CSE331 COMPUTER ORGANIZATION
PROJECT 1

Yağız Hakkı Aydın
1901042612

1. **Register Initialization:**
   - The program begins by initializing specific registers with certain values:
     - `$s0`: This register serves as an array index for various loops. It starts at 0.
     - `$s1`: Contains the ASCII value of the character ``.'`, representing empty cells in the game grid.
     - `$s2`: Holds the ASCII value of the character ``O'`, representing bombs in the game grid.

2. **Messages Initialization:**
   - In the `.data` section, various messages are defined as null-terminated strings. These messages serve different purposes, such as user prompts, error notifications, and informational messages.

3. **Grid Dimensions and Second Counter:**
   - The program proceeds to obtain user input for the grid dimensions and the initial number of seconds to play the game.
   - Registers used to store these values:
     - `$s3`: Grid width.
     - `$s4`: Grid height.
     - `$t9`: The second counter, indicating how many seconds are left to play.
   - All of these values are read by loops, jump to next step from each loop if input is valid(min 2 for dimentions and min 3 for seconds), if input is not valid, then prints error message and continues after user gives a valid input.Output is following...

```
Enter grid width -> 1
Board width cannot be less than 2, please enter a valid width...
Enter grid width -> 5
Enter grid height -> 1
Board height cannot be less than 2, please enter a valid height...
Enter grid height -> 5
Enter number of seconds to play -> 10
```

4. **Array Allocation:**
   - The program allocates memory for two arrays that will be used to represent the game grid. The sizes of these arrays are determined by multiplying the grid width and height.
   - Registers and system calls used in this process:
     - `$a0`: Stores the number of bytes to allocate.
     - `li` and `syscall` instructions allocate memory for two arrays.
     - '$s6' array allocated with grid size (first array)
     - '$s7' array allocated with grid size (second array)

5. **Initializing the Grid:**
   - The program enters a loop to initialize both grid arrays.Iterates both first and second arrays by incrementing $s0 index.
   - First array gets '.' for all cells initially
   - Second array gets 'O' for all cells initially

6. **Getting Initial Bomb Positions:**
   - The program enters another loop to allow the user to specify positions for planting bombs. The user is prompted to enter a cell number where they want to plant a bomb.
   - The program checks the validity of the input:
     - Ensures that the entered position is within the grid's bounds.
     - Allows the user to end the bomb planting phase by entering `-1`.
   - If the input is valid, a bomb is planted in the corresponding cell of the second grid.
   - If the input is invalid, error messages are displayed.
   - This loop continues until the user indicates that they have finished planting bombs by entering -1.

```
Enter grid width -> 1
Board width cannot be less than 2, please enter a valid width...
Enter grid width -> 5
Enter grid height -> 1
Board height cannot be less than 2, please enter a valid height...
Enter grid height -> 5
Enter number of seconds to play -> 10
Enter cell number to plant bomb,enter '-1' to end planting -> 3
Enter cell number to plant bomb,enter '-1' to end planting -> 7
Enter cell number to plant bomb,enter '-1' to end planting -> 22
Enter cell number to plant bomb,enter '-1' to end planting -> 11
Enter cell number to plant bomb,enter '-1' to end planting -> 17
Enter cell number to plant bomb,enter '-1' to end planting -> 9
Enter cell number to plant bomb,enter '-1' to end planting -> 14
Enter cell number to plant bomb,enter '-1' to end planting -> 25
Enter cell number to plant bomb,enter '-1' to end planting -> 33
Cell position given by you is either higher than number of all cells or less than 1,please enter a valid cell position...
Enter cell number to plant bomb,enter '-1' to end planting -> -4
Cell position given by you is either higher than number of all cells or less than 1,please enter a valid cell position...
Enter cell number to plant bomb,enter '-1' to end planting -> -1
```

7. **Bombs Are Planted:**
   - After bomb planting is complete,it prints number of seconds left
   - Then goes print loop to print first grid
   - Then prints seconds left

```
10 seconds left to play...
. . . 0 .
. . 0 . 0
. 0 . . 0
. . 0 . .
. . 0 . .
9 seconds left to play...
```

8. **Gameplay Period Begins:**
   -Print the second grid with a loop
   -Decrement $t9 second counter and print seconds left
   -Set $s0 index to 0 for next step of Period
   -If $t9 counter is 0, go to the end
   -Continue with gameplay loop

```
10 seconds left to play...
. . . O .
. . O . O
. O . . O
. . O . .
. . O . .
9 seconds left to play...
O O O O O
O O O O O
O O O O O
O O O O O
O O O O O
8 seconds left to play...
O O . . .
O . . . .
. . . . .
O . . . .
O . . . O
7 seconds left to play...
O O O O O
O O O O O
O O O O O
O O O O O
O O O O O
6 seconds left to play...
. . . O O
. . O O O
. O O O O
. . O O .
. . O . .
5 seconds left to play...
O O O O O
O O O O O
O O O O O
O O O O O
O O O O O
```

```
5 seconds left to play...
O O O O O
O O O O O
O O O O O
O O O O O
O O O O O
4 seconds left to play...
O O . . .
O . . . .
. . . . .
O . . . .
O . . . O
3 seconds left to play...
O O O O O
O O O O O
O O O O O
O O O O O
O O O O O
2 seconds left to play...
. . . O O
. . O O O
. O O O O
. . O O .
. . O . .
1 seconds left to play...
O O O O O
O O O O O
O O O O O
O O O O O
O O O O O
0 seconds left to play...
```

8.1 **Playing the Game:**
- Within the gameplay loop, the program checks each cell of the grid to identify bombs (`'O'`).
- When a bomb is found, it is detonated, involving the update of adjacent cells by replacing `'O'` with `'.'`.
-To do that,we look for second grid
-If firstGrid[$s0] has 'O',this means secondGrid[$s0] and adjacents of that cell will be detonated (Reason why two grids are used)
-If $s0 == grid size then jump out of loop

8.2 **Printing the New Board:**
- After gameplay loop, the program prints the updated state of the grid which is stored at secondGrid.
-Decrement $t9, set $s0 index to 0, jump to end program if $t9 is 0

8.3 **Refill grids for next gameplay period**
-At this point, we copy secondGrid to firstGrid,then fill secondGrid with all 'O'
-jump back to gameplay period,so repeat next steps

12. **End of Program:**
- When $t9 second counter gets 0 at any step of the gameplay period,program jumps to the end