THIS IS A PROCTORED PRACTICAL

YOU MUST SHARE YOUR SCREEN SO YOUR PARTICIPATION IN THIS PRACTICAL CAN FULLY INVIGILATED

1. Create a Github repository "Assembly\_and\_C"
2. Create a sub directory PRACTICAL\_##
3. Add Github link to CA Spreadsheet   
   e.g <https://STUDENTID.github.com/Assembly_and_c/PRACTICAL_##>
4. Invite Lab Supervisors including **MuddyGames** as a collaborators
5. Go to designated group to complete practical
6. Upload completed Practical files to Github repository

NOTE: Use of EASy68K editor and emulator allowed, use of internet allowed, use of slide deck(s) allowed. Installer located here <http://www.easy68k.com/>

Create a unique file ***e.g. practical\_##\_part#.X68*** for each practical section below.

**Objective** Understand and utilise Conditional Branches and Control Structures**:**

|  |  |  |
| --- | --- | --- |
| **1**  **Cmp – CoMPare**  **BNE – Branch NotEqual**  **BEQ – Branch Equal**  **GAME\_LOOP executes** | Create a new 68K project and name the file ***practical\_05\_part1.X68***  Edit compile and execute the code across and observe while debugging and contents of data registers D0 and D1.  Examine and note contents of status registers and.  Review questions, what is the purpose of CMP, BNE, BEQ and the Status Register. How many times does GAME\_LOOP execute? |  |
|  |
| [**Source Code Image (click here)**](https://1drv.ms/i/s!Au3XD_Li32Zeno9yFzqST1UPnNB1jg?e=3Yp2hO) |
| **2**  **Immediate addressing**  **Move.b #50, D1**  **Direct**  **Cmp D2,D1** | Create a new 68K project and name the file ***practical\_05\_part2.X68***  Edit compile and execute the code across and observe while debugging and contents of memory, data registers and address registers.  Modify the code such that the COLLISION Branch in executed when the player and enemy are at the same X and Y coordinates 55 and 55  Load Player and Enemy X and Y Positions from the Arrays PLAYER\_POSITION and ENEMY\_POSITION  Review questions, what do the BEQ instructions mean, what Addressing Modes are used? |  |
|  |
| [**Source Code Image (click here)**](https://1drv.ms/i/s!Au3XD_Li32Zeno9rT1zk1_4V4veVnw?e=PMIK68) |
| **3**  **EQU -Equate**  **LEA – Load Effective Address**  **Trap #15 looks at D0 value and completes the function asigned to that value**  **Memory is used to store PLAYER\_POS.**  **ENEMY\_POS.**  **PLAYER\_HEALTH**  **MSG\_GAME\_LOOP**  **MSG\_GAME\_OVER**  **And the value of DAMAGE**  **Immediate**  **MOVE.B #13, D0**  **Direct**  **CMP D2, D1** | Create a new 68K project and name the file ***practical\_05\_part3.X68***  Edit compile and execute the code across and observe while debugging and contents of memory, data registers and address registers.  Review questions, what are EQU, BEQ, LEA, TRAP #15, D0 used for within this code. How are MEMORY locations used. Register instructions mean, what Addressing Modes are used? |  |
|  |
| [**Source Code Image (click here)**](https://1drv.ms/i/s!Au3XD_Li32Zeno9qnx5rE_TmNYOFkQ?e=kX6Bia) |
| **4**  **BLE**  **Branch if Less or Equal**  **Immediate**  **MOVE.B #13, D0**  **Direct**  **MOVE.B D0, D4**  **Indirect**  **MOVE.L (A2), D3** | Create a new 68K project and name the file ***practical\_05\_part4.X68***  Edit compile and execute the code across and observe while debugging and contents of memory.  Examine and note contents of address registers and memory.  Review questions, what are the Branch Instructions useful for BLE and BEQ, what Addressing Modes are used?  See FLOW DIAGRAM |  |
|  |
| [**Source Code Image (click here)**](https://1drv.ms/i/s!Au3XD_Li32Zeno9mHthuAR_UtOQh4w?e=BJ8Q2c) |
| **5** | Create a new 68K project and designate the file as ***practical\_05\_part5.X68***.    Review questions, what is the instruction TST.L useful for and what Addressing Modes are used? | **4 Specification**: Using Part 4 modify the code as follows;   * Add a **POWER\_UPS** see Fig 5.1.1 Array These are locations where when player is at this position, they receive a Health **POWER\_UPS** * Modify the Code so that the player receives PO when these positions are reached * Modify the code, to improve maintainability by using the VARIABLE Memory Locations (as in Part 4)  PLAYER\_X PLAYER\_HEALTH  e.g.   **From** ADD.L #1, (A2)  **TO** ADD.L #1, PLAYER\_X |
| POWER\_UPS DC.L 10, 55, 220, 0 ; Terminate Array with a 0  ***Figure 5.1.1*** | | |
| **6** | Complete Practical Quiz which will be provided by Lab Supervisor | |

**Demonstrate completed assembly files at the end of the LAB and ensure it has been checked**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name** |  | **Student Number** |  |
| **Date** |  | **Checked** |  |

**FLOW DIAGRAM**

