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. ***Assembly Language Source File*  4**

**-Relocatabilty 5**

**2.) *Object Output File*  6**

**3.) *Listing Output File*  7**

**-Running the Program:**

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

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

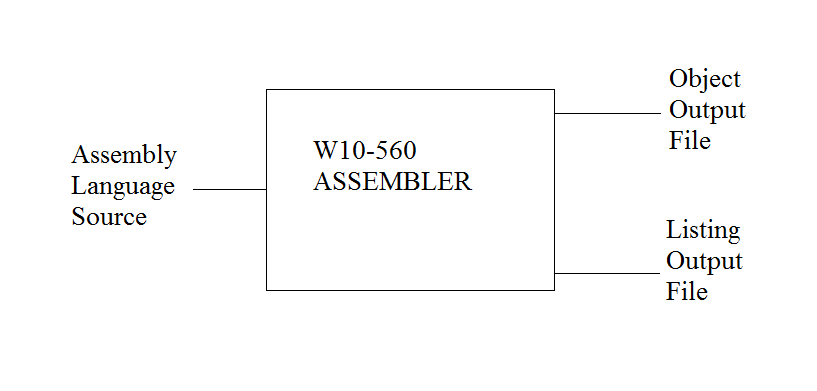
**2.) *Installation and Execution* 8**

**-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 assembler program.(Note: To have success with this assembler please first reference the **Prerequisites** section for some suggested base knowledge.) The main purpose of this assembler is to convert the given assembly language source file into an executable input file for the W10\_560 machine linking-loader and simulator. A brief diagram of the process can be seen below.



The Listing Output File contains a printout of the assembler’s operation results. This includes the before mentioned object code and the original source file. The syntax of the assemblers I/O will be discussed in detail below in 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\_lab3 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:**

As mentioned above, the user accessible inputs and outputs are as follows:

- *Assembly Language Source File*

- *Object Output File*

- *Listing Output 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.) *Assembly Language Source File:*

The *Assembly Language Source File* is a program written in the W10\_560 machine’s assembly language and is to be executed by the W10\_560 machine. Each record shown **must** follow the syntax below:

**Record Position Meaning**

**1-6 Label, if any, left justified**

**7-9 Unused**

**10-12 Operation field**

**13 Unused**

**14-end of record Operands and comments**

**Exception: A semicolon (;) in the first record position indicates that**

**the entire record is a comment. Additionally, comments may proceed**

**the proper operand in any record line.**

An example of a proper assembler code usage in the *Assembly Language Source File* is shown below.

*; This is an example*

*EG1 ORI 19*

*Y NMD 32[Y] = 32*

*Buff RES 1 Buffer for input*

*Acc EQU 1 Acc = 1*

*;*

*Start LD Acc,Y Acc <- 32*

*SHL 3,19 two instrs. to*

*SHL 3,1 ; clear R3*

*GTC 3 Input char to R3*

*ST 3,Buff ;store char to Buff*

*IO 0,IncVal Input integer*

*MUL Acc,=2 Acc <- Acc\*2*

*OR Acc,Buff(0)*

*ADD Acc,IncVal*

*BR 3,Finish Done*

*ByeMsg CCD HT*

*Finish LD 2,ByeMsg Put msg in R2*

*PTC 2 display first char*

*SHL 2,8 ready for 2nd char*

*IO 3,0(2) display 2nd char*

*BR 0,Acc(3) Halt with display*

*IncVal RES 1*

*END*

The *Assembly Language Source File* has some notable requirements to remember in order to properly run the assembler. First, remember the first non-comment line/record in the source file must contain an ORI operation with either an empty or integer operand. Similarly, the program must finish with an END operation on the last line/record. There are many more requirements for assembly language coding, please refer to the above notes and information referenced in the **Prerequisites** section.

Besides the need for correct syntax in assembler programming there is a maximum of 100 symbols, 50 literals and 200 source records allowed in source file. Exceeding these values will result in an appropriate error.

**Relocatabilty:**

Another feature provided by the assembly language is the concept of **Relocation.** This feature allows for a written program to be movable in memory, providing the opportunity to link multiple programs together and more easily adapt to memory requirements. In order to make a program relocatable, the operand in the ORI record line of the input program must be blank.

In order to implement this process, each symbol is assigned a status of being relative or absolute. An absolute symbol is not subject to relocation and will maintain the same memory value and location regardless of other related operations. A symbol is absolute if and only if it is defined by a decimally-represented integer or another absolute symbol. A program is also considered absolute if an operand is provided for the ORI line. If a symbol is not considered absolute from the previous conditions, then it is considered relative. A relative symbol is regarded only for its location compared to the ORI line.

