CSC 236 Programming Assignment

ARM

Summary

The ARM assignment is worth 4% of your grade. For this assignment, you will be submitting a file named **arm.ans**. This file is created by the grading system when you grade your program. The grade is based on correct answers, documentation and the number of instructions actually executed during testing.

For this assignment, you can work in a team of two if you want to. If you are working on a team, create a single **arm.ans** file. Each team member should submit a copy of this file to the ARM assignment in Moodle..

If you have questions about the assignment, feel free to use our Piazza forum, but be sure not to post any code from your solution in a public piazza post (even code that's not working). You can also stop by during office hours to get help from anyone on the teaching staff. Remember, if you need help, you need to make sure your code is documented, including block comments above sections of your code and line comments saying what you are trying to do with each instruction. We'll also expect you to have a design for the program that you're trying to follow.

Program description

For this assignment, you will be writing a small program in 32-bit ARM assembly language. The program is called *armkey. It's a lot like the KEY programming assignment we already did earlier. Of course, it will use ARM instructions and* ARM Software Interrupts (SWI) to access text files for input and output (rather than reading and writing to the terminal).

The *armkey* program will open an input file named *key.in* and an output file named *key.out*.

The program then a reads a line of ASCII text, possibly containing printable characters and control characters (00h-7Fh) from the file *key.in* into an input string. The read string ARM SWI will remove any end-of-line characters and replace them with a single string terminator (\0). If there are no more lines the read string ARM SWI will return a count of zero for the number of bytes read.

The program will process the characters in each line. For each character the program performs the following:

- If the character is an upper case letter (A-Z) then copy it to the output string.
- If the character is a lower case letter (a-z) then convert it to upper case and copy it to the output string.
- If the character is a blank (20h) then copy it to the output string.
- If the character is a hex zero (00h) then copy it to the output string. This will put a null terminator at the end of the output string.

- If the character is anything else then do not copy it to the output string; just ignore that character. This includes any control characters in the range of 01-1Fh including the DOS end of file character 1Ah. After processing all characters on the input line, write it and a carriage return and line feed to the output file.

  The program will continue to read and process input lines until the read string ARM Software Interrupt returns a count of zero for the number of bytes read. This indicates that you've reached the end of file on the input. The program should close the input and output files and terminate.

Important Notes

1. The specification requires you to include the code to close the files. Do not omit this code. Omitting this code will not be detected by the grading program. It will, however, be detected by the staff examining your program and will cost you some points when your work is graded.
2. You only need to handle standard ASCII characters in the range of 00h-7Fh.
3. ARM is not DOS. The DOS 1Ah end-of-file character is ***not*** used to indicate the end of the input file. Treat the 1Ah as any other character that is not copied to the output string. Since it is between 01h and 1Fh, you should throw it away and continue to the next character.
4. There will be at most 80 characters on any input line, including the CR and LF.

Example Input	Example Output
Hello, testing 123.0Dh0Ah 1Ah0Dh0Ah Line 30Dh0Ah	HELLO TESTING 0Dh0Ah 0Dh0Ah LINE 0Dh0Ah

Development Environment

The course homepage has instructions for installing ARMSim in a few different environments. This is fairly easy in Windows and Linux but it looks like it doesn't work with current versions of MacOS. If you're using a Macintosh, you will need to complete this assignment on a different system. Fortunately, there are several alternatives you can try, including virtual machines, VCL images or a team partner with a Windows or a Linux machine.

