PEEK AND POKE

ASSIGNMENT-04

BASATI SIVAKRISHNA

22976 (MTech EPD)

Code Explanation:

From the previous assignment, we are adding two more features

1. Peek mem_loc
   This command should be useful to search the given location from the memory and show the data in Serial monitor. Mem_loc should be hex value.
2. Poke mem_loc "_data_"
   This command should replace the data from the mem_loc to size of data+mem_loc with "_data_". If the mem_loc+data size exceeds the limit of aur accessible data then it should not replace and should show the Memory overflow.

Creating Symbol in Linker script:

To achieve this, we should create a symbol in linker script to get a memory allocation of dynamic number of bytes. Here iam declaring the section in SRAM.

```
1
2     .version : ALIGN (4) {
3         __version_load__ = LOADADDR (.version);
4         __version_start__ = .;
5         *(.version)
6         *(.version*)
7         . = ALIGN (4);
8         __version_end__ = .;
9     } > REGION_DATA AT> REGION_TEXT
0
```

Once we declare a variable with this attribute then there will be memory allocation according to the size of the variable in .map file

```
64 char version_str[15]__attribute__ ((section (".version")));
```

Respective .map file for the above declarations in linker script and main.c file(16 bytes has allotted in this case).

```
987
988 .version        0x20000a6c       0x10 load address 0x00004ac0
989                 0x00004ac0                __version_load__ = LOADADDR (.version)
990                 0x20000a6c                __version_start__ = .
991 *(.version)
992 .version        0x20000a6c        0xf ./main.o
993                 0x20000a6c                version_str
994 *(.version*)
995                 0x20000a7c                . = ALIGN (0x4)
996 *fill*          0x20000a7b        0x1
997                 0x20000a7c                __version_end__ = .
```

LCD initialization and Printing Data on LCD screen:

To initialize LCD, we need to send series of commands and data bytes via data pins of LCD. To switch from command mode to data mode we should pull the RS pin accordingly.

```
void LCD_Initialization();

void writeStringLCD(char s[], int line);
```

These two functions will be useful to initialize and show the data on screen.
The line argument in writestring() will take 1 or 2 as input and prints the data on first line or 2nd line of the 16*2 LCD.

PEEK AND POKE:

Based on the previous assignment, these commands has assigned with the command_type of 7 and 8 respectively.

```
516 else if(((strcmp(parsed_cmnd1,"peek")==0)))
517     command_type = 7;
518 else if(((strcmp(parsed_cmnd1,"poke")==0)))
519     command_type = 8;
```

1. peek mem_loc
   Given mem_loc(in ascii format) will be converted to hex value and typecasted as char pointer. Data from that location is accessed using dereferencing operator and respective byte will be printed on the serial monitor.

Output from the serial monitor for valid peek command:

```
COM13 ×
Initialization done
Entered command is: start
||-------------STATUS OF THE MACHINE--------------||
COMMAND GIVEN ----> start
LED CODE ---->  0
STATE ---->   RESUME
 DATA IN MEMORY LOCS: 0x20000a6c TO 0x20000a78
0x20000a6c: V
0x20000a6d: e
0x20000a6e: r
0x20000a6f: s
0x20000a70: i
0x20000a71: o
0x20000a72: n
0x20000a73:
0x20000a74: 0
0x20000a75: .
0x20000a76: 0
0x20000a77: 1
Entered command is: peek0x20000a6e
 DATA IN MEMORY LOC(0x20000a6e): r
```

For invalid address:
If the entered location after peek is not in our defined range the serial monitor should print the accessible range.

```
Entered command is: peek0x200006ac
 ******Invalid address*******
 address range: 0x20000a6c TO 0x20000a78
```

2. Poke mem_loc “_data_”
   Given mem_loc(ascii value) will be converted to the hex value and typecasted to the char pointer. And the _data_ will be parsed and stored in the poke_data variable. And the size _data_ will be stored in the poke_index variable.
   The parsed _data_ will be written from the mem_loc to mem_loc+ size of data.
   If the mem_loc _ size of data exceeds the memory that we a lot for the .version section than serial monitor will show the “DATA OVERFLOW” error.
   Note: Here we are not giving the size of data in command, it is getting calculated in the program itself.

   Poke command(valid):

```
 DATA IN MEMORY LOCS: 0x20000a6c TO 0x20000a78
0x20000a6c: V
0x20000a6d: e
0x20000a6e: r
0x20000a6f: s
0x20000a70: i
0x20000a71: o
0x20000a72: n
0x20000a73:
0x20000a74: 0
0x20000a75: .
0x20000a76: 0
0x20000a77: 1
Entered command is: poke0x20000a6dhello
||-------------STATUS OF THE MACHINE--------------||
COMMAND GIVEN ----> poke0x20000a6dhello
LED CODE ---->  0
STATE ---->   RESUME
 DATA IN MEMORY LOCS: 0x20000a6c TO 0x20000a78
0x20000a6c: V
0x20000a6d: h
0x20000a6e: e
0x20000a6f: l
0x20000a70: l
0x20000a71: o
0x20000a72: n
0x20000a73:
0x20000a74: 0
0x20000a75: .
0x20000a76: 0
0x20000a77: 1
```

Invalid poke command:

```
Entered command is: poke0x20000a75abcd
*******DATA OVERFLOW**********
**********  ENTERED COMMAND IS INVALID  *******
ENTER THE FOLLOWING COMMANDS ONLY
1. blink blink_rate        2. color color_type
3. pause                   4. resume
5. stop                    6. start
7. peek mem_loc            8. poke mem_loc "data"
```

LCD output:
Before poke command

After poke command