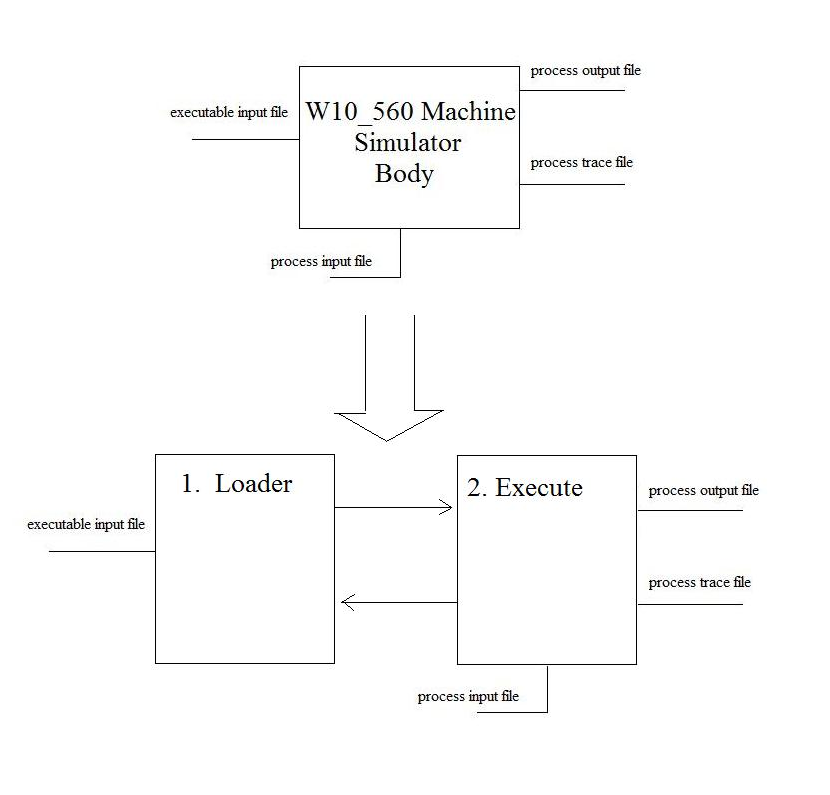
USER’S GUIDE:

**🡪Overview:**

The program **simulator** was created to act as an accurate and reliable simulator for the W10-560 machine. This program can be broken down into two major components. The first component functions as a loader. Its basic design will require an input file of desired machine code called the *executable input file*. This input will be filtered and stored into the appropriate memory and registers.

The second component will run through coded memory operations and simulate the W10\_560 machines operations. This includes an additional input, the *process input file*, which will be read with the appropriate command. This component also produces two output files, the *process output file* and the *trace output file*. The *process output file* will contain any desired output designated by the commands in *executable input file* as well as reported errors. The process trace file will contain a trace of each instructions execution. This will also include a content display of the memory and the registers.

In the figures below, the structure is highlighted.



Technical specifications and additional information for the W10\_560 machine can be found in the MACHINE\_GUIDE.pdf, located in the docs directory.

**🡪Notes Of Usage:**

Usage of the program **w10\_560\_machine.h** requires a 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.

-

**🡪Input and Output:**

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

- *executable input file*

- *process input file*

- *process output file*

- *trace 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.) The *executable input file:*

The primary function of the *executable input file* is to directly prepare the memory for simulation in the W10\_560 machine. The W10-560 machine instructions are written into a machine code equivalent which is the format for this input.

An example of a proper machine code usage in the *executable input file* is shown below.

H66SAMPLE6412

T6500002

T66B0064

T67B1000

T68B2064

T6990008

T6AB1000

T6B90004

T6CB3000

T6D90008

T6EB3000

T6F01074

T7000474

T7131065

T7290402

T73C0C00

T7400001

T7500005

This machine code input must include one Header Record on the first line of the input followed by Text Records. Specifications for Header Record and Text record can be seen below.

