Columns (MIPS Assembly) - Assignment Specification Summary

This is an original, employer-friendly summary of the CSC258 "Columns" assembly project specification. It is intended for repository documentation and quick onboarding (not a verbatim copy of the course handout).

1. Project Overview

You implement a playable version of the retro match-three puzzle game **Columns** in **MIPS assembly**, running inside a simulator such as **Saturn** or **MARS**. The game renders to a bitmap display and reads keyboard input using memory-mapped I/O (MMIO).

Gameplay concept: the player controls a vertical column of three colored gems. The column stays vertical, but the player can shuffle the order of the three gems. When three or more gems of the same color line up horizontally, vertically, or diagonally, they clear; unsupported gems fall and can create chain reactions.

2. Required Controls

The base control scheme uses the following keys:

Key	Action
a	Move current column left (blocked by walls or occupied cells).
d	Move current column right (blocked by walls or occupied cells).
w	Shuffle the order of the 3 gems within the column (cycle).
s	Move the column downward faster (one row at a time or a full drop - your choice).

3. Simulator Setup (Saturn and MARS)

Saturn: open your .asm file, open the Bitmap tab (Ctrl+T / Cmd+T), configure the bitmap display and base address, run the program, then click the bitmap window so it captures keystrokes.

MARS: open Tools -> Bitmap Display and Tools -> Keyboard and Display MMIO Simulator, configure and connect both to MIPS, assemble and run, then type in the keyboard MMIO window.

Bitmap base address: set the bitmap display base address to **0x10008000** so writes to that memory region appear on screen.

Keyboard MMIO (polling): the simulator exposes keyboard state via memory. A common pattern is: read the status word at **0xffff0000**; if it is 1, read the ASCII key value from **0xffff0004** and handle it.

4. Core Technical Expectations

Your program should be structured around a central loop that repeatedly checks for input, checks collisions, updates game state, redraws the screen, and sleeps briefly to control speed (typical guidance is up to 60 updates per second).

5. Milestones and Grading Structure

Milestone	Minimum expected capability (summary)
1 - Draw the scene	Render the playfield grid/border and draw an initial 3-gem column (random colors).
2 - Movement and controls	Respond to W/A/S/D; repaint the screen regularly; provide a quit key (e.g., q).
3 - Collision detection	Block illegal left/right moves; lock the column on landing; spawn a new column; clear any match-3+ lines and drop unsupported gems; end game when pieces reach the top.
4 - Game features	Add additional features beyond the basics (a mix of easier/harder options).
5 - More game features	Add more advanced features (larger feature set and/or harder features).

6. Deliverables for Submission

A typical submission includes:

Required file	What it contains
Columns.asm	Your MIPS source code for the game.
project_report.pdf (+ source .tex)	Your written report describing milestones completed, design decisions, diagrams, and how to play/run.

7. Notes for Including the Official Spec in a Public Repo

If you plan to publish your repository publicly, avoid uploading the official course handout unless you have permission. Instead, keep this summary and provide brief context about the assignment constraints. If you need the original spec, store it privately or link to an instructor-approved location.