CEG3136A – Lab1
Mixed C & ASM Project

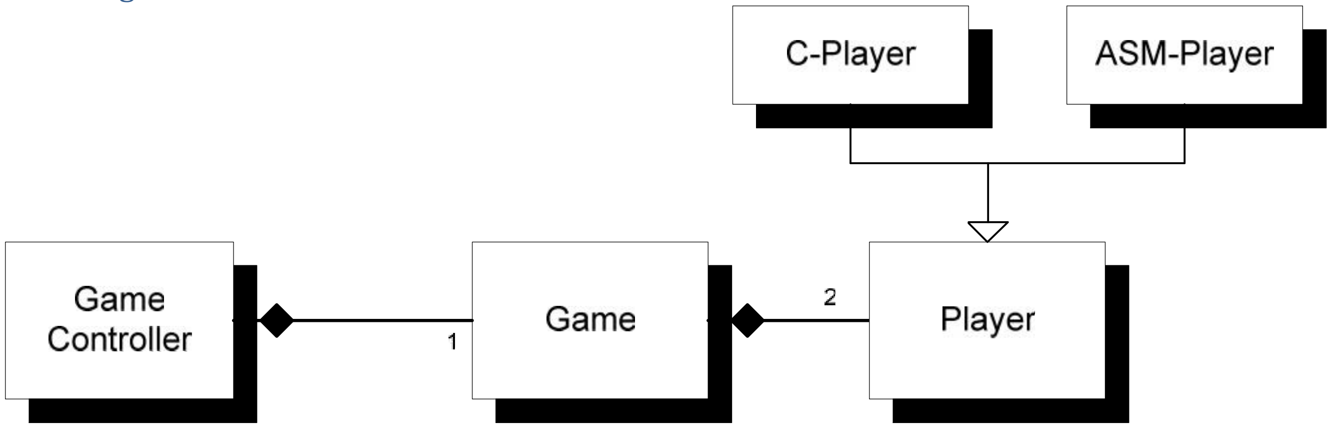
Objectives

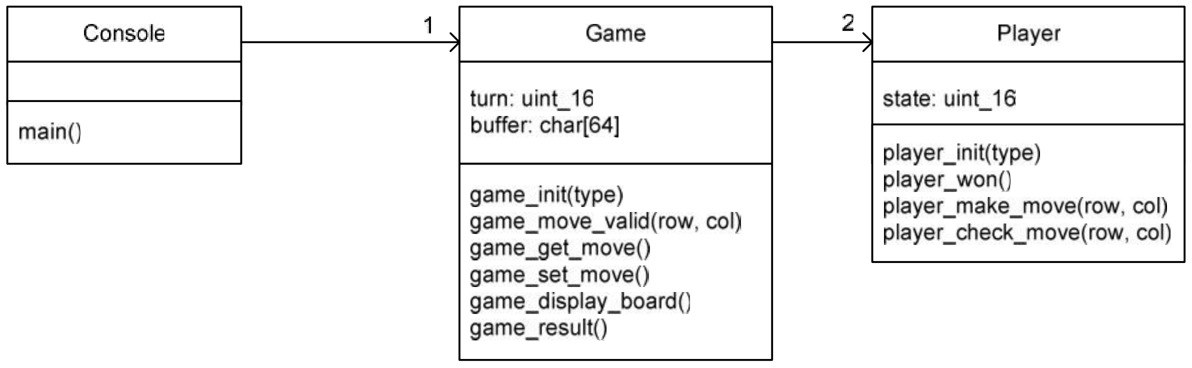
To demonstrate using Object Oriented Programming (OOP) concepts using c, even though it is a structural programming language. Also implement your first assembly language (ASM) functions for the Arm® Cortex®-M4 with FPU core.

Introduction

We implemented an OO design a Tic-Tac-Toe game. The game is a two player game (O & X) that is played on a 3x3 mesh board. Players alternate turns where they place their tokens (either O or X) on the board in attempt to make a straight line (either row, column, or even diagonal). The other player tries to make a straight line while also trying to defy the opponent plans by putting tokens in positions that break the opponent's lines. The game ends when either player successfully connects a straight line (3 tokens) or all 9 squares are filled, in which case it is a draw.

OO Design





The game console (control.c) starts the game application and creates the game board (game.c), which has 2 players of type c-player. The game also creates 2 players of type asm-player, which you are going to help developing it. Both types of players are identical and are used to check one-another's outcome.
The game class supports the following functions:

- game_init: initialize game
- game_move_valid: validates player's move by calling player_check_move() on each palyer to check the move is not already taken by any player. *Note that the game is calling the same function on both c & asm implementations and checks asm against c*.
- game_get_move: interacts with the user to get a valid move to play
- game_set_move: implement the user's move. It calls player_make_move() of the player which has the turn. *Note that the game is calling the same function on both c & asm implementations and checks asm against c*.
- game_display_board: displays the current board state for users after every move
- game_result(): check the game result after each turn. Result is either O-wins, X-wins, Draw, or not yet decided, i.e. continue playing. It calls player_won() on each player to check the winner. *Note that the game is calling the same function on both c & asm implementations and checks asm against c*.

Player State

The state variable indicate the player's token positions on the board as shown below:

	3	2	1	0
b[3..0]	0			
b[7..4]	0			
b[11..8]	0			
b[16..12]	0	0	0	0

For example a player state of 0x0214 indicates it occupied the following positions on the board:

	3	2	1	0
b[3..0]	0	1		
b[7..4]	0			1
b[11..8]	0		1	
b[16..12]	0	0	0	0

A player wins if his/her state includes one of the following positions {0x0007, 0x0070, 0x0700, 0x0111, 0x0222, 0x0444, 0x421, or 0x124}

Your Task

Your task is to develop the assembly version of the following functions:

1. player_make_move_s
2. player_check_move_s
3. player_won_s

The functions are stubbed in the file player.s
Make sure your development outcome matches the c implementation, an error message will be logged if it didn't match.

Submission

1. Run the simulation for at least 3 scenarios as follows, for each scenario copy/paste the log from the Debug (printf) Viewer into a text file. Name your files grp_NN_R.txt, where NN is your group number, R is the result, one of {O, X, D}.
    a. Player O wins
    b. Player X wins
    c. Draw
2. Clean your project, Go to Project > Clean Target
3. Save your log files at the same directory as your project files
4. Zip the project directory
5. Submit a single zipped file to Brighspace.