# COS10004 Computer Systems

## Assignment 2 Overview

This assignment requires knowledge of ARM assembly programming with the ARMlite simulator and assumes you have covered all modules up to and including Week 10.

**Purpose:** Implement a full end extended version of the game Matchsticks (based on Lab 8) in ARM assembly for the ARMlite simulator

**Task:** Follow the defined sequence of stages to implement the functionality for the game Matchsticks. Each stage should be thoroughly tested and is expected to work as described in each stage when loaded into the ARMlite simulator.

**Due**: 11:59pm Friday, November 3rd

**Assessment**: The assignment is worth 15% of your assessment for this unit.

**Submission Instructions:**
Your completed submission must be made through Canvas - (Go to Assignment 2 under "Assignments" before the due date/time. Everyday day late will incur a 10% deduction.

Each submission should be a ***zip file containing***:

- All .ASM files for each stage completed
- A file or provided link (that is accessible to markers either by providing an exclusive link, or a password for us to enter, <u>but not made publicly available</u>!) to an online video strictly **5-minutes in length** containing:
    - A well organised and succinct demonstration of your solution (making clear the stage you have completed up to!), and any unresolved problems with your program for that stage (~1 min of video)
    - Explanations of key design decisions and implementation choices (~2 mins of video)
    - A critical reflection on your design and implementation (e.g. what about your design and implementation could be improved ? Why ?) (~2 mins of video)
    - *Videos over 5 minutes will be penalised severely*

**Resources**:

- All lectures and slides from Week 7-Week 10 on Canvas
- Lab 8 !!!! If you have already done Lab 8 then you will be well on your way !
- ARMlite programming reference manual:
- Richard Pawson's "Computer Science from the Metal Up"
- ASCII character code lookup table
- Tutor consultation times

**Academic Integrity**

*<u>This assignment is not a group assignment</u>*. While high level discussions with your peers and with your tutor are fine and expected, the work submitted must be your own, and will be subject to similarity testing. You should not share your solution with anyone, or make available online. Any encountered cases of plagiarism will immediately result in a zero grade being awarded for the assignment for all parties involved, and probable further disciplinary action taken. If you are

struggling with the assignment, you should make use of the numerous hours of tutor consultation available to you (as  advertised on Canvas and above under resources).

# Matchsticks

## The game

The game starts with a specified pile of matchsticks. Two players take it in turns to remove either 1, 2, or 3 matchsticks from the remaining pile. A player wins the game by forcing the opponent to take the last matchstick. Your full implementation will pit a single player against the computer. Your full implementation will also incorporate some simple graphics output.

## The Assignment

You will be implementing the game Matchsticks as described above, following the prescribed stages below. Yes - this is the same game described in Lab 8 - however here you will have the opportunity to extend on it further and demonstrate your mastery of ARM assembly programming and conventions (some of which are covered in later weeks of this unit). Note also that the requirements for this assignment are different to Lab 8, as is the expectation of the quality of the design and implementation you provide. Of particular importance for those seeking the highest marks will be that your implementation incorporates the use of functions (covered in Week 10's module). Your correct use of functions and compliance with relevant conventions such as the Application Binary Interface (again – see Week 10 content), will be part of the assessment.

The assignment is divided into 4 stages, guiding you through the different functionality that is needed. You should work through each stage in sequence and save your solution for each stage in an ASM file named after that stage. You will submit all files at the end, along with a 5-minute video as per the submission instructions on the cover sheet.

## The Assessment

The assignment will be assessed according to the following criteria:

- Successful completion of each stage according to the specified requirements (60% of total mark)
  - Completion of Stage 1: (up to 10 of the 60 marks available)
  - Completion of Stage 2: (up to 20 of the 60 marks available)
  - Completion of Stage 3: (up to 40 of the 60 marks available)
  - Completion of Stage 4: (up to 60 of the 60 marks available)
- Quality of the solution (25% of total mark)
  - Innovation/elegance of solution (and explained in video)
  - Clarity and readability of solution
  - Quality of assembly code implementation (e.g., appropriate use of functions, arrays, and general use of ARM assembly constructs)
- 5-minute video demonstration and reflection (15% of total mark)
  - Quality and clarity of the video
  - depth of understanding of implementation and critical reflection on the design and implementation

Note that your video should be well planned and structured. It should make succinct, well thought out points about your design as per the instructions on the cover sheet of this handout.