1. 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

2. 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

2.) The *process input file*

The *process input file* accessed and processed through the I/O machine instruction. This file should consist of lines of characters, ranging from ASCII range 32 to 126 and printable tab 9. An example input is shown below.

9 15 455 65

3.) The *process output file*

The *process output file* has two main purposes. The first is to produce the appropriate output for the *process input file*. The second purpose is to report the error conditions of the users input.

4.) The *trace output file*

The *trace output file* acts as the final “printout” of commands run and memory contents. Specifically, it will contain a sample trace of each command, its instruction information and its affect on memory as well as the above mentioned complete memory print out. A sample output is shown below.

MEMORY

Address | Content

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

00 | 0

01 | 0

02 | 0

.

.

.

62 | 0

63 | 0

64 | 0

65 | 2

66 | 720996

67 | 724992

68 | 729188

69 | 589832

6A | 724992

6B | 589828

6C | 733184

6D | 589832

6E | 733184

6F | 4212

70 | 1140

71 | 200805

72 | 590850

73 | 789504

74 | 1

75 | 5

76 | 0

77 | 0

78 | 0

**.**

**.**

**.**

FE | 0

FF | 0

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

REGISTERS

r0 | r1 | r2 | r3 |

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

**🡪Running the Program, “simulator”:**

Note: The program can be run through the **stdsun**interface. The following guide will begin from the main program folder.

1.) “simulator” is designed to run with user ease, so operation is controlled from a command line input. From the correct program main folder, run the program by entering the desired input files and designate the chosen output locations in the command line as shown in the syntax below:

% **simulator** Executable-input Process-input Process-output Trace-output

An example command run would be:

% simulator execIn processIn processOut traceOut

Note: For Additional information, from the main program folder Lab2, the following options are available.

docs: Listing of all provided program documentation.

src: All program coding

test: Listing of program testing cases

2.) After running “simulator”, command prompts may ask for confirmation of output file overwrites, as shown below:

Warning: File ‘processOut’ already exist, overwrite?(y/n)

Warning: File ‘traceOut’ already exist, overwrite?(y/n)

If entered correctly, the simulator will next report the status of the machine.

Creating machine…

Starting machine…

Then finish with:

Machine finished.

3.) After the “Machine finished prompt, check the appropriately chosen output locations for w10\_560 machine simulation results. To verify output, compare with given output syntax in the Inputs and Outputs section.

**🡪Troubleshooting:**

The program w10\_560\_machine.h is designed to report errors when the input is run. Errors to the input, both syntax and memory allocation, will be reported to the *process output file*. Below is a list of error messages and information for error recovery.

- “**Execute Load Error”**

Recovery:

The *executable input file* did not load correctly. An additional error will be

provided, please reference the following errors messages, (Denoted with “--”)

-- **”invalid contents: HeaderRecord Error”**

Recovery:

Check the *executable input file* header line for syntax of address errors.

For reference, view Input and Output section above.

-- **“invalid contents: TextRecord Error”**

Recover:

Check the *executable input file* text line for syntax of address errors.

For reference, view Input and Output section above.

-- “**invalid contents: Address already allocated”**

Recovery:

Check the *executable input file* for address repetitions in text

lines. For reference, view Input and Output section above.

- “**No Operation”**

Recovery:

This error means that an accessed memory address had no operation, or an instruction

was improperly written. Check executable input for instruction or addressing mistakes.

- “**Address range error.”**

Recovery:

This error is the result of improper memory usage. This could include an attempt to

access a memory location beyond the given range of 256 addresses

- “**Attempt to divide by zero”**

Recovery:

This error is the result an improper division instruction.

- “**Error: Cannot shift more than 19 bits”**

Recovery:

This error is the result an improper shift instruction.

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

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.”**

Recovery:

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

Check the listed file.

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

Recovery:

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

Check the listed file.

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

Recovery:

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

Check the listed file.

**- “Command line argument error.”**

Recovery:

Check the amount of arguments in the command line.

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

Recovery:

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

Also check for write ability.

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

Recovery:

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

rather a directory. Check the listed file.