-make*** *linker.cpp*

**5.)** As previously mentioned, the assembler program is run through the command prompt,

(assuming location in correct directory). The syntax is shown below.

***% linker outputexecutable errorlog obj1 ... objn***

Where: ***linker*** 🡪Linking Loader Program

***outputexecutable*** 🡪*Executable Input File*

(Will be created, or prompted to replace)

***obj1 ... objn*** 🡪*Object Output Files*,

(Copied above)

***errorlog*** 🡪*Output Error Log* ,

(Will be created, or prompted to replace)

Note: All of the output files will be written to the ***src*** folder and if these output files already exist, you will be prompted with the following messages:

***Warning: File ‘outputexecutable’ already exists, overwrite?(y/n)***

***Warning: File ‘errorlog’ already exists, overwrite?(y/n)***

5.) If the program has successfully run, the message

*The footprint size is \_\_\_\_\_\_ words.*

*Provide an appropriate IPLA(hex)*

The user will now be prompted to provide a hex address between 0 and FF to serve as the initial program load address. If the rest of the program runs correctly nothing else should be produced. Even though the program may run all the way through, it does not mean the *Executable Input File* was correct. As mentioned above, please check the *Output Error Log* for any runtime errors. Assistance with these errors can be found in the **Troubleshooting** section.

**🡪Troubleshooting:**

The linking loader program is designed to report errors when the input is run. Errors to the input, both syntax and memory allocation, will be reported to the *Output Error Log File*. Below is a list of error messages and information for error recovery.

***Output Error Log* Errors**

-“Error: More than one ‘Main’ object file provided”

**Information and Recovery:**

There is more than one main program declared in the *Object Output Files*. To correct this problem check the *Object Output Files* to see how many files have the segment name “main”. Choose one to be the main program and provide new, unique segment names for the rest of the files.

-“Error: No object output files specify ‘Main’”

**Information and Recovery:**

None of the *Object Output Files* are specified as the main program. To correct this error, choose one of the *Object Output Files* to be the main program or create a new one. Be sure to make the segment name of this file be “main”.

-“Error: External symbol already defined”

**Information and Recovery:**

The symbol to be used has already been defined in another *Object Output File*. To correct this error check the *Object Output Files* for reused symbols and replace with a new, unique segment name.

-“Error: External symbol \_\_\_\_\_ not defined”

**Information and Recovery:**

The symbol to be used has not been defined in an *Object Output File* and is not in the external symbol table. To correct this error review the *Object Output Files* to make sure the symbol is defined in a subroutine and that it is set as a global variable.

**Command Line Errors:**

**- “Program Quit. Command parameter error!”**

**Information and Recovery:**

Check parameters run in command line. Additional error lines may appear for clarity.

These additional errors are listed below, (Denoted with “---”)

**---“Error: File \_\_\_\_\_ does not exist.”**

**Information and Recovery:**

One of the parameters run through the simulator does not exist.

Check the listed file.

**--- “Error: File \_\_\_\_\_\_is not readable.”**

**Information and Recovery:**

One of the parameters run through the simulator is not readable.

Check the listed file.

**--- “Error: File \_\_\_\_\_\_is not writeable.”**

**Information and Recovery:**

One of the parameters run through the simulator is not writeable.

Check the listed file.

**- “Command line argument error.”**

**Information and Recovery:**

Check the number of arguments in the command line.

**- “Error: The current directory is not writeable!”**

**Information and Recovery:**

Check the permissions of the current directories listed in the command lines.

Also check for write ability.

**- “Error: Parameter\_\_\_\_\_ is a directory.”**

**Information and Recovery:**

One of the parameters run through the simulator was not a regular file, but

rather a directory. Check the listed file.