Combining these ideas, if the operand field for an ORI line is left blank, the assembler will next mark each **relative** record line with an ‘**M**’ at the end of the object output line for a possible change of location. Remember that **Relocation** also does not permit any translation outside the allotted memory location.

When developing the assembler with respect to the other processes of the W10\_560 machine, updates were made to allow two new pseudo operations, *ENT* and *EXT*. The benefit of adding these two operations is the new opportunity of linking multiple, separately written assembly language programs together. Both new pseudo ops take advantage of the Relocation procedure developed, and further expand the global potentials of the assembler language. Below is an explanation of the function and usage of *ENT* and *EXT.*

ENT:

*ENT* defines a global symbol that is defined in the same program. This symbol may

be referenced in other programs. Up to four symbol names may be used and there is

no label allowed on the line.

*ENT symbol1,…,symbol4*

EXT:

*EXT* references an externally defined symbol. Up to four symbol names may be used

and there is no label allowed on the line.

*EXT symbol1,…,symbol4*

Both pseudo-ops must be the first operations used in the program, preceded only by the ORI operation and comment lines.

2.) *Object Output File*

The *Object Output File* contains the produced machine code. This machine code will provide the input to the linker/loader in order to generate the input for the W10\_560 machine simulator. Specifically, the *Object Output File* assigns memory locations to the appropriate instructions and assembles the code into 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**

**record positions 14: the character ‘M’ if and only if Relocatable**

**ENT Records**

**record position 1: E**

**record positions 2-8: a 6 character segment name**

**record positions 9-10: a 2 Hex character value denoting the symbol location**

**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 positions 9: the character ‘X’ if operation contains an EXT defined symbol**

**or ‘M’ if not containing an EXT defined symbol and if symbol**

**Relocatable.**

**record positions 10-15: if previous character was an ‘X, name of EXT symbol used.**

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

*H15EG1 1314*

*T1300020*

*T1501013*

*T1693013*

*T1793001*

*T18B1C00*

*T1923014*

*T1AB0025*

*T1B51026*

*T1C71014*

*T1D31025*

*T1EC3020*

*T1F48540*

*T200201F*

*T21B3800*

*T2292008*

*T23B3800*

*T24C0C01*

*T2600002*

3.) *Listing Output File*

The *Listing Output File* acts as the final “printout” of the broken-down source code and assembled memory contents. Specifically, it will contain a sample trace of each command, its instruction information. A sample output is shown below. Again, the printout is the corresponding output to the above mentioned *Assembly Language Source File* example.

+----------------------------------------------------------+

| Listing |

|----------------------------------------------------------|

| Object Code | Source |

|-----------------+----------------------------------------|

| Loc Data Rel | Rec Label OP Operands/Comments |

|(hex) (hex) |(dec) |

|-----------------+----------------------------------------+

| | EG1 ORI 19

| 13 00020 | 1 Y NMD 32 [Y] = 32

| | Buff RES 1 Buffer for input

| | Acc EQU 1 Acc = 1

| 15 0100D | 2 Start LD Acc,Y Acc <- 32

| 16 93013 | 3 SHL 3,19 two instrs. to

| 17 93001 | 4 SHL 3,1 ; clear R3

| 18 B1C00 | 5 GTC 3 Input char to R3

| 19 2300E | 6 ST 3,Buff ;store char to Buff

| 1A B0019 | 7 IO 0,IncVal Input integer

| 1B 51026 | 8 MUL Acc,=2 Acc <- Acc\*2

| 1C 7100E | 9 OR Acc,Buff(0)

| 1D 31019 | 10 ADD Acc,IncVal

| 1E C3014 | 11 BR 3,Finish Done

| 1F 48540 | 12 ByeMsg CCD HT

| 20 02020 | 13 Finish LD 2,ByeMsg Put msg in R2

| 21 B3800 | 14 PTC 2 display first char

| 22 92008 | 15 SHL 2,8 ready second char

| 23 B3800 | 16 IO 3,0(2) display 2nd char

| 24 C0C01 | 17 BR 0,Acc(3) Halt with display

| | IncVal RES 1

| | END

**🡪Running the Assembler Program**

**Contents of Software:**

The given folder, ***c560ab03\_lab3*** 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\_lab3***

folder to a desired directory. This can be done 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\_lab3*** *NEW\_USER/*

**2.)** Once the ***c560ab03\_lab3***folder is located in the desired directory, copy the desired

*Assembly Language Source File* to the ***src*** folder as shown below. We will call

this source file *sourceFile*:

% **cp *sourceFile*** *NEW\_USER/c560ab03\_lab3/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/c560ab03\_lab3/src/*

Remember, despite ***c560ab03\_lab3***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 assembler.cpp. A

sample call is shown below.

