CENG 331

Computer Organization

Fall 2014-2015

Programming Assignment 1

Due date: October 31th, 2014, Thursday, 23:55:50

This assignment aims to get you familiar with MIPS and assembly programming. You will code a MIPS program that will further develop your skills in writing MIPS code, with a focus on implementing function and system calls.

1. Problem Definition

In this assignment, you are expected to write a program that moves a virtual **point robot** on a discretized environment. You will be given dimensions of the environment, starting coordinates of the robot, locations of obstacles and move commands. Your program will move the robot with respect to move commands considering the dimensions of the environment and locations of the obstacles. Finally, your program will return the coordinates of the last position of the robot.

Input:	**Output:**
3x5#(1,2)#(0,4),(2,3)#ACCAGGGACEEEFFDB#	(2,2)

Table 1: A sample run of the desired program

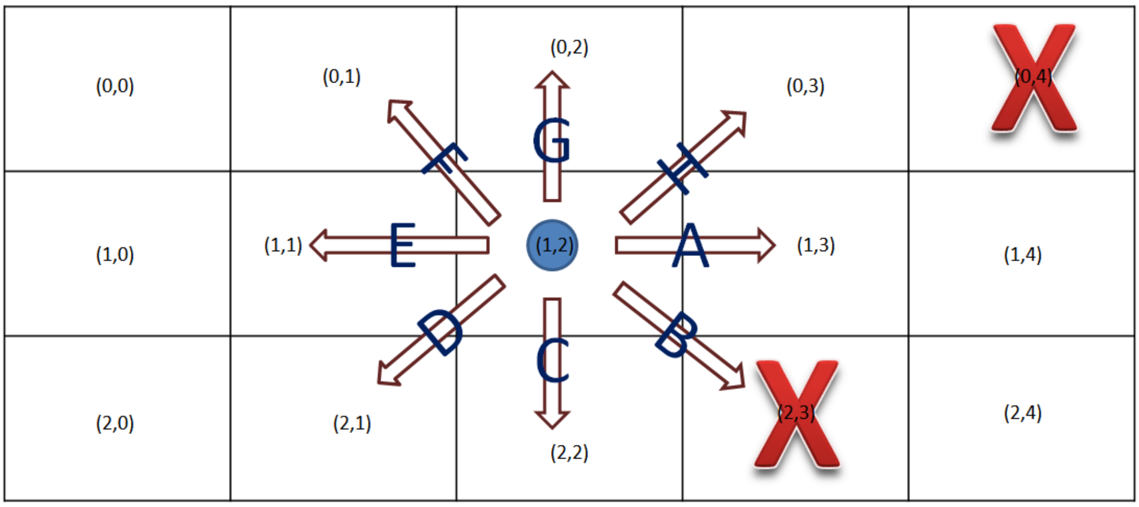

Figure 1: Visual representation of the environment in Table 1

Detailed explanation for the sample run (**Table 1**) is as follows:

- The character ‘#’ separates the input into parts.
- The first part shows the dimensions of the environment (3x5).
- The second part, (1,2), is the starting coordinates of the robot (1st row, 2nd column).
- The third part shows the coordinates of the obstacles. In our example there are two obstacles.
- The coordinates of the robot and obstacles will be shown with a two-tuple (e.g. 1,2). Here, the first integer corresponds to the row and the second integer corresponds to the column. Each two tuple corresponds to an obstacle and each obstacle fills only the related cell.
- Two tuples are separated by the character ‘,’.
- The last part contains a string that represents move commands. Each character of this string is an individual command (A, B, C, D, E, F, G or H). The directions of these commands are shown in Figure 1.
- If the robot receives one of the move commands, it moves from its current cell to one of its neighbor cell with respect to the move command.
- If the robot is in one of the border cells and the command leads to robot to the outside of the environment, the robot cannot move and keeps its position.
- If the target cell contains an obstacle, the robot cannot move and keeps its position.
- Finally, the program outputs the last coordinate of the robot ((2,2) in the example).

