ARM

Summary of the ARM assignment. The ARM assignment is worth 4% of the course grade.

There is 1 submission:

1. *Arm.ans* is worth 4% of the course grade. This file is created by the grading system when you grade ARM. Grade is based on correct answers. See calendar for the due date.
2. If you need help from the staff:
   - submit your assembler source code to the ASK4HELP locker
   - send a note to the staff stating you have submitted code to ASK4HELP
   - explain the problem to be addressed
   - Assure your code is fully documented with header blocks and line comments. The staff may request to see your design.

If working on a team … create a single arm.ans file
that is submitted by both members of the team to their respective submit lockers.

Specification for ARM

The ARM assignment is an ARM assembler program. The program name is *armkey*. The program uses ARM Software Interrupts (SWI) to access ASCII files. This is the specification for the program.

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

The program then a reads a line of ASCII text with 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 binary 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 processes the characters in that line. For each character the program performs the following:

- If the character is an upper case letter (A-Z) then move it to the output string.
- If the character is a lower case letter (a-z) then convert it to upper case and move it to the output string.
- If the character is a blank (20h) then move it to the output string.
- If the character is a hex zero (00h) then move it to the output string. This also signifies the end of the input string.
- If the character is anything else then do not move it to the output string, it just throw the character away. 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 the output string and a carriage return and line feed to the output file.

The program continues to read and process input lines until the read string ARM Software Interrupt returns a count of zero for the number of bytes read which is the an end of file indication. The program closes the input and output file and halts.

For example if the input file were:

Hello, testing 123.0Dh0Ah
Line 20Dh0Ah
1Ah

The expected output would be:

HELLO TESTING 0Dh0Ah
LINE 0Dh0Ah
0Dh0Ah

Notes:

- You only need to handle standard ASCII characters in the range of 00h-7Fh.
- ARM is not DOS. The DOS 1Ah end of file character is ***not*** used to indicate the end of the input file. It will appear as a line of text with only one character. Treat the 1Ah as any other character that is not copied to the output string. Throw that character away and write out the null line terminated by a CRLF.
- The maximum number of characters in an input line, including the CR and LF, would be 80.

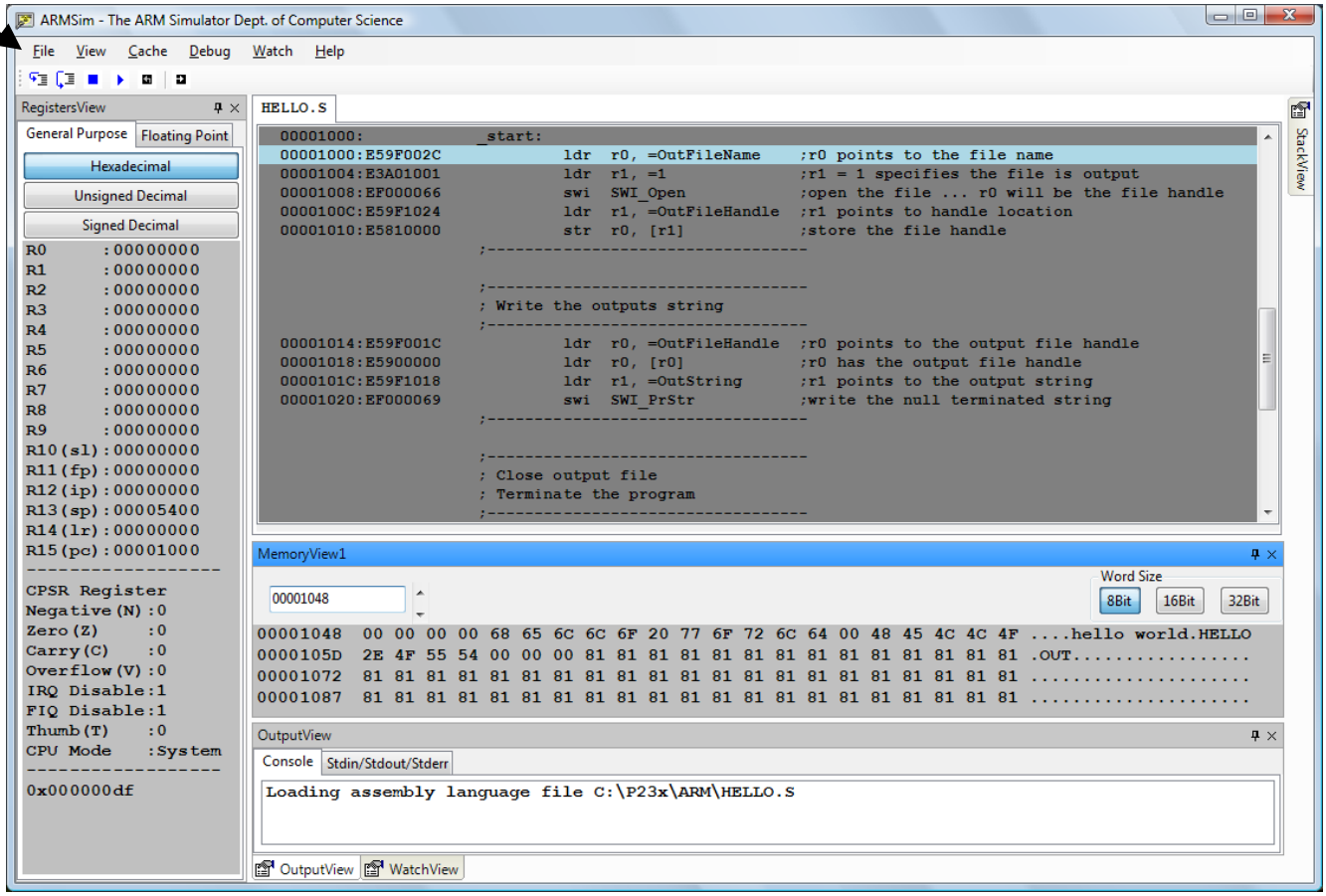
Install the ARMSim# assembler and simulator ... Windows instructions below ... see the web site for MAC or Linux

