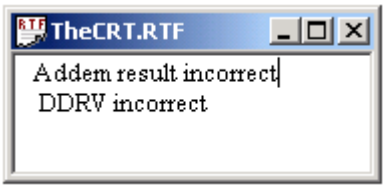
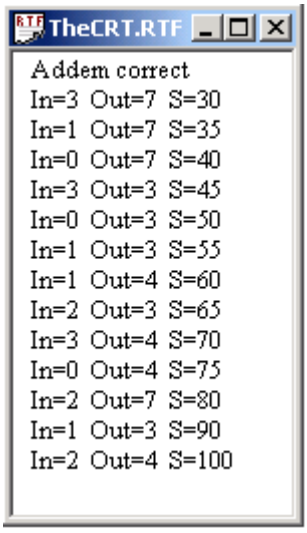
Procedure

First, you will log onto the computer and download files from the web as instructed by the TAs. Please put the three starter files called **`Exam2B.rtf Exam2B.uc Exam2B.io`** into the **`temp`** folder on drive D. You are not allowed to archive this exam. Within **TExaS** open these files, put your name on the first comment line. Before writing any code, please assemble and run the system. You should get output like this figure on the right (with no score). Each time you assemble, **TExaS** will create a backup version of your program. If you wish to roll back to a previous version, simply open one of the backup versions. If you do roll back, I suggest you perform a **SaveAs**, so a new sequence of backup files will be started.



You will write one subroutine and one FSM controller. My main program will call your subroutine, it will give you a grade on this subroutine, then my program will jump to your FSM controller. During each loop of your FSM controller, you will call my grader subroutine and points will be awarded. When my grader subroutine is done testing your FSM controller it will output a performance score of 0 to 100. *You should not modify my main program or my example data.* When you have written your subroutine and FSM controller, you should run my main program, which will output the results to the **`TheCRT.rtf`** window. You are NOT allowed to create global variables. After you are finished, raise you hand and wait for a TA. The TA will direct you how and when to print your source code. You will run your program in front of the TA, who will record your performance score on your exam paper. Please sort all materials in this order: 1) this paper, 2) software source code printout, and 3) all scratch work. These papers will be stapled together and turned in. The scoring page is the only work that will be returned to you.



Part a) The first subroutine, called **`Addem`**, will add two 16-bit numbers and return the 16-bit sum. You do not have to worry about overflow. One example is:

```
DataSet fdb 1000   ; first parameter
        fdb 2000   ; second parameter
```

Input parameter: The data set is passed call by reference on the stack

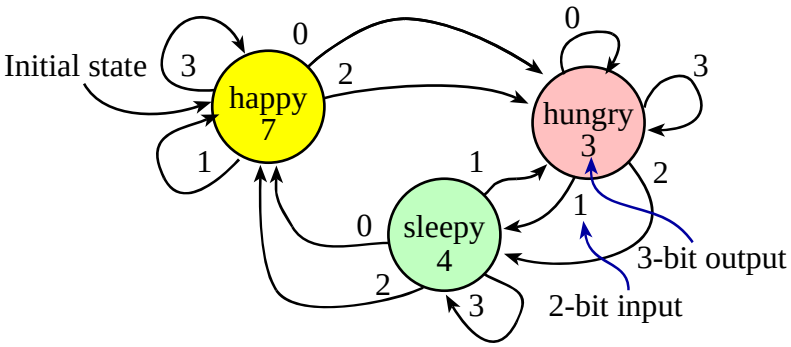
Output parameter: The result is returned as a value on the stack

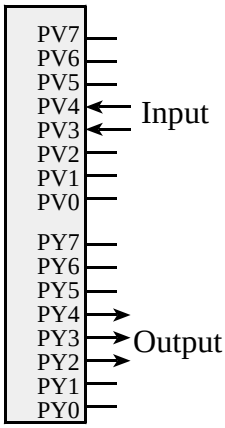
Error conditions: none

A typical calling sequence is

```
movw #DataSet,2,-sp ;call by reference on the stack
leas -2,sp          ;place for your 16-bit result
jsr  Addem          ;your subroutine
puld                ;your result from the stack
leas 2,sp           ;balance stack
```

Part b) Next, you will write a controller for a Moore FSM. The inputs are on Port V bits 4 and 3, and the outputs are on Port Y bits 4, 3, and 2. For example, if you are in the sleepy state the output is 4. If the input is 2, then go to the happy state. The output values are shown in decimal. You will leave the IO window open but minimized, because you will not interact with the IO window. The input/output testing will be performed by my grader subroutine.





You may not use global variables. Allocate one local variable to hold the current state pointer. Additional local variables can be allocated. For each local variable please include binding, allocation and access (since your FSM controller loops over and over, the local variables will never be deallocated. All I/O accesses should be performed in a friendly manner. Your FSM controller does not execute **`rts`** (notice that my program invokes **`YourFSMController`** with a **`jmp`**). There are five error codes my grader program may issue

a Your DDRV is correct but not friendly
b Your DDRY is correct but not friendly
c You wrote to the input port PTV
d Your PTY output is incorrect
e Your PTY output is correct but not friendly

My grader program will stop your FSM controller if the direction bits on any of the two inputs or three outputs are incorrect. There are three parts to this FSM controller.

Part 1) Convert the Moore FSM graph to a linked data structure (a graph) and store it in EEPROM.

Part 2) Write assembly software to initialize the system. You should create local variable(s) with binding, initialize ports, and initialize variables. The initial state is happy. Place your initialization between **`YourFSMController`** and **`loop jsr Grader`**.

Part 3) Write assembly software to run the FSM. The proper sequence is output, input, and go to next state. Place your FSM engine between **`loop jsr Grader`** and **`bsr loop`**. Notice that the grader subroutine must be called before you execute the output, input, next sequence. The grader program will change all the registers, so you have to store the state pointer in a local variable on the stack. Your FSM controller will have the following structure.

```
; Part 1) put your data structure here
YourFSMController
; Part 2) put your initialization here
loop jsr  Grader ;do not remove this line
; Part 3) put your output-input-next engine here
     bra  loop    ;do not remove this line
```

First:_____ Last:_____

Scoring

Your grade will be based both on the numerical results returned by your program and on your programming style. In particular, write code that is easy to understand, easy to debug, easy to change. Please employ good labels, pretty structure, and good comments.

Program style, rtf file printed by TA			
	labels		
	pretty structure		
	comments, name on RTF		
Performance score, Run by TA at the checkout		S=	
total			

I promise to follow these rules

This is a closed book exam. You must develop the software solution using the **TExaS** simulator. You have 55 minutes, so allocate your time accordingly. You are allowed to bring only some pencils (no books, laptops, cell phones, hats, disks, CDs, or notes). You will have to leave other materials up front. Each person works alone (no groups). You have full access to **TExaS**, with all the **TExaS** examples and the **TExaS** help. You may use the Window's calculator. You sit in front of a computer and edit/assemble/run/debug the programming assignment. You do not have access the Freescale manuals, just the help system in **TExaS**. You may not take this paper, scratch paper, or rough drafts out of the room. You may not access your network drive or the Internet. You are not allowed to discuss this exam with other EE319K students until Thursday.

The following activities occurring during the exam will be considered scholastic dishonesty:

0) reading, writing or viewing any file outside of test files on the desktop
1) running any program from the PC other than **TExaS** or the Windows calculator,
2) communicating to other students by any means about this exam until Friday,
3) using material/equipment other than a pen/pencil.

Students caught cheating will be turned to the Dean of Students.

Signed: _____ March 2010

http://users.ece.utexas.edu/~valvano/Exam2B
user **LED** *password* **red**