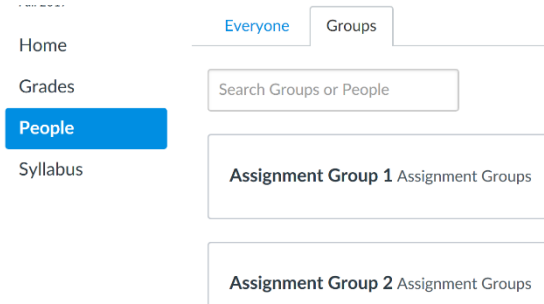


CS 131 Assignment 7

igPay atinLay Generator

Note: This is a group assignment with one to three students in a group. Each student must join a Canvas Group (Assignment Group 1 – Assignment Group 30) even if a student wants to be a group of one. The group leader shall be responsible for uploading the deliverable items (the first student that joins a group is the group leader).



Objectives: To learn how to program in the LC-3 assembly language, including:

- assembly language instructions
- psuedo-ops
- TRAPS
- basic I/O

1. Background

From Wikipedia: “Pig Latin is a language game in which words in English are altered. The objective is to conceal the meaning of the words from others not familiar with the rules.”

To translate a word from English to Pig Latin, move the starting letter of the word to the end of the word and then add “ay” to the word. Examples:

Hancock translates to **ancockHay**

binary translates to **inarybay**

2. Assignment

Implement, in LC-3 assembly language, a Pig Latin generator.

2.1 Input

Your program should prompt the user for the word to be translated:

Prompt: `English Word:`

- The user is to input a word of no more than 20 characters, followed by the <ENTER> key. A real world program should not allow for more than 20 characters to be entered but for this program, **range checking for too many characters is NOT necessary.**
- Store the word as one character per memory location, the word must be stored in memory (i.e., in an input buffer).
- The input buffer must be pre-allocated using a `.BLKW` directive. **Points will be deducted if you do not use the `.BLKW` directive.**

Hint #1: Using a TRAP x20 (GETC) followed by a TRAP x21 (OUT) will get a character from the keyboard and echo the character back to the console.


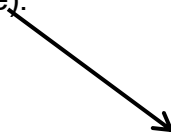
Hint #2: The <ENTER> key is mapped to the ASCII line feed character (ASCII x0A) in the Windows LC-3 simulator (and to ASCII x0D in the Unix Simulator).

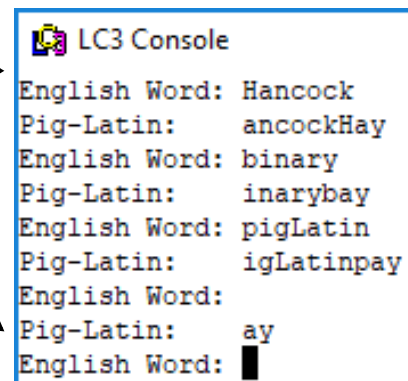
Hint #3: For reasons that will become clear to you later, **do not use R7** to store anything for this assignment.

2.2 Algorithm

1. Remove (do not print) the first letter of the word and add it to the end of the word.
2. Add “ay” to the word.

2.3 Output

- The program must display the prompt “English Word: ” and then wait for user input. The user input must be echoed to the screen.
- The program must display the string “Pig-Latin: ” and then display the translated word (see below).
- The translated word should be output to the screen and does not need to be stored in memory.
- After outputting the translated word to the screen, the program should prompt the user again, creating an endless loop.
- Example screen capture: 
- If no characters are entered, the output should be “Pig-Latin: ay” (edge case). 



```
LC3 Console
English Word: Hancock
Pig-Latin:   ancockHay
English Word: binary
Pig-Latin:   inarybay
English Word: pigLatin
Pig-Latin:   igLatinpay
English Word:
Pig-Latin:   ay
English Word: 
```

2.4 Format. Your program must be a single text file of LC-3 assembly code (i.e., a .asm file).

- Include a header block at the top of your source code with all of the pertinent information (see Assignment 5, Source Code Format for further details).
- Add comments to “blocks” of your code (you do not need to comment every line).

2.5 Extra Information

- The starting location shall be x3000.
- Your program must loop and run correctly multiple times in a row.
- Name your assembly file: `piglatin.asm`.
- Please make sure **each student's name is in the header block** of the source code.

3.0 Deliverables

Run your program with at least three different words (your choice), **test the edge case (when no character is entered)**, and then have your Group Leader submit the following files on Canvas:

1. A screen capture of your program's output (the console window).
2. Your source code (`piglatin.asm` file).

Please examine the symbol table (`.sym`) and listing file (`.lst`) generated by the LC-3 assembler. Make sure you understand the relationship between the two files.