- Prepare the DOSBox directory

Create a DOSBox directory \P23X\ARM
Retrieve the testing and grading files. All files are packed together in one self-extracting file named *unpack.exe*.
Go to the class homepage. Click on *Lockers With Program Grading System Files.* Click on the *ARM* assignment.
Download the *unpack.exe* file and save it on your hard drive in the \P23X\ARM directory.
Start DOSBox, change to the \P23X\ARM directory, unpack the files by typing the DOSBox command: ***unpack***
This will create a number of sample ARM Assembler files. One will be *hello.s*

- Install the assembler/simulator

Go to the ARMSim# site ... http://armsim.cs.uvic.ca/
Select downloads and select ARMSim# 1.91
Download the installer ... *Installer.msi* for version 1.91
Run *Installer.msi* taking the defaults.
It will create a desktop icon.
Double click the icon ... Resize the window if desired.
Select *File* then *Load* … then load the file \P23X\ARM\HELLO.S
The file automatically assembles as it is loaded.
F11 will step through the code. F5 will run the code until it terminates.
After the program ends, there should be a file named HELLO.OUT that has the one line of text *hello world*



The steps to complete this program

Step 1. Create a design

The two sample programs *copystr.s* and *copyfile.s* will provide guidance in ARM programming.

Step 2. Code your solution.

Use a text editor to enter your program into ARM assembler instructions. Name your source file ***armkey.s***

Step 3. Test and debug your solution.

There is a sample input file named *key.in* which may be used for testing and it may be modified if you wish.

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 verify *key.out* is correct.

Step 4. Grading

Follow these steps:

1. **In the \P23X\ARM directory, generate the grading input file by copying the file *gradarm.in* to be *key.in***
2. **Using the ARMSim simulator, assemble and run your *armkey.s* program**
   The simulator will read the input file *key.in* (*gradarm.in*) and create an output file *key.out*
3. **Start DOSBox and go to the \P23X\ARM directory. Type the following 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*.

Once you have arm working correctly, you may make additional grading runs to improve efficiency or documentation. To mark these as grading runs that were only made to improve efficiency or documentation, execute this command:

testarm mark

All subsequent grading runs will be marked as only being done to improve efficiency or documentation.

Your final grade will be based on the following three components.

- 60 points for getting the correct answer.
- 20 points for the number of executable instructions written to *correctly* complete this assignment.
- 20 points for documentation of a program that functions *correctly*.

Efficiency and documentation are only a concern after the code works correctly.
Documentation grading will only occur after your code passes the functional test.
Efficiency grading will only occur after your code passes the functional test.

Step 5. Submit your solution

The file you electronically submit is: ***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.