Practical 06

Assembly Language

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.

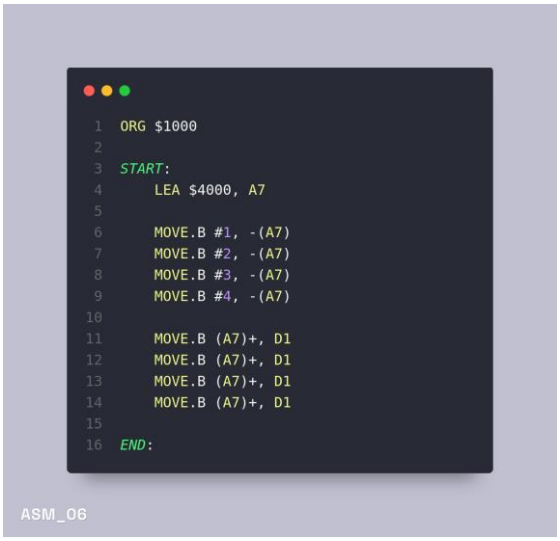
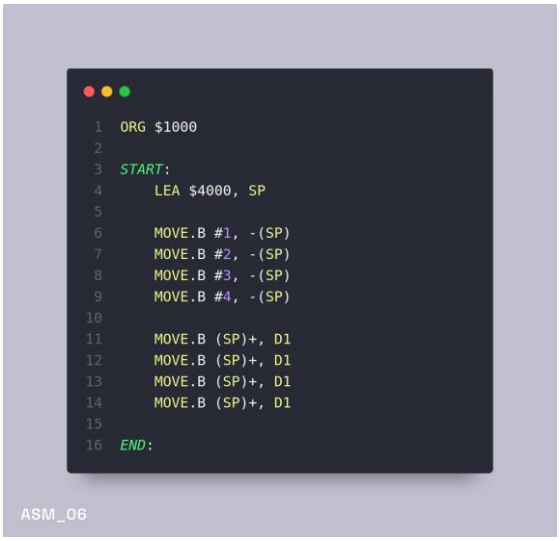
Objective Understand and utilise Conditional Branches and Control Structures**:**

<table>
<tr>
<td>1</td>
<td>Create a new 68K project and name the file practical_06_part1.X68

Edit compile and execute the code across and observe while debugging and contents of Data and Address Registers.</td>
<td> 

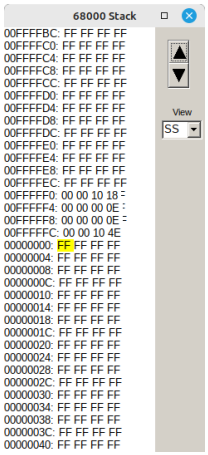
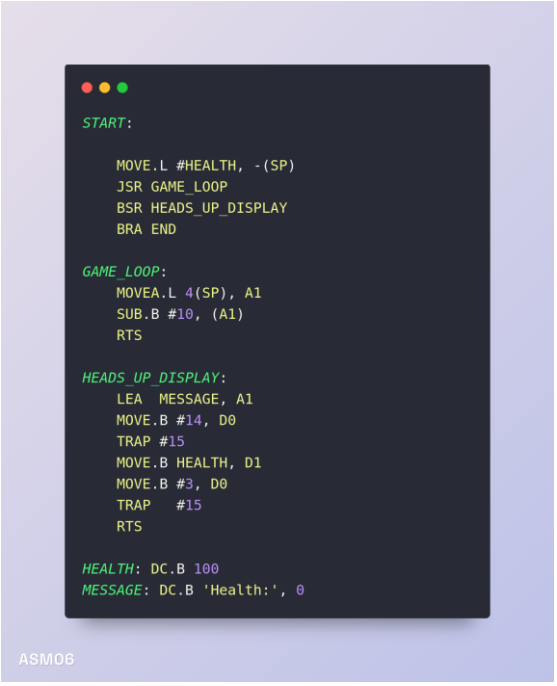
Source Code Image (click here)</td>
</tr>
<tr>
<td>2</td>
<td>Create a new 68K project and name the file practical_06_part2.X68

