Readme for Lexathon

Lexathon is a MARS (MIPS assembly) implementation of the smartphone game of the same name. The objective of the game is to find as many words as possible in a three-by-three grid of characters, each of which must contain the middle letter in the grid—and to do all of this before the clock runs out.

# How to Build and Run this Program

The main Lexathon program is contained in two files: lexathon.asm and hashtable.asm. Before running Lexathon, ensure these two files along with "hashtable.dat", "foundwords.txt”, “ninechar.txt”, and the MARS jar itself are all in the same folder.

Afterwards, build and run lexathon.asm (not hashtable.asm); Lexathon will run.

# How to Play

Immediately upon running the program, you will see a splash screen. After pressing any key, the game will start immediately! You will see a puzzle and prompt that looks like this:

/-------------\

| s o o |

| s h b |

| k a c |

\-------------/

You have 120 seconds remaining. Current Score: 0

Use the letters in this grid to form words. The trick, though, is that you can’t play just any words: they have to use the middle letter and be four characters or longer. For example, in the example above, “books” uses characters in the puzzle but does not gain any points, since it does not use the middle letter, h. “hooks” is a valid word.

When you see a word, simply type it and press Enter. Lexathon will either congratulate you and award you points and extra time or, if your word did not meet all of the conditions, tell you so. If you are typing in a nine-letter word, you will not need to press Enter.

When time is up, Lexathon will present the list of words that you found, along with your final score. After you press any key, the game will restart from the splash screen.

# Commands

Commands are used to quit the game and to perform other special actions. All commands begin with !; typing !h at the word prompt will provide a complete list along with a short description for each. The commands are:

!q - Quit game. Exits the program.

!t - Display current time

!r - Restart game. Returns to the splash screen.

!s - Shuffle game board: does not change the word or the middle character.

!h - Display short help message.

!e - End Game: shows words found and final score, and then restarts the game.

# Features and Limitations

Lexathon implements many of the features of the original game. Time taken by the user is recorded using MARS syscalls and score is kept according to the original game’s algorithm; the “shuffle” function was also implemented. A hash table, implemented in MIPS code, was written and used to look up words very quickly.

The MARS console is an awkward stage for a program as graphical as Lexathon, and therefore some aspects of Lexathon fall short of the original. The user cannot see the clock ticking, and the puzzle must be reprinted to the screen each time the user submits a word, because characters cannot be written to arbitrary locations on screen.

A few features from the original game were not included, such as the list of invalid words the user guessed along with the list of words that were in the puzzle which the user did not guess. Finding an exhaustive list of words that could be formed by a given nine-character string would be a computationally intensive task given the architecture of the dictionary system: each possible permutation of letters in the puzzle would have to be checked using the hash table. This may not have been a problem if a different data structure had been used to store the words.

# Rebuilding the Hash Table

Lexathon uses a hash table structure to quickly look up English words. The hash table, located in hashtable.dat, was itself created by a MIPS program, contained in hashtable\_builder.asm; the hashtable builder uses strings in the file dictionary.txt. These must all be nine characters, padded with the "`" character. hashtable\_builder.asm can be built and run. Before doing so, however, ensure it is in the same folder as all afore-

mentioned files, and delete hashtable.dat.