Completing the Assignment

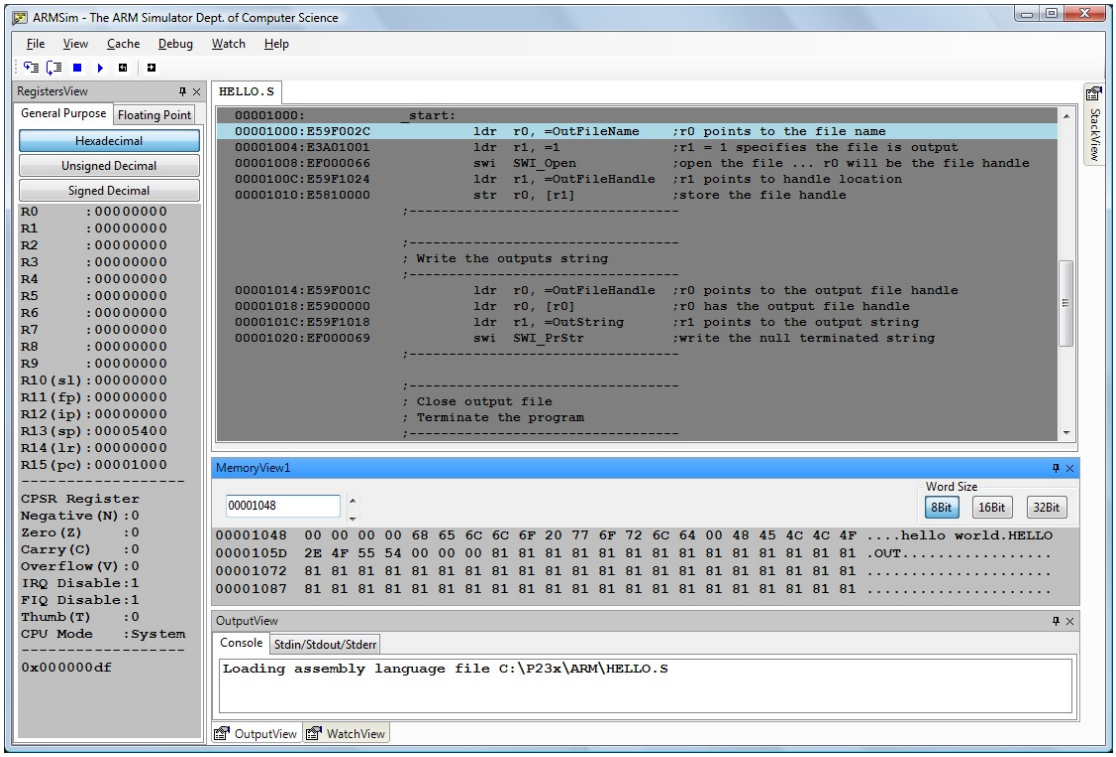
Although you will use ARMSim to run your ARM program, the grading and starter files will still be accessed through DOSBox. You should already have an ARM subdirectory in your DOSBox P23X directory. Download the self-extracting archive, **unpack.exe**, from the course homepage, copy it to your ARM directory under **P23X**. Start DOSBox, change to the \P23X\ ARM directory and unpack the files by running the **unpack.exe** executable. This should expand to all the files you need, including the HELLO.S program and the other two examples.

From your host, you can run ARMSim. Select the "File" menu then "Load …". Open the source file for HELLO.S. If you're running ARMSim in Linux, you may need to rename this

source file to give it a lower-case .s as the file extension. The source file should automatically be assembled as it is loaded by ARMSim.

There are buttons at the top of the ARMSim window. Pressing F11 will step through the code. Pressing F5 will run the program until it terminates.

After the program ends, there should be a file named HELLO.OUT that has the one line of text “hello world”. For your own ARM programs, you can edit them in the editor of your choice on your host. Then, you can load and run them in ARMSim just like you did with the HELLO.S program.



Completing the ARM Program

Step 1. Create a design

The two sample programs copystr.s and copyfile.s will provide guidance in ARM programming. Remember that your program must close the input and output files before exiting.

Step 2. Code your solution.

Use a text editor to write and edit your program. Name your source file **armkey.s**

Step 3. Test and debug your solution.

There is a sample input file named **key.in** you can use for testing. You may modify it to try out other test cases if you’d like.

Use the ARMSim assembler/simulator to run your program. It will read the file **key.in** and write an output file **key.out** Use a text editor to look at the contents of **key.out** and make sure it's correct.

Step 4. Grading

Follow these steps to run the grader on your solution.

1. In the ARM directory under your P23X DOSBox directory, create an input file to test your arm code with. Do this by copying the **gradarm.in** file provided with the starter code to the name **key.in**
2. Put a copy of your **key.in** file in your ARMSim directory. This will let your arm code see it when it's running in ARMSim.
3. Using the ARMSim simulator, assemble and run your **armkey.s** program. The simulator will read the input file **key.in** (the one you copied from **gradarm.in**) and create an output file **key.out**
4. After running your solution in ARMSim, copy your arm code (**armkey.s**) and your output and input files (**key.in** and **key.out**) back to your P23X/ARM directory.

   It looks like the grading program may use the original **gradarm.in** file for grading, so make sure it's also there in your P23X/ARM directory when you run the grader. It was included with the starter, so, as long as you don't delete or rename it, it should still be there.

   If you have to run ARMSim on another system (e.g., on a VLC image) you will need to copy the grading input file to that system, run your program in ARMSim, then copy your source file and the output file back to your ARM directory under P23X before you run the grading program.
5. Start DOSBox and go to the ARM directory under P23X. Type the DOS command: **gradarm** The gradarm procedure will verify your **key.out** has the correct output for the input file **gradarm.in** Status information is written to a file named results.

Step 5. Submit your solution

For the ARM assignment in Moodle, you need to submit the output file created by the grading program. It's called: **arm.ans**

This is the only acceptable file. Although the grading system creates the file, you are responsible to assure its content is correct. This file should contain two other files concatenated together.

- The first file is the results file.
- The second file is **armkey.s** which is your source code.