***% rcpp-make*** *assembler.cpp*

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

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

***% assembler sourcefile objectoutput listingoutput***

Where: ***assembler*** 🡪Assembler Program

***sourcefile*** 🡪*Assembly Language Source File,*

(Copied above)

***objectoutput***  🡪*Object Output File* ,

(Will be created, or prompted to replace)

***listingoutput*** 🡪*Listing Output File* ,

(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 ‘objectOutput’ already exists, overwrite?(y/n)***

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

**6.)** If the program has successfully run, the following messages will appear on the command prompt.

*Begin Assembly*

*…*

*Assembler Finished*

These messages mean the program has run correctly without an internal error. It does not mean the *Assembly Language Source File* was correct. As mentioned above, please check the *Listing Output File* for any runtime errors. Assistance with these errors can be found in the **Troubleshooting** section.

**🡪Troubleshooting:**

The assembler 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 *Listing Output File*. Below is a list of error messages and information for error recovery.

***Listing Output File* Errors**

- “**Error: Insufficient Number of records in source file.”**

**Information and Recovery:**

The *Assembly Language Source File* does not have the necessary amount of

records. This error was caused by a shortage of records which was did not satisfy the

program need to begin with an “*ORI*” operation and end with an “*END*”, excluding

commented lines . To correct this error, check input records for missing “*ORI*” or

“*END*” operations.

- “**Error: Missing ORI operation in source file.”**

**Information and Recovery:**

Similar to the error above, the *Assembly Language Source File* is missing the

appropriate beginning “*ORI*” operation. This error could have been caused by the usage

of another operation before “*ORI*” , which is not allowed in this assembler. To correct

this error, check input records for missing “*ORI*” or misplaced “*ORI*” operation and add in the first non-commented record.

- “**Error: Missing END operation in source file.”**

**Information and Recovery:**

Similar to the error above, the *Assembly Language Source File* is missing the

appropriate ending “*END*” operation. This error is caused only by the absence of

the “*END*” operation , which is necessary to end an program for this assembler. To

correct this error, check input records for missing “*END*” and add it at the desired

program end.

- “**Error: Label on END line.”**

**Information and Recovery:**

The *Assembly Language Source* contains a label on an END record.To

correct this error, remove any labels from END records.

- “**Error: Multiple ORI lines.”**

**Information and Recovery:**

The *Assembly Language Source* contains multiple ORI source record lines.To

correct this error, assess source code for correct ORI usage and remove any addition ORI

operations.

- “**Error: No label on EQU line.”**

**Information and Recovery:**

The *Assembly Language Source* contains an EQU record with no label.To

correct this error, add desired label to incomplete EQU source file record.

- “**Error: No operand on EQU line.”**

**Information and Recovery:**

The *Assembly Language Source* contains an EQU record with no operand.To

correct this error, locate incorrect source file record and add desired operand .

- “**Error: Operand on EQU line is not a symbol or an integer.”**

**Information and Recovery:**

The *Assembly Language Source* contains an EQU record with an incorrect

operand.To correct this error, locate incorrect source file record and add desired

symbol or integer operand .

- “**Error: EXT and END Pseudo ops cannot have labels.”**

**Information and Recovery:**

The *Assembly Language Source* contains an EXT or ENT record with a label

field.To correct this error, locate incorrect EXT or ENT source file record and

remove any Label .

- “**Error: EXT and END Pseudo ops must contain at least one**

**operand.”**

**Information and Recovery:**

The *Assembly Language Source* contains an EXT or ENT record with no

operand field.To correct this error, locate incorrect EXT or ENT source file record and

add the desired symbol to the empty operand field .

- “**Error: "Error: Neither EXT or ENT can contain more than four operands.”**

**Information and Recovery:**

The *Assembly Language Source* contains either an EXT or ENT record with

more than four operands (symbols) in the operand field.To correct this error, locate

incorrect EXT or ENT source file record and remove the desired symbols from the

operand field .

- “**Error: "Error: ENT and EXT operations can only be preceded by the operation ORI.”**

**Information and Recovery:**

The *Assembly Language Source* contains either an EXT or ENT record outside

of its allowed range.All EXT and ENT must occur after ORI but before any other

operations. To correct this error, locate the incorrectly placed EXT or ENT source file

record and relocate.

- “**Error: ENT symbol not used in source file.”**

**Information and Recovery:**

There is an ENT symbol written in the *Assembly Language Source*

*File* that is not labeled in the program. To correct this error, check source file for the

missing symbol reference and correct.

- “**Error: \_\_\_ is an unknown operation.”**

**Information and Recovery:**

There is an incorrectly written operation in the *Assembly Language Source*

*File*. To correct this error, check source file for the given incorrect operation and

correct spelling, case or syntax.

