MOV

Summary of the MOV assignment. The MOV assignment is worth 2% of the course grade.

There is 1 submission:

1. *Mov.ans* is worth 2% of the course grade. This file is created by the grading system when you grade MOV. Grade is based on correct answers and documentation. 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 mov.ans file
that is submitted by both members of the team to their respective submit lockers.

In 2013, Stephen Dolan, Computer Laboratory University of Cambridge, wrote a paper *mov is Turing-complete* that showed any x86 program can be written with just the *mov* instruction. Christopher Domas, Cyber Security Researcher Battelle Memorial Institute Columbus OH, then showed how the concept could be practically implemented.

The goal of the MOV assignment is to provide you with the opportunity to write a program just using the mov instruction (and int 21h to read/write data).

Two sample programs are provided to illustrate the ideas needed to code only using the mov and int instructions:

- IfThen.asm: reads two ASCII characters x and y; calculates if (x == y) then z = 'Y'; outputs z
- JmpMov.asm: sets x='0' then loops reading y and calculating and outputting x=x+1 as long as (y == '+')

Specification for MOV

- Read and echo 3 ASCII characters x y z

  x will be an ASCII uppercase letter ‘A’ - ‘Z’ or digit ‘0’ - ‘9’

  y will be an ASCII digit ‘0’ - ‘9’

  z will be an ASCII plus sign ‘+’ or other ASCII character
- Using only the mov instruction:

  - convert y from ASCII ‘0’ to ‘9’to hex 0 to 9

  - calculate … if (z == '+') then x = x + y
- Output x
- Terminate

For example if you type in this data:	The standard output should show these characters.
X Y Z	X Y Z X
A 3 +	A 3 + D
2 5 :	2 5 : 2
3 3 +	3 3 + 6
Z 0 +	Z 0 + Z

Notes to simplify the program:

The only characters that we will test with for x are the ASCII uppercase letters A to Z and ASCII digits ‘0’ to ‘9’.
The only characters that we will test with for y are the ASCII digits ‘0’ to ‘9’.

The steps you must take to complete this program

Step 1. Create a design

Since it is an unusual programming exercise, to code only using the mov instruction, I am providing you with the design I used to solve this problem. You are free to use it or improve it or use your own design.

Note that bx is the only pointer register that supports byte operations (into bl) so it will be used often to create word size pointers from byte size variables.

	Operation	Notes
Define Data	Declare byte variables: x, dummy_x, y, z	x and dummy_x must be contiguous in memory
	Create a lookup table to convert an ASCII digit to 0h-9h	
	Create a lookup table to perform the addition x + y	Copy the lookup table *inctbl* from the JmpMov.asm program
	Create an extra segment	Use the extra segment definition from the IfThen.asm program
Code	Read and echo the 3 ASCII characters x and y and z	
	Convert the ASCII value of y '0' to '9' into a hex 0 to 9	Load y into bl and use bx as an index into the conversion table to get the hex value of y Save that hex value of y in si to be used later as a double index
	Calculate x + y in the al register	Load x into bl Use the double index of bx + si to index into the inctbl table to set the al register to the sum of x + y

	If (z=='+') then x = x + y	Use the code from IfThen.asm as a model:
		if (z == '+') then x gets the value in al which is x + y
		if (z != '+') then dummy_x gets the value in al

	Output x	
	Terminate	

Step 2. Code your solution.

Use a text editor to convert your design into *mov* and *int* assembler instructions.

Your source code file must be named ***mov.asm*** and the executable program will be named *mov.exe*.

MASM
`ml  /c  /Zi  /Fl  mov.asm`
`link  /CO  mov.obj`

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 *MOV* assignment.
Download the *unpack.exe* file and save it on your hard drive in the \P23X\MOV directory.
Start DOSBox and execute unpack.exe to create the testing and grading files.

Step 3. Test and debug your solution.

Decide what input data should be typed into MOV to assure MOV functions correctly.

To run a test, type the following DOS batch command: ***testmov***

The *testmov* procedure will run your MOV program. It will allow you to enter the input data from the keyboard.

Step 4. Grading

MOV is 100% self-grading.

The system needs access to your source file and executable file. Assure ***current*** versions of ***mov.asm*** and ***mov.exe*** are located in the current directory. The *gradmov* procedure will run the grading program and give you a grade.

Type the following DOS command: ***gradmov***

Status information is written to a file named *results*. If a defect is found then the *results* file will tell you which test failed.

Your final grade will be based on the following components.

- 80 points for getting the correct answer and only using the mov and int instructions.
- 20 points for documentation of a program that functions *correctly*.

Documentation grading will only occur after your code passes the functional test.

How do you improve documentation without being accused of running the grading program too often?

After you have MOV working correctly, you may make additional grading runs to improve documentation. To mark these as grading runs that were only done to improve documentation, execute this command, one time, after your program has earned the points for getting the correct answers: ***testmov mark***

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

Step 5. Submit your solution

The file you electronically submit is: ***mov.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 ***mov.asm***, which is your source code.

When you look at mov.ans to verify its contents, you should find this line of text with your grade.
++ Grade ++ *nnn* = Total grade generated by the Grading System.
If that line is not in the file then do not submit it. You have a problem. You must resolve that problem.