Computer Systems and -architecture

MIPS: Putting your Project Together

1 Ba INF 2017-2018

Stephen Pauwels
stephen.pauwels@uantwerpen.be

Time Schedule

Exercises are made individually. Put all your files in a tgz archive, as explained on the course's website, and submit your solution to the exercises on Blackboard.

- Deadline: **December 17, 23u55**

Assignment

1. **Reading and building the maze**

   (a) Write a function (*with stackframe!*) that creates a maze based on a text file. Every character in the text file represents an element in the maze as follows:

Input Character	Element	Color
w	wall	blue
p	passage	black
s	player position	yellow
u	exit location	green
newline	end of a row	-
e	enemy	red
c	candy	white

   Every cell in the maze will be represented by a single pixel in our bitmap (make sure you set your settings correct in order to be able to see your maze correctly).

   We will not keep a special, extra representation of the maze in memory other than the one we use for the graphical bitmap representation. We only keep track of the logical position (row, column) of the player using registers (*hint: while reading the file and building the maze, keep track of the current location that you are creating*). The logical positions for a 4x4 maze look like:

$$\begin{array}{cccc} (0,0) & (0,1) & (0,2) & (0,3) \\ (1,0) & (1,1) & (1,2) & (1,3) \\ (2,0) & (2,1) & (2,2) & (2,3) \\ (3,0) & (3,1) & (3,2) & (3,3) \end{array}$$

2. **Update player's position**

   (a) Write a function (*with stackframe!*) that has four arguments (current player row, current player column, new player row, new player column) that updates the position of the player in the displayed maze and returns its (possibly new) current position using registers `$v0` and `$v1`. Make sure the update is only executed when the given move is a valid move within the maze. In order to calculate the memory address for a particular location use the function (*with stackframe!*) from the previous assignment to convert logical coordinates to memory addresses.

3. **Main game loop**

   (a) The main game loop is an endless loop that waits for user input. When input is received it performs the appropriate action depending on this input. In order to avoid that this loop takes all the system resources we add a small sleeptime of 60ms to ensure a smooth execution of our program.

   (b) The input is handled in the following way:

Input (Azerty)	Action
z	move player up
s	move player down
q	move player left
d	move player right
x	exits the game

   Use the update function for the player to update it's position in the maze.

   (c) When the player reaches to exit the game ends and a victory message is printed in the Run I/O.

4. **Improve your code**

   (a) Make sure that you add a sufficient amount of comments to your code (don't forget that you will have to extend your current program for the next assignment).

   (b) Make sure you make use of the stackframe in the correct way.

   (c) **Bonus:** reduce duplicate code by using more helper functions