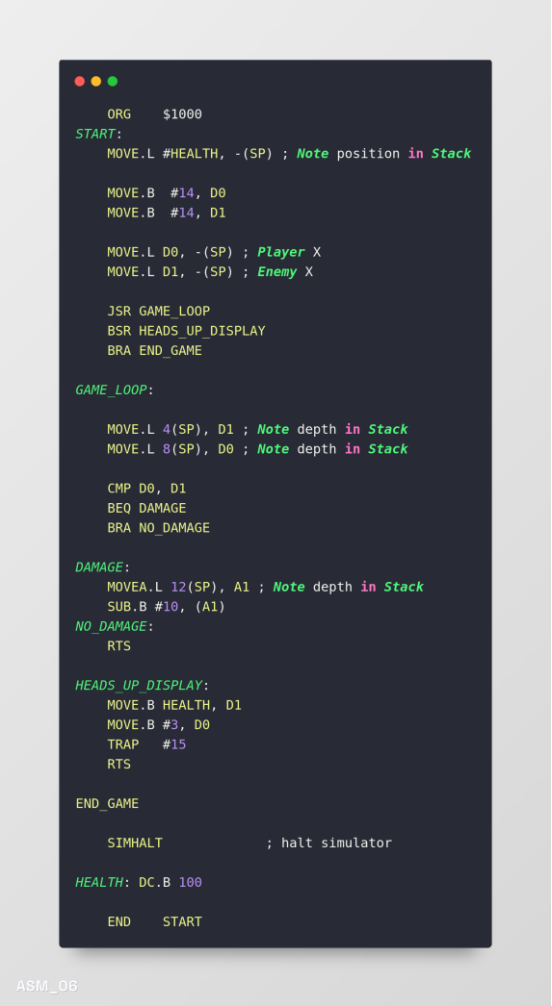
Complete code for Trap Tasks as listed here.</td>
<td>
<table>
<tr><td>0</td><td>Display string at (A1), D1.W bytes long (max 255) with carriage return and line feed (CR, LF). (see task 13)</td></tr>
<tr><td>1</td><td>Display string at (A1), D1.W bytes long (max 255) without CR, LF. (see task 14)</td></tr>
<tr><td>2</td><td>Read string from keyboard and store at (A1), NULL terminated, length retuned in D1.W (max 80)</td></tr>
<tr><td>3</td><td>Display signed number in D1.L in decimal in smallest field. (see task 15 & 20)</td></tr>
<tr><td>4</td><td>Read a number from the keyboard into D1.L.</td></tr>
<tr><td>5</td><td>Read single character from the keyboard into D1.B.</td></tr>
<tr><td>6</td><td>Display single character in D1.B.</td></tr>
<tr><td>7</td><td>Set D1.B to 1 if keyboard input is pending, otherwise set to 0.
Use code 5 to read pending key.</td></tr>
<tr><td>8</td><td>Return time in hundredths of a second since midnight in D1.L.</td></tr>
<tr><td>9</td><td>Terminate the program. (Halts the simulator)</td></tr>
<tr><td>11</td><td>The low byte holds the ROW number (0-31).
0,0 is top left 79,31 is the bottom right.
Out of range coordinates are ignored.
Clear Screen : Set D1.W to $FF00.</td></tr>
<tr><td>12</td><td>Keyboard Echo.
D1.B = 0 to turn off keyboard echo.
D1.B = non zero to enable it (default).
Echo is restored on 'Reset' or when a new file is loaded.</td></tr>
<tr><td>13</td><td>Display the NULL terminated string at (A1) with CR, LF.</td></tr>
<tr><td>14</td><td>Display the NULL terminated string at (A1) without CR, LF.</td></tr>
<tr><td>16</td><td>Adjust display properties
D1.B = 0 to turn off the display of the input prompt.
D1.B = 1 to turn on the display of the input prompt. (default)
D1.B = 2 do not display a line feed when Enter pressed during Trap task #2 input
D1.B = 3 display a line feed when Enter key pressed during Trap task #2 input (default)
Other values of D1 reserved for future use.
Input prompt display is enabled by default and by 'Reset' or when a new file is loaded.</td></tr>
<tr><td>17</td><td>Combination of Trap codes 14 & 3.
Display the NULL terminated string at (A1) without CR, LF then
Display the decimal number in D1.L.</td></tr>
<tr><td>18</td><td>Combination of Trap codes 14 & 4.
Display the NULL terminated string at (A1) without CR, LF then
Read a number from the keyboard into D1.L.</td></tr>
</table>
TrapCodes +
Trap Codes (click here)</td>
</tr>
</table>

3	Create a new 68K project and name the file ***practical_06_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 is the purpose of Address Register A7?	 **Source Code Image (click here)**
4	Create a new 68K project and name the file ***practical_06_part4.X68*** Edit compile and execute the code across and observe while debugging and contents of memory, data registers and address registers. Review questions, what is the purpose of Address Register SP (Stack Pointer)?	 **Source Code Image (click here)**

Practical 06

Assembly Language

5	Create a new 68K project and designate the file as ***practical_06_part5.X68***. Review questions parameters to subroutines can be passed through the stack what other functions can be achieved through the stack, what observations have you made when opening VIEW | STACK ? 	 [Source Code Image (click here)]

6	Create a new 68K project and designate the file as ***practical_06_part6.X68***. Review questions moving around stack can be achieved by what means other than push and pop, what problems could this cause?	 ASM_06 [Source Code Image (click here)]
7	Complete Practical Quiz which will be provided by Lab Supervisor	

```
    ORG    $1000
START:
    MOVE.L #HEALTH, -(SP) ; Note position in Stack

    MOVE.B  #14, D0
    MOVE.B  #14, D1

    MOVE.L D0, -(SP) ; Player X
    MOVE.L D1, -(SP) ; Enemy X

    JSR GAME_LOOP
    BSR HEADS_UP_DISPLAY
    BRA END_GAME

GAME_LOOP:

    MOVE.L 4(SP), D1 ; Note depth in Stack
    MOVE.L 8(SP), D0 ; Note depth in Stack

    CMP D0, D1
    BEQ DAMAGE
    BRA NO_DAMAGE

DAMAGE:
    MOVEA.L 12(SP), A1 ; Note depth in Stack
    SUB.B #10, (A1)
NO_DAMAGE:
    RTS

HEADS_UP_DISPLAY:
    MOVE.B HEALTH, D1
    MOVE.B #3, D0
    TRAP   #15
    RTS

END_GAME

    SIMHALT             ; halt simulator

HEALTH: DC.B 100

    END    START
```

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

Student Name		**Student Number**	
Date		**Checked**	