2. Specifications

You will read your input from stdin, and write your output to stdout. For these operations, you can use MIPS's operating system calls, namely read string and write string. Your program will first read a logic expression from standard input. The input consists of at most 200 characters.

When your program receives a move command: Firstly, it should check whether it is possible to move with respect to the command or not. At this point you should use the procedure “**check**:”. This procedure should take **at least one argument**: the move command. You can use additional arguments if you need. Then the procedure **must return ‘1’** if the target cell is available and **‘0’ otherwise**. Then, if it is possible, you will update the coordinates of the robot. **You must obey procedure calling convention of MIPS, otherwise, points will be taken off...**

Your output must be in this format: **(<value>,<value>).** Here, left parenthesis is followed by the coordinate of the row. Then, you should put the comma. After the comma, the coordinate of the column and the right parenthesis are put. There is no space before or after the parentheses and the comma. Make sure that you display the final coordinates on the next line after the last ‘#’ symbol.

The other specifications are as follows:

- The environment can be 2x2, 2x3, …, 2x9, …, 9x8, 9x9.
- The environment is indexed like a two-dimensional array whose first value corresponds to the row value and second value corresponds to the column value.
- The left top-most cell of the environment is always indexed with (0,0).
- Arbitrary number of obstacles can be given.

- In the environment, there will be at least one obstacle, and the input will contain at least one move command. In other words, in the input, you have four parts: dimension of the environment, starting coordinates of the robot, locations of obstacles and move commands. There will be at least one obstacle and one move command.

Notes

- We will not test your program with invalid input. The input to your program will be error-free.
- There will not be spaces in the input expression.
- **Write comments in your code.** Comments for each line are not required, but include enough comments in order to understand to steps (pseudo-code level) (Points will be taken off if missing).
- You will use **QTSpim** (Windows/Linux), or **Spim** (Linux console) to execute/debug your MIPS assembly code.

3. Submission

Submission will be done via COW. Submit a single MIPS assembly source file named hw1.s that can be run with the SPIM simulator.

Note: Your homework will be graded on department inek machines. The following command sequence is expected to run your program on a Linux system:

```
$spim -file hw1.s
```

4. Resources

- QTSpim: http://spimsimulator.sourceforge.net/
- D. A. Patterson and J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 3rd Edition (Chapter 2)
- Recitation Documents

5. How to run

```
$spim -file hw1.s
3x5#(1,2)#(0,4),(2,3)#ACCAGGGACEEEFFDB#
(2,2)
```

6. Regulations

1. Programming Language: MIPS

2. Late Submission: 8x(latedays)^2.

3. Cheating: We have zero tolerance policy for cheating. People involved in cheating will be punished according to the university regulations.

Cheating Policy:

Students/Groups may discuss the concepts among themselves or with the instructor or the assistants. However, when it comes to doing the actual work, it must be done by the student/group alone. As soon as you start to write your solution or type it, you should work alone. In other words, if you are copying text directly from someone else - whether copying files or typing from someone else's notes or typing while they dictate - then you are cheating (committing plagiarism, to be more exact). This is true regardless of whether the source is a classmate, a former student, a website, a program listing found in the trash, or whatever. Furthermore, plagiarism even on a small part of the program is cheating. Also, starting out with code that you did not write, and modifying it to look like your own is cheating. Aiding someone else's cheating also constitutes cheating. Leaving your program in plain sight or leaving a computer without logging out, thereby leaving your programs open to copying, may constitute cheating depending upon the circumstances. Consequently, you should always take care to prevent others from copying your programs, as it certainly leaves you open to accusations of cheating. We have automated tools to determine cheating. Both parties involved in cheating will be subject to disciplinary action. [Adapted from http://www.seas.upenn.edu/~cis330/main.html]

4. Newsgroup & Questions: You must follow the newsgroup (metu.ceng.course.331) for discussions and possible updates on a daily basis. Use the newsgroup for your questions.

5. Evaluation: Your program will be evaluated automatically using black-box technique so make sure to obey the input/output specifications.

6. The homework must be done individually, so there will not be any team work.