

DESIGNING A RACING VIDEO GAME

PURPOSE: To design a time-intensive program to perform a given operation in the quickest time and most efficient way possible. This assignment includes the use of arrays, branching, polled i/o, and BCD arithmetic.

LAB ASSIGNMENT:

Write a program that simulates auto racing. The track should be displayed in the program i/o window of the debugger as follows:

```
***** X *****
*****
*****
*****
*****
*****
```

As the game executes, the track should scroll upward in the debugger window. Your car (the 'X') should only appear on the top line of the i/o window. Periodically, an opponent's car should appear (an 'I') at the bottom of the screen. It should then scroll up the screen with the rest of the track. If your car comes in contact with the edge of the track (a '*') or an opponent's car, you crash. The program should indicate that a crash occurred and terminate. After 100 lines of text have been printed (scrolled), the program should terminate.

Scoring: the score should be stored as a 4-digit BCD. Each time another line of text is printed, the player's score should increase by five. The score should be printed when the game terminates.

To control the car, the player should be able to press 'L' to go left one space and 'R' to go right one space. To stop the program the player should be able to press 'X'. Use polled i/o to check if the player has pressed a key. While the program is waiting for input, it should be scrolling the track.

Note the following:

1. The shape of the track, as well as the occurrence and location of the opponent cars, should be stored as a series of byte-sized constants.
2. This program is performing animation. To get the best results, the strings must be printed to the screen very quickly. CharOut is much too slow. Instead, use PutString to output a whole line at a time.

Questions:

1. Provide a flowchart and/or pseudocode and give an overview of how your program operates.
2. Explain the steps you took to make your program execute quickly and efficiently.
3. Explain how you printed the track and the opponent's cars.