To get full credit for each stage of completion, it must work as described and will be tested in the ARMlite simulator by one of the teaching team.

*Stage 1 over page*

## Stage 1 – Game Setup

Before the game can start, some setup information needs to be entered.

To do all this, write an ARM assembly program that:

- asks for the name of the player by prompting the user with:
  *"Please enter your name*".
  Any names entered should be stored in character arrays for future use.

- asks for the starting number of matchsticks with the prompt:

  *"How many matchsticks (5-100)?"*

  which can be any number between 5 and 100. If the number entered is outside of this range, or simply invalid, then the question should be asked again until a valid number is entered.

- Once all the above is read in, the program should print the following message, each on separate lines:

  `Player 1 is` *<insert player 1 name>*

  `Matchsticks:` *<insert number of matchsticks>*

You should replace the above "<…>" parts with the actual data entered.

***Save your solution using the filename stage1.asm***

*Stage 2 over page*

## Stage 2 – Single Player Input

Extending on your implementation from Stage 1, write an ARMlite compatible assembly program that fulfills the following:

- Prints to screen:
  *"Player <name>, there are <X> matchsticks remaining*".
  where <name> is the name of the player, and X is the remaining number of matchsticks.

- Prompts the player to enter a number of matchsticks to remove with the output:
  *"Player <name>, how many do you want to remove (1-3)?"*

- Reads in the integer value entered, and ensures the number entered is valid. The number must be between 1 and 3, and not be larger than the current number of matchsticks remaining. If these conditions are not met, then your program should repeatedly ask until the number entered is valid.
- Updates the number of matchsticks remaining based on the number entered.
- Repeats the above until all matchsticks are gone.
- Once all matchsticks are gone, prints *"Game Over*" and terminates.

***Save your full solution to a file called stage2.asm.***

*Stage 3 over page*

## Stage 3 – Implement Human versus Computer

You have now implemented a program that allows a player to keep removing matchsticks until none are left. Your task now is to implement a computer player to play against. Specifically, modify your program from Stage 2 to ensure it satisfies the following:

- Alternates turns between the human player and the computer player. The human player should always go first.
- When it is the human player's turn, the same inputs and outputs should be implemented as per Stage 2.
    - When it is the computer player's turn, the screen output should read: "Computer Player's turn".
- The computer player's turn should utilise ARMlite's random number generator in order to select a number between 1 and 3 to remove from the matchsticks count. The number selected should also not exceed the number of matchsticks remaining. See Lab 8 for information on random number generation in ARMlite.
- Once a valid random number is obtained, this should be used to update the total number of matchsticks remaining.
- Once either player's turn (human or computer) is complete, the total number of matchsticks should be inspected.
    - If it is 1, then the player who's turn just occurred is the winner (i.e., they have forced the other player to pick up a single remaining matchstick).
    - If the total number of remaining matches is > 1, then play continues.
    - If it is 0, then the game is a draw.
    - Based on the above conditions, one of the following three messages should be displayed as appropriate:
      "Player <name>, YOU WIN!",
      "Player <name>, YOU LOSE!", or
      "It's a draw!"
- Upon game completion, your program should ask if the player wants to play again with the output:
  "Play again (y/n) ?"
    - If the input is "y", your program should start the game again (with the same settings as before)
    - If the input is "n", your program should terminate.
    - If the input is anything else, your program should ask again.

***Save your solution using the filename stage3.asm***

*Stage 4 over page*

## Stage 4 – Graphics

You now have a working matchsticks program, but so far only text-based output. Modify your program so that, upon commencement of the game, the number of matchsticks remaining is represented graphically. That is, you should utilise ARMlite's graphics display to show each matchstick remaining. How you choose to draw a matchstick is up to you (it may be as simple as a line, but we encourage creativity here!), however the number of matches drawn must be correct after each turn and should be easy to interpret by the player. You should also ensure you have sufficient space to show up to 100 matches as per the option. More creative and sophisticated displays may be eligible for modest bonus credit.

***Save your solution using the filename stage4.asm***

*This completes the assignment.*