Ertugrul Kasikci 200104004097

Input Format:

The program asks the user to input followings; number of seconds, number of columns and number of rows. After that user enters the board sequentially without any spaces or newlines (if we have 3x3 board user needs to enter it as ….O….)

For example:

```
Enter the number of seconds:8
Enter the number of rows:3
Enter the number of colums:3
Enter the board as line (e.g. 3x2 board should be entered as ..O...):....O....
1
...
.O.
...
```

Program Logic:

I kept some important values in the s registers.

s0: number of seconds

s2: number of rows

s3: number of columns

I kept three different arrays. board, bomb_timer, bomb_to_detonate. board holds our current board. bomb_timer holds the number of second to detonation for ever cell. bomb_to_detonate keeps the index number of the bombs that will detonate at that second. By using bomb_to_detonate I handled the condition where we for example detonated the right bomb of the current bomb. But in this case, we cannot detonate the cell at the down of the right bomb. Because the right bomb detonated before. By keeping indexes of the bombs to detonate, I handle this issue. Even if the right bomb is detonated, I can detonate right bombs adjacent cells.

Program Flow:

After the input is entered, the program prints the board in that state for every second.

```
Enter the board as line (e.g. 3x2 board should be entered as ..O...):....O....
1. Second
...
.O.
...
2. Second
OOO
OOO
OOO
3. Second
O.O
...
O.O
4. Second
OOO
OOO
OOO
5. Second
```

I have these subroutines;

jal update_bomb_timers: Updates bomb timers in every second.

jal plant_bombs: Plants bombs to the empty cells.

jal determine_the_bombs. Finds the bombs that needs to detonate in that second.

jal detonate_bombs. Detonates the bombs and their adjacent cells.

jal print_board_with_second: Prints the board and current second.

j simulation_loop: The main loop of the program. All the above subroutines repeat in this loop until entered number of seconds is reached.

Edge Cases:

Detonate Up: I checked if the current index minus number of columns is bigger or equal to 0. If so, I executed up detonation. If it is smaller than 0, it means that we are at the top row which means there is no upper cell.

Detonate Down: I checked if the current index plus number of columns is smaller than the total number of cells. If so, I executed down detonation. If it is bigger than the total number of cells, it means that we are at the bottom row which means there is no upper cell.

Detonate Left: I checked using rem instruction if the remainder of the current index from the number of columns is equal to zero. If it is not I executed left detonation. If it is 0, it means that we are at the leftmost column which means there is no left cell.

Detonate Right: I checked using rem instruction if the remainder of the current index from the number of columns plus one is equal to number of columns. If it is not, I executed the right detonation. If it is, it means that we are at the rightmost column which means there is no right cell.