- “**Error: Missing operation”**

**Information and Recovery:**

There is a missing operation in the *Assembly Language Source File*. This

was caused by an improperly written line. To correct this error, check source file for

missing operation or inappropriately short records.

- “**Error: Number of Literals exceeds 50.”**

**Information and Recovery:**

The *Assembly Language Source* has over 50 literals which is the upper limit for

this assembler.To correct this error, amend source file to reduce number of literals in

records.

- “**Error: Location Counter outside of 255 bound.”**

**Information and Recovery:**

The *Assembly Language Source* has exceeded the 0-255 memory location

provided by the W10\_560 machine. Please review the source program and adapt to be

contained my memory restrictions.

- “**Error: Number of Symbols exceeds 100.”**

**Information and Recovery:**

The *Assembly Language Source* has over 100 symbols which is the upper limit

for this assembler.To correct this error, amend source file to reduce number of symbols

in records.

- “**Error: Number of Source Records exceeds 200.”**

**Information and Recovery:**

The *Assembly Language Source* has over 200 source records which is the upper

limit for this assembler.To correct this error, amend source file to reduce number of

symbols in records.

- “**Error: R field integer is out of range.”**

**Information and Recovery:**

The *Assembly Language Source* contains a record with an incorrect usage of

the R field. To correct this error, check source program and ensure that no R field is

outside of the range 0-3.

- “**Error: R field is not a symbol or an integer.”**

**Information and Recovery:**

The *Assembly Language Source* contains a record with an incorrect usage of

the R field. To correct this error, check source program and ensure that every R field

contains either an integer or a correct symbol.

- “**Error: Invalid X field operand.”**

**Information and Recovery:**

In the *Assembly Language Source File* a source file record contains an invalid X field. To correct this error, locate source file records that contain X fields and make necessary correction.

- “**Error: Invalid S field operand.”**

**Information and Recovery:**

In the *Assembly Language Source File* a source file record contains an invalid S field. To correct this error, locate source file records that contain S fields and make necessary correction.

- “**Error: Operand in CCD line is not two characters.”**

**Information and Recovery:**

In the *Assembly Language Source File* a source file record containing the CCD pseudo op does not contain two characters in the operand field. To correct this error, check source file records for improper operands in CCD lines.

- “**Error: Program is relocatable; symbol on END line must be relative.”**

**Information and Recovery:**

The *Assembly Language Source* contains an inappropriate symbol on END line record. To correct this error, make sure the symbol used with END, if EQU has no operand, is relative. (Eg. Symbol does not act as label for EQU integer line)

- “**Error: Operand on EQU line is not a symbol or an integer.”**

**Information and Recovery:**

The *Assembly Language Source* contains an EQU record with an incorrect

operand.To correct this error, locate incorrect source file record and add desired

symbol or integer operand .

- “**Error: Operand on RES line is not an integer.”**

**Information and Recovery:**

The *Assembly Language Source* contains a RES record with a non-integer

operand.To correct this error, locate incorrect source file record and add appropriate

integer operand .

- “**Error: A label must be provided on the ORI line.”**

**Information and Recovery:**

The *Assembly Language Source* is missing a necessary label for the ORI line.

To correct this error, check first non-commented record in source file and ensure record

is labeled.

- “**Error: Invalid operation \_\_\_.”**

**Information and Recovery:**

There is an incorrectly written operation in the *Assembly Language Source*

*File*. To correct this error, check source file for the given incorrect operation and

correct spelling, case or syntax.

- “**Error: Label \_\_\_\_\_\_ cannot be used more than once.”**

**Information and Recovery:**

A label used the *Assembly Language Source File* has been repeated. For

this assembler, symbol names cannot be used more than once. To correct this error,

check input records and eliminate duplicate symbol names.

- “**Error: \_\_\_\_\_\_ has incorrect label format.”**

**Information and Recovery:**

There is an incorrect label in the *Assembly Language Source File*. This

was caused by an improperly written line. To correct this error, check labels/symbols

for missing alphabetic first character.

- “**Error: No operands present with machine op.”**

**Information and Recovery:**

In the *Assembly Language Source File* a machine operation was used with no

associated operand. To correct this error, check source file records that use machine

operations for missing operands.

- “**Error: Not enough operands with machine op.”**

**Information and Recovery:**

In the *Assembly Language Source File* a machine operation was used with

too few of associated operands. To correct this error, check source file records that use

machine operations and ensure all use correct amount of operands.

- “**Error: Operand on NMD line is not an integer.”**

**Information and Recovery:**

In the *Assembly Language Source File* a NMD pseudo op was used without an associated integer operand. To correct this error, check source file records that use NMD for non-integer operands.

Note: For addition reference of syntax or structure information, please refer to above **Input and Output** and **Prerequisites** sections.

**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: File\_\_\_\_\_ 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.